

A Project Report On

**“BMI CLASSIFIER”**

Submitted by:

**AISHWARYA DIXIT**

**AKANCHHA SINGH BHADAURIA**

**MEENAKSHI KUMARI**

**MOTAMARRI SAI RATNA**

Under the Guidance Of

**Prof. Y. N.GHOLAP**

For the partial fulfillment of SL-VI Mini Project

DEPARTMENT OF INFORMATION TECHNOLOGY

ARMY INSTITUTE OF TECHNOLOGY

DIGHI HILLS, PUNE-411015 (2016 -2017)

University of Pune

# CERTIFICATE

This is to certify that STUDENT NAME has successfully submitted the mini project report on

“**TILTE OF MINI PROJECT**”

during the academic year 2017-18 for the partial fulfillment of the requirement Software Lab VI, Final Year of Information Technology under Savitribai Phule Pune University, Pune

**Prof Y N GHOLAP Dr. (Mrs.) Sangeeta Jadhav**

**Guide HOD IT**

# ACKNOWLEDGEMENT

Before getting into the thick of the things, we would like to express our deep gratitude to the people who helped us during the course of this project. We are grateful to our project guide **Prof. Y.N. Gholap** for his guidance throughout this project research and work.

We also wish to thank all the faculty members of Information Technology and our respectable Head of Department **Prof. Mrs. Sangeeta Jadhav** for their constant help, support and efficient teaching procedures.

**Name of Student**

**Seat No**

**INDEX**

**Chapter Title Page No**

Chapter 1 Introduction

Chapter 2- Overview

Chapter 3- Requirements

Chapter 4- Design

Chapter 5- Snapshots

Chapter 6- Conclusion

References

## CHAPTER-1

## INTRODUCTION

The system takes in some information from you and predicts whether you are underweight, overweight or normal according to your BMI . The nearest neighbour Algorithm has been used to make the classifier as it gives a satisfactory result.//google nearest neighbour Each time the user checks for its value, its data gets stored in the database for the training set and hence the database increases and the classifier is produced with a better training set.

# JAVA

Java is a general-purpose computer programming language that is concurrent, class-based, objectoriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation.

One design goal of Java is portability, which means that programs written for the Java platform must run similarly on any combination of hardware and operating system with adequate runtime support. This is achieved by compiling the Java language code to an intermediate representation called Java bytecode, instead of directly to architecture-specific machine code. Java bytecode instructions are analogous to machine code, but they are intended to be executed by a virtual machine (VM) written specifically for the host hardware. End users commonly use a Java Runtime Environment (JRE) installed on their own machine for standalone Java applications, or in a web browser for Java applets.

Standard libraries provide a generic way to access host-specific features such as graphics, threading, and networking.

The use of universal bytecode makes porting simple. However, the overhead of interpreting bytecode into machine instructions makes interpreted programs almost always run more slowly than native executables. However, just-in-time (JIT) compilers that compile bytecodes to machine code during runtime were introduced from an early stage. Java itself is platform-independent, and is adapted to the particular platform it is to run on by a Java virtual machine for it, which translates the Java bytecode into the platform's machine language.Java uses an automatic garbage collector to manage memory in the object lifecycle The programmer determines when objects are created, and the Java runtime is responsible for recovering the memory once objects are no longer in use.

* **CASSANDRA**
* Apache Cassandra is an open source distributed database management system designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure. Cassandra offers robust support for clusters spanning multiple datacenters[1] with asynchronous masterless replication allowing low latency operations for all clients.

Main features of Cassandra are :

* ***Decentralized*** o Every node in the cluster has the same role. There is no single point of failure. Data is distributed across the cluster (so each node contains different data), but there is no master as every node can service any request.
* ***Supports replication and multi data center replication*** o Replication strategies are configurable. Cassandra is designed as a distributed system, for deployment of large numbers of nodes across multiple data centers. Key features of Cassandra’s distributed architecture are specifically tailored for multiple-data center deployment, for redundancy, for failover and disaster recovery.  ***Scalability*** o Read and write throughput both increase linearly as new machines are added, with no downtime or interruption to applications.
* ***Fault-tolerant*** o Data is automatically replicated to multiple nodes for fault-tolerance. Replication across multiple data centers is supported. Failed nodes can be replaced with no downtime.
* ***Data Model***

Cassandra is essentially a hybrid between a key-value and a column-oriented (or tabular) database. Its data model is a partitioned row store with tunable consistency. Rows are organized into tables; the first component of a table's primary key is the partition key; within a partition, rows are clustered by the remaining columns of the key. Other columns may be indexed separately from the primary key.

Tables may be created, dropped, and altered at run-time without blocking updates and queries.

Cassandra does not support joins or subqueries. Rather, Cassandra emphasizes denormalization through features like collections.

A column family (called "table" since CQL 3) resembles a table in an RDBMS. Column families contain rows and columns. Each row is uniquely identified by a row key. Each row has multiple columns, each of which has a name, value, and a timestamp. Unlike a table in an RDBMS, different rows in the same column family do not have to share the same set of columns, and a column may be added to one or multiple rows at any time.

Each key in Cassandra corresponds to a value which is an object. Each key has values as columns, and columns are grouped together into sets called column families. Thus, each key identifies a row of a variable number of elements. These column families could be considered then as tables. A table in Cassandra is a distributed multi-dimensional map indexed by a key. Furthermore, applications can specify the sort order of columns within a Super Column or Simple Column family.

* ***DATA MINING***
* It is the computational process of discovering patterns in large data sets involving methods at the intersection of artificial intelligence, machine learning, statistics, and database systems. The overall goal of the data mining process is to extract information from a data set and transform it into an understandable structure for further use. Aside from the raw analysis step, it involves database and data management aspects, data pre-processing, model and inference considerations, interestingness metrics, complexity considerations, post-processing of discovered structures, visualization, and online updating. Data mining is the analysis step of the "knowledge discovery in databases" process, or KDD
* The actual data mining task is the automatic or semi-automatic analysis of large quantities of data to extract previously unknown, interesting patterns such as groups of data records (cluster analysis), unusual records (anomaly detection), and dependencies (association rule mining). This usually involves using database techniques such as spatial indices. These patterns can then be seen as a kind of summary of the input data, and may be used in further analysis or, for example, in machine learning and predictive analytics. For example, the data mining step might identify multiple groups in the data, which can then be used to obtain more accurate prediction results by a decision support system. Neither the data collection, data preparation, nor result interpretation and reporting is part of the data mining step, but do belong to the overall KDD process as additional steps.

## *Nearest Neighbour Algorithm*

*k*-NN is a type of [instance-based learning](https://en.wikipedia.org/wiki/Instance-based_learning), or [lazy learning](https://en.wikipedia.org/wiki/Lazy_learning), where the function is only approximated locally and all computation is deferred until classification. The *k*-NN algorithm is among the simplest of all [machine learning](https://en.wikipedia.org/wiki/Machine_learning) algorithms.

Both for classification and regression, it can be useful to assign weight to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. For example, a common weighting scheme consists in giving each neighbor a weight of 1/*d*, where *d* is the distance to the neighbor.[[2]](https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm#cite_note-2)

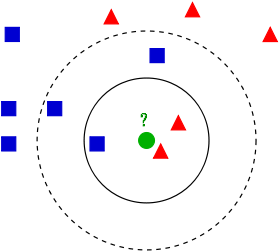
The neighbors are taken from a set of objects for which the class (for *k*-NN classification) or the object property value (for *k*-NN regression) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required.

A shortcoming of the *k*-NN algorithm is that it is sensitive to the local structure of the data

The training examples are vectors in a multidimensional feature space, each with a class label. The training phase of the algorithm consists only of storing the [feature vectors](https://en.wikipedia.org/wiki/Feature_vector) and class labels of the training samples.

In the classification phase, *k* is a user-defined constant, and an unlabeled vector (a query or test point) is classified by assigning the label which is most frequent among the *k* training samples nearest to that query point.

A commonly used distance metric for [continuous variables](https://en.wikipedia.org/wiki/Continuous_variable) is [Euclidean distance](https://en.wikipedia.org/wiki/Euclidean_distance). For discrete variables, such as for text classification, another metric can be used, such as the **overlap metric** (or [Hamming distance](https://en.wikipedia.org/wiki/Hamming_distance)). In the context of gene expression microarray data, for example, *k*-NN has also been employed with correlation coefficients such as Pearson and Spearman. Often, the classification accuracy of *k*-NN can be improved significantly if the distance metric is learned with specialized algorithms such as [Large Margin Nearest Neighbor](https://en.wikipedia.org/wiki/Large_Margin_Nearest_Neighbor) or [Neighbourhood components analysis](https://en.wikipedia.org/wiki/Neighbourhood_components_analysis" \o "Neighbourhood components analysis).



A drawback of the basic "majority voting" classification occurs when the class distribution is skewed. That is, examples of a more frequent class tend to dominate the prediction of the new example, because they tend to be common among the *k* nearest neighbors due to their large number.[[4]](https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm#cite_note-Coomans_Massart1982-4) One way to overcome this problem is to weight the classification, taking into account the distance from the test point to each of its *k* nearest neighbors. The class (or value, in regression problems) of each of the *k* nearest points is multiplied by a weight proportional to the inverse of the distance from that point to the test point. Another way to overcome skew is by abstraction in data representation. For example, in a [self-organizing map](https://en.wikipedia.org/wiki/Self-organizing_map) (SOM), each node is a representative (a center) of a cluster of similar points, regardless of their density in the original training data. *K*-NN can then be applied to the SOM

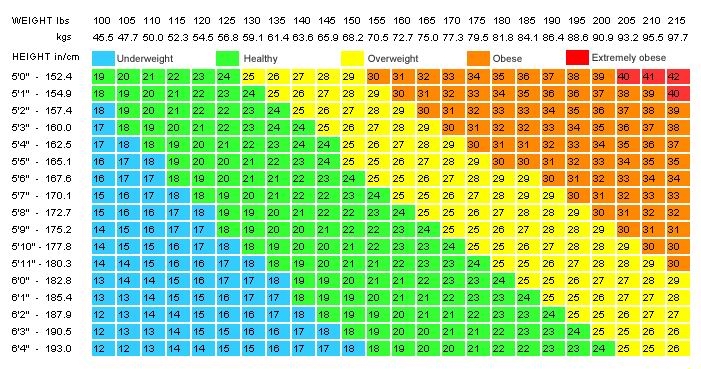
## CHAPTER-2

## OVERVIEW OF THE PROJECT

**SYSTEM DESCRIPTION**:

In today’s world people stopped caring about their health. So there is a need for proper diet management and analyse once status of BMI.

***Data mining on BMI***is aimed at developing a system which will allow user to input few information regarding their height and weight and then predict their status of health in terms of height and weight. It gives the user an idea of their health.



.

The **body mass index** (**BMI**) or **Quetelet index** is a value derived from the [mass](https://en.wikipedia.org/wiki/Mass) ([weight](https://en.wikipedia.org/wiki/Mass_versus_weight)) and height of an individual. The BMI is defined as the [body mass](https://en.wikipedia.org/wiki/Human_body_weight) divided by the [square](https://en.wikipedia.org/wiki/Square_(algebra)) of the [body height](https://en.wikipedia.org/wiki/Human_height), and is universally expressed in [units](https://en.wikipedia.org/wiki/Units_of_measurement) of kg/m2, resulting from mass in [kilograms](https://en.wikipedia.org/wiki/Kilogram) and height in [metres](https://en.wikipedia.org/wiki/Metre" \o "Metre).

The BMI may also be determined using a table[[note 2]](https://en.wikipedia.org/wiki/Body_mass_index#cite_note-3) or chart which displays BMI as a function of mass and height using contour lines or colours for different BMI categories, and which may use other units of measurement (converted to metric units for the calculation).[[note 3]](https://en.wikipedia.org/wiki/Body_mass_index#cite_note-4)

The BMI is an attempt to quantify the amount of tissue mass (muscle, fat, and bone) in an individual, and then categorize that person as *underweight*, *normal weight*, *overweight*, or *obese* based on that value

**SYSTEM GOALS**

The System aims at:

* Building a classifier from the dataset.
* Running the classifier on the user information.
* Providing the category it belongs to
* 1. UNDER WEIGHT
* 2. NORMAL
* 3.OVER WEIGHT

## CHAPTER-3

## REQUIREMENTS

The basic **softwares** used for this mini-project are listed as under:

1. **Eclipse**
2. **Cassandra Database**
3. **Weka Java API**

A team has incorporated several standard ML techniques into a software "workbench" called[WEKA,](http://www.cs.waikato.ac.nz/ml/weka) for Waikato Environment for Knowledge Analysis. With it, a specialist in a particular field is able to use ML to derive useful knowledge from databases that are far too large to be analysed by hand. WEKA's users are ML researchers and industrial scientists, but it is also widely used for teaching.

The most common components you might want to use are

* Instances - your data
* Filter - for preprocessing the data
* Classifier/Clusterer - built on the processed data  Evaluating - how good is the classifier/clusterer?
* Attribute selection - removing irrelevant attributes from your data

No such additional **hardware** is required except for the basic computer system.

For the **User**, the sound knowledge of JAVA and Netbeans and the knowledge of Cassandra DBMS is the prerequisite for this project.

Concepts of document oriented database(Cassandra), Data mining, Entity-Relationship Diagram should also be clear. A few Jar files are also required e.g.

* 1. Cassandra-driver-core-2.1.9
  2. Cassandra-driver-dse-2.1.9
  3. Cassandra-driver-mapping-2.1.9
  4. Guava-14.0.1
  5. Log4j-over-slf4j-1.7.13
  6. Matrix-core-3.0.2
  7. Netty-all-4.0.28

**CHAPTER-4:DESIGN**

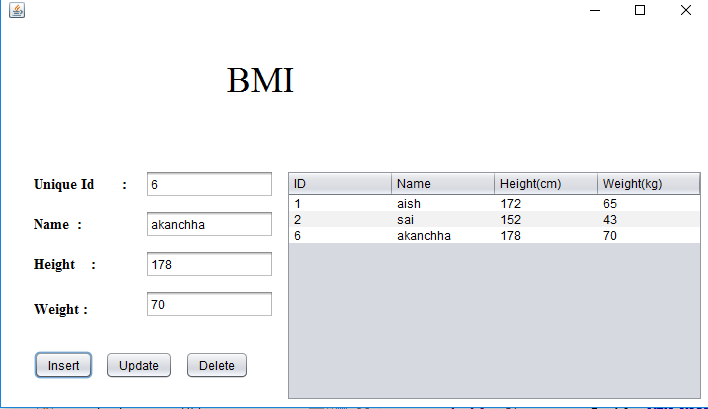
### Use Case DIAGRAM

**Data Modelling**

**Database Design**

### CHAPTER-5

### SNAPSHOTS



## CHAPTER-6

## CONCLUSION

The system provides a predictor for salary. The Nearest neighbour algorithm used provides a satisfactory success rate. The system can be used for anyone to predict their fitness according to BMI and check where the user is healthy or not. If the data set is huge , the error rate is less. The Cassandra database can be efficiently used for storing such a large dataset and for the performance of the basic CRUD operations. We showed how just a set of unrelated data of an individual can be used for the extraction of useful information and trend. This is the power of machine learning and databases like Cassandra.

Therefore, Nearest neighbour in machine learning can be used to get useful information from big database. The NoSQL databases like Cassandra give a powerful platform to store large number of rows in thousands or more and apply aggregation and big data analysis.

## REFERENCES

1. https://en.wikipedia.org/wiki/Data\_mining
2. https://en.wikipedia.org/wiki/Body\_mass\_index
3. https://en.wikipedia.org/wiki/nearest\_neighbour
4. https://weka.wikispaces.com/Use+WEKA+in+your+Java+code
5. https://en.wikipedia.org/wiki/Java\_(software\_platform)
6. https://en.wikipedia.org/wiki/Apache\_Cassandra
7. http://www.sciencedirect.com/science/article/pii/S0001457508002340
8. https://web.cse.msu.edu/~cse802/notes/ConstrainedKnn.pdf
9. http://link.springer.com/chapter/10.1007%2F978-3-642-10665-