CHAPTER 2
THEORY BACKGROUND

This chapter offers a brief review of the main technologies that have made possible the realization of this project. Throughout all this master thesis, Python was the programming language used for the implementation. In order to carry out most of the experiments, Jupyter notebooks have been used. Jupyter provides a rich framework for interactive computing, allowing the user to execute short code cells and outputting the results in the same window. The mentioned experiments need the previous tools to be enriched with some Python libraries which can be divided into data managing libraries, natural language processing libraries, and machine learning libraries. Following sections describe the fundamentals and usefulness of the different libraries that supported the development in this project. Finally, the chapter ends with a theoretical explanation about the fundamental concepts concerned with Machine Learning and Natural Language Processing fields.

2.1 Data Managing

This section encompasses libraries which ease data manipulation in Machine Learning related tasks.

2.1.1 Numpy

Numpy [9,13] is the fundamental package for scientific computing with Python. Most of the remaining libraries use NumPy in their implementation. In relation to the usefulness in this project, Numpy was used due to its powerful high performance over multidimensional arrays and matrices, and the large collection of high-level mathematical functions to operate on these arrays. In addition, Numpy also contains tools for integrating C/C++ and Fortran code, along with broadcasting functions that transparently adapt dimensions between arrays.

2.1.2 Pandas

Pandas [10,14] is an open source, BSD-licensed, Python data analysis library that provides fast, flexible, and expressive data structures. It is built on top of Numpy and is intended to integrate well within a scientific computing environment with many other 3rd party libraries. The two primary data structures of pandas are Series (1-dimensional) and DataFrames (2- dimensional).

- Series is a one-dimensional labeled object, capable of holding any data type (integers, strings, floating-point numbers, Python objects, etc.). It is similar to an array, a list, a dictionary or a column in a table. Every value in a Series object has an index.
- DataFrames is a two-dimensional labeled object with columns of potentially different types. It is similar to a database table, or a spreadsheet. It can be seen as a dictionary of Series that share the same index. During the implementation, Pandas has been useful to structure datasets into unified arrangements as well as to apply operations on whole datasets. Here are some of the functions that Pandas implements:
    - DataFrame object for data manipulation with integrated indexing.
    - Reading and writing data in different formats: CSV and text files, Microsoft Excel, SQL databases and the fast HDF5 format.
    - Intelligent data alignment and integrated handling of missing data
    - Intelligent label-based slicing, fancy indexing, and subsetting of large datasets.
    - Columns can be inserted and deleted from data structures for size mutability
    - Aggregating or transforming data with a powerful group by engine allowing splitapply-combine operations on datasets.
    - High performance merging and joining of datasets.

2.2 Natural Language Processing

The main information source employed in this project are text messages displayed in social networks. Consequently, NLP techniques acquire a role of utmost importance for the fulfillment of this thesis. Among the NLP tasks desired, text preprocessing, and feature extraction methods are the most valuable capabilities offered by the libraries described below.

2.2.1 NLTK

The Natural Language Toolkit (NLTK) [6,8,15] is the fundamental platform for building Python programs to work with human language data. It provides a suite of text processing tools for classification, tokenization, stemming, tagging, parsing, and semantic reasoning. Here are some library highlights.

- It offers easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet [2].
- Lexical analysis: word and text tokenizer.
- n-gram and collocations.
- Part-of-Speech tagger.
- Tree model and text chunker.
- Named-Entity Recognition.

2.2.2 TextBlob

TextBlob [11,16] is a Python (2 and 3) library for processing textual data. It is based on NLTK and Pattern [17] and plays nicely with both. It provides a simple API for dividing into common NLP tasks such as Part-of-Speech tagging, noun phrase extraction, sentiment analysis, classification, translation (powered by Google Translate), and more.

- Splitting text into words and sentences
- Word and phrase frequencies
- Parsing
- n-grams
- Word inflection (pluralization and singularization) and lemmatization
- Spelling correction
- WordNet integration

In the case of this project, the sentiment property from the toolbox was plenty exploited. Such property provides text polarity and subjectivity which can be directly included as features in our data processing pipeline. Initially was also thought to take into account the translation option but currently, it is failing due to deprecation issues.

2.3 Twitter API

The Twitter API [18] gives developers access to most of Twitter's functionality. You can use the API to read and write information related to Twitter entities such as tweets, users, and trends. Technically, the API exposes dozens of HTTP endpoints related to:

- Tweets
- Retweets
- Likes
- Direct messages
- Favorites
- Trends
- Media

Tweepy, as we'll see later, provides a way to invoke those HTTP endpoints without dealing with low-level details. The Twitter API uses OAuth, a widely used open authorization protocol, to authenticate all the requests. Before making any call to the Twitter API, you need to create and configure your authentication credentials. Later in this article, you'll find detailed instructions for this.

You can leverage the Twitter API to build different kinds of automations, such as bots, analytics, and other tools. Keep in mind that Twitter imposes certain restrictions and policies about what you can and cannot build using its API. This is done to guarantee users a good experience. The development of tools to spam, mislead users, and so on is forbidden.

The Twitter API also imposes **rate limits** about how frequently you're allowed to invoke API methods. If you exceed these limits, you'll have to wait between 5 and 15 minutes to be able to use the API again. You must consider this while designing and implementing bots to avoid unnecessary waits.

2.4 Tweepy

Tweepy [19] is an open source Python package that gives you a very convenient way to access the Twitter API with Python. Tweepy includes a set of classes and methods that represent Twitter's models and API endpoints, and it transparently handles various implementation details, such as:

- Data encoding and decoding

- HTTP requests
- Results pagination
- OAuth authentication
- Rate limits
- Streams

If you weren't using Tweepy, then you would have to deal with low-level details having to do with HTTP requests, data serialization, authentication, and rate limits. This could be time consuming and prone to error. Instead, thanks to Tweepy, you can focus on the functionality you want to build.

Almost all the functionality provided by Twitter API can be used through Tweepy. The only current limitation, as of version 3.7.0, is that Direct Messages don't work properly due to some recent changes in the Twitter API.

2.5 Machine Learning

This section gives an overview of the libraries used for Machine Learning purposes. Such libraries are in charge of performing Machine Learning tasks, including classification taking matrix data as input which, in this case, has been obtained using the previously described technologies.

2.5.1 Scikit-learn

Scikit-learn [5,20] is an open source, BSD-licensed, Python library providing simple and efficient tools for data mining and data analysis. It is built on NumPy, SciPy, and matplotlib. Scikit-learn implements a range of machine learning, preprocessing, cross-validation and visualization algorithms.

Scikit-learn can perform classification (identifying to which category an object belongs to), regression (predicting a continuous-valued attribute associated with an object), clustering (automatic grouping of similar objects into sets), dimensionality reduction (reducing the number of variables to consider), model selection (evaluating, validating, and selecting best hyper-parameters and models), and preprocessing (feature extraction and normalization). In this final work, we focus on classification and regression.

Classification: the aim is to assign each input vector to one of a finite number of discrete categories. Another way to think of classification is as a discrete (as opposed to continuous) form of supervised learning where one has a limited number of categories and for each of the n samples provided one is tried to label them with the correct category or class.

Regression: if the desired output consists of one or more continuous variables, then the task is called regression. So, the main difference between classification and regression is the format of the labels, discrete or continuous respectively.

Depending if we perform classification or regression, we refer to the classes as labels or values respectively. In classification or regression Scikit-learn follows the schema showed in the Figure 2.1.

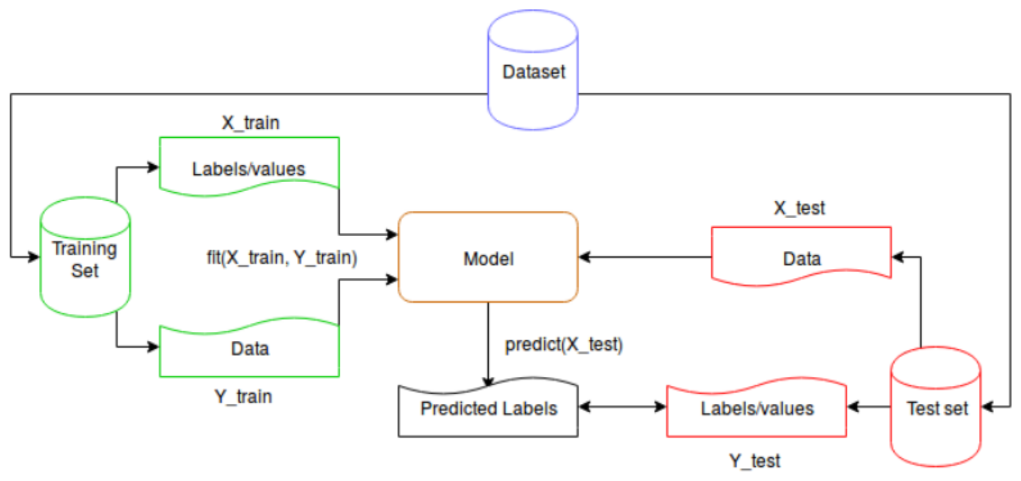


Figure 2.1 Data Splitting and Output Prediction

The first step is to split the original dataset into training and test sets, then using the training set we can create a model to predict new values or labels. We predict new values (or labels) using the model previously created and the test data set. Finally, we can compare the predicted labels and the expected labels to measure the classifier accuracy.

Scikit also provides some packages for extracting vectorizing features from texts. Special interest comes from the TF-IDF (Term Frequency – Inverse Document Frequency) vectorizer, in order to know the words importance used in a set of documents. It works converting the textual representation of information into a Vector Space Model. This is an algebraic model that represents textual information as vectors, being their components the importance of a term. The aim is modeling documents into a vector space ignoring ordering of words but focusing on information about occurrences of words.

The first step is creating a model of the documents into a vector space creating a dictionary of terms that appear on them. The TF-IDF value increases proportionally to the number of times a word appears in the document, but this value is contrasted by the frequency of the word in the corpus, which helps us to adjust according to the fact that some words appear more frequently in general. We cannot forget that some existing words do not contribute any information, these are called stop words and are formed mainly by prepositions, pronouns and articles. These words need to be preprocessed so that our learning algorithm does not consider them.

It is also necessary to preprocess documents in another way. It consists in extraction the root or lemma of each word, this is because there is more information on a specific concept than on its variations, what could mean introducing some more noise.

The next step is calculation the TF or term frequency to represent each term into the vector space. The term frequency is the number of appearances of a specific term in the vocabulary. So, for each term of the vocabulary appearing on all the documents, each document will have a matrix of their appearances on it. This way, each document of the set is represented by a vector with zeros on the terms that did not appear and the number of appearances on the terms that did appear on it.

Once we have these vectors, it is important normalizing them because the importance of a word appearing a number of times depends on the text length. Now we are in a good position to calculate the inverse document frequency. IDF calculate the impact of a word in the corpus and in this case its smooths it by computing it into a logarithmic scale.

When we have both calculated term frequency and inverse term frequency, we multiply both values and we obtain the TF-IDF value. So finally, a high TF-IDF is reached with a word with a high frequency in the document and with a low frequency in terms of the whole set. As a result, for each doc, we have a vector composed of the correspondence of words and its TF-IDF value. Applying these ideas, we can start to extract text features from document.

This final work is focused on classification tasks, where the aim is to assign each input vector to one of a finite number of discrete categories. Another way to think of classification is as a discrete (as opposed to continuous) form of supervised learning where there is a limited number of categories and for each of the samples provided, the algorithm has to label it into the correct category or class.

Finally, Scikit also perform some methods of measures and metrics to show the learning quality.

2.5.2 TF-IDF

A better practice when working with statistical measures is to take into account the frequency a term appears in a document and also the frequency this term is found in other documents. That is the basis of TF-IDF [1] representations whose main goal is to obtain the word importance used in a set of documents. It works in a similar way than BOW model, converting a text input as a vector representation, but in this case being its components the importance of each term. The TF-IDF value increases proportionally to the number of times a word appears in the document, but this value is contrasted by the frequency of the word in the corpus, which helps us to adjust the importance of a word according to the fact that some words appear more frequently in general.

The first step, again, is creating a vocabulary from the corpus creating a dictionary of terms that appear on it. Following, the term frequency is computed to represent each text in the vector space (as done in BOW model). In this way, each document within the corpus is represented by a vector with zeros on the terms that did not appear and the number of occurrences on the terms that did appear on it. Once we have the vectors, it is important normalizing them because the importance of a word appearing a number of times depends on the text length. Now we are in a good position to calculate the inverse document frequency.

The inverse document is defined as: log $|D|$ $1+|d\colon t \in d|$ being $1 + |d\colon t \in d|$ the number of documents where the term t appears and $|D|$ the number of documents in the corpus. There are many versions of this function since it tries to get the impact of a term inside a corpus and in this case, it smooths it by computing it into a logarithmic scale. When we have both calculated term frequency and inverse document frequency, we multiply both values, and we obtain the TF-IDF value. So finally, a high TF-IDF is reached with a word with a high frequency in a document and with a low frequency in terms of the whole set.

2.6 Support Vector Machine

“Support Vector Machine” (SVM) [4,21] is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems.

A Support Vector Machine (SVM) performs classification by finding the hyperplane that maximizes the margin between the two classes. The vectors (cases) that define the hyperplane are the support vectors. Support Vectors are simply the co-ordinates of individual observation. Support Vector Machine is a frontier which best segregates the two classes (hyper-plane/ line). The goal of a support vector machine is to find the optimal separating hyperplane which maximizes the margin of the training data.

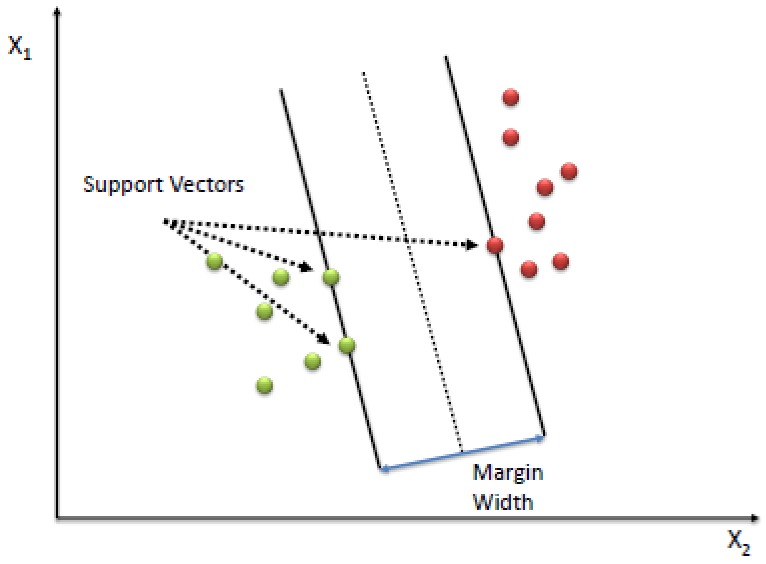
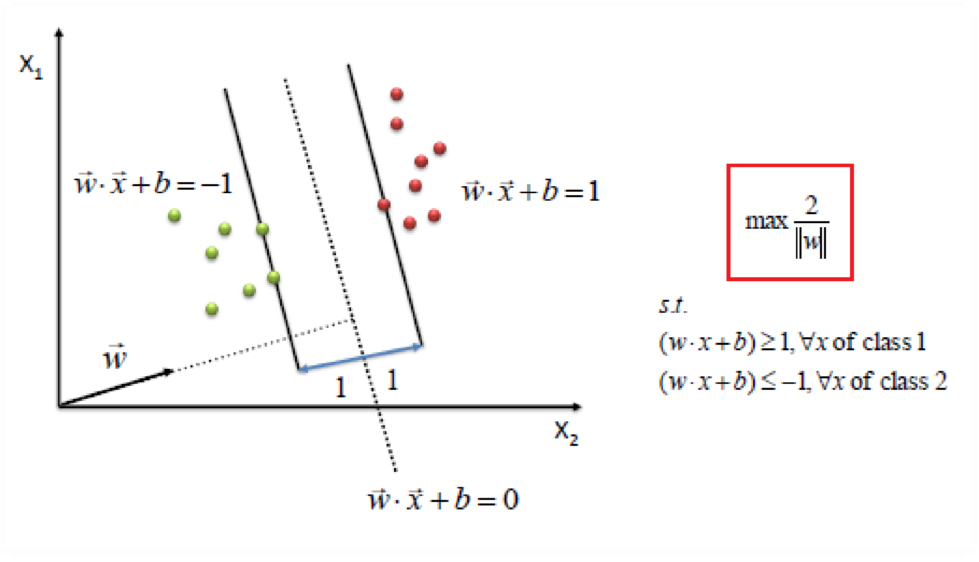


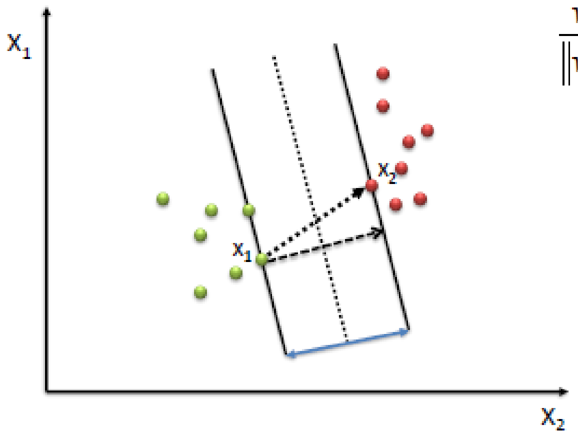
Figure 2.2 Support Vectors

Algorithm

1. Define an optimal hyperplane: maximize margin
2. Extend the above definition for non-linearly separable problems: have a penalty term for misclassifications.
3. Map data to high dimensional space where it is easier to classify with linear decision surfaces: reformulate problem so that data is mapped implicitly to this space.

To define an optimal hyperplane, the model need to maximize the width of the margin (w).


X_1
$\vec{w}\cdot\vec{x}+b=-1$
$\vec{w}\cdot\vec{x}+b=1$
$\vec{w}$
1
1
X_2
$\vec{w}\cdot\vec{x}+b=0$
$\max \frac{2}{\|w\|}$
s.t.
$(w\cdot x+b)\geq 1, \forall x$ of class 1
$(w\cdot x+b)\leq -1, \forall x$ of class 2



$$\frac{w}{\|w\|}\cdot(x_2-x_1)=\text{width}=\frac{2}{\|w\|}$$

$$w\cdot x_2+b=1 \qquad (2.1)$$

$$w\cdot x_1+b=-1 \qquad (2.2)$$

$$w\cdot x_2+b-w\cdot x_1-b=1-(-1)$$

$$w\cdot x_2-w\cdot x_1=2 \qquad (2.3)$$

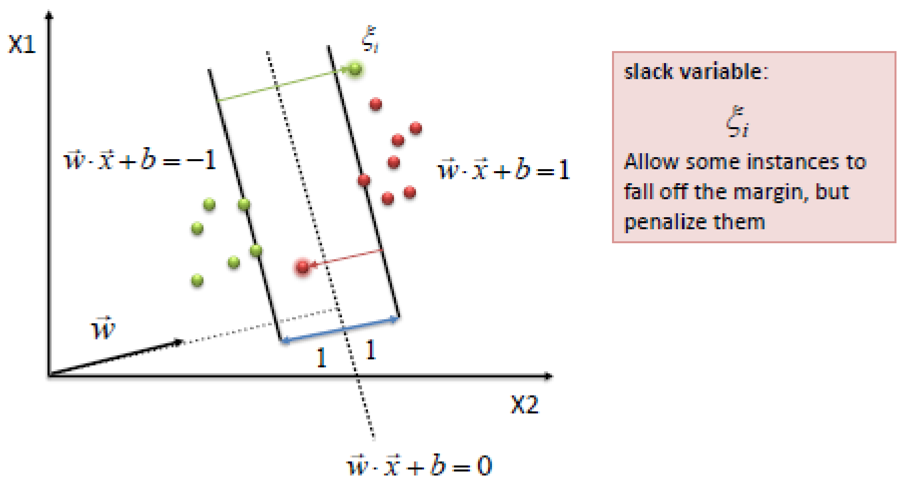
$$\frac{w}{\|w\|}(x_2-x_1)=\frac{2}{\|w\|}$$

We find *w* and *b* by solving the following objective function using Quadratic Programming.

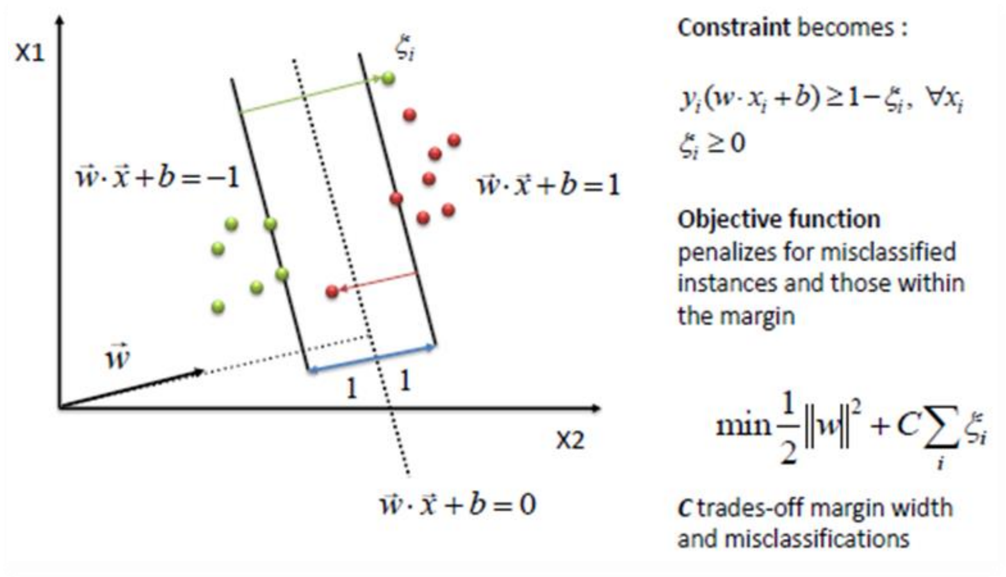
$$min\frac{1}{2}\|w\|^2$$

$$s.t.\, y_i(w.x_i+b)\geq 1, \forall x_i$$

The beauty of SVM is that if the data is linearly separable, there is a unique global minimum value. An ideal SVM analysis should produce a hyperplane that completely separates the vectors (cases) into two non-overlapping classes. However, perfect separation may not be possible, or it may result in a model with so many cases that the model does not classify correctly. In this situation SVM finds the hyperplane that maximizes the margin and minimizes the misclassifications.


X1
X2
ξ_i
$\vec{w}\cdot\vec{x}+b=-1$
$\vec{w}\cdot\vec{x}+b=1$
$\vec{w}\cdot\vec{x}+b=0$
$\vec{w}$
1
1
slack variable:
ξ_i
Allow some instances to fall off the margin, but penalize them

The algorithm tries to maintain the slack variable to zero while maximizing margin. However, it does not minimize the number of misclassifications (NP-complete problem) but the sum of distances from the margin hyperplanes.


X1
X2
ξ_i
$\vec{w}\cdot\vec{x}+b=-1$
$\vec{w}\cdot\vec{x}+b=1$
$\vec{w}\cdot\vec{x}+b=0$
$\vec{w}$
1
1
Constraint becomes :
$y_i(w\cdot x_i+b)\geq 1-\xi_i,\ \forall x_i$
$\xi_i\geq 0$
Objective function
penalizes for misclassified instances and those within the margin
$\min \frac{1}{2}\|w\|^2 + C\sum_i \xi_i$
C trades-off margin width and misclassifications

The simplest way to separate two groups of data is with a straight line (1dimension), flat plane (2 dimensions) or an N-dimensional hyperplane. However, there are situations where a nonlinear region can separate the groups more efficiently. SVM handles this by using a kernel function (nonlinear) to map the data into a different space where a hyperplane (linear) cannot be used to do the separation. It means a non-linear function is learned by a linear learning machine in a high-dimensional feature space while the capacity of the system is controlled by a parameter that does not depend on the dimensionality of the space. This is called *kernel trick* which means the kernel function transform the data into a higher dimensional feature space to make it possible to perform the linear separation.

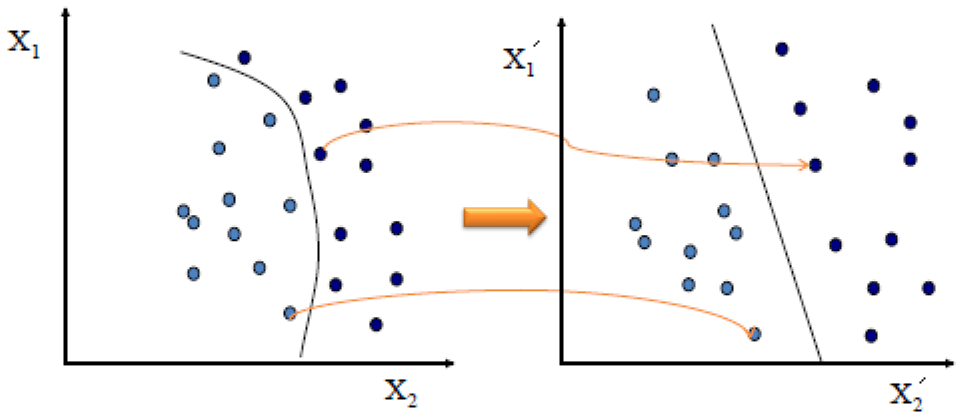
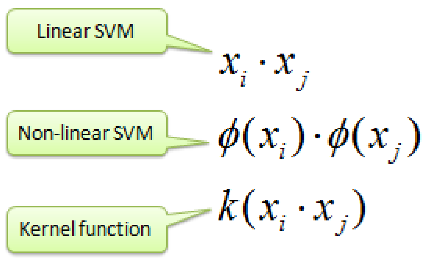


Figure 2.3 Kernel Trick



Map data into new space, then take the inner product of the new vectors. The image of the inner product of the data is the inner product of the images of the data. Two kernel functions are shown below.

Polynomial

$$k(\mathrm{x}_i, \mathrm{x}_j) = (\mathrm{x}_i . \mathrm{x}_j)^d \qquad (2.4)$$

Gaussian Radial Basis function

$$k(\mathrm{x}_i, \mathrm{x}_j) = \exp\left(-\frac{\|\mathrm{x}_i - \mathrm{x}_j\|^2}{2\sigma^2}\right) \qquad (2.5)$$

SVMs are a generally applicable tool for machine learning. Suppose we are given with training examples $\mathbf{x}_i$, and the target values y_i {-1,1}. SVM searches for a separating hyperplane, which separates positive and negative examples from each other with maximal margin, in other words, the distance of the decision surface and the closest example is maximal (see Figure. 2.4).

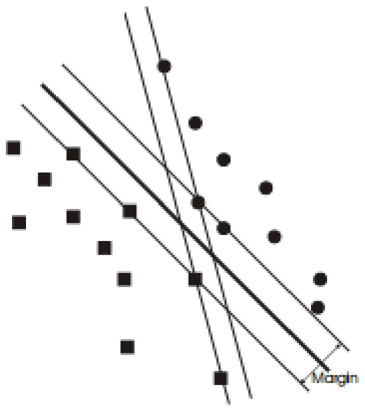


Figure 2.4 Maximal Margin SVM

The optimal decision surface with maximal margin vs. a non-optimal decision surface

The equation of a hyperplane is:

$$w^T x + b = 0 \qquad (2.6)$$

The classification of an unseen test example **x** is based on the sign of $w^T x$ + b. The separator property can be formalized as:

$$w^T x_i + b \geq 1 \text{ if } y_i = +1$$

$$w^T x_i + b \leq 1 \text{ if } y_i = -1$$

The optimization problem of SVM is the following:

Minimize over (w, b) the value of $\frac{1}{2} \cdot w^T w$ subject to:

$$\forall_{i=1}^{n} : y_i[\mathbf{w}^T \mathbf{x_i} + b] \geq 1$$

The optimal w hyperplane has the minimum expected error of classification on an unseen and randomly selected example.

SVM can be generalized to handle non-separable cases in two way. First, with slack variables: minimize over (**w**, b) the value of $1/2 \cdot \mathbf{w}^T \mathbf{w} + C \cdot \sum_{i=1}^{n} \xi_i$ subject to:

$$\forall_{i=1}^{n} : y_i[\mathbf{w}^T \mathbf{x_i} + b] \geq 1 - \xi_i$$

Where C is a constant to trade of between margin and training error.

Second, with kernel functions. Let $K(\boldsymbol{x_i}, x_j)$ denote a function that roughly speaking gives how similar two examples are. This is called a kernel-function if it satisfies the Mercers's condition [3]. The simplest case is when $K(x_i, x_j) = x_i^T \cdot x_j$, but more complicated cases makes SVM suitable for non-linearly separable problems. The optimization problem is described in [3].

An interesting property of SVM is that the normal of the decision surface is a linear combination of examples. This means, that the decision function can be written in the following form:

$$f(x) = \operatorname{sign}\left(\sum_{i=1}^{N} \alpha_i \cdot y_i \cdot K(\mathbf{x}_i, \mathbf{x}) + b\right) \qquad (2.7)\ [3].$$

Machine learning algorithms tend to overlearn when the dimensionality is high, for example when more dimension exists than example. SVM avoids this, because it does not combine features, but it linearly combines a function of the examples.

2.6.1 Pros and Cons associated with SVM

Pros:

- ➢ It works really well with clear margin of separation
- ➢ It is effective in high dimensional spaces.
- ➢ It is effective in cases where number of dimensions is greater than the number of samples.
- ➢ It uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.

Cons:

- ➢ It doesn't perform well, when we have large data set because the required training time is higher
- ➢ It also doesn't perform very well, when the data set has more noise i.e. target classes are overlapping
- ➢ SVM doesn't directly provide probability estimates, these are calculated using an expensive five-fold cross-validation. It is related SVC method of Python scikit-learn library.

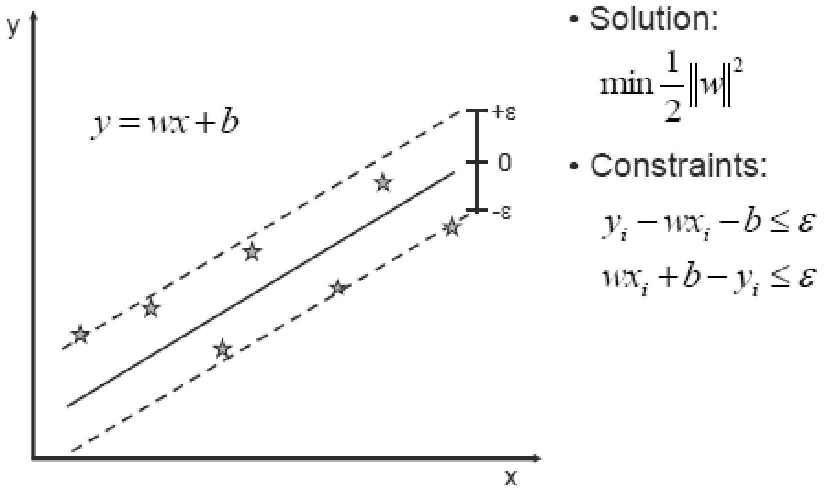
2.6.2 Support Vector Regression (SVR)

This section is about Support Vector Regression [22]. Those who are in Machine Learning or Data Science are quite familiar with the term SVM or Support Vector Machine. But SVR is a bit different from SVM. As the name suggest the SVR is a regression algorithm, so we can use SVR for working with continuous Values instead of Classification which is SVM.

Support Vector Machine can also be used as a regression method, maintaining all the main features that characterize the algorithm (maximal margin). The Support Vector Regression (SVR) uses the same principles as the SVM for classification, with only a few minor differences. First of all, because output is a real number it becomes very difficult to predict the information at hand, which has infinite possibilities. In the case of regression, a margin of tolerance (epsilon) is set in approximation to the SVM which would have already requested from the problem. But besides this fact, there is also a more complicated reason, the algorithm is more complicated therefore to be taken in consideration. However, the main idea is always the same: to minimize error, individualizing the hyperplane which maximizes the margin, keeping in mind that part of the error is tolerated.

The terms that we are going to be using frequently in this section

- Kernel: The function used to map a lower dimensional data into a higher dimensional data.
- Hyper Plane: In SVM this is basically the separation line between the data classes. Although in SVR we are going to define it as the line that will will help us predict the continuous value or target value
- Boundary line: In SVM there are two lines other than Hyper Plane which creates a margin. The support vectors can be on the Boundary lines or outside it. This boundary line separates the two classes. In SVR the concept is same.
- Support vectors: This are the data points which are closest to the boundary. The distance of the points is minimum or least.

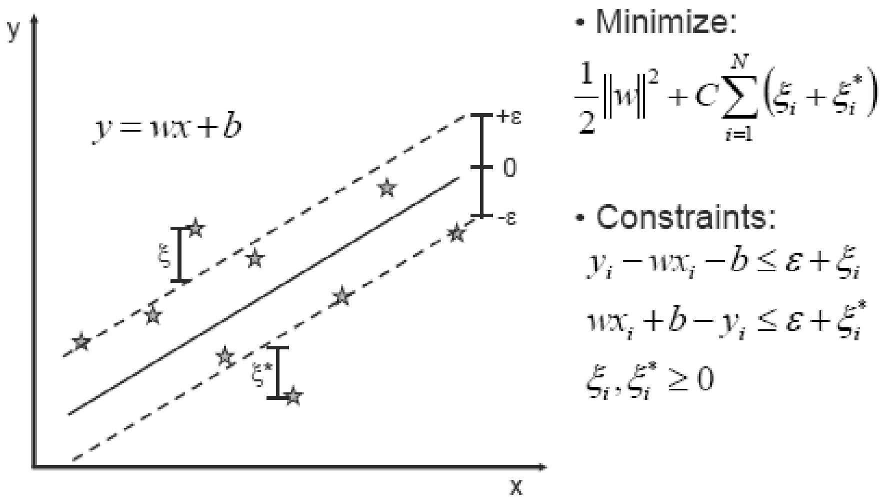


- Solution:

$$\min \frac{1}{2}\|w\|^2$$

- Constraints:

$$y_i - wx_i - b \le \varepsilon$$
$$wx_i + b - y_i \le \varepsilon$$



- Minimize:

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\left(\xi_i + \xi_i^*\right)$$

- Constraints:

$$y_i - wx_i - b \le \varepsilon + \xi_i$$
$$wx_i + b - y_i \le \varepsilon + \xi_i^*$$
$$\xi_i, \xi_i^* \ge 0$$

Linear SVR

$$y = \sum_{i=1}^{N}\left(\alpha_i - \alpha_i^*\right)\cdot\langle x_i, x\rangle + b \qquad (2.8)$$

Non-linear SVR

$$y = \sum_{i=1}^{N}\left(\alpha_i - \alpha_i^*\right)\cdot\langle \varphi(x_i), \varphi(x)\rangle + b \qquad (2.9)$$

$$y = \sum_{i=1}^{N}\left(\alpha_i - \alpha_i^*\right)\cdot K(x_i, x) + b \qquad (2.10)$$

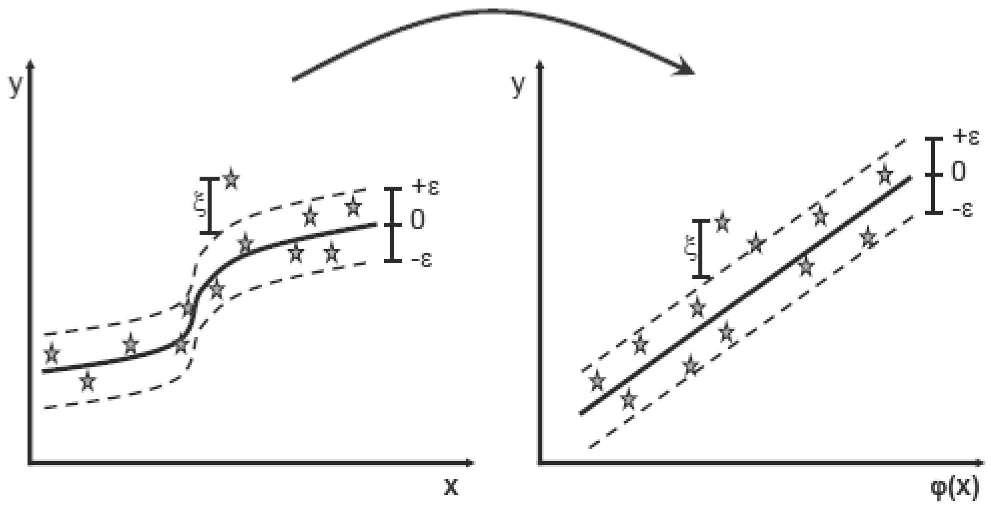


Figure 2.5 Kernel SVR

Kernel functions

Polynomial

$$k(\mathrm{x}_i, \mathrm{x}_j) = (\mathrm{x}_i . \mathrm{x}_j)^d \qquad (2.11)$$

Gaussian Radial Basis function

$$k(\mathrm{x}_i, \mathrm{x}_j) = \exp\left(-\frac{\left\|\mathrm{x}_i - \mathrm{x}_j\right\|^2}{2\sigma^2}\right) \qquad (2.12)$$

2.6.3 Difference between SVR and a simple regression model

In simple regression, the model tried to minimize the error rate. While in SVR, the model tried to fit the error within a certain threshold. See Figure 2.10, how all the points are within the boundary line (Red Line). The objective when the model moving on with SVR is to basically consider the points that are within the boundary line. The best fit line is the line hyperplane that has maximum number of points.

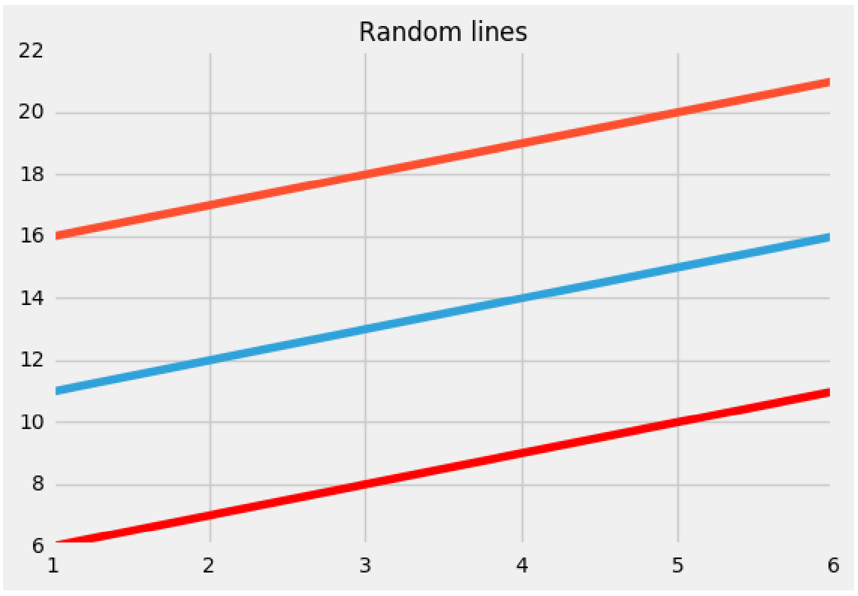


Figure 2.6 Blue line: Hyper Plane; Red Line: Boundary Line

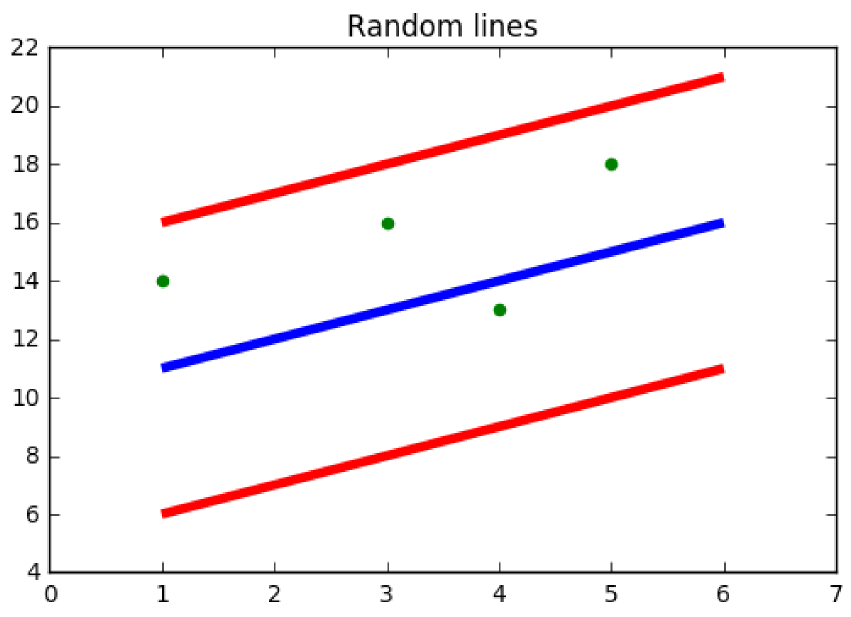


Figure 2.6 Data Points

So, Let’s Start:

So, the first thing is to understand is what is this boundary line? (yes! that red line). Think of it as to lines which are at a distance of ‘e’ (though not e its basically epsilon) but for simplicity lets say its ‘e’.

So, the lines at ‘+e’ and ‘-e’ distance from Hyper Plane.

Assuming our hyper plane is a straight line going through the Y axis

The Equation of the hyper plane is:

$$Wx + b = 0 \qquad (2.13)$$

So, The two the equations of the boundary lines are:

$Wx + b = +e$ (2.14)

$Wx + b = -e$ (2.15) respectively.

Thus, coming in terms with the fact that for any linear hyper plane the equation that satisfy SVR is:

$e \leq y - Wx - b \leq +e$

stating the fact that $y = Wx + b$:

$y - Wx - b = 0$ (2.16)

2.7 The Big Five personality traits

The Big Five personality traits [23], also known as the five-factor model (FFM) and the OCEAN model, is a taxonomy for personality traits. When factor analysis (a statistical technique) is applied to personality survey data, some words used to describe aspects of personality are often applied to the same person. For example, someone described as conscientious is more likely to be described as "always prepared" rather than "messy". This theory is based therefore on the association between words but not on neuropsychological experiments. This theory uses descriptors of common language and therefore suggests five broad dimensions commonly used to describe the human personality and psyche.

The five factors are:

- Openness to experience (*inventive/curious* vs. *consistent/cautious*)
- Conscientiousness (*efficient/organized* vs. *easy-going/careless*)
- Extraversion (*outgoing/energetic* vs. *solitary/reserved*)
- Agreeableness (*friendly/compassionate* vs. *challenging/detached*)
- Neuroticism (*sensitive/nervous* vs. *secure/confident*)

The five factors are represented by the acronym *OCEAN* or *CANOE*. Beneath each proposed global factor, there are a number of correlated and more specific primary factors. For example, extraversion is said to include such related qualities as gregariousness, assertiveness, excitement seeking, warmth, activity, and positive emotions.

Family life and the way someone was raised will also affect these traits. Twin studies and other research have shown that about half of the variation between individuals results from their genetics and half from their environments. Researchers have found conscientiousness, extraversion, openness to experience, and neuroticism to be relatively stable from childhood through adulthood.

2.7.1 Descriptions of the particular personality traits

- Openness to experience

Openness [22] is a general appreciation for art, emotion, adventure, unusual ideas, imagination, curiosity, and variety of experience. People who are open to experience are intellectually curious, open to emotion, sensitive to beauty and willing to try new things. They tend to be, when compared to closed people, more creative and more aware of their feelings. They are also more likely to hold unconventional beliefs. High openness can be perceived as unpredictability or lack of focus, and more likely to engage in risky behaviour or drug taking. Moreover, individuals with high openness are said to pursue self-actualization specifically by seeking out intense, euphoric experiences. Conversely, those with low openness seek to gain fulfillment through perseverance and are characterized as pragmatic and data-driven—sometimes even perceived to be dogmatic and closed-minded. Some disagreement remains about how to interpret and contextualize the openness factor.

Sample items:

- I have excellent ideas.
- I am quick to understand things.
- I use difficult words.
- I am full of ideas.
- I am not interested in abstractions. (*reversed*)
- I do not have a good imagination. (*reversed*)
- I have difficulty understanding abstract ideas. (*reversed*)

- Conscientiousness

Conscientiousness [22] is a tendency to display self-discipline, act dutifully, and strive for achievement against measures or outside expectations. It is related to the way in which people control, regulate, and direct their impulses. High conscientiousness is often perceived as being stubborn and focused. Low conscientiousness is associated with flexibility and spontaneity, but can also appear as sloppiness and lack of reliability. High scores on conscientiousness indicate a preference for planned rather than spontaneous behavior. The average level of conscientiousness rises among young adults and then declines among older adults.

Sample items:

- I am always prepared.
- I pay attention to details.
- I get chores done right away.
- I like order.
- I follow a schedule.
- I am exacting in my work.
- I never forget my belongings
- I always end up being helpful to most things
- I often remember where I last put my things
- I give attention to my duties

- Extraversion

Extraversion [22] is characterized by breadth of activities (as opposed to depth), surgency from external activity/situations, and energy creation from external means. The trait is marked by pronounced engagement with the external world. Extraverts enjoy interacting with people, and are often perceived as full of energy. They tend to be enthusiastic, action-oriented individuals. They possess high group visibility, like to talk, and assert themselves. Extroverted people may appear more dominant in social settings, as opposed to introverted people in this setting.

Introverts have lower social engagement and energy levels than extraverts. They tend to seem quiet, low-key, deliberate, and less involved in the social world. Their lack of social involvement should not be interpreted as shyness or depression; instead they are more independent of their social world than extraverts. Introverts need less stimulation, and more time alone than extraverts. This does not mean that they are unfriendly or antisocial; rather, they are reserved in social situations.

Generally, people are a combination of extraversion and introversion, with personality psychologist Eysenck suggesting that these traits are connected somehow to our central nervous system.

Sample items:

- I am the life of the party.
- I don't mind being the center of attention.

- I feel comfortable around people.
- I start conversations.
- I talk to a lot of different people at parties.
- I don't talk a lot. (*reversed*)
- I think a lot before I speak or act. (*reversed*)
- I don't like to draw attention to myself. (*reversed*)
- I am quiet around strangers. (*reversed*)
- I have no intention of talking in large crowds. (*reversed*)

- Agreeableness

The agreeableness [22] trait reflects individual differences in general concern for social harmony. Agreeable individuals value getting along with others. They are generally considerate, kind, generous, trusting and trustworthy, helpful, and willing to compromise their interests with others. Agreeable people also have an optimistic view of human nature.

Disagreeable individuals place self-interest above getting along with others. They are generally unconcerned with others' well-being, and are less likely to extend themselves for other people. Sometimes their skepticism about others' motives causes them to be suspicious, unfriendly, and uncooperative. Low agreeableness personalities are often competitive or challenging people, which can be seen as argumentative or untrustworthy.

Because agreeableness is a social trait, research has shown that one's agreeableness positively correlates with the quality of relationships with one's team members. Agreeableness also positively predicts transformational leadership skills. In a study conducted among 169 participants in leadership positions in a variety of professions, individuals were asked to take a personality test and have two evaluations completed by directly supervised subordinates. Leaders with high levels of agreeableness were more likely to be considered transformational rather than transactional. Although the relationship was not strong, it was the strongest of the Big Five traits. However, the same study showed no predictive power of leadership effectiveness as evaluated by the leader's direct supervisor.

Conversely, agreeableness has been found to be negatively related to transactional leadership in the military. A study of Asian military units showed leaders with a high level of agreeableness to be more likely to receive a low rating for transformational leadership skills. Therefore, with further research, organizations may be able to determine an individual's potential for performance based on their personality traits. For instance, in their journal article "Which Personality Attributes Are Most Important in the Workplace?" Paul Sackett and Philip Walmsley claim that conscientiousness and agreeableness are “important to success across many different jobs."

Sample items:

- I am interested in people.
- I sympathize with others' feelings.
- I have a soft heart.
- I take time out for others.
- I feel others' emotions.
- I make people feel at ease.
- I am not really interested in others. (*reversed*)
- I insult people. (*reversed*)
- I am not interested in other people's problems. (*reversed*)
- I feel little concern for others. (*reversed*)

- Neuroticism

Neuroticism [22] is the tendency to experience negative emotions, such as anger, anxiety, or depression. It is sometimes called emotional instability, or is reversed and referred to as emotional stability. According to Eysenck's (1967) theory of personality, neuroticism is interlinked with low tolerance for stress or aversive stimuli. Neuroticism is a classic temperament trait that has been studied in temperament research for decades, before it was adapted by the FFM. Those who score high in neuroticism are emotionally reactive and vulnerable to stress, also tending to be flippant in the way they express emotion. They are more likely to interpret ordinary situations as threatening, and minor frustrations as hopelessly difficult. Their negative emotional reactions tend to persist for unusually long periods of time, which means they are often in a bad mood. For instance, neuroticism is connected to a pessimistic approach toward work, confidence that work impedes personal relationships, and apparent anxiety linked with work.

Furthermore, those who score high on neuroticism may display more skin-conductance reactivity than those who score low on neuroticism. These problems in emotional regulation can diminish the ability of a person scoring high on neuroticism to think clearly, make decisions, and cope effectively with stress. Lacking contentment in one's life achievements can correlate with high neuroticism scores and increase one's likelihood of falling into clinical depression. Moreover, individuals high in neuroticism tend to experience more negative life events, but neuroticism also changes in response to positive and negative life experiences. Also, individuals with higher levels of neuroticism tend to have worse psychological well being.

At the other end of the scale, individuals who score low in neuroticism are less easily upset and are less emotionally reactive. They tend to be calm, emotionally stable, and free from persistent negative feelings. Freedom from negative feelings does not mean that low-scorers experience a lot of positive feelings.

Neuroticism is similar but not identical to being neurotic in the Freudian sense (i.e., neurosis.) Some psychologists prefer to call neuroticism by the term emotional instability to differentiate it from the term neurotic in a career test.

Sample items:

- I get irritated easily.
- I get stressed out easily.
- I get upset easily.
- I have frequent mood swings.
- I worry about things.
- I am much more anxious than most people.
- I am relaxed most of the time. (*reversed*)
- I seldom feel blue. (*reversed*)

2.8 Chapter Summary

In this chapter, the details of the theory background used in the system have been explained.

The system design and implementation of the system will be explained in next chapter.