APPLICATION OF VARIOUS MACHINE LEARNING APPROACHES TO ESTIMATE LIQUEFACTION RISK

by

Amin Shoari Nejad

B.S., Civil Engineering, Islamic Azad University of Tehran , 2013

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Civil Engineering

Boğaziçi University

2017

ACKNOWLEDGEMENTS

Foremost, I would like to express my sincere gratitude towards my family for the encouragement and support which helped me in completion of this thesis.

Also I would like to express my sincere gratitude towards my advisors Prof. Erol Güler and Prof. Meltem Özturan for their continuous support and caring. Without their guidance this thesis would not have been accomplished.

I am highly indebted to Dr. Huseyin Sami Karaca for intoroducing me valuable references, which build the core of this thesis, and I really appreciate his supports that helped me to pursue my future career.

I also wish to thank my jury committee members Dr. Mohammad Ebrahim Banihabib and Dr. Irem Zeynep Yıldırım for their valuable suggestions which have formed the final version of my thesis.

I thank all of my friends, espesially Reza Alizadeh Ashrafi, who helped me a lot to finish my master degree.

ABSTRACT

APPLICATION OF VARIOUS MACHINE LEARNING APPROACHES TO ESTIMATE LIQUEFACTION RISK

In this thesis, the possibility of using quadratic discriminant analysis (QDA), artificial neural networks (ANN), random forest and support vector machine (SVM), which are four famous machine learning approaches, to model the complex relationship between liquefaction risk and soil seismic features has been investigated. Nowadays with the development of computational speed, such approaches can give engineers faster and economical results and in many cases there is no need to take extreme assumptions about the structure of a problem in order to simplify and make it solvable. Machine learning techniques use data to extract information. For this thesis, a liquefaction database with 415 case histories has been used. Three soil parameters (depth to critical layer, σ_v, V_{S1}) and two seismic parameters (M_w, PGA) are considered as the models inputs and the liquefaction potential of soil is the output. It has been shown that all of the mentioned models can reasonably predict whether a soil is liquefiable or not, however, random forest outperformed the other methods and showed the most accuracy amongst the models. Finally random forest performance has been compared to the performance of the simplified approach, which is a traditional solution, to assess whether a soil is liquefiable or not.

ÖZET

ÇEŞITLI ÖZDEVIMLI ÖĞRENME TEKNIKLERININ SIVILAŞMA RISKININ TAHMIN EDILMESINDE UYGULANMASI

Yapılan çalışmada, özdevimli öğrenme de yaygın olarak kullanılan dört yaklaşım modelinin (quadratic discriminant analysis, artificial neural networks, random forest ve support vector machine), sıvılaşma riski ve toprak sismik özeliğini modellemek adına uygulanabilirliği incelenmektedir. Bu sıralar bilgisayımsal hızın gelişmesi ile birlikte, bu tür yaklaşımlar mühendislere, hızlı ve ekonomik sonuçlar ve bir çok durumda yapı ile ilgili yapılan aşırı varsayımların ihtiyacını azaltarak, kolay çozüm olanağı sağlar. Özdevimli öğrenme teknikleri verileri kullanarak bilgi çıkarımı yapabilir. Bu tez için, 415 vaka geçmişi veritabanı olarak kullanılmıştır. Üç adet toprak parametresi (derinlik kritik tabaka, σ_v, V_{S1}) ve iki sesmik parametre (M_w, PGA) model girdisi olarak ve toprağın sıvılaşma potansiyeli çıktı olarak dikkate alınmıştır. Deney sonunda belirtilen modeller, toprağın sıvılaşıp sıvılaşmayacağını mantıklı bir şekilde tahmin etmiştir, ancak random forest, diğer metotlardan daha iyi olduğunu ve modelleme olarak daha doğru sonuç verdiğini göstermiştir. Son olarak random forest performansı, diğer basitleştirilmiş (geleneksel) yaklaşımların performansı ile karşılaştırılmış ve bu şekilde toprağın sıvılaşmasının mümkün olup olmadığı değerlendirilmiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

Ave	Average of
a_{max}	Peak horizontal ground surface acceleration
$\bar{a}$	Sample mean of parameter a
$\mathbf{CV}_{(k)}$	K-fold cross-validation error
D_R	Soil relative density
$f_k(x)$	Probability density function in the kth class
$\hat{f}$	Approximated relationship
G	Gini index
$\mathbf{I}(y_i \neq \hat{y}_i)$	An indicator variable that equals 1 if $y_i \neq \hat{y}_i$ and zero if $y_i = \hat{y}_i$
M_w	Earthquake moment magnitude
n	Number of observations
$N_{1,60}$	The corrected SPT-N number
$Pr(Y\|X)$	Conditional probability of Y given X
$\hat{p}_{mk}$	Ratio of observations from k available in the mth segment
r_d	Shear stress reduction factor
Var(x)	Variance of x
V_{s1}	Normalized shear wave velocity
X	Vector of predictor variables
Y	Vector of dependent variables
Z^{*j}	Bootstrap jth sample dataset
μ_k	Mean vector in the kth class
Σ_k	Covariance matrix in the kth class
σ^2	Variance
σ_v	Total vertical stress of soil
σ'_v	Effective vertical stress of soil

LIST OF ACRONYMS/ABBREVIATIONS

ANN	Artificial Neural Network
CPT	Cone Penetration Test
CRR	Cyclic Resistance Ratio
CSR	Cyclic Stress Ratio
FS	Factor of Safety
GWT	Ground Water Table
LDA	Linear Discriminant Analysis
PGA	Peak Ground Acceleration
QDA	Quadratic Discriminant Analysis
SPT	Standard Penetration Test
SVM	Support Vector Machine

1. INTRODUCTION

Many geotechnical engineering problems depend on the application of empirical relationships provided in the form of equations or design charts, in order to find the response of a system to input parameters. This is usually due to lack of knowledge about the physical phenomena or some limitations related to obtaining information in a multivariate problem. Furthermore, sometimes the system is too complex to be explained in a deterministic way. A typical example is the determination of liquefaction potential in soils, for which empirical approaches based on in-situ test data, have been proposed. A well known approach is the simplified stress-based, proposed by Seed-Idris, that is introduced in chapter four. An alternative approach is the use of machine learning algorithms that take least assumptions about the structure of the problem, making them flexible enough to approximate any complex relationships. Goh [1] applied artificial neural network to assess liquefaction risk, using reports of 13 earthquakes which occured in Pan-America, United States and Japan between 1891-1980. He used standard peneteration test (SPT) results as input to his model. Afterwards, he used cone peneteration test (CPT) data in order to assess liquefaction risk using neural networks [2]. Furthermore, aritificial neural network has been implemented in order to predict liquefaction resistance and potential by Juang et. al [3]. Also Ali and Najjar [4] compared accuracy of liquefaction prediction using artificial neural network to that of fuzzy logic and statistical methods. In 2006, Mahesh Pal, used support vector machine to predict occurrence and non-occurrence of liquefaction based on CPT and SPT data sets [5].

Present study investigates the potential of different machine learning approaches namely QDA, ANN, SVM and random forest to predict occurrence of liquefaction in soil, using shear wave velocity test results.

1.1. An Overview of Machine Learning

Machine Learning is a subset of artificial intelligence, involving computer algorithms that are used to autonomously learn from data. This suggests that computers do not have to be explicitly programmed for the best performance since machine learning algorithms can change and improve their efficiency through their learning process.

Though the term machine learning is not so old, many of its underlying concepts were developed several years ago. Gauss and Legendre, at the beginning of the nineteenth century, published papers on the method of least squares, which implemented the oldest form of what is now known as linear regression. The idea was first applied to astronomy problems. At that time linear regression was used to predict quantitative variables. Afterwhile, in order to predict qualitative values, such as whether a patient dies or survives, or whether the stock market will be bullish or bearish, Fisher proposed linear discriminant analysis (LDA) in 1936 [6], which then developed for non-linear applications and called quadratic discriminant analysis (QDA). In the 1940s, some authors put forth an alternative method, logistic regression, which is able to find the separating boundry within qualititative variables. In the early 1970s, Wedderburn and Nelder [7] devise the term generalized linear models for an entire class of statistical learning methods that include both linear and logistic regression as special cases.

By the end of 1970s, lots of additional approaches for learning from data were available; However, most of them were linear methods, since fitting non-linear relationship was computationally cumbersome and not possible at that time. By the end of 1980s, computers had improved enough that non-linear methods were no longer computationally unachievable. In mid 1980s, Breiman, Olshen, Stone and Friedman [8] introduced classification and regression trees, and were among the pioneers to demonstrate the power of a detailed practical implementation of a method, which involves cross-validation for model tuning. Since that time, statistical learning, inspired by the advent of machine learning, has been considered as a new subfield of statistics, focused on supervised and unsupervised modeling and prediction.

Recently, progress in machine learning has been noticed by the increasing availability of relatively user-friendly and powerful tools and softwares, such as R system, which is a freely available and popular programming language [9] that also has been used for modeling in this thesis. This has the potential to continue the transformation of the field from a set of techniques used and developed by statisticians and computer scientists to an essential toolkit for a much broader community like civil engineers [10].

1.2. Supervised Learning

Generally speaking the term supervised learning implies that mapping between some inputs data and an output response, with the aim of accurately predicting the response for future observations. In such a scenario for every observations in the database there are predictors measurements x_i, i = 1, ..., n and there is at least one response or dependent variable y_i. In contrast, for some special cases one may want to find patterns in some observation x_i, i = 1,...,n without any associated responses, which is called unsupervised learning and is out of the scope of this thesis.

Many classical statistical learning methods such as linear regression and logistic regression, as well as more sophisticated approaches such as artificial neural networks, are considered as supervised learning methods. Note that all of these approaches are trying to fit to a training data set. Also the fitted model often referred to as trained model [11].

1.3. Regression Versus Classification Problems

Variables can be characterized as either quantitative or qualitative (categorical). When the response is numerical or quantitative often the problem is called regression problem and a problem with a qualitative response is referred to as classification problem. This thesis focuses on the later since the response for liquefaction problem is categorical as we want to evaluate whether liquefaction is going to happen or not. More specifically this problem often referred to as binary classification problem.

1.4. The Bias-Variace Trade Off

As it has been mentioned before, through supervised learning the goal is to approximate a relationship, which might be very complex, between inputs and an output. Let us denote the approximated relationship as $\hat{f}$. Accordingly, variance refers to the amount by which $\hat{f}$ would change if we estimated it using a different training data set. Recall that the training data are used to fit a machine learning method. As a result, different data sets will end up in a different $\hat{f}$. However, ideally we expect that $\hat{f}$ not vary too much using different training sets. A method with high variance can become unstable, which means small changes in the training data may result in large changes in $\hat{f}$. Genreally, more flexible machine learning methods tend to have higher variance.

On the other hand, bias refers to the error that is introduced by taking extreme assumptions about a real-life problem, usually to simplify an extremely complicated problem by a very simpler model. For instance, linear regression assumes that there is linear relationship between inputs and the response. As a result, it introduces some bias in the estimate of $\hat{f}$ if in fact the relationship is not completely linear which is the case in most real-world problems. Generally more flexible models tend to result in less bias.

Both bias and variance can contribute to the model's error. Ideally we are looking for a model with least bias and least variance as possible while they are contrariwise correlated. In other words, increasing variance will reduce the bias and vice versa. In Figure 1.1 it can be seen that model one has higher variance and tends to embrace the training data too closely [10]. This situation often refers to as overfitting the data. On the other hand, model two has lower variance and higher bias compared to model one and the fitted boundary is less sensitive to changes in the training set. Usually models with high variance have wiggly appearance while models with high bias look smoother.

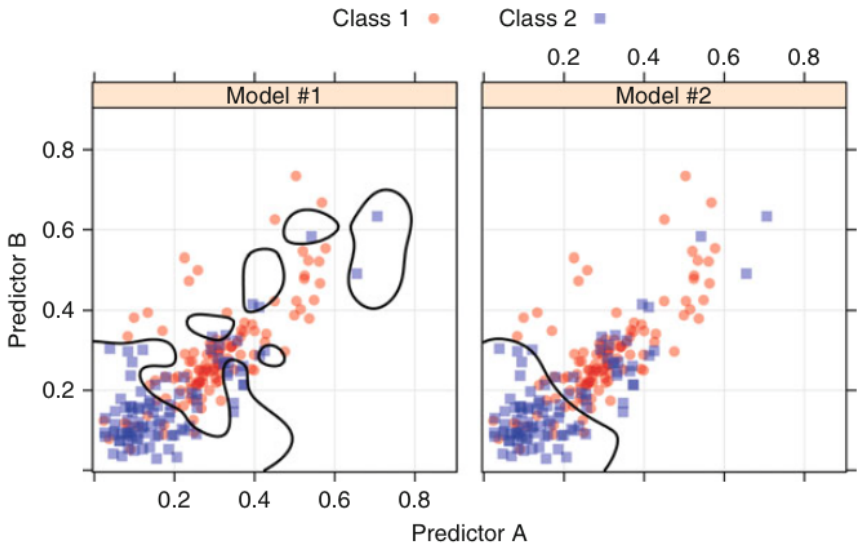

Figure 1.1. An example of a training set with two classes and two predictors. The panels show two different classification models and their associated class boundaries.

1.5. Evaluating Model Accuracy

In order to evaluate the performance of a machine learning model on a given data set, there is a straightforward approach which consists of measuring how accurately its predictions actually match the observed data. In classification setting, the most commonly-used measure is error rate which can be described formally as the number of misclassifications in total attempts. Mathematically this can be written as follows

$$\frac{1}{n}\sum_{i=1}^{n}\mathbf{I}(y_i \neq \hat{y}_i). \tag{1.1}$$

In Equation (1.1), n is the number of training observations and $\hat{y}_i$ is the predicted class label for the ith observation using $\hat{f}$. $\mathbf{I}(y_i \neq \hat{y}_i)$ is an indicator variable that equals 1 if $y_i \neq \hat{y}_i$ and zero if $y_i = \hat{y}_i$. In other words, if $y_i \neq \hat{y}_i = 0$ then ith observation is classified correctly by the classification model; otherwise it is misclassified. Note that Equation (1.1) is referred to as training error since it is calculated based on the data that is used to train the classifier. However, we are interested in error rates that result from applying our classifier to unseen data that is not used in training the classifier.

Otherwise, the error rate we obtain is optimistic because the classifier is adjusted to training data to make the least errors. This can be a serious issue especially when it comes to using more flexible models because of their potential to overfit the data. A common approach to avoid this issue and get more reliable results is to sample a percentage of the total database and hold it out before training a classifier model and use those data, which we are going to refer to as test set, to evaluate the performance of a classifier as if the test set data are future cases that we want to predict. The test error rate associated with a test set can be calculated as follows

$$Ave\ \mathbf{I}(y_i \neq \hat{y}_i). \tag{1.2}$$

Using Equation (1.2) the fraction of misclassifications by the classifier can be obtained. It is expected that the training error rate to be always below the test error rate. Also by increasing model flexibility the training error rate will be improved, however, this is not the case with the test error rate.

In Figure 1.2 the gray line is associated with training error rate while the red line is associated with test error rate. In the y axis the error rate measurement is presented and the x axis is representative of a classifier flexibility (In the next chapter we will discuss how exactly a model flexibility can be defined and measured for different machine learning algorithms). The dashed line represents the irreducible error or the minimum possible test error over all methods (this error is due to the intrinsic noise in the data) [10]. It has been mentioned before that increasing model's flexibility often reduces the bias while increasing variance. Figure 1.2 shows how more flexible models can perform better on the training set and even pass the minimum possible test error rate by overfitting and taking noise as information. As the flexibility of a machine learning method increases, we observe a U-shape in the test error rate. This is a basic property of statistical learning that holds regardless of the particular data set at hand and regardless of a machine learning algorithm that being used. In other words, increasing flexibility too much will push the learning algorithm to work too hard to find patterns in the training data which might cause selecting wrong patterns.

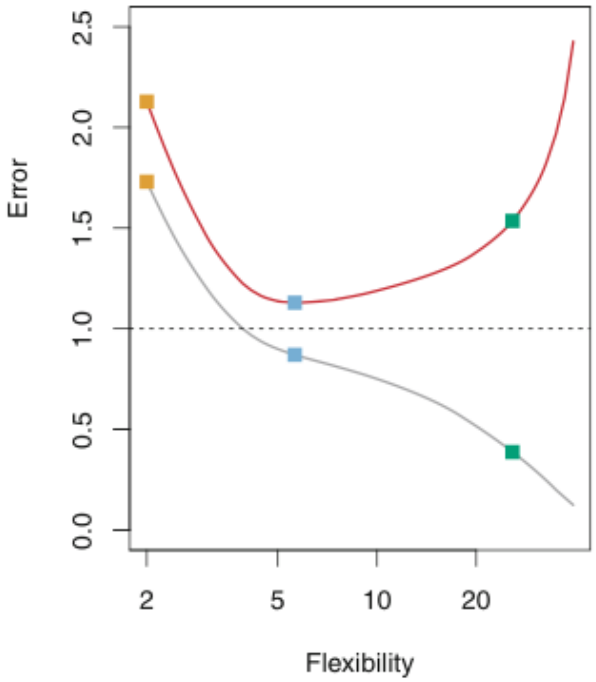

Figure 1.2. Training error (grey curve), test error (red curve), and minimum possible test error over all methods (dashed line).

Such patterns are just caused by chance resulting in high error rates in the test set as well as in prediction of future cases. For these reasons we are looking for an optimum point with least possible bias and least possible variance. The optimum point can be found by trial and error and one of the important approaches for performing this procedure is resampling methods [12–14], which we will use in upcoming chapters for tuning the machine learning algorithms.

1.6. Resampling Methods

Resampling techniques are essential tools in modern statistics. They consist of drawing samples from a training set repeatedly and refitting a model of interest on each sample to evaluate the fitted model. For instance, if we want to estimate the variability of a model such as linear regression, multiple different samples can be drawn from the training data and a linear regression model can be fitted to each sample.

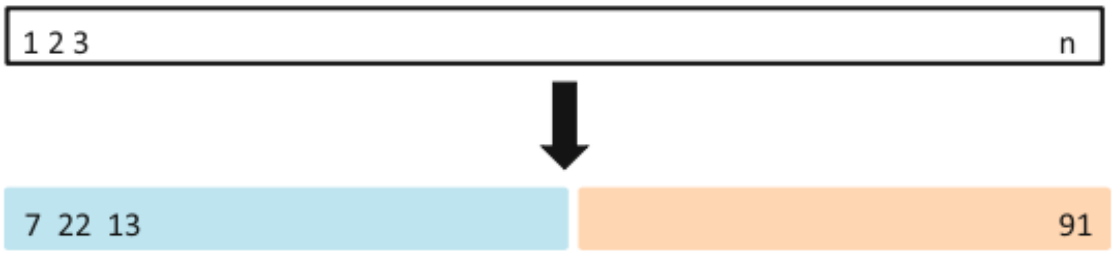

Figure 1.3. A schematic display of the validation set.

After that, it is possible to examine how the fitted models are different from each other. However, using the original training sample for fitting the linear regression once, is not going to give such information. Resampling approaches can be computationally cumbersome, since they consist of fitting the same algorithm again and again using different subsets of the training data. However, recent advances in computers have enabled us to make use of resampling methods [15].

The two most widely used resampling methods are cross-validation and bootstrap. Both methods are explained here and are going to be used in the next chapters.

1.6.1. Cross-Validation

If we would like to estimate the test error associated with a machine learning model on a set of observations, a very simple strategy is to divide the available observation set into two parts randomly, and fit the model to first part and then use it to predict the response in the second part or validation set. Note that the validation set is within the training set and is different from the final test set.

Figure 1.3 shows a set of n observations that splitted randomly into two parts [10]. The blue part can be used to fit the model and the orange part can be used to test the accuracy of the fitted model. If we try to estimate the test error rate just by doing this procedure we face with two problems. First, the validation estimate of the test error rate might be highly variable and not reliable.

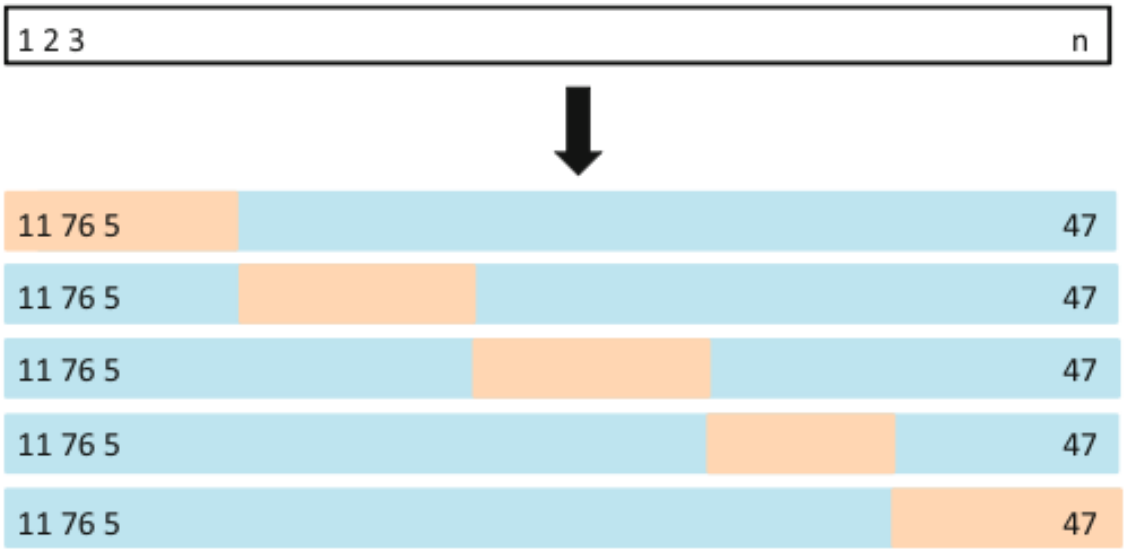

Figure 1.4. A schematic display of cross-validation.

Because it depends on directly which observations have been sampled for fitting and which are included in the validation set. Second, since we are using less data the model tends to perform worse and overestimate the test error rate. In order to mitigate these problems we can randomly divide the observations into k separated folds, train the model on k-1 folds and then test it on the held out fold. This way the model is going to be fitted and tested k times on k different validation sets. Then the average of the performances can be taken into account as a more reliable estimation for testing error rate.

Figure 1.4 shows a 5-fold cross-validation in which there are five different validation sets shown in orange and five different training sets, however they have some data in common, shown in blue [10].

The test error rate can be estimated by taking the average error resulting from fitting the model on these sets. The test error rate for a classification problem using k-fold cross-validation can be calculated using the equation

$$\mathbf{CV}_{(k)} = \frac{1}{k}\sum_{i=1}^{k}\mathbf{Err}_i \tag{1.3}$$

where $\mathbf{Err}_i = \mathbf{I}(y_i \neq \hat{y}_i)$. In this thesis, 10-fold cross-validation, which is the most common choice for model tuning, will be used in order to find and select the optimum model and determine its flexibility [16].

1.6.2. Bootsrap

Bootstrap resampling technique consists of repetitively taking random samples from the training data. A bootstrap is a random sample of the training data taken with replacement [17], meaning that a selected data point in the subset might be selected again for further selections.

Figure 1.5 represents such a procedure [10]. A bootstrap sample Z^{*j} has an identical size with the original data set (Z). Consequently, multiple number of an observation will be represented in the bootstrap sample while some observations will not be selected at all. Those not selected samples often called *out of bag* samples. A model will be built on the selected samples and evaluated its performance on the out of bag samples as the validation set.

Generally, Bootstrap resampling is a helpful tool to estimate the statistical properties of a random variable. For instance, it can be used to estimate the mean or variance of the errors done by a certain machine learning algorithm [17]. Bootstrap is an essential tool for building the Random forest model which will be discussed in chapter 2.

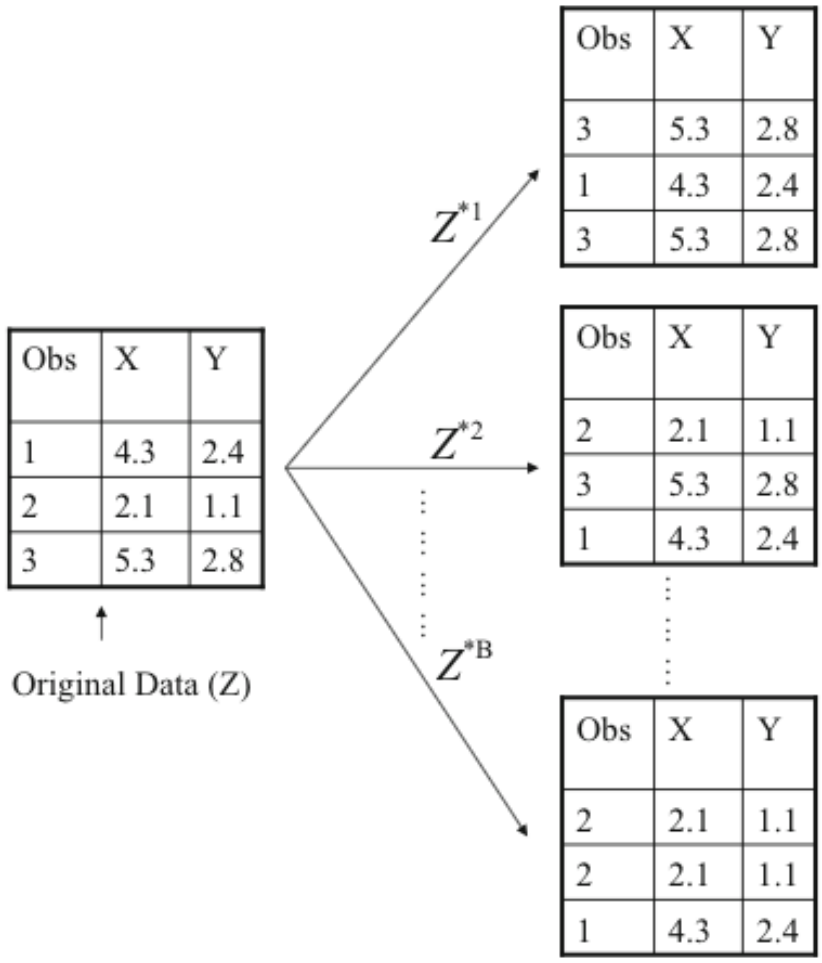

Figure 1.5. A schematic display of bootrap resampling.

1.7. Thesis Outline

The focus of this thesis is on introducing four different machine learning algorithms namely QDA, Random Forest, SVM and ANN. The details of modeling using these methods in order to solve Liquefaction prediction, which is one of the significant problems in geotechnical engineering, is provided. Then the performance of the models are compared to each other. This thesis is organized as follows.

Chapter 2 includes an introduction to QDA, ANN, Random Forest and SVM. It provides the structure of each model and presents how they are able to solve complex classification problems. Also the properties plus advantages and disadvantages of each model will be discussed in this chapter. Furthermore, It will be shown what are the tuning parameters for each model and how they can be optimized.

Chapter 3 is dedicated to applying the machine learning models discussed in the previous chapter on a liquefaction database in order to classify liquefied cases and non-liquefied ones. The structure of the database and the available features will be presented. The feature selection procedure and data preprocessing are also explained in this chapter. Finally the trained models will be compared to each other in order to select the best candidate.

In Chapter 4 the best chosen model, will be compared to the Simplified-approach, which is introduced briefly in the chapter, for assessing the liquefaction-risk.

Finally, the last chapter is summary and conclusion. It also addresses some suggestions for future works.

2. MACHINE LEARNING ALGORITHMS

Though, there are many machine learning algorithms available to use, four of them namely QDA, Random Forest, SVM and ANN are chosen in this thesis as they are widely used for different applications in recent years. This chapter talks in depth about each approach and the upsides and downsides of them.

2.1. Quadratic Discriminant Analysis

Quadratic discriminant analysis also known as QDA is an extension to Fisher's proposed approach, *linear discriminant analysis* or LDA which is based on Bayes theorem of conditional probability. The main idea is to estimate the probability of a certain event to happen given some information about it. Mathematically speaking it is desired to calculate $Pr(Y|X)$, where Y is the dependent variable of interest and X denotes a vector of predictor variables. Several methods have been proposed by researchers in order to model $Pr(Y|X)$ such as logistic regression and LDA. These models often assume an extreme assumption about the problem which then results in huge bias unless the assumption conforms with reality. QDA on the other hand relaxes this issue by giving more variability to reduce the bias. However, QDA also tries to simplify the problem by taking some assumptions and it is not considered as a very flexible model. But its simplicity plus being less biased compared to linear approaches makes it an interesting choice.

2.1.1. Classification Using Bayes Theorem

Suppose that we want to classify an observation into k classes. It has been proved that using *Bayes theorem* if a classifier puts an observation into the class for which the probability of belonging to that class is largest, makes the least misclassifications among all the classifiers and it is known as *Bayes Classifier*. In other words, after calculating $P(Y_k|X)$ for each K, given information about that observation, Bayes classifier picks the kth class for which the calculated conditional probability is highest [18].

Bayes theorem is given as

$$Pr(Y|X) = \frac{P(Y \cap X)}{P(X)}. \tag{2.1}$$

Equation (2.1) can be expanded to

$$Pr(Y = k|X = x) = \frac{P(Y_k).f_k(x)}{\sum_{i=1}^{k} P(Y_i).f_i(x)} \tag{2.2}$$

where $Pr(Y = k|X = x)$ is the probability that an obsevation with X = x comes from the kth class. For brevity we refer to $Pr(Y = k|X = x)$ as $p_k(X)$ which can also be called as *posterior* probability. $P(Y_k)$ is often known as *prior* probability that a random sample belongs to the kth class. And $f_k(X) \equiv Pr(X = x|Y = k)$ denotes the density function of X for an observation in the kth class. We expect $f_k(x)$ to be relatively large if there is high probability that a random sample in the kth class has $X \approx x$ and $f_k(x)$ to be relatively small if it is a few chance that a random sample in the kth class has $X \approx x$. Estimating $P(Y_k)$ is often simple in case we have a random sample of Y s from the population. It is proposed to calculate the ratio of the training observations that are coming from the kth class. In contrast, estimating $f_k(x)$ can be more challenging unless we assume some simple forms for such densities.

2.1.2. Estimating Probability Density Function $f_k(x)$

In order to calculate the posterior probability we need to estimate the distribution of the population in each of the k classes. For simplicity it is assumed that $f_k(x)$ is *Gaussian* or normally distributed. Approximation of $f_k(x)$ by Gaussian distribution can be done using Equation (2.3):

$$f(x) = \frac{1}{(2\pi)^{p/2}|\Sigma_k|^{1/2}} exp(-\frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1}(x - \mu_k)) \tag{2.3}$$

This is known as multivariate gaussian density function. Where Σ_k is the covariance matrix in the kth class and μ_k is the associated mean vector.

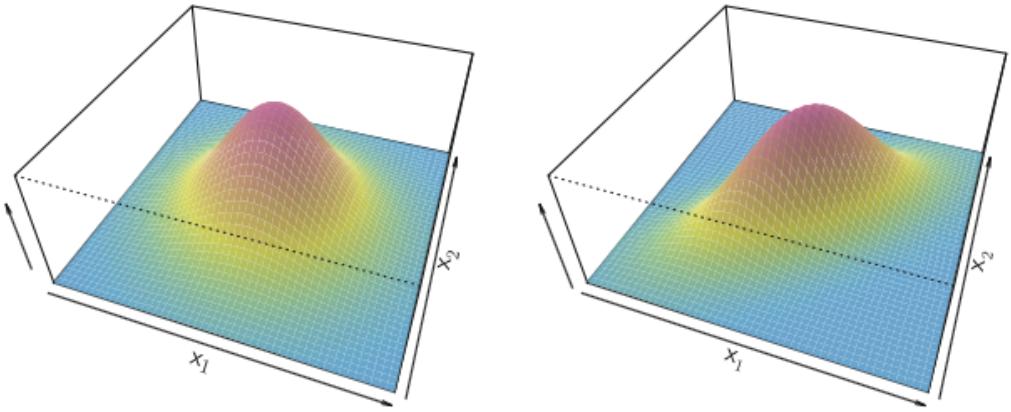

Figure 2.1. Multivariate Guassian functions with different covariances.

The parameters of this distribution can be calculated using the available sample data. In Figure 2.1 the left hand panel is associated with a multivariate Gaussian function with two uncorrelated variable, while in the right hand side the two variables are correlated [10].

2.1.3. Finding Decision Boundry

To solve a classification problem we need a line(s) for separating different classes. We refer to such a line as decision boundary [19]. So far we managed to estimate the parameters in the Equation (2.2) in order to find $p_k(x)$ and assign an observation to the class for which $p_k(x)$ is largest. Taking the log function of the Equation(2.2) and rearranging its terms it can be shown that we can use Equation(2.4) instead, that is:

$$\delta_k(x) = -\frac{1}{2}(x-\mu_k)^T{\Sigma_k}^{-1}(x-\mu_k) - \frac{1}{2}log|\Sigma_k| + logP(Y_k) \tag{2.4}$$

Now it can be noticed that the value x appears as a quadratic function and that is why this approach named as quadratic discriminant analysis. Using $\delta_k(x)$ the shape of decision boundary can be obtained. To better understand the concept of the QDA approach and finding the decision boundary let us introduce a very simple classification problem with only one predictor and two classes.

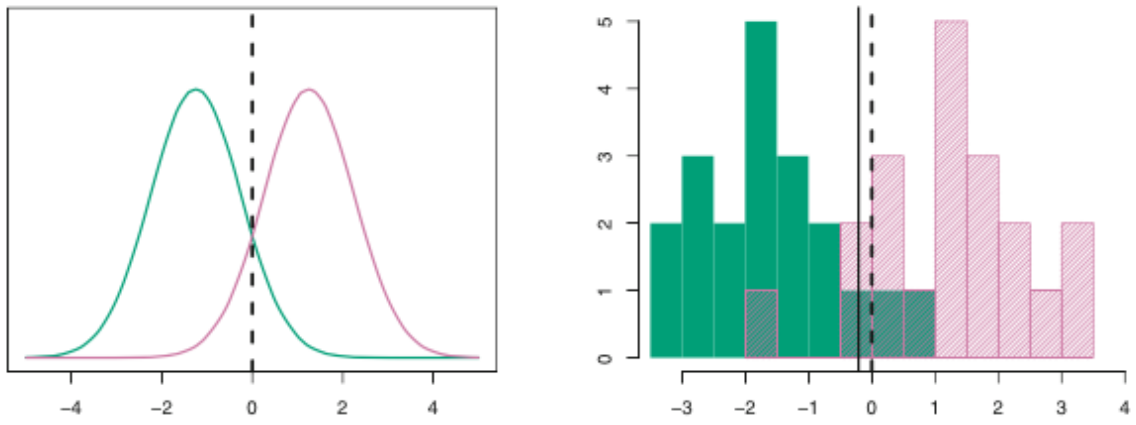

Figure 2.2. Left: Two one-dimensional normal density functions are shown. The dashed vertical line represents the Bayes decision boundary. Right: 20 observations were drawn from each of the two classes, and are shown as histograms. The Bayes decision boundary is again shown as a dashed vertical line. The solid vertical line represents the QDA decision boundary estimated from the training data.

In the right hand side of Figure 2.2 a histogram of observations in two classes is provided. Then the distribution of those observations are approximated by a Gaussian density function and it has been shown in the left panel. The dashed line is associated with the decision boundary in which $p_k(x)$ is equal for each classes [10]. In case the number of predictors is two, the Gaussian density function is shown in the Figure 2.2. In Figure 2.3 the purple dashed line is associated with the Bayes classifier decision boundary, the black dotted line is associated with LDA decision boundary and the solid green line is associated with QDA decision boundary. Comparing the left panel and right panel of this figure it can be seen that QDA is more flexible than LDA and it can perform better in more complex problems [10]. The superiority of QDA over other linear methods such as LDA can also be explained in terms of bias-variance trade off. In problems where QDA can decrease the bias significantly with the cost of increasing a little variance, better performance will be expected. However, if the cost of introducing variance is bigger than the amount by which the bias error mitigated than QDA test error would be higher. As it is mentioned before QDA itself is not considered as a very flexible model and in complex problems would give biased results.

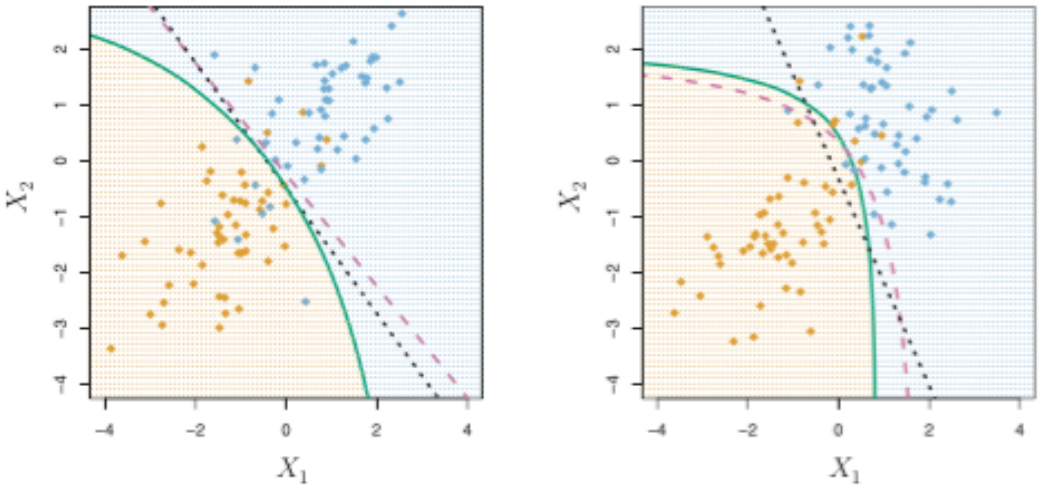

Figure 2.3. Left: The Bayes (purple dashed), LDA (black dotted), and QDA (green solid) decision boundaries for a two-class problem with $\sum_1 = \sum_2$. The shading indicates the QDA decision rule. Right: Details are as given in the left-hand panel, except that $\sum_1 \neq \sum_2$.

In the next sections more flexible approach will be introduced that can work better when the problem is highly complex and simpilified assumptions may cost in large errors [20].

2.2. Random Forest

Random forest is one of the powerful machine learning techniques for classification and prediction. This approach is developed based on the concept of decision trees in which the algorithm segments the predictor space into subregions using some splitting rules. Then in order to make a prediction, mode of each region considered as decision rule. Although decision trees are not competitive with some other supervised learning approaches in terms of accuracy, but combining multiple decision trees will result in a more powerful model known as bagged trees which then can be further improved to build a Random forest model. So before delving into the details of Random forest model, the basics of decision trees and bagged trees are explained.

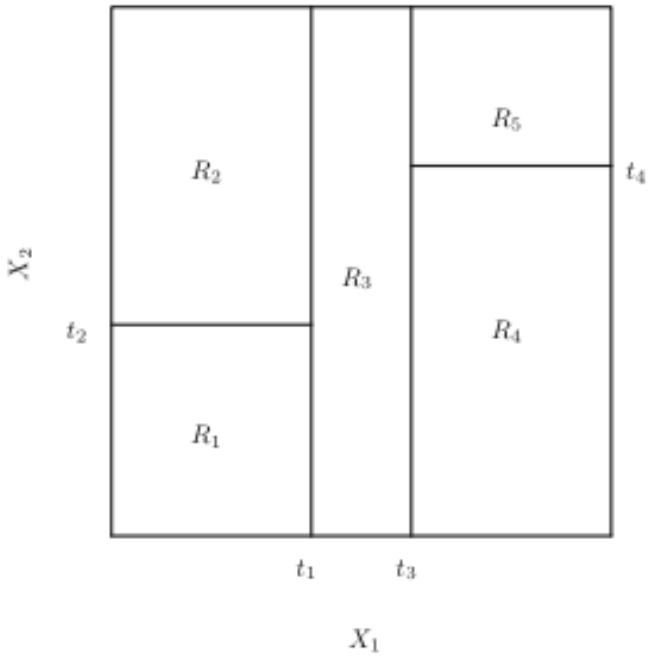

Figure 2.4. A shematic of a two-dimensional predictor space splitted into rectangular regions by a decision tree.

2.2.1. Decision Trees Fundamentals

Basically decision trees are trying to simplify problems by segmentation. They build rectangular regions in the predictor space and consider region specific responses. For prediction, decision trees find the region to which an observation of interest belongs and based on that they can predict the response. Figure 2.4 shows how the predictor space would look like after a decision tree makes segmentation on it. As it is shown five different regions namely $R_1, R_2, ..., R_5$ has been created by certain criteria or splitting rules for which t_1, t_2, t_3 and t_4 are splitting points [10]. Then for all of the observations in a distinct region R_J a same response will be considered. Such procedure can be summarized in two steps:

- Dividing the predictor space that is, the set of possible quantities for $X_1, ..., X_p$, into J non-overlapping separated areas, $R_1, ..., R_J$.

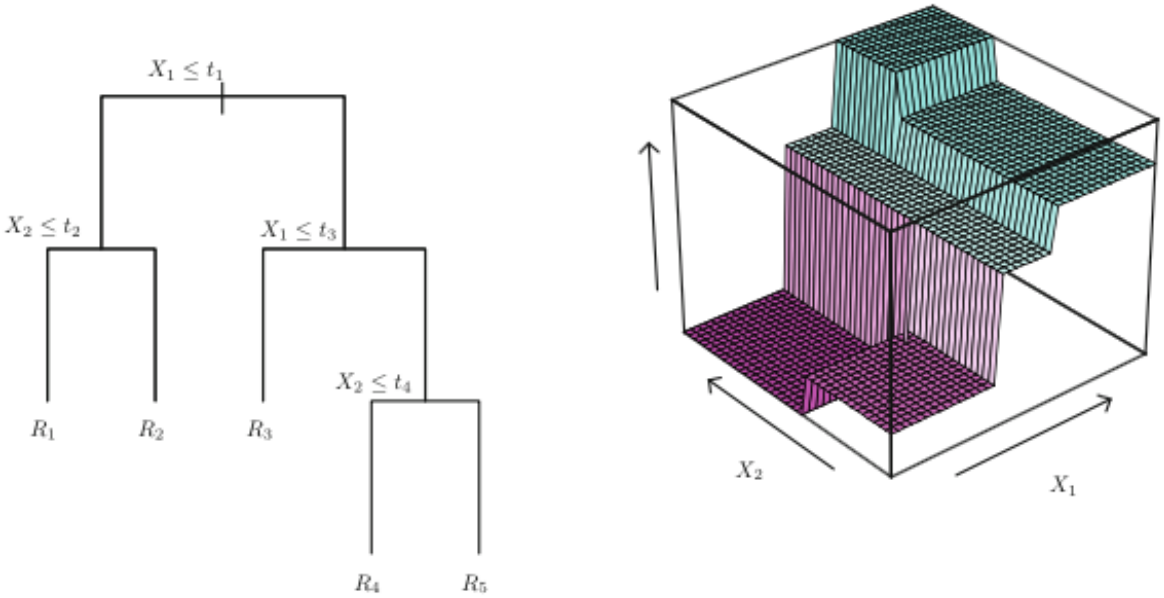

Figure 2.5. Left: A tree corresponding to the partition in the top right panel. Right: A perspective plot of the prediction surface corresponding to that tree.

- For any observations that falls into the the area R_J, the same prediction class should be considered, which is the most accuring class in that region.

For instance, by looking at the Figure 2.4, any observation with $X_i = (X_1 < t_1, X_2 < t_2)$ falls into region one or R_1 in which kth class might be dominated. Then a decision tree predicts that X_i comes from the kth class. Note that in each of $R_1, ..., R_J$ regions all the classes may be exist as a mixture. The ideal decision tree is a model which can purify each regions to decrease the misclassification error. The left panel of the Figure 2.5 provides the structure of a decision tree that shows the hierarchy of the approach is top-down because it starts from the top of the tree where splitting the regions begins [10]. The goal is to find the best splitting point that reduces the error at each step. This *greedy* approach often called *recursive binary splitting*. It is greedy because the algorithm does not consider the future of each split and only cares about how much reduction of error will be obtained by the split at each step. So it does not look ahead to find the best possible split which may result in significant error reduction in future steps.

Considering the *tree* analogy, regions $R_1, ..., R_J$ are often referred to as leaves or terminal nodes. Along the tree there are some points, where the splitting procedure happened, are known as *internal nodes*. The right panel of Figure 2.5 demonstrates the perspective plot of the prediction plane associated with that tree [21].

2.2.2. Recursive Binary Splitting

For building a decision tree, at each step a predictor and its corresponding cut point for split should be considered. The criteria for such a procedure is set using the *Gini index* which is defined by [8]

$$G = \sum_{i=1}^{k} \hat{p}_{mk}(1 - \hat{p}_{mk}) \tag{2.5}$$

where $\hat{p}_{mk}$ determines the ratio of training observations available in the mth segment that are correspond to the kth class. For a problem with two classes, the Gini index for a node m is calculated as

$$G = \hat{p}_{m1}(1 - \hat{p}_{m1}) + \hat{p}_{m2}(1 - \hat{p}_{m2}). \tag{2.6}$$

For a two-class problem $\hat{p}_{m1} + \hat{p}_{m2} = 1$, as a result the right hand side of the Equation (2.6) can be replaced by $2\hat{p}_{m1}\hat{p}_{m2}$. Therefore, the Gini index will be minimized when either of the $\hat{p}_{mk}$ is driven towards zero. In other words, the m node is pure with respect to one of the two classes. In contrast, the index will be maximized if $\hat{p}_{m1} = \hat{p}_{m2}$ which means the node m is least pure. It can be implied that the Gini index represents a node's purity. A decision tree picks the predictor X_j and its corresponding cut point value t to split the predictor space into the distinct regions $\{X|X_j < t\}$ and $\{X|X_j \geqslant t\}$ that leads to purest nodes using the Gini index measurement. The recursive binary splitting procedure goes on until a stopping criteria has been reached. This is usually a certain number of observations in the terminal nodes.

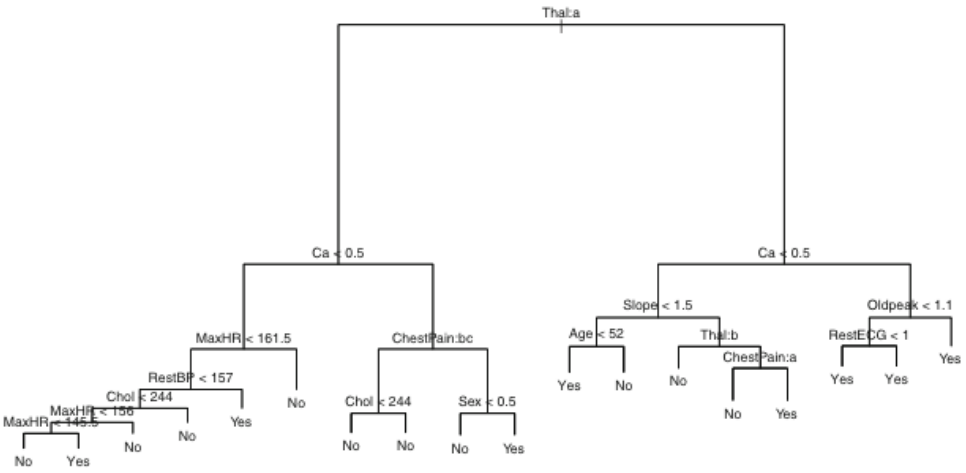

Figure 2.6. An unpruned tree that has many leaves.

2.2.3. Bagged Trees

After the tree has been grown, using the splitting procedure explained before, the model often shows a good performance on the training data and a poor performance on the test data set. Such a case often referred to as overfitting, which is described in the previous chapter, due to model complexity. A simpler tree with fewer splits might lower the variance significantly at the cost of a little bias. The remedy for the mentioned overfitting problem is to prune back the tree to obtain the desired tree which performs well on the test data set. Figure 2.6 shows a hypothetical tree which is not pruned yet and it has a lot of leaves. Such a tree makes interpretation harder [10]. For pruning a tree there is a way called *weakest link pruning*, also known as *cost complexity pruning* that enables us to find a subset of the grown tree which is simpler and is not suffering from overfitting. However, such procedure is out of the scope of this thesis.

Another strategy to decrease the variance and avoid overfitting is to use averaging, which is the basis of bagged trees. If Z is a random variable and $Var[Z] = \sigma^2$ then It has been proved that $Var[\mathrm{Z}] = \sigma^2/\mathrm{n}$, where n is the number of observations. In other words, averaging decreases the variance.

Thus, by taking different training samples from the population it is possible to build a distinct decision tree using each training set and take the average of the resulting predictions in order to reduce the variance which then results in higher prediction accuracy. Often, there is only one training set available and it is not applicable to take multiple different samples from the population to do such a procedure. However, in the absence of multiple training sets it is possible to use bootstrap resampling approach described in the previous chapter.

Bootstraping simulates the situation in which we have different training sets by repeatedly taking samples from the single training data set. Such an approach consists of generating different bootstrapped training data sets and growing a decision tree using each training data set to have multiple decision trees. Then for a given observation each tree model gives its prediction and the final result can be obtained by taking the average of all the predictions. This procedure is called bagging which can be defined as

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x) \tag{2.7}$$

where B is the number of bootstrapped samples, $\hat{f}^{*b}(x)$ is the result for bth sample and $\hat{f}_{bag}(x)$ is the final bagged result.

2.2.4. Basics of Random Forest

Random forest is the improved version of bagged trees that applies a fine adjustment to it. Hence, random forest is a tree-based model too. Just like bagged trees, random forest uses boostrap resampling method to create multiple training sets and grows decision trees using each ot them. The only difference is that random forest uses a trick for growing these trees at each split. It randomly chooses m predictors and it only searches within those randomly chosen predictors to find the best possible split [22]. Typically m will be determined as $m \approx \sqrt{p}$, where p is the total number of predictors.

However, a better approach is to find the optimized value for m using cross-validation as a model tuning technique.

The rationale behind considering m randomly selected predictors instead of all predictors is that often one predictor is stronger than the others, as a result, all of the grown trees will use this strong predictor as their candidate for best split at top of the trees. Therefore, all the trees would be similar to each other resulting in highly correlated predictions. It turns out averaging several correlated values doesn't lead to significant reduction in variance compared to uncorrelated values. This means bagged trees are not able to reduce the high variance of a single tree by far.

Random forest solves this problem by giving more chance to some of the weaker predictors to be chosen over the stronger predictors at different splits. This procedure could be considered as *decorrelating* the trees which makes their average predictions less variable and more reliable.

2.3. Support Vector Machine

Support vector machine (SVM) is an extension for two other methods namely *maximal marginal classifier* and *support vector classifier*, and it is developed by Vapnik [23], for more complex problems with non-linear responses. Therefore, before giving the details of the SVM model, the basics of maximal marginal classifier and support vector classifier are introduced.

2.3.1. Maximal Marginal Classifier

If a set of observations from two classes can be separated perfectly by a separating hyperplane, they can be separated by infinite number of different hyperplanes too as it is shown in the Figure 2.7 [10]. But they are different in terms of how much distance they have from the training observations or their *margin* which can be defined as the perpendicular distance of each data points to that separating hyperplane.

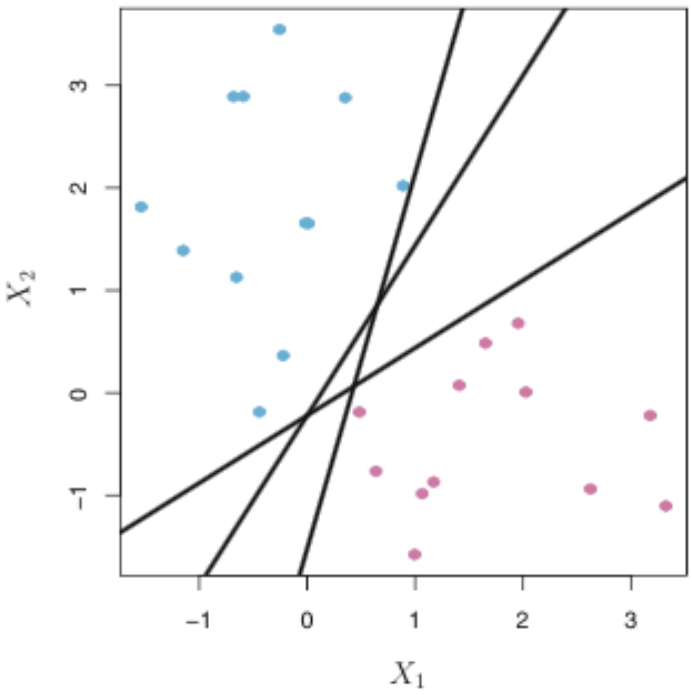

Figure 2.7. There are two classes of observations, shown in blue and in purple, each of which has measurements on two variables. Three separating hyperplanes, out of many possible, are shown in black.

Maximal marginal classifier tries to find a farthest hyperplane, which is a $p-1$ dimensional subspace of a p dimensional space, from the training observations and separates the N observations $(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)$, with $x_i \in \mathbb{R}^p$ and $y_i \in \{-1, 1\}$.. Any linear separating hyperplane has these properties:

$$if\ y_i = 1 \rightarrow \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip} > 0 \tag{2.8}$$

and

$$if\ y_i = -1 \rightarrow \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip} < 0. \tag{2.9}$$

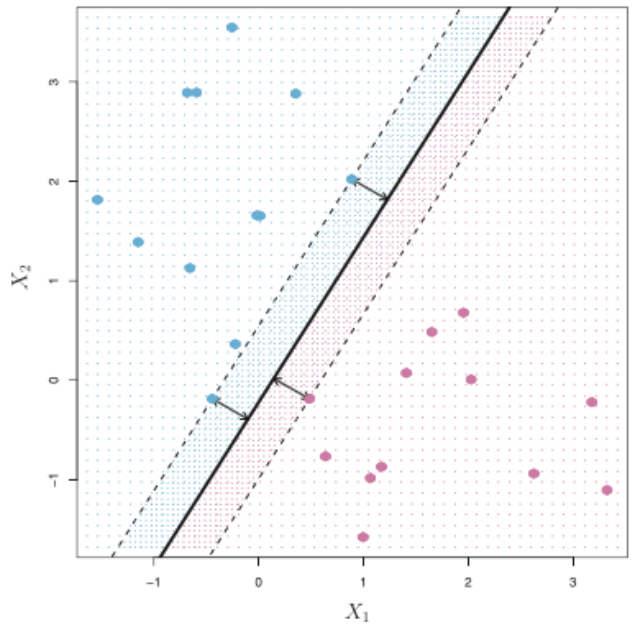

Figure 2.8. There are two classes of observations, shown in blue and in purple. The maximal margin hyperplane is shown as a solid line. The margin is the distance from the solid line to either of the dashed lines.

Alternatively it can be stated that, for a separating hyperplane Equation (2.10) is true.

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip}) > 0 \tag{2.10}$$

After constructing a separating hyperplane it is possible to predict the class of a test observation based on which side of the hyperplane it is located. Finding a separating hyperplane with maximal margin, like the one shown in Figure 2.8 [10], is being done by solving the following optimization problem:

$$\begin{aligned} &\underset{\beta_0,\beta_1,...,\beta_p}{\text{maximize}} M \\ &\text{subject to} \sum_{j=i}^{p} \beta_j^2 = 1 \\ &y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip}) \geq M \ \forall i = 1, ..., n \end{aligned} \tag{2.11}$$

The optimization problem in Equation (2.11) is a convex optimization problem which can be solved efficiently, using the Lagrange function. The details of such procedure is out of the scope of the thesis as the computer can take care of this part with no problem. Finding the maximal marginal hyperplane ensures that every single observations are in the correct side of the hyperplane and they have at least a margin M from the hyperplane.

2.3.2. Support Vector Classifier

The problem with maximal marginal classifier is that it is only applicable when the observations of two classes are not mixed otherwise the maximal marginal hyperplane would not exist. However, it is possible to extend the concept of maximal marginal classifier to the non-separable cases. This can be done by applying some adjustments to the mentioned optimization problem as follows:

$$\begin{aligned} &\underset{\beta_0,\beta_1,...,\beta_p,\epsilon_1,...,\epsilon_n}{\text{maximize}} M \\ &\text{subject to} \sum_{j=i}^{p} \beta_j^2 = 1 \\ &y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip}) \geq M(1-\epsilon_i) \ \forall i = 1, ..., n \\ &\epsilon_i \geq 0, \sum_{i=1}^{n} \epsilon_i \leq C \end{aligned} \tag{2.12}$$

Here C can be considered as a nonnegative tuning parameter. Again M is the width of the margin that it supposed to be maximized. $\epsilon_1, ..., \epsilon_n$ are known as *slack variables* which relax the constraint of being perfectly on the right side of the margin and the separating hyperplane. If $\epsilon_i = 0$ then the ith observation is on the correct side of the margin. If $0 < \epsilon_i < 1$ then the ith observation violated the margin but it is on the correct side of the hyperplane, finally if $\epsilon_i > 1$ then the ith observation it is on the wrong side of the hyperplane. This adjustment allows to obtain an optimized separating hyperplane for mixed observations. Support vector classifier predicts the class of a test observation x^* based on which side of the separating hyperplane it is located and can be represented by the sign of $f(x^*) = \beta_0 + \beta_1 {x_1}^* + ... + \beta_p {x_p}^*$.

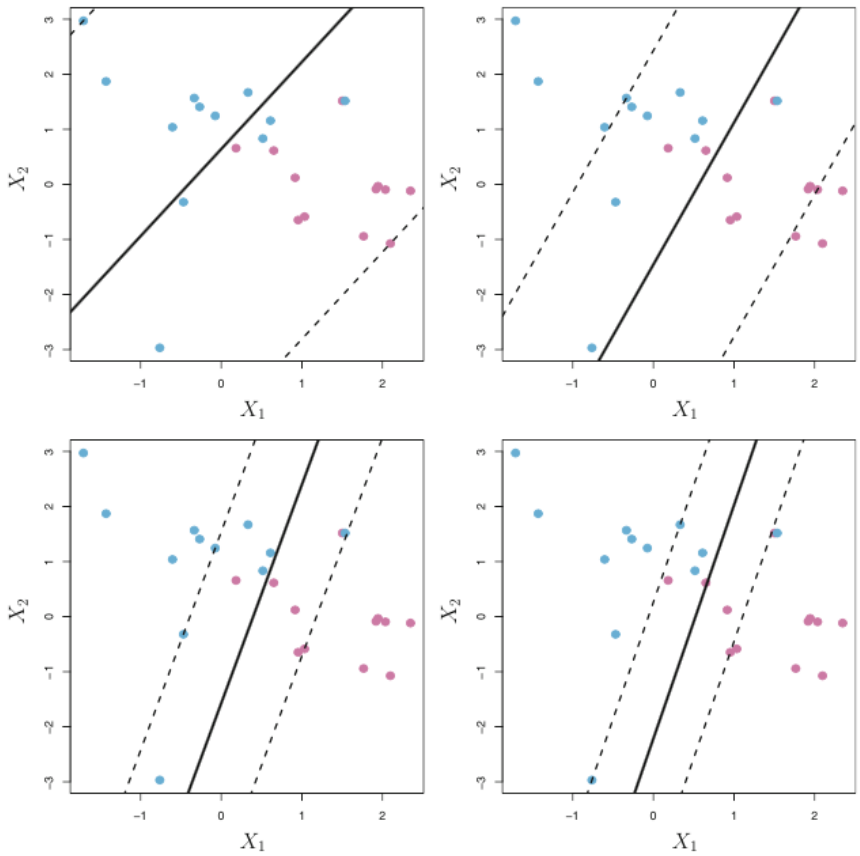

Figure 2.9. A support vector classifier was fit using four different values of the tuning parameter C. The largest value of C was used in the top left panel, and smaller values were used in the top right, bottom left, and bottom right panels. When C is large, then there is a high tolerance for observations being on the wrong side of the margin.

Unlike the maximal marginal classifier, support vector classifier is allowed to do some misclassifications which of course should be minimized by tuning the model.

2.3.3. Tuning the Support Vector Classifier

As it is mentioned before the value C is a tuning parameter. Generally speaking, C can be seen as a budget for defining to what extent violating the margin is allowed. In the Figure 2.9 [10] it can be seen how changing the value of C changes the width of the margin. Obviously setting $C = 0$ will result in a situation in which no violation to the margin is tolerable.

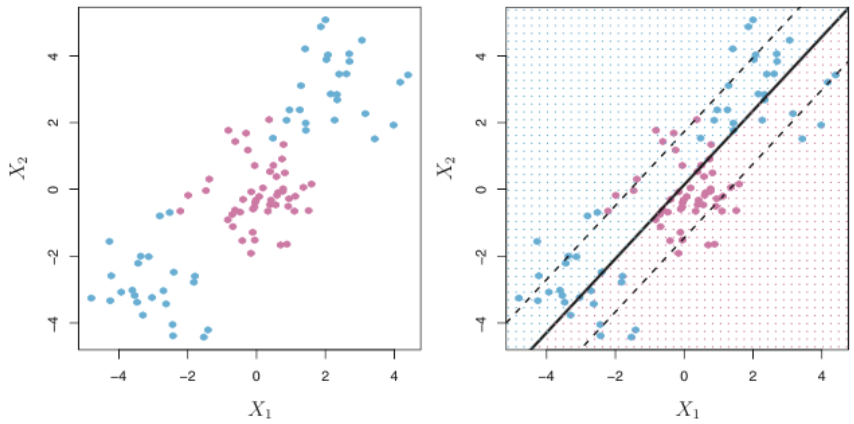

Figure 2.10. Left: The observations fall into two classes, with a non-linear boundary between them. Right: The support vector classifier seeks a linear bound- ary, and consequently performs very poorly.

That is the case in the maximal marginal classifier, which of course does not exist when the two classes are not separable. On the other hand, increasing the C makes the model more tolerable of violation to the margin resulting in widening the margin. In contrast, decreasing the C narrows the margin. One of the important properties of the optimization problem in Equation (2.12) is that only observations that violate the margin or lie on it will affect the answer. In other words, changing the position of the observations that are located in the correct side of the margin will not change the margin and hence will not change the classifier at all. Observations which lie on the margin, or violate it for their class, are known as *support vectors*. These are the observations with effects on the support vector classifier. Therefore, a support classifier with larger C and wider margin has more support vectors. This means, there are more observations that are involved to determine the hyperplane.

2.3.4. Basics of Support Vector Machine

In some cases, where the boundaries of classes are not linear, any linear method including support vector classifier will perform poorly as it is shown in Figure 2.10. In such a case the support vector classifier is almost useless [10].

The support vector machine (SVM) is an extension to the support vector classifier that is more flexible and it can perform well when the boundaries are non-linear. This is done using a different *kernel* ,which will be introduced later in the chapter.

The details of solving the mentioned optimization problem are not addressed here, however, it turns out its solution only depends on the dot product of the observations. The dot product of two observation can be defined by

$$\langle x_i, x_{i'} \rangle = \sum_{j=1}^{p} x_{ij} x_{i'j}. \tag{2.13}$$

Accordingly, the linear support classifier can be determined by

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x_{ij}, x_{i'j} \rangle \tag{2.14}$$

where $\alpha - 1, ..., \alpha_n$ will be estimated from n training observations. It can be proved that α_i for non-support vectors in the training data set equals to zero. Also it is possible to replace the dot product in the Equation (2.14) with a generalized form

$$K(x, x_i) \tag{2.15}$$

where K referred to as a *kernel* [24], which can be defined as various functions of dot product. Thus, the Equation 2.14 can be generalized as

$$f(x) = \beta_0 + \sum_{i \in S} \alpha_i K(x, x_i) \tag{2.16}$$

where S is the collection of indices for support vector points and K is the kernel function. Implementing different kernel functions allows to produce non-linear boundaries and have more flexible models. Such a classifier with non-linear kernel is known as support vector machine.

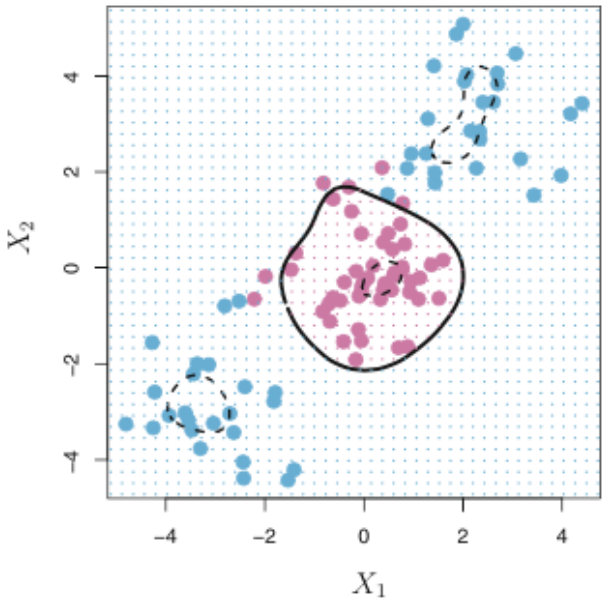

Figure 2.11. A support vector machine with a radial kernel fitted to a training set with non-linear boundries.

One of the popular non-linear kernels is the *radial kernel* and has the form

$$K(x_i, x_i') = exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2) \tag{2.17}$$

where γ is a positive constant. Using the radial kernel will result in non-linear boundaries that can be used in more complex problems in which the linear methods such as support vector classifier performs very disappointing. In the Figure 2.11, a support vector machine with radial kernel has been used on the same training data presented in the Figure 2.10 to classify the two classes and it is shown how successful it is in separating the two classes.

2.3.5. Tuning the Support Vector Machine

The cost budget C and γ in the radial kernel are tuning parameters which are going to define the flexibility of the SVM model. So far, it is shown that increasing the C increases the margin which results in more observations to be included for defining the separating boundary, thus decreasing the variability of the model.

Conversely, increasing γ results in higher variability and more flexible model. Finding the optimized value for C and γ can be done by cross-validation that enables fitting different models with different tuning parameters and evaluate their performances in order to choose the best one [19].

2.4. Artifical Neural Network

Neural networks are powerful models for prediction, inspired by the brain function. It is being suggested that brain's great learning ability comes from the structure of neurons in it. Neurons, although there are various forms of them, all transmit an electrical signal from one end to the other, from the dendrites along the axons to the terminals. Figure 2.12 shows an example of a neuron. It takes an electric input, and pops out another electrical signal [25–27]. Observations suggest that neurons do not react readily, instead suppress the input until it has grown so large that it triggers an output. It can be thought of this as a threshold that must be reached before any output is produced. A function that takes an input signal and generates an output signal, but takes into account some kind of threshold, is called an activation function. Mathematically, there are many such activation functions that could achieve this effect. A sigmoid function also known as logistic function, owns such a property and it is given as

$$y = \frac{1}{1 + e^{-x}} \tag{2.18}$$

and it is being used in the structure of the artificial neural network which is analogous to the structure of a biological neuron.

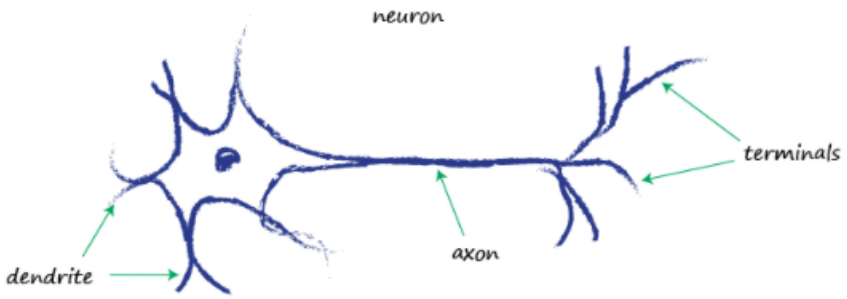

Figure 2.12. A shematic picture of a neuron.

Furthermore, real neurons can take multiple input signals and generate output signals. It can be thought that they are communicating with each other by taking information from other neurons and pass them on. Mimicking such a procedure is the fundamental of the artificial neural network in which some nodes are defined as neurons which can take weighted inputs and generate outputs using the sigmoid function. A typical artificial neuron with n inputs can be defined by the formula

$$y(x) = f(\sum_{i=1}^{n} \omega_i x_i) \tag{2.19}$$

where $\omega_i x_i$ is the weighted input from the ith neuron and f is the activation function. An artificial neuron can be depicted as it is shown in the Figure 2.13 [9]. Neurons like this are the building blocks of an artificial neural network. The topology of artificial neural networks will be introduced to show how such neurons can build a network which is able to learn.

2.4.1. Artifical Neural Network Topology

The way neurons are connected with each other is essential for an ANN model. Artificial neurons are being distinguished based on their location in the network [28]. A simple structure ANN is shown in the Figure 2.14 [9]. The nodes (neurons) in this network are arranged in groups called *layers*. x_1, x_2 and x_3 are known as input nodes.

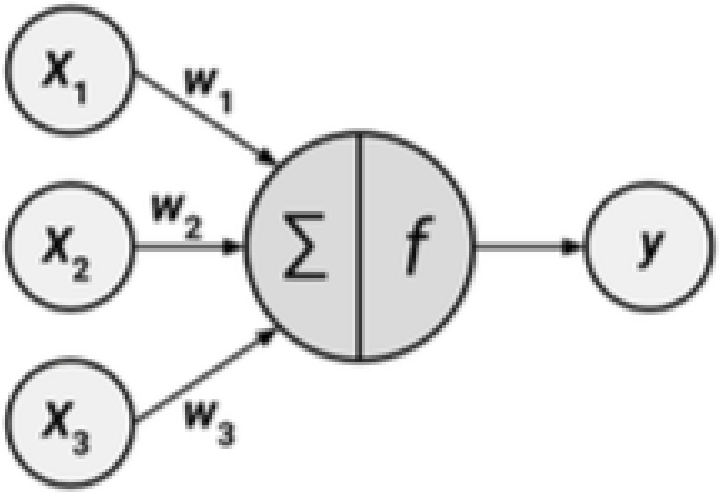

Figure 2.13. A shematic picture of an artificial neuron.

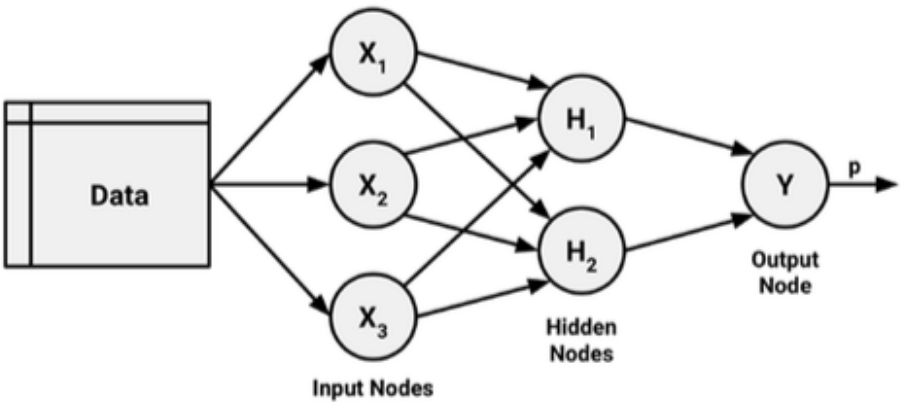

Figure 2.14. A single hidden layer ANN.

Input nodes are responsible for taking the initial values from the database and generate outputs using the Equation (2.19). H_1 and H_1 are referred to as hidden nodes, and they also follow the Equation (2.19) to generate output. The layer which includes them is called hidden layer. This is a simple ANN with three inputs, one hidden layer including two hidden neurons and only one output. It is possible to add more hidden layers between the input layer and the output node. Each node in a hidden layer is fully connected to other nodes in the vicinity layers. Equation (2.19) suggests that the connections between nodes are linear.

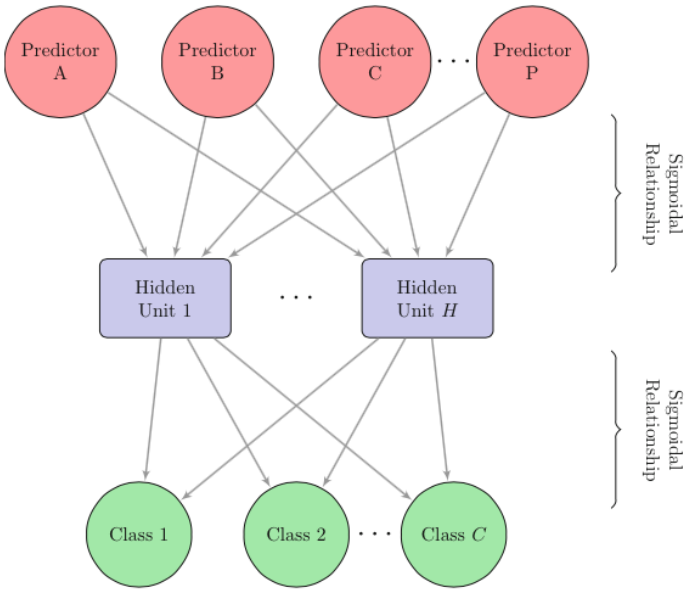

Figure 2.15. A typical ANN model constructed for a classification problem with C classes.

The weights in this linear connections should be adjusted in a way that from a certain input values, the desired output can be obtained. As a result, through the process of learning some connections fade away while some will become larger in order to give the most accurate answer. Such a learning process is similar to how humans get better at doing something by practice which makes certain neural connections in their brain stronger.

2.4.2. Learning Process in the ANN

An untrained ANN generates the weights ω randomly which results in a random answer. The learning process consists of feeding the neural network with some examples in order to adjust the connection weights, in a way that next time it gives a closer respond compared to the correct answer. For a classification problem with C classes, an ANN should have C output nodes as it is depicted in the Figure 2.15 [15]. An untrained ANN model generates random numbers between zero and one for each of those C output nodes. However, it is desired to get one for the class that an observation comes from and zero for all the others.

Although, each class is predicted by a linear combination of the hidden neurons outputs that have values between zero and one, as they are outputs of the sigmoid function, but they are not add up to one, therefore they are not "probability-like". To ensure that outputs of the neural network are adding up to one, there should be an additional constraint for the network outputs that is:

$$f^*_{i\ell}(x) = \frac{e^{f_{i\ell}(x)}}{\sum_l e^{f_{il}(x)}} \tag{2.20}$$

where $f^*_{i\ell}(x)$ is the model prediction for the ith sample and the ℓth class. So the neural network should optimize the Equation (2.21) in order to find appropriate weight estimates.

$$\sum_{\ell=1}^{C}\sum_{i=1}^{n}(y_{i\ell} - f^*_{i\ell}(x))^2 \tag{2.21}$$

In this equation, $y_{i\ell}$ is the 0/1 indicator of class ℓ. The back-propagation algorithm [29–31] is a popular approach in order to optimize the connection weights for training the neural network. A neural network can learn by comparing its output given an input with the target output to calculate its error. Then the error is propagated back through the network changing the weights, this is referred to as back-propagation procedure. Minimizing the error can be done using the gradient descent method with a proper learning rate which is proposed to be not greater than 0.1 otherwise the algorithm may not converge at all. Through this procedure the errors will be minimized until a stop condition has been reached.

2.4.3. Tuning the ANN

The topography of a neural network contributes to the flexibility of the model. Increasing the number of hidden layers and hidden neurons will increase the complexity of the model as the number of connections and calculated weights becomes larger [27].

A complex neural network is very variable and unstable and makes it susceptible to overfitting. As a result a regularization method has been introduced to control the variability of the model. This can be done using *weight decay*, which is a penalization method [32]. The idea is to penalize the model over producing large weights. In other words, any large value for weights must have a considerable effect on the model errors to be tolerated. This idea can be implemented by adjusting the Equation (2.21) through adding a penalization term $\lambda \sum \omega^2$. Increasing the λ puts larger penalties on larger weight coefficients. This approach can help stabilizing the model.

In conclusion, the number of hidden layers, number of hidden neurons and a weight decay value λ are the effective parameters for model's accuracy which can be found using cross-validation technique.

3. ASSESSING LIQUEFACTION POTENTIAL USING MACHINE LEARNING APPROACHES

Some of the most horrifying examples of earthquake damage occurred when soil deposits lost their strength and appeared to flow as fluids. In this phenomenon, termed liquefaction, strength of a soil is reduced, often drastically, to the point where it is unable to support structures or remain stable anymore. Because it only occurs in saturated soils, liquefaction is most commonly observed near rivers, bays, and other bodies of water. The phenomenon of level-ground liquefaction does not involve large lateral displacements but is easily identified by the presence of sand boils produced by groundwater rushing to the surface. Althougth not particularly damaging by themselves, sand boils indicate the presence of high groundwater pressures whose eventual dissipation can produce subsidence and damaging differential settlements [33, 34].

Liquefaction is a complicated phenomenon, but research has progressed to the point where an integrated framework of understanding can be developed. This chapter focuses on the evaluation of liquefaction potential using in-situ measurements and analyzing the obtained information via machine learning algorithms. This procedure is done using R system, which is an open source programming language, and CARET package that can be installed in R.

3.1. Database Used in Development of Various Machine Learning Models

In this thesis, a liquefaction database published in 2013, which is the result of 11 years investigation by R. Kayen and his colleagues [35], has been used for modeling. The database consists of earthquake parameters namely moment magnitude (M_w) and peak horizontal ground surface acceleration (a_{max}) plus soil parameters namely normalized shear wave velocity (V_{s1}), vertical total stress of the soil at the depth considered (σ_v), vertical effective stress of the soil at the same depth (σ'_v), shear stress reduction factor (r_d), depth of critical layer, depth to water table and the cyclic stress ratio (CSR).

In addition to the measured parameters, it has been determined whether an observation liquefied or not. Out of 415 cases in the database, 287 cases has been liquefied and in the remaining 128 cases liquefaction is not reported.

3.2. Data Preprocessing

Before building any models using the data, some preprocessing procedures have to be considered. There are some important issues that can have significant impacts on a model's performance.

- The number of parameters increases the dimensions of the feature space which can introduce errors in the case of non-informative predictors.
- Highly correlated parameters can result in unstable models without contributing much information with regard to the outcome which can increase model errors.
- The scales of parameters are different and it may significantly reduce the performance of some models.

First issue is not a problem here, since numerous studies have shown that the mentioned parameters are informative with regard to liquefaction potential. However, the second issue can deteriorate a model's performance and it should be addressed. For instance, CSR and r_d are functions of the other variables. Hence they are highly correlated with other parameters.

In order to evaluate the correlations between parameters, the Pearson correlation coefficient can be used and it is given by

$$\frac{1}{n-1}\left[\frac{\sum_a \sum_b (a-\bar{a})(b-\bar{b})}{S_a S_b}\right] \tag{3.1}$$

where n is the number of pairs of data, $\bar{a}$ and $\bar{b}$ are the sample means of all the parameter a and parameter b values, respectively, and S_a and S_b are the corresponding sample standard deviations. Pearson correlation coefficient can be calculated for all pairs of parameters to obtain the correlation matrix.

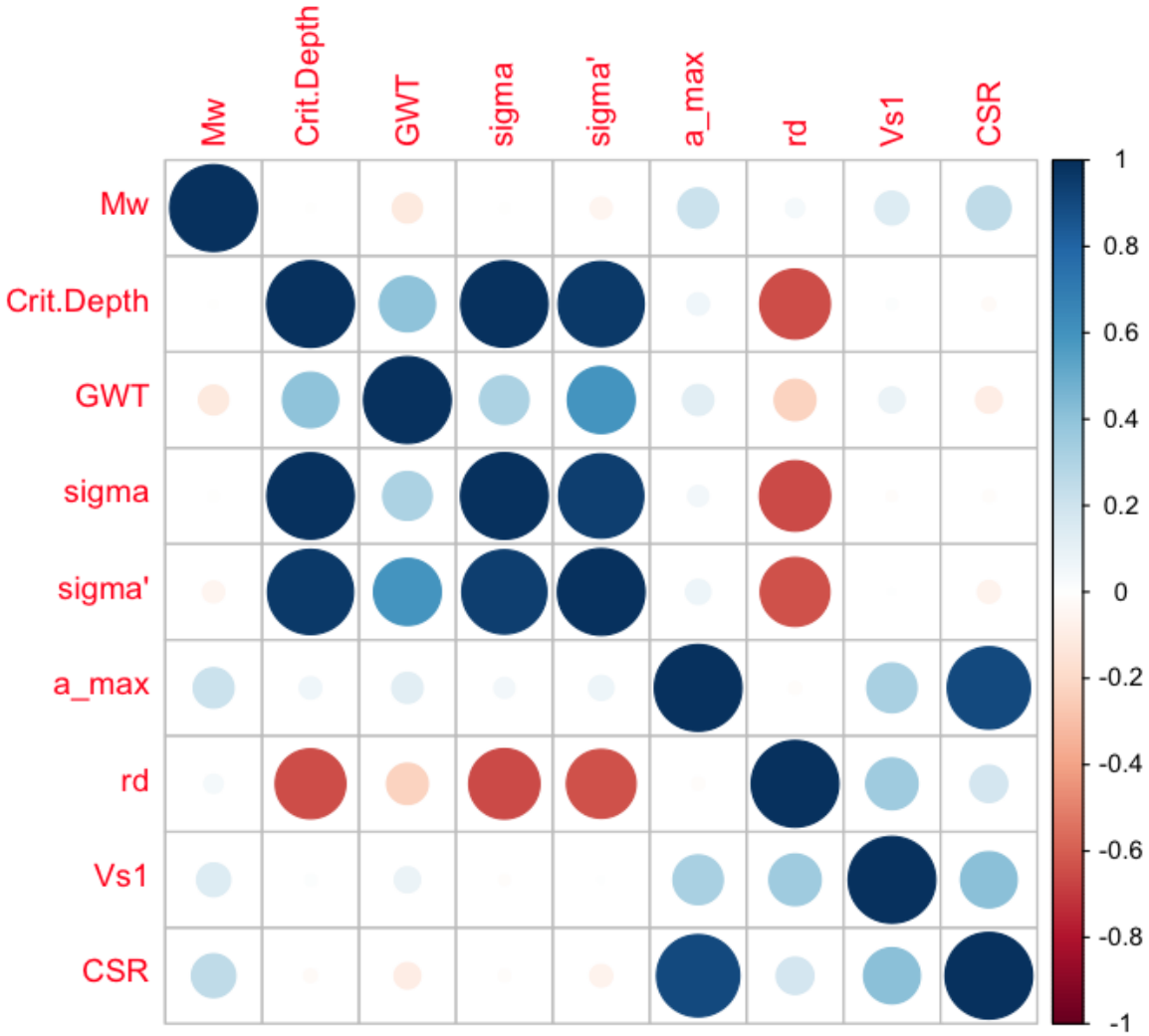

Figure 3.1. The correlation matrix of the liquefaction data.

The correlation matrix for the liquefaction data is provided in the Figure 3.1. This matrix suggests that CSR has high correlations with the other parameters specially over 90% correlation with a_{max} . Also high correlations has been observed between r_d and other parameters such as σ_v, σ'_v and the critical layer's depth. As it is mentioned before, statistical models can suffer from highly correlated parameters and this is why we expect including reduction factor parameter r_d and CSR to be problematic. Vertical effecitve stress σ'_v has high correlations with total vertical stress σ_v, ground water table depth and the critical depth. We expect excluding this parameter will improve the model performance.

Same thing is true for the critical depth since this parameter too, has high correlations with total vertical stress σ_v. The addressed issue, that some parameters have high correlations with each other, is referred to as *collinearity* in case of two parameters involved and *multicollinearity* for relationships between multiple parameters. Our recommended set of parameters, only involves the moment magnitude M_w, depth of water table, total vertical stress σ_v, peak ground acceleration a_{max} and the normalized shear wave velocity V_{s1}.

The third issue, different scales of the parameters, is problematic for the SVM and ANN models. SVM is based on the Euclidean distance, this implies that parameters with bigger scales are going to impact the model much more. In ANN, recall that the inputs are related to the output via multiplications of the inputs and the coefficients, as a result, small scale parameters are going to have approximately zero effects on the outcome with the presence of big scale parameters.

However, this is not an issue when it comes to using QDA and Random forest since the scales of the parameters has no effect on the outcome. Considering these issues, data preprocessing will be considered when building the models.

3.3. Tuning the Models

Now each model is going to be tuned in order to select the best parameters. QDA has no parameter to be considered for tuning. Other models are going to be tuned using the recommended set of parameters, the moment magnitude M_w, depth of water table, total vertical stress σ_v, peak ground acceleration a_{max} and the normalized shear wave velocity V_{s1}.

3.3.1. Tuning the Random Forest Model

For a random forest model, the number of randomly chosen parameter for each split should be determined in the model. For finding the best value for this parameter, one can use cross-validation to see at which number the best performance is obtained.

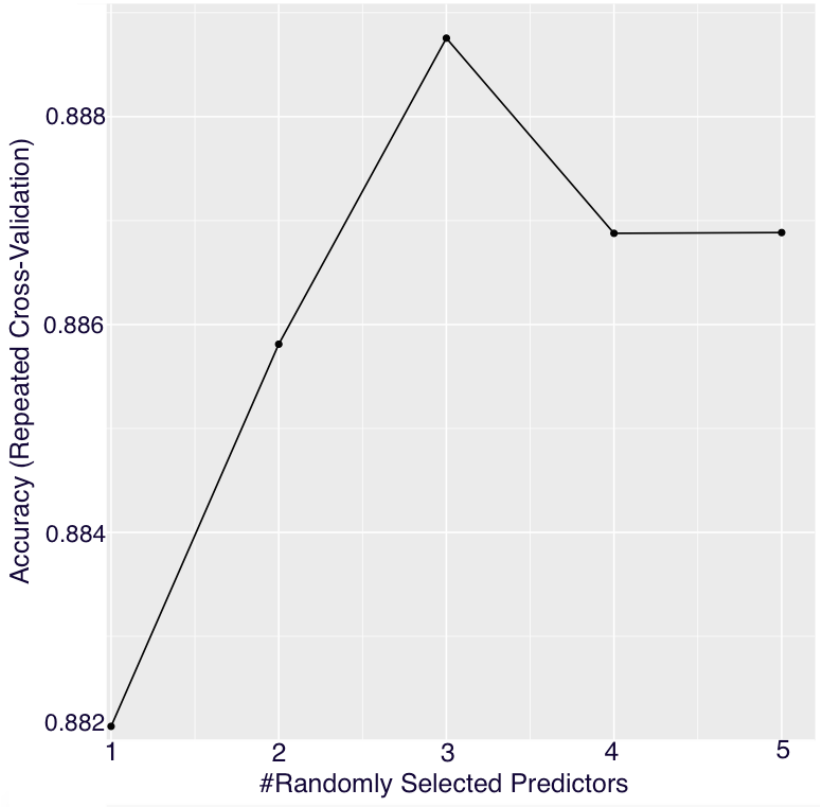

Figure 3.2. Cross-validation result for tuning the random forest model.

The result of this procedure is given in the Figure 3.2. As it is shown, a random forest model which considers only three predictors for performing the splits gives the best performance.

3.3.2. Tuning the SVM Model

The same procedure, cross-validation, has been used in order to find the best parameters, cost value (C) and gamma (γ), for SVM. It is important to note that for tuning the SVM model all of the predictors have been scaled and centered to have mean value of zero and standard deviation of one to mitigate the issue with parameters having different scales. A set of recommended values by literature for C and γ are considered as follows:

$$C = \{1, 4, 9, 16, 25, 36, 49, 64, 81, 100\}$$

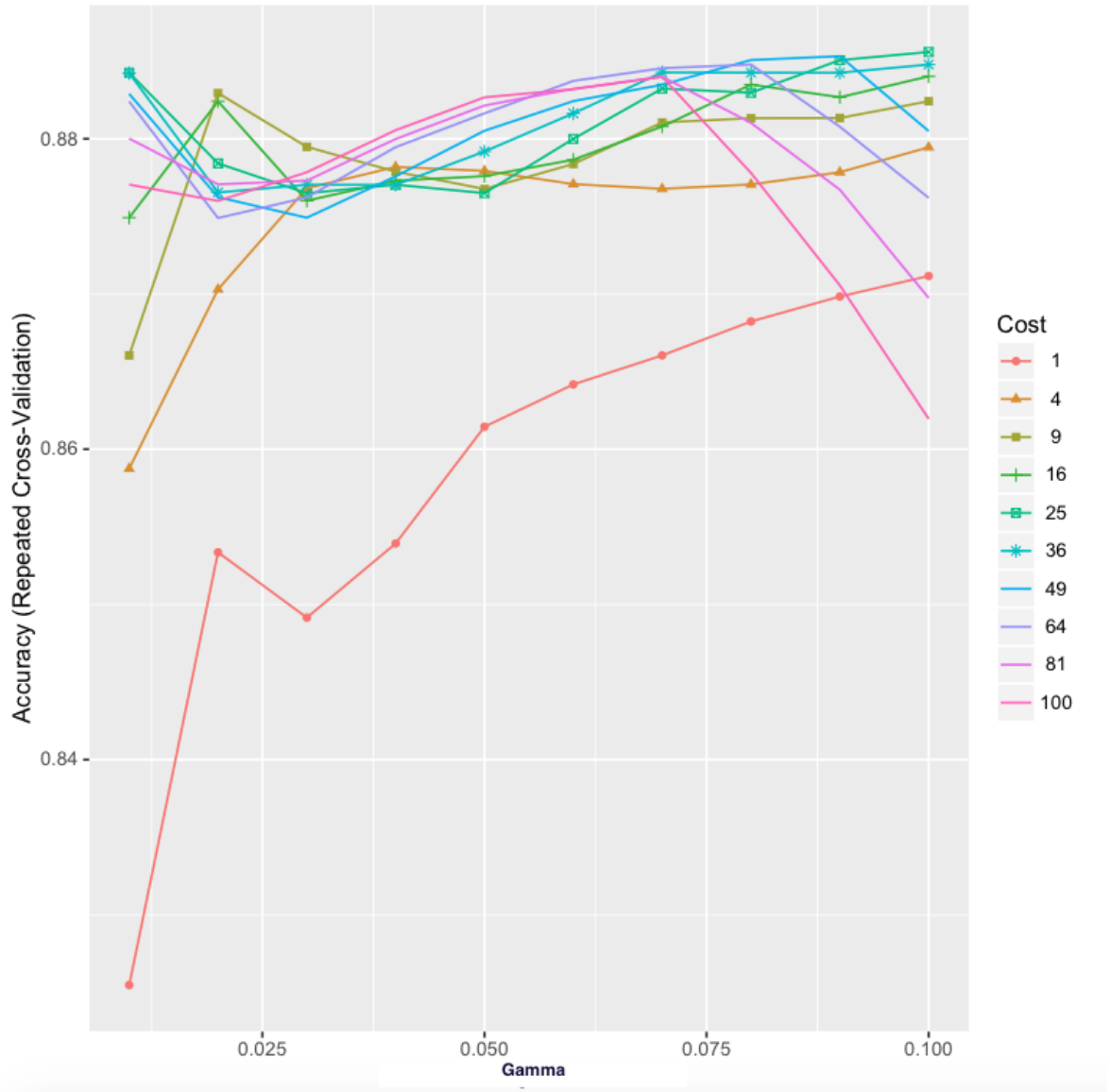

Figure 3.3. Cross-validation results for tuning the SVM model.

and

$$\gamma = \{0.025, 0.05, 0.075, 0.1\}.$$

Figure 3.3 shows that a SVM model with C = 25 and $\gamma = 0.1$ has the best performance among all other combinations of cost value and gamma. As a result this model will be used as the best candidate for predicting liquefaction phenomenon.

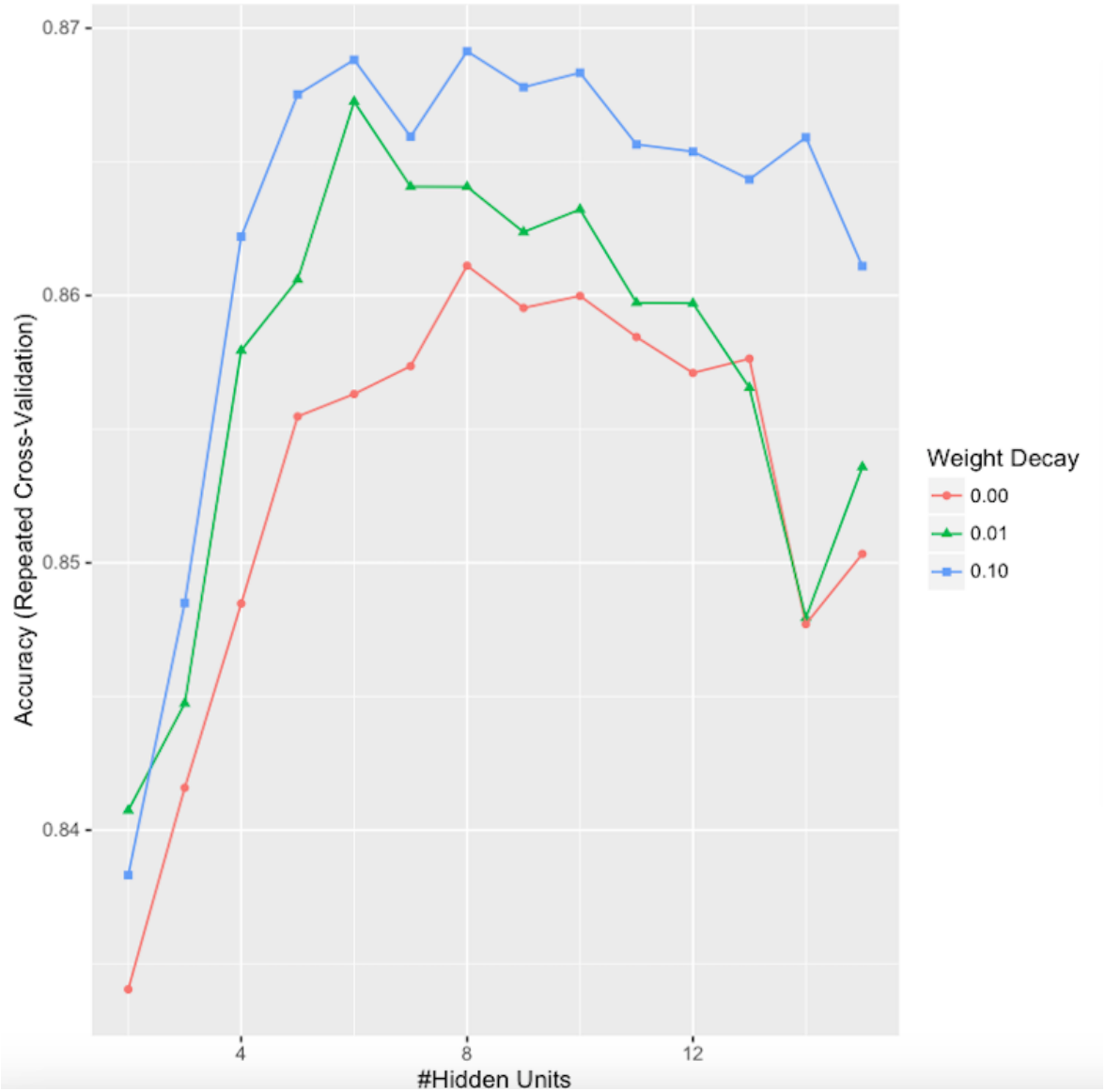

Figure 3.4. Cross-validation results for tuning the ANN model.

3.3.3. Tuning the ANN Model

For using ANN, the number of hidden neurons in the hidden layer plus the weight decay should be determined. These parameters are going to be tuned using cross-validation. The predictors are centered and scaled similar to SVM in order to deal with scale differences. Three different values for the weight decay are considered as (0 , 0.01 , 0.1). Also one to twenty hidden neurons are considered for examination.

Figure 3.4 shows that the ANN model with eight hidden neurons and weight decay of 0.1, has the best performance among all other combinations for these parameters.

3.4. Comparing the Tuned Models for Liquefaction Prediction

In this section, performances of the four models will be compared in order to choose the best candidate. All of the models will be trained using the five predictors namely the moment magnitude M_w, depth of water table, total vertical stress σ_v, peak ground acceleration a_{max} and the normalized shear wave velocity V_{s1}. The minimum, maximum, mean and standard deviation of the variables are given in the Table (3.1). For comparing the model performances, each model is trained and tested 50 times and

Table 3.1. Statistical analysis of data set (n = 415).

Variable	**Minimum**	**Maximum**	**Mean**	**Standard Deviation**
Mw	5.9	9	7.12	0.53
GWT(m)	0.4	7	1.95	1.15
σ_v(Kpa)	17.4	331.68	89.69	45
$a_{max(g)}$	0.02	0.76	0.33	0.16
V_{S1}(m/s^2)	81.7	362.9	166.7	40

the middle accuracy is taken into account. This procedure is being done by repeating 10 fold cross-validation 5 times for each model. Doing this, repeating the procedure fifty times, ensures us that the results are reliable and sampling problem is not an issue. The details of the results can be shown using box-plot, which is a graphical interpretation of statistical data based on the minimum, first quartile, median, third quartile, and maximum. According to the box-plots shown in the Figure 3.5, the QDA model has the worst performance. As it has been described before, this model assumes that all of the variables are dependent and have the Gaussian probability density function. The results suggest that this assumption is not accurate. The other three models did not take a strong assumption about the problem and showed better performances having better accuracy.

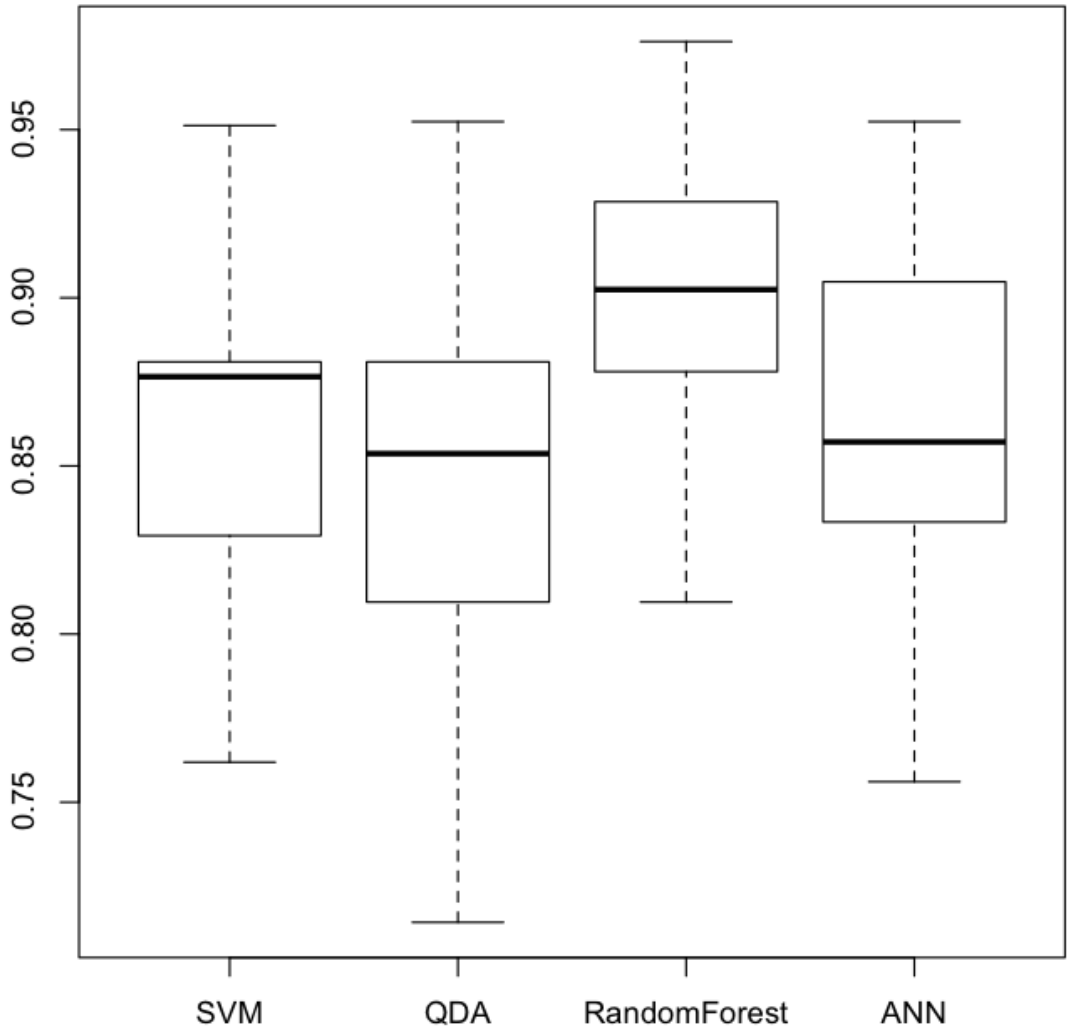

Figure 3.5. Comparing performances of the models based on their prediction accuracy.

Although, ANN needs to estimate a lot of coefficients and it demands more data compared to SVM and random forest but it managed to perform as good as SVM roughly. Finally, random forest has the highest median with the least range and variance among all of the methods and it gave the best results by performing better than other models. Hence, random forest can be considered as the best model for predicting liquefaction.

4. COMPARING THE RANDOM FOREST MODEL WITH THE STRESS-BASED APPROACH FOR LIQUEFACTION PREDICTION

This chapter gives a brief introduction to how engineers have dealt with liquefaction issue so far. The stress-based approach, which is the most common method for evaluating the liquefaction potential in the geotechnical industry, will be covered in this chapter. Finally, a comparison will be made between the stress-based approach and the Random forest model, which stood out as the best model for liquefaction prediction among the other methods described in the previous chapter.

4.1. Dealing with Liquefaction Phenomenon

Liquefaction, discovered in 1960s, is a major problem in geotechnical engineering. The way this phenomenon is dealt with in geotechnical engineering is that whenever an earthquake takes place, investigators go to the site and look for evidences of liquefaction and then they map the area, pointing the locations where liquefaction has occurred. There are several observational methods for the task. One of them is sand cone. When a soil liquefies, typically because it is liquid it cannot bare the soil on top of it so it cracks and from the cracks the liquefied soil flows out and forms a crater, which looks like a sand deposit with a void on top of it [36]. Also, spreading of embankments and dams, tilted buildings and plaster paved roads covered with sands, which did not exist before earthquake, are additional indicators of liquefaction happening in the site.

After mapping the area, spotting liquefaction regions and non liquefied ones, very intense site investigations are conducted which often last for several years. Such investigations consist of measuring soil parameters and making distinction between the places where liquefaction has occurred and places where liquefaction has not occurred.

4.1.1. Factors Affecting the Liquefaction Potential

Although, there are several known and unknown factors contributing to the liquefaction potenital, some of the important ones are listed below [37]:

- Plasticity characteristics: If a soil poses high plasticity typically it is not expected that it will undergo liquefaction.
- Grain size distribution: While well graded sands have less susceptibility to liquefaction, poorly graded sands tend to have more liquefaction potential. Also, coarse granular materials have low liquefaction potential because of their high permeability resulting in pore water pressure to dissipate easier, while fine granular soils are more susceptible to liquefaction. Finer silts with flaky or plate like shape, chemically close to clay, have low liquefaction potential. Silts with bulk shape, chemically close to sand, have high liquefaction potential.

4.2. Methods Used to Assess the Liquefaction Potential

There are different methods that have been used in order to assess the liquefaction potential. Some of them are:

- Chinese criteria.
- The simplified stress-based method.
- Regional liquefaction hazard maps and historical cases of liquefaction.
- Cyclic strain method.
- Energy-Based method.
- Laboratory and physical model testing.
- Site measurement of pore pressure generated due to dynamic loading.
- Computational mechanics methods.

Here only the first two are introduced.

4.2.1. Chinese Criteria, Wang 1979

According to this criteria, if a soil posses the following properties it is susceptible to liquefaction:

- Fraction finer than $0.005mm \leq 15\%$.
- Liquid limit $\leq 35\%$.
- Natural water content ≥ 0.9 Liquid limit.
- Liquidity Index ≥ 0.75.

4.2.2. The Simplified Stress-Based Method

Liquefaction is a phenomenon which mostly happens in the loose sandy soils that are fully saturated. Undisturbed sampling from such loose soils is not applicable for reasonable time and budget. This is why engineers prefer using in-situ tests such as standard penetration test (SPT), cone penetration test (CPT) and shear wave velocity testing to gather information about the underlying soil. As a result, a stress-based method for assessing liquefaction potential based on such tests was originally established by Whitman (1971) and Seed and Idriss (1971) [38]. Basics of this approach, often called the "simplified method", continuously modified by different researchers since 1971, however; the main idea of it did not chang and it still is the most commonly used method for liquefaction potential assessment.

According to this framework, the cyclic stresses induced by the earthquake and the resistance of the soil to such stresses should be estimated in order to judge if a soil is liquefiable or not. Estimating the cyclic horizontal stresses induced by earthquake (CSR) can be done using the Seed-Idris simplified procedure which suggests the Equation (4.1)

$$CSR = 0.65.(\frac{a_{max}}{g}).(\frac{\sigma_v}{\sigma'_v})r_d \tag{4.1}$$

where a_{max} is the horizontal component of the peak ground acceleration at the field, g the gravitational acceleration, r_d is a factor for considering the non-rigid response of the soil column, σ'_v is the initial total vertical stress in the soil and σ_v is the initial effective vertical stress in the soil [39].

For estimating soil cyclic resistance ratio (CRR) against the cyclic stresses, empirical approaches, which use the historical data have been suggested by many researchers, ending up with different equations based on different data and adjustments. however most of them are quite similar and in all of them the CRR is the boundary which separates the liquefied cases from non-liquefied ones. Finally, the safety factor against liquefaction triggering is defined as

$$FS = \frac{CRR}{CSR} \tag{4.2}$$

What makes the Simplified approach straightforward to use is charts that are developed to calculate the safety factor. Since 1971, several charts have been proposed by different researchers. Almost all of the charts are prepared for an earthquake magnitude of 7.5. Also the information used in these charts varies from each other depending on what kind of in-situ test they used. These approaches are explained in more detail.

4.2.2.1. SPT Based Chart for Triggering of Liquefaction. SPT-N number is a common criteria for calculating the liquefaction risk. However, this number should be corrected before using it for further judgments [40]. For correcting the SPT-N number, one should normalize it to a reference effective overburden pressure of 100 Kpa. N_{spt} is multiplied by the factor $(100/\sigma_v)^{0.5}$. This correction has also a limit. According to Euro code, this correction shall not be smaller than 0.5 and not greater than 2. Another correction for N_{spt} is energy correction. The energy transferred to the spoon should be 60% of the theoretical energy. If this condition is not met then there are some corrections that should be applied. Although there are many corrections available for SPT-N number but the two mentioned corrections are the most critical ones.

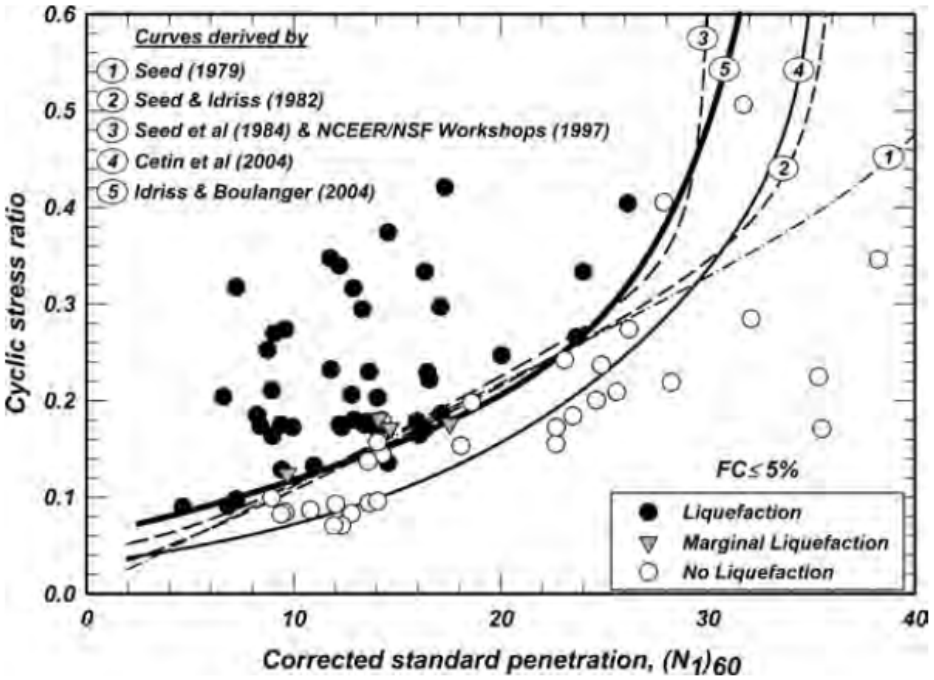

Figure 4.1. Curves relating the CRR to $N_{1,60}$ with $M = 7.5$ and $\sigma'_v = 1$ atmospheric pressure [37].

Finally, the corrected SPT-N number is referred to as $N_{1,60}$, which is used to find the correlation for liquefaction triggering, that is a separating line betwen liquefied and not liquefied observations. Figure 4.1 shows a chart, which has $N_{1,60}$ as its x-axis and CSR as its y-axis, that is being used to assess liquefaction potential using SPT in-situ results [41, 42].

4.2.2.2. CPT Based Chart for Triggering of Liquefaction. Nowadays, by the development of technology, CPT is taking SPT's place as the widely used method for in-situ investigations because of its advantages over SPT. For instance, CPT is cable of detecting thin liquefiable layers as well, while even a 10 cm layer is barely detectable by taking the SPT-N values. Also, CPT has been developed to determine the type of soil as well, so it is possible to know what kind of soil you encounter by using cone resistance q_c and skin friction f_s. It is important to mention that the CPT should be normalized with respect to atmospheric pressure to get a corrected tip resistance value.

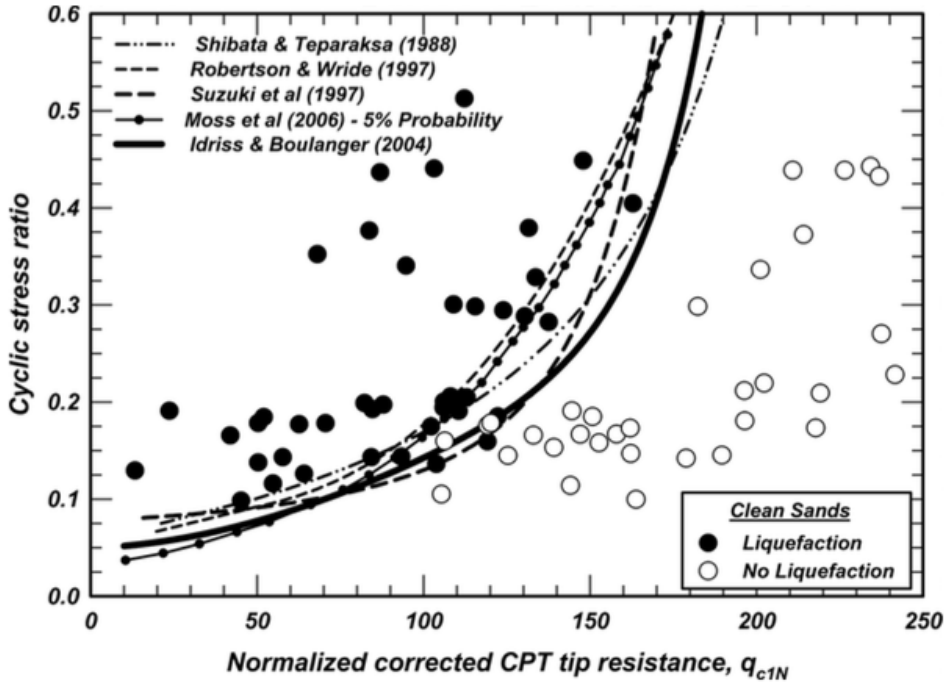

Figure 4.2. Curves relating the CRR to q_{C1N} with $M = 7.5$ and $\sigma'_v = 1$ atmospheric pressure [37].

Just like SPT, the corrected CPT measurements are used in order to develop CPT-based charts for assessing liquefaction potential. Figure 4.2 shows a chart, which has q_{C1N}, normalized corrected CPT cone resistance, as its x-axis and CSR as its y-axis, that is being used to assess liquefaction potential using CPT in-situ results [42, 43].

4.2.2.3. Shear Wave Velocity Based Chart for Liquefaction Triggering. The database, used for developing models for liquefaction potential assessment in this thesis, is based on sheare wave velocity method that is the focus of this section.

One of the earliest models that implemented the shear wave velocity for assessing liquefaction potential was proposed in 1980s. It was developed using SPT-V_s relationships. Since 1990s, the increasing number of direct V_s measurements at liquefaction test fields has led to several correlations of effective normalized shear wave velocity (V_{s1}) and cyclic stress. Figure 4.3 Shows some of these relationships.

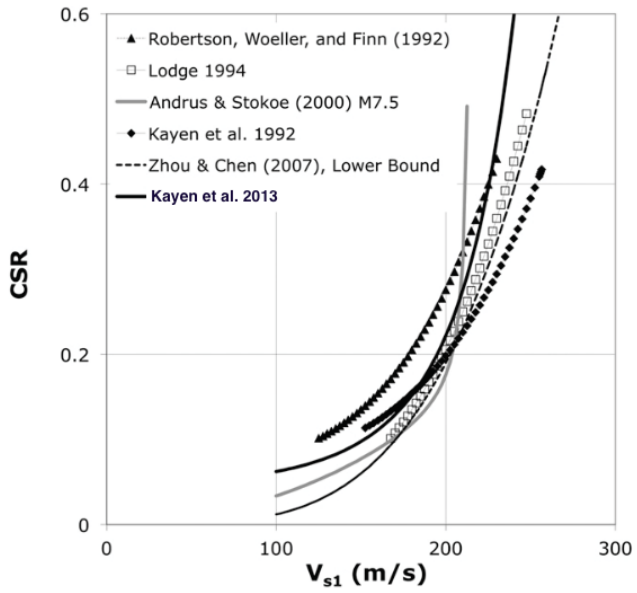

Figure 4.3. Comparison of different relationships, for deterministic shear wave velocity based assessment of liquefaction [35].

Andrus and Stokoe [44] increased the number of V_s data for liquefaction test sites by adding data sets observations from the earthquake historical cases of 1964 Niigata, 1975 Haicheng, 1979 Imperial Valley, 1983 Borah Peak, Idaho, California, 1981 Westmoreland, China, Japan earthquakes, as well as a lot of non-liquefaction case histories recorded at the Lotung LSST Facility in Taiwan [35]. Also many new liquefied sites during the 1999 kocaeli, Turkey earthquake have been added to the database. Reliability based methods (Juang *et al.* 2001; Juang *et al.* 2002), which are based on the Andrus and Stokoe (2000) data set, were developed to determine the V_{s1} magnitude dependent CSR relationship through a probabilistic framework.

Recently, the shear wave velocity based approach has developed considerably and has improved correlations and better databases compared to those of past. Some of them are summarized by Andrus and Stokoe (2000), and Andrus *et al.* (2003). Figure 4.4 depicts shear wave velocity data and the triggering liquefaction boundary after Andrus and Stokoe (2000).

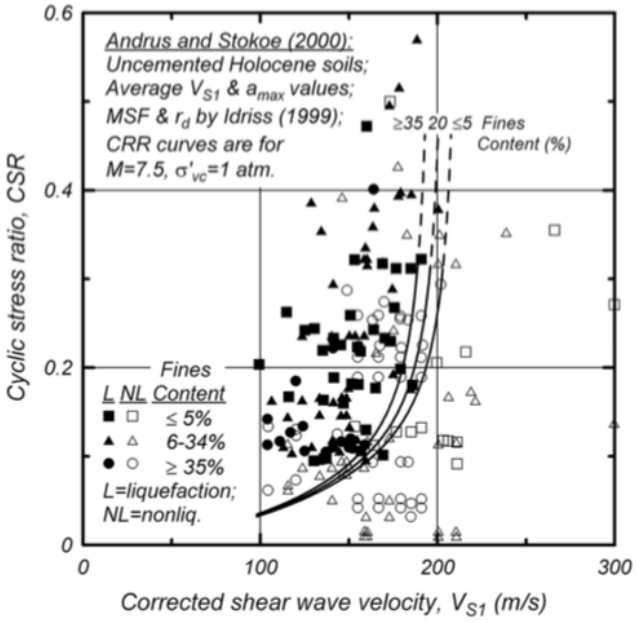

Figure 4.4. V_{s1} based liquefaction correlation for clean sands [35].

One of the main advantages of shear wave velocity (V_s) measurment over SPT and CPT values is that it is useful for sites, in which it is difficult to penetrate the soil or take samples from it (e.g., cobbles and gravels) [35]. However, SPT and CPT methods have higher correlations with relative density (D_R), which is an important factor affecting the cyclic behavior of saturated soil (Idriss and Boulanger 2008), V_s is significantly less sensitive to soil compactness problems and reduced penetration resistance in the presence of soil fines, in comparison to SPT and CPT methods. Hence, V_s does not need major corrections for fines content (FC) [45]. Typically, the measurement of V_s in the field, is corrected to a normalized V_s1 using the Equation (4.3), where P_a is the normalized reference stress.

$$V_{s1} = V_s(P_a/\sigma'_v)^{0.25} \tag{4.3}$$

The standard way of measuring V_s of soil consists of penetrometer or instrumented borehole for estimating the shear waves travel time at different depths.

Multiple borehole examinations are used to estimate the vertical and horizontal shear wave velocity characteristics of the soil. Invasive V_s testing, which includes drilling cased boreholes and large penetration equipments, is costly, however; recently, brand new developed non-invasive methods, which indirectly measures the V_s of soil through an inversion of the surface wave dispersion features of the ground (Kayen *et al.* 2002; Andrus *et al.* 1998; Stokoe *et al.* 1994), are available for lower costs.

4.3. Comparing the Accuracy of Stress-Based Approach with that of Random Forest model

Now we are going to use stress-based approach and the random forest model to predict if a soil is going to liquefy or not and then we will compare their accuracy with each other. The procedure for liquefaction prediction by the stress-based approach can be done by calculating the CSR and the CRR of the soil and compare these values. If CSR $>$ CRR then the subject soil falls in the liquefiable soils and if CSR $<$ CRR then the soil is in the non-liquefiable zone.

First, the database, introduced in the Chapter three, has been ordered based on the V_{s1} of the observations. Consequently, ordered sampling has been implemented to have a sample of 10% of the database. The remaining 90% has been used to build the random forest model and the held out 10% data is used for testing the accuracy of the model. For the exact same 10% testing data, CSR and CRR has been calculated according to the stress-based approach and predictions are carried out to see which observation will liquefy and which will not.

Then the accuracy of both models has been calculated by comparing the predictions to the real status of the observations. The accuracy of the models can be obtained by taking the ratio of correct predictions to the total number of predictions. The results are given in Table 4.1 and Table 4.2.

Table 4.1. Confusion matrix for stress-based approach.

	not liquefied in reality	**liquefied in reality**
Predicted: to not liquefy	12	5
Predicted: to liquefy	4	21

Table 4.2. Confusion matrix for random forest model.

	not liquefied in reality	**liquefied in reality**
Predicted: to not liquefy	13	0
Predicted: to liquefy	3	26

- Accuracy of the stress-based approach:

$$100 \times (12 + 21)/42 = 78.5$$

- Accuracy of the random forest model:

$$100 \times (13 + 26)/42 = 92.8$$

As can be seen in the Table 4.3, the random forest model managed to achieve 14.3% more accuracy compared to the stress-based approach for liquefaction prediction. The ordered sampling ensures that the results are reliable [46] since the models are implemented to predict liquefaction for a wide range of soils.

Table 4.3. The correct prediction ratio of the models.

	Stress-based	**Random forest**
Accuracy %	78.5	92.8

5. CONCLUSION

This thesis outlines machine learning approaches and their applications for assessing liquefaction potential in soils. Nowadays, technology is developing rapidly that resulted in faster computation speed and made gathering data cheaper than ever before. This suggests that data analyzing skills are crucial in future for engineers and it can help them to solve wide range of problems efficiently and faster than before.

Machine learning recently has gained major reseach interest in mutiple fields and it is shown to be an efficient tool for modeling and solving engineering problems. Among numerous machine learning algorithms, four of them has been chosen to be used in this thesis namely Quadratic discriminant analysis (QDA), Random forest, Artificial neural network (ANN) and Support vector machine (SVM). All of the models have shown acceptable performance and accuracy with regard to liquefaction prediction. There are many commercial softwares that can apply such models which means engineers without decent programming knowledge can also use them.

All of the models are introduced and trained, using R programming language, on the liquefaction data in order to classify soils as liquefiable or non-liquefiable. Random forest has been shown to be the most reliable and accurate model compared to other three machine learning methods.

All in one, as the main contirbution in thesis, it has been shown that the random forest model is able to predict if a soil is going to liquefy or not with 92.8 accuracy using five parameters: moment magnitude (Mw), depth to water table, peak ground accelaration (a_{max}), total vertical stress (σ_v) and the normalized shear wave velocity (V_{s1}) while the traditional stress-based approach which is also the most common approach for liquefaction evaluation gave 78.5% accuracy using more paramaters. One can be more certain about the liquefiablity of a soil using the random forest model and perform safer and more economical design by having a better risk analysis tool such as random forest.

Deep learning method, which is a sophisticated artificial neural network, is gaining a lot of attentions these days. It is revolutionizing many industries with the emergence of big data. For future researches Deep learning can be considered as a promising method to obtain more accurate predictions on liquefaction risks. However, such an approach needs a lot of data that is anticipated to be available because of future investigations on liquefiable soils.

REFERENCES

1. Goh, A. T., "Seismic Liquefaction Potential Assessed by Neural Networks", *Journal of Geotechnical engineering*, Vol. 120, No. 9, pp. 1467–1480, 1994.

2. Goh, A., "Nonlinear Modelling in Geotechnical Engineering Using Neural Networks", *Transactions of the Institution of Engineers, Australia. Civil engineering*, Vol. 36, No. 4, pp. 293–297, 1994.

3. Juang, C. H. and C. J. Chen, "CPT Based Liquefaction Evaluation Using Artificial Neural Networks", *Computer-Aided Civil and Infrastructure Engineering*, Vol. 14, No. 3, pp. 221–229, 1999.

4. Ali, H. and Y. Najjar, "Neuronet Based Approach for Assessing Liquefaction Potential of Soils", *Transportation Research Record: Journal of the Transportation Research Board*, Vol. 1633, No. 11633, pp. 3–8, 1998.

5. Pal, M., "Support Vector Machines Based Modelling of Seismic Liquefaction Potential", *International Journal for Numerical and Analytical Methods in Geomechanics*, Vol. 30, No. 10, pp. 983–996, 2006.

6. Fisher, R. A., "The Use of Multiple Measurements in Taxonomic Problems", *Annals of eugenics*, Vol. 7, No. 2, pp. 179–188, 1936.

7. McCullagh, P., "Generalized linear models", *European Journal of Operational Research*, Vol. 16, No. 3, pp. 285–292, 1984.

8. Breiman, L., J. Friedman, C. J. Stone and R. A. Olshen, *Classification and Regression Trees*, CRC press, 1984.

9. Lantz, B., *Machine Learning with R*, Packt Publishing, Birmingham, UK., 2013.

10. James, G., D. Witten, T. Hastie and R. Tibshirani, *An Introduction to Statistical Learning: with Applications in R*, Springer Publishing Company, Incorporated, New York, NY, USA, 2014.

11. Hastie, T., R. Tibshirani and J. Friedman, *The Elements of Statistical Learning*, Springer Series in Statistics, Springer New York Inc., New York, NY, USA, 2001.

12. Kohavi, R. *et al.*, "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection", *Ijcai*, Vol. 14, pp. 1137–1145, Stanford, CA, 1995.

13. Stone, M., "Cross-Validatory Choice and Assessment of Statistical Predictions", *Journal of the royal statistical society. Series B (Methodological)*, pp. 111–147, 1974.

14. Zhang, P., "Model Selection via Multifold Cross Validation", *The Annals of Statistics*, pp. 299–313, 1993.

15. Kuhn, M. and K. Johnson, *Applied Predictive Modeling*, Springer, New York, 2013.

16. Efron, B., "Estimating the Error Rate of a Prediction Rule: Improvement on Cross-Validation", *Journal of the American Statistical Association*, Vol. 78, No. 382, pp. 316–331, 1983.

17. Efron, B. and R. Tibshirani, "Improvements on Cross-Validation: The 632+ Bootstrap Method", *Journal of the American Statistical Association*, Vol. 92, No. 438, pp. 548–560, 1997.

18. Alpaydin, E., *Introduction to Machine Learning*, The MIT Press, London, England, 2010.

19. Vapnik, V., *The Nature of Statistical Learning Theory*, Springer science & business media, 2013.

20. Mika, S., G. Ratsch, J. Weston, B. Scholkopf and K.-R. Mullers, "Fisher Discrim-

inant Analysis with Kernels", *Neural Networks for Signal Processing IX, 1999. Proceedings of the 1999 IEEE Signal Processing Society Workshop.*, pp. 41–48, IEEE, 1999.

21. Rokach, L. and O. Maimon, *Data Mining with Decision Trees: Theroy and Applications*, World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2008.

22. Liaw, A. and M. Wiener, "Classification and Regression by Random Forest", *R news*, Vol. 2, No. 3, pp. 18–22, 2002.

23. Cortes, C. and V. Vapnik, "Support Vector Networks", *Machine learning*, Vol. 20, No. 3, pp. 273–297, 1995.

24. Scholkopf, B., K.-K. Sung, C. J. Burges, F. Girosi, P. Niyogi, T. Poggio and V. Vapnik, "Comparing Support Vector Machines with Gaussian Kernels to Radial Basis Function Classifiers", *IEEE transactions on Signal Processing*, Vol. 45, No. 11, pp. 2758–2765, 1997.

25. Bishop, C. M., *Neural Networks for Pattern Recognition*, Oxford university press, 1995.

26. Friedman, J. H., "An Overview of Predictive Learning and Function Approximation", *From Statistics to Neural Networks*, pp. 1–61, Springer, 1994.

27. Rashid, T., *Make Your Own Neural Network*, CreateSpace Independent Publishing Platform, 2016.

28. Ripley, B. D. and N. L. Hjort, *Pattern Recognition and Neural Networks*, Cambridge University Press, New York, NY, USA, 1995.

29. Rumelhart, D. E., G. E. Hinton and R. J. Williams, *Learning Internal Representations by Error Propagation*, Tech. rep., DTIC Document, 1985.

30. MacKay, D. J., "A Practical Bayesian Framework for Backpropagation Networks",

Neural computation, Vol. 4, No. 3, pp. 448–472, 1992.

31. Rumelhart, D. E., G. E. Hinton and R. J. Williams, "Learning Representations by Back-propagating Errors", *Cognitive modeling*, Vol. 5, No. 3, p. 1, 1988.

32. Girosi, F., M. Jones and T. Poggio, "Regularization Theory and Neural Networks Architectures", *Neural computation*, Vol. 7, No. 2, pp. 219–269, 1995.

33. Kramer, S. L., *Geotechnical Earthquake Engineering*, Prentice Hall, Upper Saddle River, New Jersey, USA, Jan. 1996.

34. Srbulov, M., *Geotechnical Earthquake Engineering Simplified Analyses with Case Studies and Examples*, Springer, United Kingdom, 2008.

35. Kayen, R., R. Moss, E. Thompson, R. Seed, K. Cetin, A. D. Kiureghian, Y. Tanaka and K. Tokimatsu, "Shear Wave Velocity Based Probabilistic and Deterministic Assessment of Seismic Soil Liquefaction Potential", *Journal of Geotechnical and Geoenvironmental Engineering*, Vol. 139, No. 3, pp. 407–419, 2013.

36. E. Kavazanjian, N. M. T. H.-H., Jr. and P. Sabatini, *Geotechnical Engineering Circular No. 3 - Earthquake Engineering for Highways, Design Examples, Volume 2*, 1997.

37. Idriss, I., R. Boulanger and E. E. R. Institute, *Soil Liquefaction During Earthquakes*, Engineering monographs on miscellaneous earthquake engineering topics, Earthquake Engineering Research Institute, 2008.

38. Seed, H. B. and I. M. Idriss, "Simplified Procedure for Evaluating Soil Liquefaction Potential", *Journal of Soil Mechanics Foundations Division*, 1971.

39. Seed, B. and K. L. Lee, "Liquefaction of Saturated Sands During Cyclic Loading", *Journal of Soil Mechanics & Foundations Div*, Vol. 92, No. ASCE# 4972 Proceeding, 1966.

40. E. Kavazanjian, C. J. H., Jr., *Geotechnical Engineering Circular No. 3 - LRFD Seismic Analysis and Design of Transportation Geotechnical Features and Structural Foundations Reference Manual*, 2011.

41. Idriss, I. and R. Boulanger, "SPT-based Liquefaction Triggering Procedures", *Rep. UCD/CGM-10*, Vol. 2, 2010.

42. Boulanger, R. and I. Idriss, *CPT and SPT Based Liquefaction Triggering Procedures Report No*, Tech. rep., UCD/CGM-14/01 Center for Geotechnical Modeling (Davis USA: University of California, Davis) 134 Department of Civil and Environmental Engineering Report, 2014.

43. Boulanger, R. W. and I. Idriss, "CPT-based Liquefaction Triggering Procedure", *Journal of Geotechnical and Geoenvironmental Engineering*, Vol. 142, No. 2, pp. 401–506, 2015.

44. Andrus, R. D. and K. H. Stokoe II, "Liquefaction Resistance of Soils from Shear Wave velocity", *Journal of geotechnical and geoenvironmental engineering*, Vol. 126, No. 11, pp. 1015–1025, 2000.

45. Robertson, P., "Comparing CPT and Vs Liquefaction Triggering Methods", *Journal of Geotechnical and Geoenvironmental Engineering*, Vol. 141, No. 9, p. 04015037, 2015.

46. Ozturan, M., B. Kutlu and T. Ozturan, "Comparison of Concrete Strength Prediction Techniques with Artificial Neural Network Approach", *Building Research Journal*, Vol. 56, No. 1, pp. 23–36, 2008.