REPUBLIC OF TURKEY
YILDIZ TECHNICAL UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

ADAPTIVE LEARNING BASED TRACING TOOL FOR WEATHER RESEARCH AND FORECASTING SOFTWARE

14011085 – Erkan Karabulut

SENIOR PROJECT

Advisor
Assoc. Prof. Dr. Mehmet S. Aktaş

January, 2019

ACKNOWLEDGEMENTS

I would like to thank to my consultant Assoc. Prof. Dr. Mehmet S. Aktaş and my project manager Alper Tufek due to their assistance and contribution to this project. It was great to have someone who has had experience with same topics to offer guidance and direction.

Erkan Karabulut

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

WRF	Weather Research and Forecast
PROV-O	Provenance Ontology
PROV-DM	Provenance Data Model
W3C	World Wide Web Consortium
ML	Machine Learning
API	Application Programming Interface
OWL2	Web Ontology Language 2

LIST OF FIGURES

ABSTRACT

Adaptive Learning Based Tracing Tool For Weather Research and Forecasting Software

Erkan Karabulut

Department of Computer Engineering
Senior Project

Advisor: Assoc. Prof. Dr. Mehmet S. Aktaş

This study aims to create a tracing tool software for Weather Research and Forecasting Model[1] based on machine learning algorithms and data provenance[5]. The tool provides effectively tracing the internal processes of the model without stopping the execution of it by reading the log files. Hence, some unwanted situations, for example faulty or non-accurate input parameters which will then cause the useless forecasting results, will be detected as soon as they appeared.

Weather Research and Forecasting Model is a well-known and commonly used system in weather forecast domain. Main issue about the model is that it can not be stopped until it finishes it's internal processes. Since the model takes long time to finish it's job, knowing the current situation about the model while it works is crucial.

The WRF model produces very detailed log records. Those records made possible for us to create a tracing tool in order to trace the internal processes of the model. Our tracing tool reads the log files of the WRF model and decides whether or not a certain row includes provenance information. After selecting the rows with provenance information, the tool decides which provenance relation that the row has.

For the purpose of selecting the rows with provenance information and deciding the provenance relation of it, our tracing tool provides Naive Bayes, Logistic Regression, Decision Tree, Random Forests, Multilayer Perceptron and OneVsAll algorithms. All of the algorithms are implemented with using Apache Spark's Machine Learning library. Also 3 of the algorithms which are Decision Tree, Naive Bayes, Logistic Regression and OneVsAll classification algorithms are implemented manually. All of the algorithms.

After deciding the provenance information about the log files, the tool creates a provenance data file which shows the provenance actors and their relations[6]. By using this provenance data file we can easily visualize the content of the provenance file as nodes and edges thanks to our provenance data visualization tool which I previously developed in the scope of my computer project[2].

Keywords: Data provenance, provenance ontology, machine learning, logistic regression, random forests, decision tree, multilayer perceptron, naive bayes, onevsall classifier, weather research and forecasting model

ÖZET

Hava Durumu Araştırma ve Tahmin Yazılımı için Adaptif Öğrenme Yeteneğine Dayalı İzleme Aracı Geliştirilmesi

Erkan Karabulut

Bilgisayar Mühendisliği Bölümü
Bitirme Projesi

Danışman: Doç. Dr. Mehmet S. Aktaş

Bu çalışma, Hava Araştırmaları ve Tahmin Modeli (WRF)[1] için makine öğrenmesi teknikleri ve veri provenansına[5] dayalı bir izleme aracı geliştirmeyi hedeflemiştir. Araç en efektif şekilde hava tahmin modelinin çalışmasını durdurmadan modele ait log dosyalarını okumak suretiyle modelin takibini mümkün kılmaktadır. Böylece modele hatalı yada eksik parametre girilmesi gibi modelin çalışması sonucunda faydasız bilgi üretilmesine sebep olabilecek bazı istenmeyen durumların tespitini modelin durdurulmasına gerek kalmadan en kısa süre içerisinde tespit edebilmeye olanak sağlamıştır.

Hava Araştırmaları ve Tahmin Modeli hava tahminleri alanında iyi bilinen ve çokça kullanılan bir sistemdir. Bu sistemle ilgili bu çalışmanın konusu olan problem modelin çalıştığında işlemlerini bitirene kadar duraklatılamamasıdır. Modelin çalışmasını tamamlamasıda uzun zaman aldığından, modelin herhangi bir andaki durumunu bilmek kritik bir öneme sahiptir.

Model olabildiğince detaylı şekilde log kayıtları üretmektedir. Bu kayıtlar bize modelin takibini her an yapabilecek bir izleme aracı geliştirmemize olanak sağlamıştır. İzleme aracı hava tahmin modelinin log kayıtlarını okur ve her bir satırın provenans bilgisi taşıyıp taşımadığına karar vermektedir. Provenans bilgisi taşıyan satırları seçtikten sonra bu satırlardaki bilginin ne tür bir provenans ilişkisini temsil ettiğine karar verir.

Provenans bilgisi içeren log satırlarını seçmek ve hangi çeşit provenans ilişkisine sahip olduğuna karar vermek için izleme aracımız Naive Bayes, Lojistik Regresyon, Karar Ağacı, Rassal Ormanlar, Çok Katmanlı Algılayıcı(Perceptron) ve OneVsAll algoritmalarını kullanmaktadır. Bütün bu algoritmalar Apache Spark'ın sağlamış olduğu makine öğrenmesi kütüphanesi kullanılarak gerçeklenmiştir. Bunun yanında Naive Bayes, Lojistik Regresyon ve OneVsAll algoritması olmak üzere algoritmalardan 3'üne ait kendi gerçeklemelerimiz de izleme aracı içerisinde yer almaktadır.

Hava tahmin modelinin provenans bilgisi içeren log satırlarına karar verdikten sonra izleme aracımız bu provenans aktörlerini ve aralarındaki ilişkiyi gösteren bir provenance dosyası oluşturmaktadır[6]. Bu provenans dosyasını kullanarak bilgisayar projesi kapsamında gerçeklemiş olduğum bir önceki çalışmamdaki görselleştirme aracı sayesinde dosyanın içeriğini kolay bir şekilde düğümler ve kenarlar olarak görselleştirebilmekteyiz.

Anahtar Kelimeler: Veri provenansı, provenans ontoloji, makine öğrenmesi, lojistik regresyon, rassal ormanlar, karar ağaçları, çok katmanlı algılayıcı, naive bayes, onevsall sınıflayıcı, hava araştırma ve tahmin modeli

1
Introduction

This study aims to create a software tool that can trace the Weather Research and Forecast Model [1] software and help to decide whether the model works correctly or not by using provenance data model. A Brief introductions of WRF Model and Provenance Data Model are located in section 1.1 and section 1.2 respectively. Basically WRF model is a next-generation mesoscale numerical weather prediction system designed for both atmospheric research and operational forecasting applications.

Because of its internal functionality, when the WRF model is started it can not be stopped until it completes its job. The point in here is that what if something went wrong during the model's execution. Thus the faulty situations will not be noticed until all the processing and the calculations are done.

Thanks to our work, detecting failures and errors is possible while the model is working. To notice these undesired situations we will use WRF's log files. These log files include all the method information that belongs to WRF software. From these method information we will produce a data provenance based workflow which only includes the methods that are related with the WRF's actual jobs.

In order to decide whether a method is related with the workflow or not, we used Machine Learning techniques including Logistic Regression, Naive Bayes Algorithm, Decision Tree Algorithm, Random Forest Algorithm, Multilayer Perceptron Classification and OneVsAll algorithms. Beside deciding a log record is related with the workflow or not, this study also aims to find the provenance relations between actors in the workflow. The definition of provenance, basic provenance structures and relations are given in Section 1.2. After creating workflows, we visualized them with a data visualization plugin which we previously designed and developed [2] on a data visualization software named Cytoscape in Software Quality Research and Development Lab, Yildiz Technical University. As a result of these steps we can effectively and adaptively trace WRF Model in real time.

The data we used in our tests and experiments is from Karma [3] and Komadu [4] tools which are both data capture and visualization systems for scientific data provenance.

1.1 Weather Research and Forecasting Model

Weather Research and Forecasting Model is a well-known and commonly used system in weather forecast domain. It has a large worldwide community of registered users (a cumulative total of over 39,000 in over 160 countries), and there are workshops and tutorials on it each year.

> The Weather Research and Forecasting (WRF) Model is a next-generation mesoscale numerical weather prediction system designed for both atmospheric research and operational forecasting applications. It features two dynamical cores, a data assimilation system, and a software architecture supporting parallel computation and system extensibility. The model serves a wide range of meteorological applications across scales from tens of meters to thousands of kilometers.

1.2 Provenance Data Model

According to the definition below from W3C PROV-DM page, in provenance domain a data can only represent an entity, an agent or an activity. And the provenance data model is developed in order to distinguish core structures, forming the essence of provenance information, from extended structures catering for more specific uses of provenance.

> Provenance is information about entities, activities, and people involved in producing a piece of data or thing, which can be used to form assessments about its quality, reliability or trustworthiness. [5]

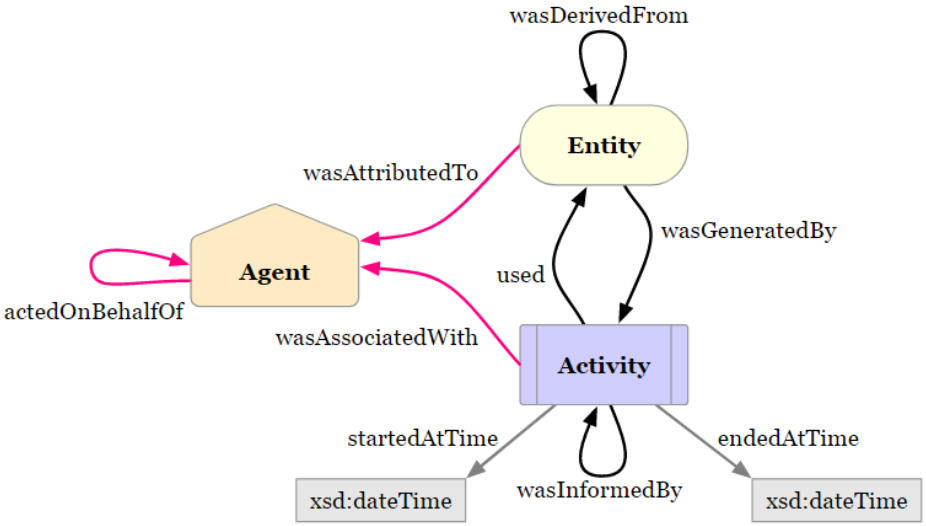

Figure 1.1 Data types and the basic relations between them in PROV-DM and PROV-O.

The language that expresses the provenance data model is the Provenance Ontology [6] (commonly known as PROV-O) which uses OWL2 Web Ontology Language. PROV-O includes different types of relations between the data types agent, activity and entity. These relations represent set of classes, properties and restrictions about the data. Figure 1.1 shows the basic types of relations.

2
Literature Summary

In the early of this year, there has been made a study about tracing WRF model. [7] It also uses the WRF Model's log files to trace and detect failures. It needs a keywords rule configuration file in order to extract the relevant lines from the log file. To be able to include all the lines containing provenance information, the necessary keywords must be added to the keywords database manually. In that study, there was no adaptive learning process to define and detect significant information from the log files. Thus, it requires someone to examine the log files and extract keywords and then put them statically to the keywords rule configuration file.

We propose an adaptive learning based tracing tool for WRF model to extract the method information from the log files and then use this information to detect failures. After filtering the related lines from the log files, we convert this information to provenance data.

The provenance data collected from the WRF model, can be visualized effectively by our visualization tool which is a plugin for the data visualization, integration and analysis tool Cytoscape. [8] Our visualization software is capable of sorting, filtering, highlighting, comparing and clustering the provenance data.

3
Feasibility

In this section, the feasibility of the project will be covered. This project will consist of only software pieces. Therefore, only the software feasibility will be covered in the technical feasibility section.

3.1 Technical Feasibility

Adaptive learning based tracing tool for the Weather Research and Forecast model will only be a software program. The tool is going to use log files of WRF model. The log file of WRF model includes records about the execution details of the WRF model. An example WRF model log file is given in Fig. 3.1.

```
DYNAMICS OPTION: Eulerian Mass Coordinate
   alloc_space_field: domain            1 ,             134701856  bytes allocated
  wrf: calling model_to_grid_config_rec
  wrf: calling set_scalar_indices_from_config
  wrf: calling init_wrfio
  Entering ext_gr1_ioinit
   DEBUG wrf_timetoa():  returning with str = [2018-04-05_00:00:00]
   DEBUG wrf_timetoa():  returning with str = [2018-04-05_00:00:00]
   DEBUG wrf_timetoa():  returning with str = [2018-04-05_12:00:00]
   DEBUG wrf_timeinttoa():  returning with str = [0000000000_000:003:000]
 DEBUG setup_timekeeping():  clock after creation,  clock start time = 2018-04-05_00:00:00
 DEBUG setup_timekeeping():  clock after creation,  clock current time = 2018-04-05_00:00:00
 DEBUG setup_timekeeping():  clock after creation,  clock stop time = 2018-04-05_12:00:00
 DEBUG setup_timekeeping():  clock after creation,  clock time step = 0000000000_000:003:000
```

Figure 3.1 Weather Research and Forecast Log File Sample

The most important thing for us to see is the method calls and other components which shows the actions of the modules of WRF.

As an example, in the third line of the log file we can see the `wrf` string, which is the main module, calls another module named `model_to_grid_config_rec`. These types of log lines which show the communication between the WRF components will be our input to produce log files after deciding that the line is relevant with the internal structure of WRF with using machine learning.

The programming language that we will be using to implement the machine learning algorithms is Java. Java has the one of the biggest community behind it comparing to other programming languages which allows us to implement machine learning algorithms and also there is so much resource written or visual about implementing machine learning algorithms in Java.

One of the most popular, easy to use, effective and efficient way to implement the algorithms in Java is with using Apache Spark MLLib library. The biggest alternative of Spark is Hadoop which is developed by the same people who developed the Spark. Spark has been found to run 100 times faster in-memory, and 10 times faster on disk. Because of these reasons, we choose to use Apache Spark to implement Machine Learning algorithms.

To produce data provenance from log files of WRF, we will be using an open source provenance collection tool named Komadu. Komadu software uses so much software component inside of it including Apache Tomcat, MySQL Server, Apache XML-Beans, Apache Axis 2, Rabbit MQ and many more.

Last version of Komadu runs properly on Ubuntu 12.04. In order to make it easy to implement and prevent from discrepancy between our software and Komadu, we decided to implement the system in Ubuntu. And lastly, we will use IntelliJ IDEA to program in java because of its countless useful and beneficial tools and plugins.

3.2 Time Feasibility

This project will be implemented in 14 weeks. Implementation of the system is consists of 9 steps. Each step affects other steps as less as it can be. Gantt diagram of the system which shows these steps and start and end dates to complete the steps is given in Fig. 3.2.

- Literature and feasibility research and system analysis
- Preparing first report
- Setting up WRF model software within a VM
- Setting up Komadu tool within a VM
- Implementing ML part of the tracing tool
- Implementing row data to provenance data conversion
- Preparing second report
- Integrating WRF model, tracing tool, Komadu and Cytoscape
- Preparing final project report

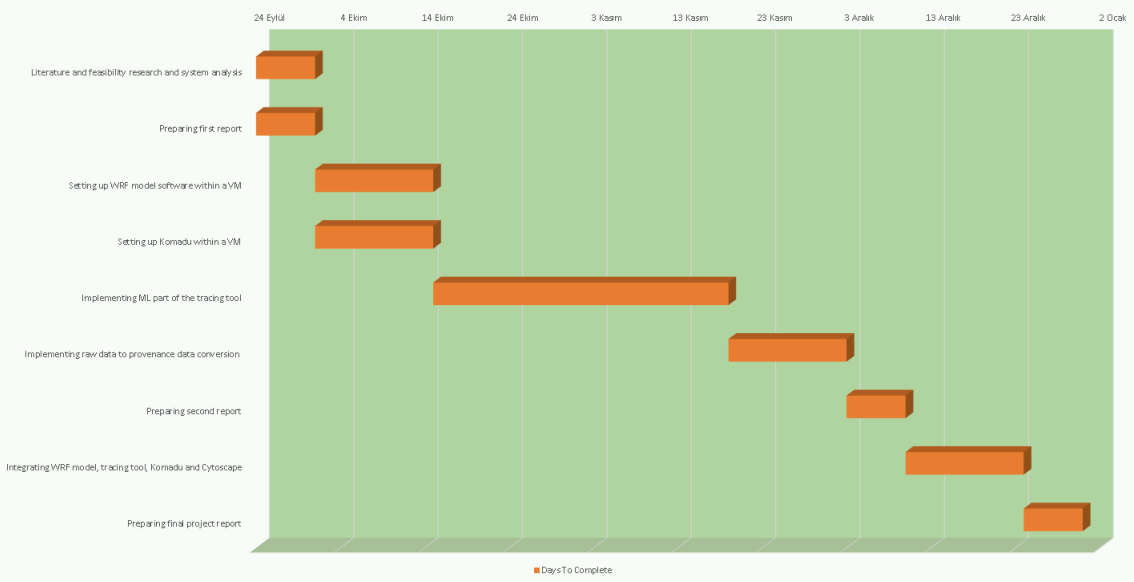

Figure 3.2 Data types and the basic relations between them in PROV-DM and PROV-O.

4
System Analysis

Adaptive learning based tracing tool for weather research and forecast tool system will include WRF model software which is the main focus area of this project, an open source provenance data collection tool named Komadu and the related subsystems with the Komadu which are mentioned under the technical feasibility title and a machine learning based tracing software. Detailed design of the system is covered in the following sections.

4.1 Software Analysis

Weather Research and Forecast Model software produces log files in different levels. WRF has 150 logging levels which are ordered from least to most detailed logging. Within the scope of this project, different logging levels will be used. Clearly, choosing the most detailed logging level will give the most information about the execution of the WRF model but it also takes more time to process the log files.

As mentioned in the previous chapter, log files include some relationship information between the components of the WRF model software. These relationships give the best clues about the internal structure of the model and also the execution and the input parameters of it.

While the WRF model is working and producing log files, our tracing tool will be reading the log files. It will take every log line and classify it as relevant with the information which we can use to understand workflow of the model or as irrelevant.

In order to classify the log lines I used both binary and multi-class classifiers including Logistic Regression, Naive Bayes Algorithm, Decision Tree, Random Forests Algoritm, Multilayer Perceptron and OneVsAll algorithms. Before the classification process we will prepare a ground truth documentation for the input log file. Since we know the ground truth values, we can easily calculate and compare the correctness of the aforementioned binary classifier algorithms.

Figure 4.1 In order to make the provenance data more understandable, we will visualize the data on a data visualization, data integration and data analysis software named Cytoscape.

After marking the lines as relevant or irrelevant in the log file, we will get the relevant log records in order to create data provenance. Creating data provenance workflows will help researchers to clearly understand if the WRF model is working fine with true parameters or is there something wrong with the workflow.

To make workflows more understandable, we will visualize the provenance data that our tracing tool produce from the log files by using a data visualization software named Cytoscape. In our previous project, we implemented a plugin for Cytoscape, for the purpose of easily converting provenance data to .csv files and visualize it. This plugin features filtering, highlighting, sorting, grouping, different visual templates and a real time provenance data visualization component. With these capabilities, integrating our tracing tool to the Cytoscape plugin will give us a better understanding of the data.

4.2 Input-Output Analysis

Figure 4.2 Input output design of the system.

The system will take the WRF model software log files as input. It will mark the log lines as relevant or irrelevant with the internal structure of the model by using machine learning algorithms mentioned in Software Design section.

After marking the log lines as relevant or irrelevant the tool will create Komadu notifications and ingest them into the Komadu database by using Komadu's APIs with a unique workflow id for each log file. Giving unique workflow id will allow us to query the data provenance workflows which represents the internal working process of the WRF model.

Querying the workflow ids on Komadu will produce an XML based Provenance Ontology (PROV-O) file. Thanks to our provenance data visualization software, we can easily transform the PROV-O files and visualize them as nodes and edges.

5
System Design

The system contains 4 major components. This components are Weather Research and Forecast Model, our adaptive learning based WRF Model tracing tool, our PROV-O producer and a Cytoscape plugin which I developed during my previous projects. Basically the system works at the given order below:

- WRF Model produces logs which contains valuble information about it's internal structure
- Our adaptive learning based WRF Model tracing tool gets the log files and produces a binary labeled and/or multi-class labeled sparse vector by processing each line.
- Sparse vector will be used to apply machine learning algorithms that mentioned at Introduction chapter.
- According to multi-class labeled data, a PROV-O based .xml file will be produced by our PROV-O producer.
- PROV-O file will be effectively visualized by our Cytoscape plugin which I developed in my previous projects.

In this chapter, the software design of our adaptive learning based WRF Model tracing tool and the input-output design of the whole system will be covered. Before going into software design part, all of the machine learning algorithms needs to be introduced.

Our adaptive learning based WRF Model tracing tool includes 4 different machine learning algorithms. As it mentioned in Chapter 1, these algorithms are Logistic Regression, Naive Bayes Algorithm, Decision Tree Algorithm, Random Forest Algorithm and Multilayer Perceptron Classification Algorithm.

All of these algorithms are used for both binary classification and multi-class classification. All of the algorithms are implemented with Apache Spark MLLib library but for the Naive Bayes algorithm and Logistic Regression algorithm there is also my implementation which I developed without using any library or framework in our tool. Here I will introduce these algorithms shortly.

- Logistic Regression: [9] Logistic regression is a statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which there are only two possible outcomes).

  In logistic regression, the dependent variable is binary or dichotomous, i.e. it only contains data coded as 1 (TRUE, success, pregnant, etc.) or 0 (FALSE, failure, non-pregnant, etc.).

  The goal of logistic regression is to find the best fitting (yet biologically reasonable) model to describe the relationship between the dichotomous characteristic of interest (dependent variable = response or outcome variable) and a set of independent (predictor or explanatory) variables. Logistic regression generates the coefficients (and its standard errors and significance levels) of a formula to predict a logit transformation of the probability of presence of the characteristic of interest:

  $$logit(p) = b_0 + b_1X_1 + b_2X_2 + b_3X_3 + \ldots + b_kX_k$$

  Figure 5.1 Logistic Regression logit formula

  where p is the probability of presence of the characteristic of interest. The logit transformation is defined as the logged odds:

  $$odds = \frac{p}{1-p} = \frac{probability\ of\ presence\ of\ characteristic}{probability\ of\ absence\ of\ characteristic}$$

  Figure 5.2 Odds formula

  and

  $$logit(p) = \ln\left(\frac{p}{1-p}\right)$$

  Figure 5.3 Logit transformation formula

This study also includes multi-class classification with using multinomial logistic regression algorithm. In statistics, multinomial logistic regression is a classification method that generalizes logistic regression to multiclass problems, i.e. with more than two possible discrete outcomes. That is, it is a model that is used to predict the probabilities of the different possible outcomes of a categorically distributed dependent variable, given a set of independent variables (which may be real-valued, binary-valued, categorical-valued, etc.). [10]

- Naive Bayes: It is a classification technique based on Bayes' Theorem[11] with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as 'Naive'. This study include both binary and multi-class classification with using Naive Bayes algorithm.[12]

- Decision Tree: A decision tree classifies data items by posing a series of questions about the features associated with the items. Each question is contained in a node, and every internal node points to one child node for each possible answer to its question. The questions thereby form a hierarchy, encoded as a tree. In the simplest form, we ask yes-or-no questions, and each internal node has a 'yes' child and a 'no' child. An item is sorted into a class by following the path from the topmost node, the root, to a node without children, a leaf, according to the answers that apply to the item under consideration. An item is assigned to the class that has been associated with the leaf it reaches. [13]

- Random Forest: Random Forest is a supervised learning algorithm. Like you can already see from it's name, it creates a forest and makes it somehow random. The „forest" it builds, is an ensemble of Decision Trees, most of the time trained with the "bagging" method. The general idea of the bagging method is that a combination of learning models increases the overall result.

  To say it in simple words: Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction. [14]

- Multilayered Perceptron Classifier: Multilayer perceptron classifier (MLPC) is a classifier based on the feedforward artificial neural network [15]. MLPC consists of multiple layers of nodes. Each layer is fully connected to the next layer in the network. Nodes in the input layer represent the input data. All other nodes maps inputs to the outputs by performing linear combination of the inputs with the node's weights w and bias b and applying an activation function. It can be written in matrix form for MLPC with K+1 layers as follows:

$$y(x) = f_k(...f_2(w_2^T f_1(w_1^T x + b_1) + b_2)... + b_K) \tag{5.1}$$

Nodes in intermediate layers use sigmoid (logistic) function:

$$f(z_i) = \frac{1}{1 + \frac{1}{e^(z_i)}} \tag{5.2}$$

Nodes in the output layer use softmax function:

$$f(z_i) = e_i^z \sum_{n=1}^{\infty} e_k^z \tag{5.3}$$

The number of nodes N in the output layer corresponds to the number of classes. [16]

- OneVsAll Classifier: One-vs.-rest[2]:182, 338 (or one-vs.-all, OvA or OvR, one-against-all, OAA) strategy involves training a single classifier per class, with the samples of that class as positive samples and all other samples as negatives. This strategy requires the base classifiers to produce a real-valued confidence score for its decision, rather than just a class label; discrete class labels alone can lead to ambiguities, where multiple classes are predicted for a single sample.[17]

5.1 Software Design

Adaptive learning based WRF tracing tool includes 5 different modules. The module names are Sparse Vector Producer, Binary Labeled Data Producer, Multi-Class Labeled Data Producer, Machine Learning Algorithms module and PROV-O Data Converter module.

The first module gets the log file of WRF Model and produces a sparse vector. A sparse vector shows the content of whole document by assigning an unique reference id to every unit. Here we used words in the log file as units.

Binary labeled data producer module gets the sparse vector and a rule list as input. From using rule list, the module marks every row as 1 or 0 by adding them at the and of each row. In our case 1 represents that the row includes provenance information and the 0 represents that the row does not includes any provenance information.

Multi-Class labeled data producer modules works similar to binary version of it. It gets the sparse vector and a set of rule list. With using the rule list it marks every line according to what PROV-O relation they have. I our case, the WRF Model log files contains only 4 of the PROV-O relations. These are Comunnication, Derivation, Usage and Generation. The module puts 0 for every line that does not contain any PROV-O information and puts 1,2,3 or 4 for the other relations.

Machine learning parts includes 5 different machine learning algorithm which I implemented in different classes. There is a base structure for these classes which contains the common parameters for all of the algorithms. For example, where to split the data as test and training or how many times the algorithms will work and gets the averages of the results.

The last module is PROV-O data converter module which gets only the PROV-O actors and their relations as input. In here PROV-O actors means among whom (which structures) the relations has been occured. For example, this line is taken from WRF log files:

> wrf: calling alloc-and-configure-domain

For this line, our PROV-O data converter will detect 2 actors which are wrf and allow-and-configure-domain actor and 1 relation which is calling. Calling refers to communication in our rules.

These modules does not dependent each other except the input and output relation. The workflow schema of the internal structure of the system is given in below figure.

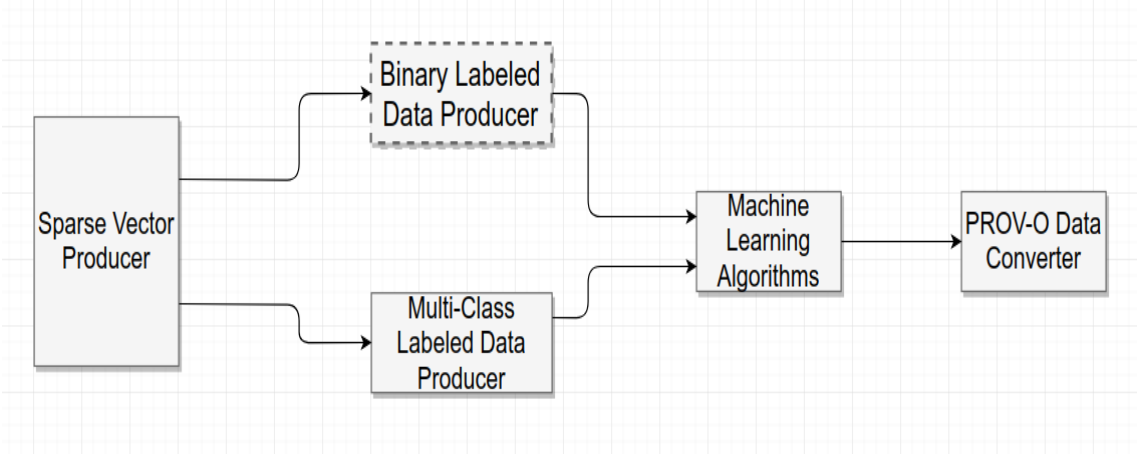

Figure 5.4 The block diagram of the Adaptive Learning Based WRF Model Tracing Tool

After describing the components of the system, now the information about the internal structure of the system will be given. The detailed UML Class Diagram of Adaptive Learning Based WRF Model Tracing Tool software is given in the appendixes.

The sequence diagram of the system is also generated. A digital version of the sequence diagram will be given with this project report.

5.2 Input-Output Design

The whole system including our Adaptive Learning Based WRF Model Tracing Tool consist of 4 major components. These components are Weather Research and Forecasting Model, Adaptive Learning Based WRF Model Tracing Tool, PROV-O data producer and again our Cytoscape data visualization and analysis plugin.

As it mentioned in Input-Output Analysis part, we implemented 3 of these 4 components which are Adaptive Learning Based WRF Model Tracing Tool, PROV-O data producer and again our Cytoscape data visualization and analysis plugin. The Cytoscape plugin component was the product of my previous project. [2]

Figure 5.5 Input - Output Design of the whole system.

First of all, the WRF Model produces some log files which includes lots of valuable information about the internal structure of the Model software. Our Adaptive Learning Based WRF Model Tracing Tool reads the log files and produces sparse vectors from them.

According to machine learning algorithm, our Adaptive Learning Based WRF Model Tracing Tool will produce binary class labeled data file or multi-class class labeled data file. With using these class labeled data files the tracing tool will apply one of the Logistic Regression, Naive Bayes Algorithm, Decision Tree, Random Forest Algorithm and Multilayer Perceptron Classification Algorithm.

After saving the accuracy, precision and recall results of the machine learning algorithms, the PROV-O data procuder will start to process the log files. It will get the lines which previously marked as contains provenance information and produces PROV-O formatted .xml files as mentioned in software design part.

Last step of the input-output design is to visualizing the PROV-O data by using our Cytoscape data visualization and data analysis tool. A sample visualization of the PROV-O data by our visalization tool is shown in below figure.

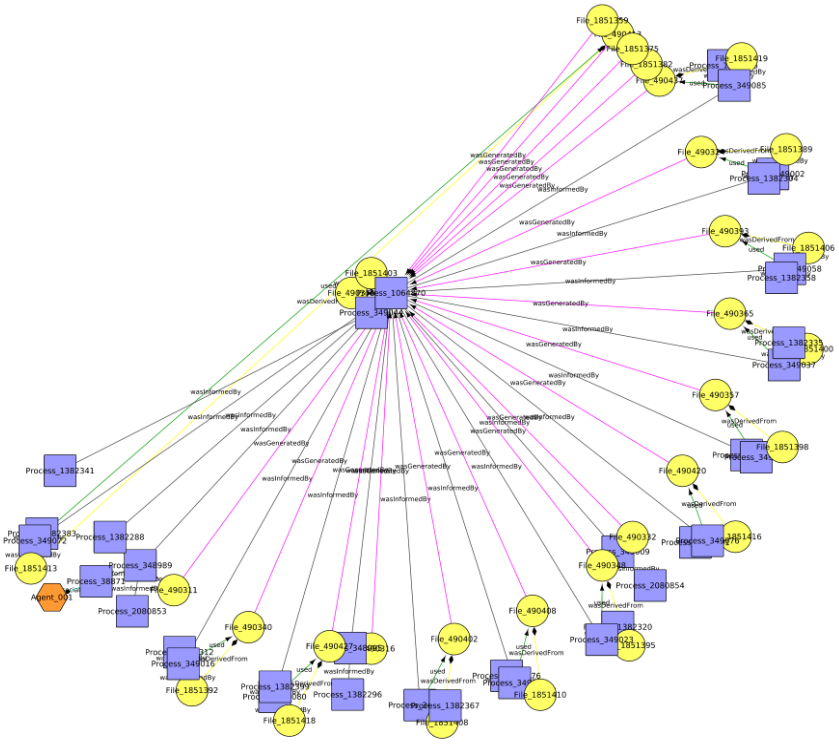

Figure 5.6 A sample visualization of PROV-O data by our Cytoscape data visualization and data analysis tool

6
Application

Adaptive Learning Based WRF Model Tracing Tool has a simple Graphical User Interface (GUI) which is developed with using JavaFX framework. The aim of the GUI is making easier to select and apply different machine learning algorithms on any of WRF Model software with using arbitrary rates of test and training data and iteration count.

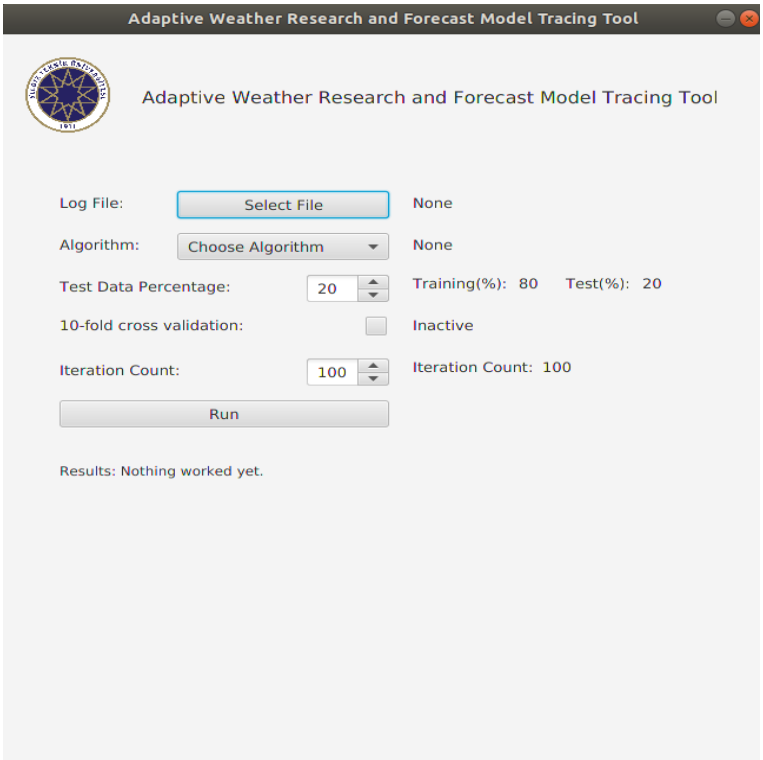

Figure 6.1 Graphical User Interface of the Adaptive Learning Based WRF Model Tracing Tool

The GUI has only one window which makes it easy to handle all the situations. A screenshot that shows the initial state of the window is shown in above figure. User can select the WRF log file, the machine learning algorithm, test and training data rate or 10-fold cross validation and iteration count.

The iteration count represents that how many times the data will split and machine learning algorithm runs with the new splitted data. For each iteration the output values which are accuracy, precision and recall is saved and at the end of the iterations, the average and the standard deviation values of the mentioned parameters will be shown to the user.

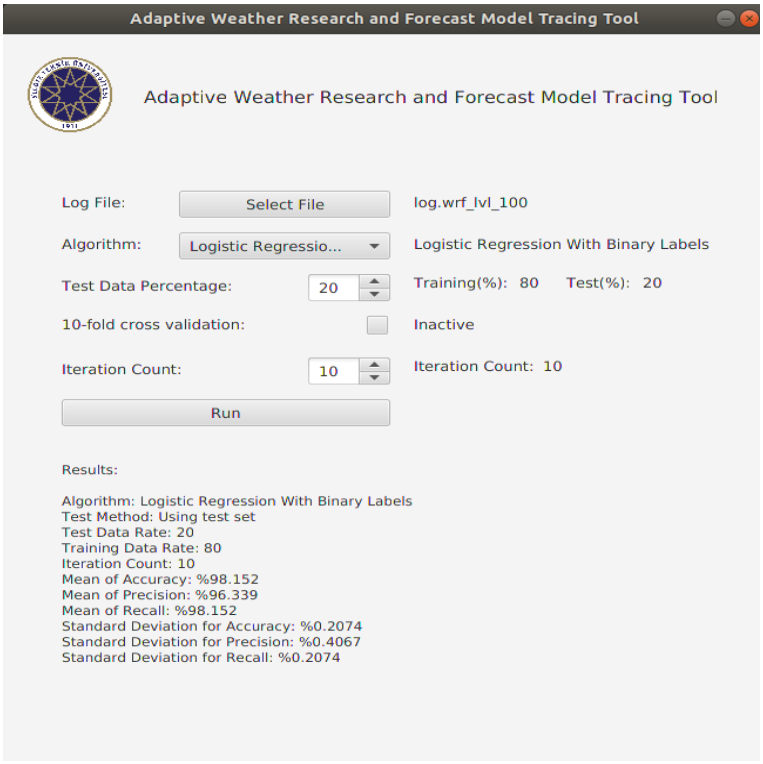

Figure 6.2 Adaptive Learning Based WRF Model Tracing Tool's GUI shows the results of the ML algorithm with given parameters.

The state of the GUI after choosing the necessary parameters is shown in Figure 6.2. Here the user choosed the log file named "log.wrf-lvl-100", the Logistic Regression algorithm with binary class labels, 80 percent as traning data rate, 20 percent as test data rate and the iteration count is 10.

The results are shown up at the bottom of the GUI. Result string includes the input parameters which user gave before running the tracing tool. The accuracy, precision and recall values for the given parameters is listed here as well.

Finally, the PROV-O file that our PROV-O data producer module generated from the WRF log file with using multi-class classification techniques will be visualized as shown in figure 6.3.

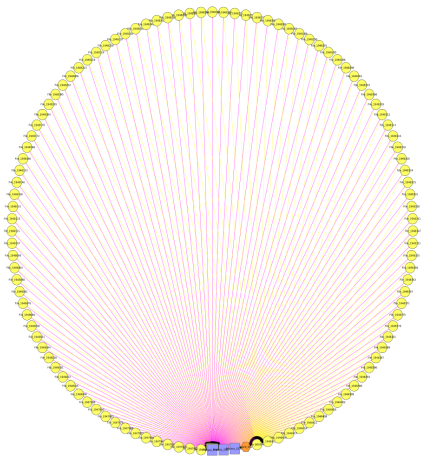

Figure 6.3 Sample illustration for PROV-O data visualization. Here the data is taken from our PROV-O data producer module.

7
Experimental Results

In this section, the experimental results of the whole system will be given. As it mentioned in other sections, the system contains 5 different machine learning algorithms. Here the accuracy, precision and recall results of those algorithms will be given.

The algorithms are tested with two different data files. Both of the data files are taken from Weather Research and Forecast Model's software. Files has different level of log records which are 100 and 150.

Tests are applied with different level of training and test data rates from %40 training - %60 test rates to %80 training - %20 test rates. While creating the test and training sets from the data, all the lines are choosen randomly. Results are given between the range of 0 and 1. In order to get the results, I applied the algorithms for several times with same parameters and took the averages of them.

Naive Bayes Binary Accuracy	%40-%60	%50-%50	%60-%40	%70-%30	%80-%20
Log_100	0.99858105711245	0.998726655348048	0.998768147822261	0.998705120659212	0.99860115404791
Log_150	0.9927042315457	0.992503532680286	0.992488628202059	0.992133884496529	0.991899189918992
Naive Bayes Binary Precision	%40-%60	%50-%50	%60-%40	%70-%30	%80-%20
Log_100	0.99868332326651	0.998812178496313	0.99884653841539	0.998794703506687	0.998708757582687
Log_150	0.9953201013746	0.995254287980486	0.995294214989683	0.995162879857394	0.995115688039392
Naive Bayes Binary Recall	%40-%60	%50-%50	%60-%40	%70-%30	%80-%20
Log_100	0.99858105711245	0.998726655348048	0.998768147822261	0.998705120659212	0.99860115404791
Log_150	0.9927042315457	0.992503532680286	0.992488628202059	0.992133884496529	0.991899189918992
Naive Bayes Multiclass Accuracy	%40-%60	%50-%50	%60-%40	%70-%30	%80-%20
Log_100	0.99763509518742	0.997594793435201	0.99762428508579	0.997645673925839	0.997202308095821
Log_150	0.99476303743829	0.994515364165449	0.994733062006225	0.993257615282739	0.992199219921992
Naive Bayes Multiclass Precision	%40-%60	%50-%50	%60-%40	%70-%30	%80-%20
Log_100	0.99866564034488	0.99886372583179	0.99877083082903	0.998715627500024	0.998569959592557
Log_150	0.99932625356453	0.999315555298049	0.999375195573497	0.999384862488042	0.999360682817497
Naive Bayes Multiclass Recall	%40-%60	%50-%50	%60-%40	%70-%30	%80-%20
Log_100	0.99763509518742	0.997594793435201	0.99762428508579	0.997645673925839	0.997202308095821
Log_150	0.99476303743829	0.994515364165449	0.994733062006225	0.993257615282739	0.992199219921992

Figure 7.1 Accuracy, precision and recall results of the Naive Bayes algorithm for 2 different data file.

In the figure 7.1, the accuracy, precision and recall results of Naive Bayes algorithm for 2 different data files and several different threshold values are given. As the results shows, when the rate of the training rate increases then the accuracy values are increasing as well.

The results belongs to the other machine learning algorithms are given below figures.

Decision Tree Binary Accuracy	%40-%60	%50-%50	%60-%40	%70-%30	%80-%20
Log_100	0.998434341622684	0.998829829509133	0.99844319833817	0.998589872471469	0.998333122205113
Log_150	0.999582449965437	0.99943669523735	0.999502038029861	0.999521024106191	0.999464995790294
Decision Tree Binary Precision	%40-%60	%50-%50	%60-%40	%70-%30	%80-%20
Log_100	0.998433921176019	0.998831235248165	0.998445763171075	0.998591949936683	0.998336026489247
Log_150	0.999582630232084	0.999437020131284	0.999502289309171	0.999521258387001	0.999465301086408
Decision Tree Binary Recall	%40-%60	%50-%50	%60-%40	%70-%30	%80-%20
Log_100	0.998434341622684	0.998829829509133	0.99844319833817	0.998589872471469	0.998333122205113
Log_150	0.999582449965437	0.99943669523735	0.999502038029861	0.999521024106191	0.999464995790294
Decision Tree Multiclass Accuracy	%40-%60	%50-%50	%60-%40	%70-%30	%80-%20
Log_100	0.997635095187419	0.997594793435201	0.99762428508579	0.997645673925839	0.997202308095821
Log_150	0.99962034959447	0.999616702201566	0.999670560047918	0.993257615282739	0.999461625889813
Decision Tree Multiclass Precision	%40-%60	%50-%50	%60-%40	%70-%30	%80-%20
Log_100	0.998665640344875	0.99886372583179	0.99877083082903	0.998715627500024	0.998569959592557
Log_150	0.999589259147647	0.999587551669114	0.999610726472465	0.999384862488042	0.999327182698103
Decision Tree Multiclass Recall	%40-%60	%50-%50	%60-%40	%70-%30	%80-%20
Log_100	0.997635095187419	0.997594793435201	0.99762428508579	0.997645673925839	0.997202308095821
Log_150	0.99962034959447	0.999616702201566	0.999670560047918	0.993257615282739	0.999461625889813

Figure 7.2 Accuracy, precision and recall results of the Decision Tree algorithm for 2 different data file.

LR Binary Accuracy	%40-%60	%50-%50	%60-%40	%70-%30	%80-%20
Log_100	0.98173111032281	0.982314657611772	0.981874175098988	0.982577987051206	0.983213848574925
Log_150	0.98694757040916	0.987066797595382	0.987401244912617	0.987438295139864	0.987698769876988
LR Binary Precision	%40-%60	%50-%50	%60-%40	%70-%30	%80-%20
Log_100	0.96379597297566	0.964942086558932	0.964076895726318	0.965459500637601	0.966709472029517
Log_150	0.97406550673655	0.974300862915204	0.974961218454985	0.975034386708722	0.975548860016514
LR Binary Recall	%40-%60	%50-%50	%60-%40	%70-%30	%80-%20
Log_100	0.98173111032281	0.982314657611772	0.981874175098988	0.982577987051206	0.983213848574925
Log_150	0.98694757040916	0.987066797595382	0.987401244912617	0.987438295139864	0.987698769876988
LR Multiclass Accuracy	%40-%60	%50-%50	%60-%40	%70-%30	%80-%20
Log_100	0.99846281187182	0.998655913978495	0.998416190057193	0.998351971748087	0.998251442559888
Log_150	0.99946031301845	0.999401240629416	0.999431410102945	0.999438134606895	0.9994599459946
LR Multiclass Precision	%40-%60	%50-%50	%60-%40	%70-%30	%80-%20
Log_100	0.99692798669099	0.997313634524222	0.996834888568321	0.996706659493293	0.996505942572897
Log_150	0.99892091729894	0.998802839771616	0.99886314350036	0.998876584906509	0.998920183647528
LR Multiclass Recall	%40-%60	%50-%50	%60-%40	%70-%30	%80-%20
Log_100	0.99846281187182	0.998655913978495	0.998416190057193	0.998351971748087	0.998251442559888
Log_150	0.99946031301845	0.999401240629416	0.999431410102945	0.999438134606895	0.9994599459946

Figure 7.3 Accuracy, precision and recall results of the Logistic Regression algorithm for 2 different data file.

Multilayered Perceptron Binary Accuracy	%40-%60	%50-%50	%60-%40	%70-%30	%80-%20
Log_100	0.999113160695282	0.999080362195812	0.999472063352398	0.998705120659211	0.999650288511977
Log_150	0.999765794250249	0.999684542586751	0.99973795693759	0.999707465480927	0.999564459930314
Multilayered Perceptron Binary Precision	%40-%60	%50-%50	%60-%40	%70-%30	%80-%20
Log_100	0.999113961091766	0.999081222350604	0.999472347062195	0.99870682485552	0.99965041285384
Log_150	0.999765223632477	0.999683838166199	0.999737302274591	0.999706588065228	0.999564651764382
Multilayered Perceptron Binary Recall	%40-%60	%50-%50	%60-%40	%70-%30	%80-%20
Log_100	0.999113160695282	0.999080362195812	0.999472063352398	0.998705120659211	0.999650288511977
Log_150	0.999765794250249	0.999684542586751	0.99973795693759	0.999707465480927	0.999564459930314
Multilayered Perceptron Multiclass Accuracy	%40-%60	%50-%50	%60-%40	%70-%30	%80-%20
Log_100	0.999645264278113	0.999292586304471	0.999560052793665	0.999764567392584	0.999125721279944
Log_150	0.999824345687687	0.99975464423414	0.99973795693759	0.999707465480927	0.999564459930314
Multilayered Perceptron Multiclass Precision	%40-%60	%50-%50	%60-%40	%70-%30	%80-%20
Log_100	0.966709472029517	0.999290614335155	0.999560249832005	0.999764623790382	0.999126498002332
Log_150	0.999824058396732	0.999754007716663	0.999737302274591	0.99970755201424	0.999564651764382
Multilayered Perceptron Multiclass Recall	%40-%60	%50-%50	%60-%40	%70-%30	%80-%20
Log_100	0.999645264278113	0.999292586304471	0.999560052793665	0.999764567392584	0.999125721279944
Log_150	0.999824345687687	0.99975464423414	0.99973795693759	0.999707465480927	0.999564459930314

Figure 7.4 Accuracy, precision and recall results of the Multilayer Perceptron algorithm for 2 different data file.

Number of tree used in tests: 20					
Random Forest Binary Accuracy	%40-%60	%50-%50	%60-%40	%70-%30	%80-%20
Log_100	0.989594418824642	0.990379173740804	0.989529256489221	0.98964096527369	0.989858366847351
Log_150	0.987027524036059	0.987258400593969	0.987610725401006	0.987638961351688	0.987938793879388
Random Forest Binary Precision	%40-%60	%50-%50	%60-%40	%70-%30	%80-%20
Log_100	0.98970355310072	0.990472486561461	0.989639738709277	0.989749038190812	0.989961907548434
Log_150	0.987195822624777	0.987420780061786	0.987764251686856	0.987791787295155	0.988084301497793
Random Forest Binary Recall	%40-%60	%50-%50	%60-%40	%70-%30	%80-%20
Log_100	0.989594418824642	0.990379173740804	0.989529256489221	0.98964096527369	0.989858366847351
Log_150	0.987027524036059	0.987258400593969	0.987610725401006	0.987638961351688	0.987938793879388
Random Forest Multiclass Accuracy	%40-%60	%50-%50	%60-%40	%70-%30	%80-%20
Log_100	0.998462811871822	0.998655913978495	0.998416190057193	0.998351971748087	0.998251442559888
Log_150	0.999460313018449	0.999401240629416	0.999431410102945	0.999438134606895	0.999459945994599
Random Forest Multiclass Precision	%40-%60	%50-%50	%60-%40	%70-%30	%80-%20
Log_100	0.996927986690986	0.997313634524222	0.996834888568321	0.996706659493293	0.996505942572897
Log_150	0.998920917298937	0.998802839771616	0.99886314350036	0.998876584906509	0.998920183647528
Random Forest Multiclass Recall	%40-%60	%50-%50	%60-%40	%70-%30	%80-%20
Log_100	0.998462811871822	0.998655913978495	0.998416190057193	0.998351971748087	0.998251442559888
Log_150	0.999460313018449	0.999401240629416	0.999431410102945	0.999438134606895	0.999459945994599

Figure 7.5 Accuracy, precision and recall results of the Random Forest algorithm for 2 different data file.

First data file named log-100 is the WRF log file with 100 log level contains about 27.000 rows. And the second data file named log-150 is the WRF log file with 150 log level contains about 83.000 rows.

Since the data is big enough to create sufficient classification models the results are not changing from different threshold levels. The results shows that actually the %40 of the randomly choosed data is not remarkably different than the %80 of the randomly choosed data.

	Mean Values			SD Values		
Logistic Regression With Binary Labels	Log_100	Log_150	Log_150v2	Log_100	Log_150	Log_150v2
Accuracy	0.981888325	0.984742361	0.992851165	5.94E-04	3.71E-04	2.79E-04
Precision	0.964108501	0.969724691	0.985754443	0.001167579	7.31E-04	5.53E-04
Recall	0.981888325	0.984742361	0.992851165	5.94E-04	3.71E-04	2.79E-04
Logistic Regression With Multi Class Labels	Log_100	Log_150	Log_150v2	Log_100	Log_150	Log_150v2
Accuracy	0.999162384	0.999744381	0.999851948	1.37E-04	6.81E-05	1.41E-05
Precision	0.998325731	0.999488857	0.99970393	2.73E-04	1.36E-04	2.82E-05
Recall	0.999162384	0.999744381	0.999851948	1.37E-04	6.81E-05	1.41E-05
Naive Bayes With Binary Labels	Log_100	Log_150	Log_150v2	Log_100	Log_150	Log_150v2
Accuracy	0.999744318	0.991448685	0.995896831	5.27E-05	2.16E-04	3.23E-05
Precision	0.999760722	0.994553591	0.997427209	4.78E-05	1.17E-04	4.02E-05
Recall	0.999744318	0.991448685	0.995896831	5.27E-05	2.16E-04	3.23E-05
Naive Bayes With Multi Class Labels	Log_100	Log_150	Log_150v2	Log_100	Log_150	Log_150v2
Accuracy	0.998483168	0.994493021	0.997502852	9.85E-05	1.13E-04	1.16E-04
Precision	0.99964435	0.999734874	0.999864111	7.43E-05	5.23E-05	2.39E-05
Recall	0.998483168	0.994493021	0.997502852	9.85E-05	1.13E-04	1.16E-04
Random Forest With Binary Labels	Log_100	Log_150	Log_150v2	Log_100	Log_150	Log_150v2
Accuracy	0.9898649	0.985193695	0.992847636	1.76E-04	6.16E-04	1.15E-04
Precision	0.989971701	0.98335173	0.98735182	1.74E-04	0.002212713	8.26E-04
Recall	0.9898649	0.985193695	0.992847636	1.76E-04	6.16E-04	1.15E-04
Random Forest With Multi Class Labels	Log_100	Log_150	Log_150v2	Log_100	Log_150	Log_150v2
Accuracy	0.999343492	0.999760358	0.999855473	1.26E-04	8.39E-05	1.50E-05
Precision	0.998687771	0.999520829	0.99971098	2.51E-04	1.68E-04	2.99E-05
Recall	0.999343492	0.999760358	0.999855473	1.26E-04	8.39E-05	1.50E-05
Multilayer Perceptron Classifier With Binary Labels	Log_100	Log_150	Log_150v2	Log_100	Log_150	Log_150v2
Accuracy	0.999840425	0.999928112	0.999883153	7.53E-05	3.39E-05	1.95E-05
Precision	0.999840585	0.999924508	0.99988276	7.52E-05	3.46E-05	1.98E-05
Recall	0.999840425	0.999928112	0.999883153	7.53E-05	3.39E-05	1.95E-05
Multilayer Perceptron Classifier With Multi Class Labels	Log_100	Log_150	Log_150v2	Log_100	Log_150	Log_150v2
Accuracy	0.999775321	0.999912984	0.999918855	1.25E-04	3.29E-05	2.25E-05
Precision	0.999775616	0.999913049	0.999918724	1.24E-04	3.27E-05	2.26E-05
Recall	0.999775321	0.999912984	0.999918855	1.25E-04	3.29E-05	2.25E-05
OneVsAll Based On Logistic Regression	Log_100	Log_150	Log_150v2	Log_100	Log_150	Log_150v2
Accuracy	0.999662868	0.999934731	0.999958426	1.76E-04	1.85E-05	1.47E-05
Precision	0.999570836	0.999913415	0.999950112	2.46E-04	3.07E-06	1.47E-05
Recall	0.999662868	0.999934731	0.999958426	1.76E-04	1.85E-05	1.47E-05

Figure 7.6 Accuracy, precision and recall results of all algorithms with using 10-fold cross validation method.

In the above figure you can see the mean and the standard deviation valeus of the accuracy, precision and recall values for all of the algorithms. If we compare the test results of 10-fold cross-validation and the traditional test-training split method, we can say that 10-fold cross validations is slightly better than the other approach in our case.

The results tells us that our learning model is working well. Because all of the accuracy, precision and recall values are higher than %95 which is high enough to claim that.

The machine learning algorithms that I implemented myself gives about same results with the Apache Spark implementations. There is a small difference about %0.5 to %1 in Naive Bayes and %1 to %3 in Logistic Regression.

Since the difference between the implementations are almost same, giving detailed test results as I did in the previous part are not necessary. Instead of this, here the accuracy, precision and recall results will be given for %80 traning and %20 test data splits which is generally gives more accurate results with some exceptions.

%80-%20 Traning and Test | log_100 log file	Accuracy	Precision	Recall
My Naive Bayes Implementation For Binary Classification	%99.05	%99.38	%99.05
Spark Naive Bayes Implementation For Binary Classification	%99.86	%99.87	%99.86
My Naive Bayes Implementation For Multi-Class Classification	%98.89	%99.86	%98.83
Spark Naive Bayes Implementation For Multi-Class Classification	%99.72	%99.85	%99.73

Figure 7.7 Accuracy, precision and recall results comparison of my implementation and Spark's implementation of Naive Bayes Algorithm.

%80-%20 Traning and Test | log_100 log file	Accuracy	Precision	Recall
My Logistic Regression Implementation For Binary Classification	%99.89	%99.72	%95.83
Spark Logistic Regression Implementation For Binary Classificati	%98.32	%96.67	%98.34

Figure 7.8 Accuracy, precision and recall results comparison of my implementation and Spark's implementation of Logistic Regression Algorithm.

%80-%20 Training and Test | log_100 file	Accuracy	Precision	Recall
Spark OneVsAll implementation based on Logistic Regression	%99.96	%99.95	%99.96
My OneVsAll implementation based on Logistic Regression	%99.91	%99.98	%99.91

Figure 7.9 Accuracy, precision and recall results comparison of my implementation and Spark's implementation of OneVsAll Algorithm.

As you can see from the figures 7.6, 7.7 and 7.8 the difference between implementations are quite small for my Naive Bayes algorithms binary and multi-class classification, Logistic Regression binary classification and OneVsAll Classifier based on Logistic Regression implementations.

Another thing that we want to share is the result of the PROV-O producer module. As it mentioned in chapter 6, PROV-O data producer creates a PROV-O based XML file from the sparse vectors which has the information of interactions between nodes at the end of each of it's lines.

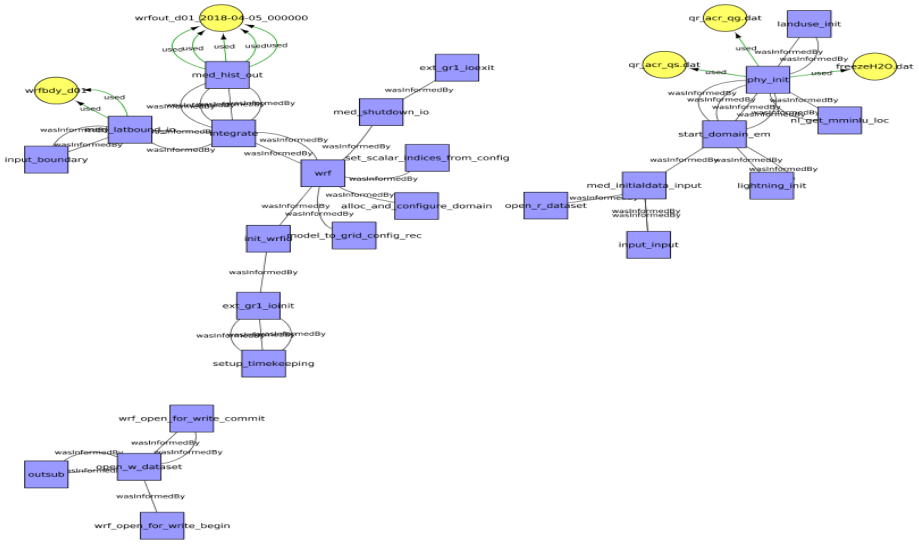

Figure 7.10 The visualization of the output file of PROV-O data producer module.

In the figure 7.6, a simple data visualization of provenance data that generated by our PROV-O producer and visualized by our Cytoscape data visualization and data analysis plugin is shown. Here the blue squares represents the activity nodes while the yellow circle represents the entity nodes.

We can easily apply different visualization techniques on the provenance data thanks to our data visualization plugin. Some of these techniques are filtering, highlighting, sorting, applying different visual templates, real time data provenance visualization and grouping.

8
Performance Analysis

In this chapter, the performance of the whole system will be analysed. Since the system consists of 4 different units, the perfromance of all units will be covered except the Weather Research and Forecast software.

The first part of the work flow of the system is reading the WRF log files and producing sparse vectors. This part takes approximately 5 seconds. And the second part of the work flow is creating either binary class labeled or multi-class class labeled sparse vectors. Here the related module inserts an additional column which shows the class labels. These both parts takes under 100 milliseconds.

These values are for 100 level file which is quite smaller than the 150 level log file that we both used in accuracy tests. A detailed results for both data files are given in the figure below.

	Log_100	Log_150
Time to create sparse vector	5131	5598
Time to insert binary class labels to sparse vectors	62	134
Time to insert multi-class class labels to sparse vectors	78	116
* Times are given in milliseconds		

Figure 8.1 Total time to creating sparse vectors and inserting class labels in milliseconds.

The second part is where we apply the machine learning algorithms. Here I applied machine learning algorithms on both 100 level and 150 level log files several times and took average of work times.

In the above figure you can see the average times to run each of the algorithms. Since the WRF model takes too much time to do it's internal processes (in hours), total time to apply machine learning algorithms on WRF model log files is not high.

And the last part of the system was to generate a provenance data file from the WRF

	Log_100	Log_150
Logistic Regression With Binary Labels	1415	4357
Logistic Regression With Multi Class Labels	2006	5779
Naive Bayes With Binary Labels	1544	5735
Naive Bayes With Multi Class Labels	1781	6109
Random Forest With Binary Labels	5465	39452
Random Forest With Multi Class Labels	8683	83514
Multilayer Perceptron Classifier With Binary Labels	7226	323986
Multilayer Perceptron Classifier With Multi Class Labels	11360	386201
Decision Tree Algorithm With Binary Labels	4599	39452
Decision Tree Algorithm With Multi Class Labels	11490	83514
OneVsAll Classifier Based On Logistic Regression	5811	17334
Manually Implemented Naive Bayes With Binary Class Labels	9803	129523
Manually Implemented Naive Bayes With Multi-Class Class Labels	11864	198467
Manually Implemented Logistic Regression With Binary Class Labels	22620	79052
Manually Implemented OneVsAll Classifier Based On Logistic Regression	63446	153412
* Times are given in milliseconds		

Figure 8.2 Total time to apply machine learning algorithms in milliseconds.

log files that we previously marked as contains provenance information. Since the count of rows that we marked as contains provenance information is relatively small comparing to whole log file, this part does not takes time to produce a provenance file.

For example for the first log file there are about 50 rows that contains provenance information among the 28.000 rows. Total time to produce provenance data file from these rows takes less then 5 milliseconds for both 100 level and 150 level log files.

9
Conclusion

In this study, we worked on an adaptive learning based tracing tool for Weather Research and Forecasting Model.The Weather Research and Forecasting (WRF) Model is a next-generation mesoscale numerical weather prediction system designed for both atmospheric research and operational forecasting applications.

Because of it's internal structure the WRF model can not be stopped during it's process time. So the model produces huge amount of log files in order to tell the researchers the current state of the model. Since the model can not be stopped while it works, it is hard to follow the processes of the model from the log file.

Figure 9.1 Work flow of the whole system.

As it shown in above figure, the system consists of 4 major components. These components are WRF Model, our log file classifier, our PROV-O data producer and

our Cytoscape plugin to visualize and analyse the data.

Here we implemented a tracing tool to successfully trace the WRF model from the different levels of log files with using data provenance. Firstly we examined the log files and extract the rows which contains provenance information. As a second step we discovered the provenance relations in the rows that we previously find as contains provenance information.

From these rule-based structure we implemented machine learning algorithms in order to create a learning model that saves the researchers time with analysing the log files. And the results that we covered in chapter 7 shows us that the learning model is working well. Since we consider the huge amount of time that WRF model took to work, the performance results that we calculated and gave in chapter 8 is negligible.

After the classification process we produce PROV-O based XML files from the log files which we marked as contains provenance information. With using these provenance data files we visualize the general work flow of the WRF model as nodes and edges in a data visualization and data analysis software named Cytsocpe.

In my previous project, I developed a plugin for Cytoscape in order to easily visualize and also analyse the provenance data. With using this plugin, work flow of the WRF model can easily be visualized and analysed in Cytoscape.

After getting the sufficient results as we mentioned in previous chapters, we can say that with in this study we achieved our goals which we defined in introduction part.

In order to develop a more time saving way of creating a learning model on WRF, maybe the data clustering techniques can be used. By using data clustering methods we can remove some middle steps which we analyse the log files manually.

10
References

[1] Mesoscale and M. M. Laboratory. (2018). Weather research and forecastingmodel, [Online]. Available:https://www.mmm.ucar.edu/weather-research-and-forecasting-model(visited on 10/20/2018).

[2] Yazıcı İ.M., Karabulut E., Aktaş M.S., "A Data Provenance Visualization Approach", The 14th International Conference on Semantics, Knowledge and Grids, Guangzhou, China., 12-14 Sept 2018, pp.1-8

[3] Simmhan, Y. L. et al., "Karma2: Provenance Management for Data Driven Workflows," to be published in Int'l J. Web Services Research, vol. 5, no. 1, 2008.

[4] Suriarachchi, I., Zhou, Q. and Plale, B., "Komadu: A Capture and Visualization System for Scientific Data Provenance", Journal of Open Research Software 3(1):e4, 2015.

[5] W3C. (2013). PROV-DM: The prov data model, [Online]. Available: https://www.w3.org/TR/2013/REC-prov-dm-20130430/ (visited on 10/25/2018).

[6] W3C. (2013). PROV-O: The prov ontology, [Online]. Available: https://www.w3.org/TR/prov-o/ (visited on 10/25/2018)

[7] Tüfek A. , Gürbüz A., Ekuklu Ö.F., Aktaş M.S., "Provenance Collection Platform for the Weather Research and Forecasting Model", The 14th International Conference on Semantics, Knowledge and Grids, Guangzhou, China., 12-14 Sept 2018, pp.1-8

[8] Chen, P., Plale, B., Aktas, M., S., "Temporal Representation for Scientific Data Provenance", 978-1-4673-4466-1/12/31.00, IEEE, 2012

[9] Logistic regression, [Online].
Available: `https://www.medcalc.org/manual/logistic_regression.php` (visited on 08/12/2018)

[10] Multinomial Logistic Regression, [Online].
Available: https://en.wikipedia.org/wiki/Multinomial_logistic_regression
(visited on 08/12/2018)

[11] Bayes' Theorem, [Online].
Available: https://en.wikipedia.org/wiki/Bayes27_theorem (visited on 09/12/2018)

[12] What is Naive Bayes Algorithm, [Online].
Available: https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/
(visited on 09/12/2018)

[13] Zadrozny B, Elkan C. Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers. In: Brodley CE, Danyluk AP, editors. Proceedings of the 18th International Conference on Machine Learning; San Francisco: Morgan Kaufmann; 2001. pp. 609–616.

[14] The Random Forest Algorithm - Towards Data Science, [Online].
Available: https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd
(visited on 09/12/2018)

[15] Feedforward neural network, [Online].
Available: https://en.wikipedia.org/wiki/Feedforward_neural_network (visited on 09/12/2018)

[16] Multilayer Perceptron Classifier, [Online]
Available: https://spark.apache.org/docs/1.5.0/ml-ann.html (visited on 09/12/2018)

A
UML Class Diagram

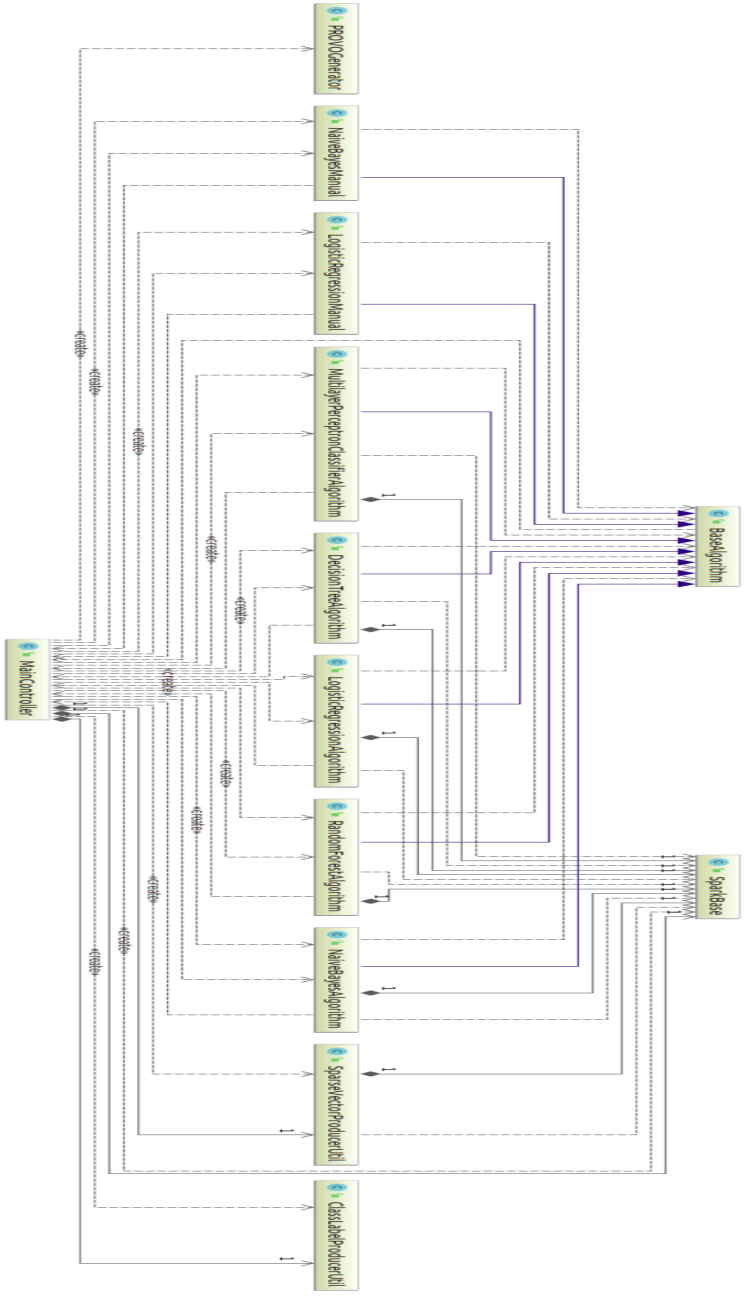

Figure A.1 UML Class Diagram

Curriculum Vitae

FIRST MEMBER

Name-Surname: Erkan Karabulut
Birthdate and Place of Birth: 18.08.1996, İstanbul
E-mail: erkan.karabulut@std.yildiz.edu.tr
Phone: +90 (534) 769 82 71
Practical Training:
(Intern - 2 months) Software Quality Research and Development Laboratory, Yildiz Technical University
(Student Assistant - 2 school terms) Software Quality Research and Development Laboratory, Yildiz Technical University
(Intern - 2 months) IPera Technology Solutions, Research and Development Department
(Part-Time Software Developer - 10 months) IPera Technology Solutions, Research and Development Department

Project System Informations

System and Software: Linux Operating System, Java
Required RAM: 20 GB
Required Disk: 50 GB