UNIVERSITY OF THE PELOPONNESE
SCHOOL OF ECONOMICS, MANAGEMENT AND INFORMATICS
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

Knowledge and Uncertainty Research Laboratory

Postgraduate Thesis

Extracting emotions from songs with Machine Learning algorithms

Philippos Pappas
A.ID 2022201602015

Supervisor:

Wallace Manolis
Assistant Professor

Tripolis, December 2018

Approved by the committee of Inquiry on December 12, 2018.

Manolis Wallace	Konstantinos Peppas	Angeliki Antoniou
Assistant Professor	Lecturer	Laboratory Instruction Staff

....
Philippos Pappas

Department of Information Technology and Telecommunications
University of Peloponnese

This work was in collaboration with the Research Group Knowledge and Uncertainty (ΓΑΒ LAB)

Abstract

The aim of this thesis is to examine the emotional state of songs based on their lyrics alone, in order to observe if the lyrics are enough to discern emotions through music. The topic of this research relies on Music Information Retrieval field whose is purpose is to extract various information from the music through research and computational tools and can be applied to different kinds of fields such as musicology, psychology, machine learning etc.

As a first step, we will study various researches that have been conducted around the scientific field of Music Information Retrieval and after that we will see the impact of machine learning techniques in Music Classification. Then, we will review all the available music databases, their musical metadata and their growing community as the user activity on these systems keeps growing daily.

In the second phase of our thesis we will describe our ground truth construction and specifically we will mention from which resource we gathered the data, in which form and how we processed it in order to create our training dataset. Once we finish these notations, we will be ready to specify our classifiers and how we built them for our goal.

Next, we will present our experimental results, analyze all the techniques and approaches used for our project, aiming for the most optimal algorithm that we have applied so far. Finally, as this analysis will be completed, we will discuss the conclusions that we have reached and possible extensions of this research.

The achievement of the project is based on the above steps and aims at extracting emotions from the lyrics of the songs and how this can be achieved with simple methods.

Περίληψη

Ο σκοπός της παρούσας διπλωματικής είναι η ανάλυση συναισθημάτων των τραγουδιών μέσα από τους στίχους τους για να παρατηρήσουμε αν μόνο οι στίχοι είναι αρκετοί για την εκτίμηση της συναισθηματικής κατάστασης των τραγουδιών. Το θέμα της εργασίας αυτής κατατάσσεται στον ερευνητικό τομέα της Ανάκτησης Μουσικής Πληροφορίας που ως στόχο έχει την εξόρυξη γνώσης από τη μουσική με διάφορες μεθόδους και με εφαρμογές όπως στη μουσικολογία, στη ψυχολογία, στη μηχανική μάθηση κ.α.

Στο πρώτο στάδιο της διπλωματικής θα μελετηθούν διάφορες έρευνες γύρω από το τομέα αυτό καθώς και τις θεωρίες στις οποίες είναι βασισμένοι. Ύστερα θα δούμε πως η μηχανική μάθηση και οι εφαρμογές της συνδέονται με το τομέα της Ανάκτησης Μουσικής Πληροφορίας όπως επίσης και τις μουσικές βιβλιοθήκες και βάσεις δεδομένων που υπάρχουν και ολοένα μεγαλώνουν αφού κάθε μέρα ο αριθμός των χρηστών τους αυξάνεται σημαντικά.

Στη δεύτερη φάση της εργασίας αυτής θα αναφέρουμε πως και από που συλλέξαμε τα δεδομένα μας και θα περιγράψουμε πως τα ετοιμάσαμε για την εκπαίδευση του συστήματος(training datasets) αφού πρώτα αναφέρουμε το μοντέλο στο οποίο κατασκευάσαμε για να βασίσουμε την έρευνα μας.

Έπειτα θα ασχοληθούμε με τα αποτελέσματα που προκύψαν καθώς και με την ανάλυση αυτών και πιο συγκεκριμένα συγκρίνοντας τις διάφορες τεχνικές και παραμέτρους που λάβαμε υπόψιν με σκοπό να αναδείξουμε το βέλτιστο αλγόριθμο για την μέχρι τώρα εξέλιξη της έρευνας. Τέλος θα μιλήσουμε στα συμπεράσματα στα οποία καταλήξαμε και πιθανές επεκτάσεις της έρευνας αυτής.

Η αιτία των προαναφερθέντων βημάτων έγκειται στη προσπάθεια εξόρυξης συμπερασμάτων σχετικά με τη συναισθηματική πληροφορία που εμπεριέχεται στους στίχους των τραγουδιών και με ποιους τρόπους αυτό μπορεί να επιτευχθεί η εξόρυξη αυτή.

Dedicated to my family for supporting me over the years.

Contents

Chapter 1

Introduction

We live in the information age, where technology mediates and redefines more and more aspects of life, changes the way people think, communicate, research, and generally see and learn things that in past decades, existed only in the imagination.

Nowadays, technology has become an integral part of people's daily lives and as its evolution is taking place rapidly, computers and smartphones make it easier and easier for someone to use websites or social networks and this has the effect of increasing users on a daily basis. Many of people's daily activities are now taking place via the Internet, where the exchange of information and interactions are possible in a matter of seconds. Such interactions could be chatting, learning, buying and selling goods or services, entertaining, publishing images or videos and more. From a technological perspective, collection of data can happen everywhere, anytime and from different sources, from your local super-market that stores transaction records, to government agencies. All this leads to large data sets or in just two words, Big Data.

But all this massive data collection would be meaningless without analyzation. Data mining is the tool for discovering hidden relations between this data as you dig through, deeper and deeper in to large data sets. It keeps pace with the limitless potential of big data and also comprises three intertwined scientific disciplines such as statistics, artificial intelligence and machine learning. But, in order to go through these large data sets and seek for relevant and useful information we need to use specific techniques. These include classifying data into predefined classes (classification), or to dividing a set of patterns into different and homologous groups (clustering) or to identify frequent patterns in the data in the form of dependencies (associations).

Data mining has offered another approach to the field of research as it is also applied in areas like health, biology, criminology, musicology, agriculture, environment, transportation, politics and more. Specifically, the science of data in musicology or Music Information Retrieval (MIR) and Music Data Mining (MDM) are new interdisciplinary sciences with small scientific field in terms of research, but they are growing rapidly in recent years. The general scope of these fields is to extend the understanding and usefulness of music data in order to create algorithms that can ”understand or make sense of” this data. Such algorithms can be for emotion, mood or genre recognition, beat, rhythm or tempo tracking, signal analysis and more.

This thesis is examining the emotional state of songs according to the lyrics. The scope is to investigate if lyrics alone are enough to estimate the emotional state of a song. Our process will include lyrics gathering from web and open music databases, labeling lyrics into predefined classes and finally, classification process in order to verify our prediction accuracy.

In the following chapters we will analyze the theoretical background as well as our aim. We will mention why it is important, how this could be useful in various areas of research.

Specifically in chapter 2 we will meet the literature review of our thesis. We will refer to the previous works that have been done in the past, as well as a brief review of these.

In chapter 3 we will deal with similar researches that we studied during our work but also how our data was collected and finally analyzed.

Chapter 4 will provide detailed information about our approaches and how the data has been converted and used by the classification algorithms. Last, we will close this chapter by mentioning the prediction accuracy rates for each approach and for each applied algorithm.

Chapter 5 will be followed by the analysis of all the results and we will see all the factors that have been taken into account to lead us to these conclusions.

Last but not least, in chapter 6, we will include our conclusions that we drew from this project but also the chapter will contain the possibilities of improvement and our planned future work.

Chapter 2

Literature Review

Since ancient times music has always been present in our lives but during the last decade with the rapid evolution of technology, music has managed to be almost everywhere in our everyday life. People listen to music during plenty of their daily primary activities like studying, exercising, relaxing, even in stores, cafes, bars and many more places and situations. Based on this, the size of digital musical data has increased dramatically and has given the opportunity for even greater research in the field of music and, as mentioned earlier, has created new research areas such as Music Information Retrieval, Music Data Mining and Music Emotion Recognition (MER).

2.1 Music Information Retrieval (MIR)

Music Information Retrieval deals with the extraction of essential knowledge and features from music and this is achieved with in a variety of computational methods. MIR is used mostly by businesses or academics to create applications and tools sto categorize and manipulate music. Such applications serve different purposes like music recommendation systems but few of these systems are based upon MIR techniques. That's because similarity between users has been used, instead of similarity between music. For example there are systems that are recommending you unheard music according your ”listening” history. In spite of this fact, MIR techniques have slowly begun to integrate into these systems. Track separation and instrument recognition is an another application example where the software can recognize and separate musical instruments from the original music song, into one track per instrument (e.g. karaoke).

The general goal of automatic music creation is a dream of many MIR researchers since the successful efforts so far are limited in terms of human appreciation of the results.

2.2 Classification

Big Data is a term that not only concerns other scientific fields beyond computer science such as health, law, biology, finance etc. but in time will affect many aspects of our everyday life. The great need for qualitative use of this data has led to the demand for large storage spaces and, therefore, to the time-consuming and complex process operations regarding the exploration of this data. Association, classification, clustering, regression are a few data mining techniques that are used to overcome these kinds of problems and make sense of the data. Each one is structured and applied in different cases depending on the data and the purpose.

Machine learning algorithms are divided in to supervised and unsupervised learning algorithms. In unsupervised learning algorithms, clustering is applied with the class labels of the data to be unknown. This technique aims to detect similarities between the data in order to create clusters of a relevant topic. On the other hand, in the unsupervised learning, classification is the preferred technique. Unlike clustering method, classification task begins with the data and the assigned classes to be known, called training set. Each supervised algorithm is based on the same training set and seeks for different types of patterns in order to classify the unlabelled data. Classification is a two-step process. First, model construction takes place where the training data form the new classifier and the classification rules. This process is also called learning. Afterwards, a single input or a set of uncategorized attributes -which are called prediction set- are analyzed according to the classifier in order to produce a prediction or make a decision. When the training data are separated into two classes, it's called binomial classification, otherwise it is called multi-class classification. This method has several evaluation measures but to make the model efficient the main factors are speed and accuracy. Classification is used in numerous applications like credit approval, business, marketing, biomedicine, fraud detection and more. [23] [25]

2.2.1 Naive Bayes

Naive Bayes belongs to the family of supervised algorithms in machine learning and is based on the so-called Bayesian theorem. The main characteristic of the NB classifier is that it doesn't take into account the correlation between the feature classes, given a class variable. It is one of the most famous classification methods and despite its simplicity, it performs really well in many real life problems and in some cases can even outperform more sophisticated classification techniques. It has easy implementation, it is fast, accurate and since it has low requirements in terms of CPU and memory, training time is significantly shorter compared to other algorithms.

Bayes' Theorem

Bayes' theorem, which is also known as Bayes' law proposed by Thomas Bayes (1702 – 1761) is a method which calculates the probability of a hypothesis given our prior knowledge. To make a prediction, the algorithm calculates probabilities of the instance belonging to each class and selects the class value with the highest probability. The following equation provides the mathematical structure of the theorem:

Figure 2.1: Bayes' Theorem

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

Analyzing this statement we have:

- P(A|B): is the probability of hypothesis A given the data B. This is called the posterior probability.
- P(A) and P(B): Probabilities of the occurrence of event A and B respectively, unrelated to each other.
- P(B|A): is the probability of data B given that the hypothesis A was true.

Usage of Bayesian Classification

Bayesian classification can be applied in many use-cases of the real world with the most common case to be email filtering[39]. Considering the content of an email, the Bayesian spam filter calculates the probability of this message being spam. This method belongs to the family of content-based spam filters which seek for words and other characteristics of typical spam. The decisions are made by a cluster of already classified spam and a cluster of good emails. When a new message arrives in the user's inbox, the filters analyze it by looking at both clusters for similarities and calculate the probability of various characteristics appearing in spam and in good emails respectively. It is important to mention that each user has their "own" spam filters, since the spam emails are pften related to the user's online activities or to their subscriptions to newsletters.

Several researches have been conducted about efficiency of Naive Bayes in categorization of news articles or even in Google Play Apps [1], and concluded that in the majority of times this method performs really well, fast and has a high percentage of accuracy. Careless data collection and analysis can lead to a problematic classifier and affect its performance. As mentioned above, Naive Bayes itself considers each word as a unique feature (which many times turns out to be a defect as we see also in the cited research), so pre-processing plays a significant role in data optimization in order to achieve satisfactory results.

This machine learning technique is used widely in the research field of data science. Several examples can be category or language detection, face recognition, recommendation systems, text expressing emotions and even disease diagnosis in order to take decisions about treatment processes.[20]

2.2.2 k-Nearest Neighbor

Another member of the supervised algorithms family is k-NN method (k-Nearest Neighbors) which is also part of instance-based classifiers [38]. The main characteristic of these classifiers is that they do not produce any model, unlike Neural Networks or Decision Trees which they create a training model. Training data is already stored in the memory and is compared whenever a new instance is requested to be classified. These type of classifiers are called lazy classifiers. As mentioned above, all calculations and comparisons take place each time a new sample occurs which has a high computational cost and is the basic drawback of these classifiers. In theory, the new instance is labeled according to the k-Nearest Neighbors of this case. To search for a nearest neighbor around the newly placed instance, a spherical area is developed to include k training samples. Hence, the majority of the k-classes of the training samples within the sphere also determine its categorization.

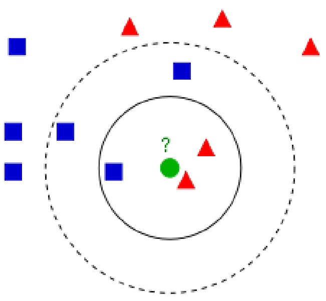
Two parameters must be considered in order to fully define the algorithm:

- The distance measure between instances a metric value in instance space, which will express the proximity, or otherwise the "similarity" between the instances.
- The K value

Selection of K-value

The best selection for k value depends on the type of the data. Large values of k reduce the noise effect on classification but they make distinct boundaries between classes. A good k can be selected by various heuristic techniques. [38] In case of binary classification problems it is preferable for k to choose an odd number in order to avoid ties. In cases with more classes, the possible ties are resolved either randomly or by assigning the new instance to the closest neighboring class. Different values of k plays an important role in the kNN classifier as different values instances are classified differently. As we see below in the figure, training data is separated into red triangles and blue squares categories while the green circle is our new instance that we need to classify. So if k=3 then we classify our new instance in the red triangles category since we have two triangles and one square inside the small inner circle. While for k = 5, which indicates the outer circle with dashed lines, the prediction is the squared class.

Figure 2.2: kNN Classification Example



The efficiency of the algorithm is based on k, since very small values will affect the result with noise while there is a chance that the result will contain many instances from other classes if the value is large. It is important to mention that if k is set to 1, then the new instance is assigned to the most similar training class.

Distance Measure

Classification algorithms are based on distance measures and they can use different distance calculation methods. For simplicity reasons we consider one classification problem, where the instances consist of two numerical fields and the class. Every instance can be seen as a point in a two-dimensional space and an instance X is spaced from another instance Y, distance d(X,Y). To measure these distances there are several approaches:

Euclidean Distance

A metric that is commonly used is known Euclidean distance. The Euclidean distance between points p, q is given by the formula:

Figure 2.3: Euclidean Distance Equation

$$d(p,q) = \sqrt{(p1-q1)^2 + (p2-q2)^2 + \ldots (p_n - q_n)^2}$$

Squared Euclidean distance

The standard Euclidean distance can be squared in order to place progressively greater weight on objects that are farther apart in order to optimize the problem of finding the distance.[34]

Cosine Similarity

Cosine similarity distance is a measure that calculates the cosine of the angle between two vectors. Calculations between documents are performed in a normalized space between the angles of the documents. The TF-iDF value for each word of each document also constitute part of this metric which is described in the following equation.

Figure 2.4: Cosine Similarity Equation

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Manhattan Distance

An alternative method of Euclidean distance is Manhattan distance with the difference being that it does not calculate the root of the square but the absolute values where it results in the following formula

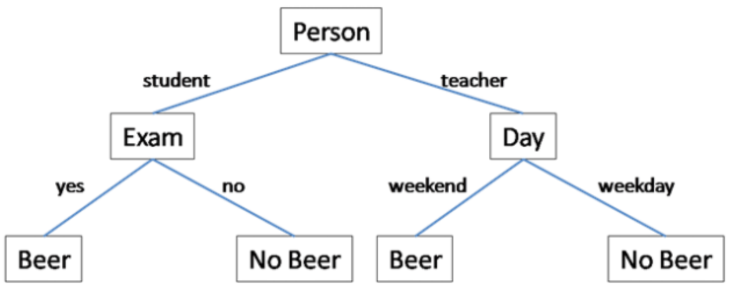
Figure 2.5: Manhattan Distance Equation

$$d(x,y) = \sum_i |x_i - y_i|$$

2.2.3 Decision Trees

Decision Trees are a principle method in data mining and machine learning that are used to solve classification problems and are considered one of the most practical and simple approaches of data mining. It is the most well-known in the family of Supervised learning algorithms and it has been successfully applied in many areas where classification is required: for example, in face recognition in images, medical diagnosis of incidents and more generally, knowledge mining. The algorithm builds from an already classified data, a tree-like form which can be described as a set of rules called classification rules and the leaves are classification classes.

Figure 2.6: Simple Tree Presenation



A Decision Tree is essentially a structure in which each node represents a choice between different nodes alternatives, and each leaf node represents a classification or a decision. [32] The Decision Tree approach is based on ”divide and conquer” technique and is very useful in classification problems. Generally, a Decision Tree in essence is a series of if-then rules (IF-THEN) that are combined from the tree root to the leaves. The nodes of the tree are characterized by the names of the attributes,the edges are named with the possible values that can be featured and the leaves with the various classes. The first stage of the classification is that the input feature begins from the very top node (root) of the tree where a check is in place to determine which child-node will follow. This process is repeated until the input feature reaches the leaf-node. All inputs on the tree that end up on a particular leaf are classified in the same way. This path is an expression of the rules that is used to classify the new inputs. During learning stage, a tree can be ”learned” by splitting the source set into subsets according to an attribute value test. With the same recursive method being followed in each subset this process achieves what is called recursive partitioning. The algorithm inserts at the root node the variable which can seperate the final classes in the best possible way. The algorithm stops when it reaches a node from which it cannot start a new split. Then this node has no children and is a leaf of the tree. There are two main types of Decision Trees in data mining; we will analyze below a few algorithms that apply to them.

- Classification tree analysis is when the predicted outcome is the class to which the data belongs.
- Regression tree analysis is when the predicted outcome can be considered a real number.

ID3

ID3 [36] is one of several simple Decision Tree classification algorithms and it begins with an original set as a root node. The main idea is to construct a top-down tree and in each iteration of the algorithm, to iterate through every unused attribute of the original set, seeking for the most relative example for the classification through information gain selection (the largest information gain value or the smallest entropy value). Then, the set is split into subsets by the selected attribute and the algorithm continues to recurse on each subset, considering only attributes never selected before in order to minimize the comparisons.
The concept used to measure information is called entropy. Entropy is used to measure the quantity of uncertainty in a set of data but there is no uncertainty if all the data of a set belongs to a single category and in this case entropy is null. The aim of a Decision Tree classifier is to repeatedly divide into subsets the original data set, in order to reach the final dataset where all the examples are from the same class. If that it is not achievable then the algorithm stops as long as there are no more attributes to be selected, but the examples still do not belong to the same class. Another case in which algorithm stops is when there are no examples in the subset. This happens when no example in the parent set was found to be matching a specific value of the selected attribute.
Entropy measures the amount of uncertainty in a dataset

Figure 2.7: Entropy Measure Equation

$$H(S) = \sum_{x \in X} -p(x) \log_2 p(x)$$

Where: S - The temporary data set (changing in each iteration) X - Set of classes that belongs to S p(x) - The proportion of the number of elements in class x to the number of elements in set S

Information gain calculates the reduction of the uncertainty in each iteration of the algorithm after splitting the data set on attribute

Figure 2.8: Information Gain Equation

$$IG(A,S) = H(S) - \sum_{t \in T} p(t)H(t)$$

Where: H(S) - Entropy of set S T - The produced subsets from set S byt attribute A p(t) - The proportion of the number of elements in t to the number of elements in set S H(t) - Entropy of subset t

C4.5

C4.5 is an extension of ID3 developed by Quinlan Ross in 1993[27] and Decision Trees of such type can be used for classification and for this reason is often called a statistical classifier. C4.5 improves ID3 because missing data is ignored in gain and entropy calculations and allows attributes to be labeled as question mark(?) for missing data. After the creation, the algorithm starts the pruning process, crossing the whole tree, deciding which useless branches (subsets) will be replaced as leaves. Two of the most common pruning approaches are Subtree Raising and Subtree Replacement. The algorithm is flexible in terms of classification as it also allows classification with rules created by Decision Trees. Handling attributes with differing costs and both continuous and discrete attributes is also an advantage.

J48

J48 is an extension of ID3 and in WEKA data mining tool is an open source Java implementation of C4.5. The algorithm produces Decision Trees with a greedy technique and includes all the features of C4.5 like pruning, missing values and discrete/continuous attributes handling. Firstly, a Decision Tree is created which is based on the attributes of the training data, identifying the attribute which is able to discriminate the data instances in the best way, in order to classify and obtain the highest information gain from them. The algorithm is an implementation of C4.5 as mentioned before, so it works similarly to how we described earlier but also can be modified to improve its performance for various purposes.[19] [4]

Random Forest

Random Forest was developed in 2001 by Leo Breiman and Aden Cutler is a Decision Tree algorithm that is composed of a collection of other tree classifiers and that prevents the statistical problem of overfitting to the training data which occurs in other Decision Tree algorithms. According to its creators, this algorithm offers the best accuracy between existing algorithms. Among other advantages this method can handle missing values, large data sets and thousands of attributes which it can be very efficient in terms of speed. [40]

Random Tree

As supervised Classifier Random Tree is a learning algorithm that generates other individual learners and all of them make up the group of tree predictors which is called forest. The classification begins with the input feature to be classified by each tree that belongs to the forest and is labeled according to the majority of the classified classes which results in final class. Theoretically Random Trees are a combination of single tree algorithm and Random Forest.

2.2.4 TF-IDF Weighting

TF-IDF is a short term and stands for frequency–inverse document frequency is a numerical statistic which aims to measure the importance of a word in a document or a collection. The importance is based on how often a word occurs in a document or in a document collection but it is also associated with the frequency of the word in this collection. This technique is usually used as a weighting factor in data mining and information retrieval. Specifically TF-iDF is one of the most famous weighting schemes and 83% of text-based recommender systems in the domain of digital libraries use TF-iDF.[2] Stopwords removal is also possible since these words are not essential and should be removed to keep up the actual useful information. Another usage of TF-iDF can be in text summarization and in classification.

Term Frequency - TF

As mentioned previously, term frequency measures the occurrences of a word in a document regardless of its length but since documents with a long length may exist, we expect more word occurrences on such documents. For normalization purposes this term is divided by the total number of terms in the document and completes the type:

Figure 2.9: TF weighting equation

$$\text{TF weight (of the term in the document)} = \frac{\text{Number of times the term appears in the document}}{\text{Total number of terms in the document}}$$

Inverse document frequency - IDF

Inverse document frequency (iDF) is a measure that calculates the importance of a term. Regarding this, idf formula does not consider important certain words, such as “and”, “is”, “for” etc. since they are used countless times in a document or in a corpus and a weighting scheme in logarithmic scale is used in order to track down the rare or the meaningful terms with the following equation.

Figure 2.10: IDF weighting equation

$$\text{IDF weight (of the term in the documents)} = \frac{\text{Total number of documents}}{\text{Number of documents in a corpus that cointain the term t}} = \log_{10}\left(\frac{N}{n(t)}\right)$$

Complete Formula

The combination of the above two equations produce the final weighting equation that is used to measure the words. In other words, TF-iDF assigns to the term t in document d, a high weight value when the term appears many times in a few documents, a low weight value when the term appears a few times in one or many a documents, the lowest weight value when the term appears in almost every document. For terms that do not appear in a document a zero weight value is given. TF-iDF consists of two mathematical terms and results in the formula:

Figure 2.11: TF-iDF equation

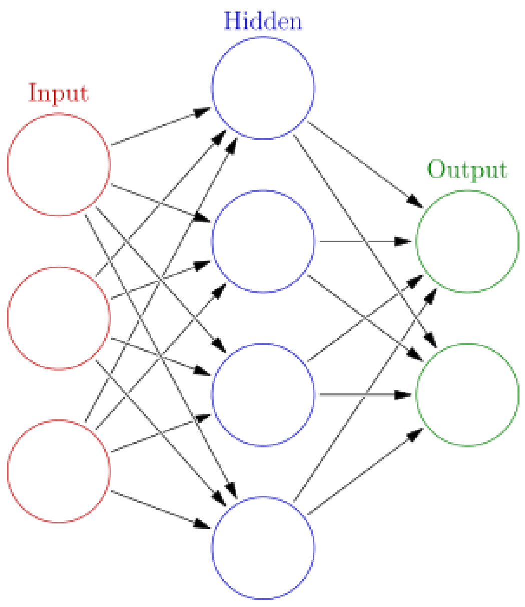
$$w_{t,d} = (1 + \log \mathrm{tf}_{t,d}) \times \log N / \mathrm{df}_t$$

Note: TF is also scaled logarithmically for distinguished reasons because if the frequency of the term in a document is 1, then the log(1) is zero.

2.2.5 Neural Networks

Neural Network refers to a circuit of interconnected neurons and is based on the function of biological Neural Networks through which humans have the ability to remember and solve problems. The human brain is quite complicated and consists of many neurons that accomplish complex calculations and operations with lightning speed.
In the field of computational neuroscience, artificial Neural Networks are modeled based on the functions of the human brain. In fact, Neural Networks are information processing systems comprised of a graph and various algorithms that access this graph. Each neuron can receive a signal, process it and then signal artificial neurons connected to it. Usually the signal transmitted between neurons has a numerical value where the output of each artificial neuron is calculated by a non-linear function from the sum of its inputs. Connection weighting values on an artificial Neural Network determine the functionality of the learning process, thus the weight value can increase or decrease the strength of a signal at a connection.

Figure 2.12: Neural Network



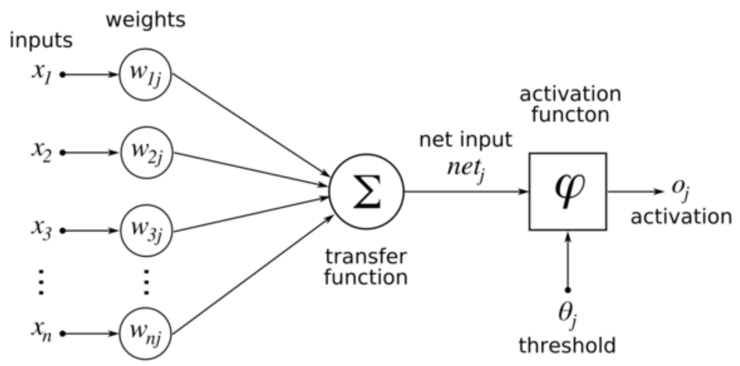
An artificial network can be considered as a directed graph which is organized into a group of 3 types of neurons:

- Input nodes
- Hidden nodes
- Output nodes

Artificial Neuron

The artificial neuron is a computational model, the parts of which try to imitate a biological neuron, and it is the basic processing unit of an artificial Neural Network. The figure below illustrates the model of such a neuron.

Figure 2.13: Artificial Neuron



As we can see from the figure, an artificial neuron receives input signals(Xi) that can vary from a weight value(Wi) where it can be positive or negative. The body of the artificial neuron is divided into two parts: the transfer function and the activation function. The transfer function adds the input signals that have been altered by their weights and produces Σ. After that the activation function apply a filter(φ) that modulates the final value(Oj) in relation to the threshold value(θj). Apart from the input signals and weights, the neuron has also a weight value which is called bias. The difference of this weight compared to others is that its value is always 1 and if the total sum of the rest of the inputs is greater than bias, then the neuron is activated. Otherwise the neuron remains deactivated. This produced by the regular biological neuron. The ultimate aim of an operating network should be characterized by a general capability: to give good outputs for unprecedented inputs and different from those with which they were trained.

The main characteristic of artificial Neural Networks is inherent learning ability, as learning can be defined as the gradual improvement of the network's ability to solve a problem. Learning is achieved through education, a repetitive process of gradually adjusting weights and bias at values suitable to successfully solving the problem. Once a network is trained these parameters freeze at the appropriate values and a functional state is reached.

There are three major learning categories each one for a different learning task.

Supervised learning

In this category, input and desired output pairs are imported to the system and in this way, an output which is different from the desired one is produced. This results in the calculation of the error and the adjustment of the weights in order to achieve better output. Supervised learning paradigms are also capable of making decisions about when the training process should be stopped, decisions on network presentation frequency, education standards, and network progress. Classification and regression are also parts of supervised learning in Neural Networks as we have mentioned in other approaches.

Unsupervised learning
Unsupervised learning algorithms are self-organized and do not require any training data set. These algorithms organize the data with various criteria and discover the important similarities between the data through the learning process. Applications of this learning include compression, clustering, filtering and the estimation of statistical distributions.

Reinforcement learning
In reinforcement learning, the training of an input-output display is performed through continuous interactions of an agent with the environment. The agent and the environment are constantly interacting with the first choosing the actions and the second reacting to them and presenting them with new states. The environment gives the agent rewards, which are ultimately intended to maximize them.

An artificial Neural Network may fail to successfully model training data and lead to incomplete learning, this is called underfitting. On the other hand, a very complex artificial Neural Network can model the training data along with the noise that they probably exists. In this case the ANN, predicts correctly all the training data but fails in any other data. The best way to overcome this, is to give the system adequate training data. Thus, for ANN with hidden layers used in classification problems with training data containing noise, it is advisable to have at least 30 times more training data than the number of network weights. Training is completed when the network quality control criterion reaches a desired value. As a test criterion, the average error of the training set is usually used.

2.3 Music Classification

Music classification is an interesting topic with many potential applications and important functionalities but it can also be problematic when it comes to labeling songs according their genre, artist, mood etc. One reason this happens is because the users might be interested only in certain music types or they are motivated by meaning or tempo of the song. Machine learning techniques are commonly used when it comes to genre categorization although the classification might sometimes be considered as subjective. Other applications aim to classify the artist or the mood of the song.

As we have seen [8], Music Classification is divided in the following tasks:

- Genre Classification
- Mood Classification
- Artist Identification
- Instrument Recognition
- Music Annotation

From now on we will deal only with the Mood Classification task. There are numerous studies that have been conducted about automatic mood classification. In the audio processing, techniques have been performed such as extracting acoustic features from audio files, sound analysis in terms of rhythm, pitch harmony or timbre and temporal examination. However it would be unreliable to say that audio classification is enough to justify the mood of a song. Mood classification based on lyrics is a way to delve even deeper in MIR research, which is reasonable since the reasons why people listen to music might differ. According to Juslin and Laukka (2004) [17], 29% of people mention that lyrics are an important factor of how music expresses emotions. In addition, Besson et al. (1998) have shown that part of the semantic information of songs resides exclusively in the lyrics. [3]

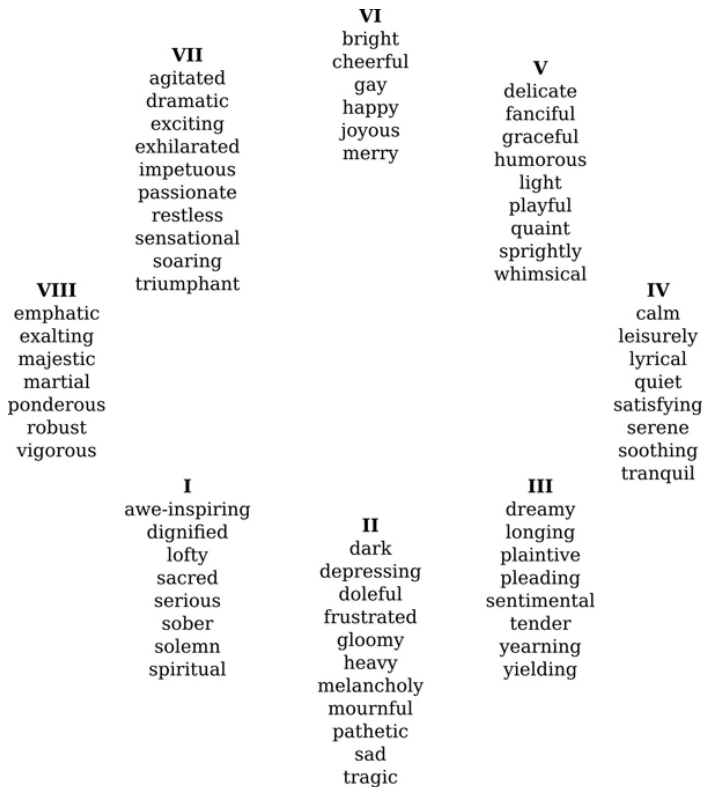
2.4 Emotion Models

There are several theoretical models according to psychologists and these models can be divided into two emotion models: categorical and dimensional.

2.4.1 Categorical Approach

The basic concept of categorical model is that people experience emotions as categories that are distinct from each other [Yang and Chen, 2012][42]. The main idea of basic emotions approach is that there is a finited number of universal and primary emotions classes such as happiness, sadness, anger, fear, disgust, and surprise, from which all other secondary emotion classes can be derived [12] [24]. These basic emotions are all around the world, regardless of different cultures and are often connected with the psychology or the emotional state of humans. However, this group of emotions has been criticized for a number of reasons, but most of all because, over time, several researchers have come up with different sets of emotions [18]. Another widely known emotion

Figure 2.14: Hevner's eight clusters [1935]

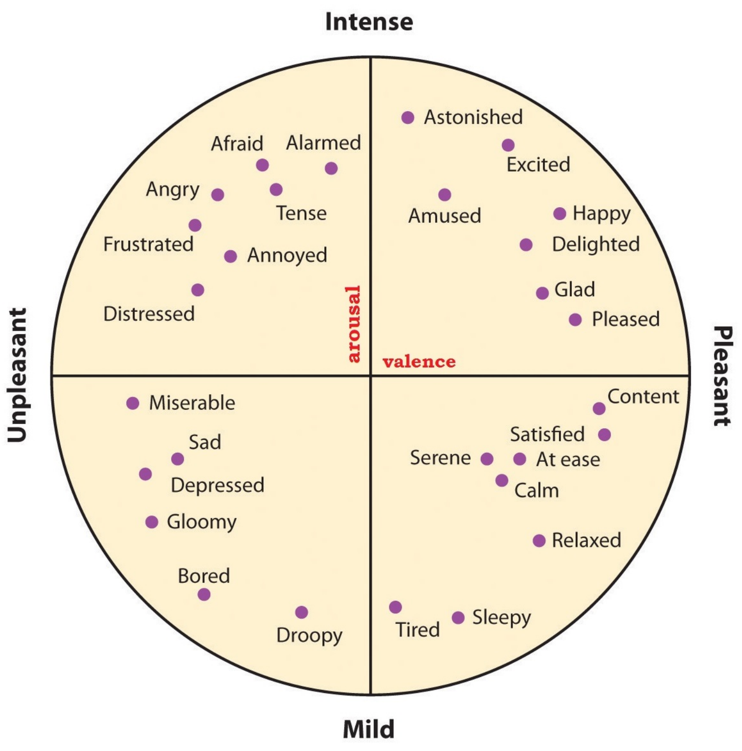


model is Hevner's adjective circle[14]. Through her years of research into musical psychology, she has shown that music is directly linked to feelings. As a result, she based her idea on clusters of adjectives (emotions), instead of using single words. She categorized these emotions into 8 groups in a cyclical way and because of the many adjectives, there were groups that contained words with the same meaning which corresponded to the same emotional state [figure 2.14]. Later Hevner's adjective clusters were regrouped and redefined into ten groups by Farnsworth (1954) and into nine adjective groups by Schubert in 2003 [29]. Nevertheless the limited number of adjectives forced the researchers to explore other emotion models.

2.4.2 Dimensional Approach

The dimensional model focuses on identifying all emotions according to where they have been placed on three-dimensional or two-dimensional axes which corresponds to internal human representations of emotions. The most famous in this emotion model category is Russel's dimensional model [26] and is commonly used in research of Music Emotion Recognition (MER). According to Russell's proposal, three main dimensions can exist as valence, arousal and potency where they respectively describe positive and negative affective states, energy and stimulation level, and the sense of control or freedom to act but in practice, for the sake of simplicity, only the two axes of this dimensional model(valence and arousal axes) are used in the most MER projects. Discrete and continuous are the two grouped categories of dimensional model. In discrete models we have 4 distinct quadrants that are distributed in the Cartesian plane and every area express different group

Figure 2.15: Russell's 2-Dimensional Axes



of adjectives emotions as we saw in Russell's statement [figure 2.15]. On the other hand, continuous models treat emotions as a continuum, and so each point on the level can represent a different emotion. As a result, continuous models have been criticized as uncertain and unstable since no subjective tags are used [42].

The main advantage of the dimensional approach and specifically the discrete model offers a simple and descriptive way to organize different emotions, concerning their affect appraisals (valence) and physiological reactions (arousal) and we can easily evaluate the significance of these two emotional axes. But as the third axis is not taken into account, this leads to failed psychological distinctions and interferes with important aspects of the emotion process, thus forming an important drawback of this concept. Fear and anger are placed close to each other in the arousal-valence plane but in actual fact they have different meaning. Hence, excluding this dimension might lead to ambiguous emotion characterization [42], as illustrated in the example of the figure.

2.5 Music Databases

As mentioned earlier, with the spread of the Internet in particular during the last decade, music has also developed rapidly in the field of information technology. Plenty of websites that provide video services like Youtube, or Spotify that provides audio streaming services, are integral in our everyday life. As a result, the amount of digital media data has increased significantly, therefore the need for music databases was necessary. There are numerous music databases which contain music data information and almost every detail regarding audio and lyrics features. We will outline a few of these in a brief review and we will mention the databases we used as sources, to construct the dataset for our ground truth.

2.5.1 Last.fm

Last.fm[6] is a music community website which uses recommendation system in order to suggest songs and music types according to user's musical taste. The algorithm records details of the tracks that users listens to via plugins or from third party music players, among others YouTube and Spotify. With the latter being selected as the primary music streaming source whereas YouTube's streaming capability is also provided as secondary option. Nonetheless the collaboration of both of them offers users a more extensive search for better service. For example, if Spotify is unable to find an artist or song in its library, it will automatically turn to YouTube for further search. Last.fm also provides an API so everyone can request data from their music collection in order to build their own programs. The data provided concerns information about artists, songs, albums, tags and even a combination of these. It also provides information on a social level such as user's favorite songs or artists, details of user's profile etc.

2.5.2 Spotify

Spotify[11] is a music and video streaming service with active users that exceeds 100 million in total. Music can be shared on social media as playlists or can be searched for regarding artists, tracks, albums, genre, playlist or record label and as for premium users, they can have improved streaming quality and offline music downloads. As a visitor you can connect directly even from your Facebook account or sign-up for free with a few simple steps. In this way this service is more attractive to those interested. For the developers perspective Spotify provides an API based on REST principles with which you can fetch data in JSON format about artists, tracks, albums, users profiling info and even more advanced information about the audio of the tracks like valence, loudness, tempo, energy, danceability and more.

2.5.3 MusicBrainz

MusicBrainz[13] is a community-maintained open source encyclopedia of music information. Their open database contains information about artists, their recorded works, and the relationships between them. Recorded works are concerned with the albums, tracks, length of each track, release date, cover art, acoustic fingerprint and many other metadata all submitted by the community according to specific style guidelines. It also provides free access through a well documented web API to distribute the stored data under public licenses.

2.5.4 MusixMatch

The title of the world's largest lyrics platform belongs to MusixMatch [5]. Users can search and retrieve lyrics simply and quickly. This service is used on various devices and it can also display lyrics while the music is being played. It can be implemented even on music streaming services like Spotify that we described previously. MusixMatch recently expanded their research in the Artificial Intelligence field and they provide to companies and researchers a large-scale lyrics dataset designed for machine learning applications and for linguistic analysis in order to detect and interpret emotions via lyrics. To have access to their API, users can sign up for free to get a API key but only 30% of the requested lyrics are returned to avoid illegal commercial uses. Researchers can request full access to the lyrics for non-commercial use but the returned data will be in bag-of-words model. We used this service on our project in order to retrieve lyrics for our songs that we used for our dataset as we are going to see in chapter 3.

2.5.5 openAudio

OpenAudio is a collection of content and software that is used in areas of music information retrieval such as audio recognition tasks or speech emotion recognition. Several researches also have been conducted around the automatic mood classification topic in order to create a way to automatically classify mood through music. Specifically one of the experiments asked four people (ages between 23 and 34 years)to listen to 2648 songs NTWICM corpus of popular UK chart music and rate accordingly the emotions that they perceived. These four users rated all the emotions according to the two-dimensional axes, where one axis describes the arousal and the other the valence. The assigned values where in range of $-2, -1, 0, 1, 2$ for both axes respectively. This has helped researchers to conduct a series of experiments to create a system for automatic music mood prediction based on musical features and lyrics. [30]

2.6 Tools

2.6.1 Jersey REST and Javax

Restful web services, which are based on HTTP methods, is a way of server-client communication and in order to simplify this, a standard API has been created in Java which is called JAX-RS. Jersey RESTful web service[16] is an open source framework, an implementation of JAX-RS and is used for developing web services in java. It has its own API and extends JAX-RS toolkit with additional features and also provides libraries to integrate with other frameworks like Spring.

2.6.2 JSON Parser

REST protocol allows the user to have different types of representation for resources like XML, JSON, text etc. and can request each one via HTTP protocol. JSON(JavaScript Object Notation)[37] is currently the most popular way for data exchange on the web -since it is lightweight-but to transfer the data a conversion of java objects to JSON format must be done. To achieve this, Google created gson, an open source parser which is also capable of reversing the conversion process and also converts from JSON string to Java objects. There are two basic data structures in JSON, objects and arrays. An object is an unordered collection of key and value pairs separated by commas. Keys must be strings and values must be a valid JSON data type such as object, number, array, string, boolean or null.

2.6.3 WEKA Library

Weka[10] is an open source suite-tool for data prediction, analysis and mining tasks. It is written in Java and can be used through command line, or via API with Java but also provides its own user interface for data prediction, clustering, regression, pre-processing, association rules, visualization, attribute selection and classification. Machine learning techniques are the main part of Weka and the prediction of them is to be based on the assumption that the data is given as a flat file or relation, where each point of data is represented by a fixed number of attributes (numeric, nominal, normally and other types). Pre-processing is about filtering the data that has been imported via a file such a database file or comma, separated-values (CSV). Classify allows to apply classification and regression algorithms on the filtered data in order to estimate the accuracy of the predictive model and visualize predictions or the model itself (e.g. Decision Tree). The role of the association rule learners is to detect all the major interrelationships among the attributes in the data, while clustering is the task of grouping all the objects that belong to the same group after analyzing the data. Attribute selection algorithms identify the most predictive attribute in a dataset. The identification of the most predictive attributes in a dataset is a task that attribute selection algorithms are responsible for and visualization provides scatter plots and bar graphs output in 2D format for further enlargement and analysis.

Chapter 3

Methodology

For our study we used various supervised learning algorithms to run our tests. K-Nearest Neighbors, Naive Bayes and Decision Tree algorithms such as Random Forest. For this purpose we also used Weka library tool, Java programming language and a few other libraries that helped us to make it completed. However, we have previously investigated similar researches around the music information retrieval field.

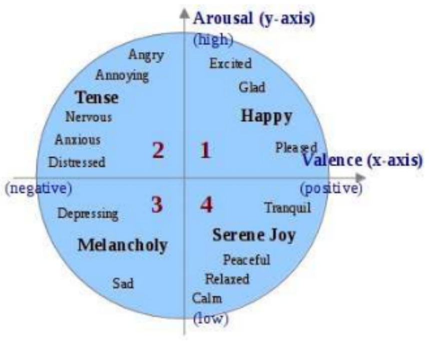
3.1 State of the Art

Music psychology has existed as a field for many decades and despite its small audience, there is plenty of research carried out in the MIR(Music Information Retrieval)/Music Data Mining(MDM) field. Scientists have covered a wide range of research on emotion extraction through music or melodies based on the analysis of lyrics, rhythm, tempo and many other factors. It was important for us to see what already exists in this area so that our own research could benefit from it, both in the construction of the theoretical level and error prevention, as well as implementation level.

3.1.1 Bi-Modal Music Emotion Recognition: Novel Lyrical Features and Dataset

In Center for Informatics and Systems of the University of Coimbra exist a large group of researchers organized into 6 groups with the most relative to MIR to be the Cognitive and Media Systems group. This research group has published quite a few papers regarding music emotion recognition, automatic mood detection and music classification. This paper aims to find the best possible models for audio and lyrics dimensions in a context of emotion recognition, using Russell's model where emotions are described in two-axes, valence and arousal. The research is based on the quadrants that you see in figure 3.1 and it is considered that a sentence belongs to quadrant 1 if both dimensions are positive; quadrant 2 if V is smaller than 0 and A is bigger than 0; quadrant 3 if both dimensions are negative and quadrant 4 if V is bigger than 0 and A is smaller than 0. To achieve the ultimate goal a dataset was constructed manually from the lyrics and the audio of 200 songs by several music genres. Firstly, the annotators gave their attention to lyrics to measure the impact of them on emotions in order to create the lyrics dataset and secondly, to the audio of the tracks for the creation of the audio dataset. Then both dimensions were fused and a bimodal analysis was performed.

Figure 3.1: Russell's circumspect model



One experiment conducted based on classification by quadrants with Vector Space model algorithm. In addition classifiers for both lyrics and audio were created and feature selection was applied. The results were surprising since lyrics-based model achieved better performance than the best audio-based model (79.3% vs 72.6%) but both dimensions are important since bimodal analysis improves significantly the results of the lyrics classifier (from 79.3% to 88.4%). This led to a conclusion that unlike most other researches lyrics performed better than audio and also that bimodal analysis is always better than each category separately. [15] [21]

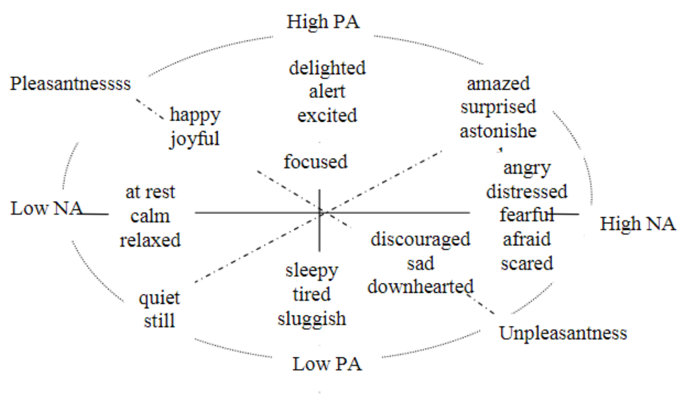
3.1.2 Music Emotion Identification from Lyrics

The main objective of Dan Yang's and Won-Sook Lee's work [41] was to extract specific emotions from the lyrical text of songs through machine learning techniques instead of human experts. For that purpose the database that was used was Allmusic.com which offers browsing by musical emotion that users have already classified thousands of songs into 183 moods. For this research a psychological model of emotions[31] is used (fig. 3.2) that describes 23 specific emotions and is used to classify 168 different music moods that represents over 10.000 songs of this database. As training set, 1032 songs have been randomly selected and each one was transformed into a feature vector of 182 psychological features using a content analysis package. Text features mostly related to the chosen emotions were selected and for classification purposes, Decision Trees and rule methods were used. To deliver the best accuracy DECORATE(Diverse Ensemble Creation by Oppositional Relabeling of Artificial Training Examples) algorithm of WEKA was used which achieved accuracy of 67% by 10-fold cross-validation which results in a 23-class classification of 1032 songs with almost 5000 words, similar success to other research approaches like 2-class classification of 22000 sentences, but the main advantage was the human comprehensibility of the generated classification model.

3.1.3 A Mood-Based Music Classification and Exploration System

In his MSc thesis Meyes[22] tried to develop a tool that would automatically generate a playlist that is based on user's mood or activity using mined information from songs with context-aware data like song lyrics. Audio analysis, lyrics processing and natural language tools are used to extract useful information through harmony, tempo, rhythm, loudness and more. The evaluation of the system performance was done using 372 songs from the database which have been applied to a Decision Tree algorithm and secondly k-NN algorithm in order to attach an emotion to each song. The results were evaluated by 3 different factors, experts evaluation, from All Music Guide and

Figure 3.2: Emotion Model of Watson, Tellegen and Clark



other projects like Pandora Internet Radio using social tagging networks like Last.fm and other. Despite the great influence on performance that caused the problems like the feature extraction stage, misclassifications in some features like tempo or mode but also in lyrical analysis, the results were satisfying with the system being able to identify correctly a good number of songs. However, the lyrics played a decisive role in the correct mood identification.

3.2 Goal

The aim of this research is the development of a program which extracts the emotional state of a song, based on its lyrics alone. There are various emotions hidden in songs and if we consider the different kinds of music contents over the decades across genres, artists, religions, political influence and more, we are referring to million of songs and so dozens of mood emotions. Especially when music databases like Last.fm (see 2.5.1) use tag attributes to describe different kinds of mood, it makes researches in this area more effective specific. In addition, more and more music streaming services are offering music depending on the user's mood as a separate section, and this creates the need for even more research on weather emotions can be described with song lyrics. The main motivation of this project, is to discover if the lyrics are the main source of emotions or just a helpful measure in the estimation of the emotional state of songs.

3.3 Ground Truth Construction

In order to see if the aim is achievable we used a dataset with the minimum available online resources. Our training dataset was eventually re-constructed and it was tested with different classification methods but with the same preprocessing procedure. The training data was cleaned, stemmed and filtered properly and in combination with the single and dual classifiers which were based on different machine learning algorithms, led us to our experimental results. At the end of the project we compare our obtained results with the original obtained dataset since it was validly annotated by humans. But first, let's fully analyze the preprocessing procedure in order to look deeper in to the dataset we created.

3.3.1 Raw Data Preprocessing

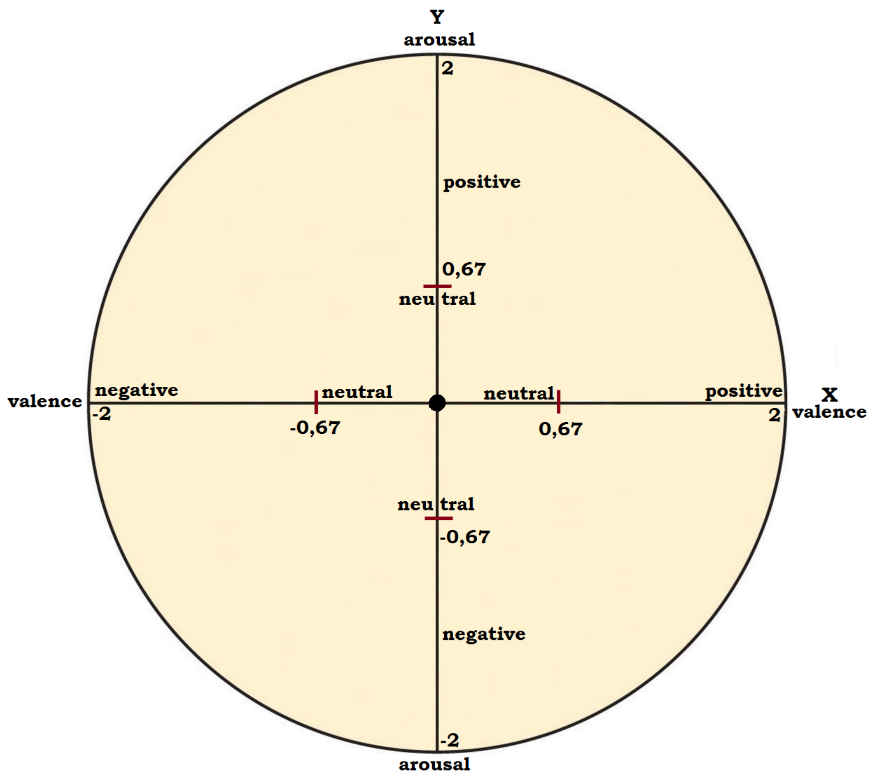
Concerning both the proper process of research and its effectiveness, we needed to rely on a valid dataset, so for this purpose, we chose the annotations from openAudio whose composition was described early in subsection 2.5.5. These annotated songs are already predefined in mood categories by four raters based on an extension of Henver's mood clusters, with integer range values of [-2, 2]. For simplicity reasons of our approach, we calculated the mean value for both arousal and valence with two thresholds of 0.67 (isometrically equal) and 0.33 (balanced classes).

0.67 - Threshold: We categorized all songs into 3 isometrically-equal-dimensions categories.

1. Positive Category: If the mean value of arousal or valence is over 0.67 in each axis respectively.
2. Negative Category: If the mean value of arousal or valence is under -0.67 in each axis respectively.
3. Neutral Category: If the mean value of arousal or valence is in the range of (-0.67, 0.67) in each axis respectively.

So in the end we have a class value for our valence-axis and a class value for our arousal-axis which we consider as axis X and axis Y respectively for our purpose as illustrated in the following image.

Figure 3.3: Our 3 Isometric Categories

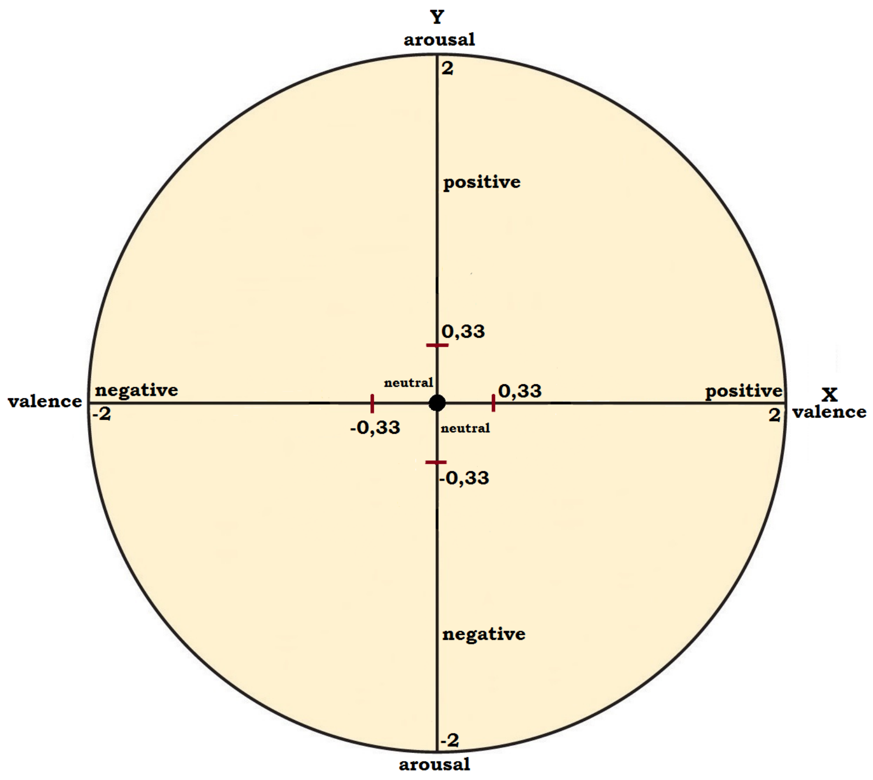


0.33 - Threshold The value of this threshold seems to produce the most balanced categories in training data compared to any other thresholding value.

1. Positive Category: If the mean value of arousal or valence is over 0.33 in each axis respectively.
2. Negative Category: If the mean value of arousal or valence is under -0.33 in each axis respectively.
3. Neutral Category: If the mean value of arousal or valence is in the range of (-0.33, 0.33) in each axis respectively.

Hence, we have for our X axis the class value for valence and for our Y axis the class value for the arousal as we see in the figure below.

Figure 3.4: Our axes with threshold of 0.33



In consideration of the original file that provided us with all the ratings of the users, it also provided us with the actual song and artist names. That helped us to search for the lyrics through online music databases via available APIs and we excluded songs that we were not able to track down as well as instrumental-only songs or songs that were not written in the English language. As this as completed we had finally collected 2604 lyrics of songs and that was only the first step of our data pre-processing process.
For the second important step before we started building our classifier, it was time to clean those lyrics in order to improve their quality. To do that, we eliminated the following:

- All symbols
- Non-letter characters
- Punctuation marks
- Numbers
- Duplicate whitespaces
- Stopwords
- Common patterns in lyrics that have no essential use such as [Chorus x2], [Intro], [Verse 1] etc.

In the remaining lyrics of each song we applied stemming method of Porter and at last, we had the useful words of lyrics and the class values for both axes(valence-arousal) or in other words, our training dataset. We converted all the data in format of arff file in order to be able to be recognized by WEKA's Java libraries.

3.3.2 Building Classifier

Since we had our training dataset already completed, it was time to start building our classifier. The procedure was simple since WEKA libraries carried out the building process. We initially started by creating our 2604 instances and next we proceeded to the application of the filter. We used the StringToWordVector filter in order to convert each word into numeric attributes that represent word occurrences and, as for our classifiers, we chose one of NaiveBayes, DecisionTrees(J48) and k-NearestNeighbor algorithms per time.

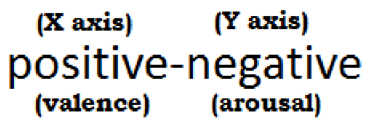
Chapter 4

Approaches

In this chapter we will implement all algorithm methods that we used, we will describe and mention the accuracy percentage for each one but first it is important to say that we divided the classifiers into two stages. Both stages produce the same output but their construction process is different.

First Stage: **Single Classifier**
As stated in the previous chapter and specifically at subsection 3.3.1 we categorized all songs into 3 classes for each axis. Single classifier classifies valence and arousal as a pair which means that combines the class of valence and the class of arousal into one step classification for both axes(X and Y) and the output is in form of: (class for X axis)-(class for Y axis) as we see in the example below.

Figure 4.1: Output example of Single Classifier



The classifier is being fed with an arff file containing all the training instances that have already predefined classes as the example above and, as a result, we have 9 available combined classes for our single classifier since valence and arousal are described as a pair in one class.

Second Stage: **Dual Classifiers**
Dual classifiers -as its name may explain- do two individual classification processes, one regarding axis X (valence) and another one regarding axis Y (arousal). These two classifications neither related nor affected by each other. Each classifier (axis) has 3 available classes to assign.

The class values for valence are hypothetically placed in the axis X like the image represents below:

While we assume the class values for arousal are placed in the Y axis like the following figure:

Adopting the logic behind the Russel's model (figure 2.15), we see below that by combining

Figure 4.2: Dual Classifier: Valence Classification

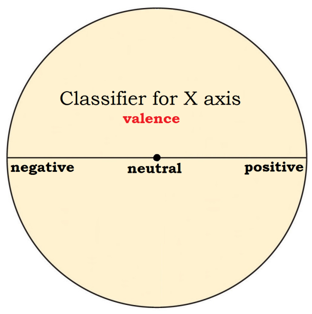
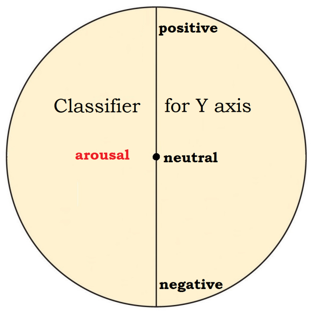
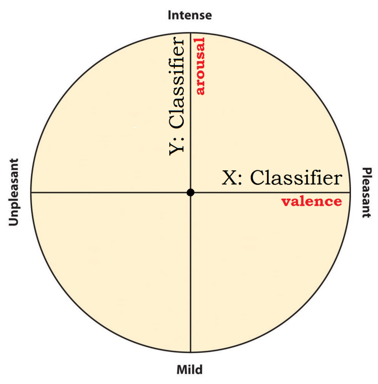


Figure 4.3: Dual Classifier: Arousal Classification



the two previous axes we basically have a quadrant representing both classifications:

Figure 4.4: Dual Classifier: Valence & Arousal Classification



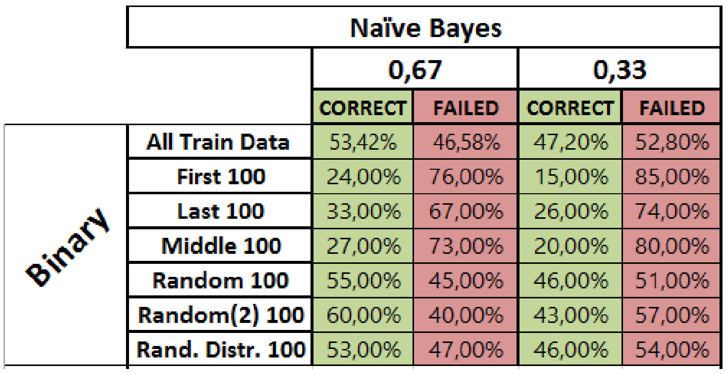
All measures have been tested through the training data set itself and rest data sets of the first, last, middle, 100 random instances, another 100 random instances and finally 100 random instances that are based on the distribution of the training data. All test data have been excluded from the training data set before the building of each classifier.

4.1 Binary Conversion with Single Classifier

Our first approach was binary conversion of string data into two numeric values, zero(0) and one (1) thanks to StringToWordVector filter. Like a matrix table, each instance contains binary values for each word of the dictionary, where value of 1 for words that are included in the instance while value of 0 for words which are not. Starting with this conversion we will see how Naive Bayes, J48 and kNN perform with the single classifier and then with the dual classifiers.

4.1.1 Binary and Naive Bayes

Figure 4.5: Single Classifier with Binary Conversion and Naive Bayes

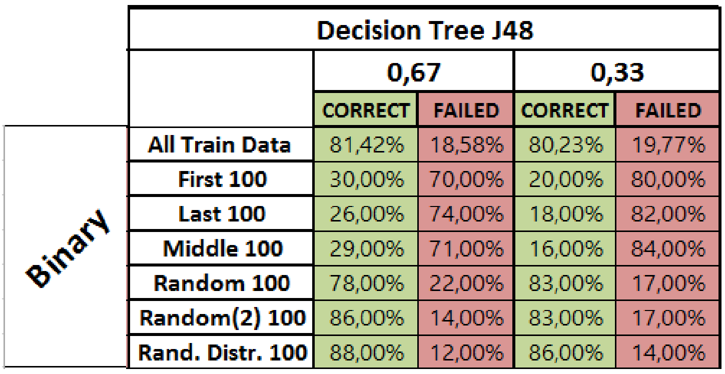


	Naïve Bayes				
		0,67		0,33	
		CORRECT	FAILED	CORRECT	FAILED
Binary	All Train Data	53,42%	46,58%	47,20%	52,80%
	First 100	24,00%	76,00%	15,00%	85,00%
	Last 100	33,00%	67,00%	26,00%	74,00%
	Middle 100	27,00%	73,00%	20,00%	80,00%
	Random 100	55,00%	45,00%	46,00%	51,00%
	Random(2) 100	60,00%	40,00%	43,00%	57,00%
	Rand. Distr. 100	53,00%	47,00%	46,00%	54,00%

Our first implemented algorithm was Naive Bayes with binary conversion on single classifier. The highest score noted in the second random 100 data while we see that the test data of non-random runs are almost the half of the randoms.

4.1.2 Binary and Decision Tree J48

Figure 4.6: Single Classifier with Binary Conversion and Decision Tree J48



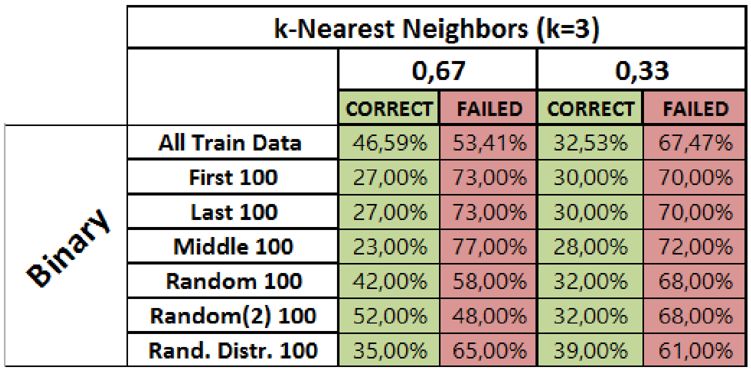
	Decision Tree J48				
		0,67		0,33	
		CORRECT	FAILED	CORRECT	FAILED
Binary	All Train Data	81,42%	18,58%	80,23%	19,77%
	First 100	30,00%	70,00%	20,00%	80,00%
	Last 100	26,00%	74,00%	18,00%	82,00%
	Middle 100	29,00%	71,00%	16,00%	84,00%
	Random 100	78,00%	22,00%	83,00%	17,00%
	Random(2) 100	86,00%	14,00%	83,00%	17,00%
	Rand. Distr. 100	88,00%	12,00%	86,00%	14,00%

Moving to the second binary conversion that took place, Decision Tree had really higher percentages compared to the Naive Bayes in each threshold.

4.1.3 Binary and k-Nearest Neighbors (3)

The results of k-Nearest Neighbors with k value 3, brought us back to square one in terms of accuracy rates, since binary conversion does not seem to work well in kNN.

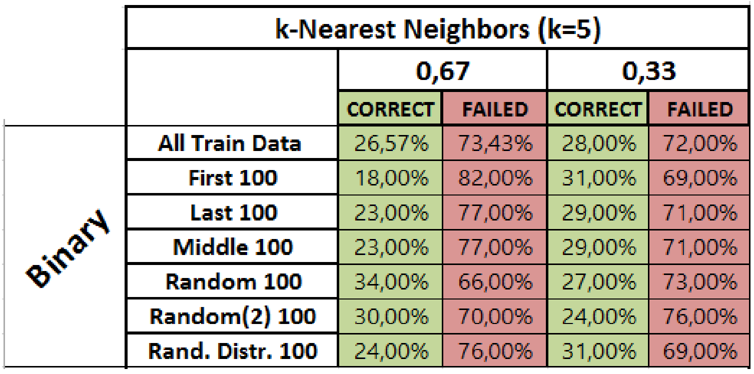
Figure 4.7: Single Classifier with Binary Conversion and k-Nearest Neighbors (3)



	k-Nearest Neighbors (k=3)				
		0,67		0,33	
		CORRECT	FAILED	CORRECT	FAILED
Binary	All Train Data	46,59%	53,41%	32,53%	67,47%
	First 100	27,00%	73,00%	30,00%	70,00%
	Last 100	27,00%	73,00%	30,00%	70,00%
	Middle 100	23,00%	77,00%	28,00%	72,00%
	Random 100	42,00%	58,00%	32,00%	68,00%
	Random(2) 100	52,00%	48,00%	32,00%	68,00%
	Rand. Distr. 100	35,00%	65,00%	39,00%	61,00%

4.1.4 Binary and k-Nearest Neighbors (5)

Figure 4.8: Single Classifier with Binary Conversion and k-Nearest Neighbors (5)



	k-Nearest Neighbors (k=5)				
		0,67		0,33	
		CORRECT	FAILED	CORRECT	FAILED
Binary	All Train Data	26,57%	73,43%	28,00%	72,00%
	First 100	18,00%	82,00%	31,00%	69,00%
	Last 100	23,00%	77,00%	29,00%	71,00%
	Middle 100	23,00%	77,00%	29,00%	71,00%
	Random 100	34,00%	66,00%	27,00%	73,00%
	Random(2) 100	30,00%	70,00%	24,00%	76,00%
	Rand. Distr. 100	24,00%	76,00%	31,00%	69,00%

K-Nearest Neighbors with k value 5 was the last in binary conversion for the single classifier with really low results in all runs.

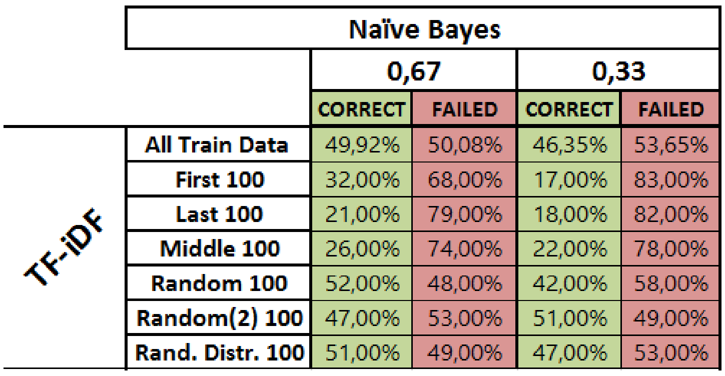
At this point we started noticing that the percentages of the test files of first, last and middle 100 data continued to hit low accuracy percentages in any algorithm. So we carefully observed them as we proceeded forward in case there are no more meaningful in the following approaches.

4.2 TF - iDF Conversion with Single Classifier

Proceeding to the next conversion of our project, TF-iDF selected since binary conversion is only a two dimensional model and it does not take into account the importance of words. Now our vectors contain weight scores for each word in each instance, based on the TF-iDF equation as described in the subsection 2.2.4. Instead of 0 and 1 values as the binary conversion assigns to vectors, this model calculates the importance of a word and, as a result, each word is described by a real number value. With this, each word is treated differently based on its importance, instead of all of them equally as the binary bag-of-words model does. For example, a word which occurs in many documents is less important than a word which occurs in just a few documents. The latter will have weight value in the scores of each vector. Hence, we expected better accuracy percentages with this type of data conversion.

4.2.1 TF-iDF and Naive Bayes

Figure 4.9: Single Classifier with TF-iDF Conversion and Naive Bayes

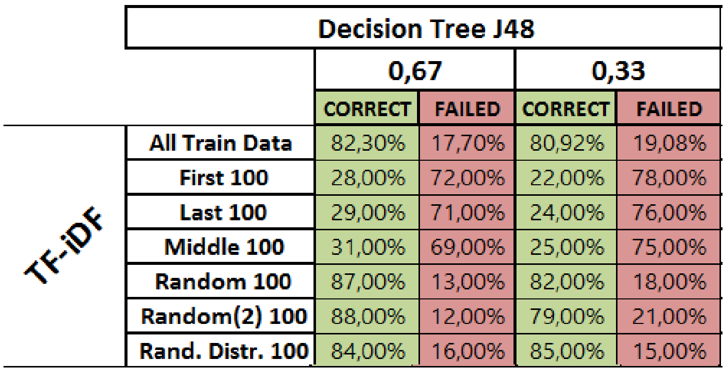


		Naïve Bayes			
		0,67		0,33	
		CORRECT	FAILED	CORRECT	FAILED
TF-iDF	All Train Data	49,92%	50,08%	46,35%	53,65%
	First 100	32,00%	68,00%	17,00%	83,00%
	Last 100	21,00%	79,00%	18,00%	82,00%
	Middle 100	26,00%	74,00%	22,00%	78,00%
	Random 100	52,00%	48,00%	42,00%	58,00%
	Random(2) 100	47,00%	53,00%	51,00%	49,00%
	Rand. Distr. 100	51,00%	49,00%	47,00%	53,00%

Firstly, we began with Naive Bayes which did not produce impressive results. In fact, the results were worse compared to the binary bag-of-words model but that it may be connected with the theoretical background of the algorithm instead of our data. However, our first, last and middle data still remain in low rates compared to other data runs.

4.2.2 TF-iDF and Decision Tree J48

Figure 4.10: Single Classifier with TF-iDF Conversion and Decision Tree J48



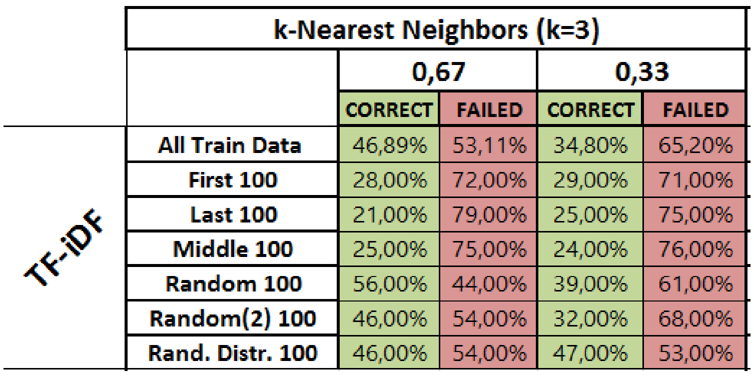
		Decision Tree J48			
		0,67		0,33	
		CORRECT	FAILED	CORRECT	FAILED
TF-iDF	All Train Data	82,30%	17,70%	80,92%	19,08%
	First 100	28,00%	72,00%	22,00%	78,00%
	Last 100	29,00%	71,00%	24,00%	76,00%
	Middle 100	31,00%	69,00%	25,00%	75,00%
	Random 100	87,00%	13,00%	82,00%	18,00%
	Random(2) 100	88,00%	12,00%	79,00%	21,00%
	Rand. Distr. 100	84,00%	16,00%	85,00%	15,00%

Next, Decision Tree (J48) algorithm has been used after our data was converted into real numeric values with TF-iDF equation. Based on the results, we saw a slight improvement overall. Train data rates have been increased up to 82,30% and 80,92% for 0,67 threshold and 0,33 threshold respectively. This percentage rate gives us the indication that the algorithm does not overfit our data since the random tests are performed well enough as the accuracy percentages show.

4.2.3 TF-iDF and k-Nearest Neighbors (3)

K-Nearest Neighbor with k to be 3, had almost the same results with the failure rate above 55% in all tested data.

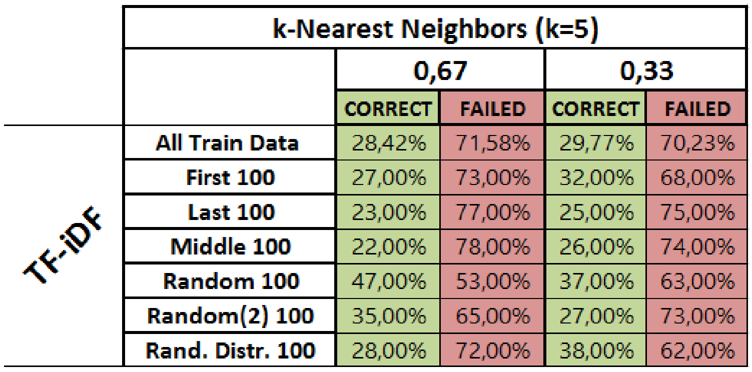
Figure 4.11: Single Classifier with TF-iDF Conversion and k-Nearest Neighbors (3)



	k-Nearest Neighbors (k=3)				
		0,67		0,33	
		CORRECT	FAILED	CORRECT	FAILED
TF-iDF	All Train Data	46,89%	53,11%	34,80%	65,20%
	First 100	28,00%	72,00%	29,00%	71,00%
	Last 100	21,00%	79,00%	25,00%	75,00%
	Middle 100	25,00%	75,00%	24,00%	76,00%
	Random 100	56,00%	44,00%	39,00%	61,00%
	Random(2) 100	46,00%	54,00%	32,00%	68,00%
	Rand. Distr. 100	46,00%	54,00%	47,00%	53,00%

4.2.4 TF-iDF and k-Nearest Neighbors (5)

Figure 4.12: Single Classifier with TF-iDF Conversion and k-Nearest Neighbors (5)



	k-Nearest Neighbors (k=5)				
		0,67		0,33	
		CORRECT	FAILED	CORRECT	FAILED
TF-iDF	All Train Data	28,42%	71,58%	29,77%	70,23%
	First 100	27,00%	73,00%	32,00%	68,00%
	Last 100	23,00%	77,00%	25,00%	75,00%
	Middle 100	22,00%	78,00%	26,00%	74,00%
	Random 100	47,00%	53,00%	37,00%	63,00%
	Random(2) 100	35,00%	65,00%	27,00%	73,00%
	Rand. Distr. 100	28,00%	72,00%	38,00%	62,00%

K-Nearest Neighbor with k to be 5, TF-iDF conversion helped slightly in improvement but the accuracy rates were far from the desirable results.

As the second conversion ended, we kept observing the first, last and middle 100 data. The results continued to be low and we gave them another chance before we dive into them and see them more closely.

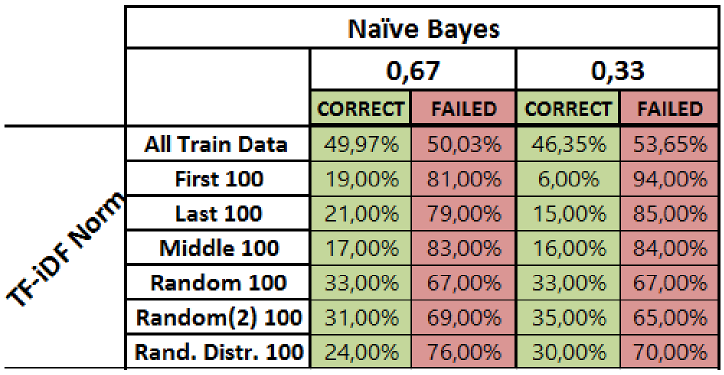
4.3 Normalized TF - iDF Conversion with Single Classifier

We chose to apply normalization on TF-iDF as our third conversion approach in order to observe if scaling our features will cause any effect to our classifiers. By creating features that have similar ranges to each other all numeric values of variables are scaled in the range [0, 1].

4.3.1 Normalized TF-iDF and Naive Bayes

As started on the previous conversions, Naive Bayes was again the first algorithm which has been tested on normalized data and we cannot say that the performance went well enough. In contrast, the successful accuracy percentages were reduced significantly in any test data.

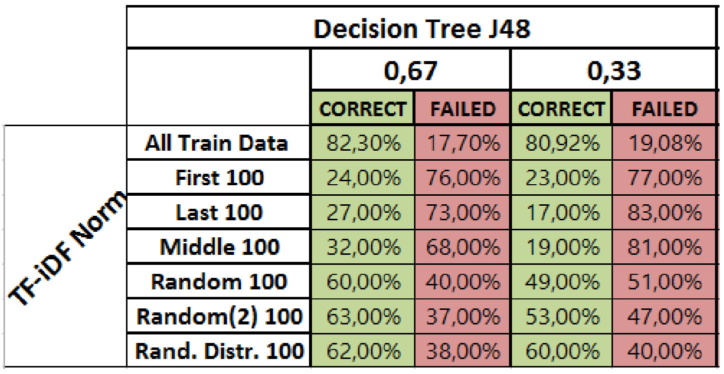
Figure 4.13: Single Classifier with Normalized TF-iDF Conversion and Naive Bayes



	Naïve Bayes				
		0,67		0,33	
		CORRECT	FAILED	CORRECT	FAILED
TF-iDF Norm	All Train Data	49,97%	50,03%	46,35%	53,65%
	First 100	19,00%	81,00%	6,00%	94,00%
	Last 100	21,00%	79,00%	15,00%	85,00%
	Middle 100	17,00%	83,00%	16,00%	84,00%
	Random 100	33,00%	67,00%	33,00%	67,00%
	Random(2) 100	31,00%	69,00%	35,00%	65,00%
	Rand. Distr. 100	24,00%	76,00%	30,00%	70,00%

4.3.2 Normalized TF-iDF and Decision Tree J48

Figure 4.14: Single Classifier with Normalized TF-iDF Conversion and Decision Tree J48

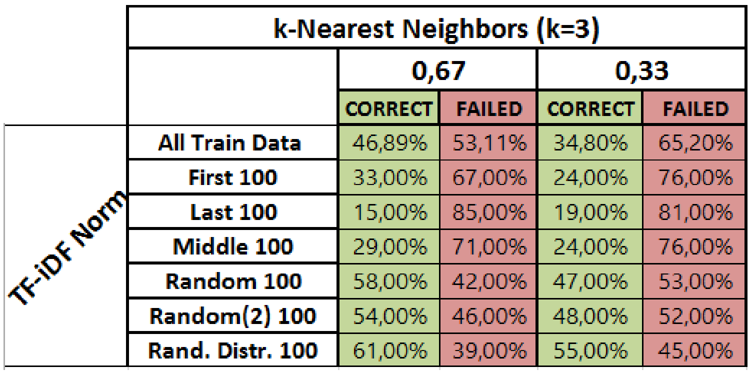


	Decision Tree J48				
		0,67		0,33	
		CORRECT	FAILED	CORRECT	FAILED
TF-iDF Norm	All Train Data	82,30%	17,70%	80,92%	19,08%
	First 100	24,00%	76,00%	23,00%	77,00%
	Last 100	27,00%	73,00%	17,00%	83,00%
	Middle 100	32,00%	68,00%	19,00%	81,00%
	Random 100	60,00%	40,00%	49,00%	51,00%
	Random(2) 100	63,00%	37,00%	53,00%	47,00%
	Rand. Distr. 100	62,00%	38,00%	60,00%	40,00%

Normalization on TF-iDF did not work well enough for the Decision Tree J48 also. The rates at the training data remained the same as the previous conversion but they have been decreased in most cases.

4.3.3 Normalized TF-iDF and k-Nearest Neighbors (3)

Figure 4.15: Single Classifier with Normalized TF-iDF Conversion and k-Nearest Neighbors (3)

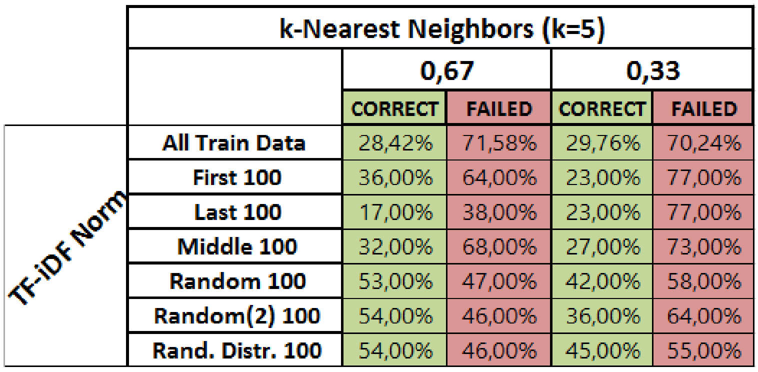


	k-Nearest Neighbors (k=3)				
		0,67		0,33	
		CORRECT	FAILED	CORRECT	FAILED
TF-iDF Norm	All Train Data	46,89%	53,11%	34,80%	65,20%
	First 100	33,00%	67,00%	24,00%	76,00%
	Last 100	15,00%	85,00%	19,00%	81,00%
	Middle 100	29,00%	71,00%	24,00%	76,00%
	Random 100	58,00%	42,00%	47,00%	53,00%
	Random(2) 100	54,00%	46,00%	48,00%	52,00%
	Rand. Distr. 100	61,00%	39,00%	55,00%	45,00%

Scaling the data had a positive impact on k-Nearest Neighbors with k value of 3. With this, the highest score achieved at the random distributed data with the percentage of 61% at threshold of 0,67 and 55% at threshold of 0,33.

4.3.4 Normalized TF-iDF and k-Nearest Neighbors (5)

Figure 4.16: Single Classifier with Normalized TF-iDF Conversion and k-Nearest Neighbors (5)



		k-Nearest Neighbors (k=5)			
		0,67		0,33	
		CORRECT	FAILED	CORRECT	FAILED
TF-iDF Norm	All Train Data	28,42%	71,58%	29,76%	70,24%
	First 100	36,00%	64,00%	23,00%	77,00%
	Last 100	17,00%	38,00%	23,00%	77,00%
	Middle 100	32,00%	68,00%	27,00%	73,00%
	Random 100	53,00%	47,00%	42,00%	58,00%
	Random(2) 100	54,00%	46,00%	36,00%	64,00%
	Rand. Distr. 100	54,00%	46,00%	45,00%	55,00%

k-Nearest Neighbors with k to be 5, did not make much progress compared to the previous related conversion, however the accuracies changed in a better way.

Notation

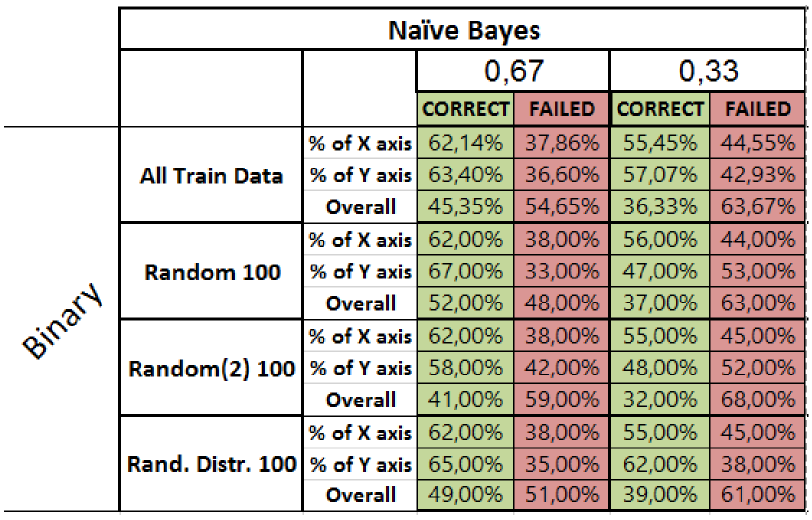
At this time we have completed all the three approaches that we selected for our project regarding the single classifier and we noticed that the first, last and middle 100 data had a really low and small range of rates in every algorithm and conversion, while the random data achieved twice and sometimes three times bigger accuracy percentage than the rest. This kind of curiosity made us look deeper into these datasets to discover that the distribution of these datasets were dissimilar to the original training distribution and since the classifier cannot recognize such dissimilarities, this leads to misclassifications. The number of features of this data had minor differences compared to the rest random datasets but in combination with the unbalanced statistical properties of these datasets proved to be unmeaningful, unhelpful and unnecessary for further research and we excluded them from testing processes.

4.4 Binary Conversion with Dual Classifiers

Considering the results on the single classifier we wanted to find a way to improve our accuracy rates without making huge changes or embedding other methods in our project, like audio processing or advanced methods of natural language processing. For this purpose an idea occurred based on the algorithm design paradigm of ”divide and conquer”[33] which led us to split the single classifier into two individual classifiers for each axis as described in the beginning of this chapter and illustrated in the figure 4.4. Each classifier has been trained to the same dataset but it was tested on two different test datasets, each one for each axis and, therefore, each classifier produced individual results where we matched the correct and false into overall percentages.

4.4.1 Binary and Naive Bayes

Figure 4.17: Dual Classifiers with Binary Conversion and Naive Bayes

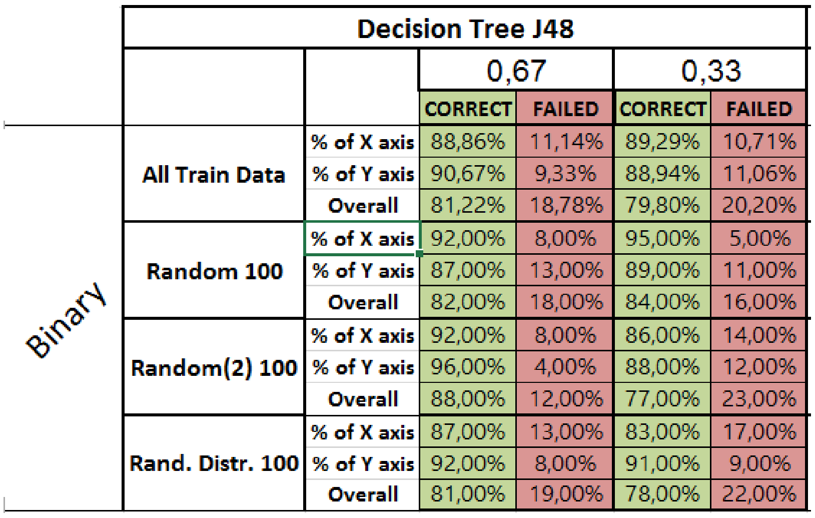


Binary			Naïve Bayes 0,67 CORRECT	Naïve Bayes 0,67 FAILED	Naïve Bayes 0,33 CORRECT	Naïve Bayes 0,33 FAILED
	All Train Data	% of X axis	62,14%	37,86%	55,45%	44,55%
		% of Y axis	63,40%	36,60%	57,07%	42,93%
		Overall	45,35%	54,65%	36,33%	63,67%
	Random 100	% of X axis	62,00%	38,00%	56,00%	44,00%
		% of Y axis	67,00%	33,00%	47,00%	53,00%
		Overall	52,00%	48,00%	37,00%	63,00%
	Random(2) 100	% of X axis	62,00%	38,00%	55,00%	45,00%
		% of Y axis	58,00%	42,00%	48,00%	52,00%
		Overall	41,00%	59,00%	32,00%	68,00%
	Rand. Distr. 100	% of X axis	62,00%	38,00%	55,00%	45,00%
		% of Y axis	65,00%	35,00%	62,00%	38,00%
		Overall	49,00%	51,00%	39,00%	61,00%

We started with binary conversion and Naive Bayes the same implementation as we did on the previous approach. As individual classifiers they produced slightly better results than the single classifier but while the results matched each other, the overall was slightly worse than the single.

4.4.2 Binary and Decision Tree J48

Figure 4.18: Dual Classifiers with Binary Conversion and Decision Tree J48

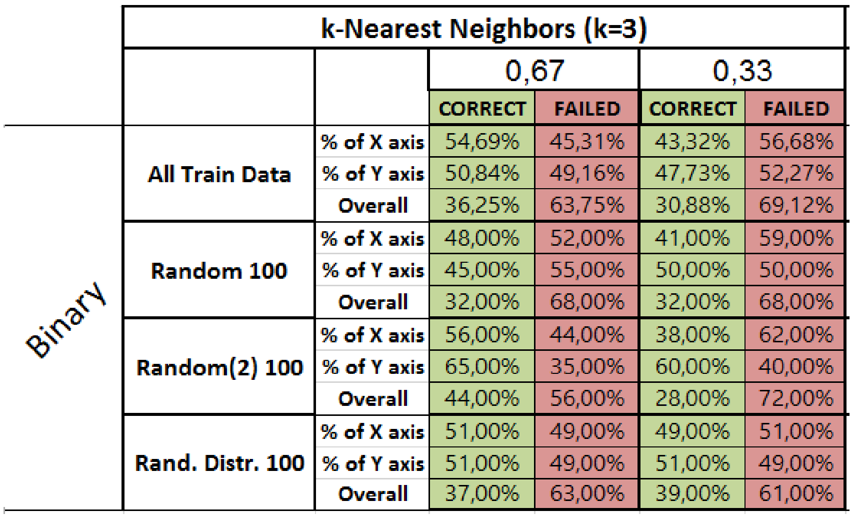


Binary			Decision Tree J48 0,67 CORRECT	Decision Tree J48 0,67 FAILED	Decision Tree J48 0,33 CORRECT	Decision Tree J48 0,33 FAILED
	All Train Data	% of X axis	88,86%	11,14%	89,29%	10,71%
		% of Y axis	90,67%	9,33%	88,94%	11,06%
		Overall	81,22%	18,78%	79,80%	20,20%
	Random 100	% of X axis	92,00%	8,00%	95,00%	5,00%
		% of Y axis	87,00%	13,00%	89,00%	11,00%
		Overall	82,00%	18,00%	84,00%	16,00%
	Random(2) 100	% of X axis	92,00%	8,00%	86,00%	14,00%
		% of Y axis	96,00%	4,00%	88,00%	12,00%
		Overall	88,00%	12,00%	77,00%	23,00%
	Rand. Distr. 100	% of X axis	87,00%	13,00%	83,00%	17,00%
		% of Y axis	92,00%	8,00%	91,00%	9,00%
		Overall	81,00%	19,00%	78,00%	22,00%

Remarkable progress has been made in Decision Tree Algorithm and specifically on 0,33 threshold. As we see on the figure above some of the classification reached rates above 90% even though it was on individual classifiers. It seems that the balanced data work better with dual classifiers on Decision Tree.

4.4.3 Binary and k-Nearest Neighbors (3)

Figure 4.19: Dual Classifiers with Binary Conversion and k-Nearest Neighbors (3)

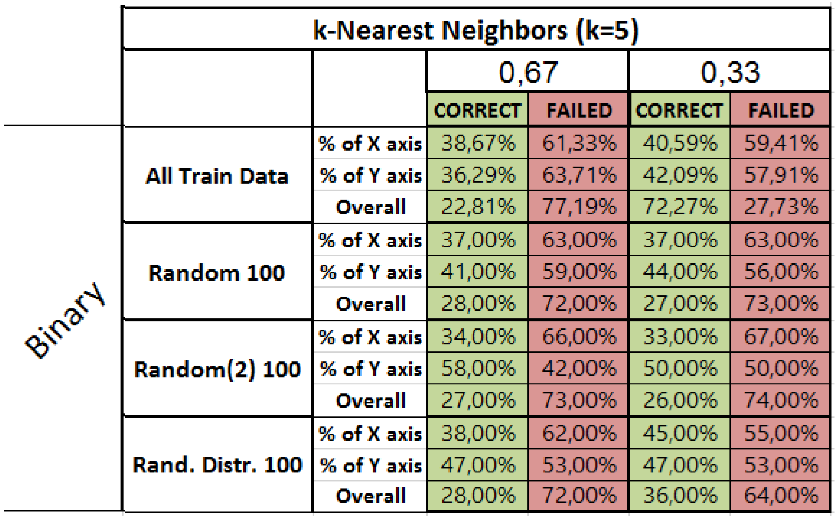


k-Nearest Neighbors (k=3)						
			0,67		0,33	
			CORRECT	FAILED	CORRECT	FAILED
Binary	All Train Data	% of X axis	54,69%	45,31%	43,32%	56,68%
		% of Y axis	50,84%	49,16%	47,73%	52,27%
		Overall	36,25%	63,75%	30,88%	69,12%
	Random 100	% of X axis	48,00%	52,00%	41,00%	59,00%
		% of Y axis	45,00%	55,00%	50,00%	50,00%
		Overall	32,00%	68,00%	32,00%	68,00%
	Random(2) 100	% of X axis	56,00%	44,00%	38,00%	62,00%
		% of Y axis	65,00%	35,00%	60,00%	40,00%
		Overall	44,00%	56,00%	28,00%	72,00%
	Rand. Distr. 100	% of X axis	51,00%	49,00%	49,00%	51,00%
		% of Y axis	51,00%	49,00%	51,00%	49,00%
		Overall	37,00%	63,00%	39,00%	61,00%

Next, we cannot say that k-Nearest Neighbors with k value 3 made any progress. On the contrary, worse rates occurred compared to the previous binary conversion of single classifier.

4.4.4 Binary and k-Nearest Neighbors (5)

Figure 4.20: Dual Classifiers with Binary Conversion and k-Nearest Neighbors (5)



k-Nearest Neighbors (k=5)						
			0,67		0,33	
			CORRECT	FAILED	CORRECT	FAILED
Binary	All Train Data	% of X axis	38,67%	61,33%	40,59%	59,41%
		% of Y axis	36,29%	63,71%	42,09%	57,91%
		Overall	22,81%	77,19%	72,27%	27,73%
	Random 100	% of X axis	37,00%	63,00%	37,00%	63,00%
		% of Y axis	41,00%	59,00%	44,00%	56,00%
		Overall	28,00%	72,00%	27,00%	73,00%
	Random(2) 100	% of X axis	34,00%	66,00%	33,00%	67,00%
		% of Y axis	58,00%	42,00%	50,00%	50,00%
		Overall	27,00%	73,00%	26,00%	74,00%
	Rand. Distr. 100	% of X axis	38,00%	62,00%	45,00%	55,00%
		% of Y axis	47,00%	53,00%	47,00%	53,00%
		Overall	28,00%	72,00%	36,00%	64,00%

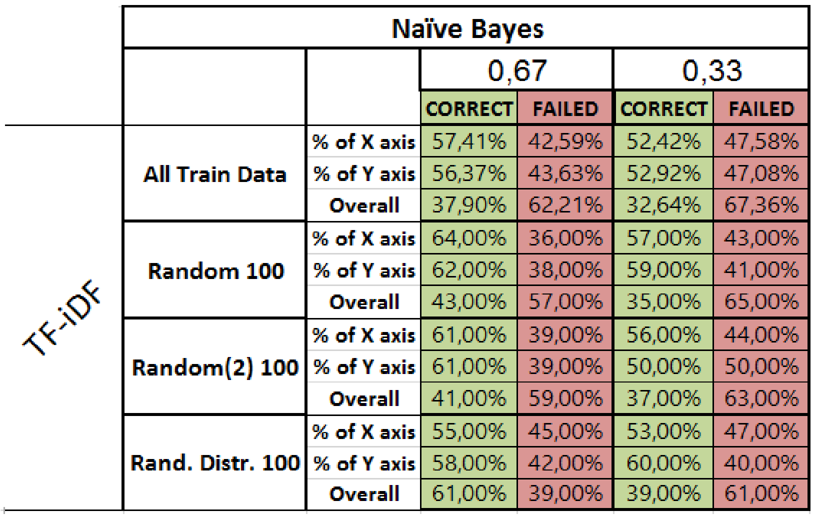
Lastly, k-Nearest Neighbors with k value 5 did not produce any different kind of results and it remained in low accuracy rates.

4.5 TF - iDF Conversion with Dual Classifiers

Thereupon, we continued with the TF-iDF transformation with the same process as implemented on the single classifier. Let's see how real numeric values behaved to dual classifiers in combination with the impact of this conversion on each bag-of-words model.

4.5.1 TF-iDF and Naive Bayes

Figure 4.21: Dual Classifiers with TF-iDF Conversion and Naive Bayes

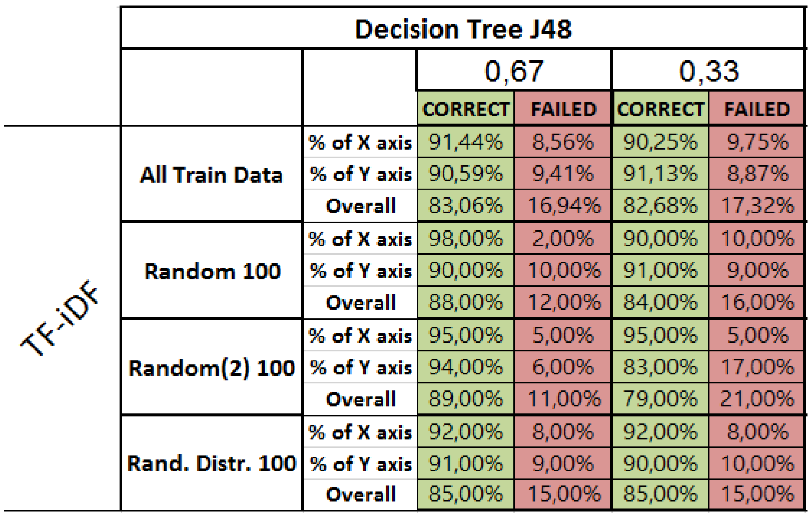


			Naïve Bayes			
			0,67		0,33	
			CORRECT	FAILED	CORRECT	FAILED
TF-iDF	All Train Data	% of X axis	57,41%	42,59%	52,42%	47,58%
		% of Y axis	56,37%	43,63%	52,92%	47,08%
		Overall	37,90%	62,21%	32,64%	67,36%
	Random 100	% of X axis	64,00%	36,00%	57,00%	43,00%
		% of Y axis	62,00%	38,00%	59,00%	41,00%
		Overall	43,00%	57,00%	35,00%	65,00%
	Random(2) 100	% of X axis	61,00%	39,00%	56,00%	44,00%
		% of Y axis	61,00%	39,00%	50,00%	50,00%
		Overall	41,00%	59,00%	37,00%	63,00%
	Rand. Distr. 100	% of X axis	55,00%	45,00%	53,00%	47,00%
		% of Y axis	58,00%	42,00%	60,00%	40,00%
		Overall	61,00%	39,00%	39,00%	61,00%

Naive Bayes in most cases noted better accuracy rates than the binary conversion in the two individual classifications. However, the matching overall rates of TF-iDF were worse than the overall classifications of Naive Bayes on the Binary approach.

4.5.2 TF-iDF and Decision Tree J48

Figure 4.22: Dual Classifiers with TF-iDF Conversion and Decision Tree J48

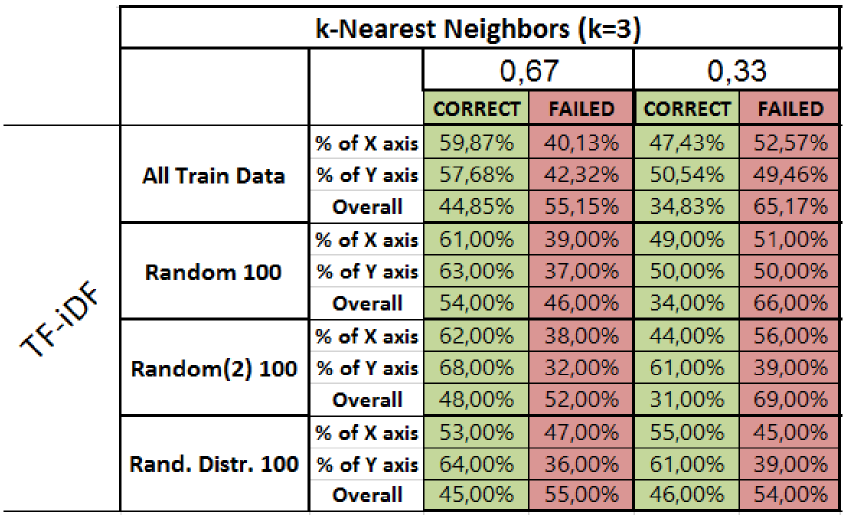


			Decision Tree J48			
			0,67		0,33	
			CORRECT	FAILED	CORRECT	FAILED
TF-iDF	All Train Data	% of X axis	91,44%	8,56%	90,25%	9,75%
		% of Y axis	90,59%	9,41%	91,13%	8,87%
		Overall	83,06%	16,94%	82,68%	17,32%
	Random 100	% of X axis	98,00%	2,00%	90,00%	10,00%
		% of Y axis	90,00%	10,00%	91,00%	9,00%
		Overall	88,00%	12,00%	84,00%	16,00%
	Random(2) 100	% of X axis	95,00%	5,00%	95,00%	5,00%
		% of Y axis	94,00%	6,00%	83,00%	17,00%
		Overall	89,00%	11,00%	79,00%	21,00%
	Rand. Distr. 100	% of X axis	92,00%	8,00%	92,00%	8,00%
		% of Y axis	91,00%	9,00%	90,00%	10,00%
		Overall	85,00%	15,00%	85,00%	15,00%

Decision Tree (J48) algorithm has made significant improvements in correct classifications. Since we had already high accuracy rates on the previous conversion of the Decision Tree, these improvements are considered important as we see above. The individual classifiers made only two, five or six misclassifications while the overall correct percentage is close to hitting 90% in some cases.

4.5.3 TF-iDF and k-Nearest Neighbors (3)

Figure 4.23: Dual Classifiers with TF-iDF Conversion and k-Nearest Neighbors (3)

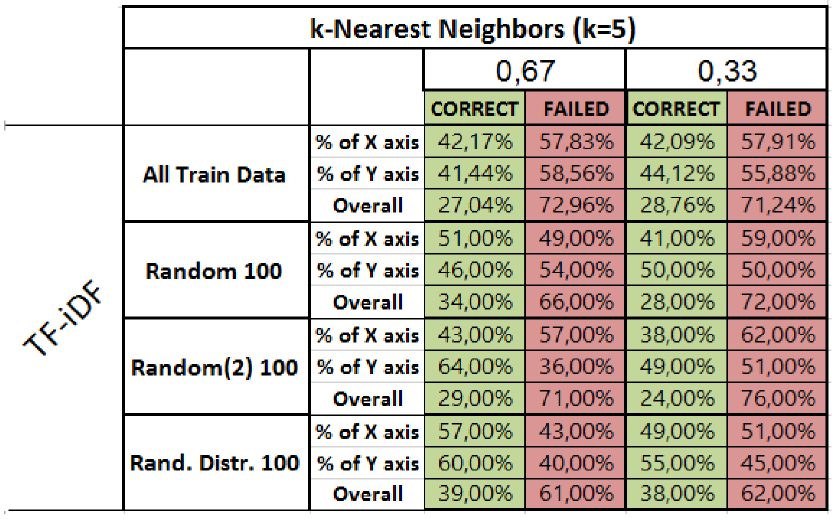


			k-Nearest Neighbors (k=3)			
			0,67		0,33	
			CORRECT	FAILED	CORRECT	FAILED
TF-iDF	All Train Data	% of X axis	59,87%	40,13%	47,43%	52,57%
		% of Y axis	57,68%	42,32%	50,54%	49,46%
		Overall	44,85%	55,15%	34,83%	65,17%
	Random 100	% of X axis	61,00%	39,00%	49,00%	51,00%
		% of Y axis	63,00%	37,00%	50,00%	50,00%
		Overall	54,00%	46,00%	34,00%	66,00%
	Random(2) 100	% of X axis	62,00%	38,00%	44,00%	56,00%
		% of Y axis	68,00%	32,00%	61,00%	39,00%
		Overall	48,00%	52,00%	31,00%	69,00%
	Rand. Distr. 100	% of X axis	53,00%	47,00%	55,00%	45,00%
		% of Y axis	64,00%	36,00%	61,00%	39,00%
		Overall	45,00%	55,00%	46,00%	54,00%

Dual classification with TF-iDF worked for k-NN(3) too. As we can see, there are clearly better results compared to the previous one. However, these results are far beyond the results of Decision Tree or our desired results.

4.5.4 TF-iDF and k-Nearest Neighbors (5)

Figure 4.24: Dual Classifiers with TF-iDF Conversion and k-Nearest Neighbors (5)



			k-Nearest Neighbors (k=5)			
			0,67		0,33	
			CORRECT	FAILED	CORRECT	FAILED
TF-iDF	All Train Data	% of X axis	42,17%	57,83%	42,09%	57,91%
		% of Y axis	41,44%	58,56%	44,12%	55,88%
		Overall	27,04%	72,96%	28,76%	71,24%
	Random 100	% of X axis	51,00%	49,00%	41,00%	59,00%
		% of Y axis	46,00%	54,00%	50,00%	50,00%
		Overall	34,00%	66,00%	28,00%	72,00%
	Random(2) 100	% of X axis	43,00%	57,00%	38,00%	62,00%
		% of Y axis	64,00%	36,00%	49,00%	51,00%
		Overall	29,00%	71,00%	24,00%	76,00%
	Rand. Distr. 100	% of X axis	57,00%	43,00%	49,00%	51,00%
		% of Y axis	60,00%	40,00%	55,00%	45,00%
		Overall	39,00%	61,00%	38,00%	62,00%

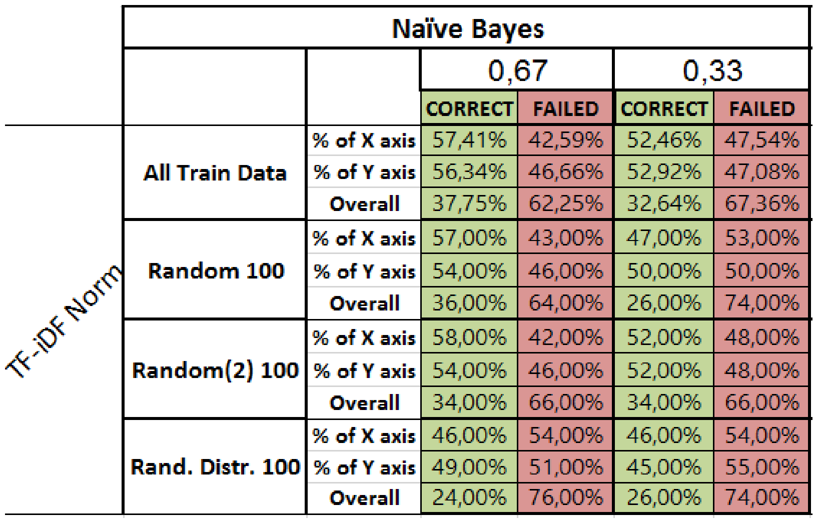
TF-iDF did not give us something noteworthy for k-Nearest Neighbors(5) and with this, TF-iDF conversion has ended for dual classifiers.

4.6 Normalized TF - iDF Conversion with Dual Classifiers

As we did on the last conversion at single classifier, we do the same here for dual classifiers. Normalizing values while having been transformed with TF-iDF weighting equation for each datasets, it may produce higher percentages than the last normalized conversion.

4.6.1 Normalized TF-iDF and Naive Bayes

Figure 4.25: Dual Classifiers with Normalized TF-iDF Conversion and Naive Bayes

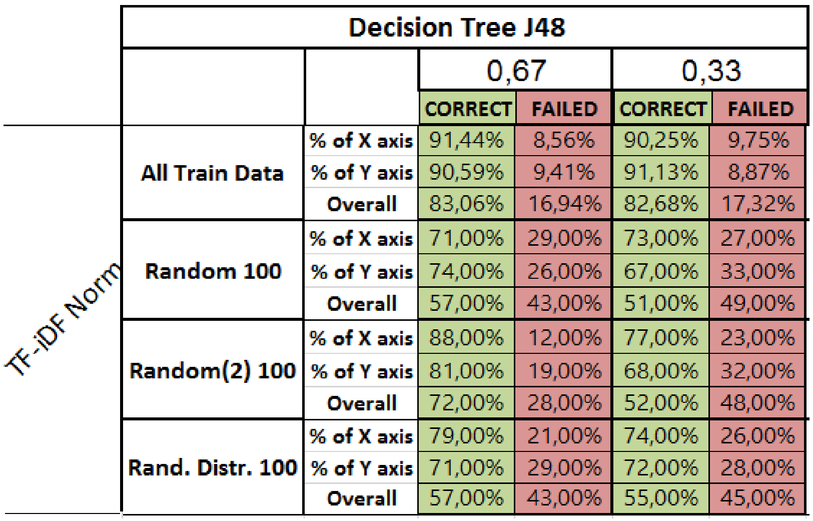


			Naïve Bayes			
			0,67		0,33	
			CORRECT	FAILED	CORRECT	FAILED
TF-iDF Norm	All Train Data	% of X axis	57,41%	42,59%	52,46%	47,54%
		% of Y axis	56,34%	46,66%	52,92%	47,08%
		Overall	37,75%	62,25%	32,64%	67,36%
	Random 100	% of X axis	57,00%	43,00%	47,00%	53,00%
		% of Y axis	54,00%	46,00%	50,00%	50,00%
		Overall	36,00%	64,00%	26,00%	74,00%
	Random(2) 100	% of X axis	58,00%	42,00%	52,00%	48,00%
		% of Y axis	54,00%	46,00%	52,00%	48,00%
		Overall	34,00%	66,00%	34,00%	66,00%
	Rand. Distr. 100	% of X axis	46,00%	54,00%	46,00%	54,00%
		% of Y axis	49,00%	51,00%	45,00%	55,00%
		Overall	24,00%	76,00%	26,00%	74,00%

Naive Bayes was the first one as usual. It seems that normalizing values did not work well enough for this particular algorithm, even though the accuracies on training datasets remained the same.

4.6.2 Normalized TF-iDF and Decision Tree J48

Figure 4.26: Dual Classifiers with Normalized TF-iDF Conversion and Decision Tree J48

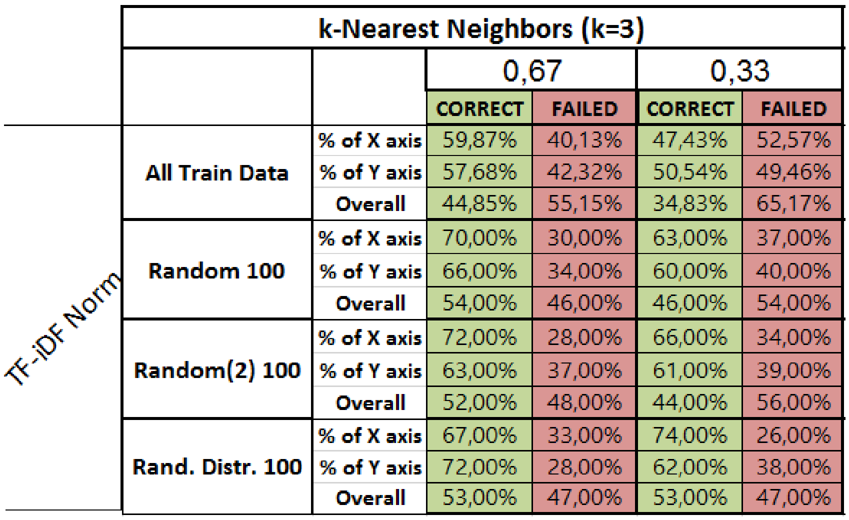


Decision Tree J48						
			0,67		0,33	
			CORRECT	FAILED	CORRECT	FAILED
TF-iDF Norm	All Train Data	% of X axis	91,44%	8,56%	90,25%	9,75%
		% of Y axis	90,59%	9,41%	91,13%	8,87%
		Overall	83,06%	16,94%	82,68%	17,32%
	Random 100	% of X axis	71,00%	29,00%	73,00%	27,00%
		% of Y axis	74,00%	26,00%	67,00%	33,00%
		Overall	57,00%	43,00%	51,00%	49,00%
	Random(2) 100	% of X axis	88,00%	12,00%	77,00%	23,00%
		% of Y axis	81,00%	19,00%	68,00%	32,00%
		Overall	72,00%	28,00%	52,00%	48,00%
	Rand. Distr. 100	% of X axis	79,00%	21,00%	74,00%	26,00%
		% of Y axis	71,00%	29,00%	72,00%	28,00%
		Overall	57,00%	43,00%	55,00%	45,00%

Interesting falls in percentages were noted in Decision Tree while normalizing the values of vectors. With the rates close to 50% this was by far the worse result that J48 gave us in any run.

4.6.3 Normalized TF-iDF and k-Nearest Neighbors (3)

Figure 4.27: Dual Classifiers with Normalized TF-iDF Conversion and k-Nearest Neighbors (3)

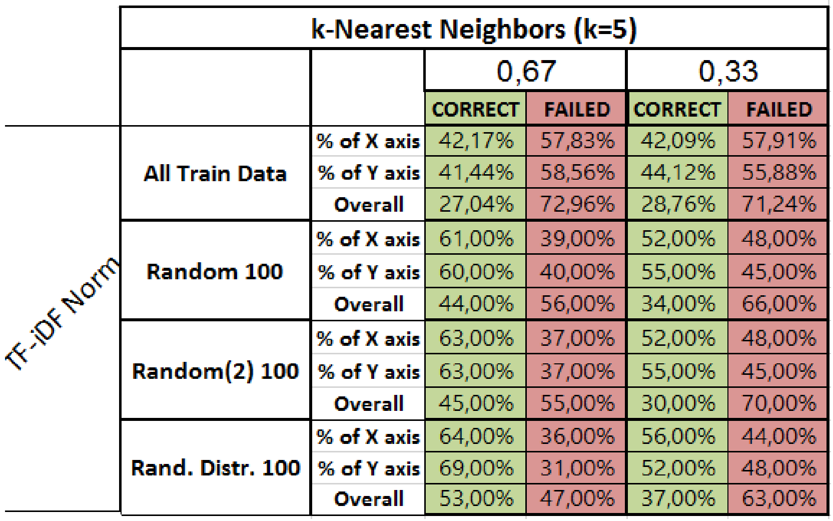


k-Nearest Neighbors (k=3)						
			0,67		0,33	
			CORRECT	FAILED	CORRECT	FAILED
TF-iDF Norm	All Train Data	% of X axis	59,87%	40,13%	47,43%	52,57%
		% of Y axis	57,68%	42,32%	50,54%	49,46%
		Overall	44,85%	55,15%	34,83%	65,17%
	Random 100	% of X axis	70,00%	30,00%	63,00%	37,00%
		% of Y axis	66,00%	34,00%	60,00%	40,00%
		Overall	54,00%	46,00%	46,00%	54,00%
	Random(2) 100	% of X axis	72,00%	28,00%	66,00%	34,00%
		% of Y axis	63,00%	37,00%	61,00%	39,00%
		Overall	52,00%	48,00%	44,00%	56,00%
	Rand. Distr. 100	% of X axis	67,00%	33,00%	74,00%	26,00%
		% of Y axis	72,00%	28,00%	62,00%	38,00%
		Overall	53,00%	47,00%	53,00%	47,00%

Converting the weights within the specific range of [0,1] offered an improvement to k-Nearest Neighbors in individual classifications as well as in overall matches. There are no huge differences compared to TF-iDF transformation but still the rates are slightly better.

4.6.4 Normalized TF-iDF and k-Nearest Neighbors (5)

Figure 4.28: Dual Classifiers with Normalized TF-iDF Conversion and k-Nearest Neighbors (5)



k-Nearest Neighbors (k=5)						
			0,67		0,33	
			CORRECT	FAILED	CORRECT	FAILED
TF-iDF Norm	All Train Data	% of X axis	42,17%	57,83%	42,09%	57,91%
		% of Y axis	41,44%	58,56%	44,12%	55,88%
		Overall	27,04%	72,96%	28,76%	71,24%
	Random 100	% of X axis	61,00%	39,00%	52,00%	48,00%
		% of Y axis	60,00%	40,00%	55,00%	45,00%
		Overall	44,00%	56,00%	34,00%	66,00%
	Random(2) 100	% of X axis	63,00%	37,00%	52,00%	48,00%
		% of Y axis	63,00%	37,00%	55,00%	45,00%
		Overall	45,00%	55,00%	30,00%	70,00%
	Rand. Distr. 100	% of X axis	64,00%	36,00%	56,00%	44,00%
		% of Y axis	69,00%	31,00%	52,00%	48,00%
		Overall	53,00%	47,00%	37,00%	63,00%

Since normalization worked on the previous run of k-Nearest Neighbors with k-value to be 3, we expected improvements for the k-value of 5 and indeed we had better accuracy rates compared to the other conversions but still far for our desired percentages.

Chapter 5

Result Analysis

This point has been reached after hundreds of hours of coding, studying and researching in order to complete all the stages for every approach. After we gathered all the results, it was time to analyze their meaning. This chapter interprets our experimental results in presentation and analyzation terms. Each figure illustrates the accuracies percentages either particular algorithm or all together and in some cases, comparisons between the approaches and our classifiers.

5.1 Experimental Results of Selected Thresholds

5.1.1 Thresholds of 0.67 and 0.33 at Single Classifier

Figure 5.1: Results of 0.67 Threshold at Single Classifier [Average Correct and False Percentages Values of the Random Test Files]

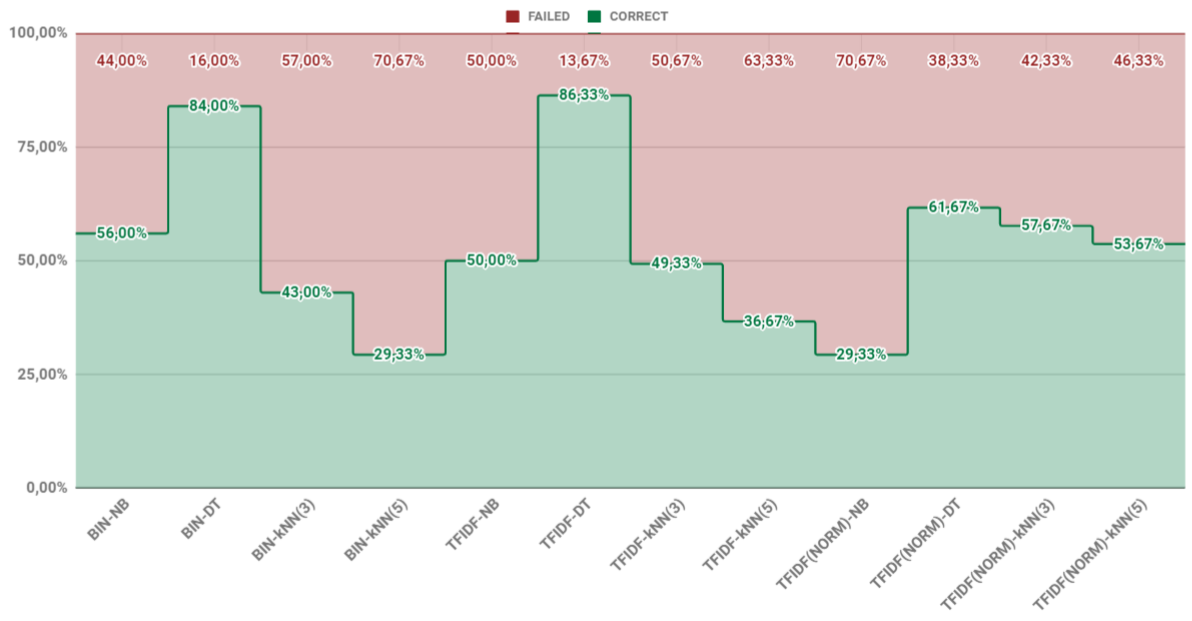
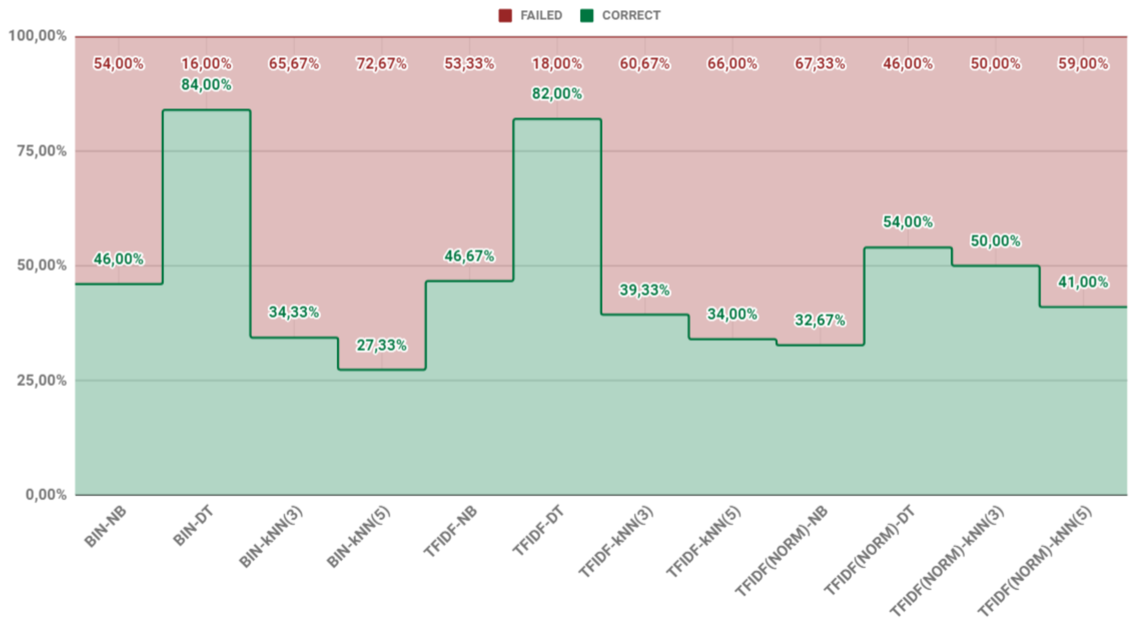


Figure 5.2: Results of 0.33 Threshold at Single Classifier [Average Correct and False Percentages Values of the Random Test Files]



As we saw in chapter 3 in pre-processing(3.3.1) stage we clustered our lyrics into 3 groups based on specific splitting values of 0.67 and 0.33 which are called thresholds. The reason for selection of 0.67 was because we wanted to find a fair way to divide our categories. So, this particular value was the result of the division of the range into three equal clusters on the quadrant. The threshold value of 0.33 was selected because it appeared to be the most suitable value among others for dividing the training dataset into the most balanced classes that we can achieve from this dataset in order to observe if that affects our classification accuracies. However, it seems that this splitting method of 0.33 does not perform better even if the classes are much more balanced than the fair method of splitting (0.67). This has to do with the subjectivity of the users because, as we saw, the annotations that we used from subsection 2.5.5 to construct our training dataset, there are users that estimate the valence value of a specific song with the maximum value of 2 while others estimate the same song with 1 or zero which means that the first rater classifies it as positive while the last one classifies the same song as neutral. This happens because each user perceives differently each song in terms of lyrics and mostly in terms of sound properties and when it comes to songs that have subtle differences or huge similarities. So, inaccuracies in the ground truth will lead to inaccuracies in the prediction results.

5.1.2 Thresholds of 0.67 and 0.33 at Dual Classifiers

Figure 5.3: Results of 0.67 Threshold at Dual Classifiers [Overall Average Correct and False Percentages Values of the Random Test Files]

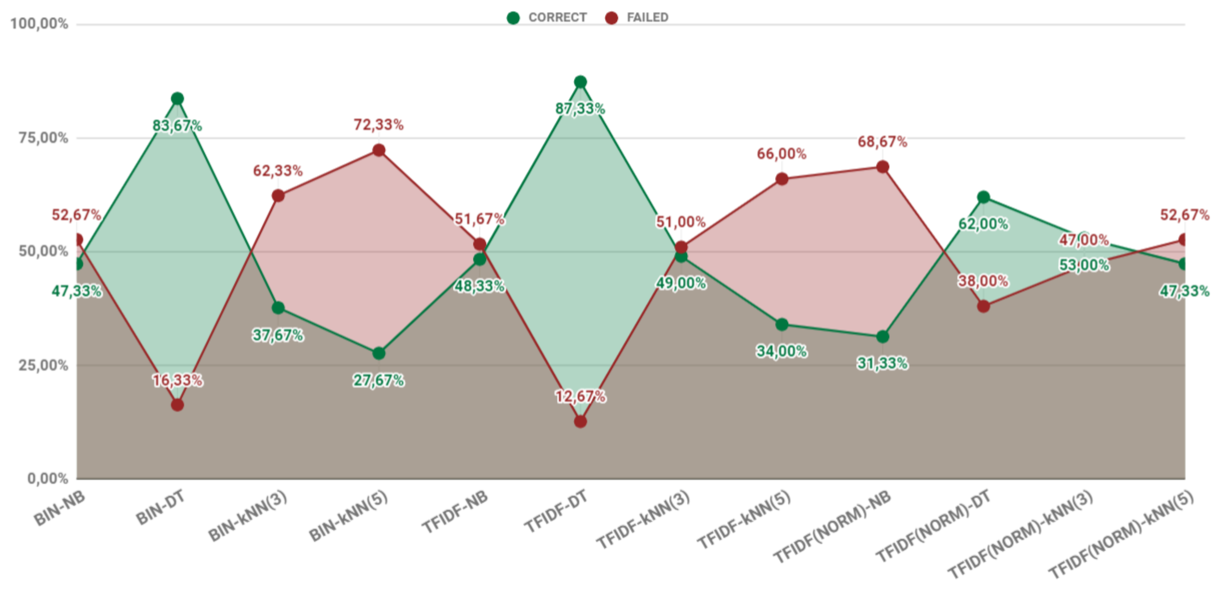
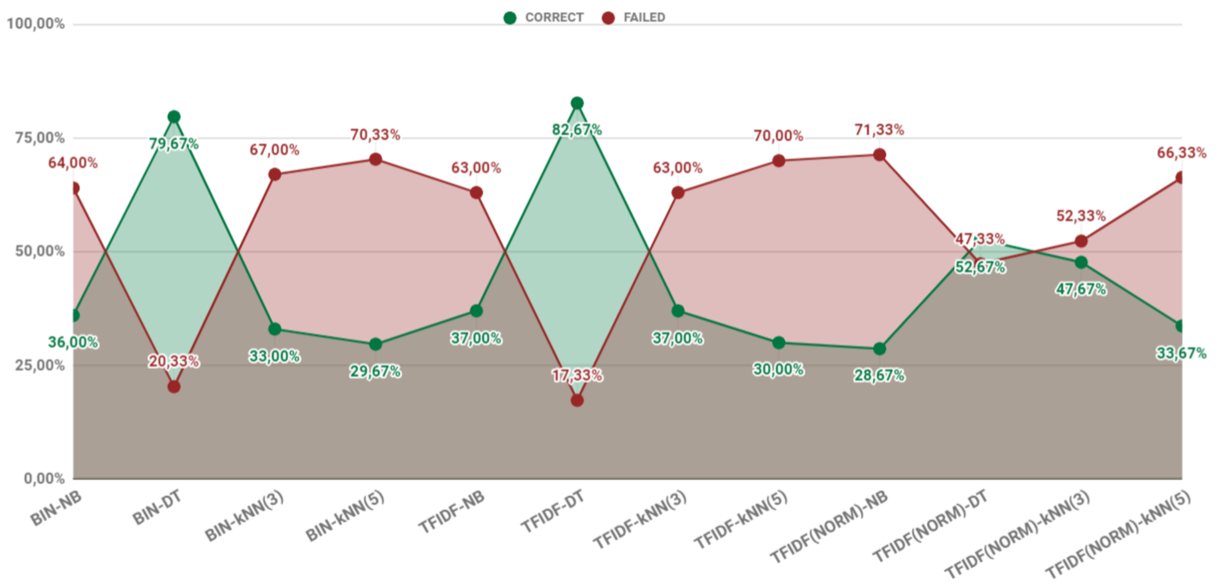


Figure 5.4: Results of 0.33 Threshold at Dual Classifiers [Overall Average Correct and False Percentages Values of the Random Test Files]



Like the previous observation that we made on single classifier, the same is true of dual classifiers. The threshold of 0.33 tends to have higher false accuracy percentages in almost any classification task for the very same reasons that we saw earlier.

5.2 Experimental Results of Machine Learning Algorithms

5.2.1 Naive Bayes Performance at Single and Dual Classifiers

Figure 5.5: Comparisons for Naive Bayes at Single and Dual Classifiers [Threshold of 0.67]

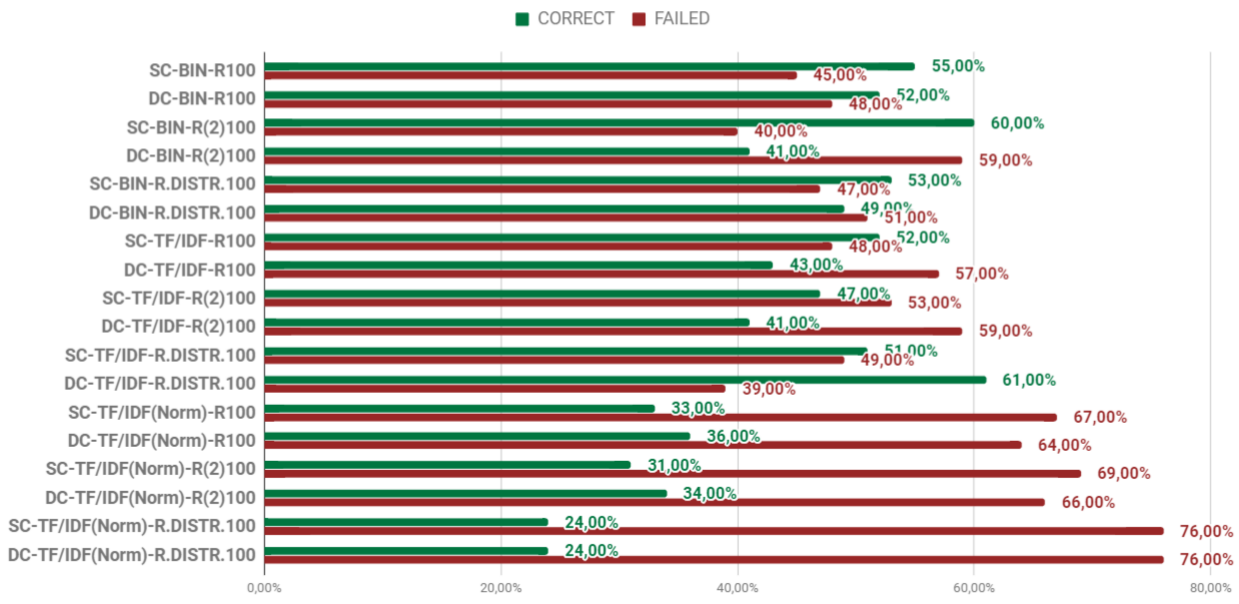
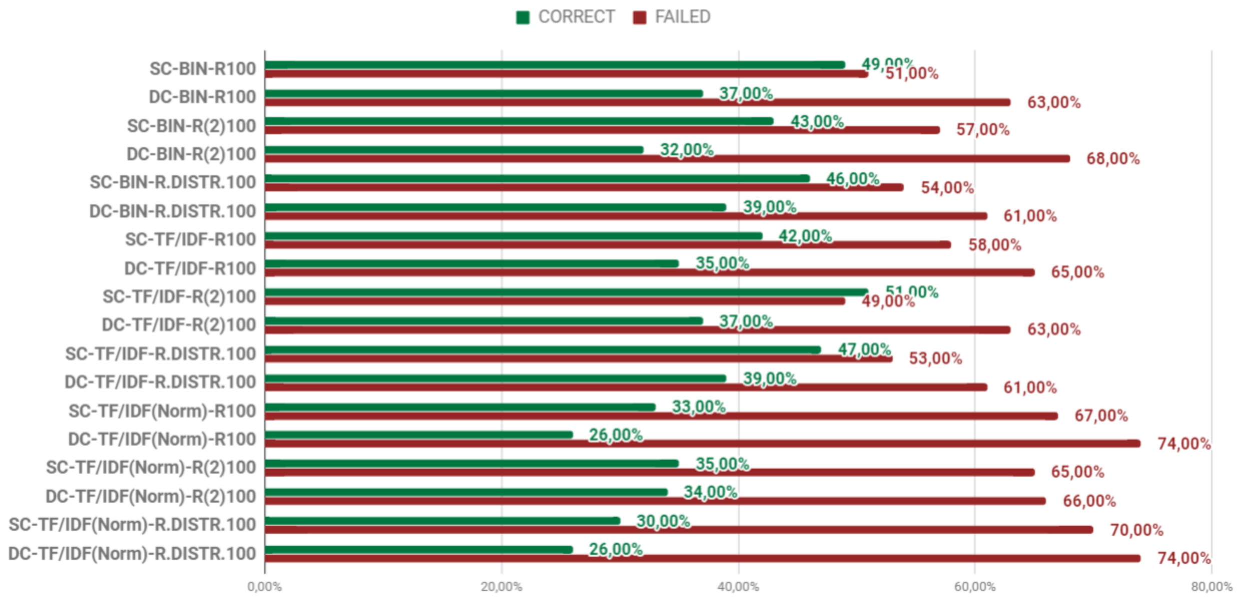


Figure 5.6: Comparisons for Naive Bayes at Single and Dual Classifiers [Threshold of 0.33]



Starting with Naive Bayes we can see that the single classifier prevails in each threshold and almost in any comparison. However, we see high false prediction rates. Naive Bayes algorithm makes a strong assumption that the features are independent of each other which does not seem to work in a good way for mood extraction since there are features that are the same and those that belong to different classes but their meaning differs depending on which class it belongs to. Moreover, not considering correlations among the features would probably lead to incorrect classi-

fication results because as Dominggos and Pazzani stated ”Naïve Bayes is when the independence assumption holds or close to optimal when the attributes are slightly dependent”.[9] Also, normalization seems to reduce accuracy rates due to the fact that the range of weights does not have excessively distant values to each other, so normalization is not always useful for Naive Bayes. In addition, as mentioned earlier, the subjectivity of users regarding thresholds in combination with Naive Bayes’s conditional independence assumption makes things even more complex for the classifier to determine the appropriate class and that is another drawback of this implemented algorithm.
Although the accuracy percentages at some distance from our desired, this experiment showed us that if we are ever going to use Naive Bayes in the future for this project, it should be with single classifier and probably with binary conversion of 0.67 threshold.

5.2.2 Decision Tree(J48) Performance at Single and Dual Classifiers

Figure 5.7: Comparisons for Decision Tree(J48) at Single and Dual Classifiers [Threshold of 0.67]

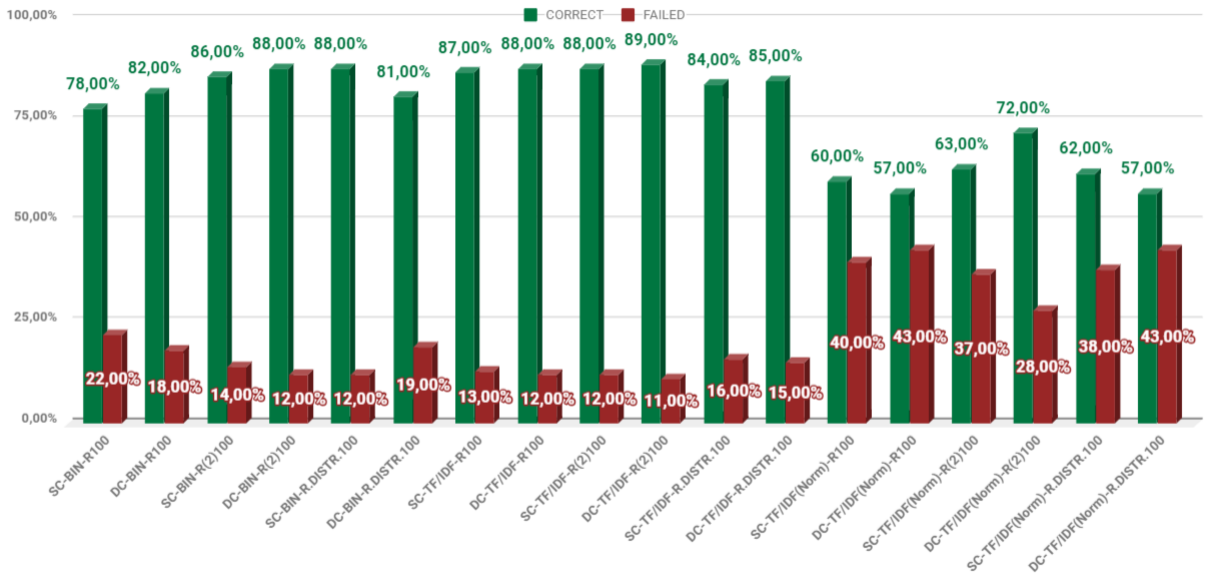
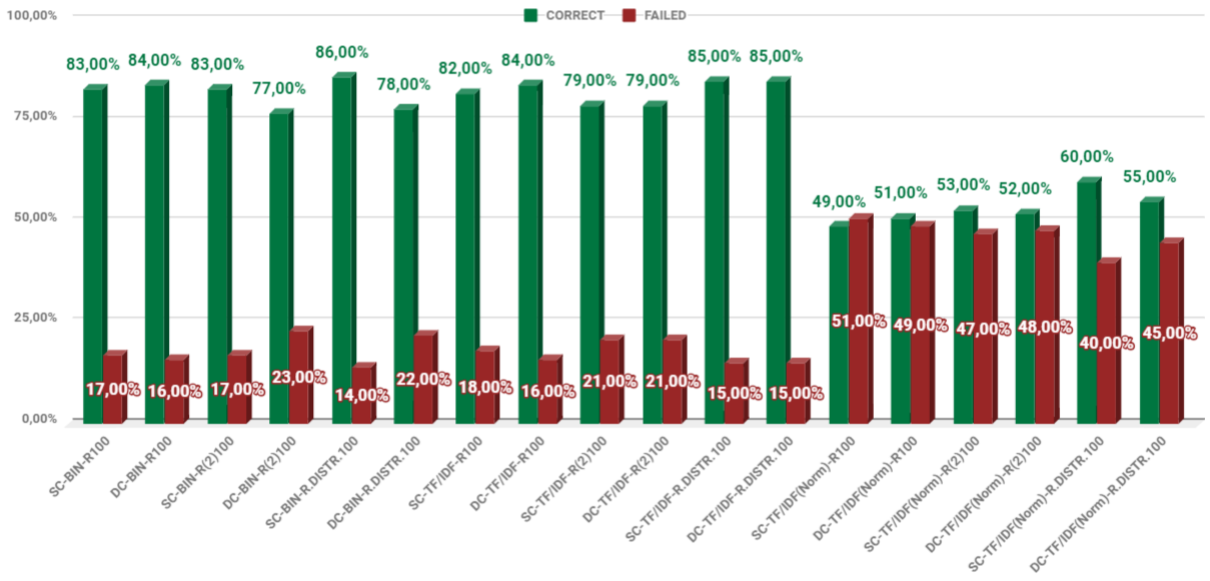


Figure 5.8: Comparisons for Decision Tree(J48) at Single and Dual Classifiers [Threshold of 0.33]



Next we used J48 on our project which proved to be by far the best applied algorithm that we have used, with correct accuracy percentages reaching almost 90%. Both classifier versions (single and dual) performed really well but dual classifiers had slightly better results in classification predictions. The reason for this is probably because at dual classifiers the algorithm had fewer classes to assign and as a result, less complex criteria for the splitting process. Decision Tree implementation performance seems to be quite impressive it terms of mood extraction since it is based on subset trees which are effective for emotion classification. One huge advantage of this is the pruning process technique that is applied to the algorithm which removes unnecessary trees in order to reduce the complexity, avoid overfitting and improve accuracy prediction rates.

That was very useful for our particular dataset because there were repetitive data that needed to be cut during the selection of the best splitting values. Speaking of which, normalization in numeric attributes appeared to not be necessary for J48 because the data are scale-senseless for splitting criteria.

5.2.3 k-Nearest Neighbors(3) Performance at Single and Dual Classifiers

Figure 5.9: Comparisons for k-Nearest Neighbors(3) s at Single and Dual Classifiers [Threshold of 0.67]

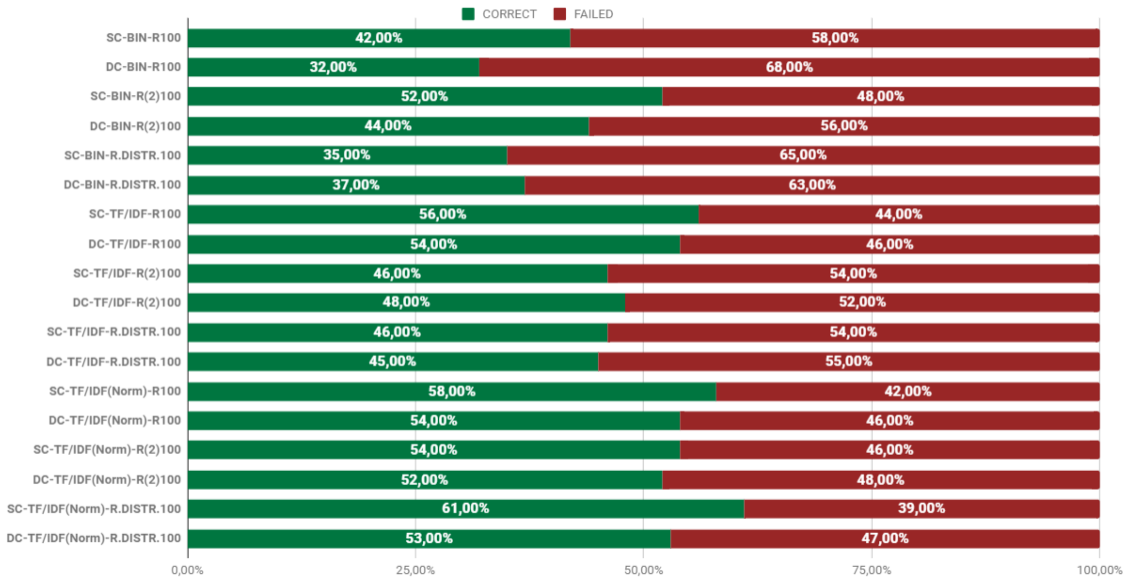
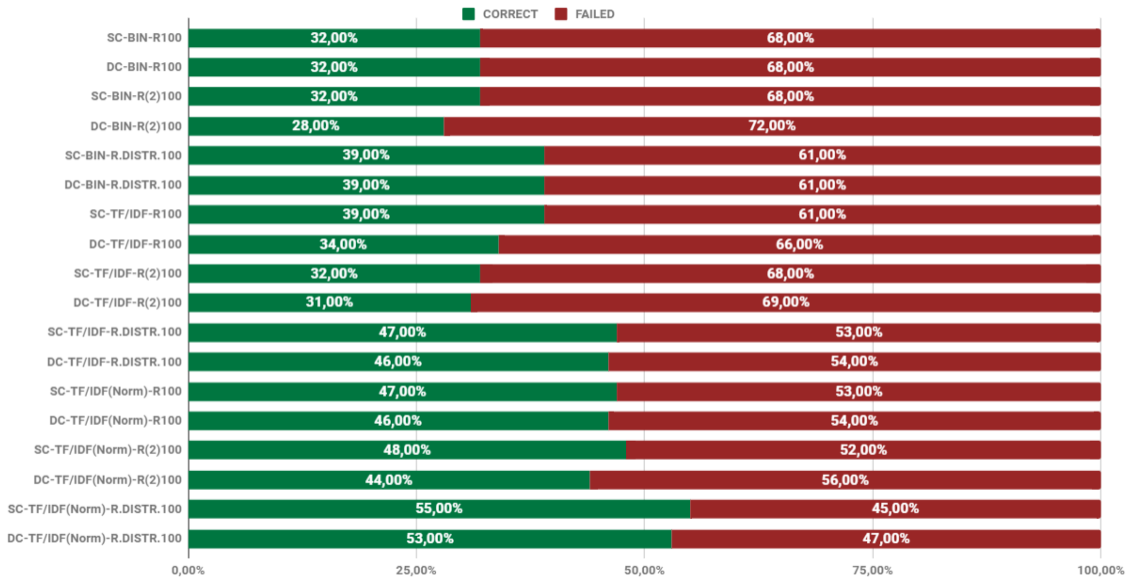


Figure 5.10: Comparisons for k-Nearest Neighbors(3) s at Single and Dual Classifiers [Threshold of 0.33]



We proceeded next to k-Nearest Neighbors algorithm. As we saw in chapter 4 from the illustrated tables, the k-value of 3 outperforms the value of 5 for our implementation with the accuracy rates of k(3) to be clearly better. The reason we selected the value of three and five for our k-parameter was because we run through each possible odd value in the range of 3 up to 15 and we saw that the two suitable values were only these two. However, we placed more emphasis on k(3) value since we have better results in comparison with k(5).

Despite the fact that k-Nearest Neighbors algorithm did not perform well enough compared to our other tested algorithms, it gave us important hints during our research. Since kNN is a distance based algorithm, it is sensitive to large scale features. While we were using Euclidean Distance as a measure metric (see 2.2.2) we saw that large feature-dimensions lead to incorrect classifications. To address, this we normalized our data which played an important role in prediction improvement [35] however, this improvement came with a major drawback which were outliers. Normalizing data will scale the ”normal” data at very small intervals so this arises our fault prediction rates in k-NN.

5.3 Experimental Results of Random 100 Distributed Data

Figure 5.11: Results of Random 100 Distributed Data at Single Classifier

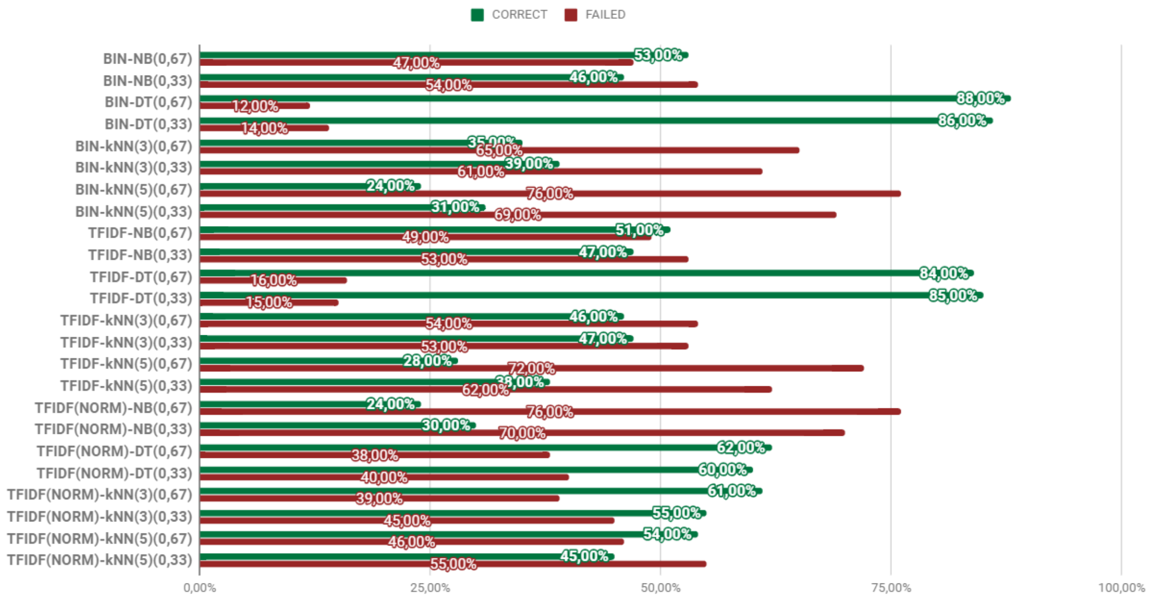
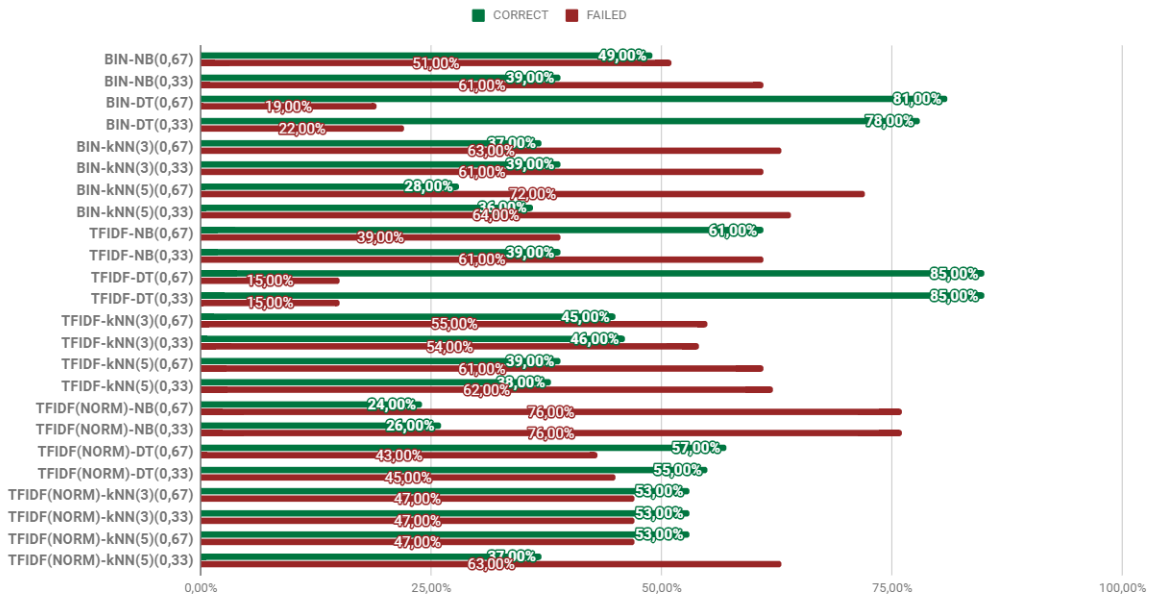


Figure 5.12: Results of Random 100 Distributed Data at Dual Classifiers

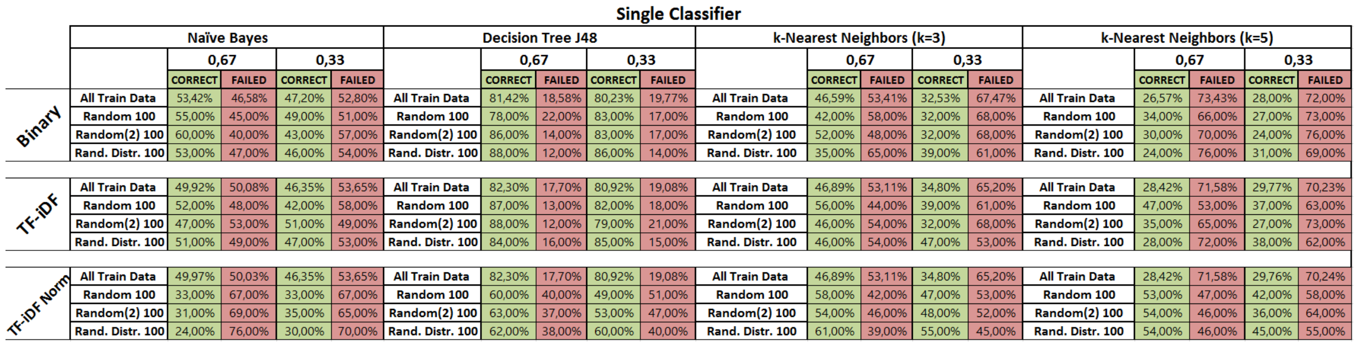


As previously seen we used a test data of random 100 data songs that were selected according to the distribution of the original training dataset. Although it had slightly better results in general compared to other random datasets, this assisted us mostly as a helping measure for our evaluation. We observed that the other random datasets follow the same distribution and they performed reasonably well, plus it gave us important tips for our future work.

5.4 Experimental Results in Total Tables

In this section illustrates the final total result tables for each classifier and each approach and algorithm that has been used on our project.

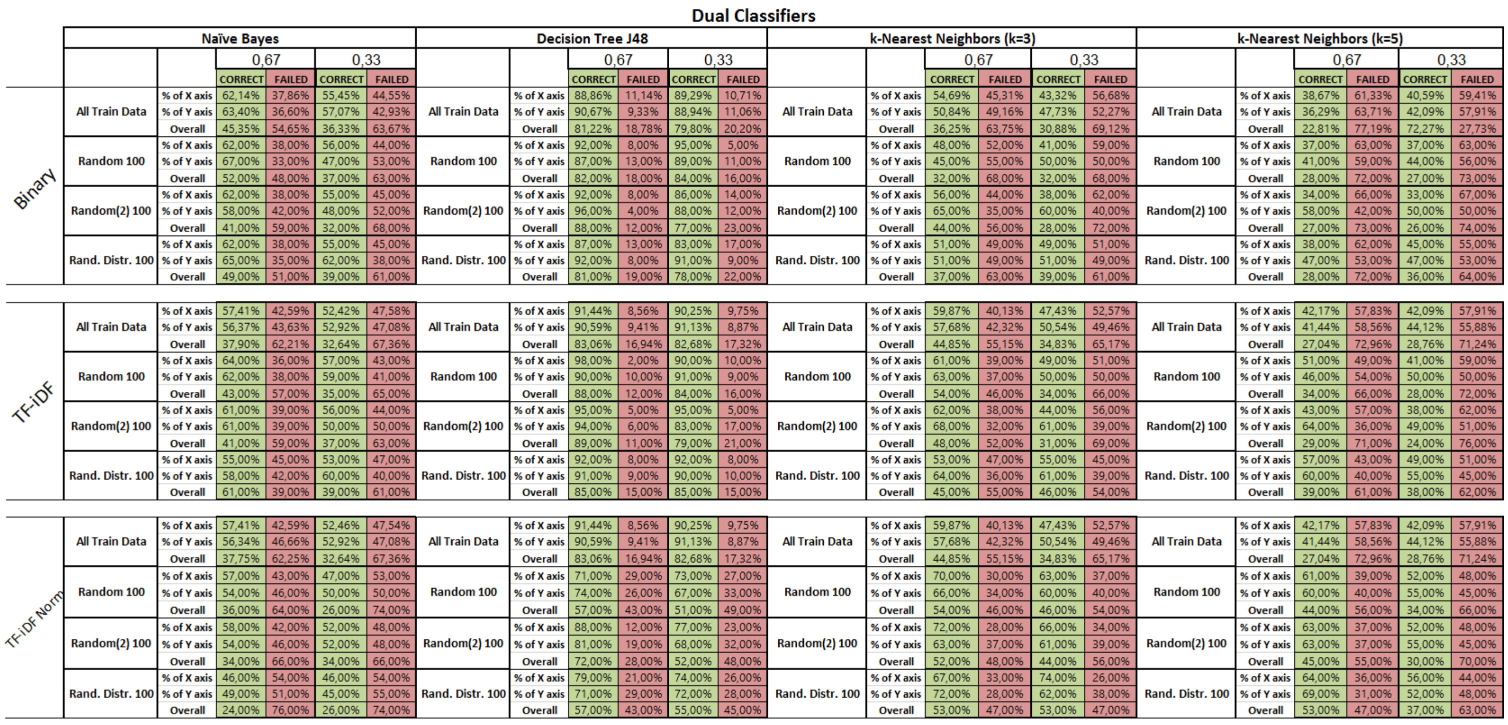
Figure 5.13: Results Total Table for Single Classifier Approach



Single Classifier

		Naïve Bayes				Decision Tree J48				k-Nearest Neighbors (k=3)				k-Nearest Neighbors (k=5)			
		0,67		0,33		0,67		0,33		0,67		0,33		0,67		0,33	
		CORRECT	FAILED	CORRECT	FAILED	CORRECT	FAILED	CORRECT	FAILED	CORRECT	FAILED	CORRECT	FAILED	CORRECT	FAILED	CORRECT	FAILED
Binary	All Train Data	53,42%	46,58%	47,20%	52,80%	81,42%	18,58%	80,23%	19,77%	46,59%	53,41%	32,53%	67,47%	26,57%	73,43%	28,00%	72,00%
	Random 100	55,00%	45,00%	49,00%	51,00%	78,00%	22,00%	83,00%	17,00%	42,00%	58,00%	32,00%	68,00%	34,00%	66,00%	27,00%	73,00%
	Random(2) 100	60,00%	40,00%	43,00%	57,00%	86,00%	14,00%	83,00%	17,00%	52,00%	48,00%	32,00%	68,00%	30,00%	70,00%	24,00%	76,00%
	Rand. Distr. 100	53,00%	47,00%	46,00%	54,00%	88,00%	12,00%	86,00%	14,00%	35,00%	65,00%	39,00%	61,00%	24,00%	76,00%	31,00%	69,00%
TF-iDF	All Train Data	49,92%	50,08%	46,35%	53,65%	82,30%	17,70%	80,92%	19,08%	46,89%	53,11%	34,80%	65,20%	28,42%	71,58%	29,77%	70,23%
	Random 100	52,00%	48,00%	42,00%	58,00%	87,00%	13,00%	82,00%	18,00%	56,00%	44,00%	39,00%	61,00%	47,00%	53,00%	37,00%	63,00%
	Random(2) 100	47,00%	53,00%	51,00%	49,00%	88,00%	12,00%	79,00%	21,00%	46,00%	54,00%	32,00%	68,00%	35,00%	65,00%	27,00%	73,00%
	Rand. Distr. 100	51,00%	49,00%	47,00%	53,00%	84,00%	16,00%	85,00%	15,00%	46,00%	54,00%	47,00%	53,00%	28,00%	72,00%	38,00%	62,00%
TF-iDF Norm	All Train Data	49,97%	50,03%	46,35%	53,65%	82,30%	17,70%	80,92%	19,08%	46,89%	53,11%	34,80%	65,20%	28,42%	71,58%	29,76%	70,24%
	Random 100	33,00%	67,00%	33,00%	67,00%	60,00%	40,00%	49,00%	51,00%	58,00%	42,00%	47,00%	53,00%	53,00%	47,00%	42,00%	58,00%
	Random(2) 100	31,00%	69,00%	35,00%	65,00%	63,00%	37,00%	53,00%	47,00%	54,00%	46,00%	48,00%	52,00%	54,00%	46,00%	36,00%	64,00%
	Rand. Distr. 100	24,00%	76,00%	30,00%	70,00%	62,00%	38,00%	60,00%	40,00%	61,00%	39,00%	55,00%	45,00%	54,00%	46,00%	45,00%	55,00%

Figure 5.14: Results Total Table for Dual Classifiers Approach



Dual Classifiers

			Naïve Bayes				Decision Tree J48				k-Nearest Neighbors (k=3)				k-Nearest Neighbors (k=5)			
			0,67		0,33		0,67		0,33		0,67		0,33		0,67		0,33	
			CORRECT	FAILED	CORRECT	FAILED	CORRECT	FAILED	CORRECT	FAILED	CORRECT	FAILED	CORRECT	FAILED	CORRECT	FAILED	CORRECT	FAILED
Binary	All Train Data	% of X axis	62,14%	37,86%	55,45%	44,55%	88,86%	11,14%	89,29%	10,71%	54,69%	45,31%	43,32%	56,68%	38,67%	61,33%	40,59%	59,41%
		% of Y axis	63,40%	36,60%	57,07%	42,93%	90,67%	9,33%	88,94%	11,06%	50,84%	49,16%	47,73%	52,27%	36,29%	63,71%	42,09%	57,91%
		Overall	45,35%	54,65%	36,33%	63,67%	81,22%	18,78%	79,80%	20,20%	36,25%	63,75%	30,88%	69,12%	22,81%	77,19%	72,27%	27,73%
	Random 100	% of X axis	62,00%	38,00%	56,00%	44,00%	92,00%	8,00%	95,00%	5,00%	48,00%	52,00%	41,00%	59,00%	37,00%	63,00%	37,00%	63,00%
		% of Y axis	67,00%	33,00%	47,00%	53,00%	87,00%	13,00%	89,00%	11,00%	45,00%	55,00%	50,00%	50,00%	41,00%	59,00%	44,00%	56,00%
		Overall	52,00%	48,00%	37,00%	63,00%	82,00%	18,00%	84,00%	16,00%	32,00%	68,00%	32,00%	68,00%	28,00%	72,00%	27,00%	73,00%
	Random(2) 100	% of X axis	62,00%	38,00%	55,00%	45,00%	92,00%	8,00%	86,00%	14,00%	56,00%	44,00%	38,00%	62,00%	34,00%	66,00%	33,00%	67,00%
		% of Y axis	58,00%	42,00%	48,00%	52,00%	96,00%	4,00%	88,00%	12,00%	65,00%	35,00%	60,00%	40,00%	58,00%	42,00%	50,00%	50,00%
		Overall	41,00%	59,00%	32,00%	68,00%	88,00%	12,00%	77,00%	23,00%	44,00%	56,00%	28,00%	72,00%	27,00%	73,00%	26,00%	74,00%
	Rand. Distr. 100	% of X axis	62,00%	38,00%	55,00%	45,00%	87,00%	13,00%	83,00%	17,00%	51,00%	49,00%	49,00%	51,00%	38,00%	62,00%	45,00%	55,00%
		% of Y axis	65,00%	35,00%	62,00%	38,00%	92,00%	8,00%	91,00%	9,00%	51,00%	49,00%	51,00%	49,00%	47,00%	53,00%	47,00%	53,00%
		Overall	49,00%	51,00%	39,00%	61,00%	81,00%	19,00%	78,00%	22,00%	37,00%	63,00%	39,00%	61,00%	28,00%	72,00%	36,00%	64,00%
TF-iDF	All Train Data	% of X axis	57,41%	42,59%	52,42%	47,58%	91,44%	8,56%	90,25%	9,75%	59,87%	40,13%	47,43%	52,57%	42,17%	57,83%	42,09%	57,91%
		% of Y axis	56,37%	43,63%	52,92%	47,08%	90,59%	9,41%	91,13%	8,87%	57,68%	42,32%	50,54%	49,46%	41,44%	58,56%	44,12%	55,88%
		Overall	37,90%	62,21%	32,64%	67,36%	83,06%	16,94%	82,68%	17,32%	44,85%	55,15%	34,83%	65,17%	27,04%	72,96%	28,76%	71,24%
	Random 100	% of X axis	64,00%	36,00%	57,00%	43,00%	98,00%	2,00%	90,00%	10,00%	61,00%	39,00%	49,00%	51,00%	51,00%	49,00%	41,00%	59,00%
		% of Y axis	62,00%	38,00%	59,00%	41,00%	90,00%	10,00%	91,00%	9,00%	63,00%	37,00%	50,00%	50,00%	46,00%	54,00%	50,00%	50,00%
		Overall	43,00%	57,00%	35,00%	65,00%	88,00%	12,00%	84,00%	16,00%	54,00%	46,00%	34,00%	66,00%	34,00%	66,00%	28,00%	72,00%
	Random(2) 100	% of X axis	61,00%	39,00%	56,00%	44,00%	95,00%	5,00%	95,00%	5,00%	62,00%	38,00%	44,00%	56,00%	43,00%	57,00%	38,00%	62,00%
		% of Y axis	61,00%	39,00%	50,00%	50,00%	94,00%	6,00%	83,00%	17,00%	68,00%	32,00%	61,00%	39,00%	64,00%	36,00%	49,00%	51,00%
		Overall	41,00%	59,00%	37,00%	63,00%	89,00%	11,00%	79,00%	21,00%	48,00%	52,00%	31,00%	69,00%	29,00%	71,00%	24,00%	76,00%
	Rand. Distr. 100	% of X axis	55,00%	45,00%	53,00%	47,00%	92,00%	8,00%	92,00%	8,00%	53,00%	47,00%	55,00%	45,00%	57,00%	43,00%	49,00%	51,00%
		% of Y axis	58,00%	42,00%	60,00%	40,00%	91,00%	9,00%	90,00%	10,00%	64,00%	36,00%	61,00%	39,00%	60,00%	40,00%	55,00%	45,00%
		Overall	61,00%	39,00%	39,00%	61,00%	85,00%	15,00%	85,00%	15,00%	45,00%	55,00%	46,00%	54,00%	39,00%	61,00%	38,00%	62,00%
TF-iDF Norm	All Train Data	% of X axis	57,41%	42,59%	52,46%	47,54%	91,44%	8,56%	90,25%	9,75%	59,87%	40,13%	47,43%	52,57%	42,17%	57,83%	42,09%	57,91%
		% of Y axis	56,34%	46,66%	52,92%	47,08%	90,59%	9,41%	91,13%	8,87%	57,68%	42,32%	50,54%	49,46%	41,44%	58,56%	44,12%	55,88%
		Overall	37,75%	62,25%	32,64%	67,36%	83,06%	16,94%	82,68%	17,32%	44,85%	55,15%	34,83%	65,17%	27,04%	72,96%	28,76%	71,24%
	Random 100	% of X axis	57,00%	43,00%	47,00%	53,00%	71,00%	29,00%	73,00%	27,00%	70,00%	30,00%	63,00%	37,00%	61,00%	39,00%	52,00%	48,00%
		% of Y axis	54,00%	46,00%	50,00%	50,00%	74,00%	26,00%	67,00%	33,00%	66,00%	34,00%	60,00%	40,00%	60,00%	40,00%	55,00%	45,00%
		Overall	36,00%	64,00%	26,00%	74,00%	57,00%	43,00%	51,00%	49,00%	54,00%	46,00%	46,00%	54,00%	44,00%	56,00%	34,00%	66,00%
	Random(2) 100	% of X axis	58,00%	42,00%	52,00%	48,00%	88,00%	12,00%	77,00%	23,00%	72,00%	28,00%	66,00%	34,00%	63,00%	37,00%	52,00%	48,00%
		% of Y axis	54,00%	46,00%	52,00%	48,00%	81,00%	19,00%	68,00%	32,00%	63,00%	37,00%	61,00%	39,00%	63,00%	37,00%	55,00%	45,00%
		Overall	34,00%	66,00%	34,00%	66,00%	72,00%	28,00%	52,00%	48,00%	52,00%	48,00%	44,00%	56,00%	45,00%	55,00%	30,00%	70,00%
	Rand. Distr. 100	% of X axis	46,00%	54,00%	46,00%	54,00%	79,00%	21,00%	74,00%	26,00%	67,00%	33,00%	74,00%	26,00%	64,00%	36,00%	56,00%	44,00%
		% of Y axis	49,00%	51,00%	45,00%	55,00%	71,00%	29,00%	72,00%	28,00%	72,00%	28,00%	62,00%	38,00%	69,00%	31,00%	52,00%	48,00%
		Overall	24,00%	76,00%	26,00%	74,00%	57,00%	43,00%	55,00%	45,00%	53,00%	47,00%	53,00%	47,00%	53,00%	47,00%	37,00%	63,00%

Chapter 6

Conclusions

One of the conclusions that we drew is that the meaning of the data varies and relies not only on audio-criteria but also on the song's origin. For example, words that may have positive meaning in one song, will most likely have a negative meaning in another song. For instance, we have a song which says ”I love you” with a positive label assigned and another song which says ”I don't love you” with a negative label. After stemming and stopwords removal processes, what remains is the word ”love” for both instances but for each one has a different meaning. This negation is a very common linguistic construction that affects word/sentence polarity. That causes a confusion to the classifier and in order to improve the classification accuracy of this, we need to gather a larger dataset with much more information regarding songs and, therefore, the emotions of them. Hence, an interesting technique that can be tested for this would be n-gram model, which splits the text into sentences with each one to containing n-words or TF-iDF Word Negation approach.[7]

However, this is connected not only by the nature of the song but also by the annotations of the users that rated these songs in the first place. The subjectivity of the users had, as a result, the unclear label definition for many songs so that made predictions even more complex. That happened because the raters appeared to be highly influenced by the musical features of the songs, as we saw from their ratings during their classification annotations that they made. So, since we observed that audio features play a significant role in songs's emotional state and that the lyrics is a strong and useful tool to use towards to the extraction of this information, it is necessary to say that we must include audio processing in an expanded version of this project in order to improve the performance.

Empirically we see that the best algorithm we used was J48 Decision Tree which turned out to be logical since Decision Trees are low-bias and high-variance models. Additionally, we have ensured that we have avoided overfitting due to post-pruning process as we see the rates on the training data to fluctuate from 80% to 90%. So J48 tends to learn our data fairly well compared to other tested algorithms. Although J48's performance was outstanding, the selection of a different training dataset is a very important factor in machine learning, especially in emotional analysis, and as a result, the selection of the appropriate algorithm may differ because of it.

We have come to the conclusion that the lyrics are related to the emotional state of the songs and it seems that the users are highly influenced by that for their estimations. Although the evaluation in Music Information Retrieval is really hard[28], our completion came with a decent implementation and usage of simple models such as TF-iDF and Decision Trees which seems promising for further investigation (see section 6.1) in combination with advanced natural processing techniques and algorithms such as Neural Networks. Lastly, it is worth to mentioning that this can be used for transfer learning implementations.

6.1 Future Work

Our future work should be focused on the improvement of classification predictions by testing different algorithms such as Neural Networks, Vector Space Models or Hidden Markov Models as well as investigating this implementation with multi-modal learning models for example, advanced natural language processing on lyrics correlated with analysis on audio features like rhythm, tempo, loudness etc.
As briefly mentioned in the previous chapter section, an interesting improvement would be unigrams, bigrams and n-grams in general. This model manipulates a sentence as a group of n-words so it is worth observing if stopwords play any role in this approach.
Another technique that we are curious to test is scoring-words, in which you can give to a word a score in a specific range which describes negative(for low values), neutral (for middle values) and positive (for high values). As each word is assigned with a score, the summary of the words will correspond to a relative class. Perhaps an extended version of n-grams can be developed in combination with this technique.
Last but not least, a larger training dataset with more raters in order to avoid the subjectivity of the users would be much more preferable.

Bibliography

[1] Olabenjo Babatunde. “Applying Naive Bayes Classification to Google Play Apps Categorization”. In: *CoRR* abs/1608.08574 (2016). arXiv: 1608.08574. URL: http://arxiv.org/abs/1608.08574.

[2] Joeran Beel, Bela Gipp, Stefan Langer, and Corinna Breitinger. “Research-paper recommender systems: a literature survey”. In: *International Journal on Digital Libraries* 17.4 (Nov. 1, 2016), pp. 305–338. ISSN: 1432-1300. DOI: 10.1007/s00799-015-0156-0. URL: https://doi.org/10.1007/s00799-015-0156-0.

[3] M. Besson, F. Faïta, I. Peretz, A.-M. Bonnel, and J. Requin. “Singing in the Brain: Independence of Lyrics and Tunes”. In: *Psychological Science* 9.6 (1998), pp. 494–498. DOI: 10.1111/1467-9280.00091. URL: https://doi.org/10.1111/1467-9280.00091.

[4] P. Chandrasekar, K. Qian, H. Shahriar, and P. Bhattacharya. “Improving the Prediction Accuracy of Decision Tree Mining with Data Preprocessing”. In: *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*. Vol. 2. July 2017, pp. 481–484. DOI: 10.1109/COMPSAC.2017.146.

[5] Massimo Ciociola. *Musixmatch*. 2010. URL: https://www.musixmatch.com/.

[6] CBS Corporation. *Last Fm*. 2002. URL: https://www.last.fm/.

[7] Bijoyan Das and Sarit Chakraborty. “An Improved Text Sentiment Classification Model Using TF-IDF and Next Word Negation”. In: *CoRR* abs/1806.06407 (2018).

[8] Asoke Kumar Datta, Sandeep Singh Solanki, Ranjan Sengupta, Soubhik Chakraborty, Kartik Mahto, and Anirban Patranabis. *Signal Analysis of Hindustani Classical Music*. 1st. Springer Publishing Company, Incorporated, 2017. ISBN: 9789811039584, 9811039585.

[9] Pedro M. Domingos and Michael J. Pazzani. “Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier”. In: *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*. ICML’96. Bari, Italy: Morgan Kaufmann Publishers Inc., 1996, pp. 105–112. ISBN: 1-55860-419-7. URL: http://dl.acm.org/citation.cfm?id=3091696.3091710.

[10] Mark A. Hall Eibe Frank and Ian H. Witten (2016). “The WEKA Workbench. Online Appendix for ”Data Mining: Practical Machine Learning Tools and Techniques””. In: (2016). URL: https://www.cs.waikato.ac.nz/ml/weka/.

[11] Daniel Ek. *Spotify*. 2008. URL: https://www.spotify.com/.

[12] Paul Ekman. “An argument for basic emotions”. In: *Cognition and Emotion* 6.3-4 (1992), pp. 169–200. DOI: 10.1080/02699939208411068. eprint: https://doi.org/10.1080/02699939208411068. URL: https://doi.org/10.1080/02699939208411068.

[13] MetaBrainz Foundation. *MusicBrainz*. 2000. URL: https://musicbrainz.org/.

[14] Kate Hevner. “Experimental Studies of the Elements of Expression in Music”. In: *The American Journal of Psychology* 48.2 (1936), pp. 246–268. ISSN: 00029556. URL: http://www.jstor.org/stable/1415746.

[15] Xiao Hu and J. Stephen Downie. “Improving Mood Classification in Music Digital Libraries by Combining Lyrics and Audio”. In: *Proceedings of the 10th Annual Joint Conference on Digital Libraries*. JCDL ’10. Gold Coast, Queensland, Australia: ACM, 2010, pp. 159–168. ISBN: 978-1-4503-0085-8. DOI: 10.1145/1816123.1816146. URL: http://doi.acm.org/10.1145/1816123.1816146.

[16] *Jersey REST and Javax*. URL: https://jersey.github.io/.

[17] Patrik N. Juslin and Petri Laukka. “Expression, Perception, and Induction of Musical Emotions: A Review and a Questionnaire Study of Everyday Listening”. In: *Journal of New Music Research* 33.3 (2004), pp. 217–238. DOI: 10.1080/0929821042000317813. eprint: https://doi.org/10.1080/0929821042000317813. URL: https://doi.org/10.1080/0929821042000317813.

[18] Patrik N. Juslin and John A. Sloboda. *Music and Emotion: Theory and Research*. Oxford University Press, 2001.

[19] Gaganjot Kaur and Amit Chhabra. “Article: Improved J48 Classification Algorithm for the Prediction of Diabetes”. In: *International Journal of Computer Applications* 98.22 (July 2014). Full text available, pp. 13–17.

[20] Joanna Kazmierska and Julian Malicki. “Application of the Naïve Bayesian Classifier to optimize treatment decisions”. In: *Radiotherapy and Oncology* 86.2 (2008), pp. 211–216. ISSN: 0167-8140. DOI: https://doi.org/10.1016/j.radonc.2007.10.019. URL: http://www.sciencedirect.com/science/article/pii/S0167814007005221.

[21] Cyril Laurier, Jens Grivolla, and Perfecto Herrera. “Multimodal Music Mood Classification Using Audio and Lyrics”. In: *Proceedings of the 2008 Seventh International Conference on Machine Learning and Applications*. ICMLA ’08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 688–693. ISBN: 978-0-7695-3495-4. DOI: 10.1109/ICMLA.2008.96. URL: https://doi.org/10.1109/ICMLA.2008.96.

[22] Owen Craigie Meyers, Barry Vercoe, and Owen Craigie Meyers. *A Mood-Based Music Classification and Exploration System*. 2007.

[23] Oracle. *Data Mining Concepts: Classification*. [Online; released 2005]. 1977. URL: https://docs.oracle.com/cd/B28359_01/datamine.111/b28129/classify.htm#i1005746.

[24] R. W. Picard, E. Vyzas, and J. Healey. “Toward machine emotional intelligence: analysis of affective physiological state”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23.10 (Oct. 2001), pp. 1175–1191. ISSN: 0162-8828. DOI: 10.1109/34.954607.

[25] Associate Professor in Computer Engineering Pier Luca Lanzi. *Machine Learning and Data Mining: 10 Introduction to Classification*. [Online in SlideShare; created 2007]. 2007. URL: https://www.slideshare.net/pierluca.lanzi/machine-learning-and-data-mining-10-introduction-to-classification.

[26] J. A. Russell. “A circumplex model of affect.” In: *Journal of Personality and Social Psychology* 39 (1980), pp. 1161–1178. URL: http://dx.doi.org/10.1037/h0077714.

[27] Steven L. Salzberg. “C4.5: Programs for Machine Learning by J. Ross Quinlan. Morgan Kaufmann Publishers, Inc., 1993”. In: *Machine Learning* 16.3 (Sept. 1, 1994), pp. 235–240. ISSN: 1573-0565. DOI: 10.1007/BF00993309. URL: https://doi.org/10.1007/BF00993309.

[28] Markus Schedl, Emilia Gómez, and Julián Urbano. “Music Information Retrieval: Recent Developments and Applications”. In: *Foundations and Trends® in Information Retrieval* 8.2-3 (2014), pp. 127–261. ISSN: 1554-0669. DOI: 10.1561/1500000042. URL: http://dx.doi.org/10.1561/1500000042.

[29] Emery Schubert. “Update of the Hevner Adjective Checklist”. In: *Perceptual and Motor Skills* 96.3_suppl (2003). PMID: 12929763, pp. 1117–1122. DOI: 10.2466/pms.2003.96.3c.1117. eprint: https://doi.org/10.2466/pms.2003.96.3c.1117. URL: https://doi.org/10.2466/pms.2003.96.3c.1117.

[30] B. Schuller, J. Dorfner, and G. Rigoll. “Determination of Non-Prototypical Valence and Arousal in Popular Music: Features and Performances”. In: *EURASIP Journal on Audio, Speech, and Music Processing (JASMP), Special Issue on ”Scalable Audio-Content Analysis* Article ID 735854 (2010). Hindawi Publ. Corp., 19 pages.

[31] Auke Tellegen, David Watson, and Lee Anna Clark. “On the Dimensional and Hierarchical Structure of Affect”. In: *Psychological Science* 10.4 (1999), pp. 297–303. DOI: 10.1111/1467-9280.00157. eprint: https://doi.org/10.1111/1467-9280.00157. URL: https://doi.org/10.1111/1467-9280.00157.

[32] Wikipedia contributors. *Decision tree learning — Wikipedia, The Free Encyclopedia*. [Online; accessed 16-November-2018]. 2018. URL: https://en.wikipedia.org/w/index.php?title=Decision_tree_learning&oldid=869075485.

[33] Wikipedia contributors. *Divide and conquer algorithm — Wikipedia, The Free Encyclopedia*. [Online; accessed 18-November-2018]. 2018. URL: https://en.wikipedia.org/w/index.php?title=Divide_and_conquer_algorithm&oldid=867712321.

[34] Wikipedia contributors. *Euclidean distance — Wikipedia, The Free Encyclopedia*. [Online; accessed 16-November-2018]. 2018. URL: https://en.wikipedia.org/w/index.php?title=Euclidean_distance&oldid=862708907.

[35] Wikipedia contributors. *Feature scaling — Wikipedia, The Free Encyclopedia*. [Online; accessed 18-November-2018]. 2018. URL: https://en.wikipedia.org/w/index.php?title=Feature_scaling&oldid=868504514.

[36] Wikipedia contributors. *ID3 algorithm — Wikipedia, The Free Encyclopedia*. [Online; accessed 16-November-2018]. 2018. URL: https://en.wikipedia.org/w/index.php?title=ID3_algorithm&oldid=864293085.

[37] Wikipedia contributors. *JSON — Wikipedia, The Free Encyclopedia*. [Online; accessed 16-November-2018]. 2018. URL: https://en.wikipedia.org/w/index.php?title=JSON&oldid=869165672.

[38] Wikipedia contributors. *K-nearest neighbors algorithm — Wikipedia, The Free Encyclopedia*. [Online; accessed 16-November-2018]. 2018. URL: https://en.wikipedia.org/w/index.php?title=K-nearest_neighbors_algorithm&oldid=867125760.

[39] Wikipedia contributors. *Naive Bayes spam filtering — Wikipedia, The Free Encyclopedia*. [Online; accessed 16-November-2018]. 2018. URL: https://en.wikipedia.org/w/index.php?title=Naive_Bayes_spam_filtering&oldid=863941718.

[40] Wikipedia contributors. *Random forest — Wikipedia, The Free Encyclopedia*. [Online; accessed 16-November-2018]. 2018. URL: https://en.wikipedia.org/w/index.php?title=Random_forest&oldid=867261266.

[41] Dan Yang and Won-Sook Lee. "Music Emotion Identification from Lyrics". In: *2009 11th IEEE International Symposium on Multimedia* (2009), pp. 624–629.

[42] Yi-Hsuan Yang and Homer H. Chen. "Machine Recognition of Music Emotion: A Review". In: *ACM Trans. Intell. Syst. Technol.* 3.3 (May 2012), 40:1–40:30. ISSN: 2157-6904. DOI: 10.1145/2168752.2168754. URL: http://doi.acm.org/10.1145/2168752.2168754.