UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

SON: PREDICTING THE NATURE OF SERVICE DISRUPTIONS IN CELLULAR NETWORKS

A THESIS

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

MASTER OF SCIENCE IN TELECOMMUNICATIONS ENGINEERING

By

HEMANTH MADABUSHI
Norman, Oklahoma
2016

SON: PREDICTING THE NATURE OF SERVICE DISRUPTIONS IN CELLULAR NETWORKS

A THESIS APPROVED FOR THE
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

BY

_____
Dr. Pramode Verma, Chair

_____
Dr. Gregory Macdonald

_____
Dr. Robert Huck

Acknowledgements

The satisfaction that accomplishing and completing of any task would be incomplete without naming the people who made it possible and whose constant guidance and encouragement made the author seek perfection.

I would like to express my sincere gratitude to my thesis guide **Dr. Gregory MacDonald**, Professor, Department of Telecommunications Engineering, OU for assisting and guiding me throughout the thesis work and also for his valuable teachings in machine earning and networking, suggestions, and comments.

I am also deeply thankful to my thesis advisor **Dr. Pramode Verma**, Director and Professor, Department of Telecommunications Engineering, OU for giving me an opportunity to pursue my master's degree and without whose constant encouragement, the completion of the work would not have been possible.

I am thankful to **Dr. Ali Imran**, professor, Department of Telecommunications Engineering, OU for helping me choose this research and also for his teachings in cellular technologies, which were helpful in the development of this thesis.

I am thankful to **Dr. Robert Huck**, professor, Department of Telecommunications Engineering, OU who agreed to be on my defense committee and for his valuable teachings in networking.

I'm thankful to the researchers whose papers and works have served as resources for me to understand the concerned topic and that motivated me to carry out the thesis work. I wish to express my deep sense of gratitude to my parents for their constant encouragement and support. Finally, I thank my friends who helped me during the course of work.

Table of Contents

List of Tables

List of Figures

Abstract

An important aspect of communication is involved in its cellular network. To meet the demands, communication requires the next generation cellular network, i.e., self organizing networks (SON). In order to implement a self-organizing network, its subsections have to be known and optimized using certain rules. The objective of this document is to deal with one of the subsections called “Self-healing: Fault identification,” in particular by conducting analysis on the Telstra cellular network and predicting its disruptions. First, the prediction of the disruptions can be determined by establishing the machine learning algorithms upon Telstra data. Thus, the classification of faults could be used for finding the nature of the disruptions. Because the appropriate algorithm is chosen by the trial-and-error method, there is no one particular algorithm that fits particular data. Thus, data has to be pre-processed for the algorithms to be applied. Here, the Python Sci-kit module was used as a tool for developing the predictive model. As a note, there are many other tools like R, MATLAB, Rattle, KNIME, etc. that can be used for machine learning. Then, the nature of the faults was identified and investigated to drive customer advocacy.

Chapter 1: Introduction

Let us first take a look at how self-organizing networks (SON) came into existence in 5G cellular networks and how machine learning came into existence. Then I'll proceed to discussing them.

1.1 Genesis of Self-Organizing Networks

Previously, self-organization was used in wireless sensor networks and autonomous computer networks but not in the cellular networks. Self-organization for any system means to have its own *intelligence*, *adaptability*, *agility*, and *scalability*. Self-organization is present in not only the communication field but also in other fields such as mathematics, science, and engineering. Self-organizing proactively is going to be extensively used and will serve as the heart of the future fifth generation (5G) cellular networks.

Ever wonder what is/was life of the legacy networks without self-organization? It probably included more time consumption, more Capex and Opex, more human labor, etc. This is not what people wanted. With growing technology and growing needs, hunger began to make the systems better and better. People wanted the systems to be smart, take commands, and perform the tasks in a much less time. Without self-organization, configuring a base station (BS) parameters to optimize the configured parameters, makes changes in any existing services, detect the faults in cellular network and rectify them requires much time and effort. The systems did not have even the slightest intelligence, and all the tasks had to be done manually. This method could not satisfy people and proved tedious as well.

The quality of service from these was very low compared to the present systems. The travelling distance of control signals between nodes in a cellular network was great. The BS had to wait for the command from the BSC for handover and other processes. Inter-networks and intra-networks cannot cooperate with each other. At least now in LTE systems, there is flat architecture with a limited number of nodes, but before that in GSM, UMTS networks suffered latency issues. There were very limited data services available due to the bitrate constraint. When that type of network failed, fixing the problem required days of waiting. In the future, problem resolution will occur in just a matter of minutes to rectify.

If we go a few years further back, there were no traces of self-organizing even in wireless sensory and computer networks. However, for the 1G, 2G and 3G cellular networks, if we take a look at the 1G or analog networks, they were used only for voice communication. That was sufficient in the previous decades, but, as said earlier, it could not satisfy human needs. The speeds provided by these networks were also a reason for poor quality of service. 2G GSM provided speeds of 14.4 kbps; where as in UMTS, the speed was 2 Mbps on 5 MHz bandwidth. Nevertheless, this is not as bad as it looks since both voice and data were shared on the 5 MHz bandwidth. UMTS failed to provide feature rich multimedia service.

Structural limitation of legacy networks and lack of available resources were the main reasons for not having the self-organization in these networks. In between there were GPRS and EDGE networks for the sole purpose of improving data services. They provided data rates of around 160 kbps and 400 kbps, respectively, which were still insufficient. Currently, HSPA+ and LTE networks are providing competition and

acting as the ground work for the full development of an autonomous intelligent 5G cellular network. Here in LTE networks the number of nodes have been decreased, thereby decreasing the signaling distance and, eventually, latency. It has been reduced to 100 ms to 10 ms in present LTE networks. There are things like time transmission interval (TTI) and grant requests that can be reduced. So, we can say that there is little intelligence or self-organization in LTE. We can say that this is the introduction phase for the SON in 5G.

In legacy networks things were completely manual, and now in LTE it is still manual labor with little introduction of the self–0rganisation intelligence. The old systems were weak and fragile. The systems like LTE are in reactive state, which needs to shift to the proactive state with agility. The next chapter discusses these characteristics of SON in detail. But these are the scenarios that led to the development of intelligent self-organizing systems in cellular networks apart from sensory and computer networks.

1.2 Genesis of Machine Learning

We saw that the main reason for the SON's existence was the growing need for multimedia services. In this section, the pillar for Machine learning is *data*. Along with the growing population, data also began increasing tremendously in a way that no one could ever imagine. Statistics of International Data Corporation (IDC) shows that there were 1.8 zettabytes of data in the year 2011 itself. Facebook takes up approximately 1 petabyte of storage, Large Hadron Collider near Geneva, Switzerland, which is the world's largest and most powerful particle accelerator produces approximately 15

petabytes of data/year. The list goes on… So, we collect the data to find the interesting and hidden facts in it. For this sole purpose, there are many platforms and analytics, e.g., Hadoop, Spark, Machine Learning, SQL queries.

"*We are drowning in data, but starving for knowledge."*

- *John Naisbitt*

The collected data cannot be used readily for analysis purposes. It has to be transformed and cleaned. This is called as data pre-processing. Once it has been processed, it is loaded into the above-referenced platforms and obtain the required results. You might be wondering what we do analyzing all this data? It can be used for various purposes such as to predict the customers and grow the business. Social media can look at the data models and introduce new features to grow business. It is used in almost every field like medicine, engineering, banking, etc. In this situation, collecting data means not collect the actual information so there is no violation of human privacy. Just the statistical numbers are collected to satisfy the objective of growing the business. So, data analysis is a win-win situation for both producers and consumers. Each and every field has some form of data requested and collected from people. Similarly, knowingly or unknowingly, everyone is contributing data through a medium called the internet. These are stored in the form of internet archives in the cloud services and servers.

There are a lot of security protocols in practice that preserve people's privacy. The negative side of that fact is that hackers who try to steal data misuse the security protocols. Hackers' purpose is to try and break the security algorithms. However, another strong algorithm will be developed to counter the hackers' efforts. So, there's a

tug-of-war going on between good and bad between internet hackers and security developers. No need to worry because current algorithms have been developed that are so strong that it is almost impossible to break them. Even with the help of super computers, it would take more than a lifetime. So, there are both positive and negative aspects of this data. But, it can serve as a very powerful tool once harnessed. Previously, there was big data to analyze these huge amounts of data. But again, human hunger came into action and there was need for something more than just analysis. They needed predictions and for this we need *artificial intelligence*-based algorithms and machines. Therefore, algorithms also began to become complex or, in other terms, we can say that the analysis maturity level increased from raw data to predictive modelling. In the present generation, both big data and machine learning are used together to find the insights. We shall discuss about Machine Learning and the applications involved in detail later in the chapter 3. The terms *machine learning* or *data mining* or *advanced analytics* mean the same mining of interesting or previously unknown knowledge. The SQL tools can be used for *surface* analysis of the data, Statistical methods can be used for the *shallow* data analysis, whereas machine learning should be used for *hidden* data analysis. Hence, machine learning and its algorithms came into existence. This is not data warehousing, query processing. They need to be implemented on technologies like Python, R, etc. Some of the machine learning tools are Weka, Knime, Orange, Rapid Miner, Rattle, Mahout, etc. So, now it may seem obvious how powerful this *c* data can be when transformed with proper intelligence-based algorithms.

Chapter 2: Self Organizing Networks (SON)

Self Organizing makes the networks flexible and Quality of Service (QOS) achievable. The other technologies such as Densification, Control Data Separation Architecture (CDSA) help SON to increase *capacity* and *energy efficiency*. SON will be able to reduce the OPEX, eliminating human labor.

2.1 Understanding SON

2.1.1 Scalability

For example, consider an algorithm that can change the antenna tilts for Load Balancing (LB) and Coverage and Capacity Optimization (CCO). As the number of antennas (or of antenna tilts?) increases, the complexity of the system increases for the co-ordination. This does not make the network Self Organizing. So, for a system to satisfy this scalability property, the increase in the size of system should be inversely proportional to the complexity.

2.1.2 Stability

A system should change itself from one state to another in a finite number of times and come back to being stable. If it does not satisfy the finite condition and keeps oscillating infinitely, then it cannot be considered stable.

2.1.3 Agility

As much as agility is important for self organizing systems, it is also important to have moderate or correct agility. These changes should neither be too fast nor too slow. Hence, this state is somewhat difficult to achieve.

There are different types in the Self Organizing Networks. They are:

i. Self-Configuration

ii. Self-Optimization

iii. Self-Healing

This research work comes under the subtype, Self-Healing, which predicts errors proactively.

2.2 Self Configuration

In 5G networks, the SON feature is expected to possess self-configuration. The cells will automatically configure the radio parameters, IP address, and a neighbor cell list, which is needed whenever a new cell is added or an existing cell has an issue. Configuring eNodeBs (eNB) requires small cells at the time of deployment or upgrade of the network or when there are any faults in the network. This self-configuration replaces manual configuration and saves lot of time. The Next Generation Mobile Network (NGMN) group of 2006 determined many use cases for SON. Instead of PDN-GW assigning the *IP address*, the UEs can request the DHCP servers and get their IP address. This saves the initial configuration time in future networks. By communicating and co-operating with neighboring cells, the eNB will be able to compute its own *physical cell ID*. In the same process, it can add/delete any number of neighbors. The *RF parameters* such as interference, tilt, frequency, propagation, and transmitted power to the UE will be configured by the UE on its own in the 5G networks. Researchers are investigating whether the eNBs can support the *self-test* mechanism, leaving the validation work for humans. The future eNBs will be able to *authenticate* themselves

with the MME instead of piggybacking the NAS messages with RRC messages. In this self-configuration, there are parameters that influence neighboring cells as well their own cells. The policies for these configurations will be set by the operators such as ATT, Verizon, etc. The operators present a framework which is important and serves as the basis for all configuration changes.

2.3 Self Optimization

With the increase in technologies, optimizing and maintaining high quality networks for customers has never been more difficult. Sometimes we need to choose between maintaining high level of quality and being cost-effective. This Self-Optimization in the networks translates into little improvement. Otherwise, Self-Optimization automatically restores the original parameters and tries something else. Self-Configuration has to be followed by the Self-Optimization for the systems to perform efficiently. In this, Load Balancing (LB) and Coverage and Capacity Optimization (CCO) are the primary focus. Optimum distribution of users among cells based on cells throughput is called *load balancing*. Whereas, when the received minimum downlink power of each user is above a certain threshold, that state is called CCO. These both can be formulized using the Shannon capacity equation by substituting *tilts and cell individual offsets (CIO)* in the SINR model. Load balancing can be achieved with antenna adaption, power adaption, and hybrid of both antenna and power. *Neighbor Optimization* includes automatic detection and deletion of neighboring cells. Followed by this the eNB computes its cell ID (part of self-configuration). By coordinating with neighboring cells on their power levels, *interference* can be reduced.

Handoff optimization can be achieved by monitoring the KPIs, which promote automatic inter and intra handoffs. *Energy savings* can be automated using the cell-on-demand approach, which turns the base stations on only when they are needed. So, self-optimization all boils down to the optimization of antenna parameters, transmit power, and frequency reuse.

2.4 Self Healing

Self-healing involves automatic fault detection by diagnosis and fault correction by recovery actions. This research is based mainly on the automatic fault detection subpart of self-healing, which, in turn, is a subtype of the self-organizing networks. For self-healing to occur, we need artificial intelligence algorithms, which machine learning provides and is discussed in the next chapter. So, focusing on the self-healing, it performs network *maintenance and updates*.

The following are the recommendations of the Next Generation Mobile Networks (NGMN) group. The faults, or *outage*, in the cells should be detected instantly. The network should then reconfigure the neighboring cells to compensate for the failed cell for the radio resources. Performing *equipment traces* is a troubleshooting activity. Many self-healing functions are coming into picture at a very slow rate. The *relay stations* were being used as self-healing agents at the starting stages of self-healing. They were robust and reliable. They can route traffic to neighboring cells' eNBs when the existing eNB fails. We detect the outages when there is a drop in the performance of the network or its components. These outages can be categorized as little, medium, or critical faults (where the network is completely down). Bayesian

analysis predicts the fault probability in the system using other features' values. The accuracy of the algorithm will depend on the training of the data and the number of classifications of the fault severity.

Another way of classifying network faults is the pattern recognition method. But whatever may be the method, there still might be a few outliers undetected because of the working conditions and complexity of the algorithms. Compensation time depends on the severity of the faults detected. The neighboring cells would take reconfiguration steps to increase/decrease antenna tilts, transmission power, etc. Table 1 summarizes the tasks of the Self Organizing Network (SON) types:

Table 1. Overview of the SON functions

Self-Configuration	Self-Optimization	Self-Healing
Cell ID	Neighbor Optimization	Fault detection
RF Parameters	Handoff	Fault Compensation
Self-test	Interference	Equipment traces
Self-authentication	Energy Savings	Analysis

Chapter 3: Machine Learning

With the development of modern society and the increase in population, data is growing tremendously, causing cellular networks to be prone to faults. There is a need for artificial intelligence in cellular networks. This can be achieved through machine learning algorithms and optimization. One of the famous machine learning definitions is a computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E. An algorithm is a sequence of instructions given to a computer to solve a problem. However, there are no algorithms for some tasks.

For example, differentiating spam e-mails from legitimate e-mails does not have a ready-made algorithm for sorting that type of data. In this case, we would like the computer to *learn* and extract output automatically.

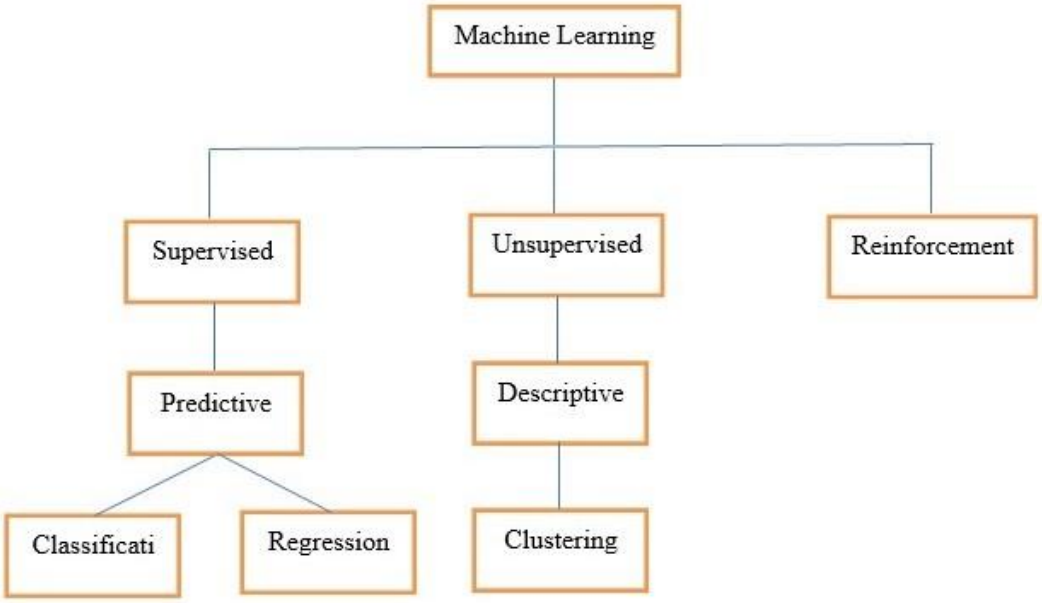
With the increase in stored data from terabytes and petabytes to zettabytes and exabytes, computer technology to access to huge amounts of data stored long distances from the user is available. Also, we have technology to store and process large amounts of data. Application of machine learning to large databases is known as data mining. The terms *machine learning, predictive analytics, advanced analytics and data science* are sometimes used interchangeably, which is okay. So, machine learning can be said to be a database problem combined with artificial intelligence. SQL queries are used for simple queries and reporting. Machine learning is used to dig deep into data stores to find hidden patterns that maybe were unknown. What can be *hidden* in data? Examples are associations, sequences, classifications, forecasts, anomalies, grouping/clustering.

The environments or inputs/outputs keep changing constantly, and the machines that can adapt to these changes reduce the need for re-design. Machine learning is more like an on-the-job improvement. Hence, machine learning can be said to be the collection of various studies like statistics, brain models, artificial intelligence, control theory, etc. Basically, the machine learns the structure of some sort. Some of the tasks involved in machine learning are prediction, diagnosis, grouping, etc. How does machine learning work? The computer analyzes data, finds patterns, and performs predictions. These activities can be called categories of machine learning models.

Predictive methods use some of data features to predict some unknown or future value of other features. We are trying to predict some value of interest like fault severity. Some of features might be descriptors as well. The machine learning function finds human interpretable patterns that can describe the data. We can look at this model and gain knowledge of its characterization or how it is getting some of its general properties. Some of the algorithms might have both properties.

Other interesting topics of discussion in machine learning is supervised learning, unsupervised learning, reinforcement learning. The classification and regression belong to the supervised learning category, predictive methods.

Figure 1. Machine Learning Tree Diagram



Classification means predicting yes/no. In a banking, when the bank loans an amount to a customer, the bank has to predict the risks associated with the customer's ability to repay the loan. The bank analyzes the credit score of the customer to see if he will repay the amount promptly within the given timeline. These predictions are based on the information stored about the customer, namely past payment history, income earned, savings amount, etc. This is called *training the algorithm using past data*. Predicted output here will be either 1/0. Sometimes we might have to calculate probabilities given the attributes of the customer.

Like the banking system, other examples for classification analysis could be pattern recognition, face recognition, speech recognition, medical diagnosis, biometrics used to authenticate people based on their physical characteristics. Training gives us knowledge about the data. In turn, the data can be compressed or normalized according to one's situation.

Regression means the output is a number, for example, predicting the price of cars. In predictive methods, a training dataset is typically provided with labeled examples. Like a teacher correcting student answers, we use that training data to train our models and find patterns. So our input features are provided along with the target class feature, and our goal is to predict the target class.

Clustering belongs to the *unsupervised learning* category, descriptive methods. Here there is no teacher. The data is provided with no knowledge of any information except the data itself. So, we have no information about the hidden patterns contained within the data. There are no target values provided for each of our instances, so we say the data is unlabeled. The goal here is to find naturally occurring patterns or groupings or clustering or segmentations. After these groupings are made, it is easy to assign policies, provide services and, thereby, improve business strategies. Examples of Clustering could be image clustering, document compression, molecular biology, and biometrics.

In *Reinforcement learning*, the output is a series of actions, rather than a single action, for example, playing a chess game. The game has simple rules but complex possibilities. A single move by itself is not important. What matters is a sequence of moves. Robot navigation could be another example of this reinforcement learning.

There are lots of new algorithms in the machine learning subsections, as previously discussed, that make machine learning a trial–and-error method of analysis. That is, there is no way to tell which algorithm best fits specific data. However, one can decide whether the data belongs to Supervised or Unsupervised category. But, after determining the data's category, one needs to know how the algorithm works, and then

prepare the data has to fit the algorithm. Even at this stage, we might learn that the algorithm is not performing as expected. That is the disadvantage of using machine learning. Once an algorithm fails to deliver the expected results, the algorithm must be modified, or a new algorithm developed and tried, or the dataset must be reformatted.

Machine learning is applied in science, including chemistry, physics, medicine, pharmaceuticals, health care, energy, smart city, financial industry, E-commerce, market analysis, risk analysis, and sports and entertainment. In hospitals, it is used to classify and treat cancer, tumors, and diseases. Banks use it to determine loan risk. Call centers use to apply inferred relationships to prevent churn. Telecommunications use it to analyze call records to optimize the network and improve the quality of service to the customers. Airlines use it to predict passengers who are likely to miss their flights to predict an overbooking number in hopes of increasing revenue.

Chapter 4: Thesis Description and Data Preparation

4.1 Problem Statement Description

Telstra is one of the Australia's largest telecom network operators. The goal is to predict the severity of service disruptions on its network using a dataset of features from its service logs. We need to determine whether the disruption is a momentary glitch or total interruption of network connectivity. By predicting the faults beforehand, Telstra will be able to serve its customers more reliably.

Fault severity is divided into three categories: 0, 1, and 2., i.e., 0 meaning no fault, 1 meaning only a few faults, and 2 meaning many faults. There are two main datasets provided namely train.csv and test.csv. We need to use the train data to train our predictive machine learning algorithms and then test the model on the provided test data. So, as said earlier in chapter 3, this train data acts as a teacher guiding the students (test data).

Different data files and their descriptions are provided in Table 2.

Table 2. Data Files

File Name	Description
severity_type.csv	Severity warning message from logs
train.csv	Training set for fault severity
test.csv	Test set for fault severity
log_feature.csv	Features extracted from log files
resource_type.csv	Type of resource related to main data (train, test)
event_type.csv	Events related to main data

4.2 Data Pre-Processing using Python

The foremost step before applying whatever the given data, the data needs to be changed into a form suitable for the algorithm. This can be done only once, so we have a thorough understanding of how the algorithm works. In this section, this paper deals with the data preprocessing used and the 'Sci-kit' module for machine learning in Python. Some of the essential Python libraries used in data preprocessing are numerical Python or Numpy – Package for scientific computing and deals with arrays, Pandas. Another package of Python used mainly for data frames and series, Matplotlib that produces 2D plots and visualizations, Scipy that performs statistical analysis, and Seaborn that si another data visualization library. In this work with Telstra data with SVM, Interactive Python, or IPython, was used for coding in Python 2.7.

Preparation means cleaning, combining, normalizing, reshaping, slicing of data for data analysis. The Telstra data consisted of 6 different datasets in total along with the Train and Test datasets. There were no any NULL items which did not require cleaning but in case if they do in your data, then they might require cleaning. All these data sets had 'id' column as primary key in them. So, first step I did was to combine or merge them on 'id' primary key with 'left' join. There is a concept of 'Feature Scaling' in machine learning. The idea is that if we have a problem with multiple features make sure that the features have similar range of values. This will be useful for gradient descent to converge more quickly to the minimum of optimization problem. This can be achieved in data preprocessing by normalization which is given by:

$$\square orm = \frac{x_i - mean\ of\ feature}{Range\ of\ feature}$$

Where, $x_i = intance\ of\ feature\ (column)$

$$Range = maximum\ value - minimum\ value$$

Reshaping of the data in python can be done either by using pivot, dummies etc. where the instances are reshaped into features and transformed into binary values of 0 or 1. i.e. value ‘1’ for the corresponding ‘id’ and ‘0’ elsewhere. This is also called as binarization. This idea will be highly useful for Regression algorithms and sometimes categorical values.

The library which includes all the methods for preprocessing is sklearn. preprocessing. These are some of the things used in my data processing model. But, these will be different for different data and algorithms. So, this should serve as an idea to kick start while you work with your own data sets and algorithms.

4.2.1 Program

```
#Imports
import numpy as np
import pandas as pd
from pandas import Series, DataFrame
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
#Load data
train = pd.read_csv('dataset/train.csv')
test = pd.read_csv('dataset/test.csv')
event_type = pd.read_csv('dataset/event_type.csv')
```

```
log_feature = pd.read_csv('dataset/log_feature.csv')

resource_type = pd.read_csv('dataset/resource_type.csv')

severity_type = pd.read_csv('dataset/severity_type.csv')

train.head()

#Concatenate test and train

train['source'] = 'train'

test['source'] = 'test'

data = pd.concat([train,test],ignore_index=True)

data.head()

print data.shape

data.describe()

#Count of each category

data['fault_severity'].value_counts()

# Now we merge the dataframes

# We can choose which DataFrame's column to use, this will choose left

data1 = pd.merge(data,event_type, on='id', how='left')

data1.head()

print data1.shape

data2 = pd.merge(data1,log_feature, on='id', how='left')

data2.head()

print data2.shape

data3 = pd.merge(data2,resource_type, on='id', how='left')

data3.head()
```

```
print data3.shape

data4 = pd.merge(data3,severity_type, on='id', how='left')

data4.head()

print data4.shape

# We now drop duplicates and rearrange columns

data5 = data4.drop_duplicates()

data5 =

DataFrame(data5,columns=['id','location','source','event_type','log_feature','volume','res

ource_type','severity_type','fault_severity'])

data5 = data5.drop('source',axis=1)

data5.head()
```

```
In [22]: data5.head()
```

Out[22]:

	id	location	event_type	log_feature	volume	resource_type	severity_type	fault_severity
0	14121	location 118	event_type 34	feature 312	19	resource_type 2	severity_type 2	1
1	14121	location 118	event_type 34	feature 232	19	resource_type 2	severity_type 2	1
2	14121	location 118	event_type 35	feature 312	19	resource_type 2	severity_type 2	1
3	14121	location 118	event_type 35	feature 232	19	resource_type 2	severity_type 2	1
4	9320	location 91	event_type 34	feature 315	200	resource_type 2	severity_type 2	0

```
data5.to_csv(path_or_buf='dataset/mergedData1.csv')
```

Chapter 5: Algorithms – Analysis and Implementation

5.1 Support Vector Machines (SVM)

One of the most popular classification algorithms is the Support Vector Machine (SVM). This is the algorithm which was used on the Telstra data to classify the faults. So, going forward, we will discuss about this algorithm in this paper. SVM has many applications such as the digit recognition, pattern recognition, text classification etc. SVM consists of two cases namely Separable case and Non-Separable case. As the name, itself indicates, Separable data is the one which can be linearly separated. Let us imagine that, we need to classify our data into two types and separate them. This separation is called Margins and they need to be as large as possible (Wide road). We call this concept as Maximal Margin Hyperplane. In 2-D this margin is just a line and in multi dimension this is called Hyperplane. Non-Separable data is the one which cannot be separated linearly. In such cases, we use 'Kernel' trick where the trick is to transform them into another space and slice the data with hyperplane.

Before we get into the details of SVM there are few concepts which needs to be understood like Hypothesis, Cost function, Optimization etc. These are concepts are almost similar to Regression algorithms also. Firstly, we take our training data, apply our learning algorithm. Then it is the duty of the learning algorithm to output a function 'h' called the *hypothesis*. This hypothesis takes the input variables or features (x) and output the estimate value of the target class (y). It can also be called as mapping function from x's to y's. In simple linear case this is represented as below:

$$h_\theta(x) = \theta_0 + \theta_1 x$$

Where, $\theta_0, \theta_1 = parameters$. Imagine this as a line equation $y = c + mx$.

To formalize this, we need the difference between the expected value, h(x) and actual vale, y to be small. Then we sum this difference over all the instances (rows) of the training set. This gives us the *cost function* and mathematically this can be represented as below:

$$J(\theta_0,\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x_i) - y_i)^2$$

Where, $x_i, y_i = value\ of\ instances\ for\ 1\ feature$. We will denote $x_i^i, y_i^i\ for\ multiple\ features.$

$$\min_{\theta_0,\theta_1} J(\theta_0,\theta_1) = Optimization\ problem$$

Minimizing cost function over these parameters is called the optimization. If we are going to find the extremum of a function, then we have to use Lagrange multipliers. This gives us new expression to maximize or minimize without thinking about constraints. Another interesting concept to know is about bias vs variance and overfitting vs under fitting. If the margin does not fit the training data very well, then we call it as 'under fitting' or having 'high bias'. This usually appears in data with less parameters and features. If the margin fits through all our training instances, then it is called as 'overfitting' or having 'high variance'. This case appears in higher order polynomials where the parameters and features are high.

5.1.1 How it works?

(i) Linear SVM: Separable case

In our SVM algorithm let us denote these parameters $\theta_0, \theta_1$with b, w. Also, $\boldsymbol{x_i} = (x_i^1, x_i^2, \ldots x_i^n)^T,\ y \in \{1, -1\}$. The decision boundary for classification can be written as:

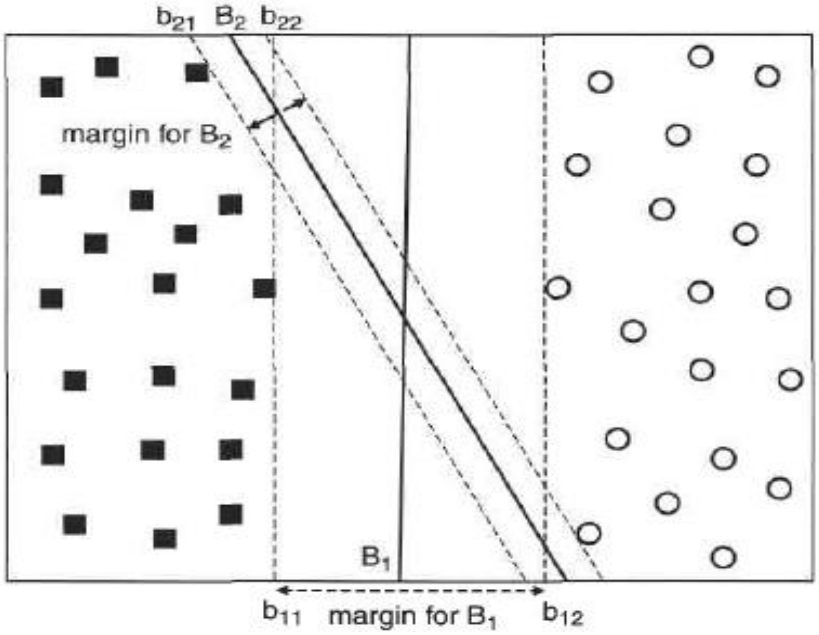
$$\boldsymbol{w}.\boldsymbol{x} + b = 0$$

As mentioned earlier in introduction, classification algorithms output either 1/0 or yes/no or positive/negative. Let us consider two points namely $x_p, x_n$for positive and negative points present in our training data. If any two points $x_p, x_n$are present on the boundary, then they can be written as:

$$\boldsymbol{w}.\boldsymbol{x_p} + b = 1 \qquad \rightarrow (1)$$

$$\boldsymbol{w}.\boldsymbol{x_n} + b = -1 \qquad \rightarrow (2)$$

Figure 2. Separable Linear SVM



For the points, above or below the decision boundary line, these equations become

$$\boldsymbol{w}.\boldsymbol{x_p} + b > 1 \; and \; \boldsymbol{w}.\boldsymbol{x_n} + b < -1$$

The width of the boundary is given by the distance between the two hyperplanes. Let us consider a point x_p on equation (1) and a point x_n on equation (2). Subtracting (1) and (2) we get

$$\boldsymbol{w}.(\boldsymbol{x_p} - \boldsymbol{x_n}) = 2$$

$$\|\boldsymbol{w}\|.d = 2$$

$$WIDTH, d = \frac{2}{\|\boldsymbol{w}\|} \; is\ the\ maximum\ margin\ between\ hyperplanes$$

The train phase of the SVM is where the algorithm learns. This involves estimating the parameters w and b of the boundary from train data. The conditions for choosing these parameters are given below:

$$\boldsymbol{w}.\boldsymbol{x_i} + b \geq 1 \quad if\ y_i = 1$$

$$\boldsymbol{w}.\boldsymbol{x_i} + b \leq 1 \quad if\ y_i = -1$$

These equations mean that all the instances belonging to category y = 1 must be on or above the decision boundary $wx_p + b = 1$. All the instances belonging to category y = -1 must be on or below the boundary $wx_n + b = -1$. Both these conditions can be summarized as:

$$y_i(\boldsymbol{w}.\boldsymbol{x_i} + b) \geq 1\ where\ i = 1,2,3 \ldots n \qquad \rightarrow (3)$$

Additionally, SVM has another condition that the margins should be maximized which is equal to minimizing the below function by Lagrange.

$$f(\boldsymbol{w}) = \frac{\|\boldsymbol{w}\|^2}{2} \qquad \rightarrow (4)$$

Optimizing this function gives us the Lagrange dual optimization solution for SVM. We write the objective function by substituting equation (3) and the constraint into Lagrange

$$L = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{n} \lambda_i [\, y_i(\boldsymbol{w}.\boldsymbol{x_i} + b) - 1] \qquad \rightarrow (5)$$

Where, $\lambda_i = Lagrange\ Multiplier$

$y_i(w.x_i + b) - 1 = \text{Inequality constraint}$

To find the extremum, we have to take the derivative with respect to w and b and set them to zero. Doing so, we get the below equations:

$$\frac{dL}{d\boldsymbol{w}} = \boldsymbol{w} - \sum_{i=1}^{n} \lambda_i\, y_i x_i = 0 \;\Rightarrow \boldsymbol{w} = \sum_{i=1}^{n} \lambda_i\, y_i \boldsymbol{x_i} \qquad \rightarrow (6)$$

$$\frac{dL}{db} = -\sum_{i=1}^{n} \lambda_i y_i = 0 \;\Rightarrow\; \sum_{i=1}^{n} \lambda_i y_i = 0 \qquad \rightarrow (7)$$

Substituting equations (6) and (7) in equation (5) and simplifying we get

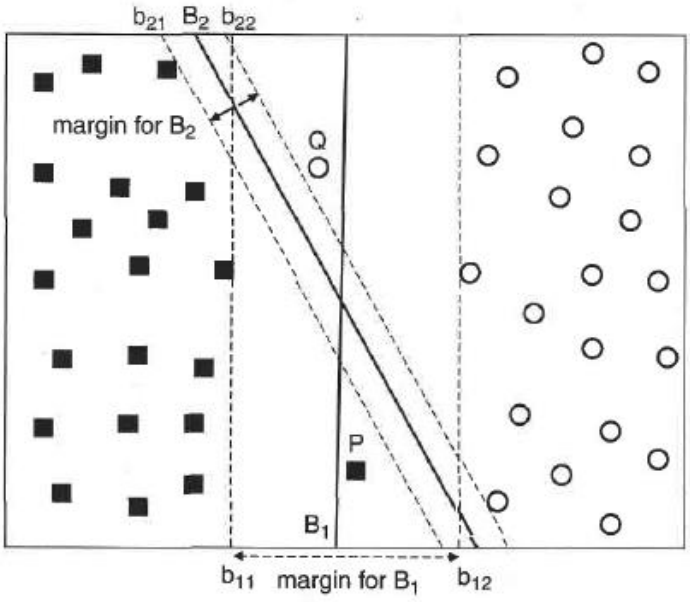
$$L_D = \sum_{i=1}^{n} \lambda_i - \frac{1}{2}\sum_{i}\sum_{j} \lambda_i\, \lambda_j y_i y_j \boldsymbol{x_i}.\boldsymbol{x_j} \qquad \rightarrow (8)$$

This is the final Dual Lagrange solution for SVM. The difference between the primal and dual Lagrangians is that the dual Lagrangian consists of Lagrange multiplier, $\lambda_i$and the train data. Whereas the primary Lagrangian consists of Lagrange multiplier and the decision boundary parameters.

(ii) Linear SVM: Non-Separable case

Previous case is applicable to error-free case. In this section, SVM constructs the decision boundary where there are little errors (not linearly separable) as shown in the below figure. The margin should be tolerable to small training errors and is called *Soft margin*.

Figure 3. Non-Separable SVM



Here, we can see that there are few samples of squares and circles lying in the decision boundary. In such cases, there will be a tradeoff between width and the training errors in the boundary. While the objective function in eq. (4) still is applicable, the constraints in eq. (3) no longer satisfies the given condition. Therefore, a slack variable (ξ) is introduced into the constraints to relax the inequality and this is of positive value. The conditions now become:

$$\boldsymbol{w}.\boldsymbol{x_i} + b \geq 1 - \xi_i \text{ for } y_i = 1$$

$$\boldsymbol{w}.\boldsymbol{x_i} + b \leq -1 + \xi_i \text{ for } y_i = -1$$

Where $\xi_i > 0 \; \forall i$

The modified objective function is given by:

$$f(\boldsymbol{w}) = \frac{\|\boldsymbol{w}\|^2}{2} + C(\sum_{i=1}^{N} \xi_i)^k \qquad \rightarrow (9)$$

Let k=1 for simplification, C is user specific parameter and can be chosen according to the performance of the model.

The Lagrangian for the constrained optimization can be written as:

$$L_p = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{n}\lambda_i[\,y_i(\boldsymbol{w}.\boldsymbol{x}_i + b) - 1 + \xi_i] - \sum_{i=1}^{N}\mu_i\xi_i \quad \rightarrow (10)$$

Here, the first two terms are objective function to be minimized. Third term is the inequality constraint consisting slack variables. Fourth term is the non-negative requirement of ξ. The third term, inequality constraints can be transformed into equality constraints by using KKT conditions.

$$\xi_i > 0,\ \lambda_i > 0,\ \mu_i > 0$$

$$y_i(\boldsymbol{w}.\boldsymbol{x}_i + b) - 1 + \xi_i = 0$$

$$\mu_i\xi_i = 0$$

Taking first-order derivative with respect to $\boldsymbol{w}$, b, ξ_i and setting it to zero, we get

$$\frac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_{i=1}^{N}\lambda_i y_i x_{ij} = 0 \Rightarrow \boldsymbol{w} = \sum_{i=1}^{N}\lambda_i y_i x_{ij} \qquad \rightarrow (11)$$

$$\frac{\partial L}{\partial b} = -\sum_{i=1}^{N}\lambda_i y_i = 0 \qquad \rightarrow (12)$$

$$\frac{\partial L}{\partial \xi_i} = C - \lambda_i - \mu_i = 0 \Rightarrow \lambda_i + \mu_i = C \qquad \rightarrow (13)$$

Substituting eq. (11), (12) and (13) in Lagrangian eq. (10) we get

$$L_D = \frac{1}{2}\sum_i\sum_j \lambda_i\,\lambda_j y_i y_j \boldsymbol{x_i}\boldsymbol{x_j} + \mathrm{C}\sum_i \xi_i - \sum_i \lambda_i \left\{ y_i \left(\sum_j \lambda_j y_j \boldsymbol{x_i}\boldsymbol{x_j} + b \right) - 1 + \xi_i \right\}$$

$$- \sum_i (C - \lambda_i)\xi_i$$

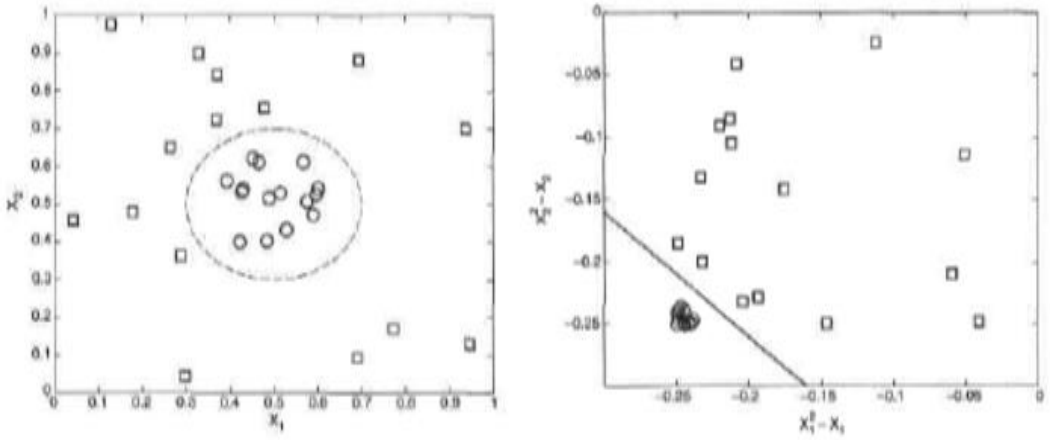
$$= \sum_{i=1}^{N}\lambda_i - \frac{1}{2}\sum_i\sum_j \lambda_i\,\lambda_j y_i y_j \boldsymbol{x_i}\boldsymbol{x_j} \qquad \rightarrow (14)$$

The above equation which we got is similar to the eq. (8). In linear separable case, Lagrange multipliers $\lambda_i > 0$. Whereas, in non-Separable case, $0 \leq \lambda_i \leq C$

(iii) Nonlinear SVM:

In this section, we will see how to apply SVM to nonlinear decision boundaries. The trick is to transform the data from **x** to a new space called $\Phi(\boldsymbol{x})$ such that model can be simplified and a linear decision boundary can be applied. The data from original feature space is mapped onto a new space.

Figure 4. Non-linear SVM



Nonlinear SVM can be formalized as the below optimization problem:

$$\min_{\boldsymbol{w}} \frac{\|\boldsymbol{w}\|^2}{2}$$

subject to $y_i(\boldsymbol{w}.\Phi(\boldsymbol{x}_i) + b) \geq 1, \qquad i = 1,2,\ldots N$

The main difference here is that instead of taking attribute **x**, learning is performed on the transformed $\Phi(\boldsymbol{x})$**.** So, based on the previous approach taken for linear case, writing primal Lagrangian and setting the derivative to zero, the parameters **w** and b can be derived using the following eqns.

$$\boldsymbol{w} = \sum_{i=1}^{N} \lambda_i y_i \Phi(\boldsymbol{x_i}) \qquad \rightarrow 15$$

$$\lambda_i \left\{ y_i \left(\sum_j \lambda_j y_j \Phi(\boldsymbol{x_i}).\Phi(\boldsymbol{x_j}) + b \right) - 1 \right\} \qquad \rightarrow 16$$

we can derive Dual Lagrangian as

$$L_D = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_i \sum_j \lambda_i \, \lambda_j y_i y_j \Phi(\boldsymbol{x_i}).\Phi(\boldsymbol{x_j}) \qquad \rightarrow (17)$$

Dot product between the vectors in the transformed space could be difficult as we move to higher dimensions and the solution is **kernel trick.**

The Kernel trick is a method of computing similarity in the transformed space using the original attributes. For example, the dot product between vectors u, v in the transformed space can be written as:

$$\Phi(\boldsymbol{u}).\Phi(\mathbf{v}) = \left(u_1^2, u_2^2, \sqrt{2}u_1, \sqrt{2}u_2, 1\right).\left(v_1^2, v_2^2, \sqrt{2}v_1, \sqrt{2}v_2, 1\right)$$

$$= u_1^2 v_1^2 + u_2^2 v_2^2 + 2u_1 v_1 + 2u_2 v_2 + 1$$

$$= (\boldsymbol{u}.\boldsymbol{v} + 1)^2$$

$$K(u, v) = \Phi(\boldsymbol{u}).\Phi(\mathbf{v}) = (\boldsymbol{u}.\boldsymbol{v} + 1)^2 \qquad \rightarrow (18)$$

Mercer's principle ensures that kernel functions can be always expressed as dot product between two input vectors in some high dimensional space

Mercer's Theorem:

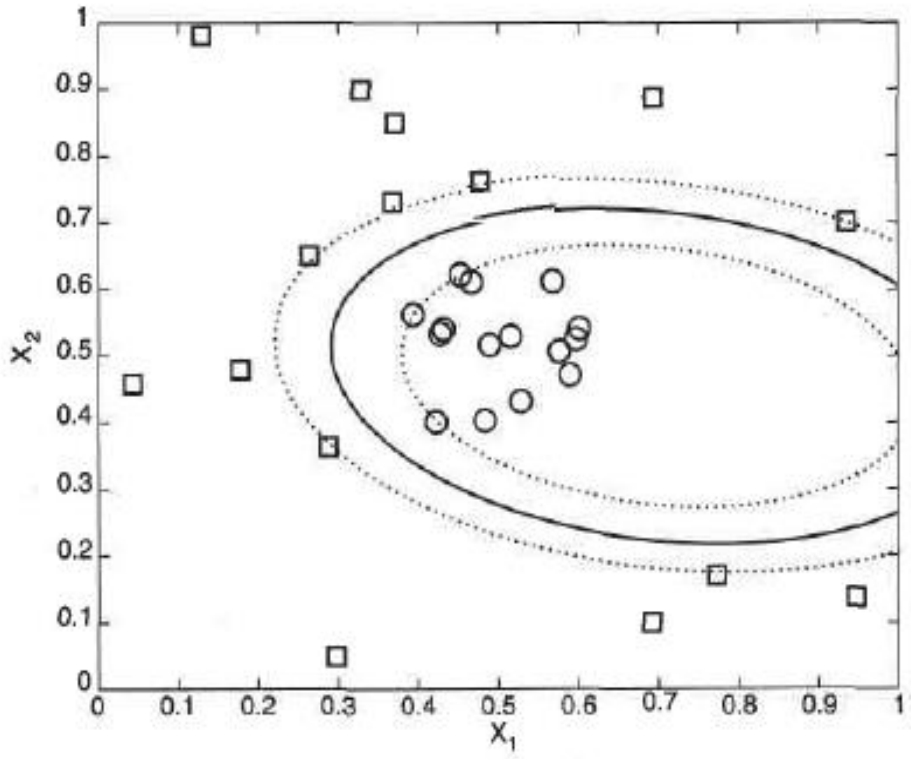
A kernel function K can be expressed as

$$K(u, v) = \Phi(\boldsymbol{u}).\Phi(\mathbf{v})$$

if and only if, for any function of $g(x)$ such that $\int g(x)^2 dx$ is finite, then

$$\int k(x,y)\ g(x)g(y)dx\ dy \geq 0$$

Figure 5. Nonlinear decision boundary obtained by SVM using polynomial kernel function



5.1.2 Log-Loss Calculation

Log loss is called as Logarithmic likelihood function of a Bernoulli random distribution. This is an error metric used when the constraints have to predict something as True/False with probability (likelihood) of 1 (true) to 0.5 (equally true) to 0 (false). If a prediction is true (1) when it is actually false (0) i.e., for wrong prediction the punishment will add ∞ (large number) to your error score. So, log loss closer to zero means more accurate is your prediction. This can be formulated in a mathematical way as below:

$$\log loss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} \log (p_{ij})$$

Where, N = Number of observations

M = Number of class labels

$$y_{ij} = \begin{cases} 1, & \text{if observation i is present in class j} \\ 0, & \text{otherwise} \end{cases}$$

p_{ij} = Predicted probability that observation i is in class j

The SVM model was run in "WEKA", popular suite of machine learning software written in Java. The data set was sent through the pre-process stage. Then in the classification algorithms SVM is selected and run as 66-34 % split because we need to train the data on 66% split and test the data to check and calculate our log loss.

Figure 6. Sample data for SVM

id	location	event_type	log_feature	volume	resource_type	severity_type	fault_severity
14121	L118	E34	F312	19	R2	S2	One
14121	L118	E34	F232	19	R2	S2	One
14121	L118	E35	F312	19	R2	S2	One
14121	L118	E35	F232	19	R2	S2	One
9320	L91	E34	F315	200	R2	S2	Zero
9320	L91	E34	F235	116	R2	S2	Zero

Using SVM our output and confusion matrix looks like below:

=== Run information ===

Scheme: weka.classifiers.functions.SMO -C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions.supportVector.PolyKernel -E 1.0 -C 250007"

Instances: 61839

Attributes: 7

Correctly Classified Instances 15966 75.9382 %

Incorrectly Classified Instances 5059 24.0618 %

a	b	c	Classified as
3466	1519	671	a = Zero
1384	10447	586	b = One
590	309	2053	c = Two

We predicted 3466+1384+590 = 5440 to be fault severity = 0.

Out of this set, the probability of being correct is 3466/5440 = 0.637

Inner log loss for this piece = 3466 × log (0.637) = -678.54

Similarly, we predicted 1519+10447+309 = 12275 to be fault severity = 1

Out of this set, the probability of being correct is 10447/12275 = 0.851

Inner log loss for this piece = 10447 × log (0.851) = -731.60

We predicted 671+586+2053 = 3310 to be fault severity = 2

Out of this set, the probability of being correct is 2053/3310 = 0.620

Inner log loss for this piece = 2053 × log (0.620) = -425.872

⇨ $Log\ loss = \frac{-(-678.54-731.6-425.872)}{(5440+12275+3310)} = 0.087$

5.2 Naïve Bayes

5.2.1 How it works?

Bayes Theorem:

Let A, B be two random variables. Bayes theorem states that probability of A can be calculated given the probability of B or vice-versa. This is called *conditional probability*.

$$P(A|B) = \frac{P(A \text{ and } B)}{P(B)} \text{ or } \frac{P(B|A)P(A)}{P(B)} \qquad -(1)$$

According to our data set let us consider variables X and Y. Where, X = instance set (rows) and Y = Class (column to be predicted). Here, $P(Y|X)$ is called *Conditional probability or Posterior probability* where we have to predict the class based on different row values. This can be done by Bayes Theorem. Whereas, $P(X|Y)$ is called *Class-Conditional probability* where the class is given and rows are to be predicted. For this we have two types of Bayesian Classification methods: *Naïve Bayes and Bayesian Belief Network*. We are going to look into the Naïve Bayes method.

Example Illustration:

Let us use an online dating data to explain conditional probability and Bayes theorem.

Table 3. Example - Online Dating Data

		Age				
		18-29	30-49	50-64	65+	Total
Used online dating	Yes	50	82	55	20	207
	No	200	522	400	350	1472
	Total	250	604	455	370	1679

Using these data, we can calculate the percent of 30 to 49-year-olds use online dating sites = 82/604 ≈ 0.14

Formally this is P (use online dating site | 30-49 year old). We calculated this Conditional probability as simply a ratio of two frequencies, but we can formalize things a bit more. Let event A represent using an online dating site. And event B represent being 30 to 49 years old. The frequency in the numerator corresponds to the number of times events A and B happened at the same time. And the denominator corresponds to the number of times event B happened. Probability of A given B is equal to probability of A and B divided by probability of B, $\frac{P(A\ and\ B)}{P(B)}$. This is Bayes' rule.

So why is this formula called Bayes' rule? Thomas Bayes, who lived between 1702 and 1761, was a mathematician who established a mathematical basis for probability inference, that is, a means of calculating from the number of times an event has not occurred. Let's recalculate that same probability using Bayes' rule.

$$P\ (Use\ online\ dating\ site \mid 30-49\ year\ old)$$

$$= \frac{P(use\ online\ dating\ site\ \&\ 30-49\ year\ old)}{P(30-49\ year\ old)}$$

$$= \frac{\left(\frac{82}{1679}\right)}{\left(\frac{522}{1679}\right)} \approx 0.14$$ which is same as above.

Naïve Bayes Classifier:

Let us assume that the attributes (columns) are independent to each other.

$$P(\boldsymbol{X}|Y=y) = \prod_{i=1}^{d} P(X_i|Y=y)$$

Where, each instance set $\boldsymbol{X} = \{X_1, X_2, \ldots X_d\}$ up to d attributes or features or variables (columns).

The working principle behind Naïve Bayes is that instead of computing Class-Conditional probability for every combination of X, we have to estimate the conditional

probability of each of the X_i given the class Y. Therefore, we are converting a Class-Conditional probability to Conditional probability and using the Bayes theorem for simplification. It can be mathematically formulated as given below:

$$P(Y|\boldsymbol{X}) = \frac{P(Y)\prod_{i=1}^{d} P(X_i|Y)}{P(\boldsymbol{X})} \qquad -(2)$$

The denominator $P(\boldsymbol{X})$ is always constant and can be ignored. Conditional probability can be computed for both categorical and continuous data. For categorical values, the conditional probability is just the normal fraction, which is simple. Whereas, Gaussian distribution is used for the continuous features. Our data set belongs to categorical case.

5.2.2 Log-Loss Calculation

Program:

```
# Program to compute Naïve Bayes using R
 library(e1071)
data<-read.csv(file="C:/Users/Hemanth/Anaconda2/IPython
Notes/dataset/Data1_training.csv")
data$id<-as.factor(data$id)
data$location<-as.factor(data$location)
data$event_type<-as.factor(data$event_type)
data$log_feature<-as.factor(data$log_feature)
data$resource_type<-as.factor(data$resource_type)
data$severity_type<-as.factor(data$severity_type)
data$fault_severity<-as.factor(data$fault_severity)
data$volume<-as.numeric(data$volume)
```

```
numObs<-dim(data)

samps<-floor(0.80*numObs[1])

set.seed(350)

theSample<-runif(samps,1,numObs[1])

trainingDS<-data[theSample,]

testDS<-data[-theSample,]

nbM<-naiveBayes(fault_severity ~ .,trainingDS,laplace=0.25)

predictions<-predict(nbM,testDS)

table(testDS$fault_severity,predictions)
```

Figure 7. Sample of the dataset for Naive Bayes

id	location	event_type	log_feature	volume	resource_type	severity_type	fault_severity
14121	L118	E34	F312	19	R2	S2	1
14121	L118	E34	F232	19	R2	S2	1
14121	L118	E35	F312	19	R2	S2	1
14121	L118	E35	F232	19	R2	S2	1
9320	L91	E34	F315	200	R2	S2	0
9320	L91	E34	F235	116	R2	S2	0

a. Three confusion matrices are obtained for each “laplace” value and different “seed” values.

b. Then, the average of the log loss at each laplace is computed and plotted in the graph.

Using Naïve Bayes algorithm our output and confusion matrix looks like below:

Laplace =1 and seed = 50

```
  predictions
       0     1     2
0 13770  1205  1560
1   736  5782   727
2     2    46  3878
```

We predicted 13770+736+2 = 14508 to be fault severity = 0.

Out of this set, the probability of being correct is 13770/14508 = 0.949

Inner log loss for this piece = 13770 × log (0.949) = -312.22

Similarly, we predicted 1205+5782+46 = 7033 to be fault severity = 1

Out of this set, the probability of being correct is 5782/7033 = 0.822

Inner log loss for this piece = 5782 × log (0.822) = -491.83

We predicted 1560+727+3878= 6165 to be fault severity = 2.

Out of this set, the probability of being correct is 3878/6165 = 0.629

Inner log loss for this piece = 3878 × log (0.629) = -780.74

⇨ $Log\ loss\ = \frac{-(-312.22-491.83-780.74)}{(14508+7033+6165)} = 0.057$

Laplace =1 and seed = 75

```
  predictions
       0     1     2
0 14017  1120  1543
1   774  5872   751
2     6    46  3821
```

We predicted 14017+774+6 = 14797 to be fault severity = 0.

Out of this set, the probability of being correct is 14017/14797 = 0.947

Inner log loss for this piece = 14017 × log (0.947) = -329.66

We predicted 1120+5872+46 = 7038 to be fault severity = 1.

Out of this set, the probability of being correct is 5872/7038 = 0.834

Inner log loss for this piece = 5872 × log (0.834) = -461.91

We predicted 1543+751+3821 = 6115 to be fault severity = 2.

Out of this set, the probability of being correct is 3821/6115 = 0.625

Inner log loss for this piece = 3821 × log (0.625) = -780.32

⇨ Log loss = 0.056

Laplace =1 and seed = 100

```
  predictions
       0     1     2
0 13796  1104  1524
1   743  5884   738
2     7    47  3918
```

We predicted 13796+743+7 = 14546 to be fault severity = 0.

Out of this set, the probability of being correct is 13796/14546 = 0.948

Inner log loss for this piece = 13796 × log (0.948) = -317.18

We predicted 1104+5884+47 = 7035 to be fault severity = 1.

Out of this set, the probability of being correct is 5884/7035 = 0.836

Inner log loss for this piece = 5884 × log (0.836) = -456.55

We predicted 1524+738+3918 = 6180 to be fault severity = 2.

Out of this set, the probability of being correct is 3918/6180 = 0.634

Inner log loss for this piece = 3918 × log (0.634) = -775.47

⇨ Log loss = 0.055

Laplace = 0.75 and seed = 125

```
  predictions
       0     1     2
0 13900  1045  1536
1   653  5937   731
2     5    47  3886
```

We predicted 13900+653+5 = 14558 to be fault severity = 0.

Out of this set, the probability of being correct is 13900/14558 = 0.955

Inner log loss for this piece = 13900 × log (0.955) = -279.21

We predicted 1045+5937+47 = 7029 to be fault severity = 1.

Out of this set, the probability of being correct is 5937/7029 = 0.845

Inner log loss for this piece = 5937 × log (0.845) = -435.34

We predicted 1536+731+3886 = 6153 to be fault severity = 2.

Out of this set, the probability of being correct is 3886/6153 = 0.632

Inner log loss for this piece = 3886 × log (0.632) = -775.58

⇨ Log loss = 0.054

Laplace = 0.75 and seed = 150

	predictions		
	0	1	2
0	13836	1053	1570
1	739	5967	685
2	3	47	3859

We predicted 13836+739+3 = 14578 to be fault severity = 0.

Out of this set, the probability of being correct is 13836/14578 = 0.949

Inner log loss for this piece = 13836 × log (0.949) = -313.90

We predicted 1053+5967+47 = 7067 to be fault severity = 1.

Out of this set, the probability of being correct is 5967/7067 = 0.844

Inner log loss for this piece = 5967 × log (0.844) = -438.45

We predicted 1570+685+3859 = 6114 to be fault severity = 2.

Out of this set, the probability of being correct is 3859/6114 = 0.631

Inner log loss for this piece = 3859 × log (0.631) = -771.22

- ⇨ Log loss = 0.055

Laplace = 0.75 and seed = 175

```
   predictions
         0     1     2
0 13987   959  1482
1   661  5961   807
2     7    42  3871
```

We predicted 13987+661+7 = 14655 to be fault severity = 0

Out of this set, the probability of being correct is 13987/14655 = 0.954

Inner log loss for this piece = 13987 × log (0.954) = -283.39

We predicted 959+5961+42 = 6962 to be fault severity = 1

Out of this set, the probability of being correct is 5961/6962 = 0.856

Inner log loss for this piece = 5961 × log (0.856) = -401.86

We predicted 1482+807+3871 = 6160 to be fault severity = 2

Out of this set, the probability of being correct is 3871/6160 = 0.628

Inner log loss for this piece = 3871 × log (0.628) = -781.00

- ⇨ Log loss = 0.053

Laplace = 0.5 and seed = 200

```
   predictions
         0     1     2
0 14105   921  1549
1   593  6174   594
2     1    46  3827
```

We predicted 14105+593+1 = 14699 to be fault severity = 0

Out of this set, the probability of being correct is 14105/14699 = 0.960

Inner log loss for this piece = 14105 × log (0.960) = -252.69

We predicted 921+6174+46 = 7141 to be fault severity = 1

Out of this set, the probability of being correct is 6174/7141 = 0.865

Inner log loss for this piece = 6174 × log (0.865) = -390.15

We predicted 1549+594+3827 = 5970 to be fault severity = 2

Out of this set, the probability of being correct is 3827/5970 = 0.641

Inner log loss for this piece = 3827 × log (0.641) = -739,05

⇨ Log loss = 0.050

Laplace = 0.5 and seed = 225

	predictions		
	0	1	2
0	13946	889	1437
1	626	6237	575
2	4	38	3897

We predicted 13946+626+4 = 14576 to be fault severity = 0

Out of this set, the probability of being correct is 13946/14576 = 0.957

Inner log loss for this piece = 13946 × log (0.957) = -267.61

We predicted 889+6237+38 = 7164 to be fault severity = 1

Out of this set, the probability of being correct is 6237/7164 = 0.871

Inner log loss for this piece = 6237 × log (0.871) = -375.34

We predicted 1437+575+3897 = 5909 to be fault severity = 2

Out of this set, the probability of being correct is 3897/5909 = 0.660

Inner log loss for this piece = 3897 × log (0.660) = -704.52

⇨ Log loss = 0.049

Laplace = 0.5 and seed = 250

	predictions		
	0	1	2
0	14098	859	1476
1	561	6187	647
2	3	32	3908

We predicted 14098+561+3 = 14662 to be fault severity = 0

Out of this set, the probability of being correct is 14098/14662 = 0.962

Inner log loss for this piece = 14098 × log (0.962) = -240.17

We predicted 859+6187+32 = 7078 to be fault severity = 1

Out of this set, the probability of being correct is 6187/7078 = 0.874

Inner log loss for this piece = 6187 × log (0.874) = -361.51

We predicted 1476+647+3908 = 6031 to be fault severity = 2

Out of this set, the probability of being correct is 3908/6031 = 0.648

Inner log loss for this piece = 3908 × log (0.648) = -736.40

⇨ Log loss = 0.048

Laplace = 0.25 and seed = 300

	predictions		
	0	1	2
0	14466	685	1315
1	378	6494	491
2	4	36	3863

We predicted 14466+378+4 = 14848 to be fault severity = 0

Out of this set, the probability of being correct is 14466/14848 = 0.974

Inner log loss for this piece = 14466 × log (0.974) = -163.75

We predicted 685+6494+36 = 7215 to be fault severity = 1

Out of this set, the probability of being correct is 6494/7215 = 0.9

Inner log loss for this piece = 6494 × log (0.9) = -296.93

We predicted 1315+491+3863 = 5669 to be fault severity = 2

Out of this set, the probability of being correct is 3863/5669 = 0.681

Inner log loss for this piece = 3863 × log (0.681) = -643.51

- ⇨ Log loss = 0.040

Laplace = 0.25 and seed = 325

	predictions		
	0	1	2
0	14263	743	1256
1	426	6608	453
2	3	38	3864

We predicted 14263+426+3 = 14792 to be fault severity = 0

Out of this set, the probability of being correct is 14263/14792 = 0.964

Inner log loss for this piece = 14263 × log (0.964) = -183.57

We predicted 743+6608+38 = 7389 to be fault severity = 1

Out of this set, the probability of being correct is 6608/7389 = 0.894

Inner log loss for this piece = 6608 × log (0.894) = -320.59

We predicted 1256+453+3864 = 5573 to be fault severity = 2

Out of this set, the probability of being correct is 3864/5573 = 0.693

Inner log loss for this piece = 3864 × log (0.693) = -614.58

- ⇨ Log loss = 0.042

Laplace = 0.25 and seed = 350

	predictions		
	0	1	2
0	14576	635	1338
1	362	6482	498
2	3	21	3917

We predicted 14576+362+3 = 14941 to be fault severity = 0

Out of this set, the probability of being correct is 14576/14941 = 0.976

Inner log loss for this piece = 14576 × log (0.976) = -156.57

We predicted 635+6482+21 = 7138 to be fault severity = 1

Out of this set, the probability of being correct is 6482/7138 = 0.908

Inner log loss for this piece = 6482 × log (0.908) = -271,39

We predicted 1338+498+3917 = 5753 to be fault severity = 2

Out of this set, the probability of being correct is 3917/5753 = 0.681

Inner log loss for this piece = 3917 × log (0.681) = -653.91

⇨ Log loss = 0.038

Laplace = 0.1 and seed = 400

```
  predictions
       0     1     2
0 14864   504  1063
1   248  6865   279
2     5    26  3877
```

We predicted 14864+248+5 = 15117 to be fault severity = 0

Out of this set, the probability of being correct is 14864/15117 = 0.983

Inner log loss for this piece = 14864 × log (0.983) = -108.95

We predicted 504+6865+26 = 7395 to be fault severity = 1

Out of this set, the probability of being correct is 6865/7395 = 0.928

Inner log loss for this piece = 7395 × log (0.928) = -221.72

We predicted 1063+279+3877 = 5219 to be fault severity = 2

Out of this set, the probability of being correct is 3877/5219 = 0.743

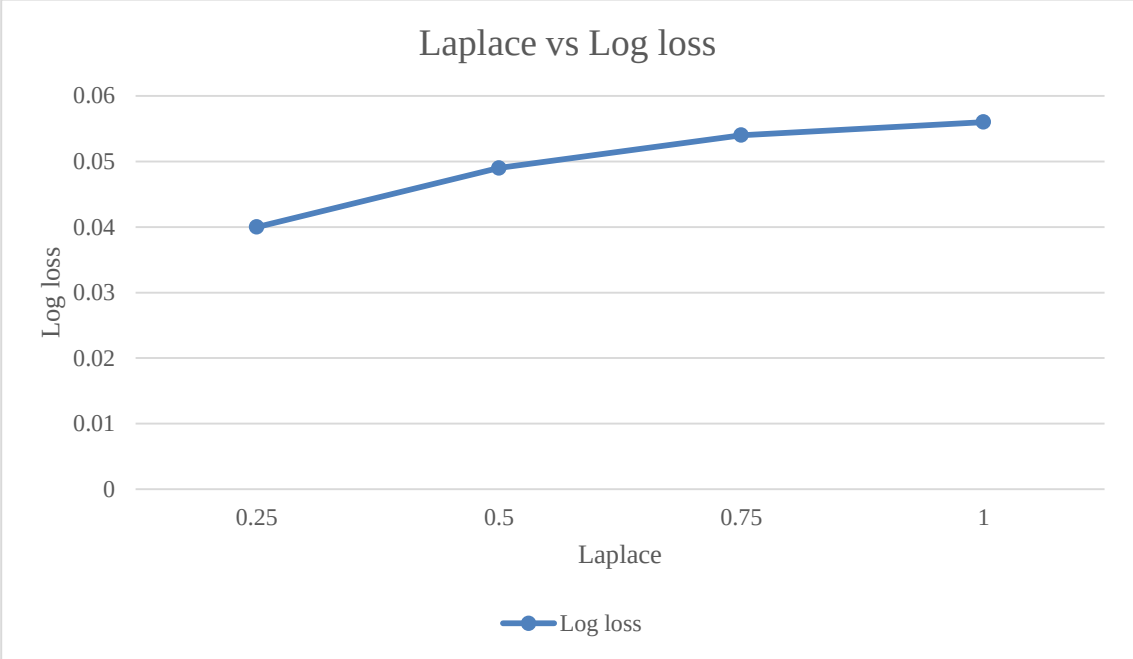
Inner log loss for this piece = 5219 × log (0.743) = -500.49

⇨ Log loss = 0.030

Therefore, From the 3 log loss values at each “laplace”, the average log losses are:

At 1.0 = 0.056, at 0.75 = 0.054, at 0.5 = 0.049, at 0.25 = 0.040

Now we plot these values in the graph – laplace vs log loss.



5.3 Random Forest

5.3.1 How it works?

Bagging or aggregating is a technique where the samples are drawn out repeatedly from the data set with replacement in a uniform probability distribution manner. Since it is done with replacement, some instances may re-appear and some may not appear while sampling. Decision trees are weak learners thereby weak predictors.

Random Forest is an ensemble algorithm which combines (bags) many decision trees to produce a strong classifier output. They average out the output results. Uniform probability distribution generates the *random vectors* which in turn generate decision trees. Whereas, the Ada boost, another ensemble method uses adaptive instead of fixed distribution. As said earlier, since 'n' samples are selected randomly with replacement, randomness is injected into the system. When there are large number of trees, the error rate can be given by:

$$error\ rate\ \leq \frac{\bar{\rho}(1-s^2)}{s^2} \quad \rightarrow (1)$$

Where, $\bar{\rho}$ = Average correlation between the trees

s = Strength or number of trees in the classifier

As the dependency between the trees increases or as the number of trees decreases, the error rate increases. More trees, less error and the classifier can be improved. Steps involved in the implementation of Random forest are as follows:

a) From an input training data, random vectors are created (Uniform probability distribution sampling).
b) Based on the random vectors, decisions trees are generated.
c) At the output, all these decision trees are combined and averaged out.

One way to introduce *random vectors* into the trees is to select N Features. By, doing so the number of features to examine decreases. This is called Random Input (RI) approach. Now, the strength and correlation among the trees depend on the number of features N. Smaller the N value, lesser the correlation thereby lesser is the error rate which can be deduced from eqn. (1)

Number of features, $N = \log_2 d + 1$ where, d = Total number of features.

Secondly, If the d is too small, then it becomes difficult to select the random vectors and in such situations, linear combinations of features are used to increase the feature space. This is called Random Combination (RC) approach.

Random vectors are more robust and faster than the AdaBoost ensemble method.

5.3.2 Log loss Calculation

Program 1

```
# Program to generate Random Forest
library(randomForest)
data <- read.csv("C:/Users/Hemanth/Anaconda2/IPython
Notes/dataset/smotedData.csv")
data$id<-as.factor(data$id)
data$location<-as.factor(data$location)
data$event_type<-as.factor(data$event_type)
data$log_feature<-as.factor(data$log_feature)
data$resource_type<-as.factor(data$resource_type)
data$severity_type<-as.factor(data$severity_type)
```

```
data$fault_severity<-as.factor(data$fault_severity)
data$volume<-as.factor(data$volume)
numObs <- dim(data)
samples <- floor(0.50 * numObs[1])
rand.samples <- runif(samples,1,numObs[1])          #generate random deviates
trainingDS <- data[rand.samples,]
testDS <- data[-rand.samples,]
rf.model <- randomForest(fault_severity ~ ., trainingDS)
print(rf.model)
```

This is the Random Forest algorithm created initially to run in RStudio. But, Since R does not support more than 33 categorical values, this dataset could not be run in R. So, I had to move to Weka software.

Program 2

```
#R Program to Smote the data to solve class imbalance problem
library(DMwR)
# Read in the csv file
originalData<-read.csv("C:/Users/Hemanth/Anaconda2/IPython
Notes/dataset/Data1_training_catfs.csv")
# Omit any rows with missing data for now
originalData<-na.omit(originalData)
# Look at the class spread (composition)
```

```
table(originalData$fault_severity)

# Resample (play with the percentages - execution may take awhile)

smotedData<-SMOTE(fault_severity ~ ., originalData, perc.over = 400,perc.under=100)

# Look at the new composition

table(smotedData$fault_severity)

# Save the "smoted" data

write.csv(smotedData, file="C:/Users/Hemanth/Anaconda2/IPython

Notes/dataset/smotedData.csv")
```

Figure 8. Sample Dataset for Random Forest - Smoted data

id	location	event_type	log_feature	volume	resource_type	severity_type	fault_severity
12186	L1093	E20	F219	1	R2	S1	Zero
6443	L995	E11	F80	2	R8	S1	One
2918	L362	E11	F232	1	R8	S1	Zero
9037	L473	E35	F235	1	R2	S2	Zero
1444	L1052	E11	F82	2	R8	S1	One

The outputs and confusion matrix for Random Forest are given below along with the log loss computations and Graph between Log loss vs No. of trees.

Number of Trees = 10:

Correctly Classified Instances 23933 89.714 %

Incorrectly Classified Instances 2744 10.286 %

a	b	c	Classified as
7142	562	435	a = Zero
547	2768	437	b = One
358	405	14023	c = Two

We predicted 7142+547+358 = 8047 to be fault severity = 0

Out of this set, the probability of being correct is 7142/8047 = 0.888

Inner log loss for this piece = 7142 × log (0.888) = -370.06

We predicted 562+2768+405 = 3735 to be fault severity = 1

Out of this set, the probability of being correct is 2768/3735 = 0.741

Inner log loss for this piece = 2768 × log (0.741) = -360.18

We predicted 435+437+14023 = 14935 to be fault severity = 2

Out of this set, the probability of being correct is 14023/14935 = 0.939

Inner log loss for this piece = 14023 × log (0.939) = -362.53

⇨ Log loss = 0.041

Number of Trees = 30:

Correctly Classified Instances 24016 90.0251 %

Incorrectly Classified Instances 2661 9.9749 %

a	b	c	Classified as
7161	543	435	a = Zero
547	2780	425	b = One
329	382	14075	c = Two

We predicted 7161+547+329 = 8037 to be fault severity = 0

Out of this set, the probability of being correct is 7161/8037 = 0.891

Inner log loss for this piece = 7161 × log (0.891) = -358.91

We predicted 543+2780+382 = 3705 to be fault severity = 1

Out of this set, the probability of being correct is 2780/3705 = 0.750

Inner log loss for this piece = 2780 × log (0.750) = -346.79

We predicted 435+425+14075 = 14935 to be fault severity = 2

Out of this set, the probability of being correct is 14075/14935 = 0.942

Inner log loss for this piece = 14075 × log (0.942) = -362.53

⇨ Log loss = 0.040

Number of Trees = 40:

Correctly Classified Instances 24031 90.0813 %

Incorrectly Classified Instances 2646 9.9187 %

a	b	c	Classified as
7173	543	423	a = Zero
555	2777	420	b = One
330	375	14081	c = Two

We predicted 7173+555+330 = 8058 to be fault severity = 0

Out of this set, the probability of being correct is 7173/8058 = 0.890

Inner log loss for this piece = 7173 × log (0.890) = -362.43

We predicted 543+2777+375 = 3695 to be fault severity = 1

Out of this set, the probability of being correct is 2777/3695 = 0.752

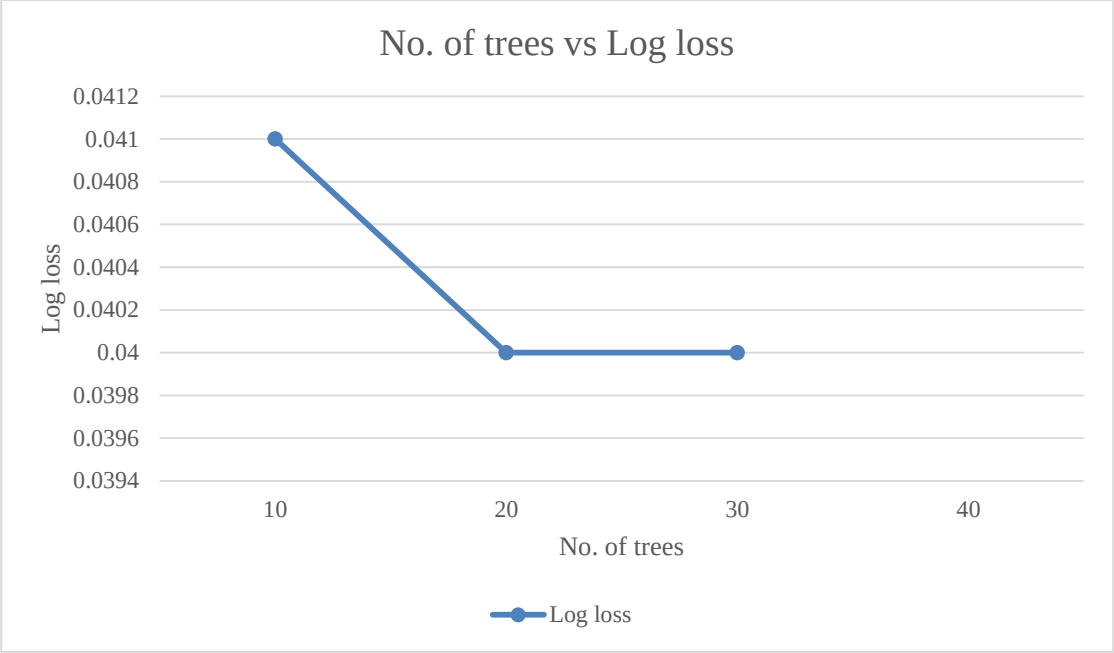
Inner log loss for this piece = 2777 × log (0.752) = -344.46

We predicted 423+420+14081 = 14924 to be fault severity = 2

Out of this set, the probability of being correct is 14081/14924 = 0.944

Inner log loss for this piece = 14081 × log (0.944) = -355.57

⇨ Log loss = 0.040


No. of trees vs Log loss
0.0412
0.041
0.0408
0.0406
0.0404
0.0402
0.04
0.0398
0.0396
0.0394
Log loss
10
20
30
40
No. of trees
Log loss

Chapter 6: Conclusion and Future work

6.1 Conclusion

We have seen how Self Organizing Networks (SON) can be used along with machine learning techniques to make powerful predictions. From the results of machine learning algorithms, we can deduce that Naïve Bayes best suits our dataset with a minimum log loss of < 0.04. We attain this value with lesser value of laplace parameter (< 0.25). So, from the graph laplace vs. log loss is always a decreasing curve. The next better algorithm would be Random Forest, which is an ensemble method that has a minimum log loss value of 0.04. But its graph becomes *constant* at 0.04 after number of trees > 30. Support Vector Machines performed poorly with a log loss value of 0.08, resulting in low accuracy rate in terms of classification.

6.1.1 Detailed accuracy by class

Precision and recall are the two widely used metrics in the classification problems. Precision is the fraction of *actual* records that are positive. It is calculated column wise in a confusion matrix. While recall means the fraction of *predicted* records that are positive, it is calculated row wise in a confusion matrix.

Mathematically they are represented as:

$$Precision, p = \frac{True\ Positive}{True\ Positive + False\ Positive}\ or\ \frac{TP}{TP + FP}$$

$$Recall, r = \frac{TP}{TP + FN(False\ Negative)}$$

F Measure is the harmonic mean of precision and recall and is given by:

$$F = \frac{2}{\frac{1}{p}+\frac{1}{r}} = \frac{2pr}{p+r}$$

Naïve Bayes:

```
predictions
        0     1     2
  0 14864   504  1063
  1   248  6865   279
  2     5    26  3877
```

For fault severity 0,

Precision, p = 14864/15117 = 0.983

Recall, r = 14864/16431 = 0.905

$$F\ Measure = \frac{2 \times 0.983 \times 0.905}{0.983 + 0.905} = 0.942$$

For fault severity 1,

Precision, p = 6865/7395 = 0.928

Recall, r = 6865/7392 = 0.929

$$F\ Measure = \frac{2 \times 0.928 \times 0.929}{0.928 + 0.929} = 0.928$$

For fault severity 2,

Precision, p = 3877/5219 = 0.743

Recall, r = 3877/3842 = 1.009

$$F\ Measure = \frac{2 \times 0.743 \times 1.009}{0.743 + 1.009} = 0.856$$

Table 4. Naive Bayes detailed accuracy

Precision	Recall	F-Measure	Class
0.983	0.905	0.942	Zero
0.928	0.929	0.928	One
0.743	1.009	0.856	Two

Random Forest:

a	b	c	Classified as
7142	562	435	a = Zero
547	2768	437	b = One
358	405	14023	c = Two

Similarly, we get:

Table 5. Random Forest detailed accuracy

Precision	Recall	F-Measure	Class
0.888	0.878	0.882	Zero
0.741	0.738	0.739	One
0.941	0.948	0.945	Two

Support Vector Machines:

a	b	c	Classified as
3466	1519	671	a = Zero
1384	10447	586	b = One
590	309	2053	c = Two

Similarly, we get:

Table 6. Support Vector Machines detailed accuracy

Precision	Recall	F-Measure	Class
0.851	0.841	0.846	Zero
0.637	0.613	0.625	One
0.620	0.695	0.656	Two

For classes 0 and 1, the performance of $NB > RF > SVM$

For class 2, the performance of $RF > NB > SVM$

6.2 Future Work

With Naïve Bayes algorithm, we can find out the exact predicted fault severity (0 or 1 or 2) of any individual instances. But, with Random Forest, as of now, we cannot find out what could be the fault severity of a particular 1000^{th} or 1746^{th} record. This could be taken up as an extension of the current work in the future to build the algorithm further. Also, with SVM performing poorly, we could research on any other algorithms (neural networks) and try to see if they provide greater accuracy.

References

1. Alpaydın, E., Introduction to Machine Learning. Second ed. 2010, Cambridge, Massachusetts: The MIT Press.

2. Mehryar Mohri, A.R.a.A.T., Foundations of Machine Learning. Thomas Dietterich ed. 2012, Cambridge, Massachusetts: The MIT Press.

3. PANG-NING TAN, M.S.a.V.K., Introduction to Data Mining. 2006, United States: Pearson Education, Inc.

4. R. Srikant, L.Y., Communication Networks: An Optimization, Control and Stochastic Networks perspective. 2014, United States of America: Cambridge University Press.

5. Rao, S.S., Engineering Optimization: Theory and Practice. Fourth ed. 2009, United States of America: John Wiley & Sons, Inc.

6. Ajesh, A., J. Nair, and P.S. Jijin. A random forest approach for rating-based recommender system. in 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI). 2016.

7. Birla, S., K. Kohli, and A. Dutta. Machine Learning on imbalanced data in Credit Risk. in 2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON). 2016.

8. Bousmina, A., C. Jlassi, and N. Arous. Combining ensemble methods of Bagging, Subagging and Random Subspace for phoneme recognition. in 2016 2nd International Conference on Advanced Technologies for Signal and Image Processing (ATSIP). 2016.

9. Li, Y., et al. Research and application of random forest model in mining automobile insurance fraud. in 2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD). 2016.

10. Liao, Q. and S. Stanczak. Network State Awareness and Proactive Anomaly Detection in Self-Organizing Networks. in 2015 IEEE Globecom Workshops (GC Wkshps). 2015.

11. Ma, W., K. Tan, and P. Du. Predicting soil heavy metal based on Random Forest model. in 2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS). 2016.

12. Moraes, R.M. and L.S. Machado. A Fuzzy Binomial Naive Bayes classifier for epidemiological data. in 2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE). 2016.

13. Mrinalini, S., N.S. Abinayalakshmi, and C.V. Kumar. Wavelet feature based SVM and NAIVE BAYES classification of glaucomatous images using PCA and Gabor filter. in 2016 10th International Conference on Intelligent Systems and Control (ISCO). 2016.

14. Tabassum, N. and T. Ahmed. A theoretical study on classifier ensemble methods and its applications. in 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom). 2016.

15. Trivedi, S.K. A study of machine learning classifiers for spam detection. in 2016 4th International Symposium on Computational and Business Intelligence (ISCBI). 2016.

16. Wang, S., et al. The airborne hyperspectral image classification based on the random forest algorithm. in 2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS). 2016.

17. Ying, W., et al. Decision tree based validation of load model parameters. in 2016 IEEE Power and Energy Society General Meeting (PESGM). 2016.

18. Zhehan, Y. and A.H. Etemadi. A novel detection algorithm for Line-to-Line faults in Photovoltaic (PV) arrays based on support vector machine (SVM). in 2016 IEEE Power and Energy Society General Meeting (PESGM). 2016.

19. Aliu, O.G., et al., A Survey of Self Organisation in Future Cellular Networks. IEEE Communications Surveys & Tutorials, 2013. 15(1): p. 336-361.

20. Andrews, J.G., et al., What Will 5G Be? IEEE Journal on Selected Areas in Communications, 2014. 32(6): p. 1065-1082.

21. Imran, A., A. Zoha, and A. Abu-Dayya, Challenges in 5G: how to empower SON with big data for enabling 5G. IEEE Network, 2014. 28(6): p. 27-33.

22. Machine Learning: General Concepts. Available from: http://www.astroml.org/sklearn_tutorial/general_concepts.html.

23. Scikit-learn in Python. Available from: http://scikit-learn.org/stable/.

24. Kolter, Z. Convex Optimization Overview. 2008; Available from: http://cs229.stanford.edu/section/cs229-cvxopt.pdf.

25. Ng, A. Support Vector Machines Lecture Notes. Available from: http://cs229.stanford.edu/notes/cs229-notes3.pdf.