CHAPTER 9

SUPERVISED MACHINE LEARNING TECHNIQUES
FOR RISK IDENTIFICATION

9.1 INTRODUCTION

Various studies in the field of software have identified common issues that affect software project schedules, such as: cultural differences, high number of distributed sites, different knowledge expertise and domains, many communication dependencies, time zone differences. According to market research done by Data Monitor, the size of the worldwide software industry in 2011 was US$ 267 billion and growing exponentially. To sustain such growth the importance of Software Engineering concepts like Risk Analysis is immense as any unforeseen and unprepared for event can lead to losses amounting to Billions. Risk analysis is the study of risks and the evaluation of the probability or likelihood of their occurrence engineering (Ebert et al 2008; Hillegersberg& Herrera 2007; Hussey &Hall 2008; Prikladnicki&Yamaguti 2000).

Traditional methods of risk management and analysis involve holding risk discussion workshops to identify potential issues and risks ahead of time before these were to pose cost and/or schedule-negative impacts. These risk analysis workshops need to be chaired by a large group so as to cover every risk element from different perspectives. The outcome is a creation or review of the risk register to identify and quantify risk elements to a project and its impact on the same. As this is a continuous and iterative
process, such workshops need to be held at regular intervals as well as at different project milestones to make appropriate changes (Kuipers et al 2003; Kitchenham & Jorgensen 2004).

These traditional methods require a lot of manpower and critical time which can be used in a more productive fashion. In this work, further automation of Risk Analysis is targeted, thereby making it easier to take mitigation measures and increase productivity using supervised machine learning algorithms (Rich & Alexandru 2006).

Despite of the existing tools and methodologies for risk assessment, there’s no quantitative approach that takes into account the variability and complex nature of software engineering. Here, the use of a Supervised Risk Identification and Assessment algorithm is proposed, which will identify risks, based on various project parameters input to it and also on the learning from previous projects having similar characteristics. Fast, accurate, and automated detection of these risks is a requirement for the success of this system to be used for risk management.

Furthermore, the risk identification and assessment detection needs to take place in real-time (Adailton 2010). The risk identification and assessment system should be an online system which monitors data and continuously learns from every use and adapts to project characteristics peculiar to an organization. The system also needs to be aware and address the dynamic nature of projects. Most importantly the system should exhibit good accuracy rate.

In this chapter, the results of two efficient algorithms namely Logistic Regression with Regularization and Neural networks with Back propagation are compared.
9.2 SUPERVISED MACHINE LEARNING METHODS FOR RISK IDENTIFICATION

9.2.1 Logistic Regression with Regularization

A supervised learning task is considered where \( M \) training instances \( \{(x^{(i)}, y^{(i)}), i = 1, ..., M\} \) are given. Here each \( x^{(i)} \in \mathbb{R}^N \) is an \( N \)-dimensional feature vector, and \( y^{(i)} \in \{0, ..., K\} \) is a class label. Logistic regression models the probability distribution of the class label \( y \) given a feature vector \( x \) as follows:

\[
p(y = 1 \mid x; \theta) = \sigma(\theta^T x) = \frac{1}{1 + \exp(-\theta^T x)}
\]  

(9.1)

Here \( \theta \in \mathbb{R}^N \) are the parameters of the logistic regression model and \( \sigma(.) \) is the sigmoid function defined by the second equality.

Under the Laplacian prior \( p(\theta) = (\beta / 2)^N \exp(-\beta \|\theta\|_1) \) (with \( \beta > 0 \)), the maximum a posteriori (MAP) estimate of the parameter \( \theta \) is given by:

\[
\min_{\theta} \sum_{i=1}^{m} - \log p(y^{(i)} | x^{(i)}; \theta) + \beta \|\theta\|_1
\]

This optimization problem is referred to as \( L_1 \) regularized logistic regression. Often it will be convenient to consider the following alternative parameterization of the \( L_1 \) regularized logistic regression problem:

\[
\min_{\theta} \sum_{i=1}^{m} - \log p(y^{(i)} | x^{(i)}; \theta)
\]

subject to \( \|\theta\|_1 \leq C \)
The optimization problems are equivalent, in the following sense: for any choice of $\beta$, there is a choice of $C$ such that both optimization problems have the same minimizing argument. This follows from the fact that, (up to a constant that does not depend on $\beta$) the optimization problem (is the Lagrangian of the constrained of another optimization problem, where $\beta$ is the Lagrange multiplier. In practice, $C$ (and/or $\beta$) may be either chosen manually or via cross-validation (Su-In et al 2006).

**9.2.2 Neural networks with back propagation**

For second experiment a neural network model with 3 layers – an input layer, a hidden layer and an output layer is considered. The cost function for neural networks with regularization is derived by experts and after substitution the equation is obtained as (9.2)

$$
J(i) = -\frac{1}{m} \left[ \sum_{i=1}^{m} \left( \sum_{k=1}^{K} \left[ y_{k}^{(i)} \log(h_{j}(x^{(i)}))_{k} + (1-y_{k}^{(i)}) \log(1-h_{j}(x^{(i)}))_{k} \right] \right) \right] + \frac{\lambda}{2m} \left[ \sum_{j=1}^{25} \sum_{k=1}^{400} (\Theta_{j,k}^{(1)})^{2} + \sum_{j=1}^{25} \sum_{k=1}^{25} (\Theta_{j,k}^{(2)})^{2} \right]
$$

Given a training example $(x^{(i)}, y^{(i)})$, first a “forward pass” is run to compute all the activations throughout the network, including the output value of the hypotheses $h_{\omega}(x)$. Then for each node $j$ in layer $l$, “error term” $\delta_{j}^{(l)}$ is computed which measures how much that node was “responsible” for any errors in the output. For an output node, the difference between the network’s activation and the true target value is measured, and is used to define $\delta_{j}^{(3)}$. For hidden units $\delta_{j}^{(l)}$ can be computed based on a weighted average of the error terms of the nodes in layer $(l+1)$.
9.3 COMPUTATIONAL EXPERIMENTS AND RESULTS

The experiments are conducted using a dataset of 9 features which covers various parameters of a software project being worked upon by an organization. The features considered are $x_1$ – Working Days to complete project, $x_2$ – Number of Personnel Involved, $x_3$ – Total Lines of Code, $x_4$ – AvgLines of Code written per person per working day, $x_5$ – Number of Teams Involved, $x_6$ – AvgExperience of Team Members in Years, $x_7$ – Avg Team Skill on a scale of 1-5, $x_8$ – Development Lifecycle Model Selected, $x_9$ – Avg experience of Project Managers. All these features are normalized using Feature Normalization $x' = \frac{x - \mu}{\sigma}$ where $\mu$ is mean and $\sigma$ is the variance. After feature normalization all the features are converted to Gaussian distributions if they already do not resemble one.

The different risks that can be identified and assessed by this project are $y_1$ – Conflicts between users, $y_2$ – Users not committed to the projects, $y_3$ – Continually changing requirements, $y_4$ – System requirement not adequately identified, $y_5$ – Unclear System Requirements, $y_6$ – High Level of technical complexity, $y_7$ – Lack of Project Progress Monitoring, $y_8$ – Inadequate estimation of required resources, $y_9$ – Inexperience Project Managers, $y_{10}$ – Ineffective communications, $y_{11}$ – Inexperience team members, $y_{12}$ – Team Members Lack Specialized skill required by the project, $y_{13}$ – real-time performance shortfalls.

The Dataset is divided into training set and test set. The model is trained using the training set and then the value of $\varepsilon$ is optimized on the cross validation set. The F1 Score is calculated on the test set.

A preliminary evaluation was conducted using data from a survey of several software development projects from an Industry. The supervised machine learning algorithms were applied on these projects to identify and
assess the risks in these projects. The performance of Logistic Regression with Regularization is compared against Neural Networks with Back propagation. The F1 score was used as a measure of the performance of the algorithms and Table 9.1 and 9.2 shows the confusion matrix logistic regression and Neural Networks with Back propagation respectively (Powers 2011).

### Table 9.1 Confusion matrix for logistic regression

<table>
<thead>
<tr>
<th>True Positive</th>
<th>False Positive</th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>False Negative</td>
<td>True Negative</td>
</tr>
</tbody>
</table>

Precision = \(\frac{tp}{tp + fp}\)  
(9.3)

Precision = 0.9

Recall = \(\frac{tp}{tp + fn}\)  
(9.4)

Recall = 0.75

F1 Score = \(2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}\)  
(9.5)

F1 Score = 0.818

### Table 9.2 Confusion matrix for neural network with back propagation

<table>
<thead>
<tr>
<th>True Positive</th>
<th>False Positive</th>
</tr>
</thead>
<tbody>
<tr>
<td>34</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>False Negative</td>
<td>True Negative</td>
</tr>
</tbody>
</table>
Precision = \frac{tp}{tp + fp}

Precision = 0.944

Recall = \frac{tp}{tp + fn}

Recall = 0.91

F1 Score = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}

F1 Score = 0.931

9.4 CONCLUSION AND DISCUSSION

This article addressed the issue of software risk identification and assessment using two supervised learning techniques like logistic regression with regularization and neural networks with back propagation. As the software industry grows and projects become more complex, automated techniques for Risk Identification and Assessment become important. The proposed method holds a lot of promise to make the tasks of risk management easier and more accurate. The above method is lightweight, easy to deploy, and has shown to be effective in the experimental evaluations. As ongoing work, various extensions are being explored to the basic ranking mechanism so as to provide better risk assessment and to make the system more robust.