Fall 2015

Predicting Autism over Large-Scale Child Dataset

Arpit Arya
San Jose State University

Predicting Autism over Large-Scale Child Dataset

A Writing Project Report

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science

By

Arpit Arya

December 2015

The designated Project Committee Approves the Project Titled

Predicting Autism over Large-Scale Child Dataset

By

Arpit Arya

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSE STATE UNIVERSITY

December 2015

Dr. Thanh D. Tran (Department of Computer Science)

Dr. Thomas Austin (Department of Computer Science)

Ms. Saranya Venkateswaran (IT Security Consultant, Coalfire Systems Inc.)

Abstract

Data Analytics and Machine learning in healthcare are one of the most emerging and needed fields in current time. Also, a lot of research has been performed and is still being done in this field. In healthcare, gone are those days when only doctor examines and patient listens. Now doctor has a lot of technologies which can assist him and help in accurately diagnosing the disease with which his patient is suffering. The backbone of such technologies is data analytics and machine learning where we can make out a lot of inferences from tons of patients' data already available. This project aims at performing research and implementation of big data and machine learning techniques on the data related to the patients suffering from the disease called Autism. Autism is a neural disorder disease characterized by impaired social communication, verbal and non-verbal interaction, restrictive and repetitive behavior [4]. Autism is majorly noticed in children under or about the age of two years. One very important thing to be observed here is that autism is highly heritable and the cause includes both environmental factors and genetic susceptibility. Hence it is very important to have such data which contains details of patients including their symptoms, lab test data, history, vaccination details etc. which gives specific details of patients and their history. The project ultimately aims at training the data model with the set of training data and then testing and evaluating the data model using the test data. In this way, it should be a research and solution for implementing machine learning to detect and diagnose autism.

Acknowledgements

I am greatly indebted to my project advisor Dr. Thanh D. Tran for suggesting me this topic and sparing me his valuable time for discussions in spite of his busy schedule. It would not have been possible for me to complete this project work without his valuable suggestions, critical analysis, constant interest and persistent encouragement. His knowledge, experience, energy and vision had been constant source of inspiration. I sincerely thank him and consider myself extremely fortunate to get a chance to work under his supervision which had been a learning and enjoyable experience.

I sincerely thank my committee members Dr. Thomas Austin and Ms. Saranya Venkateswaran for their valuable time to review my report, important suggestions and timely advice.

I take this opportunity to express deep sense of reverence and gratitude to my parents and family members without whom love, affection, everlasting patience, perpetual motivation and blessings I wouldn’t have reached up to this stage.

Table of Contents

List of Figures

1. Introduction

With the emergence of technology and science in every possible field, there has been a need for automating the processes so as to make them fast and efficient. Health care is one of those industries which are very complex in terms of diagnosis and processes involved with human health issues. In terms of diversity of diseases also, health care industry is quite vast as according to WHO, there are about 30,000 known diseases and out of which there are effective treatments available for only one-third of them i.e. about 10,000 diseases [5]. When it comes to the sensitive subjects like health and lives of people, then it becomes more important to deliver the diagnosis and treatment followed by diagnosis on time. The time taken in diagnosis is one of the other major challenges which need to be taken care of because in some cases, even the slight delay can deteriorate the situation of the patient or make his/her recovery slow.

With all the complexities in the nature of diagnosis and time consuming processes, there is an immense scope for inhibiting more technology solutions in the health care industry. One such important inhibition can be leveraging the big data and machine learning technologies to predict and fasten the complex and time consuming processes of diagnosis and treatment. Such system can be developed which utilizes immense amount of health / medical data available towards predictive modeling and predictive analysis.

In this project, text classification is used as a machine learning technique. In text classification, data is preprocessed so that prediction can be made on the basis of different categories into which text is classified as. There are various classification algorithms which can be utilized. Technically, every classifier is different in its way of data accumulation, data filtering, feature extraction and utilizing these processes towards learning the model.

To learn the data model, supervised learning technique has been used in this project. Supervised learning is a technique in which there are two types of data sets: Training data set and Test data set. Training data set contains the data instances and a class or label is provided to each data instance. Test data set contains the new data instances which are not there in training data set. Additionally, test data set contains class or label because ultimately it is for the purpose of evaluating the prediction results. To start with, training data set is provided to the classifier so that it learns the data in its own way. Then to evaluate, test and predict, the test set is provided to the classifier. It then predicts the class or label of the data instances in the test data set and accuracy of the prediction results is computed.

As this project focuses on predicting Autism, the data sets are related to the patients’ details in the form of text containing the detailed explanation of the symptoms from which they are suffering. The data sets contain details of both, patients with Autism and without Autism so as to train with both kinds of instances. There are various stages in the project such as data pre-processing, data filtering, feature extraction, prediction (evaluation), testing the accuracy.

These stages are described in further sections.

2. Background

This project has been highly motivated from the idea and vision of Dr. Tran as he has been doing research in this field and had a clear and precise goal in his mind. During the course of the project, some research papers were studied so as to research about the strategies that have been applied or research that has been in progress towards detection and diagnosis of Autism.

The authors of research paper [1] focused on the idea of rapid detection of risk of autism. They suggested that the current approaches which are used for diagnosing autism have a high validity of diagnosing the disease but the disadvantage is that it is very time consuming. This can result in high delays in reaching to a decision. They focused on a relatively small set of children with and without autism. The algorithm or method which is used currently to diagnose autism is “gold-standard Autism Diagnostic Observation Schedule-Generic” (ADOS-G). By using machine learning to derive a classifier, they were able to reduce the length by 72% compared to ADOS-G.

Because of the nature of the disease, Autism is primarily diagnosed through behavioral evaluations and to achieve the measure of impairments three core developmental domains have been designed:

1. Language and Communication
2. Reciprocal Social Interactions
3. Restricted and Repetitive behaviors

The instrument used to ADOS-G and now its updated version is ADOS-2. To examine, an exam has been devised which consists of four modules based on the above mentioned domains. It is also devised to cover variety of ages and behaviors. An updated version of exam has been designed for ADOS-2 which is an updated version of instrument. In the updated version of exam, there are two distinct domains:

1. Social affect, and
2. Restricted, Repetitive behaviors

Also in the updated version, there are five modules devised for different ages and behaviors. ADOS-2 has overall higher accuracy than ADOS-G. 10 activities have been designed for ADOS-G and 10 + 4 new activities for ADOS-2. On the basis of these activities, ADOS uses a new scoring algorithm. It generates a comparison score in the form of metric. All the domains or factors are considered while calculating the comparison score which ranges on the scale of 1 to 10 (10 corresponds to the most severe).

The downside of ADOS exams is their length or the complexity because of which they take a lot of time. They also require clinical facility administered by the trained professionals. These factors cause a lot of delays in diagnosis process. Also because of such lengthy and complex procedure, the diagnosis cannot be provided to all the population which needs treatment. Hence this results in unequal and inconsistent distribution or coverage. The clinical facilities and trained professionals tend to be available more in major cities. They are overall quite less than the population which needs treatment. Due to lack of resources and time constraints, initial diagnostic screenings do not get conducted consistently. It can be so severe that families might have to wait as long as 13 months from initial screening to clinical diagnosis. There is estimation that 27% of the cases remain undiagnosed until the age of 8 years. In the US, the average age of autism diagnosis is above 4 years [1].

If the diagnosis is delayed, it is quite obvious that the treatment therapies will also get delayed. The treatment therapies consist of speech delivery and behavioral therapies which are quite significant for improvements if delivered under the right age, earlier in life. If delivered later than the particular age, its impact does not remain as beneficial as before [1].

As the benefit of the therapies is quite significant and enormous, there is a huge need of something immediate, significant and efficient so as to deliver some method which is rapid with high accuracy and not much lengthy. By using Machine Learning approach, machine learning classifier has been built we which can utilize the large scale datasets and perform text classification with high accuracy.

3. Problem Definition

While doing study about the diagnosis of autism, it was clear that there is a need of something fast and accurate so as to deliver the results of diagnosis on time and then the related therapies and treatment can begin efficiently.

To contribute towards the problem, search for the reasonable datasets is a very important step. After searching and finalizing the dataset, text classification was chosen to perform machine learning techniques and steps. Text classification is a kind of problem in which sentences are processed and then classified under the labels or classes. For example, the classes in this project are Positive and Negative. Positive refers that the patient has autism and negative refers that the patient doesn't have autism.

The dataset used for this project is the collection of data from "Vaccine Adverse Event Reporting System" (VAERS) [6]. The raw dataset had been downloaded from their government website [6]. Every data instance had to be provided with certain label or class (Positive/Negative) so as to achieve text classification on the data.

The main attraction towards the dataset was that it contained the description of the symptoms (symptom_text) each patient was suffering from. Also it contained other useful attributes such as lab_data, other_medications, condition_history, prior_vaccination which seemed to be useful in performing text classification towards prediction of autism. The dataset and its processing are explained in the further sections.

4. Data

The raw dataset had been downloaded from the government website of "Vaccine Adverse Event Reporting System" (VAERS) [6]. It contains several attributes and each attribute is a medical detail of the patient. Each instance is a detail of one particular patient. The first attribute is vaers_id which can be considered as unique id given to every patient.

Number of instances in the dataset are 145,000 and the size of the data is approximately 2 GB.

5. Proposed Solution

The main problem in the diagnosis of Autism today is the complexity and the delay which is caused by that. So the solution proposed in this project is to achieve predictive modeling with high accuracy of predicting if the patient has autism or not.

5.1. Architecture

The architecture of the project can be described by considering the following process flow:

1. Collecting the raw datasets related to both Autism and Non-Autism cases
2. Performing the data pre-processing on the raw datasets. Preprocessing includes data cleaning and data sampling.
3. Loading the data over the usable storage.
4. Performing data operations towards feature extraction on the pre-processed data. Feature extraction includes several steps.
5. Training and tuning the classifier to achieve high accuracy.
6. Testing or evaluating the classifier or predictive model.

The process flow can be understood pictorially from the following diagram:

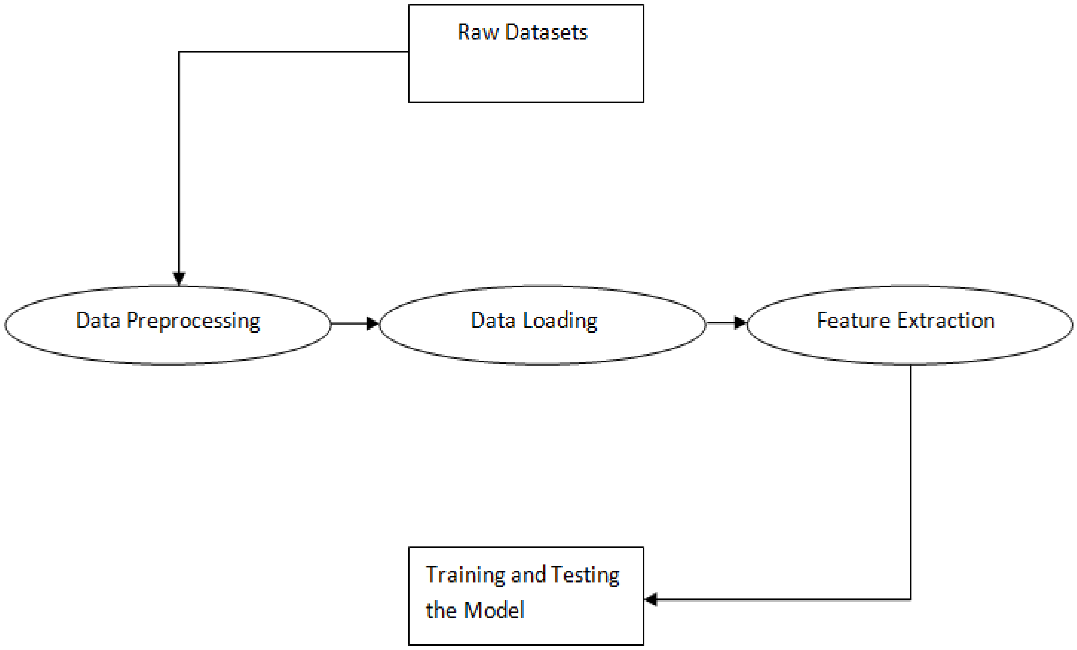


Figure 1: Architecture of Proposed Solution

6. Implementation using Weka

Weka is open source software. It is a collection of machine learning algorithms used for data mining. The machine learning algorithms can be applied on the datasets directly. Weka contains tools and methods for data preprocessing and predictive modeling. New machine learning schemes can also be developed using Weka [7].

6.1 Data Preprocessing

Data preprocessing is one of the most important steps. When we have the raw data, we have to transform it in such a way that it can be used constructively towards accomplishing our task. Often the raw data is not ready to use directly towards the application or towards the sub-processes we want to design.

For every machine learning API, platform or framework, the preprocessing of data can vary. The libraries and their usage are different for different frameworks.

6.1.1 Data Cleaning

There were a lot of attributes in the data set and not all were important for training the classifier. For text classification, attribute “symptom_text” has been considered. There is one more attribute called “vaers_id” which is a unique id provided to every patient. “symptom_text” contains description of the state of patients in the form of sentences.

There are several csv files for different years from 1991 to 2014. First of all, we separated the dataset into two parts, one with positive (with autism) instances and other with negative (without autism) instances.

For Weka, the instances of “symptom_text” had to be converted into separate text files such that each text file contains the symptom_text of one patient. To execute this, a python script was written to automate the process. For example, the vaers_id of a particular instance is 22356. Python script created text file with the name as “pos22356” if it was positive instance and “neg22356” if it was negative instance. The text files with positive instances were stored in directory named “Pos” and text files with negative instances were stored in directory named “Neg”.

After getting two separate directories for positive and negative text files, next aim was to create an arff (Attribute-Relation File Format) file which is efficiently supported by Weka. Arff file is less memory intensive and faster. For converting text files into arff file, TextDirectoryLoader class is used. TextDirectoryLoader loads the directory into an arff file. The exact command used on Weka command line interface is:

```
> java.weka.core.conerters.TextDirectoryLoader –dir “Path of source directory” > “Path of destination directory where final arff file is to be kept”
```

In this way, arff file containing the data was produced. Weka automatically adds an attribute called “class” and provides a value of class to each instance based on the directory in which they have been stored. For example, names of directories here are Pos and Neg, so the values of class attribute given to positive instances are “Pos” and to negative instances are “Neg”.

6.1.2 Data Sampling

To ultimately achieve predictive modeling, the data must have to be divided into training set and test set. Ideally, the training set consists of 66% of the whole data and test set consists of 34%. This is done because model should get enough data (two-third of data) to get trained and after that there should be completely new data (Rest one third of data) with which model must be tested for calculating its accuracy towards prediction.

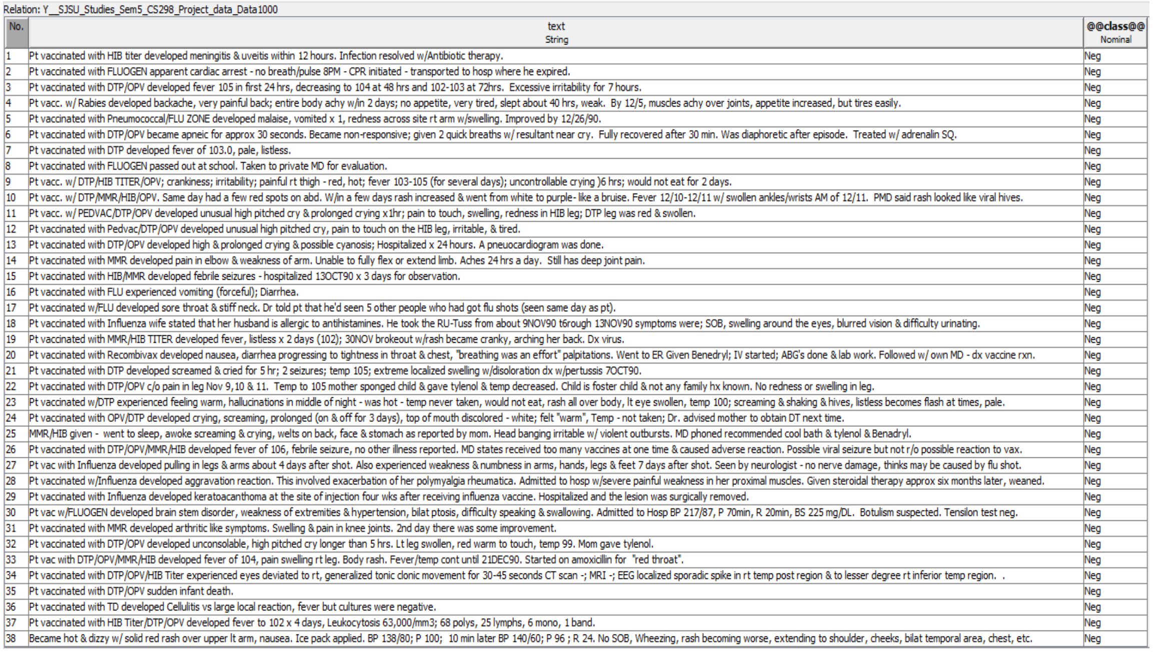
Data Sampling in Weka is quite straight forward because when the classifier is executed, it gives an option of splitting the whole data in training and test set. So 66% can be chosen for training data.

6.2 Predictive Modeling

6.2.1 Importing Data and Feature Extraction

Feature extraction is first step towards the data modeling. Features can be understood as the most important words in the data set which can be considered as deciding factor behind the predictions, For example, features in this project can be the most prominent words in the “symptom_text”. Features play a significant and vital role in further classification algorithms. There are several ranking algorithms which can be used to select features out of the text.

After importing data (arff) in Weka, it looked like:



Relation: Y__SJSU_Studies_Sem5_CS298_Project_data_Data1000

No.	text String	@@class@@ Nominal
1	Pt vaccinated with HIB titer developed meningitis & uveitis within 12 hours. Infection resolved w/Antibiotic therapy.	Neg
2	Pt vaccinated with FLUOGEN apparent cardiac arrest - no breath/pulse 8PM - CPR initiated - transported to hosp where he expired.	Neg
3	Pt vaccinated with DTP/OPV developed fever 105 in first 24 hrs, decreasing to 104 at 48 hrs and 102-103 at 72hrs. Excessive irritability for 7 hours.	Neg
4	Pt vacc. w/ Rabies developed backache, very painful back; entire body achy w/in 2 days; no appetite, very tired, slept about 40 hrs, weak. By 12/5, muscles achy over joints, appetite increased, but tires easily.	Neg
5	Pt vaccinated with Pneumococcal/FLU ZONE developed malaise, vomited x 1, redness across site rt arm w/swelling. Improved by 12/26/90.	Neg
6	Pt vaccinated with DTP/OPV became apneic for approx 30 seconds. Became non-responsive; given 2 quick breaths w/ resultant near cry. Fully recovered after 30 min. Was diaphoretic after episode. Treated w/ adrenalin SQ.	Neg
7	Pt vaccinated with DTP developed fever of 103.0, pale, listless.	Neg
8	Pt vaccinated with FLUOGEN passed out at school. Taken to private MD for evaluation.	Neg
9	Pt vacc. w/ DTP/HIB TITER/OPV; crankiness; irritability; painful rt thigh - red, hot; fever 103-105 (for several days); uncontrollable crying)6 hrs; would not eat for 2 days.	Neg
10	Pt vacc. w/ DTP/MMR/HIB/OPV. Same day had a few red spots on abd. W/in a few days rash increased & went from white to purple- like a bruise. Fever 12/10-12/11 w/ swollen ankles/wrists AM of 12/11. PMD said rash looked like viral hives.	Neg
11	Pt vacc. w/ PEDVAC/DTP/OPV developed unusual high pitched cry & prolonged crying x1hr; pain to touch, swelling, redness in HIB leg; DTP leg was red & swollen.	Neg
12	Pt vaccinated with Pedvac/DTP/OPV developed unusual high pitched cry, pain to touch on the HIB leg, irritable, & tired.	Neg
13	Pt vaccinated with DTP/OPV developed high & prolonged crying & possible cyanosis; Hospitalized x 24 hours. A pneuocardiogram was done.	Neg
14	Pt vaccinated with MMR developed pain in elbow & weakness of arm. Unable to fully flex or extend limb. Aches 24 hrs a day. Still has deep joint pain.	Neg
15	Pt vaccinated with HIB/MMR developed febrile seizures - hospitalized 13OCT90 x 3 days for observation.	Neg
16	Pt vaccinated with FLU experienced vomiting (forceful); Diarrhea.	Neg
17	Pt vaccinated w/FLU developed sore throat & stiff neck. Dr told pt that he'd seen 5 other people who had got flu shots (seen same day as pt).	Neg
18	Pt vaccinated with Influenza wife stated that her husband is allergic to antihistamines. He took the RU-Tuss from about 9NOV90 t6rough 13NOV90 symptoms were; SOB, swelling around the eyes, blurred vision & difficulty urinating.	Neg
19	Pt vaccinated with MMR/HIB TITER developed fever, listless x 2 days (102); 30NOV brokeout w/rash became cranky, arching her back. Dx virus.	Neg
20	Pt vaccinated with Recombivax developed nausea, diarrhea progressing to tightness in throat & chest, "breathing was an effort" palpitations. Went to ER Given Benedryl; IV started; ABG's done & lab work. Followed w/ own MD - dx vaccine rxn.	Neg
21	Pt vaccinated with DTP developed screamed & cried for 5 hr; 2 seizures; temp 105; extreme localized swelling w/disoloration dx w/pertussis 7OCT90.	Neg
22	Pt vaccinated with DTP/OPV c/o pain in leg Nov 9,10 & 11. Temp to 105 mother sponged child & gave tylenol & temp decreased. Child is foster child & not any family hx known. No redness or swelling in leg.	Neg
23	Pt vaccinated w/DTP experienced feeling warm, hallucinations in middle of night - was hot - temp never taken, would not eat, rash all over body, lt eye swollen, temp 100; screaming & shaking & hives, listless becomes flash at times, pale.	Neg
24	Pt vaccinated with OPV/DTP developed crying, screaming, prolonged (on & off for 3 days), top of mouth discolored - white; felt "warm", Temp - not taken; Dr. advised mother to obtain DT next time.	Neg
25	MMR/HIB given - went to sleep, awoke screaming & crying, welts on back, face & stomach as reported by mom. Head banging irritable w/ violent outbursts. MD phoned recommended cool bath & tylenol & Benadryl.	Neg
26	Pt vaccinated with DTP/OPV/MMR/HIB developed fever of 106, febrile seizure, no other illness reported. MD states received too many vaccines at one time & caused adverse reaction. Possible viral seizure but not r/o possible reaction to vax.	Neg
27	Pt vac with Influenza developed pulling in legs & arms about 4 days after shot. Also experienced weakness & numbness in arms, hands, legs & feet 7 days after shot. Seen by neurologist - no nerve damage, thinks may be caused by flu shot.	Neg
28	Pt vaccinated w/Influenza developed aggravation reaction. This involved exacerbation of her polymyalgia rheumatica. Admitted to hosp w/severe painful weakness in her proximal muscles. Given steroidal therapy approx six months later, weaned.	Neg
29	Pt vaccinated with Influenza developed keratoacanthoma at the site of injection four wks after receiving influenza vaccine. Hospitalized and the lesion was surgically removed.	Neg
30	Pt vac w/FLUOGEN developed brain stem disorder, weakness of extremities & hypertension, bilat ptosis, difficulty speaking & swallowing. Admitted to Hosp BP 217/87, P 70min, R 20min, BS 225 mg/DL. Botulism suspected. Tensilon test neg.	Neg
31	Pt vaccinated with MMR developed arthritic like symptoms. Swelling & pain in knee joints. 2nd day there was some improvement.	Neg
32	Pt vaccinated with DTP/OPV developed unconsolable, high pitched cry longer than 5 hrs. Lt leg swollen, red warm to touch, temp 99. Mom gave tylenol.	Neg
33	Pt vac with DTP/OPV/MMR/HIB developed fever of 104, pain swelling rt leg. Body rash. Fever/temp cont until 21DEC90. Started on amoxicillin for "red throat".	Neg
34	Pt vaccinated with DTP/OPV/HIB Titer experienced eyes deviated to rt, generalized tonic clonic movement for 30-45 seconds CT scan -; MRI -; EEG localized sporadic spike in rt temp post region & to lesser degree rt inferior temp region. .	Neg
35	Pt vaccinated with DTP/OPV sudden infant death.	Neg
36	Pt vaccinated with TD developed Cellulitis vs large local reaction, fever but cultures were negative.	Neg
37	Pt vaccinated with HIB Titer/DTP/OPV developed fever to 102 x 4 days, Leukocytosis 63,000/mm3; 68 polys, 25 lymphs, 6 mono, 1 band.	Neg
38	Became hot & dizzy w/ solid red rash over upper lt arm, nausea. Ice pack applied. BP 138/80; P 100; 10 min later BP 140/60; P 96 ; R 24. No SOB, Wheezing, rash becoming worse, extending to shoulder, cheeks, bilat temporal area, chest, etc.	Neg

Figure 2: Data after importing into Weka

StringToWordVector:

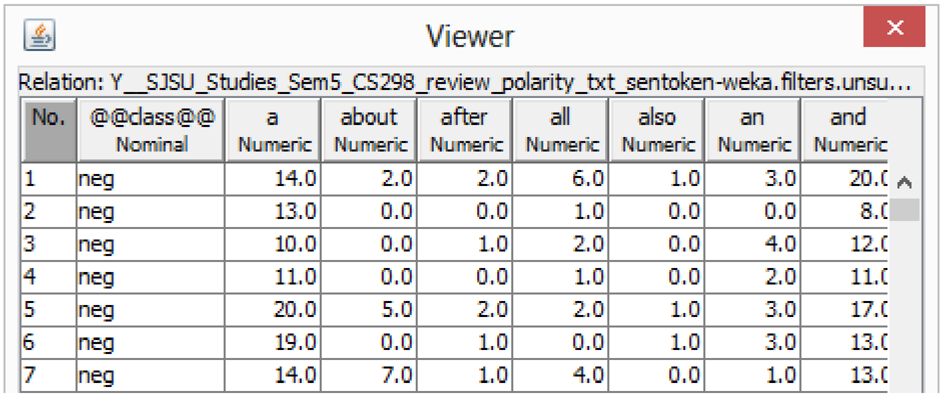
String is a datatype which machine learning classifiers usually cannot process. Hence using StringToWordVector, random text has to be transformed into document vectors. Table is formed in which Text Documents are rows (Document vectors), words as columns and values as numbers. In this way, numbers can represent text. StringToWordVector converts String data into numeric or nominal data which learning algorithms can process. There are several settings or parameters which are needed to be set according to the requirements of the problem. Parameters are explained as follows:

- **wordsToKeep**

  This is the total number of words desired to be considered in the data. This value has to be decided on predicting the intensity of the data and also on the basis of a particular number we want to deal with after the first filter. Maximum 6-digit number can be assigned, so all the words will be considered.

- **outputWordsCount**

  If this setting is put true, then values in the table will be the number of times that word occurs in that document.



Relation: Y__SJSU_Studies_Sem5_CS298_review_polarity_txt_sentoken-weka.filters.unsu...

No.	@@class@@ Nominal	a Numeric	about Numeric	after Numeric	all Numeric	also Numeric	an Numeric	and Numeric
1	neg	14.0	2.0	2.0	6.0	1.0	3.0	20.
2	neg	13.0	0.0	0.0	1.0	0.0	0.0	8.
3	neg	10.0	0.0	1.0	2.0	0.0	4.0	12.
4	neg	11.0	0.0	0.0	1.0	0.0	2.0	11.
5	neg	20.0	5.0	2.0	2.0	1.0	3.0	17.
6	neg	19.0	0.0	1.0	0.0	1.0	3.0	13.
7	neg	14.0	7.0	1.0	4.0	0.0	1.0	13.

Figure 3: Values as outputWordsCount

- **doNotOperateOnPerClassBasis**

  If this setting is put true, then the number of “wordsToKeep” is considered in total irrespective of class (Pos/Neg) otherwise the number of “wordsToKeep” is considered on per class basis. For example, if the number of “wordsToKeep” is 1000, then in case of true the 1000 words will be considered otherwise 1000 from each class. But this does not mean that the number will be 2000 because there might be overlapping also as some words might be same in both the classes. So the number might be between 1000 and 2000.

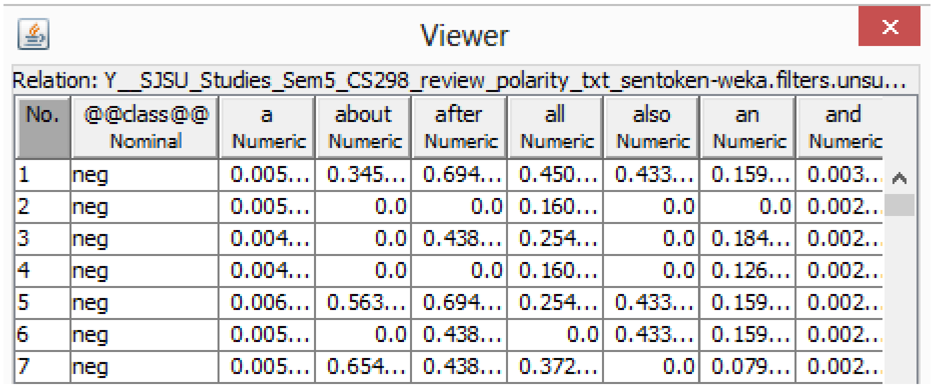
- **IDFTransform and TFTransform**

  TF-IDF is a way to find words and documents that are strongly related. For example, if a word “super” appears in 3 of the 2000 documents, that’s low document frequency (DF). Out of the 3, which document is “super” related the most? “super” appears 100, 2, 5 times in 3 documents respectively. So “super” has high term frequency (TF) in the first document as it appears 100 times and a very high TF-IDF ranking. Higher value of TF-IDF score means the word is important for that document.

Words that rarely appear in document collection and frequently appear in particular documents:

IDFTransform: True (Low DF)

TFTransform: True (High TF)



Relation: Y__SJSU_Studies_Sem5_CS298_review_polarity_txt_sentoken-weka.filters.unsu...

No.	@@class@@ Nominal	a Numeric	about Numeric	after Numeric	all Numeric	also Numeric	an Numeric	and Numeric
1	neg	0.005...	0.345...	0.694...	0.450...	0.433...	0.159...	0.003...
2	neg	0.005...	0.0	0.0	0.160...	0.0	0.0	0.002...
3	neg	0.004...	0.0	0.438...	0.254...	0.0	0.184...	0.002...
4	neg	0.004...	0.0	0.0	0.160...	0.0	0.126...	0.002...
5	neg	0.006...	0.563...	0.694...	0.254...	0.433...	0.159...	0.002...
6	neg	0.005...	0.0	0.438...	0.0	0.433...	0.159...	0.002...
7	neg	0.005...	0.654...	0.438...	0.372...	0.0	0.079...	0.002...

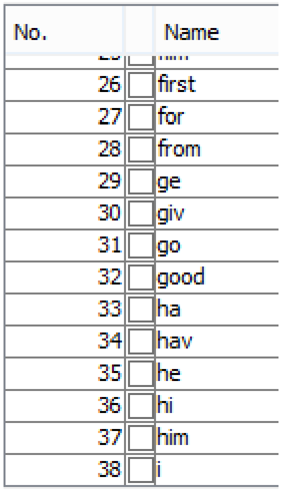
Figure 4: Values as TF-IDF Score

- **normalizeDocLength**

  Normalization refers to measurements taken on different scales and re-measuring them on a common scale. For example, There are two documents: doc1=100 words and doc2=150 words. The word “about” appears 2 times in both but the normalization value of “about” for doc1 will be more than in doc2 because doc1 has less words and effect of "about" is more in doc1.

- **Stemmer**

  Stemming tries to use words better by breaking them into a smaller form called stem. To perform stemming, it can inspect both sides of a word to try to remove letters from either of side and typically it removes from the suffix side. In Weka, default is NullStemmer which doesn't do any stemming. Lovins Stemmers and Porter Stemmers are two popular types. The stem is not necessarily a linguistically valid word. E.g. the word "have" may lose "e" and become "hav".



No.		Name
26	☐	first
27	☐	for
28	☐	from
29	☐	ge
30	☐	giv
31	☐	go
32	☐	good
33	☐	ha
34	☐	hav
35	☐	he
36	☐	hi
37	☐	him
38	☐	i

Figure 5: Words after stemming

- **Stop words**

  Stop words may tend to be irrelevant for classification. "the", "is", "at", "on" etc. are stop words. StringToWordVector by default uses an English language stop words list in Weka.

- **Tokenizer**

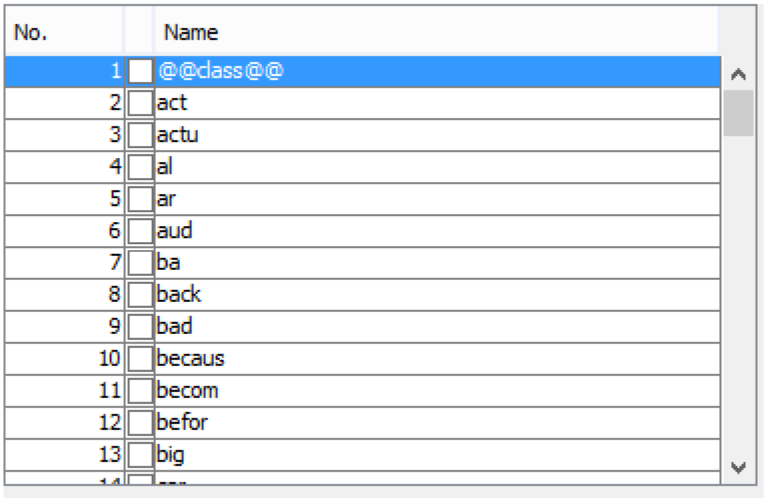
  Tokenizer algorithms have different ways of splitting up the text. They split into tokens.

  Ngram Tokenizer

  Finding potentially predictive unigrams, bigrams, trigrams, …, five-grams. It's about the unit of measurement. Ngram Tokenizer in Weka by default tries to find predictive single word units, predictive two word units, predictive three word units. Using Ngram Tokenizer, output is words and phrases.

  Alphabetic Tokenizer

  With the default word tokenizer, some words contain signs like @, &, ~, -, --, whereas using Alphabetic Tokenizer, all of those get eliminated and every token is 100% letters in the alphabet. Alphabetic Tokenizer would have unigrams only, unlike the Ngram Tokenizer where there are words and phrases both. We can see in the fig. that there are only unigrams with only English words and without any signs.



No.		Name
1	☐	@@class@@
2	☐	act
3	☐	actu
4	☐	al
5	☐	ar
6	☐	aud
7	☐	ba
8	☐	back
9	☐	bad
10	☐	becaus
11	☐	becom
12	☐	befor
13	☐	big

Figure 6: Words after applying Alphabetic Tokenizer

- **minTermFrequency**

  minTermFrequency is the minimum number for which any word has to appear to be considered as an attribute. This can be set as per the requirement of the data.

- **lowerCaseTokens**

  If lowerCaseTokens setting is off, both lower case and upper case words are considered different attributes.

  If lowerCaseTokens setting is on, both lower case and upper case words are considered to be lower case and as a same attribute.

  This setting is important because many a times, words mean same but they have just been written in different cases.

AttributeSelectionFilter

The AttributeSelectionFilter often compliments the StringToWordVector as high quality input data is created. StringToWordVector changed all the symptom_text and their words into document vectors. AttributeSelection is different. It does not change characters into different numbers. It ranks the attributes and further improves the input data.

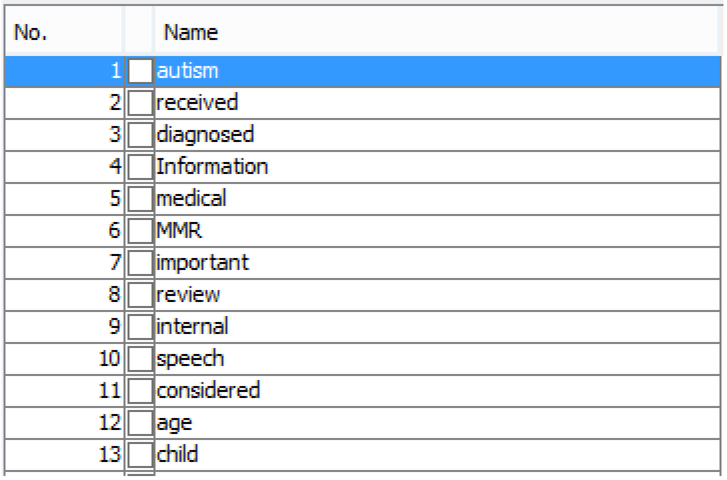
Under the settings of AttributeSelectionFilter, Evaluator and Search can be chosen which are explained as follows:

Evaluator – InfoGainAttribueEval

Evaluator is the judge for judging the predictive quality of the attribute.

Search – Ranker

Search consults the Judge (Evaluator) to make the final decision to accept or reject the attribute.



No.		Name
1	□	autism
2	□	received
3	□	diagnosed
4	□	Information
5	□	medical
6	□	MMR
7	□	important
8	□	review
9	□	internal
10	□	speech
11	□	considered
12	□	age
13	□	child

Figure 7: Words after applying AttributeSelectionFilter

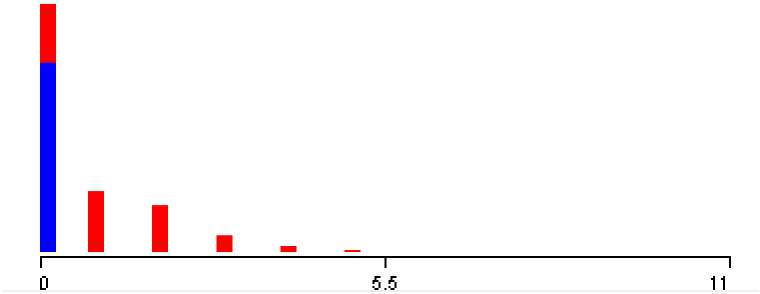


Figure 8: (Value, Frequency) distribution of word “autism”

From figure 7, we can see that “autism” is ranked # 1. “received” is ranked # 2 and so on. Figure 8 is the (value, frequency) distribution. The value is word count. Hence x-axis is word count and y-axis is frequency. The word “autism” appears 5 times in 11 docs, 4 times in 44 docs, 3 times in 141, 2 times in 386 docs, 1 time in 507 documents, 0 times in 2077 documents. Blue color represents Negative class and Red color represents Positive class. In all the documents in which “autism” appears, it can be seen that the bar is red (Positive). It becomes the deciding factor that “autism” word is always contributing towards Positive and hence it is kept on rank # 1.

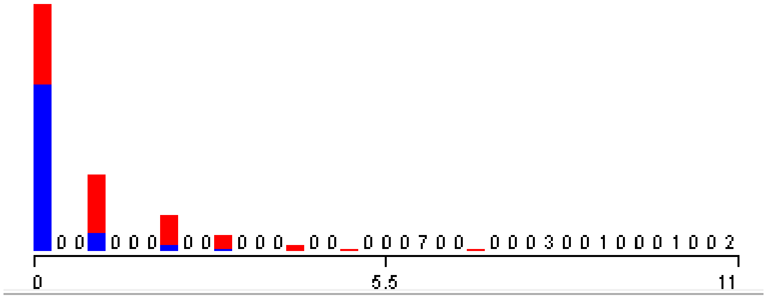


Figure 9: (Value, Frequency) distribution of word “received”

From figure 9, it can be seen that in all the documents in which “received” appears, the maximum percentage of bar is red (Positive). It becomes the deciding factor that “received” word is mostly contributing towards Positive and hence it is kept on rank # 2.

In this way, words are ranked on the basis of their one-sided contribution and decisive capability towards any one of the classes. All of the words’ distribution can be seen and analyzed as shown in figure 10.

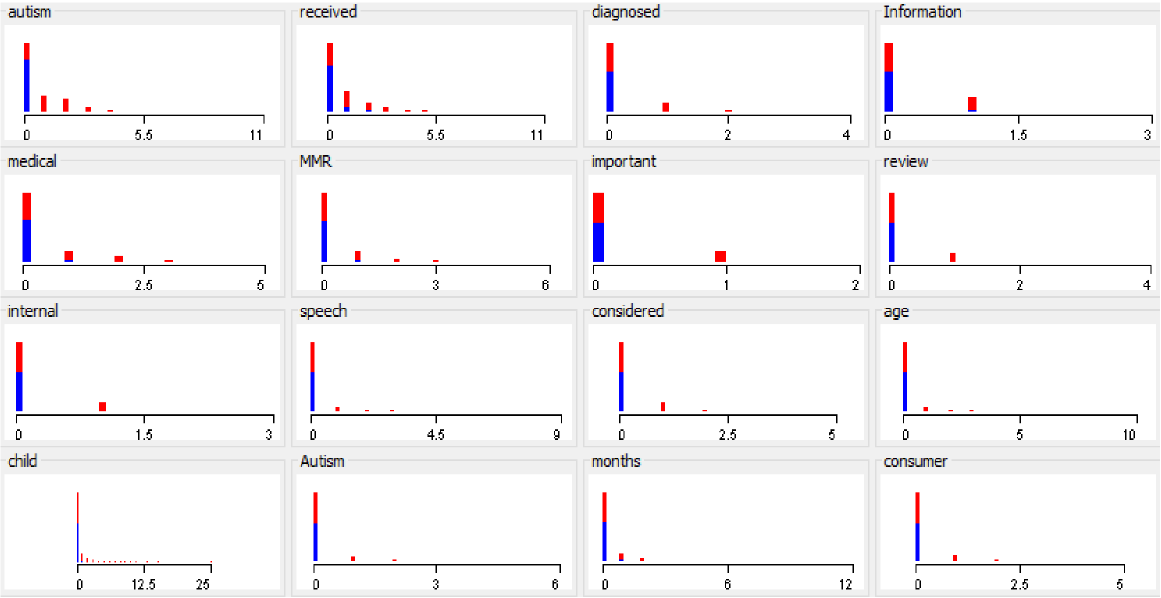
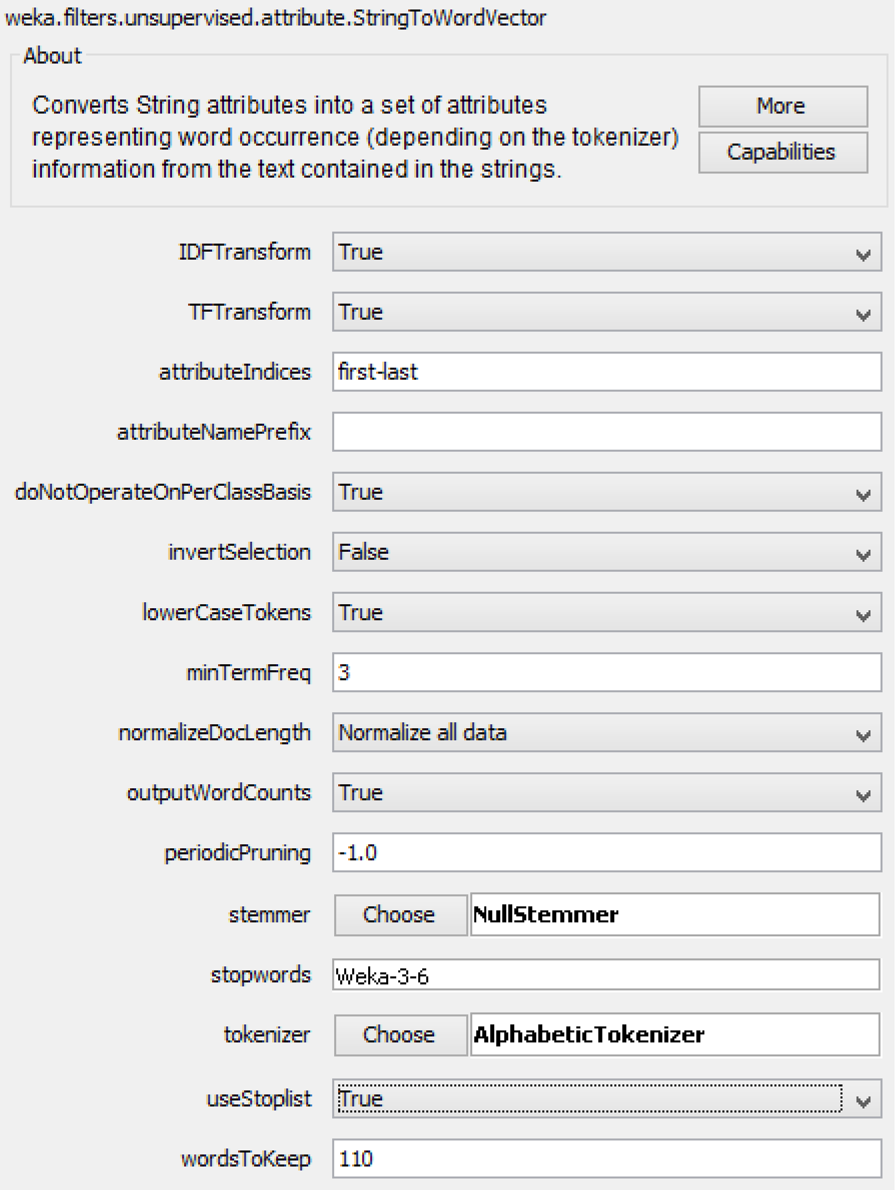


Figure 10: (Value, Frequency) distribution of some other words

The settings applied to the data in this project are as shown below:

The settings for StringToWordVector applied to the data in this project are as shown below in figure 11:



weka.filters.unsupervised.attribute.StringToWordVector

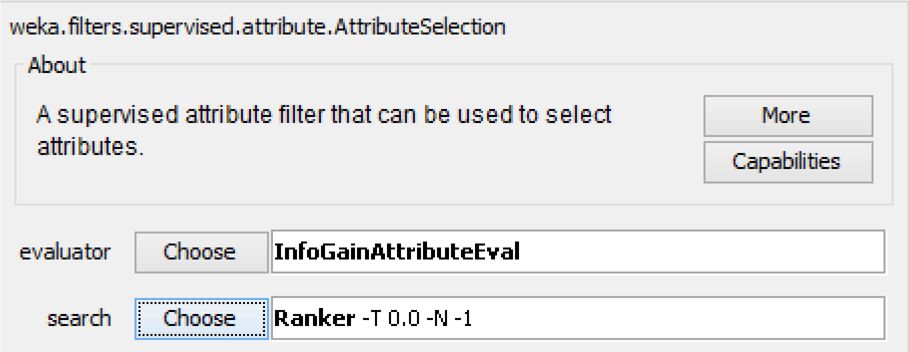
About

Converts String attributes into a set of attributes representing word occurrence (depending on the tokenizer) information from the text contained in the strings.

```
IDFTransform                 True
TFTransform                  True
attributeIndices             first-last
attributeNamePrefix
doNotOperateOnPerClassBasis  True
invertSelection              False
lowerCaseTokens              True
minTermFreq                  3
normalizeDocLength           Normalize all data
outputWordCounts             True
periodicPruning              -1.0
stemmer                      NullStemmer
stopwords                    Weka-3-6
tokenizer                    AlphabeticTokenizer
useStoplist                  True
wordsToKeep                  110
```

Figure 11: StringToWordVector Settings

The settings for AttributeSelectionFilter applied to the data in this project are as shown below in figure 12:



weka.filters.supervised.attribute.AttributeSelection

About

A supervised attribute filter that can be used to select attributes.

evaluator InfoGainAttributeEval

search Ranker -T 0.0 -N -1

Figure 12: AttributeSelectionFilter Settings

6.2.2 Executing the classifier model

After performing data preprocessing and feature extraction, final set of features have been retrieved. Now in this stage we can execute the classifier of our wish. For this project, Naïve Bayes classifier and Logistic Regression have been chosen.

After executing Naïve Bayes Classifier and Logistic Regression model, accuracies are as follows:

Naïve Bayes Classifier: 84.73% (Correctly classified instances)

Logistic Regression Model: 89.17% (Correctly classified instances)

Confusion Matrix (Naïve Bayes)

(Neg)	(Pos)	
86501	10166	Neg error: 10.51%
11976	36357	Pos error: 24.77%

Confusion Matrix (Logistic Regression)

(Neg)	(Pos)	
90793	5874	Neg error: 6.07%
8771	39562	Pos error: 18.14%

7. Implementation using Apache Spark

Apache Spark is an open source framework used primarily for cluster computing. Spark allows user to load data into cluster memory and also it's quite efficient in querying the data repeatedly. Because of high efficiency of Spark, it is very well suited for machine learning algorithms.

Spark MLlib is one of the four big libraries built on the top of Spark.

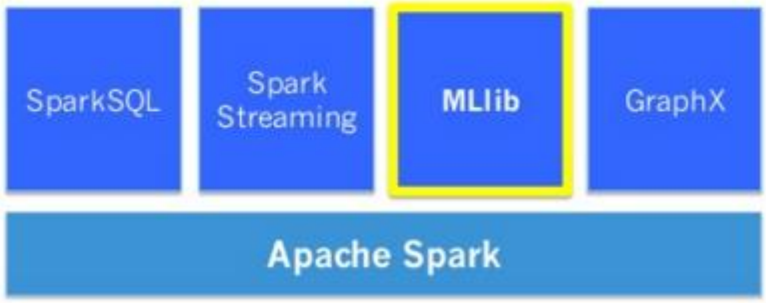


Figure 13: Major libraries in Apache Spark

Spark has a number of algorithms and it allows users to quickly tie those algorithms and use their custom algorithm.

The basic workflow adapted in this project is as shown below in figure 14:

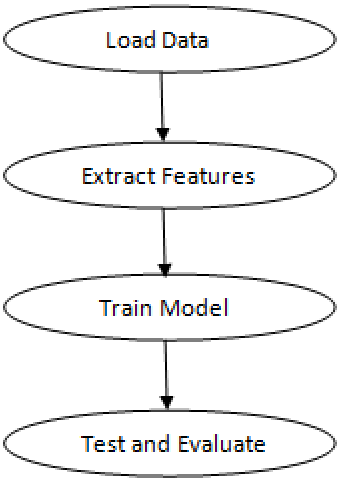


Figure 14: Basic Workflow

Version and system requirements:

For this project, when running locally, Spark 1.4.1 has been used and when running on cluster, Spark 1.5.0 has been used. Spark provides high-level API's for Java, Scala, Python and R. For machine learning, Spark provides higher-level tool or library "MLlib".

For this project, Scala has been used as programming language and Spark has been installed on Linux based system Ubuntu 14.04.

Java must be installed on the system and it's environment variables must be set properly. Spark supports Java 7+, Python 2.6+, R 3.1+. In case of Scala API, Spark uses Scala 2.10. Hence it is needed to be compatible with Scala 2.10.x version.

Spark can be run interactively through spark-shell. Once spark-shell is launched, interactive programs can be written in Scala.

Command to launch spark-shell after installation:

```
$ ./bin/spark-shell
```

7.1 Data Preprocessing

7.1.1 Data Cleaning, Validation and Sampling

As mentioned earlier, we have csv files containing data about patients from 1991 to 2014. It has both positive and negative instances. The csv files have several attributes.

The data operations were performed on Linux environment. Hence the commands on Linux terminal and their explanation are as follows:

To start with, first the total number of lines/instances is counted in total. This will be useful for validating the data.

To count the number of lines in each csv files:

```
$ wc -l *.csv  (Also outputs total number of instances in all the files)
```

Now all the files will be concatenated into one file.

To concatenate all the csv files and store the concatenated result in output.csv:

```
$ cat *.csv > outputpos.csv
$ cat *.csv > outputneg.csv
```

To validate the data of output.csv, the command for counting the instances is run again and it should output double the total number of instances because now there is one more concatenated file which contains all the instances:

```
$ wc -l *.csv
```

For text classification, only the relevant attributes have to be considered and other attributes have to be removed.

To remove the extra attributes and consider only those required for text classification (vaers_id and symptom_text):

```
$ cut –d, -f1,9 outputpos.csv > outputpos1.csv
$ cut –d, -f1,9 outputneg.csv > outputneg1.csv
```

(vaers_id and symptom_text were 1st and 9th attributes)

So now we have two final csv files (with two attributes): one with positive instances and second with negative instances.

Now we would remove duplicates and sort the instances so that they get randomized and don't remain in sequence of their years. In this case, as we have merged the files, so duplicates are header names vaers_id, symptom_text. So they will be removed.

To remove duplicates:

```
$ sort –u outputpos1.csv –o outputpos2.csv
$ sort –u outputpos1.csv –o outputpos2.csv
```

So now we have files in which there are no duplicates.

Also we have to define headers while loading the data in Spark. So this time we have to remove headers, otherwise after loading data into Spark headers will be included two times.

To remove header information:

To search if line exists:

```
$ grep ‘/VAERS_ID/d’ outputpos2.csv
$ grep ‘/VAERS_ID/d’ outputneg2.csv
```

To remove line:

```
$ sed “‘/VAERS_ID/d’ outputpos2.csv > outputpos3.csv
$ sed “‘/VAERS_ID/d’ outputneg2.csv > outputneg3.csv
```

An attribute called “label” has to be added which will be decisive for text classification. For positive instances (with autism) the value of “label” is given 0. For negative instances (without autism) the value of “label” is given 1.

To add the attribute “label” with values 0 and 1:

```
$ awk –F”,” ‘BEGIN{OFS=”,”}{$3=0;print}’ outputpos3.csv > outputpos4.csv
$ awk –F”,” ‘BEGIN{OFS=”,”}{$3=1;print}’ outputneg3.csv > outputneg4.csv
```

So now the whole usable data has been created. We have two csv files, one with positive instances and another with negative instances.

Now we have to create training dataset and test dataset. For this purpose, 66% of data is kept for training the model and 34% for testing and evaluating the model.

To sample data for training and testing:

```
$ head -1050 outputpos4.csv > training_positive.csv
$ head -1050 outputneg4.csv > training_negative.csv
$ tail -540 outputpos4.csv > test_positive.csv
$ tail -540 outputneg4.csv > test_negative.csv
$ wc –l training_*.csv test_*.csv
cat training_*.csv > trainingData_Label01.csv
cat test_*.csv > testData_Label01.csv
```

7.2 Predictive Modeling

7.2.1 Loading Data

In Spark, the data has been loaded by using DataFrame API. DataFrame can be understood as a data collection in distributed manner organized into named, specified columns. DataFrame can also be thought of as conceptually equivalent to a table in relational database. There are several formats or sources using which DataFrames can be built such as tables in Hive, existing RDDs (Resilient Distributed Datasets), external databases or structures data files.

First of all, after importing relevant libraries, training data is loaded from the location where it is kept locally and stored into trainData. While building case class, the structure or attributes (with data types) of data is defined. Then according to attributes, the values have been split using “,” and also the values have been mapped with attributes. The result is stored in “autismtrain”. “autismtrain” is then converted into DataFrame using toDF() method to perform further operations on DataFrame. Hence autismtrain_DF is the DataFrame created. To view the data and schema of DataFrame, show() and printSchema() methods have been used.

Code Snippet for loading data

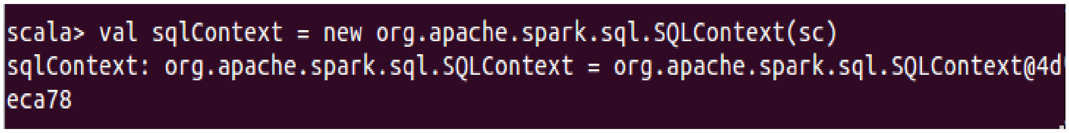
```
val sqlContext = new org.apache.spark.sql.SQLContext(sc)
import sqlContext.implicits._
import org.apache.spark.sql._

val trainData = sc.textFile("/media/arpitarya07/New
Volume1/SJSU_Studies/Sem5_CS298/Project_data/Filtered_csvs/training
Data_Label01.csv")

case class AutismTrain(vaers_id: String, symptom_text: String, label:
Double)

val autismtrain = trainData.map(_.split(",")).map(p =>
AutismTrain(p(0),p(1),p(2).toDouble))
```

val autismtrain_DF = autismtrain.toDF()

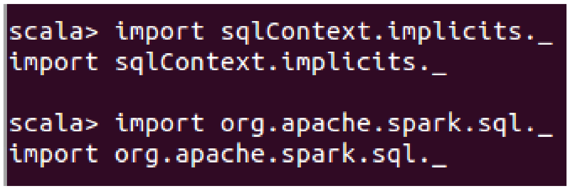
autismtrain_DF.show()

autismtrain_DF.printSchema()

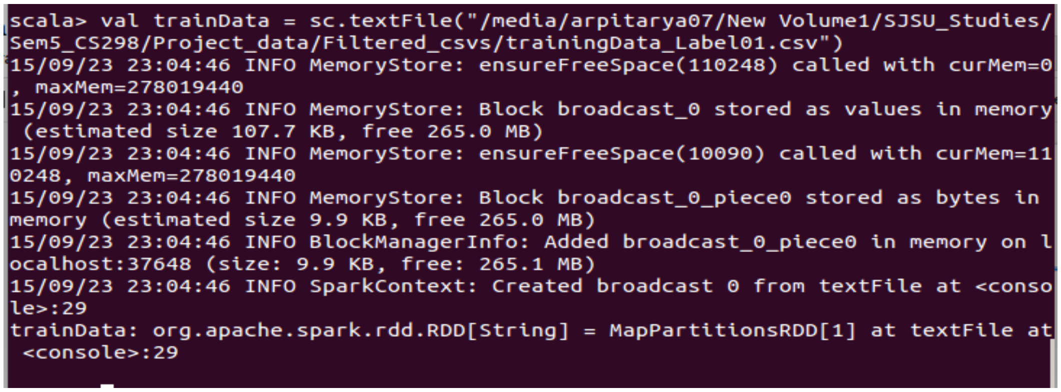
Snapshots of code and its results



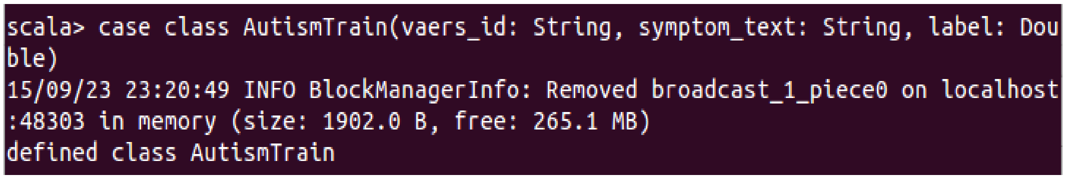
```
scala> val sqlContext = new org.apache.spark.sql.SQLContext(sc)
sqlContext: org.apache.spark.sql.SQLContext = org.apache.spark.sql.SQLContext@4d
eca78
```



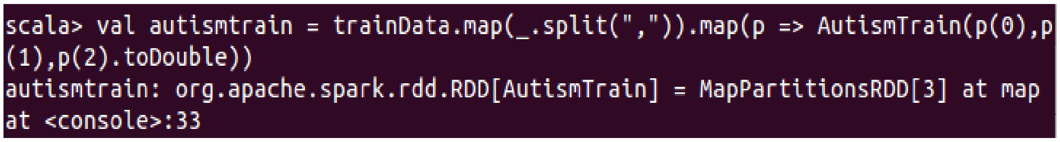
```
scala> import sqlContext.implicits._
import sqlContext.implicits._

scala> import org.apache.spark.sql._
import org.apache.spark.sql._
```



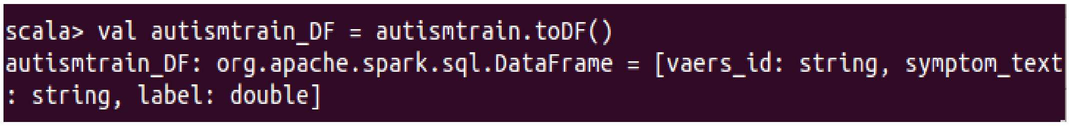
```
scala> val trainData = sc.textFile("/media/arpitarya07/New Volume1/SJSU_Studies/
Sem5_CS298/Project_data/Filtered_csvs/trainingData_Label01.csv")
15/09/23 23:04:46 INFO MemoryStore: ensureFreeSpace(110248) called with curMem=0
, maxMem=278019440
15/09/23 23:04:46 INFO MemoryStore: Block broadcast_0 stored as values in memory
 (estimated size 107.7 KB, free 265.0 MB)
15/09/23 23:04:46 INFO MemoryStore: ensureFreeSpace(10090) called with curMem=11
0248, maxMem=278019440
15/09/23 23:04:46 INFO MemoryStore: Block broadcast_0_piece0 stored as bytes in
memory (estimated size 9.9 KB, free 265.0 MB)
15/09/23 23:04:46 INFO BlockManagerInfo: Added broadcast_0_piece0 in memory on l
ocalhost:37648 (size: 9.9 KB, free: 265.1 MB)
15/09/23 23:04:46 INFO SparkContext: Created broadcast 0 from textFile at <conso
le>:29
trainData: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[1] at textFile at
 <console>:29
```



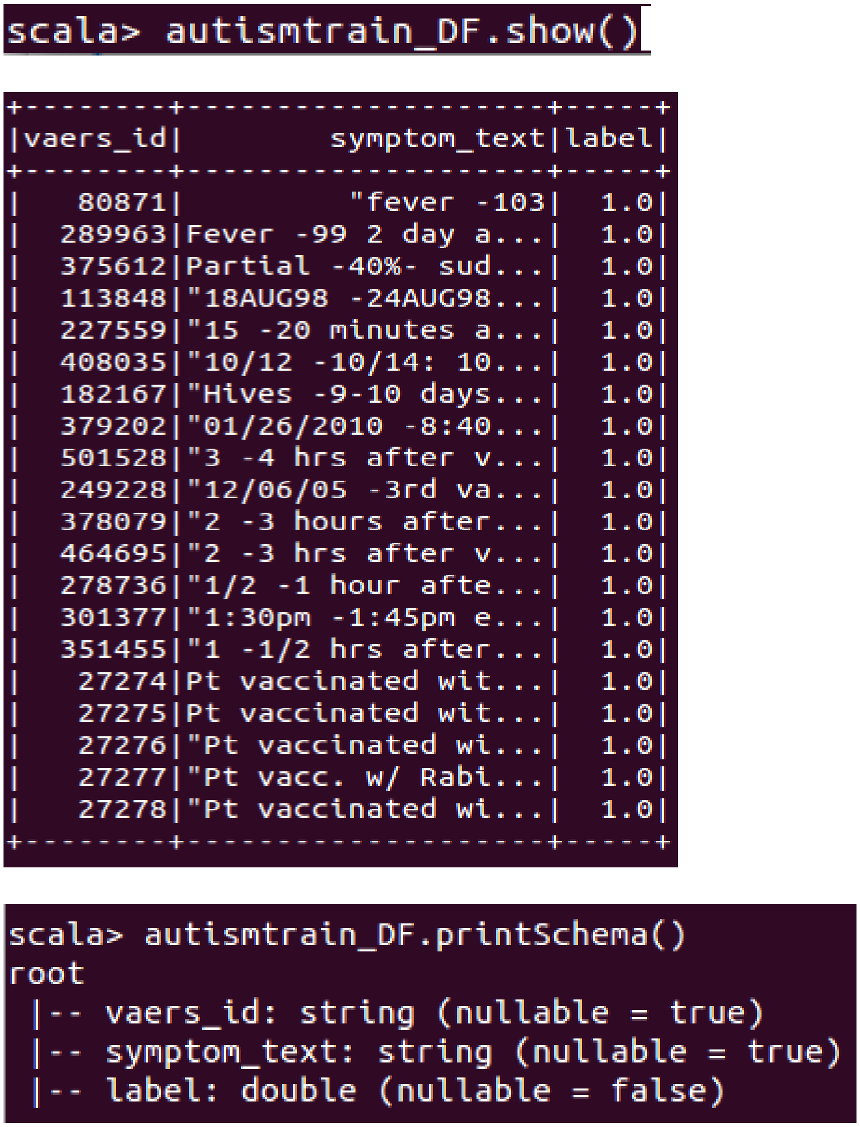
```
scala> case class AutismTrain(vaers_id: String, symptom_text: String, label: Dou
ble)
15/09/23 23:20:49 INFO BlockManagerInfo: Removed broadcast_1_piece0 on localhost
:48303 in memory (size: 1902.0 B, free: 265.1 MB)
defined class AutismTrain
```



```
scala> val autismtrain = trainData.map(_.split(",")).map(p => AutismTrain(p(0),p
(1),p(2).toDouble))
autismtrain: org.apache.spark.rdd.RDD[AutismTrain] = MapPartitionsRDD[3] at map
at <console>:33
```



```
scala> val autismtrain_DF = autismtrain.toDF()
autismtrain_DF: org.apache.spark.sql.DataFrame = [vaers_id: string, symptom_text
: string, label: double]
```



```
scala> autismtrain_DF.show()
```

```
+--------+--------------------+-----+
|vaers_id|        symptom_text|label|
+--------+--------------------+-----+
|   80871|          "fever -103|  1.0|
|  289963|Fever -99 2 day a...|  1.0|
|  375612|Partial -40%- sud...|  1.0|
|  113848|"18AUG98 -24AUG98...|  1.0|
|  227559|"15 -20 minutes a...|  1.0|
|  408035|"10/12 -10/14: 10...|  1.0|
|  182167|"Hives -9-10 days...|  1.0|
|  379202|"01/26/2010 -8:40...|  1.0|
|  501528|"3 -4 hrs after v...|  1.0|
|  249228|"12/06/05 -3rd va...|  1.0|
|  378079|"2 -3 hours after...|  1.0|
|  464695|"2 -3 hrs after v...|  1.0|
|  278736|"1/2 -1 hour afte...|  1.0|
|  301377|"1:30pm -1:45pm e...|  1.0|
|  351455|"1 -1/2 hrs after...|  1.0|
|   27274|Pt vaccinated wit...|  1.0|
|   27275|Pt vaccinated wit...|  1.0|
|   27276|"Pt vaccinated wi...|  1.0|
|   27277|"Pt vacc. w/ Rabi...|  1.0|
|   27278|"Pt vaccinated wi...|  1.0|
+--------+--------------------+-----+
```

```
scala> autismtrain_DF.printSchema()
root
 |-- vaers_id: string (nullable = true)
 |-- symptom_text: string (nullable = true)
 |-- label: double (nullable = false)
```

Figure 15: Snapshots of code and its results for loading data

7.2.2 Feature Extraction

Features have to be extracted from the text such that machine learning algorithm can understand. There are basically two steps in feature extraction:

1. Tokenizer
2. Hashed Term Frequency

Tokenizer and Hashed Term Frequency:

Tokenizer takes the entire text and breaks it into a bunch of words. In this way, a new column called “Words” gets appended to this DataFrame. This DataFrame is then passed to the next module “Hashed Term Frequency” and it outputs a new column called “Features”. It is a fixed length vector which is numerical and easily understood by machine learning algorithms.

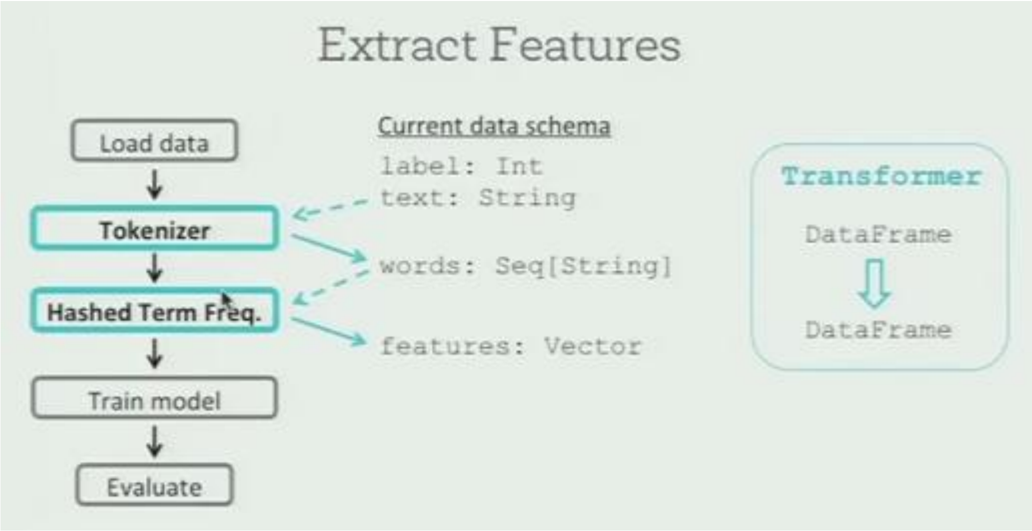


Figure 16: Feature Extraction

Code Snippet for feature extraction

```
import org.apache.spark.ml.Pipeline

import org.apache.spark.ml.classification.LogisticRegression
```

import org.apache.spark.ml.feature.{HashingTF, Tokenizer}

import org.apache.spark.mllib.linalg.Vector

import org.apache.spark.sql.Row

val tokenizer = new

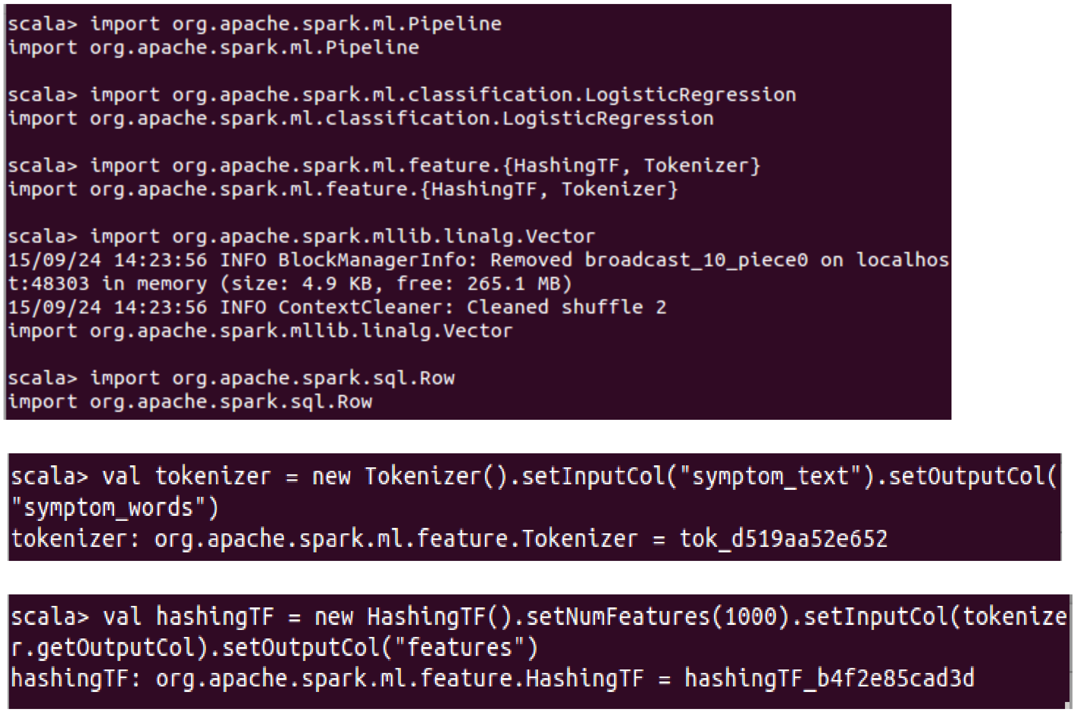
Tokenizer().setInputCol("symptom_text").setOutputCol("symptom_words")

val hashingTF = new

HashingTF().setNumFeatures(1000).setInputCol(tokenizer.getOutputCol).

setOutputCol("features")

Snapshots of code and its results



```
scala> import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.Pipeline

scala> import org.apache.spark.ml.classification.LogisticRegression
import org.apache.spark.ml.classification.LogisticRegression

scala> import org.apache.spark.ml.feature.{HashingTF, Tokenizer}
import org.apache.spark.ml.feature.{HashingTF, Tokenizer}

scala> import org.apache.spark.mllib.linalg.Vector
15/09/24 14:23:56 INFO BlockManagerInfo: Removed broadcast_10_piece0 on localhos
t:48303 in memory (size: 4.9 KB, free: 265.1 MB)
15/09/24 14:23:56 INFO ContextCleaner: Cleaned shuffle 2
import org.apache.spark.mllib.linalg.Vector

scala> import org.apache.spark.sql.Row
import org.apache.spark.sql.Row
```

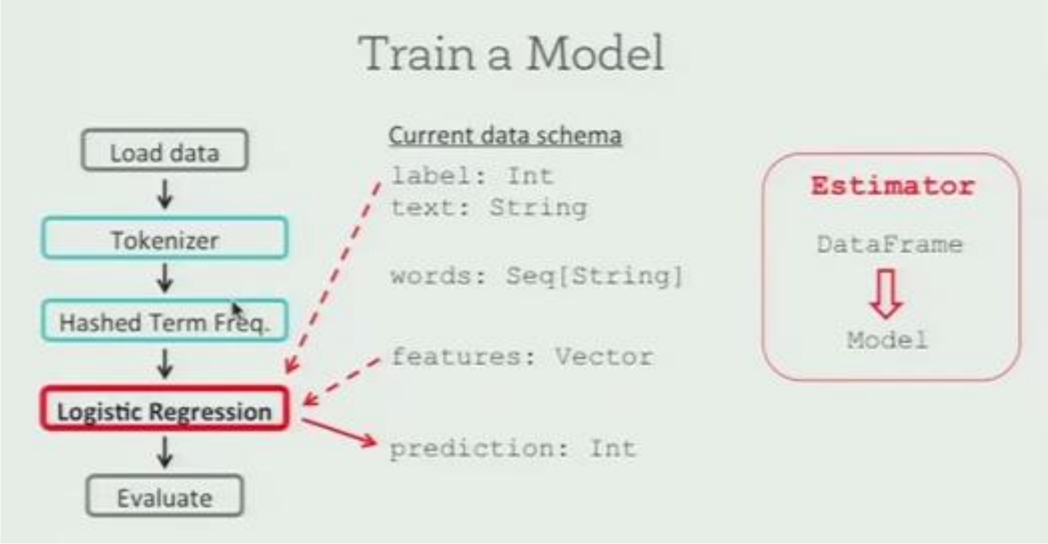
```
scala> val tokenizer = new Tokenizer().setInputCol("symptom_text").setOutputCol(
"symptom_words")
tokenizer: org.apache.spark.ml.feature.Tokenizer = tok_d519aa52e652
```

```
scala> val hashingTF = new HashingTF().setNumFeatures(1000).setInputCol(tokenize
r.getOutputCol).setOutputCol("features")
hashingTF: org.apache.spark.ml.feature.HashingTF = hashingTF_b4f2e85cad3d
```

Figure 17: Snapshots of code and its results for feature extraction

7.2.3 Train Model

Now features have been transformed and all the data can flow to the training module and “Logistic Regression” is used for that. It selects the useful columns which are “label” and “features”, train on those to learn how to predict that label and produce a logistic regression model which can then make predictions on every instance of the dataset and in turn a new column called “Predictions” is added.



Train a Model

Current data schema

```
label: Int
text: String

words: Seq[String]

features: Vector

prediction: Int
```

Figure 18: Training model

One important thing to be noticed here is that any of these modules can basically select from any of the previously generated columns and also they can output one or more columns as necessary. Hence in this way, new columns are always appended as and when data flows through the work flow. This is always useful because we can always go back and inspect the intermediate results. All of these data is not necessary to be kept physically around or in memory. This process has been made efficient by DataFrames, which is one more reason for choosing to use them. In DataFrames, if it is needed to inspect a column at some point, it can be materialized as needed.

ML Pipelines

Pipeline can be termed as the combination of the three modules:

1. Tokenizer
2. Hashed Term Frequency
3. Logistic Regression

If we wrap these into a single object, then it can be run again on some new data in exactly the same way with a single call. Also it can let us avoid mistakes.

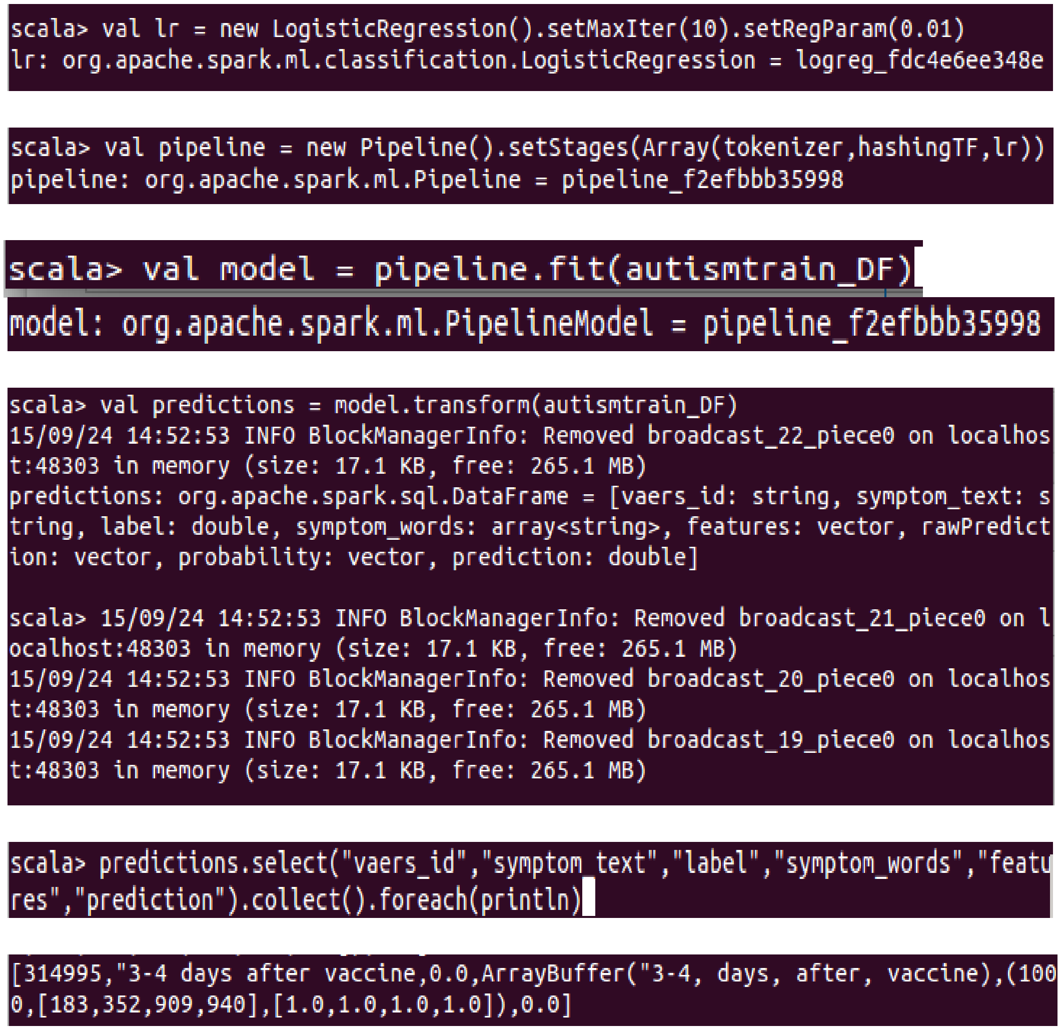
After the pipeline is created, it can be applied on the data to get the predictions.

Code Snippet for training model

```
val lr = new LogisticRegression().setMaxIter(10).setRegParam(0.01)

val pipeline = new Pipeline().setStages(Array(tokenizer,hashingTF,lr))

val model = pipeline.fit(autismtrain_DF)

val predictions = model.transform(autismtrain_DF)
```

predictions.select("vaers_id","symptom_text","label","symptom_words","features","prediction").collect().foreach(println)

Snapshots of code and its results:



```
scala> val lr = new LogisticRegression().setMaxIter(10).setRegParam(0.01)
lr: org.apache.spark.ml.classification.LogisticRegression = logreg_fdc4e6ee348e
```

```
scala> val pipeline = new Pipeline().setStages(Array(tokenizer,hashingTF,lr))
pipeline: org.apache.spark.ml.Pipeline = pipeline_f2efbbb35998
```

```
scala> val model = pipeline.fit(autismtrain_DF)
model: org.apache.spark.ml.PipelineModel = pipeline_f2efbbb35998
```

```
scala> val predictions = model.transform(autismtrain_DF)
15/09/24 14:52:53 INFO BlockManagerInfo: Removed broadcast_22_piece0 on localhos
t:48303 in memory (size: 17.1 KB, free: 265.1 MB)
predictions: org.apache.spark.sql.DataFrame = [vaers_id: string, symptom_text: s
tring, label: double, symptom_words: array<string>, features: vector, rawPredict
ion: vector, probability: vector, prediction: double]

scala> 15/09/24 14:52:53 INFO BlockManagerInfo: Removed broadcast_21_piece0 on l
ocalhost:48303 in memory (size: 17.1 KB, free: 265.1 MB)
15/09/24 14:52:53 INFO BlockManagerInfo: Removed broadcast_20_piece0 on localhos
t:48303 in memory (size: 17.1 KB, free: 265.1 MB)
15/09/24 14:52:53 INFO BlockManagerInfo: Removed broadcast_19_piece0 on localhos
t:48303 in memory (size: 17.1 KB, free: 265.1 MB)
```

```
scala> predictions.select("vaers_id","symptom_text","label","symptom_words","featu
res","prediction").collect().foreach(println)
```

```
[314995,"3-4 days after vaccine,0.0,ArrayBuffer("3-4, days, after, vaccine),(100
0,[183,352,909,940],[1.0,1.0,1.0,1.0]),0.0]
```

Figure 19: Snapshots of code and its results for training model

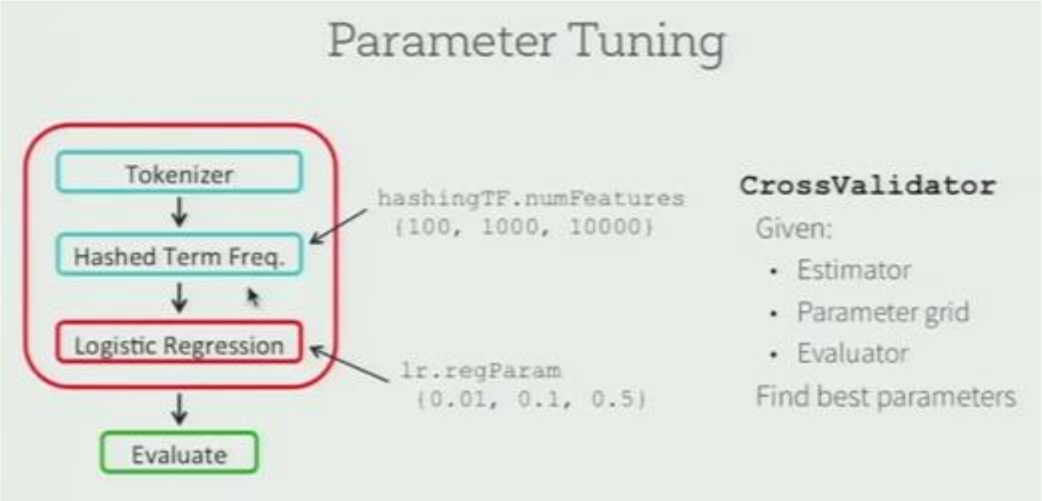
7.2.4 Parameter Tuning and Evaluation

Each of these components may have various parameters. For example, logistic regression takes the regularization parameter and adjusting that can significantly affect performance on new data. This may not be experienced during training. So while tuning, we may want to sweep over a few values of regularization, experiment and see which do the best on held out data and choose a data dependent idea of regularization parameter.

In case of Hashed Term Frequency, we now know that it outputs the fixed length feature vector but we don't know that how long that feature vector should be. You need 100 numbers to represent a text or 10 billion? You may not know and this depends on your dataset. We would like to optimize the performance by experimenting or tuning these values. There is one more important concept worth noticing here. It's Cross Validation. To be more precise we would like to term it as "k-fold cross validation". In "k-fold cross validation", the data set is equally split into k parts and the model is trained on k-1 parts and tested on remaining part. And all the k parts become a part for testing once. So the part which behaved as a test part in first run becomes a training part in next run. So for example, if it is a 5-fold cross validation, then the model is trained 5 times with all the combinations of 4 parts and all the 5 parts behave as a test part once in each run. In this way, model calculates the accuracy in each run and then it may take the average of all the accuracies or the best accuracy. For managing parameters, CrossValidator has been provided in this API which takes an Estimator (in this case the pipeline), Parameter

Grid and an Evaluator (letting you compare the models that you have learnt). It then automatically finds the best parameters.

Evaluation is a very important stage because we want to evaluate how well model has performed. It uses “Label” column which is the true label from the dataset as well as “Prediction” which is the predicted label. It can compare them and tell how well model has performed. For evaluation “area under ROC” has been considered. More the value of “area under ROC” is near 1.00, more is the accuracy. For Logistic Regression Model, area under ROC for training dataset is 0.9994 and for test dataset is 0.8996.



Parameter Tuning

Tokenizer

Hashed Term Freq.

Logistic Regression

Evaluate

hashingTF.numFeatures
{100, 1000, 10000}

lr.regParam
{0.01, 0.1, 0.5}

CrossValidator

Given:

- Estimator
- Parameter grid
- Evaluator

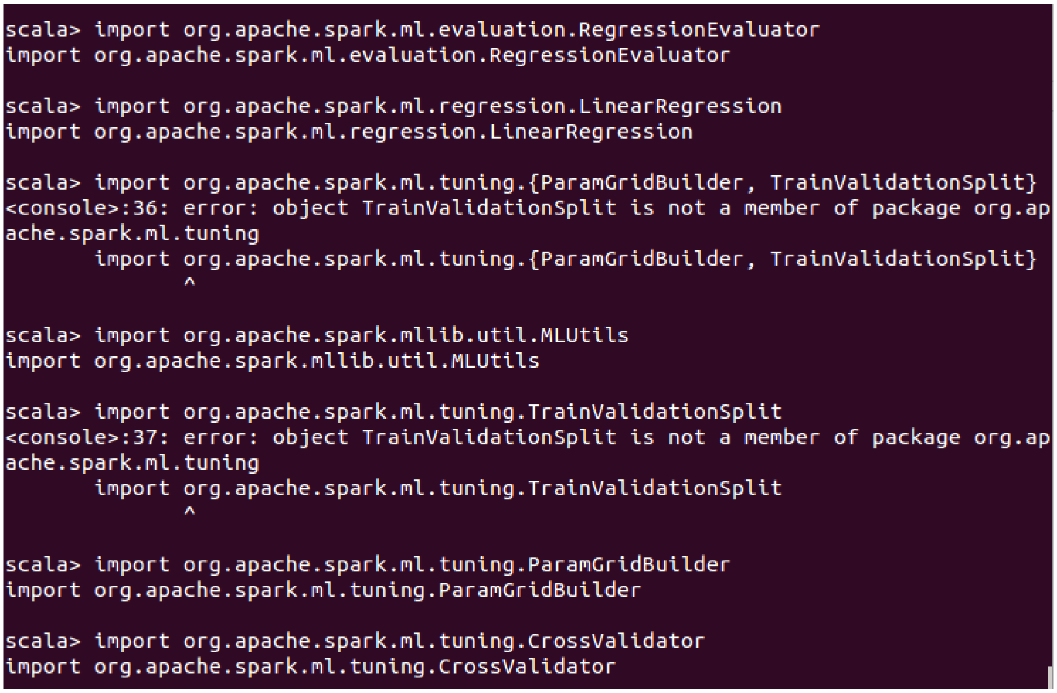
Find best parameters

Figure 20: Parameter Tuning

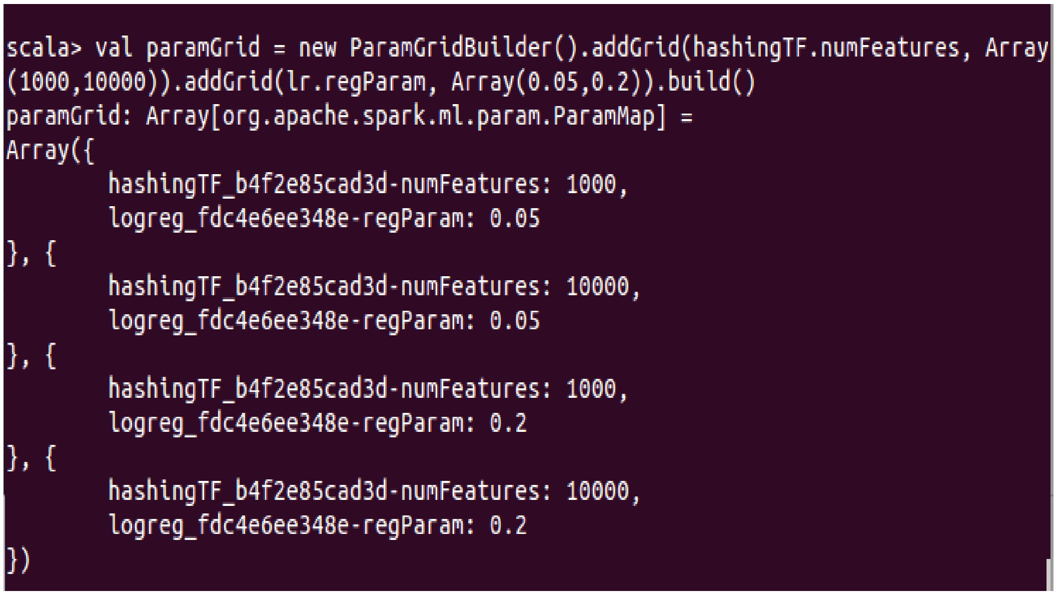
Code Snippet for parameter tuning and evaluation

```
import org.apache.spark.ml.evaluation.RegressionEvaluator
import org.apache.spark.ml.regression.LinearRegression
import org.apache.spark.mllib.util.MLUtils
import org.apache.spark.ml.tuning.ParamGridBuilder
import org.apache.spark.ml.tuning.CrossValidator
import org.apache.spark.ml.evaluation.BinaryClassificationEvaluator
val paramGrid = new
ParamGridBuilder().addGrid(hashingTF.numFeatures,
Array(1000,10000)).addGrid(lr.regParam, Array(0.05,0.2)).build()
val crossval = new
CrossValidator().setEstimator(pipeline).setEvaluator(new
BinaryClassificationEvaluator).setEstimatorParamMaps(paramGrid).setNumFolds(2)
val cvModel = crossval.fit(autismtrain_DF)
val evaluator =
newBinaryClassificationEvaluator().setMetricName(“areaUnderROC”)
evaluator.evaluate(cvModel.transform(autismtrain_DF))
```

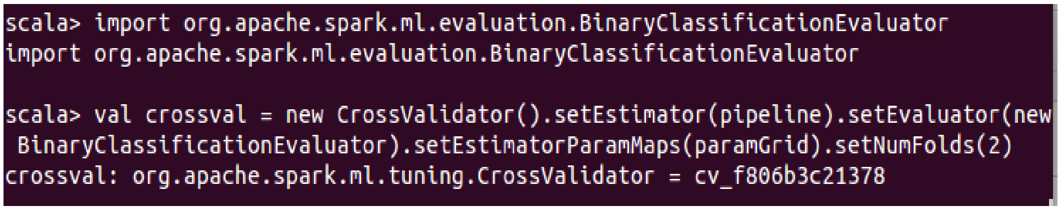
Snapshots of code and its results:



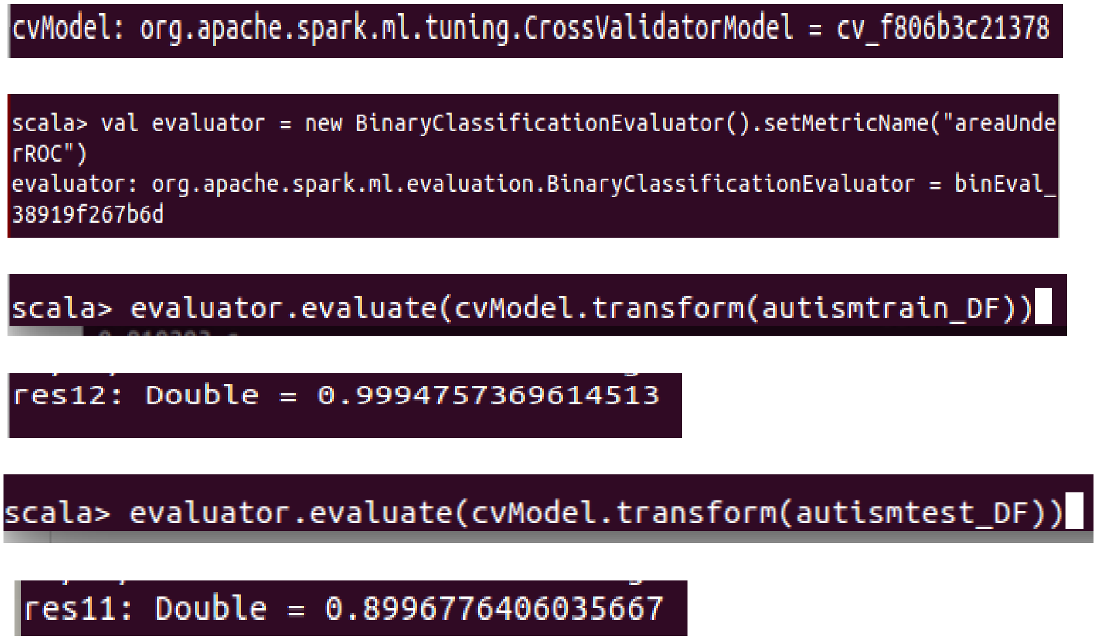
```
scala> import org.apache.spark.ml.evaluation.RegressionEvaluator
import org.apache.spark.ml.evaluation.RegressionEvaluator

scala> import org.apache.spark.ml.regression.LinearRegression
import org.apache.spark.ml.regression.LinearRegression

scala> import org.apache.spark.ml.tuning.{ParamGridBuilder, TrainValidationSplit}
<console>:36: error: object TrainValidationSplit is not a member of package org.ap
ache.spark.ml.tuning
       import org.apache.spark.ml.tuning.{ParamGridBuilder, TrainValidationSplit}
              ^

scala> import org.apache.spark.mllib.util.MLUtils
import org.apache.spark.mllib.util.MLUtils

scala> import org.apache.spark.ml.tuning.TrainValidationSplit
<console>:37: error: object TrainValidationSplit is not a member of package org.ap
ache.spark.ml.tuning
       import org.apache.spark.ml.tuning.TrainValidationSplit
              ^

scala> import org.apache.spark.ml.tuning.ParamGridBuilder
import org.apache.spark.ml.tuning.ParamGridBuilder

scala> import org.apache.spark.ml.tuning.CrossValidator
import org.apache.spark.ml.tuning.CrossValidator
```



```
scala> val paramGrid = new ParamGridBuilder().addGrid(hashingTF.numFeatures, Array
(1000,10000)).addGrid(lr.regParam, Array(0.05,0.2)).build()
paramGrid: Array[org.apache.spark.ml.param.ParamMap] =
Array({
        hashingTF_b4f2e85cad3d-numFeatures: 1000,
        logreg_fdc4e6ee348e-regParam: 0.05
}, {
        hashingTF_b4f2e85cad3d-numFeatures: 10000,
        logreg_fdc4e6ee348e-regParam: 0.05
}, {
        hashingTF_b4f2e85cad3d-numFeatures: 1000,
        logreg_fdc4e6ee348e-regParam: 0.2
}, {
        hashingTF_b4f2e85cad3d-numFeatures: 10000,
        logreg_fdc4e6ee348e-regParam: 0.2
})
```



```
scala> import org.apache.spark.ml.evaluation.BinaryClassificationEvaluator
import org.apache.spark.ml.evaluation.BinaryClassificationEvaluator

scala> val crossval = new CrossValidator().setEstimator(pipeline).setEvaluator(new
 BinaryClassificationEvaluator).setEstimatorParamMaps(paramGrid).setNumFolds(2)
crossval: org.apache.spark.ml.tuning.CrossValidator = cv_f806b3c21378
```



```
cvModel: org.apache.spark.ml.tuning.CrossValidatorModel = cv_f806b3c21378
```

```
scala> val evaluator = new BinaryClassificationEvaluator().setMetricName("areaUnde
rROC")
evaluator: org.apache.spark.ml.evaluation.BinaryClassificationEvaluator = binEval_
38919f267b6d
```

```
scala> evaluator.evaluate(cvModel.transform(autismtrain_DF))
```

```
res12: Double = 0.9994757369614513
```

```
scala> evaluator.evaluate(cvModel.transform(autismtest_DF))
```

```
res11: Double = 0.8996776406035667
```

Figure 21: Snapshots of code and its results for parameter tuning and evaluation

8. Implementation using Apache Spark over Amazon EMR

Amazon EMR (Elastic MapReduce) provides the managed framework that distributes the computation of data over multiple Amazon EC2 (Elastic Compute Cloud) instances.

To get started, data is loaded into Amazon S3 (Simple Storage Service). Then Amazon EMR cluster is launched and cluster starts processing the data. When the job is completed, output can be retrieved from Amazon S3. Cluster can be left running if more data processing is needed. The data in Amazon S3 can be accessed by multiple EMR clusters. Clusters can also be terminated when they are not needed anymore.

With Amazon EMR, a variety of powerful applications and frameworks can be used and Apache Spark is one of those.

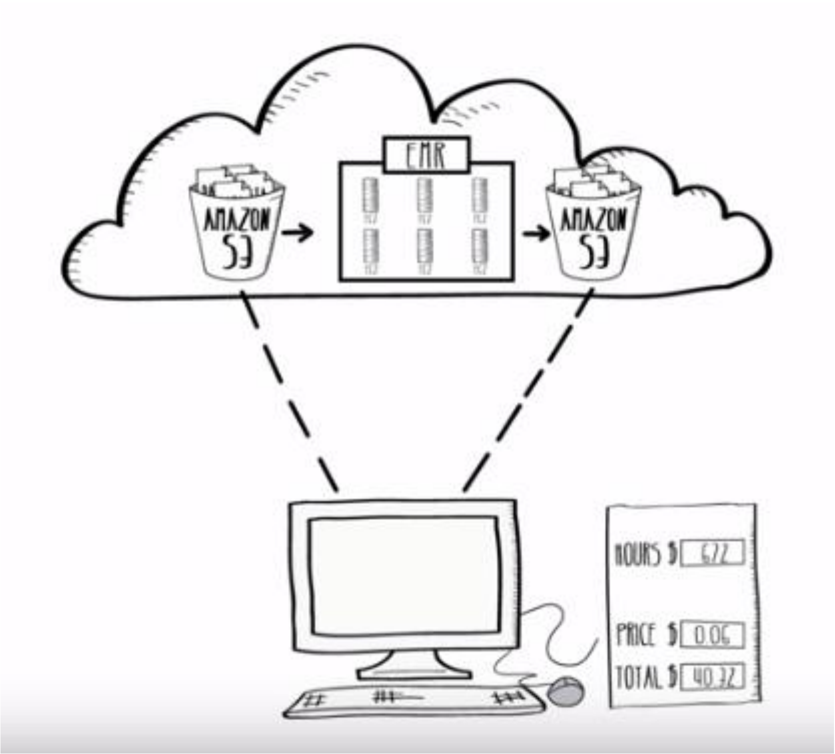


Figure 22: Amazon EMR Workflow

AWS (Amazon Web Services) Account

AWS account has to be created to leverage the above mentioned services. While creating a new user account on AWS, “Access key id”, “Secret access key” and “Password” is provided which are useful in further processes.

Installing AWS CLI (AWS Command Line Interface)

To work on CLI, AWS CLI has to be installed using the following command.

$ pip install awscli

$ sudo pip install --upgrade awscli

Configuring AWS CLI

Following command is the fastest way to setup AWS CLI installation:

```
$ aws configure
```

By running this command, following details have to be entered:

AWS Access Key ID, AWS Secret Access Key, Default region name (us-west-2), Default output format (json)

The CLI stores credentials specified with “aws configure” in a local file named “credentials” in a folder named .aws in your home directory.

Create a cluster with Spark

First there is a one-time step required to create the default roles necessary for creation of cluster:

```
$ emr create-default-roles
```

Cluster can be created with the following command:

```
$ aws emr create-cluster –name “SparkCluster” --ami-version 3.9 --applications Name=Spark --ec2-attributes KeyName=myKey --instance-type m3.xlarge --instance-count 3 --use-default-roles
```

Create Amazon S3 bucket

Amazon EMR can use Amazon S3 to store input data, output data or log files. In the “Create a bucket wizard”, bucket can be created by providing “Bucket name”, “Region”. After this, folder can be created in the bucket and the data which we want to use can be uploaded.

Create EMR Cluster

There is .pem file containing key value pair which was created. That file is to be copied into .aws folder and its permissions have to be changed:

```
$ chmod 400 mykeypair.pem
```

Connect to master node using SSH

To connect to the master node using SSH, public DNS name of the master node and Amazon EC2 key pair private key are needed.

```
$ ssh hadoop@ec2-###-##-##-###.us-west-2.compute.amazonaws.com -i ~/mykeypair.pem
```

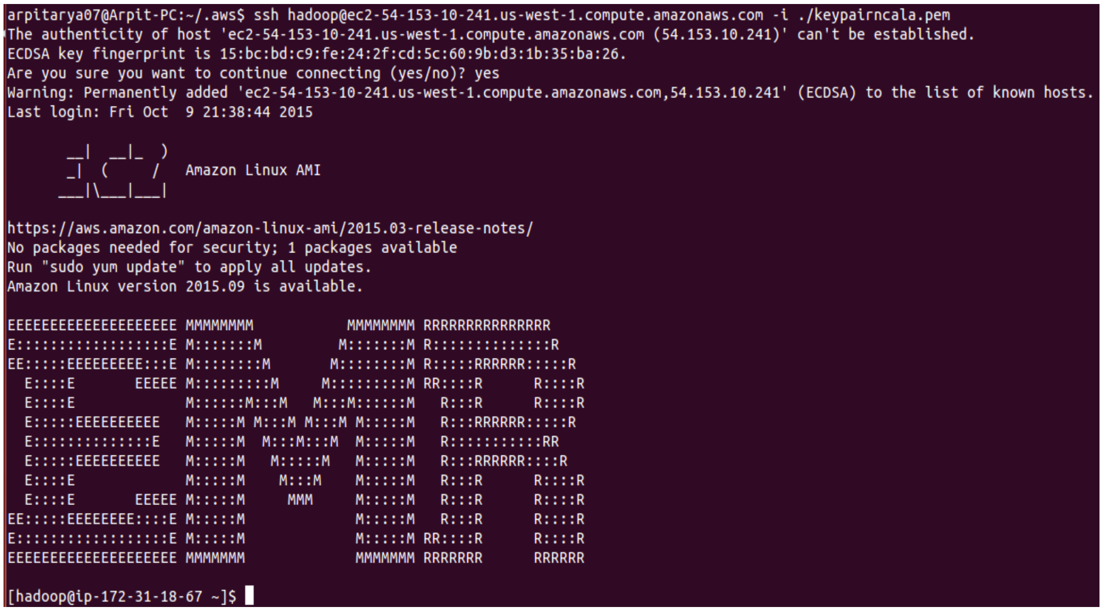
hadoop@ec2-###-##-##-###.us-west-2.compute.amazonaws.com:

Master public DNS name

~/mykeypair.pem:

Location and file name of .pem file

Then following screen will appear which shows that we are connected:



```
arpitarya07@Arpit-PC:~/.aws$ ssh hadoop@ec2-54-153-10-241.us-west-1.compute.amazonaws.com -i ./keypairncala.pem
The authenticity of host 'ec2-54-153-10-241.us-west-1.compute.amazonaws.com (54.153.10.241)' can't be established.
ECDSA key fingerprint is 15:bc:bd:c9:fe:24:2f:cd:5c:60:9b:d3:1b:35:ba:26.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ec2-54-153-10-241.us-west-1.compute.amazonaws.com,54.153.10.241' (ECDSA) to the list of known hosts.
Last login: Fri Oct  9 21:38:44 2015

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-ami/2015.03-release-notes/
No packages needed for security; 1 packages available
Run "sudo yum update" to apply all updates.
Amazon Linux version 2015.09 is available.

EEEEEEEEEEEEEEEEEEEE MMMMMMMM           MMMMMMMM RRRRRRRRRRRRRRR
E::::::::::::::::::E M:::::::M         M:::::::M R::::::::::::::R
EE:::::EEEEEEEEE:::E M::::::::M       M::::::::M R:::::RRRRRR:::::R
  E::::E       EEEEE M:::::::::M     M:::::::::M RR::::R      R::::R
  E::::E             M::::::M:::M   M:::M::::::M   R:::R      R::::R
  E:::::EEEEEEEEEE   M:::::M M:::M M:::M M:::::M   R:::RRRRRR:::::R
  E::::::::::::::E   M:::::M  M:::M:::M  M:::::M   R:::::::::::RR
  E:::::EEEEEEEEEE   M:::::M   M:::::M   M:::::M   R:::RRRRRR::::R
  E::::E             M:::::M    M:::M    M:::::M   R:::R      R::::R
  E::::E       EEEEE M:::::M     MMM     M:::::M   R:::R      R::::R
EE:::::EEEEEEEE::::E M:::::M             M:::::M   R:::R      R::::R
E::::::::::::::::::E M:::::M             M:::::M RR::::R      R::::R
EEEEEEEEEEEEEEEEEEEE MMMMMMM             MMMMMMM RRRRRRR      RRRRRR

[hadoop@ip-172-31-18-67 ~]$
```

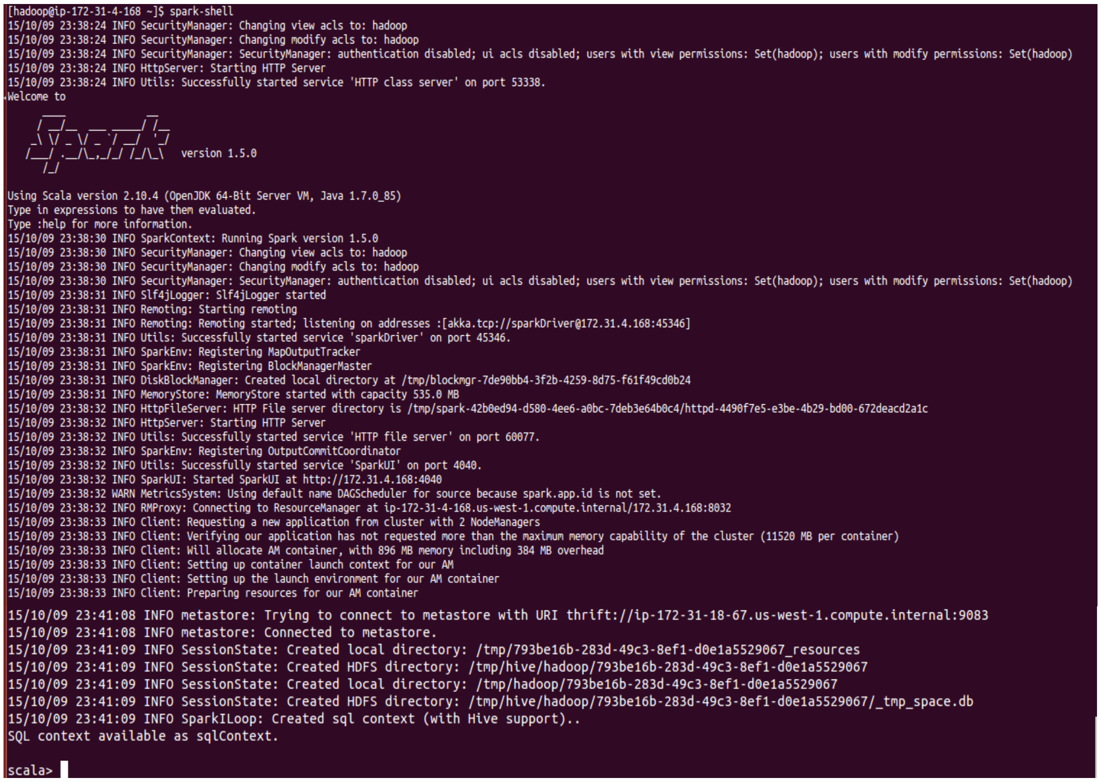
Figure 23: Screen after connecting to Master Node

Access the spark shell on Master Node

Run:

$ spark-shell

Then following screen will appear:



```
[hadoop@ip-172-31-4-168 ~]$ spark-shell
15/10/09 23:38:24 INFO SecurityManager: Changing view acls to: hadoop
15/10/09 23:38:24 INFO SecurityManager: Changing modify acls to: hadoop
15/10/09 23:38:24 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: Set(hadoop); users with modify permissions: Set(hadoop)
15/10/09 23:38:24 INFO HttpServer: Starting HTTP Server
15/10/09 23:38:24 INFO Utils: Successfully started service 'HTTP class server' on port 53338.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 1.5.0
      /_/

Using Scala version 2.10.4 (OpenJDK 64-Bit Server VM, Java 1.7.0_85)
Type in expressions to have them evaluated.
Type :help for more information.
15/10/09 23:38:30 INFO SparkContext: Running Spark version 1.5.0
15/10/09 23:38:30 INFO SecurityManager: Changing view acls to: hadoop
15/10/09 23:38:30 INFO SecurityManager: Changing modify acls to: hadoop
15/10/09 23:38:30 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: Set(hadoop); users with modify permissions: Set(hadoop)
15/10/09 23:38:31 INFO Slf4jLogger: Slf4jLogger started
15/10/09 23:38:31 INFO Remoting: Starting remoting
15/10/09 23:38:31 INFO Remoting: Remoting started; listening on addresses :[akka.tcp://sparkDriver@172.31.4.168:45346]
15/10/09 23:38:31 INFO Utils: Successfully started service 'sparkDriver' on port 45346.
15/10/09 23:38:31 INFO SparkEnv: Registering MapOutputTracker
15/10/09 23:38:31 INFO SparkEnv: Registering BlockManagerMaster
15/10/09 23:38:31 INFO DiskBlockManager: Created local directory at /tmp/blockmgr-7de90bb4-3f2b-4259-8d75-f61f49cd0b24
15/10/09 23:38:31 INFO MemoryStore: MemoryStore started with capacity 535.0 MB
15/10/09 23:38:32 INFO HttpFileServer: HTTP File server directory is /tmp/spark-42b0ed94-d580-4ee6-a0bc-7deb3e64b0c4/httpd-4490f7e5-e3be-4b29-bd00-672deacd2a1c
15/10/09 23:38:32 INFO HttpServer: Starting HTTP Server
15/10/09 23:38:32 INFO Utils: Successfully started service 'HTTP file server' on port 60077.
15/10/09 23:38:32 INFO SparkEnv: Registering OutputCommitCoordinator
15/10/09 23:38:32 INFO Utils: Successfully started service 'SparkUI' on port 4040.
15/10/09 23:38:32 INFO SparkUI: Started SparkUI at http://172.31.4.168:4040
15/10/09 23:38:32 WARN MetricsSystem: Using default name DAGScheduler for source because spark.app.id is not set.
15/10/09 23:38:32 INFO RMProxy: Connecting to ResourceManager at ip-172-31-4-168.us-west-1.compute.internal/172.31.4.168:8032
15/10/09 23:38:33 INFO Client: Requesting a new application from cluster with 2 NodeManagers
15/10/09 23:38:33 INFO Client: Verifying our application has not requested more than the maximum memory capability of the cluster (11520 MB per container)
15/10/09 23:38:33 INFO Client: Will allocate AM container, with 896 MB memory including 384 MB overhead
15/10/09 23:38:33 INFO Client: Setting up container launch context for our AM
15/10/09 23:38:33 INFO Client: Setting up the launch environment for our AM container
15/10/09 23:38:33 INFO Client: Preparing resources for our AM container
15/10/09 23:41:08 INFO metastore: Trying to connect to metastore with URI thrift://ip-172-31-18-67.us-west-1.compute.internal:9083
15/10/09 23:41:08 INFO metastore: Connected to metastore.
15/10/09 23:41:09 INFO SessionState: Created local directory: /tmp/793be16b-283d-49c3-8ef1-d0e1a5529067_resources
15/10/09 23:41:09 INFO SessionState: Created HDFS directory: /tmp/hive/hadoop/793be16b-283d-49c3-8ef1-d0e1a5529067
15/10/09 23:41:09 INFO SessionState: Created local directory: /tmp/hadoop/793be16b-283d-49c3-8ef1-d0e1a5529067
15/10/09 23:41:09 INFO SessionState: Created HDFS directory: /tmp/hive/hadoop/793be16b-283d-49c3-8ef1-d0e1a5529067/_tmp_space.db
15/10/09 23:41:09 INFO SparkILoop: Created sql context (with Hive support)..
SQL context available as sqlContext.

scala> 
```

Figure 24: Running spark-shell over EMR

Machine learning programming can be performed using Scala. Data can be utilized from Amazon S3.

9. Conclusion and Future Work

The project work proposes a new aspect and approach to fasten the procedure of detecting and diagnosing a complex disease called Autism. This approach uses Machine Learning or Predictive Modeling techniques. If this approach gets to be adapted in real life scenario, then doctors can get immensely assisted by technology towards diagnosing this disease efficiently, accurately and in lesser time.

In this project, two open source frameworks were used: Weka and Apache Spark. Weka is good to understand the concepts of machine learning as it is simpler than Spark. But when it comes to complex, large-scale data processing and predictive modeling, Spark is faster. Also for cluster computing, Spark becomes quite compatible and scalable.

In future, several different kinds of datasets can be consumed and utilized towards building data models and perform predictive modeling. The major advantage of such technique is that it has immense potential to get utilized in other areas of medical science or any other field.

References

[1] ‘Testing the accuracy of an observation-based classifier for rapid detection of autism risk’, Nature Publishing Group (accessed Nov, 2015)

[2] Chaturvedi, V., ‘The potential of accelerating early detection of Autism through content analysis of YouTube Videos’, Public Library of Science (accessed Nov, 2015)

[3] Rogers, S., ‘Use of artificial intelligence to shorten the behavioral diagnosis of Autism’, Public Library of Science (accessed Nov, 2015)

[4] ‘Autism, Wiki’, https://en.wikipedia.org/wiki/Autism (accessed Nov, 2015)

[5] ‘Disease Information’, http://answers.google.com/answers/threadview?id=492472 (accessed Nov, 2015)

[6] ‘Vaccine Adverse Event Reporting System’ (VAERS), https://vaers.hhs.gov/data/data (accessed Nov, 2015)

[7] ‘Weka – The University of Waikato, New Zealand’, http://www.cs.waikato.ac.nz/ml/weka/ (accessed July, 2015)

[8] ‘Weka Text Classification’, https://www.youtube.com/watch?v=IY29uC4uem8 (accessed July, 2015)

[9] ‘Apache Spark Documentation’, https://spark.apache.org/docs/latest/ (accessed Nov, 2015)

[10] ‘Apache Spark, Wiki’, https://en.wikipedia.org/wiki/Apache_Spark (accessed Nov, 2015)

[11] ‘Building, Debugging and Tuning Spark Machine Learning Pipelines’, https://www.youtube.com/watch?v=OednhGRp938&list=PLIxzgeMkSrQ8Pq30lCaHmYA7r_Qprg6-H (accessed Nov, 2015)

[12] McDonald, C. (2015), ‘Using Apache Spark DataFrames for processing of Tabular Data’, 2015

[13] ‘Amazon EMR Documentation’, https://aws.amazon.com/elasticmapreduce/ (accessed Nov, 2015)

[14] ‘Amazon S3 Documentation’, https://aws.amazon.com/s3/ (accessed Nov, 2015)

[15] ‘Amazon EC2 Documentation’, https://aws.amazon.com/ec2/ (accessed Nov, 2015)