Individual Diploma Thesis

NON-INTRUSIVE PHYSIOLOGICAL WEARABLE DEVICES FOR IDENTIFYING INDIVIDUAL DIFFERENCE PARAMETERS USING SUPERVISED CLASSIFICATION LEARNING ALGORITHMS

Christoforos Galazis

UNIVERSITY OF CYPRUS

DEPARTMENT OF COMPUTER SCIENCE

May 2017

UNIVERSITY OF CYPRUS

DEPARTMENT OF COMPUTER SCIENCE

Non-Intrusive Physiological Wearable Devices for Identifying Individual Difference Parameters using Supervised Classification Learning Algorithms

Christoforos Galazis

Supervisor

Chryssis Georgiou

The Individual Diploma Thesis was submitted towards partially meeting the requirements for obtaining the degree of Computer Science of the Department of Computer Science of the University of Cyprus

May 2017

Acknowledgments

I would like to express, first and foremost, my gratitude to my supervising professor Chryssis Georgiou for trusting me with this thesis topic and also for the support and guidance to see it through. Also, I would like to thank professor Maria Karekla and the Department of Psychology of University of Cyprus for offering experimental data samples that were already analysed to determine the classification of each sample. This data was used to experiment with and obtain a better understanding of the classification algorithms described in this thesis.

Abstract

The scope of this thesis was to explore non-intrusive wearable devices that can obtain biosignal information from the wearer. The sensors the devices must have in combination is electrocardiography or photoplethysmography, galvanic skin response and electroencephalography. Also, they must offer an easy way to extract the data so they can in turn be processed and analysed. From the market research it was concluded that the devices *MS Band 2* and *Muse* were the most suitable for this case. Because the devices do not offer a way to directly access the sensors' data, an Android application was built, which is responsible for communicating with the devices, retrieving the raw signal and storing the data.

After collecting the data from the devices, each of the raw signals were extracted and the data stream was summarised in a set of features using statistical methods. The data vector created was then used in classification algorithms so it can predict cases of high or low risk regarding a specific medical condition. This process was implemented in a Python application to automate the analysis of a received data sample.

To better understand the classification algorithms and to determine initially which one will be used for the above methods three separate datasets were analysed, given from the Department of Psychology of the University of Cyprus. The first dataset contained information on experiential avoidance of smokers and for each sample they are classified as either avoidance or acceptance. The second dataset contained information on diagnosis of eating disorders and for each sample they are classified as either high or low risk. Finally, the last data set contained information on experiential avoidance for anxiety and are classified as either high or low risk.

For all three of the datasets a total of 10 supervised classification algorithms were used to compare their effectiveness between them. For the first case, the algorithms that had the best performance were *K-Nearest Neighbours*, *Naive Bayes*, *Support Vector Machine* and *Logistic Regression* which perform well on linear correlated data sets and for smaller data sets. For the second case the algorithms with the highest results were *Random Forest* and *Support Vector Machine*, which can handle better linear data sets that have noise and outliers. Finally, the last case the algorithm that had the highest accuracy was the *Random Forest* as for it can handle a larger number of features and irrelevant features.

Table of Contents

Chapter 1

Introduction

1.1 Motive

The main motive is to explore the capability to monitor peoples' different bio signals so it can be used to identify possible health problems and provide assistance in coping with them. Finding faster and more accurately conditions or health problems of a person also leads to a more rapid response and management, improving the quality of life of the person. For a lot of cases finding a diagnosis can be long and tedious procedure, requiring a lot of tests, lab visits and constant communication with multiple doctors. While nothing can replace the consultancy of a doctor, one way to reduce the time consumed can be done by monitoring their bio signals from home rather than at a lab. This can be achieved by taking advantage of the recently rapid development of wearable devices that monitor the wearer. This will allow to monitor throughout the day the patient from the comfort of their home or while continuing normally their daily activities. Also, it can detect some irregularities from the monitored bio signals that otherwise wouldn't be detected from the small duration while being monitored at a lab.

In recent years, development of nonintrusive wearable devices used to measure different bio signals has increased and more companies are competing in this market. The main push for the need of these devices is people in today's society are more health-aware and seek to be more fit. This has a result of more competitive and more accurate

devices are available on the market and are becoming more affordable. In turn, this allows such devices to be used effectively for monitoring patients, with some being more suitable than others.

While the volume of data is increasing both from the larger number of people that can be monitored at a single time and from the prolonged monitoring the data cannot be effectively analysed individually by hand. This is where machine learning algorithms [55] come in to help automate the procedure. By having manually analysed a few data samples and identifying their class this opens the road in using machine learning algorithms. For identifying medical problems classification algorithms can be used to predict whether a person is at high risk or low risk of a specific condition or illness within a margin of error.

1.2 Purpose

The purpose of this thesis is to explore what currently wearable devices are available in the market that can collect data from the wearer without being intrusive. The data collected from the wearer while monitoring them throughout the day will be used to find any possible correlations to psychological or medical related problems and in turn to find ways to improve their quality of life. Finding correlations will be done through extracting statistical features from the data stream and using it in a classification algorithm for predictions. The device or devices that are suitable are ones that can obtain data from the signals photoplethymography or electrocardiography, galvanic skin response and electroencephalography. In addition, they should be able to export the raw signal data so it can be processed and stored.

To better understand the main classification algorithms used today, the thesis studies three data sets. The data sets have been obtained from the Department of Psychology of University of Cyprus and contain samples for experiential avoidance of smokers, diagnosis of eating disorders and diagnosis of experiential avoidance for anxiety. Using a broad range of classification algorithms to attempt to accurately predicted the classified class of each data sample, with being acceptance or avoidance for the first and high risk or low risk for the last two data sets. Having a highly accurate

classification algorithm we can use it to automate and hasten the process of identifying health conditions in people.

1.3 Methodology

The methodology followed was to initially identify and understand the different machine learning algorithms available, with a more focus and in depth look on classification algorithms. All classification algorithms are taken from libraries that are implemented in Python. Having a preference on code implemented in python rather than any other programming language as for it is highly used for data analyses and machine learning, having the benefit on focusing on the implementation and the increase speed of development rather than the code performance. The classification algorithms are evaluated using the three data sets obtained from the Department of Psychology. Each algorithm was evaluated based on specific metrics from the output results. Using these data examples they allow to obtain a better idea on how they work and how effective, in general, the algorithms are, so one of them can be used for when receiving data from the wearable devices.

A market research is needed to be conducted on the availability or feature availability of wearable devices that can be used for the purposes of the study. The device or devices that will be used will dictate the method on which the data will be extracted. Also, because the data received is a continuous stream of data, even though it is static, it can't be used as such in classification problem to make predictions. So there is a need to explore ways of extracting features and summarising the characteristics from the data stream to a specific number of features.

1.4 Thesis Organisation

The thesis is separated into eight chapters including the current one. The following seven chapters are:

- Chapter 2 documents different supervised and unsupervised algorithms, their general functions and their main advantages and disadvantages for using each one of them.
- Chapter 3 explores ways to evaluate the algorithms between them, based on a specific data set, with a main focus on classification techniques.
- Chapter 4 briefly describes the different sensors used in the two experimental data sets and the ones required by the wearable devices.
- Chapter 5 analyses and compares the classification algorithms on the three data sets.
- Chapter 6 documents all the currently available wearable devices, the ones that are selected to be used and how their data from the sensors are extracted to be further used.
- Chapter 7 describes the console application for computers that will receive the data from the wearable devices, extract statistical features and use to make predictions.
- Chapter 8 concludes this thesis and states the main problems needed to overcome in the course of this work. Also, some general directions for future expansions are presented.

Chapter 2

Supervised and Unsupervised Machine Learning Algorithms

2.1 Supervised Learning

Supervised learning [18] is a method used in machine learning [55] with the purpose of either classifying or either predicting a mean value for a given vector of input data. The model which will be used must be first trained with a known data set before it can make predictions over unknown data. This known data set is composed of the input vector, also referred as the feature set, and the expected output of the model for each sample.

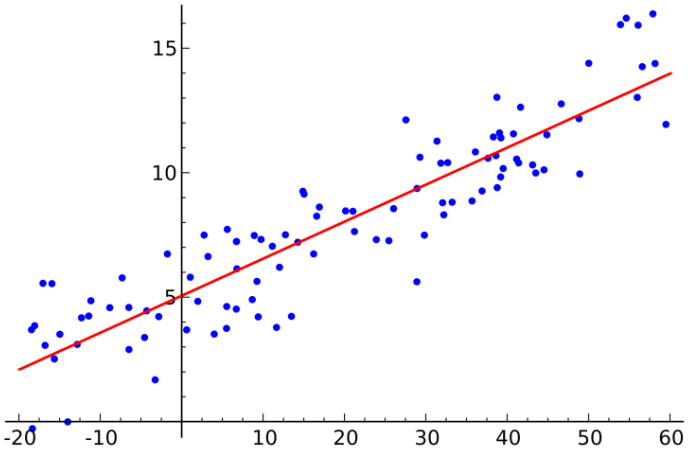
This data is separated to the training data set and the testing/validation data. The training data is used to create an internal mapping or weight adjustment based on the input vector. The internal variables are changed in such a way to reduce the error rate of the estimated output compared to the actual output of the model, so it can make better predictions on unknown data. The testing data is not used in the training phase and so they are unknown to the model created, but still have the actual output. This data set is used to calculate the models' actual prediction accuracy which can be used to compare it to other models and also gives a general idea of the confidence level for future unknown data samples.

The supervised learning is split into two subcategories, classification and regression, based on the required output of the model. For a classification algorithm the actual and predicted output is a label which can have two or more classes, usually represented as integer values. On the other hand, a regression algorithm doesn't have class labels as output but continuous values. Also, based on the algorithm they can solve linear and/or non linear problems [18].

2.1.1 Linear Regression

Linear regression is used to describe the relation between one (simple linear regression) or more (multiple linear regression) dependent variable with one or more independent variables. This model is capable of solving regression problems, that is, predicting a real value, and linear problems. Based on the training data, the algorithm has a target of creating a straight line (simple regression) (Graph 2.1) or a flat surface

(multiple regression) that crosses as best as possible the training input vectors when placed on a graph, so that that it can predict the output for each new sample [23][18].



Graph 2.1. Simple Linear Regression [44]

Simple Linear Regression:

The equation for the linear relation with one dependent variable is:

$y = \beta_0 + \beta_1 x + \varepsilon$, where

y: predicted output variable

x: input variable

β_0: constant that expresses the point of contact of the line with the y axis

β_1: regression coefficient

ε: random error

Multiple Linear Regression:

In case that the linear relation is dependent on more than one variable (p >1), then the equation is:

$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p + \varepsilon.$

Because $\beta_{0,}$ β_1, β_2 , ... and β_p are unknown coefficients, it is required to make some predictions $(\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p)$ from the training data and so we get the equation $\hat{y}$ which is an estimation of y:

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_{i1} + \hat{\beta}_2 x_{i2} + \ldots + \hat{\beta}_p x_{ip}$$

For both cases, in order to find the optimal straight line, the method of the least squared distance for each pair of points is used $\{x_i, y_i\}$ [23]:

$$RSS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1} - \hat{\beta}_2 x_{i2} - \ldots - \hat{\beta}_p x_{ip})^2$$

Advantages:

- Has optimal results when the input data are purely linear.
- Easy to represent the data on a graph.
- It's fast to train the model and also in estimating the results for a given data sample.

Disadvantages:

- Can only solve linear and regression problems.
- Has for most cases low accuracy results compared to other regression algorithms.
- It assumes that the features of the data is independent of one another.
- It's sensitive to noise and outlier data [23][12][42].

2.1.2 Logistic Regression

Logistic regression [23] is the generalisation of linear regression so that it can also handle classification problems. It uses one or more independent input variables so that it can predict the probability of it to be classified as one of the classes defined for the problem. For this to be achieved a sigmoid line is created from a logarithmic function [23].

For the case where there are only binary results for a specific problem, either being 0 or 1, the probability of entering input data *X* to be classified as *Y=1* is found by the following equation:

$p(X) = Pr(Y = 1|X) = \frac{e^{\beta_0+\beta_1 X}}{1+ e^{\beta_0+\beta_1 X}}$, with

β_0: constant that expresses the point of contact of the line with the y axis

β_1: regression coefficient.

From the above equation we can take the logarithmic equation to create the curve in the diagram:

$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X.$

Similarly with linear regression, the coefficients β_0 and β_1 are not known and have to be estimated ($\hat{\beta}_0$ and $\hat{\beta}_1$), so we can take the estimation probability $\hat{p}(X)$. For the estimation, the method of maximum likelihood is used. This method tries to find the coefficients $\hat{\beta}_0$ and $\hat{\beta}_1$ in such a way that it will find the smallest deviation between the observation and prediction values.

$L(\beta_0, \beta_1) = \prod_{i=1}^{n} p(x_i)^{y_i}(1 - p(x_i))^{1-y_i}.$

In case we have *p* prognostic factors, the equations are adjusted as follows (for X = $(X_1, X_2, \dots, X_p)$) [23]:

$$p(X) = \frac{e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}}{1 + e^{\beta_0+\beta_1 X_1+\cdots+\beta_p X_p}},$$

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X_1 + \dots + \beta_p X_p \text{ and}$$

$$L(\beta_0, \beta_{1,\dots,} \beta_p) = \prod_{i=1}^{n} p(x_i)^{y_i}(1 - p(x_i))^{1-y_i}.$$

Advantages:

- Except from the classification result it also shows with what probability it was selected.
- Fast training of the model and fast prediction of the output result.
- Has a low variance.

Disadvantages:

- Only solves linear classification problems.
- The accuracy of the model is dependent from the independent coefficient values.
- It has a low tolerance to irrelevant features and to high noise and outliers [23][12][42].

2.1.3 Naive Bayes

The *Naive Bayes classifier* [23] is based on the theorem of Bayes [41] so it can classify an input vector to the appropriate class based on the analysis of the probabilities of the training data. This classifier assumes that the input features of the data are strongly independent from each other [23].

For a class problem *C* where there are in total *k* class labels, to estimate in which class a new data set sample *x* is in, which has in total *n* features ($x = (x_1, x_2, \dots, x_n)$) the following equation is used to find the probability:

$$P(C_k|x) = \frac{P(C_k)P(x|C_k)}{P(x)}$$

Since the probability for *x* is independent of the class and its features have a fixed value it can be written as a constant *z*. The chain rule can be used for the above equation because the denominator doesn't depend on the class *C*:

$$\begin{aligned} P(C_k)P(x|C_k) &= P(C_k, x_1, x_2, \dots, x_n) = P(x_1, x_2, \dots, x_n, C_k) \\ &= P(x_1 | x_2, \dots, x_n, C_k)\, P(x_2 | x_3, \dots, x_n, C_k) \dots P(x_n | C_k)\, P(C_k). \end{aligned}$$

Having each feature as independent from on another we get:

$$P(C_k)P(x|C_k) \approx P(\ C_k)\ P(x_1|\ C_k)\ P(x_2\ |\ C_k) \dots\ P(x_n\ |\ C_k)\ =\ P(\ C_k) \prod_{i=1}^{n} P(x_i\ |C_k),$$

and so the final equation can be written as [23][40]:

$$P(C_k\ |\ x_1, x_2, \dots, x_n) = \frac{1}{Z} P(\ C_k) \prod_{i=1}^{n} P(x_i\ |C_k).$$

Advantages:

- Handles cases where the result must be estimated to a class.
- Can be trained with a small data sample.
- Not resource demanding and so it is fast to train the model and estimate the output results.
- Performs well even when having a problem requiring to estimate more than two classes.
- Has better performance than logistic regression when the features of the data can be easily separated.
- It has high tolerance to irrelevant features and missing values.
- Easy to understand and represent the data and its results.

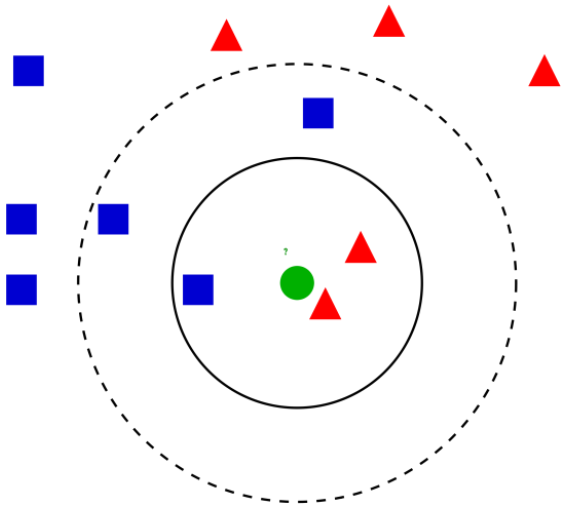
Disadvantages:

- Can only solve classification problems.
- Has relative low accuracy and performs well only for some specific problems.
- Assumes that all the features are independent from on another [12][27][42].

2.1.4 K-Nearest Neighbours

The *K-Nearest Neighbours* (k-NN) algorithm [23] is capable of solving both classification and regression problems. The training process of the algorithm is very simple and does not require any processing time. The training data is just stored with its actual output and any calculations will be made at the prediction stage. With each input of a vector of data the prediction of the class or the real value is dependent on the *k* closest data points to it (Graph 2.2). For classification the result is the majority of the

data samples' classification. While for regression the result is the average value of the closest data samples.



Graph 2.2. K-Nearest Neighbours [1]

The selection of *k* is very important, especially for classification problems, because for a large enough value it can reduce the interference or errors but a too large value will result into the borders of each class becoming less distinct and so will increase the error. In the case with only two classes, a good method for selection of *k* is by testing odd numbers so that equal votes are avoided and have a distinct winner. One way of finding the optimal *k* can be done by using the bootstrap method [23].

Finding the *k* nearest data points is achieved by finding the smallest distances from the input vector *x* in relation to a test vector *p* which is part of the training set. The most frequent way to calculate the distance is by using the Euclidian distance [23]:

$$D(x,p) = \sqrt{\sum_{k=1}^{n}(x_k - p_k)^2},$$

where $x = (x_1, x_2, \dots, x_n)$ and $p = (p_1, p_2, \dots, p_n)$.

When the algorithm selects the *k* nearest vectors it treats all the vectors equally. A stored vector that is closer to the input vector will have a higher probability to have the same class rather than with a more remote *k* vector. A solution to take into account

the distance is by adding weights to the vectors with each *k* closest neighbour a weight of $\frac{1}{d}$ is set, where *d* is the distance of the test vector with the input vector, while all others vectors that are not within the selected *k* will have a weight of 0 [18].

Advantages:

- Solves both regression and classification problems.
- Does not require any training time.
- The method that is used is non-parametric.
- More suitable for multimodal classes where information is extracted from multiple sources and where a vector can belong to multiple categories.
- Simple enough algorithm to be easily implemented and understood.

Disadvantages:

- There are cases where finding the optimal *k* is difficult or even impossible for some cases.
- Needs a large training sample to have good accuracy results.
- Having a larger data sample means slower prediction time to find the closest neighbours because it will have to find the distance for all samples stored.
- It has low tolerance to irrelevant features and data noise [12][27][56].

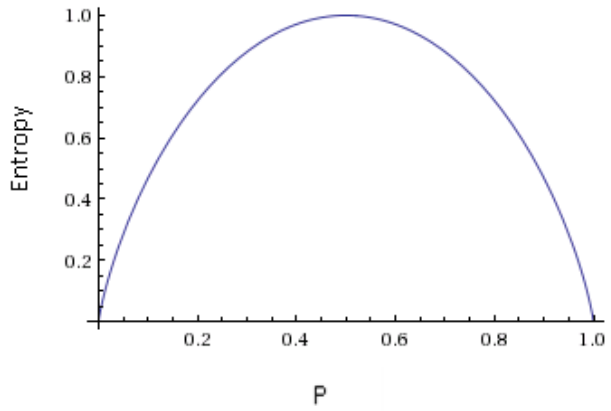
2.1.5 Decision Trees

A D*ecision Tree* [23] is constructed starting from its root leading to its leaves based on the passed training data sample and the constructor algorithm. At each node of the tree the value of a specific feature is examined and the result of this test is indicated on the edges of the node and also determine which will be the next test (node) it will examine. The final decision of the algorithm is found at the leaves of the tree which will be reached through the previous decision nodes [23].

The decision tree method can solve either classification, regression or both problems at the same time based on the used algorithm. Two of the algorithms used are the *C4.5* which can solve classification problems and the *CART* which can solve both regression and classification problems.

For classification (C4.5):

The *C4.5* algorithm [31][33] creates each node of the tree by using the metric obtained from the information gain, which determines which features will be examined. To obtain the information gain value it requires to define the entropy value. The entropy can have values between 0 and 1, with a value of 0 it defined that the data sample is homogeneous and with a value of 1 defines that the data sample is equally split (Graph 2.3).



Graph 2.3. Entropy curve in relation to P value

For a data sample *S* and *n* classes *C*, there is a value p_i, which expresses the percentage of the samples which belong to the *i*-th class. The entropy's value is obtained from the equation:

$$Entropy(S) = \sum_{i=1}^{n} -p_i \log_2 p_i.$$

To calculate the information gain it's necessary to calculate first the average weight of the entropy value for each feature, with *A* the feature that is evaluated:

$$I(S, A) = \sum_{i=1}^{n} \frac{|S_i|}{|S|} * Entropy(S_i).$$

With the final formula to calculate the gain for a single feature is [31][33]:

$$Gain(S, A) = Entropy(S) - I(S, A).$$

For classification and regression (CART):

The *Classification And Regression Tree* (CART) algorithm [31][33] is capable to handle both regression and classification problems building a binary tree. The different between the *C4.5* algorithm with this one is instead of using the entropy and information gain to define which feature is examined at each node, it uses the Gini impurity metric. The Gini impurity is a measure of incorrect classification when taking a random value/label from the available set. When minimising the metric the algorithm can find which feature optimal splits the data in the training data set:

$$I_G(t) = 1 - \sum_{i=1}^{C} p(i|t)^2,$$

where:

t: is the node which is examined

i: is the class which is being examined if it belongs to [31][33].

Advantages:

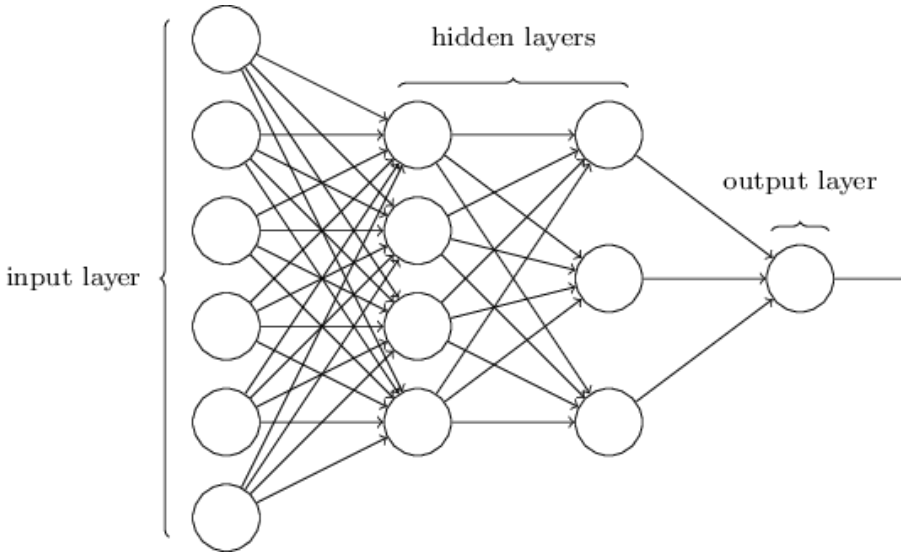
- Using the different available decision tree algorithms it can handle data input as values and/or classes and can solve classification and regression problems.
- The input data can be linear or non-linear.
- Fast output estimation.
- Handles cases where there are missing or irrelevant data.
- Easy to represent the created decision tree graphically and to understand the tree.

Disadvantages:

- Does not usually work well with small training data sample which can result to not covering all possible cases when building the tree.
- It presents a higher risk to overtraining.
- It is not tolerant to noise [12][27][31][56].

2.1.6 Neural Networks

A *Neural Network* [52] is capable of solving regression and/or classification problems depending on the choice of the model. The network consists of multiple nodes separated into three main categories. The first is the input layer which takes the input of the data sample, followed by one or more hidden layers which transforms and processes the data and finally the output layer which normalises the data and gives the predicted output as shown on Graph 2.4. Each layer of the neural network can have one or more nodes depending on the input vector size and the required output value. To train a neural network it requires to execute many cycles with each cycle consists of passing to the network each training data sample. To complete the training of a single sample it must be pushed through the network reaching the output layer and then from the output layer back to the input layer.



Graph 2.4. Multilayer Neural Network [37]

For single data sample, the data is initially propagated from the input layer all the way to the output layer, with this process known as feedforward. For each edge of the network, except from the initial edges data which pass the data to the network and the output edges, there is a weight which can be either negative, positive or zero. Larger the weight value is at an edge then the input value of that node will have a larger impact, either positive or negative depending on the weights sign. The impact of the weight is the result of its multiplication with the input value on that edge. The weights

are usually initialised to a small random value between -1 and 1. Each node obtains all the output values of the nodes from the previous layer by multiply each with the corresponding weight value [52]:

$$\sum_{j=0}^{n} w_{ij} x_j = o_i \, , where$$

j: the neuron of the previous layer and

i: the neuron which we want to find its output value

w: the weight value

x: input value of the previous neuron

o: estimated output value.

For each data sample apart from calculating the estimated output it also has already defined its actual output which is used to train the network by changing its weights. Altering the weights is achieved through the process of back propagation, which starts from the output layer and goes until the input layer. By calculating the difference between the estimated and the actual output it's used as an error indicator in an attempt to reduce it by changing appropriately the weights at each edge [37][52].

Advantages:

- Handles both regression and classification problems.
- Is able to generalise effectively from previous training samples.
- Has high accuracy when setting the appropriate parameters of the network.
- Can solve linear and non-linear problems.
- It can estimate the output quickly especially as it can be run in parallel.

Disadvantages:

- There is no way to access the internal network process.
- Training of the network can be slow requiring a lot of training cycles and in some cases very hard or impossible.

- Requires a large training sample so it can be effectively trained.
- It's not tolerant to large noise, missing values and irrelevant data features [6][27][37].

2.1.7 Support Vector Machine (SVM)

The *Support Vector Machine* (SVM) algorithm [23] can analyse and predict data for regression and classification problems. The training data passed to the algorithm are recorded on a p-dimensional graph, where *p* is the total number of features of the data. The target of the algorithm is to divide the data to their appropriate classes, using a linear or non-linear separation line. For regression the linear or non-linear line created tries to minimise the cost function between actual output and estimated value [23].

For linear SVM:

It is necessary to find an optimal *p*-1 dimensional hyperplane so as to separate the training data into the corresponding categories of classes. It will be separated in such a way as to maximize the margin between the data of each class. It's important that the hyperplane achieves the maximum possible data separation between two classes so that the system can be easily generalised when classifying or predicting correctly new data [23].

The equation of the hyperplace is the following:

$$w^T x + w_0 = 0 ,$$

where w_0 is the value that determines the point of intersection with the vertical axis.

If the problem is linearly separable then we can safely assume the following:

$$w^T x_i + w_0 \geq 0 \ \gamma\iota\alpha \ \forall \ i, so \ that \ y_i = 1$$

$$w^T x_i + w_0 \leq 0 \ \gamma\iota\alpha \ \forall \ i, so \ that \ y_i = -1$$

where there exists two classes {1, -1} and combining both of them we get:

$$y_i(w^T x_i + w_0) \geq 1 \ for \ \forall \ i$$

The optimal hyperplane is the hyperplane that maximises the total distance between all points of a class to another. In other words, the distance between the all points of a class and the adjoining points on the two support vectors, where the width of the margin is determined by $\frac{2}{\|w\|}$.

So we get the optimisation problem where we want to get the minimum $\|w\|$ in relation to the equation $y_i(w^T x_i + w_0) \geq 1 \; for \; \forall \; i$.

For non-linear SVM:

For cases when the data can't be linearly separated then the data must be mapped to a higher multidimensional space. This now allows the problem to be solved normally as a linear problem using a kernel.

$\Phi: X \rightarrow H$, where Φ is a kernel mapping function of points X to a higher multidimensional space *H*, so that each element *x* is replaced with *Φ(x)* [11][23].

Advantages:

- Can solve regression and classification problems and the data can be linearly or non-linearly separable.
- Estimation is accurate for a lot of problems and requires very little time.
- By finding the optimal margin it allows the algorithm to be generalised for new data samples.
- It can tolerate noise and outliers.

Disadvantages:

- Long training time especially with a large data sample.
- Difficult selecting the appropriate kernel for non-linear cases.
- For classification of more than two classes it's required to create a binary connection between all classes.
- It is prone to overtraining [12][27][56].

2.1.8 Ensemble Learning

In *Ensemble Learning methods* [20] the target is to reduce the classification or regression error, by executing multiple times different or the same, already existing, algorithms and combining the final results. With having the algorithms train on different data set and/or using different algorithms that complement each other, they reduce the probability of making the same errors on a specific input. Thus they can easily be generalised for any data set obtaining better results than using a single algorithm [20].

2.1.8.1 Bagging (Bootstrap Aggregating)

The *Bagging method* [20] can be used both for classification and regression problems, because it depends on the base learning algorithm which defines the types of problems able to be solved. The bagging process consists of running multiple times the defined algorithm taking a random reusable subsample from the training set. For classification algorithms the decision is obtained through a popularity vote between all classifiers created. In the case for regression the decision is obtained by averaging all the results [8]. The most usual algorithms that are used are typically those that are defined as weak learners, which have a slight better prediction than randomised prediction. This allows avoiding problems of overfitting and the combination of the multiple weak learners will allow the creation of a strong learner. On the other hand, using strong classifiers as the base algorithm will result in possible worse performance [20].

Advantages:

- Simple to implement if the base algorithm is already available. The only thing required is a loop which selects a random sample from the training set and passes it to the learner and an aggregator that combines the results.
- Gives better results than using weak learners on their own.
- There will be slower overfitting after many iterations.

Disadvantages:

- Using strong classification or regression algorithms for its base can result in poorer results than using them without bagging method.
- Difficult to visualise and interpret that data [8].

2.1.8.2 Boosting (AdaBoosting)

Boosting and more specifically an implementation of it is *Adaptive Boosting* (AdaBoosting) [43], which uses the same general idea as bagging. That is it uses repeatedly a weak base learner algorithm having as input the training data. It will then decide either the average value or the majority vote for regression or classification algorithms respectively. The main difference of boosting from bagging is that a weight value is maintained for all input vectors which determines the focus of the algorithm for following iterations.

After each iteration the weights of the input data set is changes in such a way that for each sample if it was wrongly classified or estimated then its weight will be increased while if it was correctly classified or estimated then it will be decreased [43].

For a binary classification problem:

x_i: input vector

y_i: output vector (binary classification)

g: weak classifier

m: current iteration set initially to 1 and increased by 1 by each iteration

The weights are initialised to$w_i^0 = \frac{1}{N}$, where *N* is the total input samples.

For each classifier g with weighted input$w_i * x_i$so that$g^m(x_i)$will result to values of {0, 1}

The error is calculated by:

$$E^m = \sum_{i=1}^{N} w_i^{(m-1)} I\left(y_i \neq (\hat{g})^m(x_i)\right) \Big/ \sum_{i=1}^{N} w_i^{(m-1)}.$$

From the error the weight is calculated for the classifier g^m:

$$a^m = log\left(\frac{1 - E^m}{E^m}\right).$$

Then the weights are updated as such:

$$\tilde{w}_i = w_i^{(m-1)} exp\left(a^m I\left(y_i \neq (\hat{g})^m(x_i)\right)\right),$$

$$w_i^m = \tilde{w}_i \Big/ \sum_{j=1}^{N} \tilde{w}_j.$$

The weights are fitted to the classifiers and updated until the error rate is reduced to an acceptable rate or until the max iteration which was set has been reached. Then a strong classifier *C* is constructed [27]:

$$C = \underset{y \in \{0,1\}}{argmin} \sum_{m=1}^{m_{stop}} a^m I((\hat{g})^m(x) = y).$$

Advantages:

- Shows good performance when using weak learners as the base classifier or predictor.
- It has a slower overfitting after many iterations.

Disadvantages:

- Weak learners must be executed sequentially because of the dependency on the weighted input set, unlike bagging which can be executed parallelly.
- Difficult to visualise and interpret the data [10][43].

2.1.8.3 Gradient Boosting

Gradient boosting [19] is a method that generalises from the boosting techniques like *AdaBoosting.* It will sequentially train many weak learners, usually a decision tree, to construct a strong learner. This is achieved by using a gradient decent method to minimise the error from the actual to the estimated output for each model built. The output function for each model is added and the final additive function will be used to give the estimated results for the set [19].

The general methodology is:

Having a set of N inputs with *x* the features and *y* the output and having selected a base learner *h* and an error function Ψ.

The negative gradient is calculated for the input:$\theta_t = g_t(x)$.

A new weak learner is trained: $h(x, \theta_t)$.

Then the smallest gradient descent step size is found:

$$p_t = argmin_p \sum_{i=1}^{N} \Psi \left[y_i, \hat{(f)}_{(t-1)}(x_i) \ + \ ph(x_i, \theta_t) \right].$$

And finally, the additive function is updated:

$$\hat{f_t} = \hat{(f)}_{(t-1)} \ + \ p_t h(x, \theta_t).$$

This will be executed until a max number of iterations *t* or until the error has been sufficiently reduced [36].

Advantages:

- Can use any weak leaner for the base of the model.
- The flexibility in choosing the error function allows the model to be properly fitted for many data sets and for classification or regression problems. There are many error functions readily available that can be used, but also one can be implemented specifically for the problem.

- The generalisation from the error function and including the boosting technique provides better accuracy than using a single execution of a weak leaner and in some cases than strong learners.
- Resistance to overfitting phenomenon.

Disadvantages:

- Requires longer time to obtain predictions because of the sequential nature of the methodology. This also creates limitations for a parallel version.
- Requires a lot of storage space, which depends on the number of iterations, for storing the parameters of each base learner.
- Apart from selecting the right base learner, it requires also to make an optimal decision on which error function the method will use [36].

2.1.8.4 Random Forests

The *Random Forest algorithm* [30] is a special case of boosting method which uses specifically a classification or regression tree as its base algorithm. When constructing the tree it also adds an additional randomness when deciding to split the node. Instead of splitting a node based on the best feature from all the available predictors, it will obtain a random sample and choose the best predictor from those selected. Adding the randomness and constructing many trees, the final estimator can lead to very good results by obtaining the majority vote for classification or the average value for regression trees [30].

Building the classification or regression tree is different from that from the decision tree algorithm. It's constructed in such a way that the results from the sub-trees have the smallest possible association between them. Thus the main differences from a decision tree are:

- For each tree, only one sub-sample of the available data is selected to be used for its training.
- For each node the choice of the decision feature is determined by the choice of *m* random features from *p*, where *p* is the number of input variables ($m \leq p$) (a

indicative value for classification: $m = \sqrt{p}$ and for regression: $m = \frac{p}{3}$). The value m is a constant for the execution of the entire algorithm. From the available features the best class is selected and its split into two nodes.

- The tree grows to its largest possible depth [9][18].

The same methodology to construct a classification or regression tree can be used for the previous ensemble algorithms too.

Advantages:

- Can predict classification and regression problems with the same accuracy or better of that of other boosting methods. Outperforms a lot of the non-ensemble algorithms for most problems.
- Tolerable to outlier values and noise.
- Can handle missing values in the feature set.
- Obtains results faster than the other base algorithms used for bagging and boosting methods.
- With the randomness of the algorithm it avoids problems of overfitting.
- The algorithm requires less parameter tuning.
- Can be parallelised to construct simultaneously many trees as for they are not dependent on each other.

Disadvantages:

- A lot of random trees must be generated to obtain good results.
- When having a lot of features and iterations it will require significant amount of memory.
- When estimating the error on the training data a sufficient number of random trees must be created or the error value will be inaccurate, tending to result in a higher error value [7][30].

2.1.8.5 Majority Voting Classifier

With *Majority Voting Learner* [24], the basic principal is to train multiple classifiers and combine their results by a majority vote, which will be the final decision. This method is based on that the classifiers will each have a small error rate and so the majority of them will conclude to the right classification. This allows the algorithms to complement each other and produce higher accuracy results. This can be extended to a weighted majority vote so that the estimated more reliable classifier will influence appropriately the final class [24].

Advantages:

- Has high accuracy basing on the fact that the majority of the classifiers will be able to classify correctly the output class.
- Any number of classifiers can be used for the vote.

Disadvantages:

- Highly depended on classifiers used for the specific problem and if proper parameters have been selected.
- This method can only be used for classification problems [24].

2.2 Unsupervised Learning

Unsupervised learning [18] is the second category of machine learning. The objective of these algorithms that are part of the unsupervised learning is to cluster together the input data into appropriate categories based on their features, hence on how similar they are between each other. In contrast to supervised learning algorithms, the input data for these algorithms do not require to have an expected class or value as for they do not require any prior training. However, in cases where there are no expected outputs there is no clear way to evaluate the correctness and effectiveness of the results of the clustering from an algorithm. Apart from clustering together the data, unsupervised learning algorithms can be used for data analyses and find hidden relations between the data. In addition, correlations and similarities can be found

between the features of a data sample which can lead to reducing the dimensionality of the vector without losing information [18].

2.2.1 Clustering Analysis

The goal of *Clustering algorithms* [49] is to cluster together the data in subgroups based on their features, so that the data of one group is quite similar between them and quite dissimilar to data in other groups. With the execution of a clustering algorithm the results of input data will be returned with a label which indicates in which group they are each part of.

There are three different types of clustering models:

- Centroid-based clustering
- Hierarchical clustering,
- Density-based clustering

2.2.1.1 Centroid-based Clustering (k-means)

The *Centroid Based Clustering* method [49] clusters the data by selecting initially *k* data vectors, which are found as far as possible from each other, that they will act as the central points of the *k* clusters. Sequentially the data that isn't clustered yet will be paired to the nearest cluster. A cluster is consists of the initial centroid and all additional vector points added. One of the main centroid based clustering algorithms is the *K-Means* algorithm [49].

For the k-means algorithm it's necessary to define from the beginning the total number of clusters *k* it will finally create, with each data vector only located in one of the available clusters. The centroid of each cluster is selected randomly from all the available input data vectors [49]. The objective of the algorithm is to group the data to the available clusters in such a way that the total difference of the points in the cluster and the total weight of the clusters is the smallest possible:

$$\underset{C_1, C_2, \ldots, C_k}{minimise} \left\{ \sum_{k=1}^{K} W(C_k) \right\},$$

where C_k is the k-th cluster and W the total weight of the cluster.

Each data vector is linked to the nearest cluster:

$$m_k = \frac{1}{|C_k|} \sum_{i \epsilon C_k} x_i,$$

where m_k is the centre of the *k* cluster.

The central points of the clusters are recalculated, where C(i) is the cluster that contains the *i*-th value:

$C(i) = \arg\min \|x_i - m_k\|^2$.

To find the internal weight of the cluster, which is calculated from the total difference of the data, it's usually used the Euclidian's squared distance and we get the following equation [23][25]:

$$W(C_k) = \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(j)=k} \left\| x_i - x_j \right\|^2 = \sum_{k=1}^{K} |C_k| \sum_{C(i)} \|x_i - m_k\|^2$$

Advantages:

- It's computationally fast for a small *k* as for the distance of each data point is calculated to each cluster, when the algorithm is limited to only finding the local optimal.
- It's more suitable for data that are easily separable and they create a spherical shape in space.
- Easy to represent the final clusters and in which cluster each data point belongs to.

Disadvantages:

- Difficult to define from the beginning the total number of clusters

- The initial random selection of the clusters centres drastically influences in which clusters the data will finally belong to
- Usually stops at a local optimal point. To find the global optimal increases the complexity of the algorithm
- It's sensitive to outliers and noise [26][35][45][49]

2.2.1.2 Hierarchical Clustering

The *Hierarchical Clustering* [18][49] has as an objective to build a dendrogram of clusters based on the two methods of agglomerative or divisive. For the agglomerative method each data point is initially considered to be also a cluster of its own. The method will successively merge the two closest clusters together until only one cluster is available, forming a tree hierarchy of the clusters for each step. On the other hand, the divisive method has all the data points initially merged to a single cluster. The divisive method splits each cluster that will be the furthest between them until eventually there are as much clusters as the total number of data points. In both cases they can be stopped earlier reaching to a desirable number of clusters [18][49].

To determine which clusters have to be merged or split usually the distance between the data points and a composition criteria that determines the disparities between the clusters is used. The most common way to calculate the distance between arithmetic points is the Euclidian distance:

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2},$$

While for non arithmetic values the Hamming or Levenshtein distance is used, which measure the differences between two strings.

There are different composition criteria which some of them are:

Complete linkage clustering: $\max \{ d(x,y): x \in A, y \in B \}$,

Single linkage clustering: $\min \{ d(x,y): x \in A, y \in B \}$,

Average linkage clustering: $\frac{1}{|A||B|} \sum_{x \epsilon A} \sum_{y \epsilon B} d(x, y)$,

where A and B are two clusters [49][58].

Advantages:

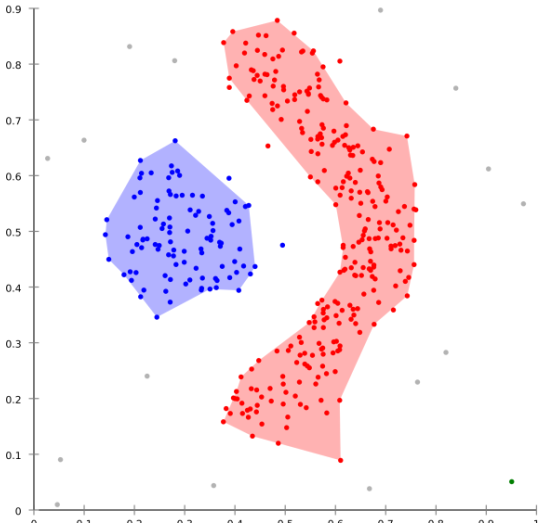
- There is no need to determine the number of clusters from the beginnings, but it can be determined at which depth of the tree hierarchy the algorithm will stop.
- It's more suitable for problems associated with connection points.
- Has good clustering results for a small data sample.
- Multiple executions of the algorithm result in the same clustering of the data points.
- Easy to represent the hierarchical presentation of the clusters.

Disadvantages:

- It's not possible for the algorithm to do corrections once after a decision has been made regarding a merge or split of a cluster.
- For a large number of data or for data that have a lot of features it requires a lot of processing time.
- Selecting a composition criteria is important because it can create problems for when dealing with noise, splitting large clusters and/or handling clusters of a variety of sizes.
- It does not have a general purpose function to optimise the clustering and reduce the overall execution time. Merging or splitting decision of a cluster are taken locally at every step [26][35][45].

2.2.1.3 Density-based Clustering (DBSCAN)

With the *Density Based Spatial Clustering of Applications with Noise* (DBSCAN) algorithm [49] the clusters are created based on the density of the data when placed in space as shown in Graph 2.5. The data that are found to be more dense create the kernel of the cluster, while that data that is more sparingly create the boundaries of the cluster. The data that are isolated will not be clustered as for they will be considered as noise and outliers [49].



Graph 2.5. Density Based Clustering [14]

The algorithm requires two parameters, the *Eps* (the radius of the cluster) and *MinPts* (the minimum number of data required to create a cluster). A data point to be a part of a cluster it must have in a radius *Eps* at least another data point:

$$N_{Eps}(p) = \{q \epsilon D | dist(p,q) < Eps\},$$

where N_{Eps} is the Eps-neighbourhood of the point.

One point can be either a border point:

$$p \epsilon N_{Eps}(q)$$

or kernel point:

$$\left|N_{Eps}(q) \geq MinPts\right|.$$

Additionally two clusters can be merged together when the following cases are true:

1. $\forall\, p, q: if\ p \epsilon C\ and\ q\ is\ density\ reachable\ from\ p\ in\ relation\ to\ Eps\ and\ MinPts\ then\ q \epsilon C$
2. $\forall\, p, q \epsilon C: p\ is\ density\ connected\ to\ q\ in\ relation\ to\ Eps\ and\ MinPts$

where,

density reachable: one point p is density reachable from point q, if there is a chain of direct link points linking the two points, and

density connected: two points p and q are density connected, if it's density reachable from p to q and vice versa. [15]

Advantages:

- There is no need to determine the number of clusters.
- It's tolerant to noise and irregular data.
- Capable of locating clusters of arbitrary size and shape.
- It's fast for data that have a few features.

Disadvantages:

- It's not optimal for cluster separation when the points between them have different densities or when there are many features.
- Can be hard to define the values of the parameters *Eps* and *MinPts*, which have a large impact on the final clustering of the data [5][45][49].

2.2.2 Dimensionality Reduction (PCA)

The *Principal Component Analysis* (PCA) algorithm [59][60] aims to reduce the total number of features of the data set to new linear non-correlated features. The data is represented in a matrix of size *N*X*, where *N* is the total number of data samples and *X* the initial number of features of each sample, is transformed to a matrix of size *N*Y*, where *Y* is the final number of features for each sample. The reduction of the number of features is done in such a way that the features created will be able to express the same information without any loss.

The reduction is achieved through the *p* main components where *p* is the number of features in the data set. The first component is the linear combination of the initial values of the features so that they have the largest possible difference between them. For each subsequent component until all *p* are covered take the next highest difference of the linear combination of the features. All the components are independent from one another [59][60].

Finding the value of each component is achieved through the calculation of the eigenvectors and eigenvalues of the covariance matrix:

$$cov(X,Y) = \sum_{i=1}^{N} \frac{(x_i - \bar{x})(y_i - \bar{y})}{N},$$

where

N: the total number of data samples,

$\bar{x}$: the mean value of feature X,

$\bar{y}$: the mean value of feature Y,

The mean value of each feature is subtracted from each respectively feature from the data sample so that the data is normalised towards the centre. The sign of the result from the above equation also defines the relation between the two features. If it's positive then the two dimensions are increased, otherwise if it's negative then they decrease. However, if the result is 0 then the two features are independent from one another.

From the covariance matrix the values of eigenvectors and eigenvalues can be calculated. Then we sort the eigenvectors based on their eigenvalues in descending order. This gives the most important components in the same order sorted with the highest value indicating the highest importance. This allows us to ignore the less important ones in an attempt to reduce the dimension of the final table [46][60].

Advantages:

- Reduces the complexity and the total number of features of the data set.
- Reduces the noise in the data as for the new features which are selected have the largest difference and so those that have the smallest will not be included.
- Reduces the total storage size of the data set.

Disadvantages:

- It's difficult to evaluate and validate the accuracy of the covariance matrix.

- It's based on linear assumptions.
- It's based on rectangular transformations [25][46].

2.2.3 Self-Organising Maps

The *Self Organising Maps* (SOM) [18] is a type of neural network which in contrast with the most type of networks it is placed in the category of unsupervised learning and it's based on competitive learning. So the training of the network is done based on the features of the data set which do not contain a label or actual output value and do not need any interfering with.

The network creates a topology which maintains a match from a multidimensional space depending on the number of features of the data set to a network of neurons. In other words the network from a two dimensional neural network mesh is mapped to a single level. The SOM maintaining the topology keeps the relative distance between the points. So the points that have similar patterns are found in neighbouring neurons in the network [18].

For training and tuning, similarly to neural networks used for supervised learning, the SOM network initialises the weights of the input edges to small random weights. Sequentially, a data sample is selected to training the network until all the available data is exhausted. For each neuron the distance is calculated to the input data so the neuron where its weight is loser to the input vector can be found, which is called *Best Matching Unit* (BMU). The BMU us calculated from the square of the Euclidean distance:

$$dist = \sum_{i=0}^{n} (I_i - W_i)^2,$$

where

I: the input vector,

W: the neuron's weight,

n: the total number of neurons in the network.

The radius of the BMU neighbourhood is then calculated:

$$\sigma(t) = \sigma_0 e^{(-\frac{t}{\lambda})},$$

where

t: the current iteration,

σ_0: the width of the map grid for the time point 0,

λ: time constant which is calculated from: $\lambda = \frac{total\ number\ of\ iterations}{map\ radius}$.

All the neurons that are found in the radius of the BMU previously then they will need to change their weights so that there are closer to the input vector. A neuron that is found closer to the BMU neuron then it will have a larger weight change so it will become relative close to the input vector.

The new weight is calculated from [22][61]:

$$W(t+1) = W(t) + \Theta(t)L(t)\big(I(t) - W(t)\big),$$

where Θ is the distance from BMU: $\Theta(t) = e^{-\frac{BMUdist}{2\sigma^2(t)}}$

and L the learning rate: $L(t) = L_0 e^{-\frac{t}{\lambda}}$.

Advantages:

- Makes it easy to observe and verify data clustering.
- Easy to represent the output results and the state of the network.
- Maintains wither strong or weak correlation between the clusters.
- Capable to organise a lot and complicated data.
- It's tolerable to large noise.

Disadvantages:

- The number of clusters must be selected appropriately based on the problem.

- Needs sufficient and a variety of data samples so that it can create accurate and representative.
- It is required that the nearby data points in the space must have similar behaviour.
- Matching can lead to divided clusters.
- It's computational expensive to calculate on a large volume of data and especially if they have a lot of features [21][22][57].

2.3 Category Selection

The two of the categories for machine learning are supervised and unsupervised learning [18]. A supervised learning algorithm model is required first to be trained with a sample data that also contains the actual output class or value. After the model is trained the model can make predictions over new data samples to classify in which class they belong to or predict their values. An unsupervised learning model does not require any prior training nor the input data to have an actual output value. The model finds the most similar input data and clusters them together.

This study will use three different data samples obtained from the Department of Psychology to explore how machine learning can aid in indentifying health conditions. So the algorithms that will be used will be focused on trying to accurately predict, given a data set containing biological measurements, if a person is at one of the two categories of high or low risk for a specific health condition. Hence the analyses will be focused on supervised learning models and more specifically on binary classification algorithms, as for there are only two possible classes.

Chapter 3

Metric Analysis of Supervised Learning Algorithms

Given a data set, containing a vector set of features and the actual value or classification for each sample, for the purposes of training a supervised learning algorithm it is required to be split. The data set has to be separated into the training and testing set, with only the former will be used for training the algorithm and the latter will be used to evaluate its performance. The testing set is required to be independent so the metric analysis of the algorithm can be more representative of any new unknown data sample passed to be predicted. There are different metrics depending if the problem is a regression or classification problem and the given evaluation importance of true positives/negatives or false positives/negatives [55].

3.1 Error Rate

The *error rate* is used to calculate the difference between the actual and predicted values for each test sample passed. The total error rate for a specific algorithm is the mean value of each sample's error rate. It's dominantly used for calculating the performance of predicted numerical values for regression problems on the testing set. Also, the calculation of the error rate is used by some supervised learning algorithms, either classification or regression, for the runtime evaluation of the algorithm's training progress. This is done with the purpose to properly tune the internal parameters of the algorithm so it can better reflect the training set, minimising the predicted from the actual values [55].

Different performance evaluators can be used to estimate the error rate of the test set (with *a* the actual value, *p* the predicted value and *n* total samples) [55]:

Mean Squared Error: $\frac{(p_1-a_1)^2 + \ldots + (p_n-a_n)^2}{n}$

Root Mean Squared Error: $\sqrt{\frac{(p_1-a_1)^2 + \ldots + (p_n-a_n)^2}{n}}$

Mean Absolute Error: $\frac{|p_1-a_1| + \ldots + |p_n-a_n|}{n}$

Relative Squared Error: $\frac{(p_1-a_1)^2 + \ldots + (p_n-a_n)^2}{(a_1-\bar{a})^2 + \ldots + (a_n-\bar{a})^2}$

Root Relative Squared Error: $\sqrt{\frac{(p_1-a_1)^2 + \ldots + (p_n-a_n)^2}{(a_1-\bar{a})^2 + \ldots + (a_n-\bar{a})^2}}$

Relative Absolute Error: $\frac{|p_1-a_1| + \ldots + |p_n-a_n|}{|a_1-\bar{a}| + \ldots + |a_n-\bar{a}|}$

Correlation Coefficient: $\frac{\frac{\sum_{i=1}^{n}(p_i-\bar{p})(a_i-\bar{a})}{n-1}}{\sqrt{\frac{\sum_{i=1}^{n}(p_i-\bar{p})^2}{n-1} * \frac{\sum_{i=1}^{n}(a_i-\bar{a})^2}{n-1}}}$

3.2 Accuracy

Measuring *accuracy* is one of the most common metric analysis for classification algorithms, while it can't be used for regression analysis [47]. The accuracy expresses the ability of the algorithm to classify correctly the data samples during testing and therefore to also successfully classify future data, expressed as a percent. This percent is calculated from the total positive classifications with the respect to the total number of data samples [55].

This metric can't be used as a sole performance rating for a classification algorithm since a high percent value doesn't also necessarily mean a good classification. As a weakness for the accuracy metric, it can't express whether the false positives or the false negatives are more. Also, for a more representative accuracy there has to be about the same number of samples in the test set for each class defined [29][55].

They following terms are required to be defined for which are used for binary classification [47]:

TP: True Positive Results - The results that are classified as positive and were expected to be classified as positive.

TN: True Negative Results - The results that are classified as negative and were expected to be classified as negative.

FP: False Positive Results - The results that are classified as positive and were expected to be classified as negative.

FN: False Negative Results - The results that are classified as negative and were expected to be classified as positive.

The accuracy is calculated by:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

3.3 Positive Predicted Value - Precision

The *Positive Predicted Value* (PPV) is the metric that evaluates the classifier at how capable it's to not classify a sample as positive while it's truly negative. The percentage is calculated from the total number of true positives divided by the total number of true and false positives [47]:

$$positive\ predicted\ value = \frac{TP}{TP + FP}$$

3.4 Negative Predicted Value

The N*egative Predicted Value* (NPV) is the metric that evaluates the classifier at how capable it's to not classify a sample as negative while it's truly positive. The percentage is calculated from the total number of true negatives divided by the total number of true and false negatives [38]:

$$negative\ predicted\ value = \frac{TN}{TN + FN}$$

3.5 Sensitivity

The *sensitivity* is the metric that evaluates the classifier at how capable it's to successfully detect a sample as positive if it's truly positive. The percentage is calculated from the total number of true positives divided by the total number of true positives and false negatives [47]:

$$sensitivity = \frac{TP}{TP + FN}$$

3.6 Specificity

The *specificity* is the metric that evaluates the classifier at how capable it's to successfully detect a sample as negative if it's truly negative. The percentage is calculated from the total number of true negatives divided by the total number of false positives and true negatives [47]:

$$specificity = \frac{TN}{FP + TN}$$

3.7 Confusion Matrix

The *confusion matrix* is a matrix of dimensions $n * n$, where *n* the number of classes defined, which allows for easier representation and extraction of information regarding the performance of a classification algorithm. On the matrix the rows represent the actual classifications of the data, while the columns represent the predicted classifications of the data (Table 3.1) [62].

	Predicted Positive Class	**Predicted Negative Class**
Actual Positive Class	Number of True Positives (TP)	Number of False Negatives (FN)
Actual Negative Class	Number of False Negatives (FP)	Number of True Negatives (TN)

Table 3.1. Confusion Matrix

3.8 Receiver Operating Characteristics - ROC

The *Receiver Operating Characteristics* (ROC) curve is a graphical representation in a two dimensional space which expresses the association between the rate TP on the y axis and the rate FP on the x axis, where [16]:

$$TP\ rate\ = \frac{TP}{TP + FN}, \text{and}$$

$$FP\ rate\ = \frac{FP}{FP + TN}.$$

The graphical representation is divided diagonally from the point (0,0) until (1,1) from the boundary limit (Figure 3.1). This boundary, for binary decisions, has usually value of 0.5 which determines if the decision of a sample belongs to the one or

the other class. When a classifier has at its output only the classification of the data, then pair of rate TP and FP is created which represents only a single point on the graph. In case there is a changing boundary limit then a graph plot/curve is created from all possible points.

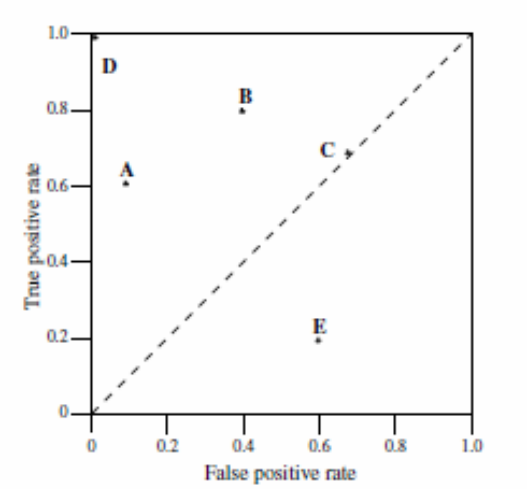


Figure 3.1. Graphical representation of ROC with 5 classifiers [16]

On the graph, the point (0,0) represents the case where the classifier never gives false positives nor true positives. While at the point (1,1) will always give true positives but will also give false positives. The ideal point is as close as possible to the point (0,1) which gives the highest percentage of true positive results while having the lowest percentage to give false positive results [16][55].

3.9 Area Under the Curve - AUC

From the ROC curve it is possible to calculate the *Area Under the Curve* (AUC) which determines the total probability that a classifier can classify correctly a random positive sample higher than a random negative sample. Because the AUC calculates the area under the ROC curve, which has a range from 0 to 1 included, it can have values also ranging from 0 to 1 included. However, the diagonal line created from points (0,0) to (1,1) covers a total of surface area of 0.5 which can be obtained from binary classifier that classifies randomly. So a binary classifier can't theoretically have worse performance than 0.5 surface area, with the optimal values being close to 1 [16].

3.10 Metric Selection

From the listed metrics that can be used for supervised learning algorithms not all of them can be used for classification problems. The error rate will not be used as for not all of the classification algorithms are able to take advantage of it and its more focused towards regression algorithms [55]. From the classification algorithms the only one that can take advantage of the error rate are the different variances of the *Neural Network* [52], which it can return as an intermediate value a real value before its rounded to the closest class representation. So the final metrics that will be used to evaluate the binary classification algorithms for the given datasets are PPV, NPV, sensitivity, specificity and AUC, which are all suitable for classification algorithms.

Chapter 4

Sensors

The sensors presented are the ones used in total for the three data sets, for experiential avoidance in smokers, eating disorders and experiential avoidance in people with anxiety using the sensors *electrocardiography*, *galvanic skin response* and *facial electromyography*. Also, the sensors that are as a requirement for the wearable devices are *photoplethysmography* or *electrocardiography*, *galvanic skin response* and *electroencephalography*.

Recording the physiological measurements using the appropriate sensors for each case it is possible to extract significant information on the biological function and behaviour throughout a specific duration of time. Under specific conditions the measurements obtained will indicate a differentiation of the values from what otherwise they would have been. Depending on which of the sensors recorded different than usual measurements we can expect a higher risk for a specific health condition. The data from each person collected from the sensors can be used as the set of input features for the classification algorithms to easily indentify health problems. This is possible as for there's a strong correlation between the signals recorded and some health conditions, as it will show in the three experiments.

4.1 Photoplethysmography (PPG)

The *Photoplethysmography* **(**PPG) [4] is a method that uses optical measurements so that it can record and calculate the volume variance of the blood in the microvascular bed of tissue through time. To obtain these measurements it is required to have one or more sources of light from LEDs and a very sensitive photodetector. A PPG sensor can be placed either on the finger, wrist or head to take measurements effectively. At these sites the photodetector can measure the change in the light reflected back from the skin omitted by the LEDs. The intensity of the light received back is dependent on the blood volume passed through the arteries. The recordings from the PPG can be used to monitor the values of the blood oxygen saturation, blood pressure, cardiac output, cardiac blood supply and respiration of a person [4].

The pulsatile waveform created from the measurements of the PPG consists of the alternating current (AC) and the direct current (DC). The AC component captures the changes in the volume of the blood in the arteries for each heart pulse. While the DC component can indicate the optical property and the small energy variations of the skin tissue [3][13]. The DC component can be derived from the respiration, sympathetic nervous system activity and thermoregulation. The higher frequency AC component is superimposed on the DC baseline, which is a lower frequency. There are also some factors the affect the light received from the photdetector which must be taken into account such as the blood volume, movement on the blood vessel wall and the orientation of red blood cells [4].

The AC waveform that is created is separated in two main phases, the anodic and the cathodic pulse phases. The rising part of the waveform until the peak, the anodic phase, indicates the systole of the heart. Followed by the declining part of the waveform, the cathodic phase, which indicates the diastole of the heart, completing one full cardiac cycle [4].

4.2 Electrocardiography (ECG)

The *Electrocardiography* (ECG) [34] is a method that records the electrical activity of the heart. This electrical signal is created from the sinoatrial node, the natural pacemaker of the heart, and is transmitted to the heart. This signal controls the systole

and diastole of the cardiac muscle from the depolarisation and repolarisation of the myocardial vessels. The electrical activity that is created is also reflected from the heart to the surrounding tissues and to the skin, which can then be detected through sensors [34]. From the measurements taken from the ECG there is the capability to monitor values regarding the heart pulse, blood pressure and the respiratory system [34][63][64].

The electrical activity that is recorded from the ECG sensor is illustrated as a waveform which shows the systole and diastole of the heart. When there is a rise on the waveform it shows the point when a electrical signal is created and sent to the heart to become polarised so it can be contracted. After the peak of the waveform it follows a declination, which indicates that no signal is transmitted to the heart. This allows the heart to become depolarised resulting to its expansion to its normal state [34].

4.3 Galvanic Skin Response (GSR)

The *Galvanic Skin Response* (GSR) [65] measures how good of an electrical conductor is the skin of a person for a specific time period. The GSR is taken by placing two electrodes close to the skin which let through a small voltage between them. The voltage through the skin is measured and is used to calculate the conductivity of the individual's skin compared to the voltage initially passed. The measurements are more effective when they are taken from places located close to eccrine sweat glands. When the sweat glands produce more humidity (sweat) they become a better electrical conductor.

The sweat generated from the glands are caused by the need for thermoregulation of the skin, but is also regulated from the sympathetic nervous system. The sweat glands are closely linked with the sympathetic nervous system and so are influenced by the psychological state of the individual. Any psychological arousals, either from external or internal stimulus, increase the conductivity of the skin as for it leads to an increase of sweat production. The measurements from the GSR indicate the arousal levels which in turn is linked to be one main components that are related to emotion. This has as a result of it in being able to predict and calculate information of the emotional state of a person [65].

4.4 Electroencephalography (EEG)

The *Electroencephalography* (EEG) [2] is a method that records the electrical signal that is generated from the electrical activity of the brain. The neural cells in the brain produce rhythmic electrical pulses which create specific patterns, depending on the state of the individual. The measurements of the electrical signal are recorded by using electrodes that are placed on the individual's head [2]. From the EEG measurements it can be used to detect abnormal brain activity caused by a seizure, stroke, brain tumour or head injury and sometimes from dizziness, headache and sleeping disorders [17].

The recording of the signal from the brain can be separated to five different bands that depended on the frequency rate. The signals are distinct to delta, theta, alpha, beta and gamma frequencies. The signal recorded is influenced by the age, behavioural conditions, neuropathological conditions and metabolic disorders, while also the usage of some medications [2].

4.5 Facial Electromyography (fEMG)

The *Facial Electromyography* **(**fEMG) [53] is a technique for identifying the facial muscle activity by measuring the electrical activity sent to the muscle. For each muscle location two electrodes are placed in close proximity to each other which detect the muscle contraction and relaxation by recording the electrical signal. The locations which the electrodes can be placed are on the frontalis, corrugator supercilli, orbicularis oculi, levator labii superioris alaeque nasi, zygomaticus major, orbicularis oris, depressor anguli oris and mentalis muscles and a single reference electrode is placed at the hair border line. Using measurements from a combination of the electrodes it is possible to detect the elementary emotions expressed such as happiness, surprise, fear, anger, sadness and disgust [53].

Chapter 5

Case Analysis

A total of three data sets, regarding experiential avoidance in smokers, eating disorders and experiential avoidance in people with anxiety, are used to evaluate and compare a set of supervised classification algorithms. The classification algorithms used for each set are *Logistic Regression, Naive Bayes, K-Nearest Neighbours, Classification Tree, Neural Network, Support Vector Machine, Bagging, AdaBoosting, Gradient Tree Boosting* and *Random Forest.* From the analysis of the algorithms on the three sets and taking into account their advantages and disadvantages to determine which one will be more suitable to use as part of the console implementation.

5.1 Diagnosis of Experiential Avoidance in Smokers

There is available a data sample from 32 people which contains measurements taken from the signals electrocardiogram (ECG), corrugator muscle (COR - using facial electromyography) and galvanic skin response (GSR). This data was obtained from the Department of Psychology at the University of Cyprus. The measurements were taken following a specific procedure which was the same for all of the participants. For each person the measurements were taken from five different continuously time frames. From these time frames, the measurements from the first time frame are not used, but it's there so the person can be in a state of calm. For the following two, a short film was

shown which should create neutral emotions to the viewer. Finally, for the last two, again, a short film was shown which this time should create negative emotions. The data obtained from the participants the Department of Psychology was able to classify them to two categories, the people which accept the experiential avoidance (represented as a value of 1) and the people which do not accept it (represented as a value of 0) [50].

Based on a previous research [50], to obtain the best results in classifying a sample in the dataset the only signal that will be used in the supervised learning algorithms is that from the GSR. Since the first time frame obtained in the recording it is used for the purpose to relax the person and so only the remaining four will be used for classification. The feature obtained from the stream of data from each time frame will be the mean value of that time frame.

From the data sample there are cases which it wasn't possible to determine the mean value of the GSR for one or more of the time frames, but not more than three. For these cases, the missing values were filled in with an estimated value using the mean of the other available time frames, excluding the first one. Most classification algorithms cannot handle missing values on their own and as such require some pre-processing so the data isn't discarded.

For the training and testing of the classification algorithms which will be used it's necessary to separate the data sample. It will be split into two class balanced subsamples which $\lceil 2/3 \rceil$ will be used for training and the rest will be used for testing the algorithms. That is, $\lceil 2/3 \rceil$ of the data which is actually classified as acceptance and $\lceil 2/3 \rceil$ of the data which is actually classified as avoidance are taken for the training and the rest are used for the testing subsample. So they are in total 22 data samples for training (10 acceptance and 12 avoidance) and 10 data samples for testing (5 acceptance and 5 avoidance).

In comparison to the previous analysis [50], a different distribution of the data set was used. In total 25 data samples were used for the training and 8 for the test set (having available an additional data sample). Also, the distribution of the data wasn't necessary done in a classed balanced way.

Each of the algorithms will be executed 10 times with a different distribution of the data between training and testing. But between the algorithms the 10 data distributions used will be the same. In the analysis of the algorithms the results will be used from the distribution that returns the best results.

Logistic Regression:

In Table 5.1 the outputs of the logistic regression algorithm is summarised.

	Predicted Positive Class	**Predicted Negative Class**
Actual Positive Class	4	1
Actual Negative Class	1	4

Table 5.1. Results of logistic regression in a confusion matrix

Accuracy:= 0.8 (Average value of accuracy for 10 executions:= 0. 54)

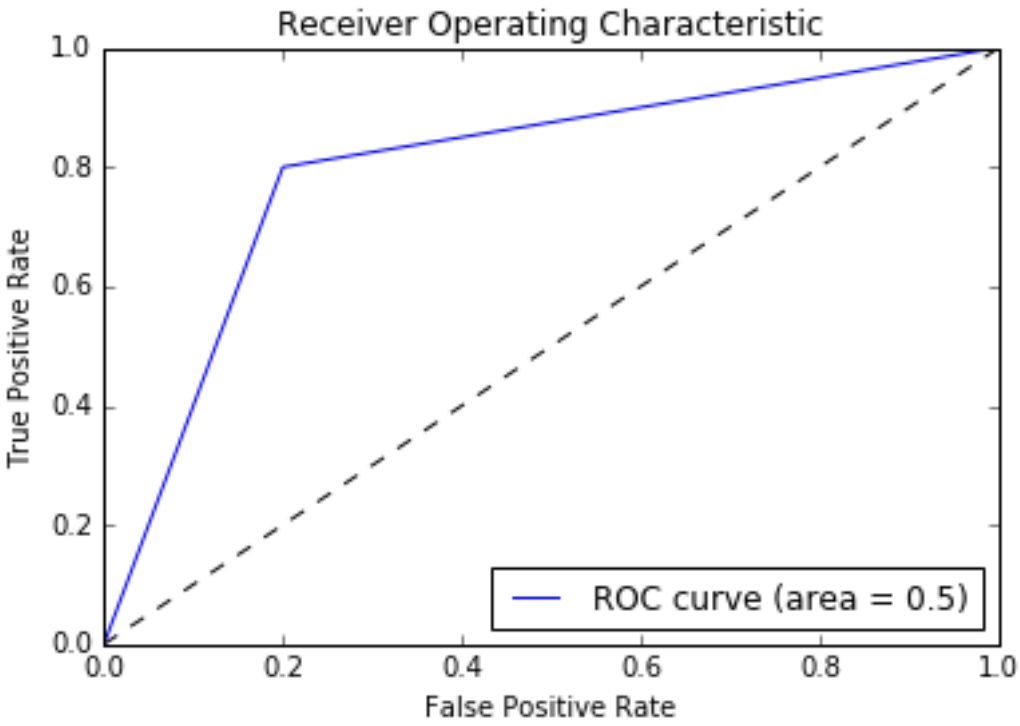
Positive Predicted Value:= 0.8

Negative Predicted Value:= 0.8

Sensitivity:= 0.8

Specificity:= 0.8

AUC: = 0.8 (from Graph 5.1)



Graph 5.1. Receiver operating characteristic curve for logistic regression algorithm

Naive Bayers:

In Table 5.2 the outputs of the Naive Bayes algorithm is summarised.

	Predicted Positive Class	**Predicted Negative Class**
Actual Positive Class	4	1
Actual Negative Class	1	4

Table 5.2. Results of Naive Bayes in a confusion matrix

Accuracy:= 0.8 (Average value of accuracy for 10 executions:= 0. 58)

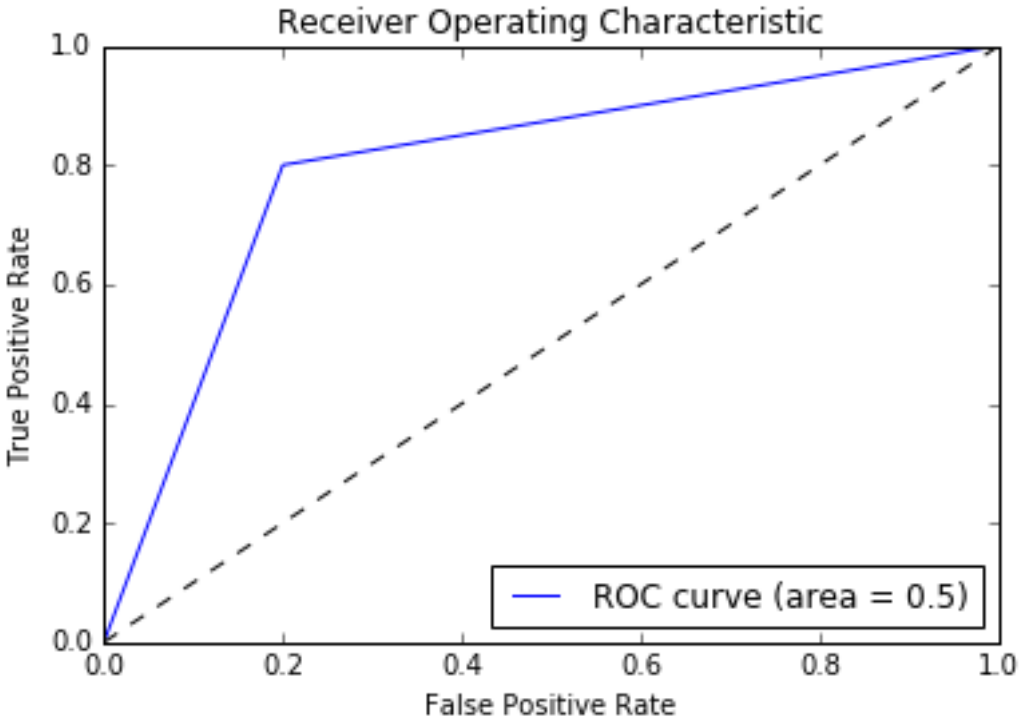
Positive Predicted Value:= 0.8

Negative Predicted Value:= 0.8

Sensitivity:= 0.8

Specificity:= 0.8

AUC: = 0.8 (from Graph 5.2)



Graph 5.2. Receiver operating characteristic curve for Naive Bayes algorithm

K-Nearest Neighbours (K = 3):

The number of nearest points to examine has been determined to be 3. The value has been obtained by executing on the dataset the algorithm with the number of clusters from 1 to 5. With using 3 nearest neighbours resulted to the highest accuracy and as such will be used to evaluate the algorithm.

In Table 5.3 the outputs of the k-nearest neighbours algorithm is summarised.

	Predicted Positive Class	**Predicted Negative Class**
Actual Positive Class	4	1
Actual Negative Class	1	4

Table 5.3. Results of k-nearest neighbours in a confusion matrix

Accuracy:= 0.8 (Average value of accuracy for 10 executions:= 0.59)

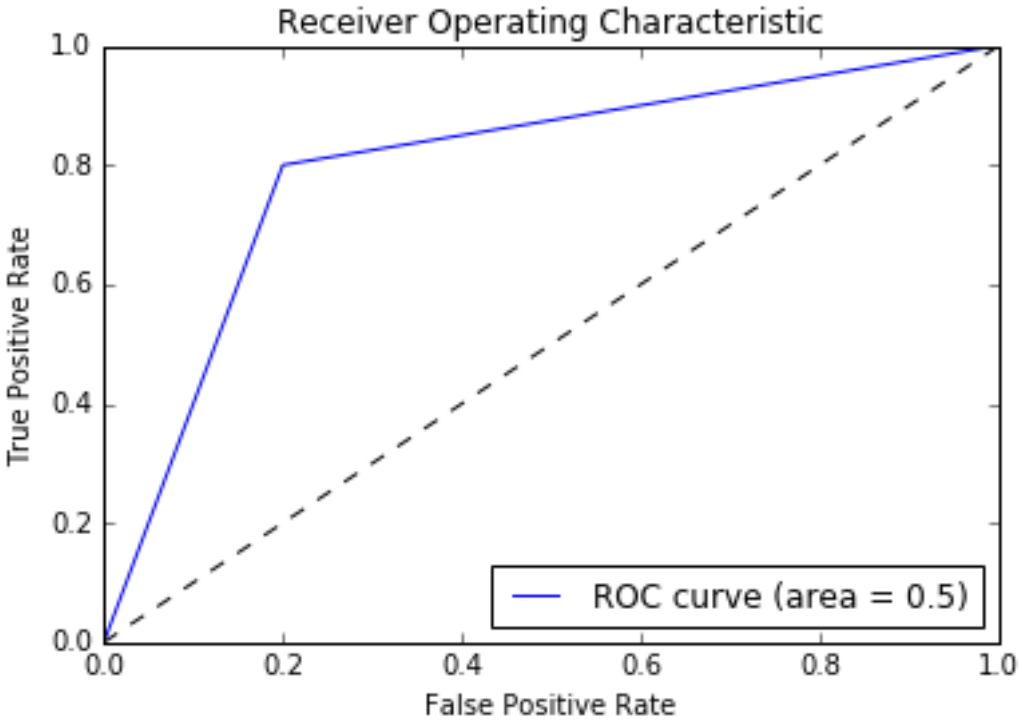
Positive Predicted Value:= 0.8

Negative Predicted Value:= 0.8

Sensitivity:= 0.8

Specificity:= 0.8

AUC: = 0.8 (from Graph 5.3)



Graph 5.3. Receiver operating characteristic curve for k-nearest algorithm

Classification Tree:

In Table 5.4 the outputs of the classification tree algorithm is summarised.

	Predicted Positive Class	Predicted Negative Class
Actual Positive Class	2	3
Actual Negative Class	1	4

Table 5.4. Results of classification tree in a confusion matrix

Accuracy:= 0.6 (Average value of accuracy for 10 executions:= 0.48)

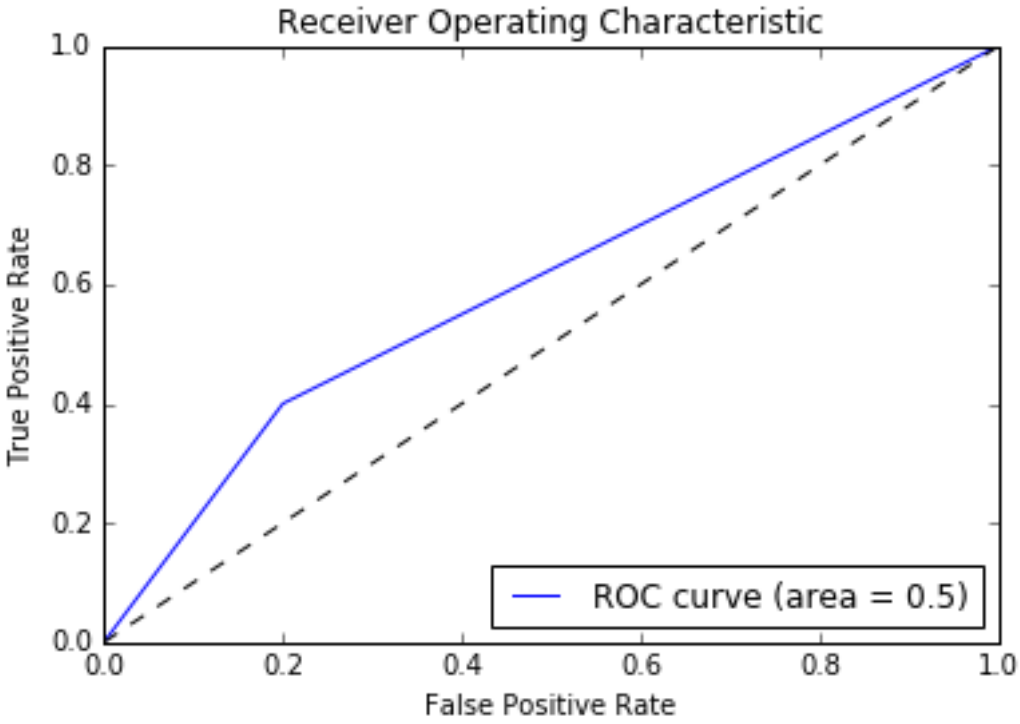
Positive Predicted Value:= 0.666666666667

Negative Predicted Value:= 0.571428571429

Sensitivity:= 0.4

Specificity:= 0.8

AUC: = 0.6 (from Graph 5.4)



Graph 5.4. Receiver operating characteristic curve for classification tree algorithm

Neural Network:

In Table 5.5 the outputs of the neural network algorithm is summarised.

	Predicted Positive Class	**Predicted Negative Class**
Actual Positive Class	3	2
Actual Negative Class	1	4

Table 5.5. Results of neural network in a confusion matrix

Accuracy:= 0.7 (Average value of accuracy for 10 executions:= 0.54)

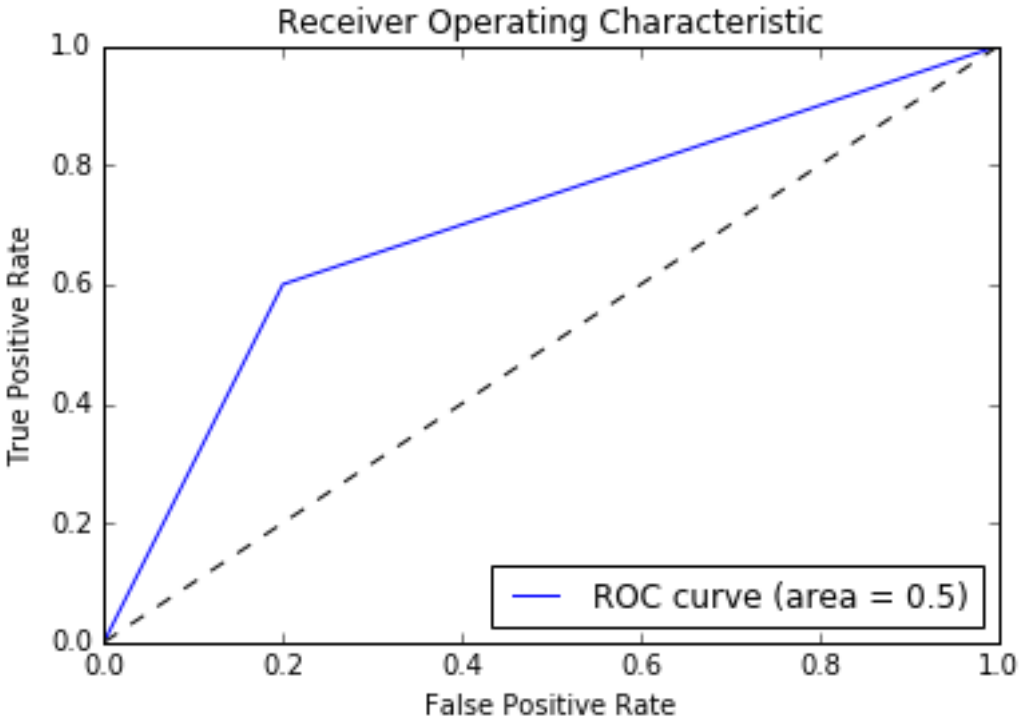
Positive Predicted Value:= 0.75

Negative Predicted Value:= 0.666666666667

Sensitivity:= 0.6

Specificity:= 0.8

AUC: = 0.7 (from Graph 5.5)



Graph 5.5. Receiver operating characteristic curve for neural network algorithm

SVM:

In Table 5.6 the outputs of the SVM algorithm is summarised.

	Predicted Positive Class	**Predicted Negative Class**
Actual Positive Class	4	1
Actual Negative Class	1	4

Table 5.6. Results SVM in a confusion matrix

Accuracy:= 0.8 (Average value of accuracy for 10 executions:= 0.55)

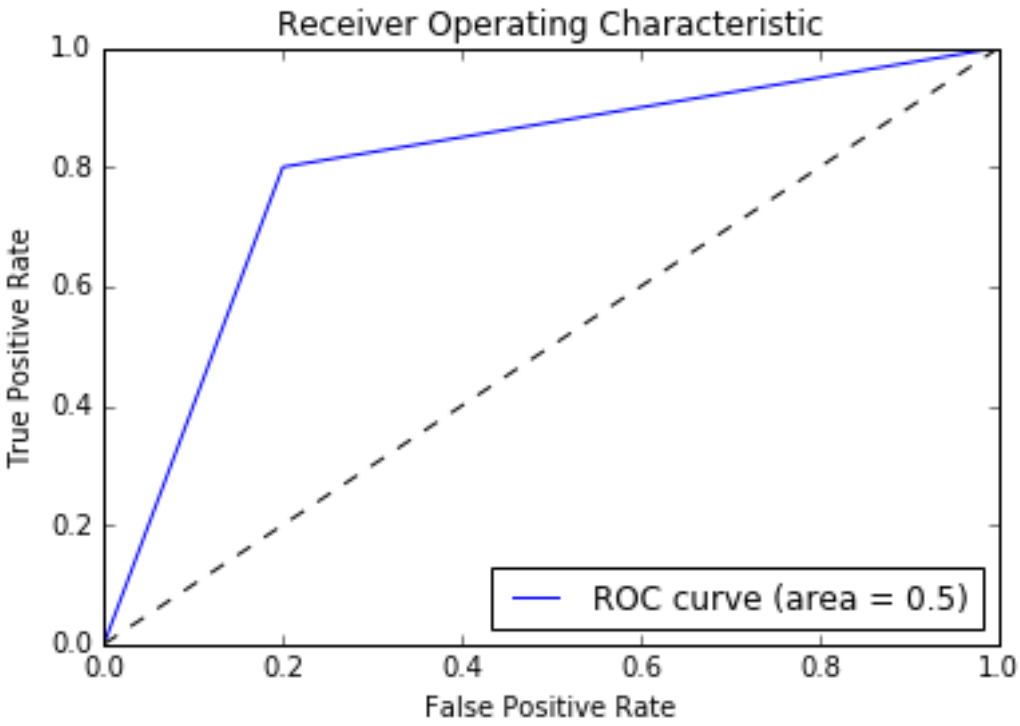
Positive Predicted Value:= 0.8

Negative Predicted Value:= 0.8

Sensitivity:= 0.8

Specificity:= 0.8

AUC: = 0.8 (from Graph 5.6)



Graph 5.6. Receiver operating characteristic curve for SVM algorithm

Bagging - Decision Tree as Base Learner:

In Table 5.7 the outputs of the bagging algorithm is summarised.

	Predicted Positive Class	**Predicted Negative Class**
Actual Positive Class	3	2
Actual Negative Class	1	4

Table 5.7. Results of bagging in a confusion matrix

Accuracy:= 0.7 (Average value of accuracy for 10 executions:= 0.54)

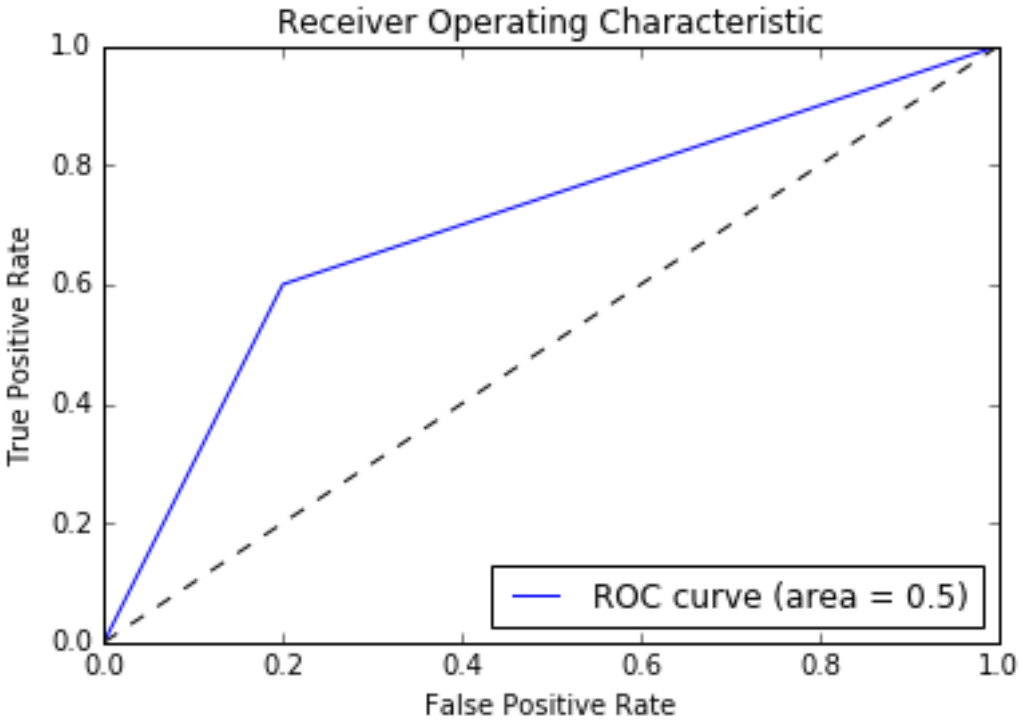
Positive Predicted Value:= 0.75

Negative Predicted Value:= 0.666666666667

Sensitivity:= 0.6

Specificity:= 0.8

AUC: = 0.7 (from Graph 5.7)



Graph 5.7. Receiver operating characteristic curve for bagging algorithm

AdaBoosting - Decision Tree as Base Learner:

In Table 5.8 the outputs of the adaboosting algorithm is summarised.

	Predicted Positive Class	**Predicted Negative Class**
Actual Positive Class	3	2
Actual Negative Class	1	4

Table 5.8. Results of adaboosting in a confusion matrix

Accuracy:= 0.7 (Average value of accuracy for 10 executions:= 0.51)

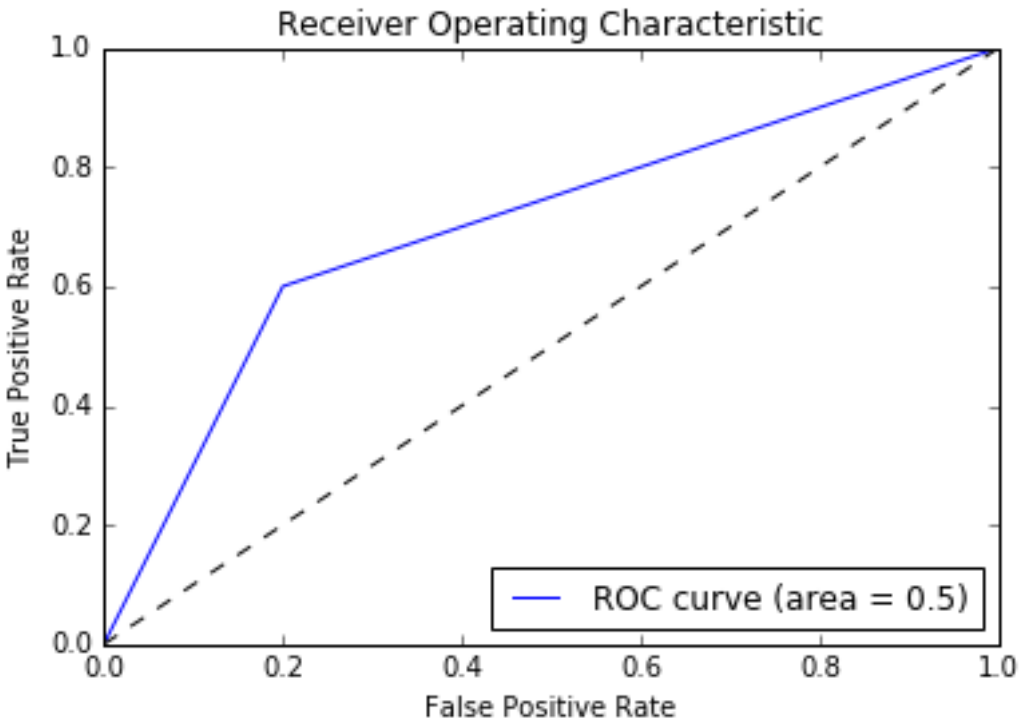
Positive Predicted Value:= 0.75

Negative Predicted Value:= 0.666666666667

Sensitivity:= 0.6

Specificity:= 0.8

AUC: = 0.7 (from Graph 5.8)



Graph 5.8. Receiver operating characteristic curve for adaboosting algorithm

Gradient Tree Boosting:

In Table 5.9 the outputs of the gradient tree boosting algorithm is summarised.

	Predicted Positive Class	**Predicted Negative Class**
Actual Positive Class	4	1
Actual Negative Class	3	2

Table 5.9. Results of gradient tree boosting in a confusion matrix

Accuracy:= 0.6 (Average value of accuracy for 10 executions:= 0.5)

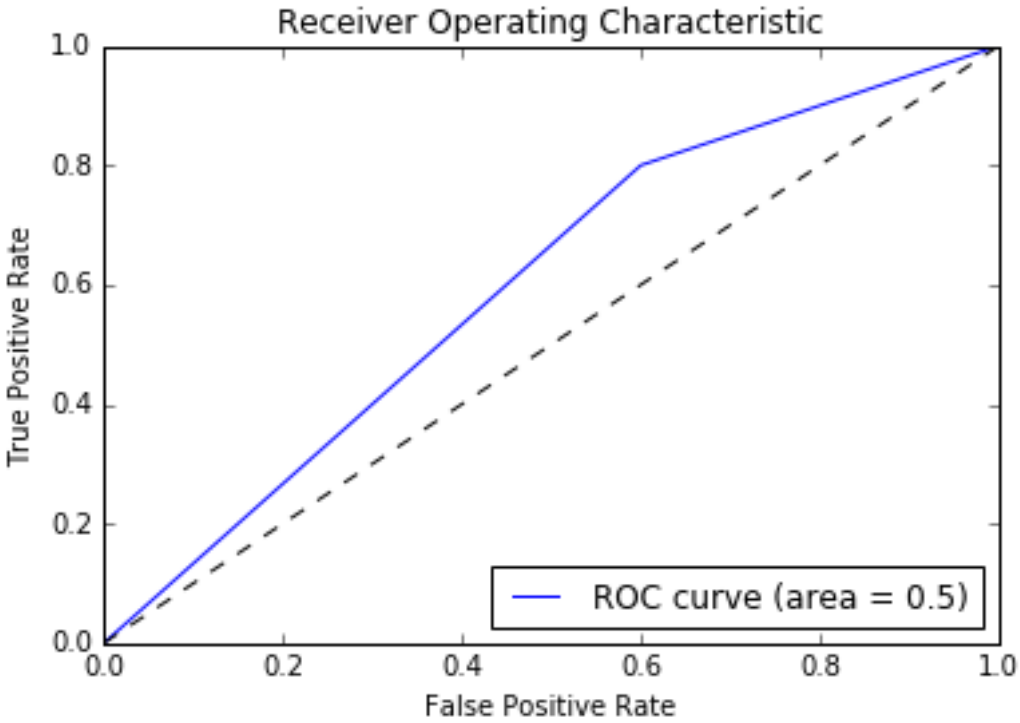
Positive Predicted Value:= 0.571428571429

Negative Predicted Value:= 0.666666666667

Sensitivity:= 0.8

Specificity:= 0.4

AUC: = 0.6 (from Graph 5.9)



Graph 5.9. Receiver operating characteristic curve for gradient tree boosting algorithm

Random Forest:

In Table 5.10 the outputs of the random forest algorithm is summarised.

	Predicted Positive Class	**Predicted Negative Class**
Actual Positive Class	4	1
Actual Negative Class	2	3

Table 5.10. Results of random forest in a confusion matrix

Accuracy:= 0.7 (Average value of accuracy for 10 executions:= 0.55)

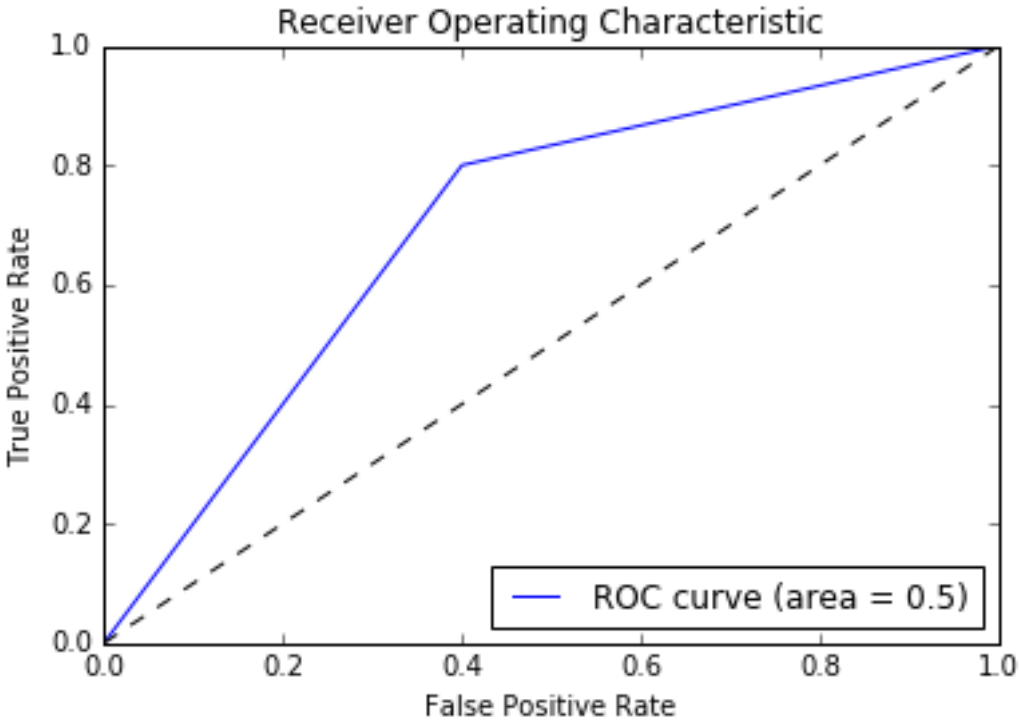
Positive Predicted Value:= 0.666666666667

Negative Predicted Value:= 0.75

Sensitivity:= 0.8

Specificity:= 0.6

AUC: = 0.7 (from Graph 5.10)



Graph 5.10. Receiver operating characteristic curve for random forest algorithm

Evaluation of Classification Algorithms:

From the algorithms which were analysed the best performing seem to be *K-Nearest Neighbours, Naive Bayes, Support Vector Machine* and *Logistic Regression,* which all had a value of 0.8 on all six metrics. However, the best out of the four algorithm was determined to be *K-Nearest* because it obtained a slightly better average accuracy from the 10 executions with a value of 0.59. The results from the algorithms aren't absolutely representative of their capabilities, because of the small training and testing sample. Having a bigger data set will allow to clearly distinguish the classification algorithm that has the best results for this problem.

The worst performance is shown by the *classification tree* algorithm with accuracy of 0.6. This is due to the small number of data to train and so it cannot cover all the possible cases when building a decision tree. Also, it is possible the algorithm

has fallen into a case of overfitting and cannot handle correctly any new data from the test set.

The ensemble algorithms show as expected better performance than the classification tree algorithm, as for they use the algorithm as the base learner. Although, the limited sample size reduces the capabilities of the algorithms to obtain high accuracy, having the same problem as the classification tree.

In comparison to the previous analysis [50], the data set was evaluated on two algorithms, on the *K-Nearest Neighbours* and *Support Vector Machine*. The results from these two algorithms are different from what is obtained here, but both have relative high accuracy as is concluded from this analysis. More specifically the k-nearest neighbours obtained an accuracy of 0.75, with misclassifying only 2 cases as positive. For the SVM algorithm it obtained an accuracy of 0.875, with misclassifying 1 case as positive. This difference can be caused by the different number of testing data and the way the data was split between testing and training sets, as stated previously.

Ranking of Classification Algorithms in Descending Order as shown in Table 5.11:

Algorithm	Accuracy	PPV	NPV	Sensitivity	Specificity	AUC
K-Nearest Neighbours	0.8	0.8	0.8	0.8	0.8	0.8
Naive Bayes	0.8	0.8	0.8	0.8	0.8	0.8
SVM	0.8	0.8	0.8	0.8	0.8	0.8
Logistic Regression	0.8	0.8	0.8	0.8	0.8	0.8
Random Forest	0.7	0.6667	0.75	0.8	0.6	0.7
Bagging	0.7	0.75	0.6667	0.6	0.8	0.7
Neural Network	0.7	0.75	0.6667	0.6	0.8	0.7
AdaBoosting	0.7	0.75	0.6667	0.6	0.8	0.7
Gradient Tree Boosting	0.6	0.5714	0.6667	0.8	0.4	0.6
Classification Tree	0.6	0.6667	0.5714	0.4	0.8	0.6

Table 5.11. Ranking of classification algorithms in descending order for experiential avoidance of smokers

The ordering has been done based on each algorithm's accuracy on the tested data set. For rating purposes when they have equal accuracy value then the rating of negative predicted value will be used followed by specificity, sensitivity and lastly the positive predicted value. For the rating of each algorithm, if they have the same values then the average accuracy of the 10 executions will also be taken into account.

For medical applications in machine learning it is more important to not wrongly classify a person as low risk when they are actually high risk, being the negative predicted value metric. In the other case, if someone is classified wrongly as high avoidance additional means can be taken to verify the classification before taking any measures , but will cause unnecessary concern to the person. Also, it's important to have accurate results for people who are actually low avoidance and have been predicted as low avoidance, obtaining the percentage from specificity. Usually in a large sample of medical data higher percentage will be classified as low rather than high avoidance. So

this will limit by a significant amount the total cases that will have to be validated individually by possibly more costly and time consuming methods [51].

5.2 Diagnosis of Eating Disorders

There is available a data sample from 79 people which contains measurements taken from the signals electrocardiogram (ECG), corrugator muscle (COR - using facial electromyography) and galvanic skin response (GSR) and also results from the questionnaire *Body Image Acceptance and Action Questionnaire* (BI-AAQ). This data was obtained from the Department of Psychology at the University of Cyprus. The measurements were taken following a specific procedure which was the same for all of the participants. For each person the measurements were taken from five different continuously time frames. From these time frames, the measurements from the first time frame are not used, but it's there so the person can be in a state of calm. For the following two, a short film was shown which should create neutral emotions to the viewer. Finally, for the last two, again, a short film was shown which this time should create negative emotions. The data obtained from the participants the Department of Psychology was able to classify them to two categories, the people which are at high risk (represented as a value of 1) and the people which are at low risk (represented as a value of 0) [50].

The BI-AAQ is a questionnaire that allows a person to self evaluate his or her own body image flexibility. A higher result obtained on the questionnaire have shown a link with an also increase in psychological flexibility and in body image satisfaction. Consequently the person is less likely to suffer from an eating disorder. On the questionnaire a ranking system is used from 1 (never true) to 7 (always true) and there are in total 12 questions. The final results are taken from the inverse rating from the questions and then summed up [39].

Based on a previous research [50], to obtain the best results in classifying a sample in the dataset the only signal that will be used in the supervised learning algorithms is that from the ECG. Since the first time frame obtained in the recording it is used for the purpose to relax the person and so only the remaining four will be used for classification. The feature obtained from the stream of data from each time frame

will be the mean value of that time frame. In addition to the ECG signal the results from the BI-AAQ will also be used as a feature.

From the data sample there are cases which it wasn't possible to determine the mean value of the ECG for one or more of the time frames, but not more than three. For these cases, the missing values were filled in with an estimated value using the mean of the other available time frames, excluding the first one. Most classification algorithms cannot handle missing values on their own and as such require some pre-processing so the data isn't discarded.

For the training and testing of the classification algorithms which will be used it's necessary to separate the data sample. It will be split into two class balanced subsamples which $\lceil 2/3 \rceil$ will be used for training and the rest will be used for testing the algorithms. That is, $\lceil 2/3 \rceil$ of the data which is actually classified as acceptance and $\lceil 2/3 \rceil$ of the data which is actually classified as avoidance are taken for the training and the rest are used for the testing subsample. So they are in total 53 data samples for training (28 high risk and 25 low risk) and 26 data samples for testing (14 high risk and 12 low risk).

In comparison to the previous analysis [50], a different distribution of the data set was used. In total 57 data samples were used for the training and 14 for the test set (having 8 less data samples). Also, the distribution of the data wasn't necessary done in a classed balanced way. In the analysis only the data from the signal ECG was used and the BI-AAQ results weren't included.

Each of the algorithms will be executed 10 times with a different distribution of the data between training and testing. But between the algorithms the 10 data distributions used will be the same. In the analysis of the algorithms the results will be used from the distribution that returns the best results.

Logistic Regression:

In Table 5.12 the outputs of the logistic regression algorithm is summarised.

	Predicted Positive Class	Predicted Negative Class
Actual Positive Class	13	1
Actual Negative Class	2	10

Table 5.12. Results of logistic regression in a confusion matrix

Accuracy:= 0.884615384615 (Average value of accuracy for 10 executions:= 0.754)

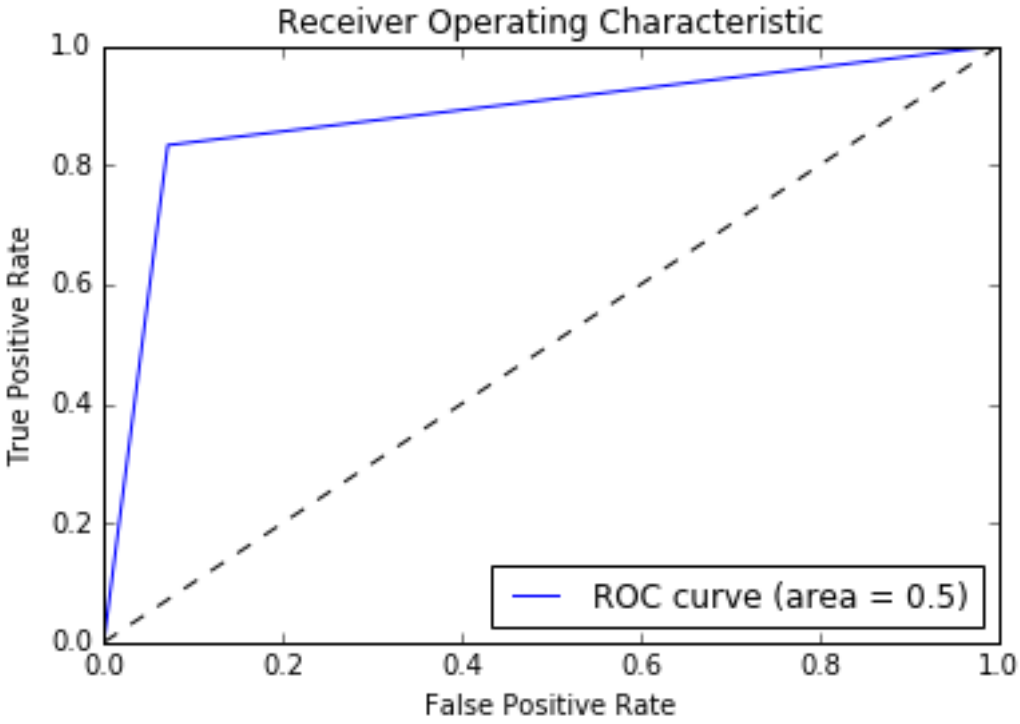
Positive Predicted Value:= 0.866666666667

Negative Predicted Value:= 0.909090909091

Sensitivity:= 0.928571428571

Specificity:= 0.833333333333

AUC: = 0.880952380952 (from Graph 5.11)



Graph 5.11. Receiver operating characteristic curve for logistic regression algorithm

Naive Bayers:

In Table 5.13 the outputs of the Naive Bayes algorithm is summarised.

	Predicted Positive Class	Predicted Negative Class
Actual Positive Class	11	3
Actual Negative Class	1	11

Table 5.13. Results of Naive Bayes in a confusion matrix

Accuracy:= 0.846153846154 (Average value of accuracy for 10 executions:= 0.797)

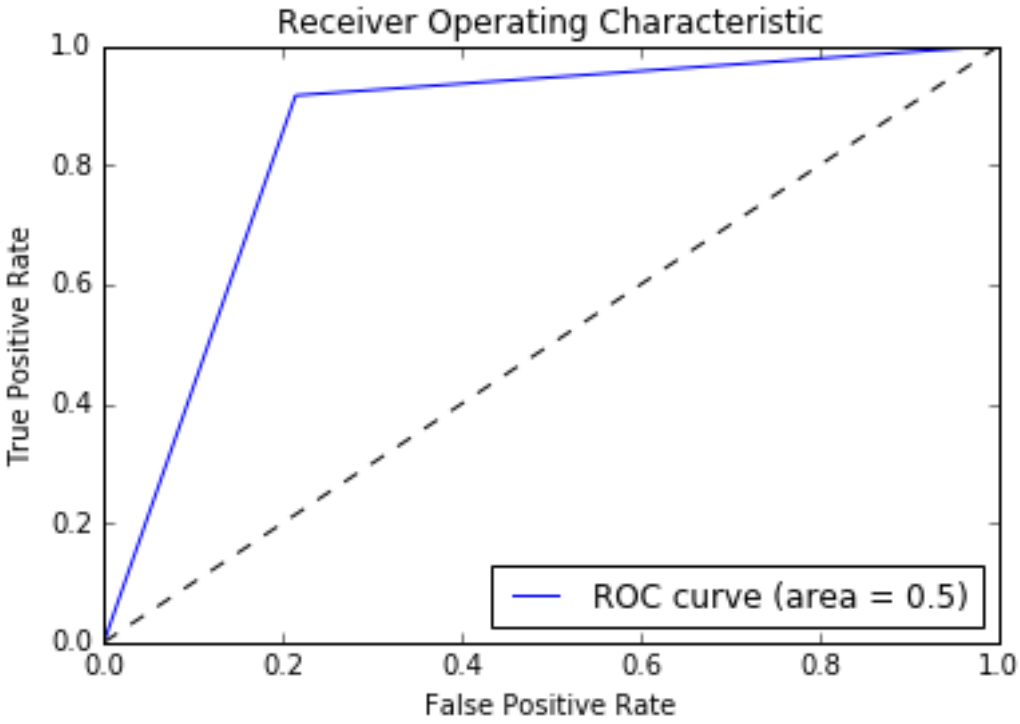
Positive Predicted Value:= 0.916666666667

Negative Predicted Value:= 0.785714285714

Sensitivity:= 0.785714285714

Specificity:= 0.916666666667

AUC: = 0.85119047619 (from Graph 5.12)



Graph 5.12. Receiver operating characteristic curve for Naive Bayes algorithm

K-Nearest Neighbours (K = 3):

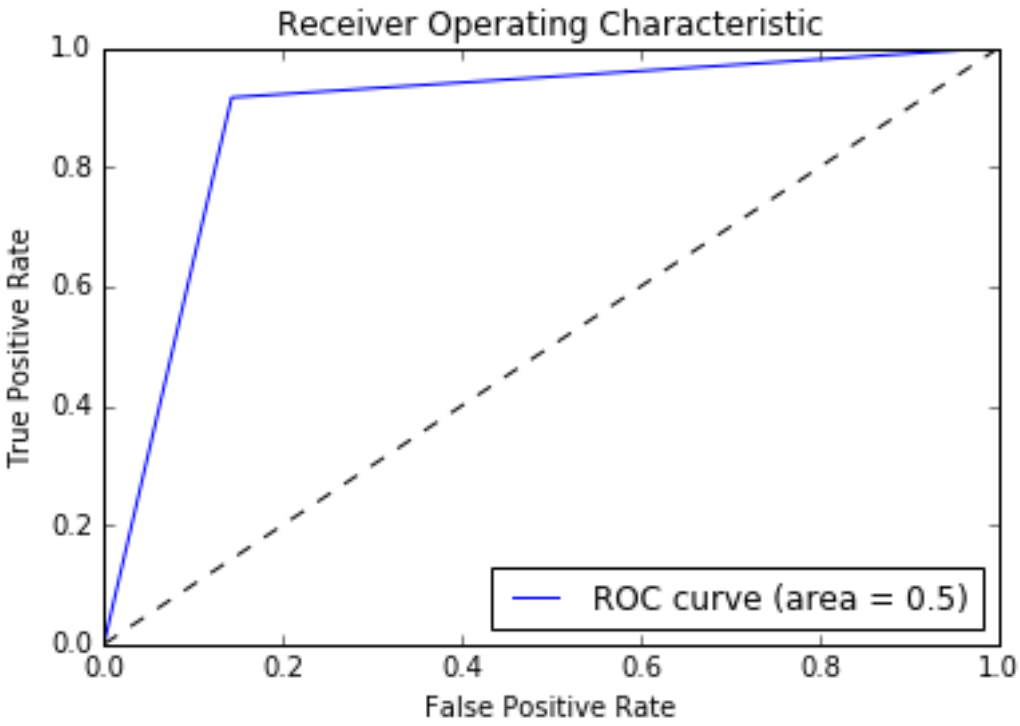
The number of nearest points to examine has been determined to be 3. The value has been obtained by executing on the dataset the algorithm with the number of clusters from 1 to 5. With using 3 nearest neighbours resulted to the highest accuracy and as such will be used to evaluate the algorithm.

In Table 5.14 the outputs of the k-nearest neighbours algorithm is summarised.

	Predicted Positive Class	**Predicted Negative Class**
Actual Positive Class	12	2
Actual Negative Class	1	11

Table 5.14. Results of k-nearest neighbours in a confusion matrix

Accuracy:= 0.884615384615 (Average value of accuracy for 10 executions:= 0.76)
Positive Predicted Value:= 0.923076923077
Negative Predicted Value:= 0.846153846154
Sensitivity:= 0.857142857143
Specificity:= 0.916666666667
AUC: = 0.886904761905 (from Graph 5.13)



Graph 5.13. Receiver operating characteristic curve for k-nearest algorithm

Classification Tree:

In Table 5.15 the outputs of the classification tree algorithm is summarised.

	Predicted Positive Class	**Predicted Negative Class**
Actual Positive Class	13	1
Actual Negative Class	2	10

Table 5.15. Results of classification tree in a confusion matrix

Accuracy:= 0.884615384615 (Average value of accuracy for 10 executions:= 0.7595)

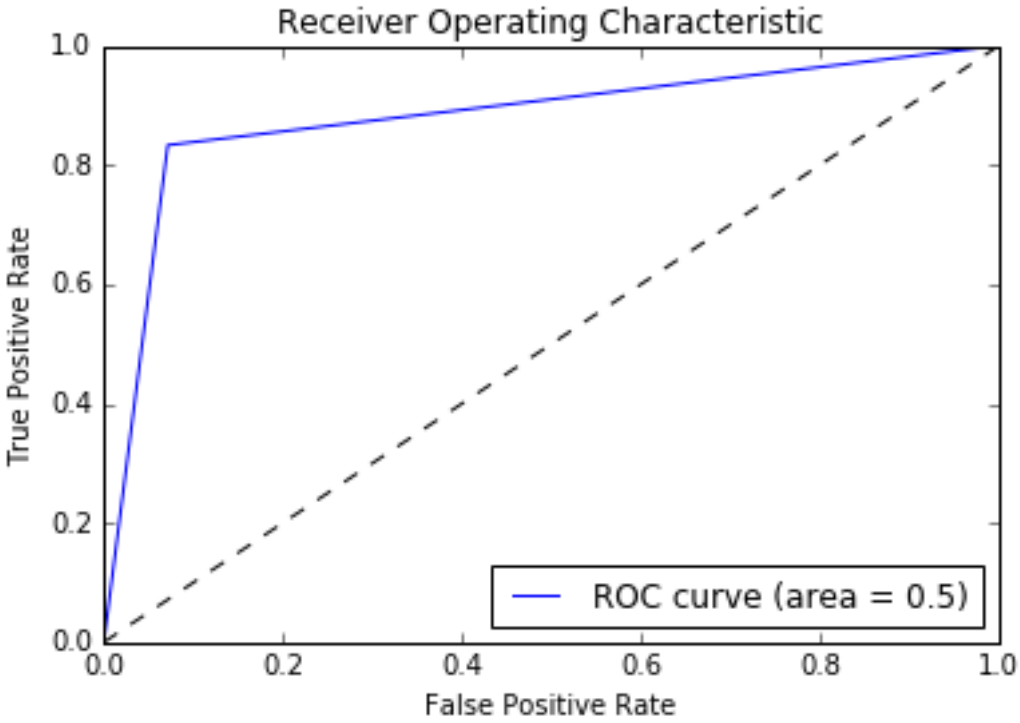
Positive Predicted Value:= 0.866666666667

Negative Predicted Value:= 0.909090909091

Sensitivity:= 0.928571428571

Specificity:= 0.833333333333

AUC: = 0.880952380952 (from Graph 5.14)



Graph 5.14. Receiver operating characteristic curve for classification tree algorithm

Neural Network:

In Table 5.16 the outputs of the neural network algorithm is summarised.

	Predicted Positive Class	**Predicted Negative Class**
Actual Positive Class	11	3
Actual Negative Class	2	10

Table 5.16. Results of neural network in a confusion matrix

Accuracy:= 0.807692307692 (Average value of accuracy for 10 executions:= 0.6957)

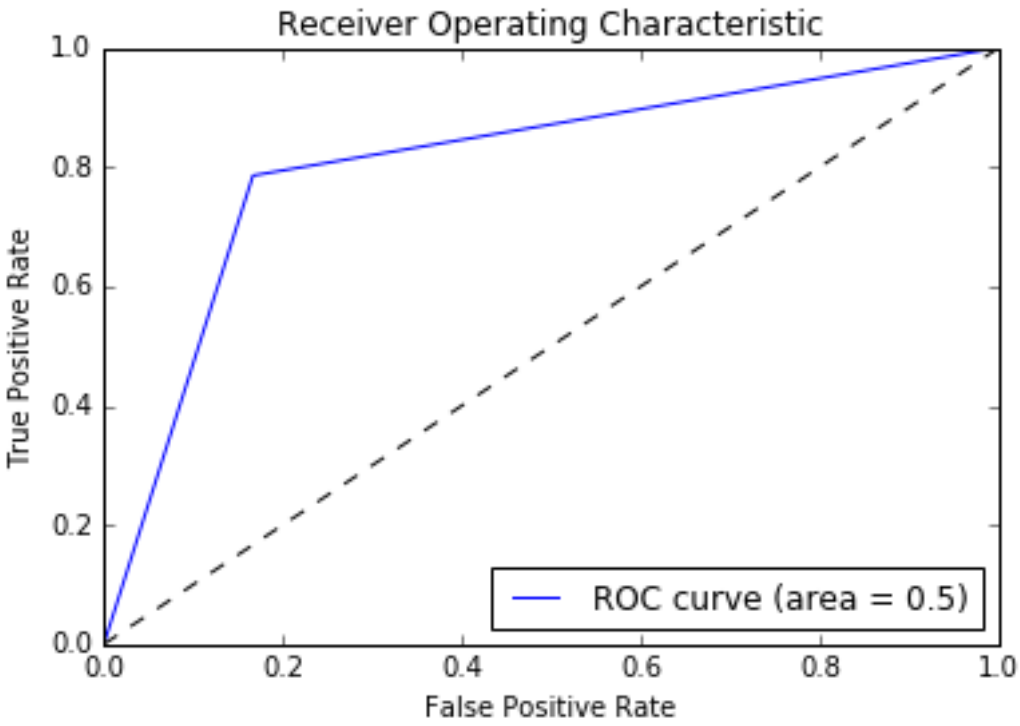
Positive Predicted Value:= 0.846153846154

Negative Predicted Value:= 0.769230769231

Sensitivity:= 0.785714285714

Specificity:= 0.833333333333

AUC: = 0.809523809524 (from Graph 5.15)



Graph 5.15. Receiver operating characteristic curve for neural network algorithm

SVM:

In Table 5.17 the outputs of the SVM algorithm is summarised.

	Predicted Positive Class	**Predicted Negative Class**
Actual Positive Class	13	1
Actual Negative Class	1	11

Table 5.17. Results of SVM in a confusion matrix

Accuracy:= 0.923076923077 (Average value of accuracy for 10 executions:= 0.799)

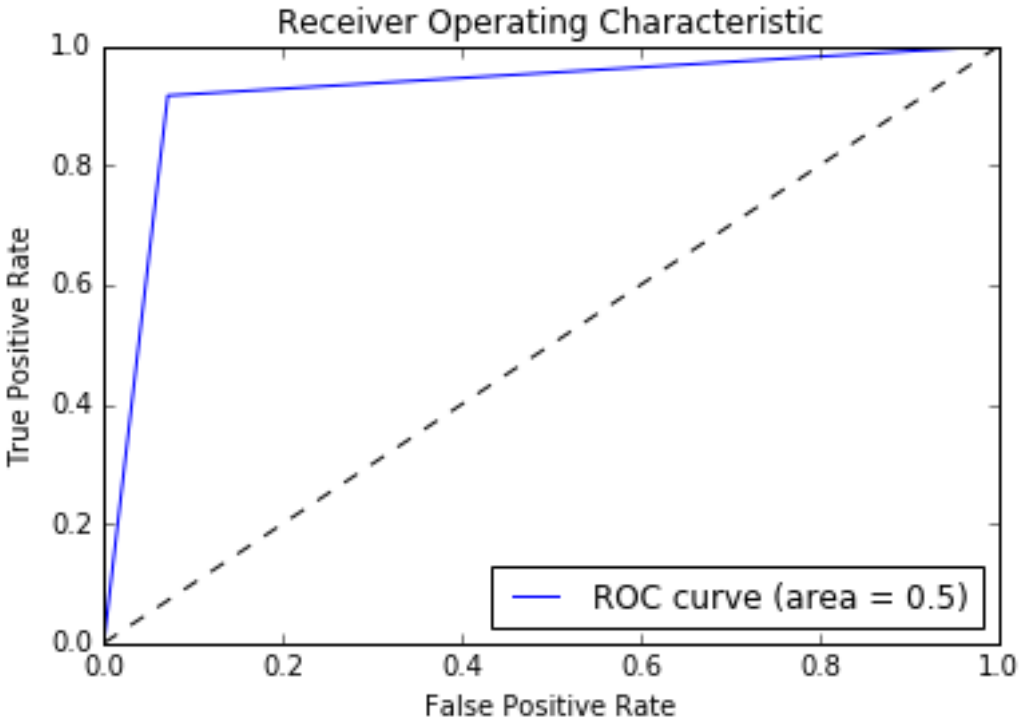
Positive Predicted Value:= 0.928571428571

Negative Predicted Value:= 0.916666666667

Sensitivity:= 0.928571428571

Specificity:= 0.916666666667

AUC: = 0.922619047619 (from Graph 5.16)



Graph 5.16. Receiver operating characteristic curve for SVM algorithm

Bagging - Decision Tree as Base Learner:

In Table 5.18 the outputs of the bagging algorithm is summarised.

	Predicted Positive Class	**Predicted Negative Class**
Actual Positive Class	12	2
Actual Negative Class	1	11

Table 5.18. Results of bagging in a confusion matrix

Accuracy:= 0.884615384615 (Average value of accuracy for 10 executions:= 0.7875)

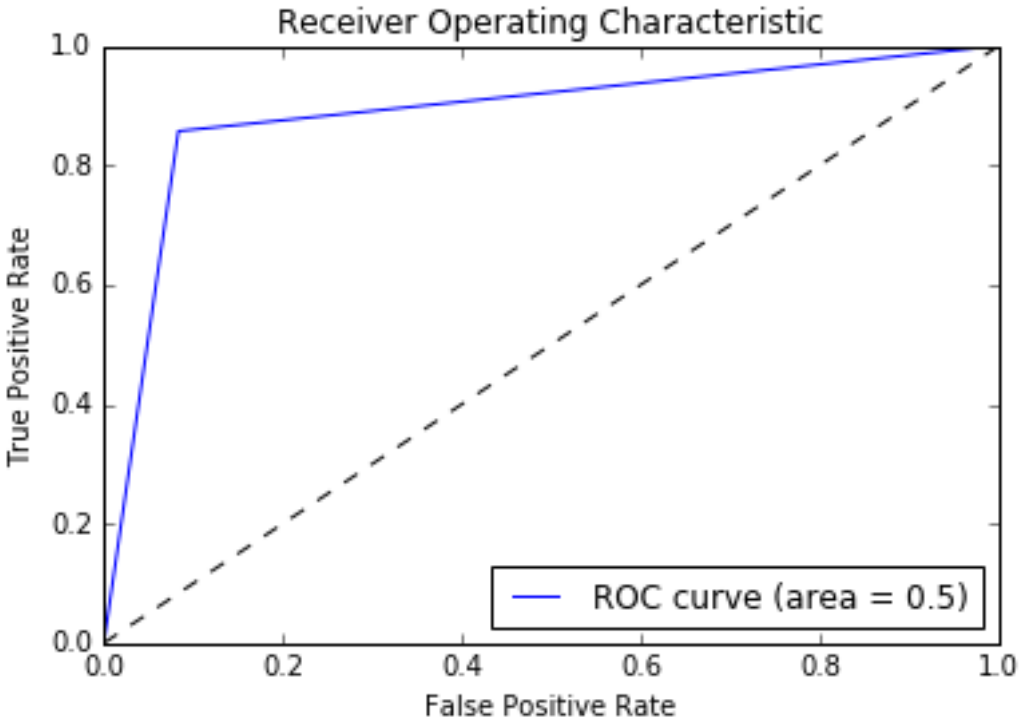
Positive Predicted Value:= 0.923076923077

Negative Predicted Value:= 0.846153846154

Sensitivity:= 0.857142857143

Specificity:= 0.916666666667

AUC: = 0.886904761905 (from Graph 5.17)



Graph 5.17. Receiver operating characteristic curve for bagging algorithm

AdaBoosting - Decision Tree as Base Learner:

In Table 5.19 the outputs of the adaboosting algorithm is summarised.

	Predicted Positive Class	**Predicted Negative Class**
Actual Positive Class	12	2
Actual Negative Class	1	11

Table 5.19. Results of adaboosting in a confusion matrix

Accuracy:= 0.884615384615 (Average value of accuracy for 10 executions:= 0.771)

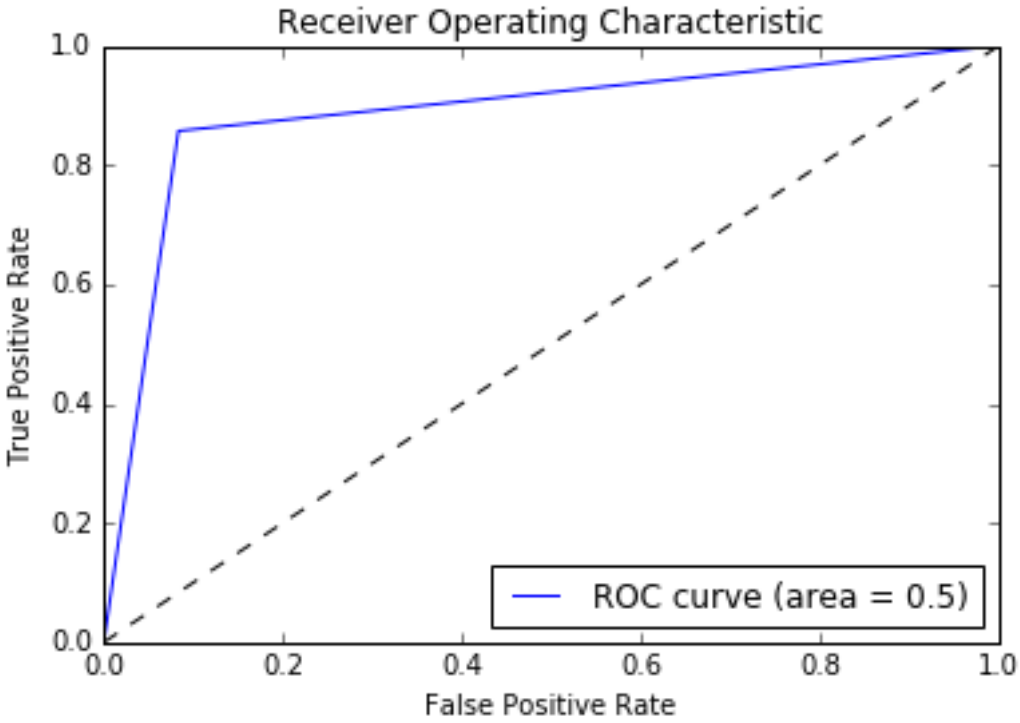
Positive Predicted Value:= 0.923076923077

Negative Predicted Value:= 0.846153846154

Sensitivity:= 0.857142857143

Specificity:= 0.916666666667

AUC: = 0.886904761905 (from Graph 5.18)



Graph 5.18. Receiver operating characteristic curve for adaboosting algorithm

Gradient Tree Boosting:

In Table 5.20 the outputs of the gradient tree boosting algorithm is summarised.

	Predicted Positive Class	**Predicted Negative Class**
Actual Positive Class	12	2
Actual Negative Class	1	11

Table 5.20. Results of gradient tree boosting in a confusion matrix

Accuracy:= 0.884615384615 (Average value of accuracy for 10 executions:= 0.7555)

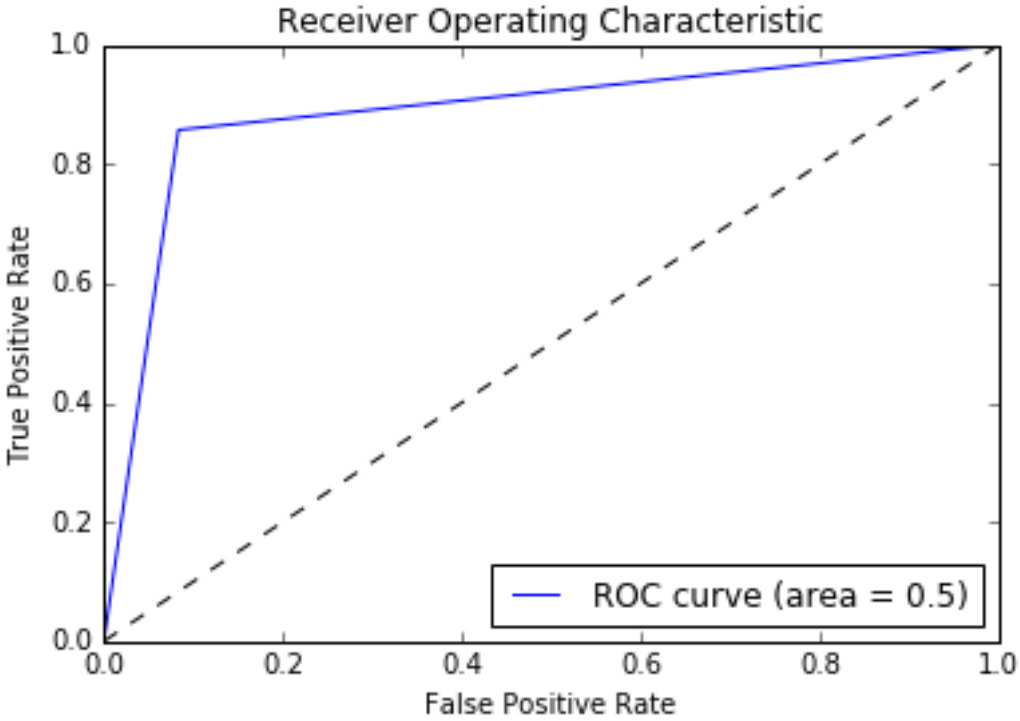
Positive Predicted Value:= 0.923076923077

Negative Predicted Value:= 0.846153846154

Sensitivity:= 0.857142857143

Specificity:= 0.916666666667

AUC: = 0.886904761905 (from Graph 5.19)



Graph 5.19. Receiver operating characteristic curve for gradient tree boosting algorithm

Random Forests:

In Table 5.21 the outputs of the random forest algorithm is summarised.

	Predicted Positive Class	**Predicted Negative Class**
Actual Positive Class	13	1
Actual Negative Class	1	11

Table 5.21. Results of random forest in a confusion matrix

Accuracy:= 0.923076923077 (Average value of accuracy for 10 executions:= 0.8075)

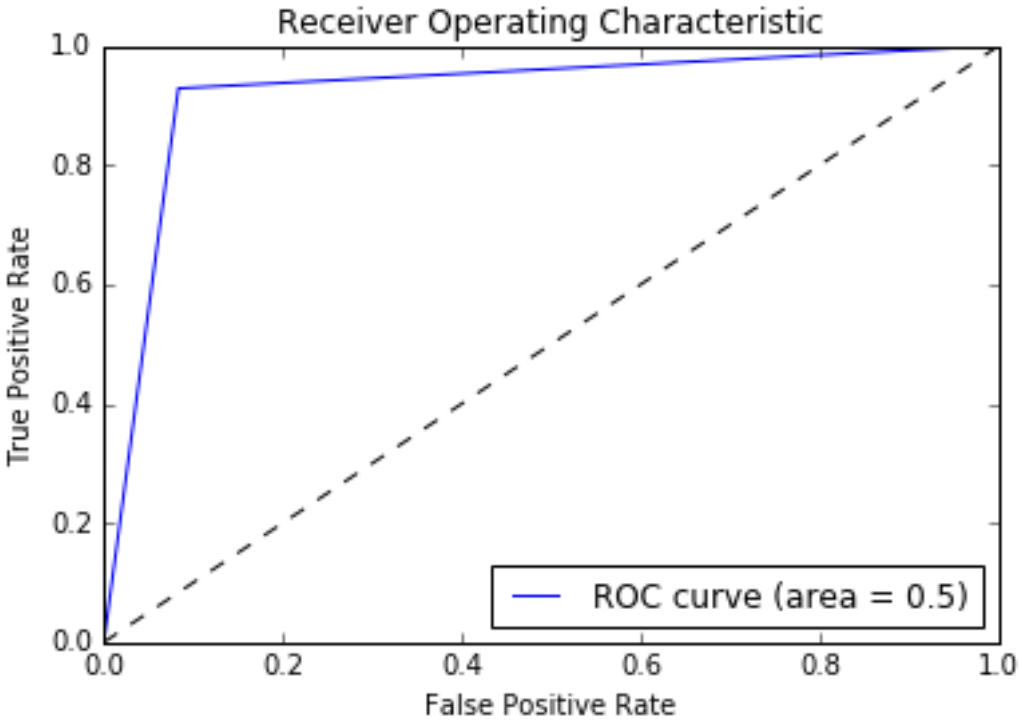
Positive Predicted Value:= 0.928571428571

Negative Predicted Value:= 0.916666666667

Sensitivity:= 0.928571428571

Specificity:= 0.916666666667

AUC: = 0.922619047619 (from Graph 5.20)



Graph 5.20. Receiver operating characteristic curve for random forest algorithm

Evaluation of Classification Algorithms:

The algorithms that have the best performance on all six metrics is the *Random Forest* and *Support Vector Machine* on the 10 executions from the algorithms tested. They are able to classify correctly the 26 data samples in the test set except from 2, one which should be classifies as high risk and the other as low risk. All other algorithms can classify correctly the test set except from 3 samples with at least one from each class. Exception is the *Neural Network* which shows the worst results, unable to correctly classify 5 data samples from the test set. Comparing the two algorithms the one that achieved more consistently better results on the 10 executions is determined to

be the *Random Forest*, because it achieves a better average accuracy with 0.8075 compared to 0.799 of that of the *Support Vector Machine*.

While all algorithms' results are sufficient, the *Neural Network* (NN) has the biggest drop in performance with an accuracy of around 0.8. This can be resulted to the need of NNs for high tuning and parameter optimisation compared to the other algorithms. Also, there might be a case where the network is over trained and can memorise the training data, unable to generalise to the test data. Lastly, the network can benefit from multiple executions of each of the data distributions to training and testing sets. This is because of the randomised weights given to each neuron.

In comparison to the previous analysis [50], the data set was evaluated on two algorithms, on the *K-Nearest Neighbours* and *Support Vector Machine*. The results from the SVM is very similar to what is obtained here, while the k-nearest neighbours has a larger difference. More specifically the k-nearest neighbours obtained an accuracy of 0.79, with misclassifying only 3 cases as negative. For the SVM algorithm it obtained an accuracy of 0.93, with misclassifying 1 case as negative. This difference can be caused by the different number of testing data and the way the data was split between testing and training sets, as stated previously.

Ranking of Classification Algorithms in Descending Order as shown in Table 5.22:

Algorithm	Accuracy	PPV	NPV	Sensitivity	Specificity	AUC
Random Forest	0.9231	0.9286	0.9167	0.9286	0.9167	0.9226
SVM	0.9231	0.9286	0.9167	0.9286	0.9167	0.9226
Classification Tree	0.8846	0.8667	0.9091	0.9286	0.8333	0.8806
Logistic Regression	0.8846	0.8667	0.9091	0.9286	0.8333	0.8806
Bagging	0.8846	0.9231	0.8462	0.8571	0.9167	0.8869
AdaBoosting	0.8846	0.9231	0.8462	0.8571	0.9167	0.8869
K-Nearest Neighbours	0.8846	0.9231	0.8462	0.8571	0.9167	0.8869
Gradient Tree Boosting	0.8846	0.9231	0.8462	0.8571	0.9167	0.8869
Naive Bayes	0.8846	0.9167	0.7857	0.7857	0.9167	0.8512
Neural Network	0.8077	0.8461	0.7692	0.7857	0.8333	0.8095

Table 5.22. Ranking of classification algorithms in descending order for diagnosis of eating disorders

The ordering has been done based on each algorithm's accuracy on the tested data set. For rating purposes when they have equal accuracy value then the rating of negative predicted value will be used followed by specificity, sensitivity and lastly the positive predicted value. For the rating of each algorithm, if they have the same values then the average accuracy of the 10 executions will also be taken into account.

For medical applications in machine learning it is more important to not wrongly classify a person as low risk when they are actually high risk, being the negative predicted value metric. In the other case, if someone is classified wrongly as high risk additional means can be taken to verify the classification before taking any measures , but will cause unnecessary concern to the person. Also, it's important to have accurate results for people who are actually low risk and have been predicted as low risk, obtaining the percentage from specificity. Usually in a large sample of medical data higher percentage will be classified as low rather than high risk. So this will limit by a

significant amount the total cases that will have to be validated individually by possibly more costly and time consuming methods [51].

5.3 Diagnosis of Experiential Avoidance for Anxiety

There is available a data sample from 51 people which contains measurements taken from the signals electrocardiogram (ECG), galvanic skin response (GSR), orbicularis oculi muscle (ORB - using facial electromyography), corrugator muscle (COR - using facial electromyography) and zygomaticus muscle (ZOR - using facial electromyography). This data was obtained from the Department of Psychology at the University of Cyprus. A total of 72 different recordings were taken in a single session for each participant. For each of the 72 tasks a different image is shown, which should stimulate the person differently based on if they show signs of anxiety or not. The data obtained from the participants the Department of Psychology was able to classify them to two categories, the people which are at high risk (represented as a value of 1) and the people which are at low risk (represented as a value of 0). This data sample wasn't examined in the previous analysis that was conducted [50] as for the data samples weren't available and not all samples were classified yet so they can be used in a supervised learning model.

For each time frame the data from the stream will be summarised into features, which will be obtained from the mean value of each segment and for each signal. The ECG signal is first processed and the heart rate per minute is calculated for each time stamp of the signal. Then the mean value of the heat rate per minute is calculated instead.

Having a total of 360 features, 72 time frames for each of the five signals, it is necessary to determine which of the signals are useful to positively effect in estimating the classification output from the supervised classification algorithms. To select the optimal signal features the unique combination of the signals will be each used to train and test a classification algorithm. The classification algorithm which is used is the *Random Forest* classifier, because it uses as a base classifier the decision tree which it can rank the importance of each feature and can effectively handle small and large number of features.

Selected Features	**Accuracy**
ecg	0.5625
gsr	0.5
orb	0.5625
cor	0.75
zyg	0.625
ecg, gsr	0.625
ecg, orb	0.5625
ecg, cor	0.6875
ecg, zyg	0.625
gsr, orb	0.375
gsr, cor	0.5
gsr, zyg	0.625
orb, cor	0.75
orb, zyg	0.625
cor, zyg	0.625
ecg, gsr, orb	0.5
ecg, gsr, cor	0.6875
ecg, gsr, zyg	0.5625
ecg, orb, cor	0.625

ecg, orb, zyg	0.6875
ecg, cor, zyg	0.625
gsr, orb, cor	0.5625
gsr, orb, zyg	0.625
gsr, cor, zyg	0.6875
orb, cor, zyg	0.6875
ecg, gsr, orb, cor	0.5625
ecg, gsr, orb, zyg	0.6875
ecg, gsr, cor, zyg	0.625
ecg, orb, cor, zyg	0.6875
gsr, orb, cor, zyg	0.625
ecg, gsr, orb, cor, zyg	0.75

Table 5.23. Accuracy results for the combination of each signal set when executed on the *RandomForest* algorithm

In Table 5.23, the signal combinations that result to the highest accuracy are [cor], [orb, cor] and [ecg, gsr, orb, cor, zyg] with a value of 0.75 accuracy. From the three previous sets selected it's preferable to use the set that contains only the mean values of the corrugator muscle sensor, rather than using the corrugator and orbicularis oculi muscle sensors or all five signals that were tested. Having only one signal, taking into account that each signal contains a total of 72 time frames and hence features, is more beneficial when it doesn't affect the accuracy because it's less processing demanding for the algorithms, avoids problems for some algorithms that can't handle a large amount of features and removes features that have a negative impact on the final result. In addition, the *Random Forest* algorithm can handle irrelevant features and minimise their impact on deciding the class prediction, giving more weight towards the single signal set. So selecting the set which contains only one feature will result into a higher accuracy for other non-recursive algorithms.

With selecting only the feature that brings the highest results from the algorithm for this problem, being the COR signal trials only, the data can now be used to analyse the classification algorithms. For the training and testing of the classification algorithms which will be used it's necessary to separate the data sample. It will be split into two class balanced subsamples which $\lceil 2/3 \rceil$ will be used for training and the rest will be used for testing the algorithms. That is, $\lceil 2/3 \rceil$ of the data which is actually classified as high and $\lceil 2/3 \rceil$ of the data which is actually classified as low are taken for the training and the rest are used for the testing subsample. So they are in total 35 data samples for training (18 high risk and 17 low risk) and 16 data samples for testing (8 high risk and 8 low risk).

Each of the algorithms will be executed 10 times with a different distribution of the data between training and testing. But between the algorithms the 10 data distributions used will be the same. In the analysis of the algorithms the results will be used from the distribution that returns the best results.

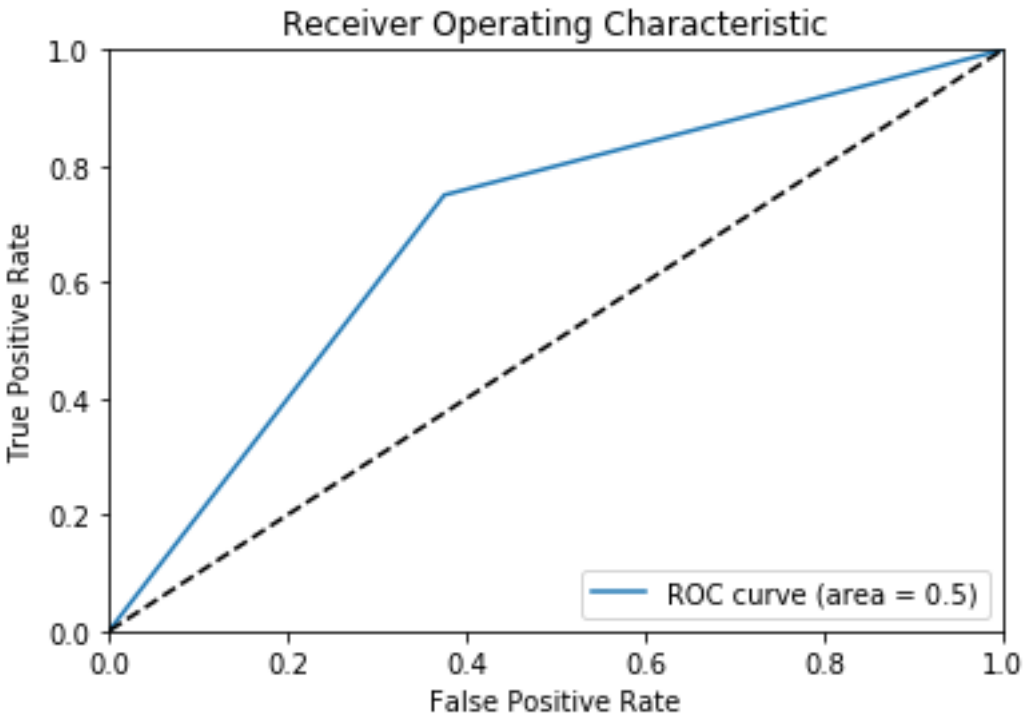
Logistic Regression:

In Table 5.24 the outputs of the logistic regression algorithm is summarised.

	Predicted Positive Class	**Predicted Negative Class**
Actual Positive Class	5	3
Actual Negative Class	2	6

Table 5.24. Results of logistic regression in a confusion matrix

Accuracy:= 0.6875 (Average value of accuracy for 10 executions:= 0.45625)
Positive Predicted Value:= 0.666666666667
Negative Predicted Value:= 0.714285714286
Sensitivity:= 0.75
Specificity:= 0.625
AUC: = 0.6875 (from Graph 5.21)



Graph 5.21. Receiver operating characteristic curve for logistic regression algorithm

Naive Bayers:

In Table 5.25 the outputs of the Naive Bayes algorithm is summarised.

	Predicted Positive Class	**Predicted Negative Class**
Actual Positive Class	8	0
Actual Negative Class	5	3

Table 5.25. Results of Naive Bayes in a confusion matrix

Accuracy:= 0.6875 (Average value of accuracy for 10 executions:= 0.53125)

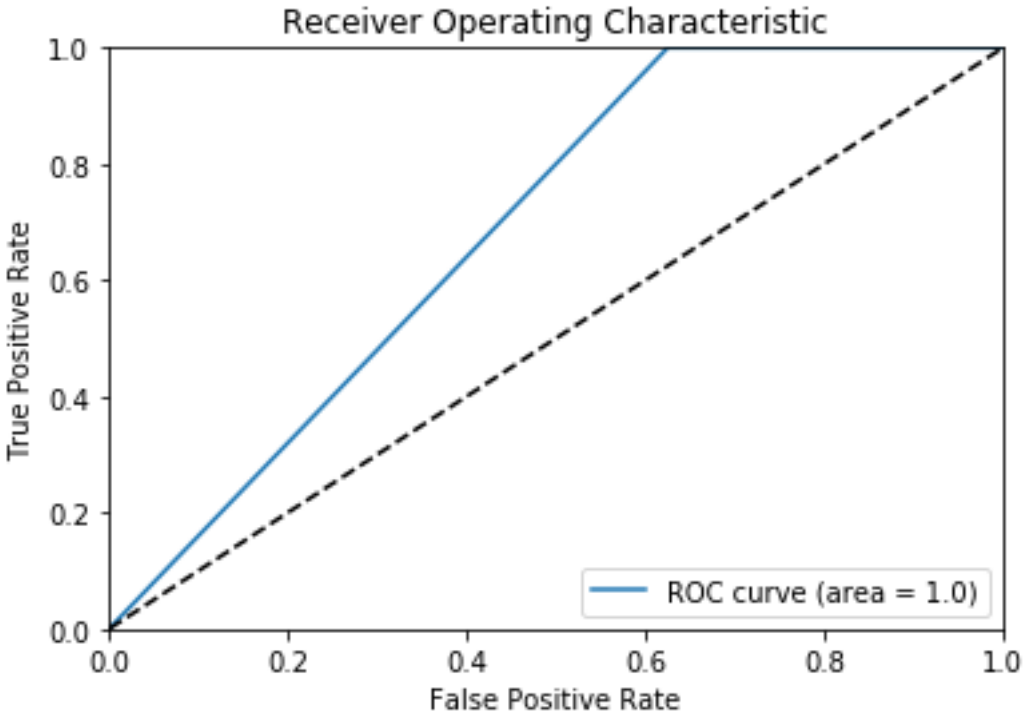
Positive Predicted Value:= 0. 615384615385

Negative Predicted Value:= 1.0

Sensitivity:= 1.0

Specificity:= 0.375

AUC: = 0.6875 (from Graph 5.22)



Graph 5.22. Receiver operating characteristic curve for Naive Bayes algorithm

K-Nearest Neighbours (K = 3):

The number of nearest points to examine has been determined to be 3. The value has been obtained by executing on the dataset the algorithm with the number of clusters from 1 to 5. With using 3 nearest neighbours resulted to the highest accuracy and as such will be used to evaluate the algorithm.

In Table 5.26 the outputs of the k-nearest neighbours algorithm is summarised.

	Predicted Positive Class	**Predicted Negative Class**
Actual Positive Class	6	2
Actual Negative Class	3	5

Table 5.26. Results of k-nearest neighbours in a confusion matrix

Accuracy:= 0.6875 (Average value of accuracy for 10 executions:= 0.6125)

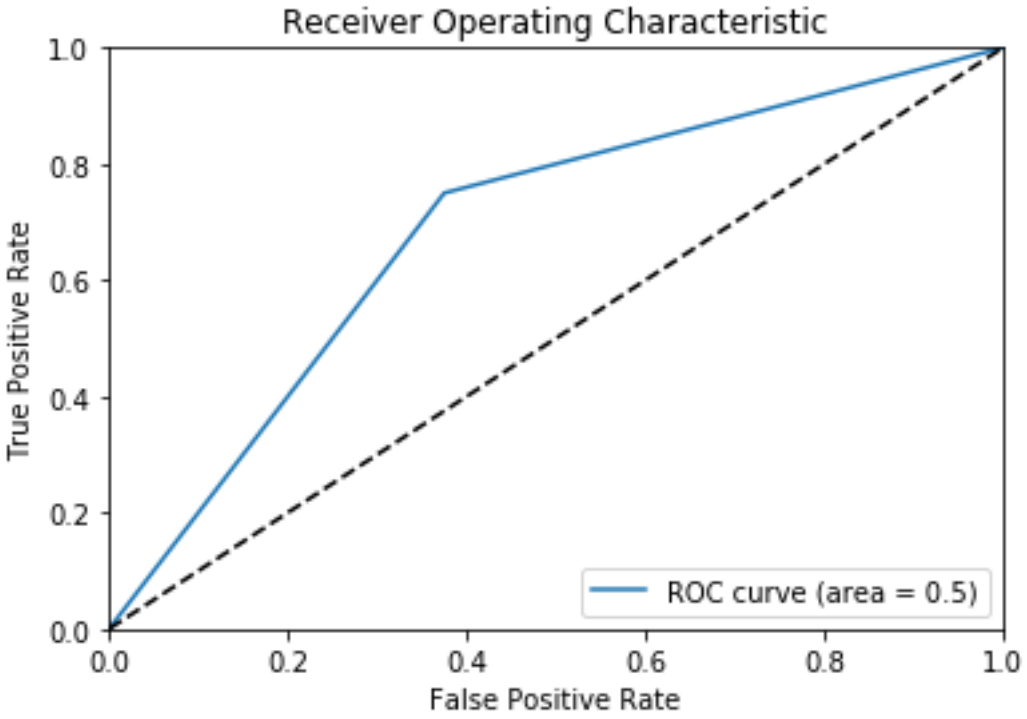
Positive Predicted Value:= 0.666666666667

Negative Predicted Value:= 0.714285714286

Sensitivity:= 0.75

Specificity:= 0.625

AUC: = 0.6875 (from Graph 5.23)



Graph 5.23. Receiver operating characteristic curve for k-nearest algorithm

Classification Tree:

In Table 5.27 the outputs of the classification tree algorithm is summarised.

	Predicted Positive Class	**Predicted Negative Class**
Actual Positive Class	6	2
Actual Negative Class	3	5

Table 5.27. Results of classification tree in a confusion matrix

Accuracy:= 0.6875 (Average value of accuracy for 10 executions:= 0.50625)

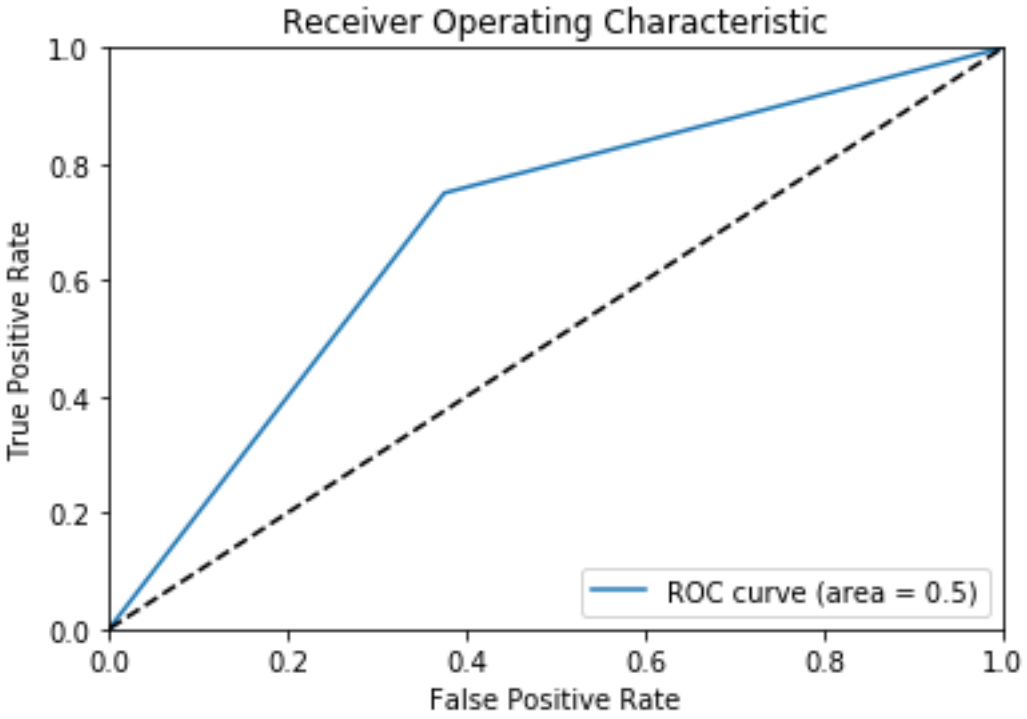
Positive Predicted Value:= 0.666666666667

Negative Predicted Value:= 0.714285714286

Sensitivity:= 0.75

Specificity:= 0.625

AUC: = 0.6875 (from Graph 5.24)



Graph 5.24. Receiver operating characteristic curve for classification tree algorithm

Neural Network:

In Table 5.28 the outputs of the neural network algorithm is summarised.

	Predicted Positive Class	**Predicted Negative Class**
Actual Positive Class	7	1
Actual Negative Class	5	3

Table 5.28. Results of neural network in a confusion matrix

Accuracy:= 0.625 (Average value of accuracy for 10 executions:= 0.5)

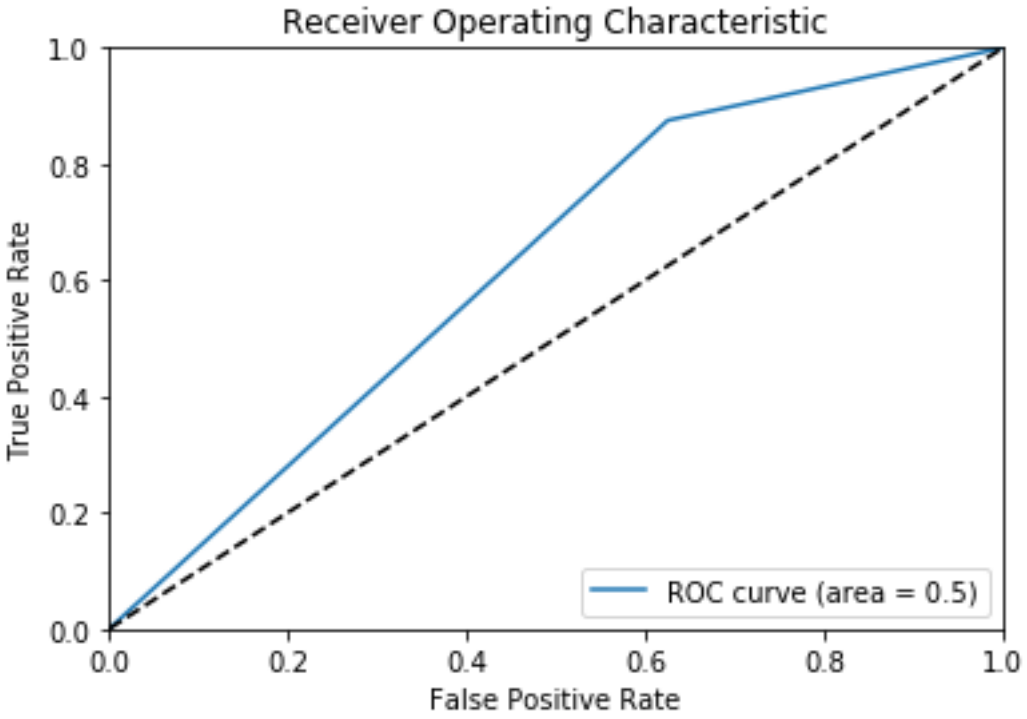
Positive Predicted Value:= 0.583333333333

Negative Predicted Value:= 0.75

Sensitivity:= 0.875

Specificity:= 0.375

AUC: = 0.625 (from Graph 5.25)



Graph 5.25. Receiver operating characteristic curve for neural network algorithm

SVM:

In Table 5.29 the outputs of the SVM algorithm is summarised.

	Predicted Positive Class	**Predicted Negative Class**
Actual Positive Class	6	2
Actual Negative Class	3	5

Table 5.29. Results of SVM in a confusion matrix

Accuracy:= 0.6875 (Average value of accuracy for 10 executions:= 0.4625)

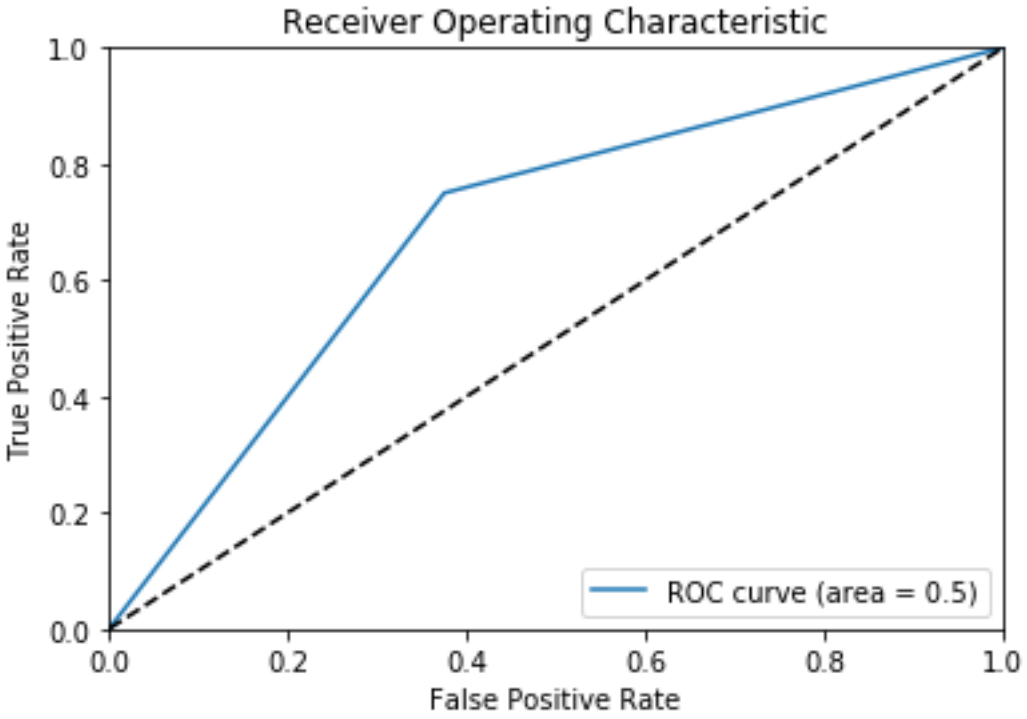
Positive Predicted Value:= 0.666666666667

Negative Predicted Value:= 0.714285714286

Sensitivity:= 0.75

Specificity:= 0.625

AUC: = 0.6875 (from Graph 5.26)



Graph 5.26. Receiver operating characteristic curve for SVM algorithm

Bagging - Decision Tree as Base Learner:

In Table 5.30 the outputs of the bagging algorithm is summarised.

	Predicted Positive Class	**Predicted Negative Class**
Actual Positive Class	7	1
Actual Negative Class	2	6

Table 5.30. Results of bagging in a confusion matrix

Accuracy:= 0.8125 (Average value of accuracy for 10 executions:= 0.6)

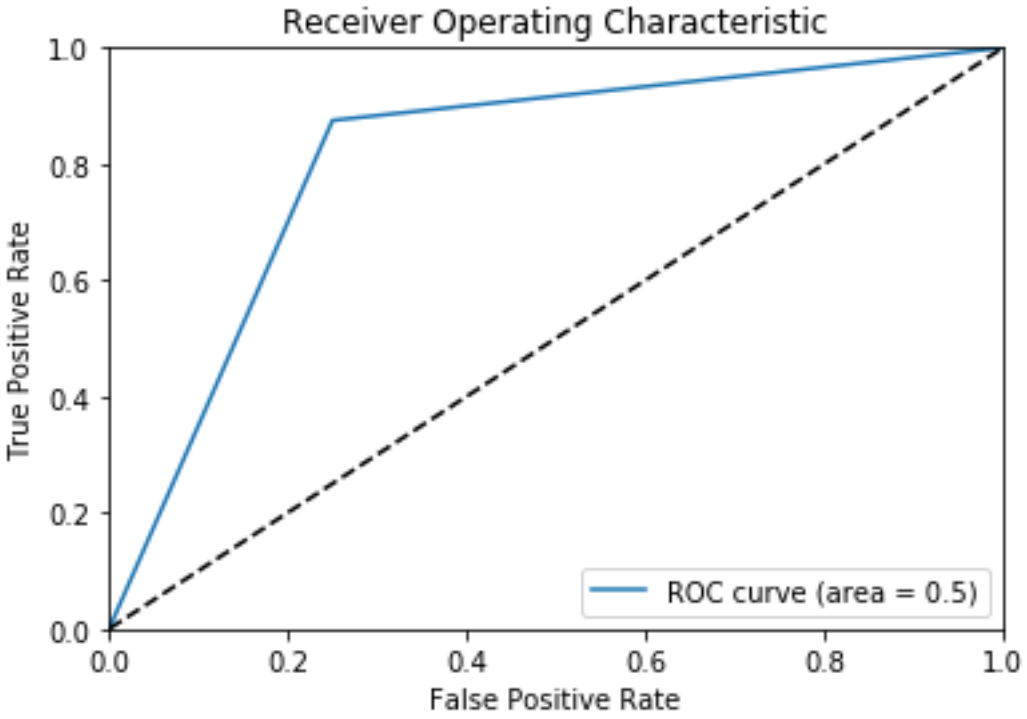
Positive Predicted Value:= 0.777777777778

Negative Predicted Value:= 0.857142857143

Sensitivity:= 0.875

Specificity:= 0.75

AUC: = 0.8125 (from Graph 5.27)



Graph 5.27. Receiver operating characteristic curve for bagging algorithm

AdaBoosting - Decision Tree as Base Learner:

In Table 5.31 the outputs of the adaboosting algorithm is summarised.

	Predicted Positive Class	**Predicted Negative Class**
Actual Positive Class	7	1
Actual Negative Class	3	5

Table 5.31. Results of adaboosting in a confusion matrix

Accuracy:= 0.75 (Average value of accuracy for 10 executions:= 0.6)

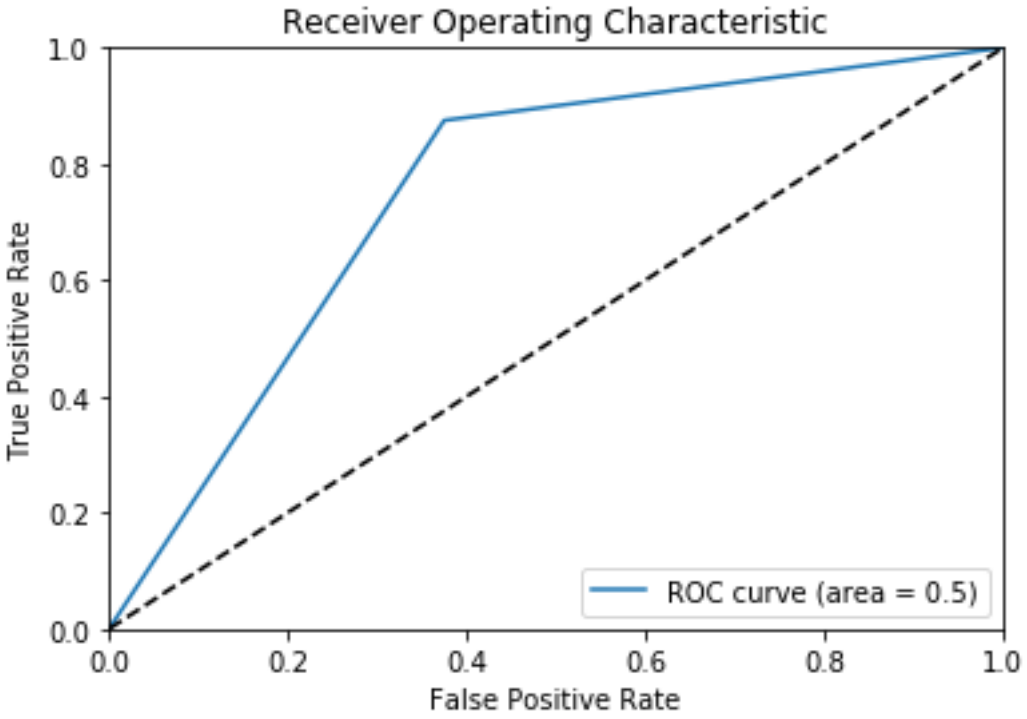
Positive Predicted Value:= 0.7

Negative Predicted Value:= 0.833333333333

Sensitivity:= 0.875

Specificity:= 0.625

AUC: = 0.75 (from Graph 5.28)



Graph 5.28. Receiver operating characteristic curve for adaboosting algorithm

Gradient Tree Boosting:

In Table 5.32 the outputs of the gradient tree boosting algorithm is summarised.

	Predicted Positive Class	**Predicted Negative Class**
Actual Positive Class	7	1
Actual Negative Class	3	5

Table 5.32. Results of gradient tree boosting in a confusion matrix

Accuracy:= 0.75 (Average value of accuracy for 10 executions:= 0.5072)

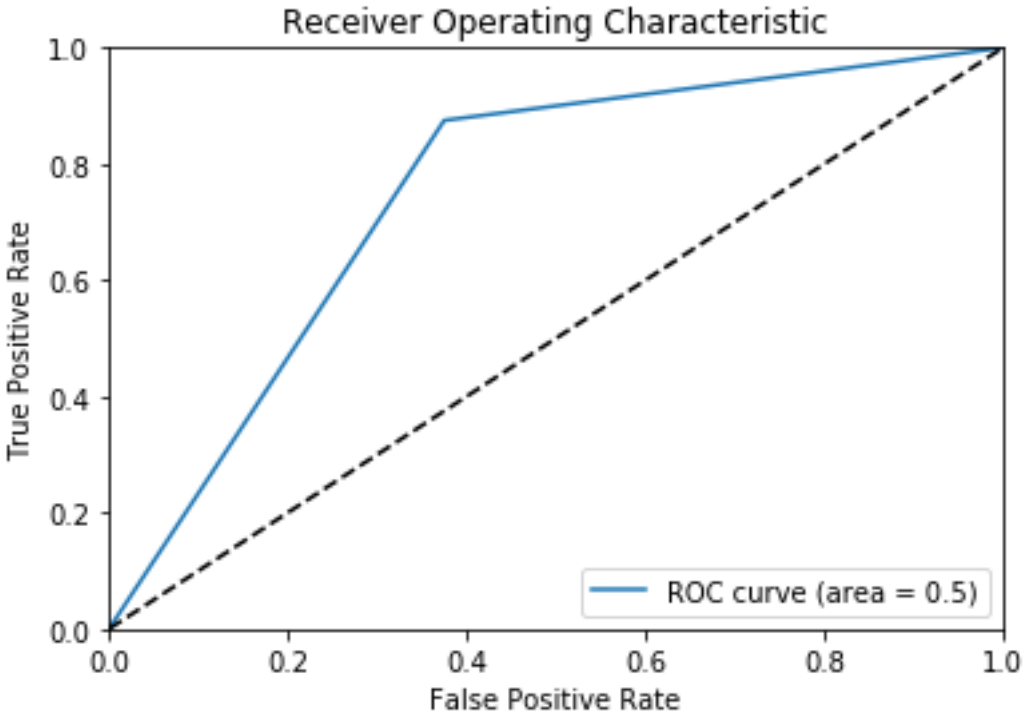
Positive Predicted Value:= 0.7

Negative Predicted Value:= 0.833333333333

Sensitivity:= 0.875

Specificity:= 0.625

AUC: = 0.75 (from Graph 5.29)



Graph 5.29. Receiver operating characteristic curve for gradient tree boosting algorithm

Random Forests:

In Table 5.33 the outputs of the random forest algorithm is summarised.

	Predicted Positive Class	**Predicted Negative Class**
Actual Positive Class	7	1
Actual Negative Class	1	7

Table 5.33. Results of random forest in a confusion matrix

Accuracy:= 0.875 (Average value of accuracy for 10 executions:= 0.625)

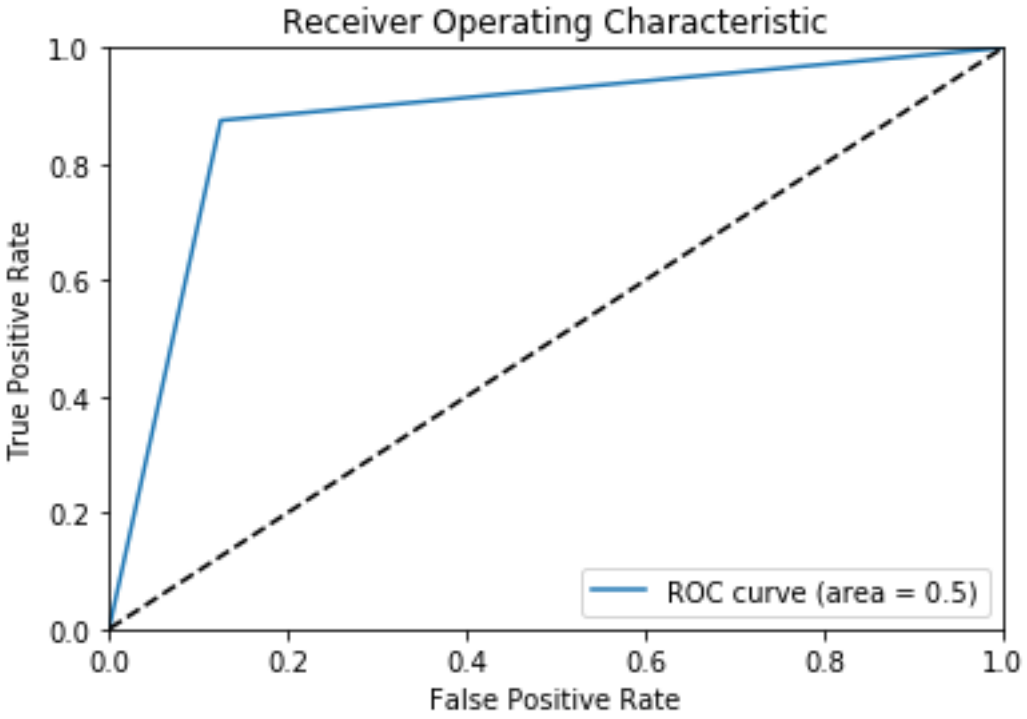
Positive Predicted Value:= 0.875

Negative Predicted Value:= 0.875

Sensitivity:= 0.875

Specificity:= 0.875

AUC: = 0.875 (from Graph 5.30)



Graph 5.30. Receiver operating characteristic curve for random forest algorithm

Evaluation of Classification Algorithms:

The algorithm that has the best performance on all six metrics is the *Random Forest* on the 10 executions, with 0.875 accuracy. It is able to correctly classify 14 out of the 16 data samples in the test set, with misclassified one each of the high and low risk classes. Its accuracy is significantly higher than that of the second ranking algorithm, bagging, with an accuracy of 0.8125. While the gap to the following two is even larger with an accuracy of 0.75. It's evidently that the *Naive Bayes* algorithm, even though has a significant lower accuracy of 0.6875, it can predict correctly all eight high classes.

It's noticeable that the algorithms with the highest results are all part of the ensemble algorithms being *Random Forest*, *Bagging*, *AdaBoosting* and *Gradient Tree Boosting*. Their higher results compared to the other algorithms can be placed on the fact that they can better handle a large number of features. Even though only one of the signals is maintained it still contains a total of 72 features, one for each time frame recorded. The ensemble algorithms all use a classification tree as their base learner, combining it with the recursive executions and sub sampling of features and data samples, they are more tolerable to irrelevant features.

From the examined algorithms the *Neural Network* (NN) has the lowest results with 0.625 accuracy. This can be resulted to the need of NNs for high tuning and parameter optimisation compared to the other algorithms. Also, there might be a case where the network is over trained and can memorise the training data, unable to generalise to the test data. Lastly, the network can benefit from multiple executions of each of the data distributions to training and testing sets. This is because of the randomised weights given to each neuron.

Ranking of Classification Algorithms in Descending Order as shown in Table 5.34:

Algorithm	Accuracy	PPV	NPV	Sensitivity	Specificity	AUC
Random Forest	0.875	0.875	0.875	0.875	0.875	0.875
Bagging	0.8125	0.7778	0.8571	0.875	0.75	0.8125
AdaBoosting	0.75	0.7	0.8334	0.875	0.625	0.75
Gradient Tree Boosting	0.75	0.7	0.8334	0.875	0.625	0.75
Naive Bayes	0.6875	0.6154	1.0	1.0	0.375	0.6875
K-Nearest Neighbours	0.6875	0.6667	0.7143	0.75	0.625	0.6875
Classification Tree	0.6875	0.6667	0.7143	0.75	0.625	0.6875
SVM	0.6875	0.6667	0.7143	0.75	0.625	0.6875
Logistic Regression	0.6875	0.6667	0.7143	0.75	0.625	0.6875
Neural Network	0.625	0.5833	0.75	0.875	0.375	0.625

Table 5.34. Ranking of classification algorithms in descending order for diagnosis of experiential avoidance for anxiety

The ordering has been done based on each algorithm's accuracy on the tested data set. For rating purposes when they have equal accuracy value then the rating of negative predicted value will be used followed by specificity, sensitivity and lastly the positive predicted value. For the rating of each algorithm, if they have the same values then the average accuracy of the 10 executions will also be taken into account.

For medical applications in machine learning it is more important to not wrongly classify a person as low risk when they are actually high risk, being the negative predicted value metric. In the other case, if someone is classified wrongly as high risk additional means can be taken to verify the classification before taking any measures , but will cause unnecessary concern to the person. Also, it's important to have accurate results for people who are actually low risk and have been predicted as low risk, obtaining the percentage from specificity. Usually in a large sample of medical data higher percentage will be classified as low rather than high risk. So this will limit by a

significant amount the total cases that will have to be validated individually by possibly more costly and time consuming methods [51].

5.4 Conclusion

In the first dataset the highest accuracy is achieved by the algorithms *K-Nearest Neighbours, Naive Bayes, Support Vector Machine* and *Logistic Regression* with a value of 0.8. For the seconds dataset the best performing algorithms seem to be *Support Vector Machine* and *Random Forest* with 0.9231. While for the last dataset the best results is achieved by only the *Random Forest* algorithm with 0.875.

From the data set for diagnosis of experiential avoidance in smokers it's noticeable that the algorithms that achieve higher accuracy are algorithms that can solve linear problems better. Hence we can conclude that there is a strong linear connection between the feature set and the class output. For the second data set, diagnosis of eating disorders, we also notice a linear relation between the input features and the output class, especially in regards to BI-AAQ results. For the BI-AAQ we can notice that a higher result in the questionnaire results to usually higher risk, meaning that there is a linear relation. However, the feature set contains higher noise/outliers, as for in the data set the BI-AAQ values might wrongly indicate the sample's classification. So the two algorithms that have better accuracy are algorithms that can handle outliers and noise in the dataset. Lastly, the third dataset, there is a much larger number of features with a total of 72 in comparison to 4 and 5 respectively to the previous datasets. The highest accuracy is achieved by the *Random Forest* algorithm because it is capable of handling a large number of features by ignoring irrelevant features and feature importance selection through the general method of constructing multiple classification trees and selecting each time a subset of features and samples. The least important features will be consistently located at a larger depth in the tree and so when the final tree is built they will not be included.

The results from the evaluation of the three datasets the algorithm that will be used for the implementation of the console will be *Random Forest* (RF), giving the best performance for two out of three datasets. The benefits of the RF are, as it was previously noted, that it can handle noise and outliers in the dataset, which is something to be expected especially taking physiological measurements from non-lab equipment.

Also, it can handle effectively a large number of features. Lastly, it can solve linear and non-linear classification problems. Even for a stronger linear relation between the input features and the output class, as in the first dataset, the RF algorithm still achieved a decent accuracy of 0.7.

Chapter 6

Biosignal Extraction from Wearable Devices

There is a need to explore a set of nonintrusive wearable devices that can obtain physiological measurements in such a way that they can monitor for a prolonged duration the wearer, while also not intervening in their daily activities. A total of 34 wearable devices will be categorised based on where it can be worn. For each one of the devices it will be listed their positive characteristics indicated by a '+' (plus) sign and negative indicated by a '-' (minus) sign, based on the general capabilities to access and extract the raw signal data and to develop applications that can communicate with the device.

From the listed devices the most suitable ones are found to be *MS Band 2* and *Muse* being able together to obtain data from the sensors PPG, GSR and ECG. For the two devices a single Android application is developed which will communicate with them, extract the signal data, store it and will be used as a client to send to the local server the stored data.

6.1 Wearable Devices

The initial market research has been done in this thesis to determine which of the available wearable devices can be used for retrieving data collected from their sensors so they can be sequentially analysed. It's required to take measurements from the sensors PPG or ECG, GSR and EEG from one or more wearable devices. These measurements will be taken throughout the day and not at an especially equipped lab. This makes it so that it's important for these devices to be as little as possible intrusive in the daily life of the wearer. The following listed wearable devices have been found which meet the nonintrusive requirement and have at least one of the stated sensors, based on the market availability as of September 2016:

6.1.1 Wearables Worn on the Wrist

Contains devices that are either bracelets or smart watches that are worn on the wrist:

- E4 Wristband: https://www.empatica.com/e4-wristband
  Available sensors: PPG and GSR
  + The device can store raw sensor data on a locally available storage, while also having the ability to store the data on their cloud platform. The locally stored data will be synchronised when there is a connection to the internet, directly or through a smartphone, so there is no need for it to be continuously paired. The access of the data stored on their cloud platform can be achieved through their online web portal. In addition, the data from the device can be viewed in real time through their Android and iOS applications. Also, they provide the ability to develop applications on Windows, Android and iOS devices through their available API, obtaining direct access to the raw data stream from the sensors.
  – This devices is designed for medical and research purposes and so offers a high accuracy of the reading from the sensors. This comes with a higher cost compared to other wearable devices, especially when multiple devices might be purchased to monitor multiple people simultaneously.

- Embrace Watch: https://www.empatica.com/product-embrace

  Available sensor: GSR

  + There is an available Android and iOS application for viewing analysed data from the device. The data are stored and synchronised with their cloud based platform.
  – They do not allow exporting or accessing in real time the sensor data, raw or analysed.

- BLOCKS: https://www.chooseblocks.com

  Available sensor: PPG

  + The device is developed based on open hardware and software allowing for easier and faster improvements from the community. In addition to the openness of the software, they will offer in the future an API for direct communication with the device and handling and storing the data from the sensor. There are also plans to add additional sensors to the device, which can be easily added to an already purchased device because of its modular design.
  – Currently there is not estimated date for when the API will be available, Additionally, the device will be available for ordering in December, with possibly delays as for it is still in development phase.

- Microsoft Band 2: https://www.microsoft.com/microsoft-band/en-gb

  Available sensors: PPG and GSR

  + There are available applications for Windows, Android and iOS for viewing basic and processed data from the device. The data obtained from their application is also stored and synchronised with their cloud platform *Microsoft Health*, which can also be used for viewing the data. They also offer an API for the platforms Windows, Android and iOS which allow they development of applications that communicate with the device and have direct access to the sensor data so they can be handled and stored. For Windows, they additionally have an SDK for easier development

  - The device doesn't have any available local storage for the sensor data and so needs to be always paired with a smartphone or connected to a pc. On their platform they only store processed data and not the actual raw stream from the sensors.

- Angel Sensor M1: http://angelsensor.com
  Available sensor: PPG
  + They offer an API and SDK for Android and iOS for developing applications that can have direct access to stream data from the sensor. They have an Android and iOS application for viewing data from the device.
  - There are currently no available stock of the device or they have discontinued its sale. There is no local storage on the device and so requires continuous pairing with a smartphone.

- Jawbone UP3: https://jawbone.com
  Available sensors: Bio impedance and GSR
  + The data from the device are stored on their own cloud based platform through pairing on a Android or iOS device and having their application. They also offer local storage of processed data from the sensors so it can go without continuously being paired with a smartphone. The data will be synchronised the next time it will be connected with their application. Also, they offer a web API for RESTful services, which allow access to processed data stored on their cloud. In addition, they have SDK for developing applications on Android and iOS.
  - There is currently no available access to the raw sensor data through their API, while there are plans to make them available at an unspecified time in the future.

- Basis Peak: https://www.mybasis.com
  Available sensors: PPG and GSR

+ The processed data from the sensors can be saved locally on the device or on their cloud platform. It can also be integrated with the health platforms *Apple Health* and *Google Fit.*
– There is no available API to directly access the stream data from the sensors or any other way to export the raw data. Also, it has been temporary withdrawn from the market due to cases of overheating of the device.

- Amiigo: https://amiigo.com
  Available sensor: PPG
  + They have available Android and iOS applications for viewing the processed data from the device.
  – It's planned to offer an API for direct access to the sensors' data and at an undefined time in the future they will have an SDK for easier application development. Also, there are no available devices for purchase and there is a waiting list for pre ordering.

- Moto 360 2: http://www.motorola.co.uk/products/moto-360
  Available sensor: PPG
  + It's compatible with most fitness and health applications on Android and iOS smartphone devices for viewing data. Also, there is available API through the Android wear for direct access to the data from the sensor and implementing Android applications. The device has also the ability to store data locally, so there is no need for continuous pairing with a smartphone.
  – The device can only be connected with an Android device for purposes of extracting basic data from the wearable's sensor through the development of a 3rd party application.

- Samsung Simband: https://www.simband.io
  Available sensors: PPG, ECG and GSR
  + They have available an API for the *Tizen* OS, which gives the capability to get direct access to the data stream from the sensors and also process

the data directly on the wearable. They have plans to offer in the near future a complete SDK for easier developing on the device itself. The data can be either stored locally on the device or they can be stored on the *Artik Cloud* platform. The export of the data from their cloud platform can be achieved through the usage of REST and WebSockets protocols.

  - Because the device is currently at a development phase it is not freely available for purchase from the public. To obtain a device it can only be done through the submission of a request application and awaiting approval of usage. Also, the only way to extract raw or processed data from the device is through their cloud platform and can't be achieved through smartphone pairing. The *Artik Cloud* platform is a paid service based on the number of messages received from the server, while the free plan is very limiting to the amount of messages received and is used only for testing purposes.

- Samsung Gear Fit 2: http://www.samsung.com/global/galaxy/gear-fit2/
  Available sensor: PPG
  + There is available API for the *Tizen* platform and also an SDK for developing applications on the device. They data from the device can be viewed through applications on Android and iOS.
  - It doesn't allow the extracting of the data from the device so they can be stored and processed.

- Fitbit Surge: https://www.fitbit.com/eu/surge
  Available sensor: PPG
  + It can be paired with Android, iOS and Window devices to view processed data from the device's sensor. The data can be stored locally until it can synchronised again with the paired smartphone. The data are also synchronised with their cloud platform which can be also viewed from. Also, the processed data can be extracted through their application or their web platform.

  - They do not allow exporting the raw data stream from the sensor, but only the processed data.

- Biovotion: http://www.biovotion.com

  Available sensor: PPG

  + The offer an online platform for viewing the data from the device.
  - There is no way to extract the data from their device or platform so they can be processed and stored. Also, the device is still currently in development phase.

- SiDLY Care: http://www.sidly-care.eu/en/

  Available sensor: PPG

  + They offer a web portal and Android and iOS applications for viewing the data from the device.
  - They do not offer a way to extract the data from the device either from their application or their web portal.

- Ki Fit: http://www.kiperformance.co.uk

  Available sensor: GSR

  + They offer Android and iOS applications and a web platform for viewing the data from the device. The device can also store locally data collected from the sensors until it can be synchronised with their cloud based database.
  - There is no capability to extra the data recorded from the sensors either from the web platform or the application. Also, it's requires a monthly subscription for using and accessing their platform.

- HELO: https://www.worldgn.com/wear/

  Available sensor: ECG

  + They have available Android and iOS applications which are used for storing and viewing the data from the device.

– There is no ability to directly access the raw data from the sensors so they can be processed. Also, it requires to be continuously paired with a smartphone so it can show the data.

6.1.2 Patch Wearables

Contains devices that are a sticky path that is placed directly on the skin without the need of any straps, usually around the chest or leg area:

- BodyGuardian Heart:
  http://www.preventicesolutions.com/services/body-guardian-heart.html
  Available sensor: ECG
  + They allow to store data from the device directly to their online platform and so doesn't require intermediate devices. Through the portal they allow viewing of the data stored and generate different health reports based on resent recordings using *PatientView* and *PatientFlow* online applications respectively.
  – They do not offer an API or any other way to directly access the raw data from the sensor, so that they can be processed and stored independently from their platform.

- HealthPatch MD: http://www.vitalconnect.com/healthpatch-md
  Available sensor: ECG
  + They allow the data obtained from the sensors to be stored on their cloud platform. Also, they have available a platform *VitalConnect* which allows the integration of services and applications from 3rd party developers for processing the data.
  – The device is only available through a few partners which usually they target doctors and hospitals.

- VitalPatch: http://www.vitalconnect.com/vitalpatch
  Available sensor: ECG
  + They allow the data obtained from the sensors to be stored on their cloud platform. Also, they have available a platform *VitalConnect* which

allows the integration of services and applications from 3rd party developers for processing the data.

- The device is only available through a few partners which usually they target doctors and hospitals.

6.1.3 Wearables Strapped on Chest

Contains devices that are strapped and tightly secured around the chest and/or shoulders so the sensor comes into direct contact with the skin on the chest:

- BioHarness 3: https://www.zephyranywhere.com/products/bioharness-3
  Available sensor: ECG
  + They have available an API for directly accessing the data stream of the sensor. They support the development of applications for Android and Windows, using their API.
  - The device doesn't have the capability to locally store data and so will require to be continuously paired with an additional device or pc.

- EcgMove 3:
  http://www.movisens.com/en/products/ecg-and-activity-sensor-ecgmove-3/
  Available sensor: ECG
  + They offer an API for windows which allows the device to connect via USB to a PC and also an API for Android connection via Bluetooth. Both of the APIs allow accessing and storing the raw data from the sensor. Also, the device can store locally data from the sensor until it is connected with a PC or Android device.
  - The device is targeted towards studies and research purposes and so will require to submit a request form. For student purposes, they offer the device for a maximum of 6 weeks.

- QardioCore: https://www.getqardio.com
  Available sensors: ECG and GSR

+ They offer the ability to view data from the device on Android and iOS through their application. The data can be stored locally until it can be paired with a smartphone in which it will then synchronise the data.
– They do not give the possibility to directly access the raw data from the sensors so they can be processed and stored, but only already processed data. Also, the device is currently not available for purchase.

- H7 Heart Rate Sensor:
  https://www.polar.com/ca-en/products/accessories/H7_heart_rate_sensor
  Available sensor: ECG
  + It is compatible with most fitness apps which allows the viewing of the results. From their own fitness app, they allow the transfer of the data to their website which then can be download as TCX, GPX or CSV format. Also, any iOS, Android and Windows smartphone application can be developed which handles Bluetooth connection to receive heart rate variance information.
  – The raw ECG data is not available but only processed as HRV value. Also, to export data from the native app requires the purchase of the Polar data transfer accessory and an additional app. While 3rd applications can be developed, it does not offer any SDK or API for easier application integration with the device.

6.1.4 Wearables Integrated in Clothes

Contains devices that are integrated and are part of a complete set of clothing wear, either shirt, trousers or both. The clothing have no visible indication that they contain any sensors within. The clothes are tightly worn so that the sensors in the clothing can come to direct contact with the skin:

- Hexoskins: http://www.hexoskin.com/
  Available sensor: ECG
  + They offer an Android and iOS application for viewing and storing the data and also an online dashboard which can be synced with. The wearable also has local storage which allows it to record without a

smartphone. The stored data can be then transferred through a USB to a pc and then update the data on the dashboard. The raw and processed data can be exported from the dashboard. In addition, the data can be exported through the available API either from RESTful webservice or as JSON format.

- – The data exported is in binary format and requires processing to convert to human readable, which can be done by passing the data to a suggested converter. Also, they do not offer any SDK or API so the wearable can be directly integrated and connected with developed applications.

- Athos: https://www.liveathos.com/

  Available sensor: ECG

  + Offers iOS application which can connect with the wearable to view and store processed data.
  - – Raw data from the sensor are not available and they do not offer a way to extract any data from the device.

- OMShirt: https://omsignal.com/

  Available sensor: ECG

  + Offers iOS application which can connect with the wearable to view and store processed data.
  - – Raw data from the sensor are not available and they do not offer a way to extract any data from the device. Currently the wearable is unavailable for men.

- hWear: http://www.personal-healthwatch.com/

  Available sensor: ECG

  + Offers an Android application and cloud platform to send and view real time data. The wearable can store locally data obtained until it is paired with a smartphone so it can synchronise the data.
  - They don't offer a way to extract data from their cloud platform. They also don't have available an API to develop and integrate with the wearable 3rd party applications so the data can be handled and stored.

6.1.5 Headset Wearables

Contains devices that are worn on the head that can be sensors integrated in either a headband, a hat or earphones:

- Muse: http://www.choosemuse.com/

  Available sensor: EEG

  + They have available Android and iOS applications for viewing real time data. Also, they support an API for the two platforms which allows obtaining and handling the stream of raw data. In addition, they offer a set of research tools which allow the device to be connected with a pc either Windows, Mac or Linux OS via usb and transfer the data stream as OSC messages.
  - The device can't locally store data and must be continuously paired with a smartphone or connected to a pc via usb.

- EMOTIV Insight: http://emotiv.com/

  Available sensor: EEG

  + They have available Android and iOS applications for viewing real time data. Through the usage of the premium SDK they allow the development of applications for Windows, Mac, Linux, Android, iOS, Arduino and Raspberry Pi that can directly access the raw data. In addition, they have a software for pc which allows viewing and recording of real time data streams.
  – To access the raw data through their software a monthly subscription is required which also imposes a quota on the recordings. Also, the premium SDK is not freely available and the use of it must be approved by the company first.

- Neurosky BrainLink Pro: http://neurosky.com/

  Available sensor: EEG

  + They offer SDK for Android, iOS, Mac and Windows which can access the stream of raw data from the device. They also offer research tools

which allows to easily view and record raw data and export the stored data as CSV file.

  - The research tools are not included and have an additional cost. Only available to regions that have 60Hz electrical frequency.

- Sport Pulse: http://www.jabra.com/sports-headphones/jabra-sport-pulse-wireless
  Available sensor: PPG
  + Available applications for viewing data for Android and iOS. Also they offer a SDK platform for Windows, Linux and Mac to develop applications.
  - Through the available API, currently it's not possible to access raw sensor data or processed heart rate measurements. In addition, there is no support for developing Android or iOS applications. The device will require continuous pairing with a smartphone and as a headphone it must be always in-ear.

- SMS Audio BioSport: https://smsaudio.com/
  Available sensor: PPG
  + Available applications for viewing data for Android and iOS. Also they offer a SDK for Android and iOS allowing to obtain the calculated heart rate per minute so it can be handled and stored.
  - The device will require continuous pairing with a smartphone and as a headphone it must be always in-ear

- lifeBeam Smart Hat: http://lifebeam.com/
  Available sensor: PPG
  + It's compatible with most fitness apps for Android and iOS so it can display the heart rate.
  - Doesn't support any way to develop applications for the device or to extract the raw data of the sensor or the processed heart rate value.

- Spree Smartcap: http://spreewearables.com/
  Available sensor: PPG

+ They offer an app for Android and iOS to view heart rate values and also is compatible with most fitness apps for Android and iOS.
− Doesn't support any way to develop applications for the device and extract the raw data of the sensor or the processed heart rate value.

6.2 Implementation

From the current devices available the Microsoft Band 2 and Muse have been selected to be used to obtain data from the sensors photoplethysmography, galvanic skin response and electroencephalography, offering together all the data required from the sensors. These devices have been selected because of their relative low cost and no additional ongoing costs from subscriptions, allowing for a much larger purchase of devices to have more people simultaneously monitored. Also, they both allow easily to extract and handle sensor data through an API for at least mobile devices Android and iOS. In addition, the devices are independent from intermediate cloud based platforms to obtain the data offered by the companies, which allow for a higher longevity of the devices in case of discontinuation. However, their main limitation is that they do not offer any local storage of the sensors' data for when developing applications. Thus they will be continuously depended on an external device to communicate with and transfer data.

6.2.1 Technical Information

The communication between the two devices will be made through an Android device so all technical information, which will be presented, will be focused as such.

Microsoft Band 2:
The supported Android version for their API is version 17 (Android 4.2) and later. It’s required for the device to connect with third party Android applications and retrieve sensor data to have their *Microsoft Health App* also installed on the smartphone.

Sensor Data:

The information from the electroencephalography is not directly available but they offer the processed form of the heart rate and RR interval. The data retrieved from each sensor is summarised in Table 6.1.

Data	**Retrieval Frequency**	**Return Type**	**Comments**
Heart Rate	1Hz	Number of heart beats per minute (Integer)	The heart rate available from the device is optimised for when the user is in "resting state" (low activity). All other states are not available to extract
RR interval	When changed	Seconds between last two heartbeat intervals (Double)	The RR interval available from the device is optimised for when the user is in "resting state" (low activity). All other states are not available to extract
Galvanic Skin Response	0.2/5Hz	Skin resistance in kohms (Integer)	-

Table 6.1. Summary of the return data frequency and type from each sensor for *MS Band 2*

Muse:

The supported Android version for their API is version 19 (Android 4.4) and later.

Sensor Data:

The data retrieved from the sensor is summarised in Table 6.2.

Data	Retrieval Frequency	Return Type	Comments
Electroencephalogram	256Hz	Returns the channel's EEG measurement in micro volts (Double)	The EEG signal is received from 4 different channels

Table 6.2. Summary of the return data frequency and type from the sensor for *Muse*

6.2.2 Android Application Design

The Android application to communicate with the devices will have a minimum SDK version of 19 derived from the Muse library. The libraries used for the *MS Band 2* and *Muse* devices are microsoft-band-1.3.20307.2 [66] and libmuse-android-5.13.0 [67] respectively.

The application allows the user to select which of the devices they want to connect to and also which of the sensors to activate to receive stream data from. The wearables must be first paired with the smartphone before they can be used. For discovering and connecting to the devices through the application it is required for the smartphone to have enabled Bluetooth connection and for *Muse* additionally will require access to approximate location for using the Bluetooth low energy library. The attempt to connect to the wearable devices will last for 2 minutes before it will timeout, which is sufficient time for both devices. Also, when selecting stream data for HR and/or RR interval the user must give consent for the application to be able to receive and handle the data. The main view of the Android application can be seen in Image 6.1.

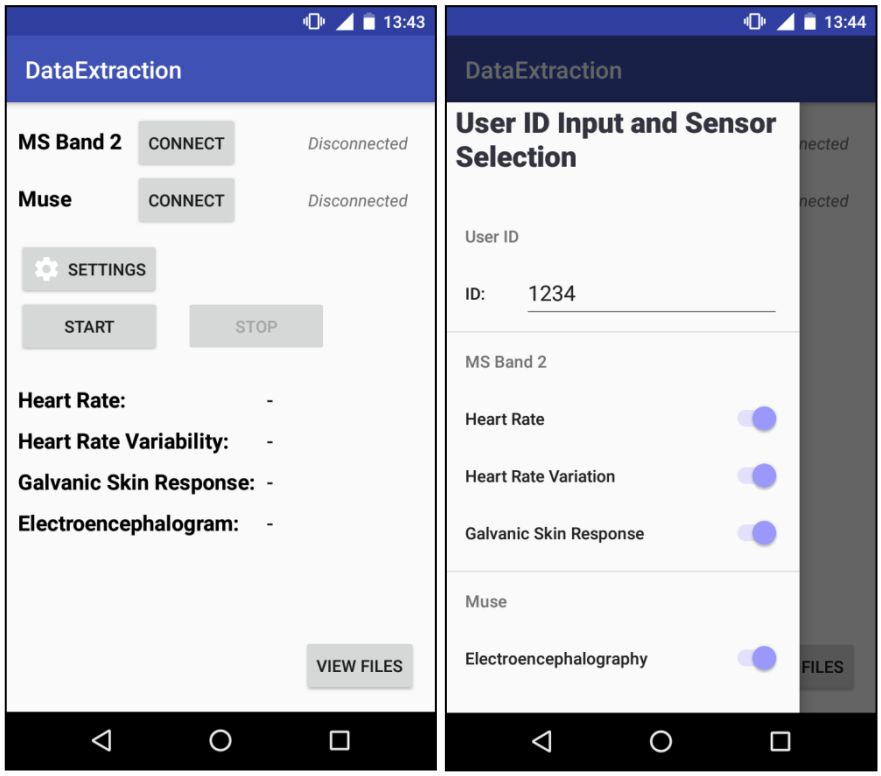


Image 6.1. The main view of the Android application (left) and the ID input and signal selection in the navigation bar (right)

From the available data streams from the devices only the *MS Band 2* for the GSR sensor offers two retrieval frequencies of 0.2 and 5 Hz. The 0.2 frequency has been selected so the battery consumption of the wearable can be reduced, because it will send less frequently messages to the smartphone. Also, it reduces the power consumption of the smartphone as for it will receive less data through Bluetooth. This is a compromise of the accuracy of the GSR readings in favour of prolonged monitoring of the patient.

The data received from the wearable devices is stored on the external SD of the Android smartphone as a CSV comma separated file. The storage location can be either on a physically dedicated external SD card or on the dedicated external SD located on the internal storage, which is allocated by the manufacture. The name of the file is the ID value passed and stored on the application appended with the date and time of which the recording started. The format of the date and time is year, month, day, hours and minutes (yyyyMMddhhmm) so they can be easily sorted just by name through the system file explorer from oldest to newest for a specific ID.

For each data stream selected the application will temporary store only the last values received in main memory as the same type they were sent (HR, GSR: Integer and RR, EEG: Double). Because from the EEG signal 4 values are received only the mean value of them is stored. Each temporary variable with its current value is then periodically stored in the output file together with the specific time stamp of when the data were stored. Because of the requirement for prolonged monitoring, the periodic time to store data to the output file has been selected to be 2 seconds. This is a trade off of the accuracy of the sensors because a lot of the data received will be discarded, highly effecting the EEG and less significantly the GSR signal, so the problem of limited free storage space on the smartphones can be minimised.

The output files aren't exclusively private to the application so they can be transferred to a PC through a USB cable or uploaded to a cloud platform. This procedure can be tedious as for the user must enable file transfer when connecting to the PC or transfer from the cloud platform to the PC and must know the directory stored, which may vary from manufacturer to manufacturer. To overcome this a server(PC) - client(Android application) has been developed to simplify the process and allow the transfer of files within a local network.

The server is implemented as a local server and will not have a static IP address and so it's split to 2 servers a UPD broadcaster and TCP file server. The UDP broadcaster sends to the local network on a specific port its IP address which will be received from the client to establish a TCP connection with the server. When a TCP connection is created the client sends first the file's name ending with a null value then followed by the data which are then stored locally on the server. The server will wait to receive the file data until the client closes the connection.

Chapter 7

Console Implementation

It is important to have a central and complete application that can obtain the data from the wearable devices, through the Android application, and to be able to classify them to a category of high risk (1) or low risk (0). All these features will be handled automatically through the development of an application run on a computer, in an intuitive and simple way for the user.

First, there needs to be an easy way to transfer the data to a local computer, as for one of the uses of the Android device is as an intermediated file storage for the exported data from the wearable devices. This will be achieved by using a local client-server model with the server being implemented on the application for a personal computer. Both the Android smartphone and computer must be connected to the same local network. After the data is received, as for they are a continuous data stream, they must be summarised to a set of characteristics using some statistical feature extraction methods. Finally, the processed data samples can be used to make class predictions on an already trained classification model.

7.1 General Design

The implementation of an application console with a user interface is necessary to easily handle the data from the wearable devices stored on the Android device using the previously mentioned Android application. The features and functionality implemented through this console are:

- An intuitive and simple user interface to handle all features.
- Handle the local server which is responsible for the file transfer from the Android device to the local computer.
- Structurally store received and processed data.
- Extract features from the raw data received from the wearable devices.
- The processed data (extracted features) to be used for either predicting its classification or allow to state its actual value/class by the user so it can be used for training purposes of the model.
- Uses a classification algorithm (*Random Forest*) to predict the classification label of the sample. Once the algorithm is trained it will store the model so it can be loaded on next start up.

7.2 Architectural Design

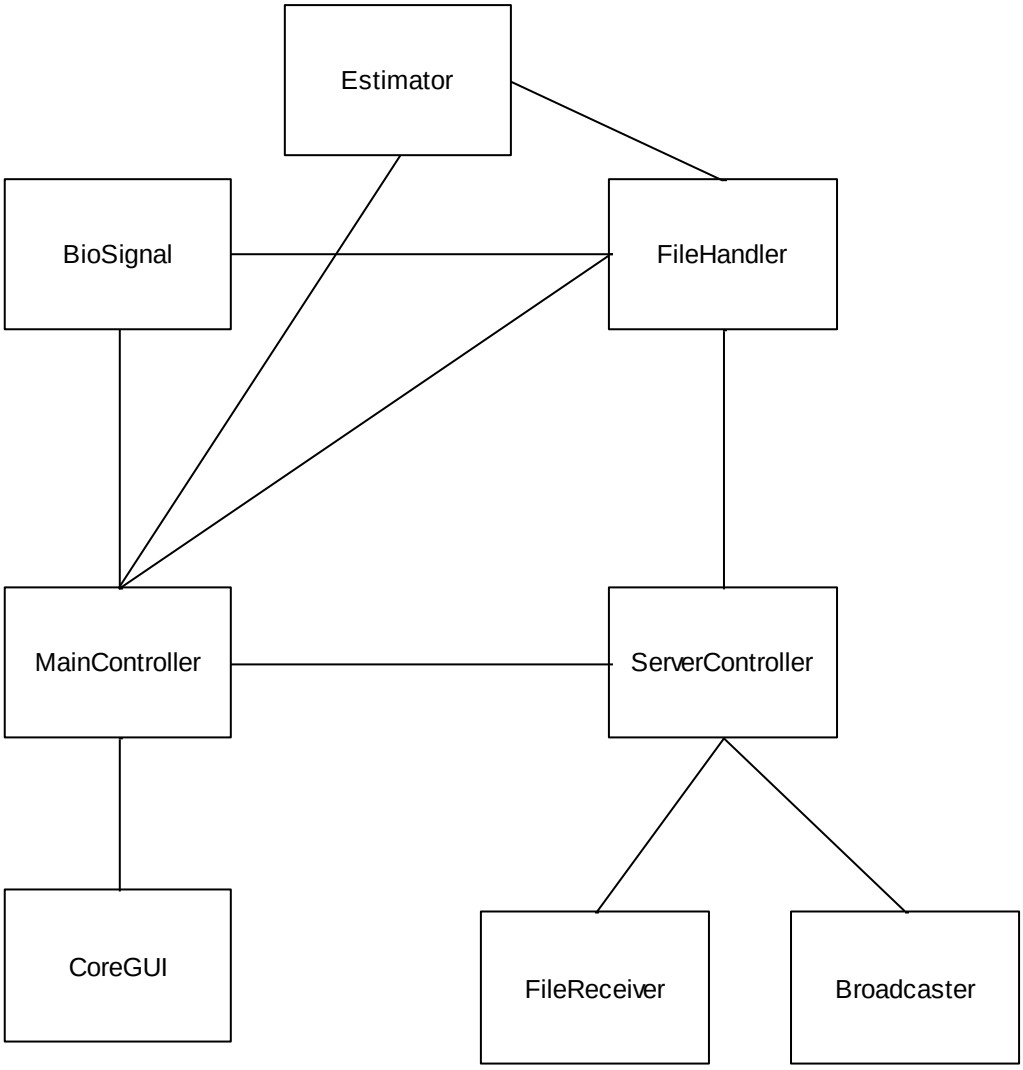
The console has been designed based on the principles of the *Model View Controller* (MVC) architecture. As the name says, the MVC architecture splits the design into three main parts, the view, model and controller. The view displays the user interface and handles user events and model representation, the model implements the underlying logic of the application and the controller handles the communication between the model and the view. The main advantages of the MVC model, other than the fact that it allows to clearly separate the code and logic of the application, it offers an easier way to expanded and improve one part of the application without affecting the others, as for it has low coupling. This also leads to it being highly maintainable and also promotes code usability [28].

The MVC model is used to offer a clear separation of each part of the implementation for better readability, maintenance and future improvements. As the

initial version of the program and also that there isn't yet sufficient data collected from the wearable devices it is expected to have changes in the program's model logic. When enough data is available then the machine learning algorithm that has the best results from their analysis can be used with fine tuned parameters and using the extracted features that bring the best results which will be able to be identified. All these changes can be done without requiring any changes to the view of the program.

In addition to the short term changes, using an MVC model accommodates easily any possible future changes in requirements and features. The console can be expanded to solve multiple machine learning problems, having the benefit of reusing both components from the view and model. The design allows to train, predict and store multiple learning models and to use different methods to extract features from the raw data. All these changes can be followed with minor changes to the user interface so that it will allow the user to select which problem they want to handle, while reusing a large portion of the already existing code. Also, the UI can be changed just for aesthetic and usability improvements without interfering with the underlying model.

The console is designed using two controllers, a controllers handing the local server and the main controller which will handle the communication between the model and the view, while both can communicate between them. The model contains the main classes 'Estimator', 'FileHandler', 'Biosignal', 'Broadcaster' and 'FileReceiver'. While the view contains 'CoreGUI' and other supplementary classes. The interconnection of these classes can be seen in the class diagram in Graph 7.1.



Graph 7.1. Class diagram of the console showning the main classes and their connections

7.3 Main Controller

The main controller of the console is the central coordinator for its logic and functionality. The controller will handle retrieving data from the model for the view and also send commands to the model to processes data. Also, it will handle all communications with the user and requests through the view model notifying the appropriate model functions. In addition it will communicate with the secondary controller, the server controller, so it can indicate when to start or stop the local servers and to be notified when a file is received.

The controller will separate any intense functions of the model from the main thread and will execute them in the background from a thread pool which will be created once and reused. Currently, only one thread will be available in the thread pool as for the console handles only one machine learning problem and to avoid possible conflicts, such as trying to predict on the model before its training is completed. Because the graphical user interface must be thread safe and avoid race conditions when notifying the user, the parallelised model's results will be serialised before updating the UI. A thread safe queue is used for this purpose where once a component from the model has finished it's execution the controller will store its output and the appropriate command from the view. The queue will only allow one thread to access it at a time placing lock and blocking any other attempts until it's released again from unlocking. A separate thread is used to periodically check the queue and remove and execute from it any UI commands.

7.4 Servers

The console will manage two different local servers which will be used to transfer the stored CSV file from the Android device. The first server will be a UDP IP broadcaster and the second one will be a TCP file transfer. The servers will start automatically once the console is started without any interaction from the user. The servers will continue running throughout the console's process life spam and will terminate once it is closed.

The UDP IP broadcaster is required before enabling the Android device to connect to the local server. This is because the server is a local server and so it does not have a static IP address (the console can be run on different machines or the same machine may get a different IP if not set to a static one) as for it is not executed on a dedicated server and there is no domain name server to obtain the host from the domain name. So, it's important for the Android device to discover the local server's IP address whenever attempting to transfer files. Therefore the console/servers and the Android device must be connected to the same local network.

The IP broadcaster runs on its own thread preventing any interrupts to the user interface and other functionalities. When the server starts it will initially obtain the local

machine's IP address and store it as a null terminated string maintaining also the decimal points. This string is sent to the UPD broadcast socket with the port number 8200. The IP address will be transmitted every 4 seconds so it can compensate for the possibility of data loss through the UDP method, will not be aware when a device will be trying to connect and to not create any unnecessary network flooding. The transmission frequency is sufficient enough to not cause any unwanted delay when connecting to the server, as for the most time consuming will be the actual file transfer.

The TCP file transfer server will be used to receive the file from the Android device. The device can establish a connection once it has received the IP address from the broadcaster. The TCP server, as the broadcaster, runs on its own thread to prevent blocking of the user interface or other functionalities. The TCP socket created will be used only for receiving information, as for the server will not send any messages back to the device, and will be bound to the port 8150. The client successfully sending a file can be seen in Image 7.1 and the server successfully receiving and storing the file can be seen in Image 7.2.

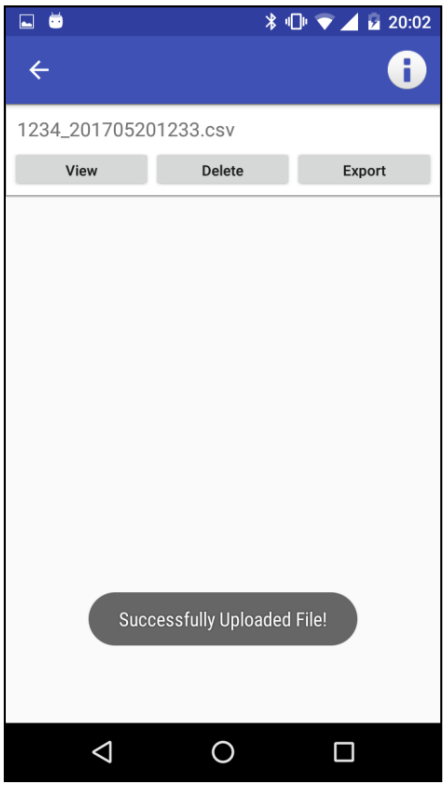


Image 7.2. TCP successful file transferred from the client (Android application) to the server

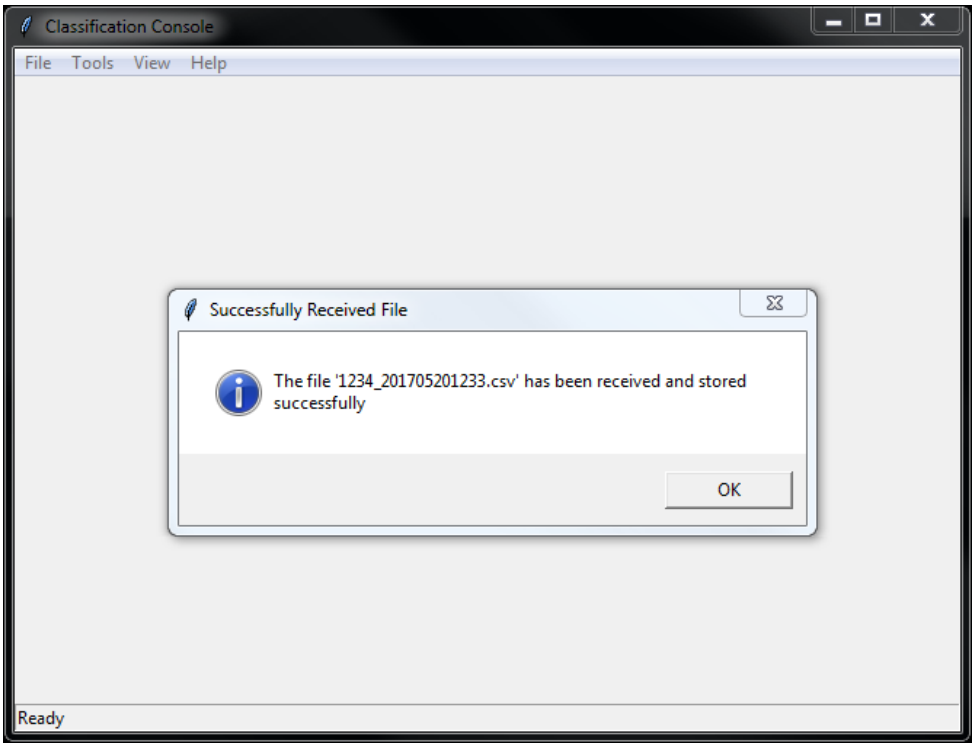


Image 7.3. TCP successful file received at the server and stored locally

The server's execution is blocked in the listener state awaiting for connections before it can continue. It can establish a maximum of five connections at any given time to receive and handle files. Any other attempts to connect to the server when the maximum number of connections have been reached they will be refused. They way the Android application is implemented one device can only transfer one file at a time and so to reach the maximum connections five different devices must be attempting to transfer files. This number should be more than sufficient to cover the needs of the local server.

When a TCP connection is established the server will start reading the data passed until the client has closed the connection and so will assume that there are no more data to read from. The data format excepted from the client will be a utf-8 byte sequence with the first data representing the name of the file, so it can store it with the same name, followed by the file's data. The server will start by reading one by one each byte and will convert it to a character an append it to as the file name. It will continue to do so until it reaches the first null byte character and so will indicate that the full file name has been read. Using the received file name the server creates a local file with that

name, replacing a possible old file with the same name. The server reads in bulk a total of 32KB in its buffer at a time and writes it to the local file to reduce IO delays because of the large file size that will be transferred.

7.5 Feature Extraction

The raw data from the wearable devices is obtained as a time series of values which for each sample the time frame can be different. For classifying a time series data sample either one of the two following approaches can be taken, to either transform the data to static or extract statistical features from it. Because not all algorithms can use a static window frame of data for classification and storing extracted features from the raw data require much less storage space, the second option will be followed.

For each of the stored data, heart rate (HR), heart rate variability (RR), galvanic skin response (GSR) and electroencephalogram (EEG), from the wearable devices a total of seven features are extracted. However, the time stamp of the vector of values will not be used as a part of the selected features. The statistical features used to describe each stored data is maximum, minimum, mean, standard deviation, variance, kurtosis and skewness values. Each of the extracted features from each raw data passed will be used as the features of the classification algorithm.

For each stored data, n is the total samples, x_i the value in the i-th position with $1 \leq i \leq n$:

Maximum value ($\max(x_i)$): The maximum value found in the data.

Minimum value ($\min(x_i)$): The minimum value found in the data.

Mean value ($\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$): The sum of all the values is the data divided by the total number of samples. It expresses the average value of the data set.

Standard deviation value ($\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$): It is a measure that shows the variation of the data sample from the mean value. A smaller value indicates a small variation of the data while a larger values show more disparity of the data.

Variance value ($\sigma^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$): It is the squared value of the standard deviation [54].

Kurtosis value ($a_4 = \sum_{i=1}^{n}\frac{(x_i-\bar{x})^4}{n\sigma^4}$): It measures the tail heaviness of peak points on the data set. Kurtosis is the fourth standardized central moment for the probability model.

Skewness value ($a_3 = \sum_{i=1}^{n}\frac{(x_i-\bar{x})^3}{n\sigma^3}$): It is a metric to define the symmetry of the data between them. With a value of 0 the data is perfectly symmetric while a larger value shows how much asymmetric it is. Skewness is the third standardized central moment for the probability model [32].

7.6 Classification Algorithm

The classification algorithm selected and used is the *Random Forest Classification* (RFC) as implemented in the sklearn 0.18.1 library with default parameters [68]. The exception in the parameters are the total estimators created changed to 1000 and to run parallelly with the maximum number of available cores. The specific algorithm was selected because when implementing the console there were no available training data to analyse and compare any of the classification algorithms. Also, because of the lack of data there cannot be done any analysis on which of the extracted features contribute to a higher accuracy result and so all features will be used with a total of 28 (7 extracted features from the data received from the wearable devices HR, RR, GSR and EEG). The RFC can give good results without much parameter tuning and can easily handle a larger number of features compared to other algorithms and so can handle effectively a future tasked classification problem.

The algorithm can be trained using a data sample with the appropriate features extracted and having a binary output classification value of either high risk (represented with the value 1) or low risk (represented with the value 0). The training data should ideally come from the raw data obtained from the wearable devices and passed to the console. Having the raw data they should be then processed to extract the required features. Lastly, with the processed data sample it is required to append the actual output class of the sample so it can be used for training. All the data samples that are

ready to be used for training will be concentrated into a single file so it can be easily read and passed to the algorithm.

Training of the algorithm is only required to be done once before any predictions can be carried out. Once trained, a backup will be created of the trained model as is in a local file, which will be used to load the model next time the console is executed. If any new training samples are obtained they can be appended to the training file and then explicitly used to retrain the algorithm. This new model will then replace the old stored file.

7.7 File Storage

Each file obtained from the server, the processed data samples and the training data will not be removed from local storage, until a specific classification problem is defined and the features and classification algorithm are optimised. This will prevent any data loss in case of future changes are made with more critically the data obtained from the wearable devices. Even though the raw data is summarised from the feature extractions they will still persist in case of any additional analysis of the data. The processed data will only take very little space compared to the raw data and so will not have a large impact on its storage.

Because there will be many files and they will also containing different type of data, the files are logically stored in subfolders. The root folder is the folder ‘Data’ which will contain all the files and subfolder stored used by the application. The subfolder ‘raw’ will contain all the raw data samples from the server which it received them from the client. Each file will be named the same as the file sent which is ‘id_yyyyMMddhhmm.csv’, where 'id' is the client’s ID stored in the Android client application and ‘yyyyMMddhhmm’ is the year, month, day, hour and minute of when the recording on the client started. Another subfolder used is the ‘processed’ which will contain two files one for the extracted features from a raw data sample and the other the extracted features which also contain the actual output classification used for training. Each time a raw data sample is processed the results will be appended to the appropriate file. If a processed data sample is going to be used for training purposes it will first obtain its classification either 1 (high risk) or 0 (low risk), then it will be appended to

the training file. Finally, there is also a subdirectory ‘model’ which will contain the serialised trained model.

Chapter 8

Conclusion

8.1 Conclusions

For the theoretical and practical analysis of the classification algorithms three data sets have been used, obtained from the Department of Psychology, the first regarding the experiential avoidance in smokers, the second for diagnosis in eating disorders and third for diagnosis of experiential avoidance for anxiety. For all three sets a total of 10 algorithms (logistic regression, Naive Bayes, k-nearest, classification tree, neural network, SVM, bagging, adaptive boosting, gradient tree boosting and random forest) have been used to make predictions on a test set and evaluate their performance. Their criteria wasn't only based on accuracy but also on the positive predicted value, negative predicted value, sensitivity, specificity and area and curve (ROC). Taking all these into account the best performing algorithms for the first data set are k-nearest, Naive Bayers, SVM and logistic regression, for the second data set are random forest and SVM, while for the third only the random forest algorithm.

Even though we see that the SVM and random forest algorithm are classified for two out of the three cases as one of the highest accuracy models examined we can't state that they're the best algorithms out of the 10 evaluated. This is because we have a small data sample for each cases and only three problems have been evaluated which isn't sufficient to make strong conclusions. Other than the case with the SVM and random forest algorithm, we notice that for the three data sets different algorithms are found to

have the best results and so we can state that no single algorithm can be the best to use for every case. The choice of the algorithm depends on the problem and factors such as if the problem can be solved linear or non-linear, data noise, outliers, correlation of features and the total number of features. In conclusion, every problem must be analysed individually and multiple algorithms must be compared between each other on the data set before deciding the final model.

A noticeable comparison between the algorithms is the classification decision tree with the ensemble algorithms that use as a weak learner the same classification tree. The ensemble algorithms execute the base weak learner multiple times using each time a different subset of the training data set. We can see from executing multiple times and adding an element of randomness is better than just building one tree with all the training data. In general the accuracy of the ensemble algorithms in comparison to the classification tree is equal or better performance wise.

From the market research on the nonintrusive wearable devices it is concluded that most manufactures use a closed eco system regarding the availability of the signal data to 3rd party developers. The manufactures usually allow to just view processed data, some to extract the processed values and even fewer allow access to the raw signal, this is even the case for some devices used to monitor patients. Most devices are marketed towards general fitness and sport performance, as for they target a much larger consumer range rather than for health monitoring for patients. For the two consumer groups the demand in accuracy is higher for monitoring medical conditions, which is reflected on the price range. The criteria to select one or more devices was that together they must obtain signals from ECG or PPG, GSR and ECG, able to extract the raw signal, preferably with a similar method, not be constrained through their cloud platform and must be within a budget. The devices that were finally selected are *MS Band 2* and *Muse.*

Having a stream of data can't be used as such in a machine learning algorithm as for the stream varies in total vector size it has and hence the number of features. Also, they do not take into account the time series of the data when training the model and then predicting the output. One way to turn the stream of data to a static number of features is to extract statistical features. For the experimental data sets conducted in the

lab for each trial period for each data sample the stream is summarised by the average values. On the other hand, the data from the wearable devices are continuous throughout the day without being able to expect specific behaviour at a specific time period. So more statistical features will be extracted to accommodate this additional constraint by including minimum, maximum, average, standard deviation, variance, kurtosis and skewness.

8.2 Problems

From the three experimental data set analysed we can see that the results from the classification algorithms have very similar results, without anyone of them having a distinct better performance. This is mainly due to the fact that each data set has a relative small number of data having 32 samples for the experiential avoidance of smokers, 79 samples for diagnosis of eating disorders and 51 for the experiential avoidance for anxiety. Having any noise obtained from the sensors or any outliers in the data samples have a greater negative impact as for they can't be successfully located and filtered out from such a small data sample. Also, the algorithms will be more prone to presenting the overfitting problem in the training set and possibly in the testing data too. This will lead them to being incapable of accurately predicting the class for any new unknown data samples and skewing the accuracy, which will lead to wrongly ranking the algorithms.

The selected wearable device *MS Band 2* has a PPG sensor used to monitor the heart. However, the device does not offer a way to directly obtain the raw PPG signal but only the processed values of it being the heart rate and the heart rate variability whenever they change. This is not a problem by itself, as for the processed values are preferred to the raw signal, but the fact that they do not offer any insight or the used algorithm on how these data are extracted. In addition, the device filters out variation values from the sensor regarding to users motion and breathing, but again does not offer the exact methodology this is done. So the obtained processed values from the PPG signal loses in trustworthiness.

Having the requirement for nonintrusive for the wearable devices and the devices do not store locally the data an intermediate device is needed which will be used

mostly for storing the data, for this case an Android smartphone. Combining the fact that measurements will be taken for a prolonged time throughout the day and a total of four measurements will be stored it will require a significant amount of storage space for a single session. However, the typical smartphone has limited storage space available and more so the fact that the average user will have little to no free storage space. To reduce this problem the Android application created will only store the current instance of values every 2 seconds, but having a smaller data sample means there will be a compromise to the overall accuracy when using the data for classification purposes.

8.3 Direction for Future Work

As discussed previously for the limited available data one way to mitigate the problem is by using cross validation prediction method rather than a train/test data split to evaluate the algorithms. Using cross validation prediction method means training the algorithm a total times as equal as the number of samples in the data set with each time leaving one unique sample out. This way the maximum number of samples are used to train the model to accurately predict the value of the sample that remained out. The results from cross validation prediction will be more in par with the expected results from estimating the output value of new data samples, as for the whole known data will be used to train the final model. If in the future more data is available for each set then the train/test split method should be sufficient.

The current version of the console is just the initial implementation and isn't using a problem tailored classification algorithm and uses all available extracted features which might not bring the best results. When enough data is collected from the wearable devices and analysed and classified it should be used to compare possible classification algorithms to find that that has the highest accuracy. Also, it can be used to compare which subset of the extracted features help in brining better results and/or reduce training time. In addition the accuracy of the wearable devices must be evaluated with the lab equipment, so the data can be shifted towards the ideal values of that from the lab. This way all data received will be scaled making the console and predictor undependable from the external device used to collect the data.

The current methodology used with the data from the wearable devices is to store the data stream and then extract statistical features, which will be used to solve a classification problem. However, this method as already stated requires a lot of storage space which is a limited resource for smartphones. Another possible strategy is to explore the possibility of obtaining an estimated classification based on the dynamic data stream. This way no data is stored on the smartphone device, but will require more processing power and hence more battery usage. This trade off should be explored to identify whether the battery life is sufficient so it can collect a prolonged data stream and not interfere with the daily usage of the user's smartphone. One such possible model is the associative classification algorithm for data streams, which is based on the Lossy counting and the landmark window model [48].

Bibliography

[1] A. Ajanki, *https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm#/media/File:KnnClassification.svg*, 2007.

[2] M. Akay, *Wiley Encyclopaedia of biomedical engineering*, New York, NY, United States: John Wiley, 2006.

[3] A. A. Alian and K. H. Shelley, *Photoplethysmography*, Best Practice & Research Clinical Anaesthesiology, Vol. 28, No. 4, 2014, pp. 395-406.

[4] J. Allen, *Photoplethysmography and its application in clinical physiological measurement*, Physiological Measurement, Vol. 28, No. 3, 2007, pp. 1-39.

[5] H. Bäcklund, A. Hedblom and N. Neijman, *DBSCAN A Density-Based Spatial Clustering of Application with Noise*, 2011.

[6] P. J. Braspenning and F. Thuijsman, *Artificial neural networks: An introduction to ANN theory and practice*, A. J. M. M. Weijters, Ed. Berlin: Springer, 1995.

[7] L. Breiman, *Random Forests*, Machine Learning, Vol. 45, No. 1, 2001, pp. 5-32.

[8] L. Breiman, *Bagging Predictors*, Machine Learning, Vol. 24, 1996, pp. 123-140.

[9] L. Breiman and A. Culter, *Random forests*. [Online]. Available: https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm. Accessed: Oct. 3, 2016.

[10] P. Buhlmann and T. Hothorn, *Boosting Algorithms: Regularization, Prediction and Model Fitting*, Statistical Science, Vol. 22, No. 4, 2007, pp. 477-505.

[11] C.J.C Burges, *A Tutorial on Support Vector Machines for Pattern Recognition*, Data Mining and Knowledge Discovery, Vol. 2, No. 2, 1998, pp. 121-167.

[12] R. Caruana and A. Niculescu-Mizil, *An Empirical Comparison of Supervised Learning Algorithms*, Proceeding ICML '06 Proceedings of the 23rd international conference on Machine learning, 2006, pp. 161-168.

[13] S. Cheriyedath, Photoplethysmography *(PPG)*, in Health, News-Medical, 2016. [Online]. Available: http://www.news-medical.net/health/Photoplethysmography-(PPG).aspx. Accessed: Nov. 12, 2016.

[14] Chire, *https://en.wikipedia.org/wiki/DBSCAN#/media/File:DBSCAN-density-data.svg*, 2011.

[15] M. Ester, H. Kriegel, J. Sander and X. Xu, *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise*, Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining, 1996.

[16] T. Fawcett, *An introduction to ROC analysis*, Pattern Recognition Letters, Vol. 27, No. 8, 2006, pp. 861-874.

[17] D. Fricker, Electroencephalogram (EEG). [Online]. Available: https://www.mayfieldclinic.com/PE-EEG.HTM. Accessed: Nov. 26, 2016.

[18] J. Friedman, T. Hastie and R. Tibshirani, *The elements of statistical learning: Data mining, inference, and prediction*, 2nd ed. New York, NY: Springer-Verlag New York, 2009.

[19] J. Friedman, *Greedy Function Approximation: A Gradient Boosting Machine*, The Annals of Statistics, Vol. 29m No. 5, 2001, pp. 1189-1232.

[20] M. Galar et al., *A Review on Ensembles for the Class* Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Application and Reviews), Vol. 42, No.4, 2011, pp. 463-484.

[21] T. Germano, *Self-organizing maps*, 1999. [Online]. Available: http://davis.wpi.edu/~matt/courses/soms/. Accessed: Oct. 4, 2016.

[22] S. M. Guthikonda, *Kohonen Self-Organizing Maps*, 2005.

[23] G. James, D. Witten, T. Hastie and R. Tibshirani, *An Introduction to statistical learning: With applications in R*, 1st ed. New York: Springer-Verlag New York, 2013.

[24] G. James, *Majority Vote Classifiers: Theory and Applications*, 1993.

[25] S. Karamizadeh et al. *An Overview of Principal Component Analysis*, Journal of Signal and Information Processing, Vol. 4, 2013, pp. 173-175.

[26] M. Kaur and U. Kaur, *Comparison Between K-Mean and Hierarchical Algorithm Using Query Redirection*, International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 3, No. 7, 2013, pp. 1454-1459.

[27] S. B. Kotsiantis, *Supervised Machine Learning: A Review of Classification Techniques*, Informatica 31, 2007, pp. 249-268.

[28] G. E. Krasner, S. T. Pope, *A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System*, Journal of Object-Oriented Programming, Vol. 1, No. 3, 1988, pp. 26-49.

[29] J. Lever, M. Krzywinski and N. Altman, *Points of Significance: Classification evaluation*, Nature Methods, Vol. 13, No. 8, 2016, pp. 603-604.

[30] A. Liaw and M. Wiener, *Classification and Regression by randomForest*, R News, Vol. 2, No. 3, 2002, pp. 18-22.

[31] O. Maimon and L. Rokach, *Data Mining and Knowledge Discovery Handbook*. New York: Springer-Verlag New York, 2005.

[32] B. McNeese, *Are the Skewness and Kurtosis Useful Statistics?*, 2016. [Online]. Available: https://www.spcforexcel.com/knowledge/basic-statistics/are-skewness-and-kurtosis-useful-statistics Accessed: Apr. 24, 2017.

[33] T. M. Mitchell, *Machine learning*, 1st ed. New York: McGraw Hill Higher Education, 1997.

[34] F. Morris, J. Edhouse, W. J. Brady and J. Camm, *ABC of clinical electrocardiography*, London: BMJ, 2003.

[35] M. Namratha and T. R. Prajwala, *A Comprehensive Overview of Clustering Algorithms in Pattern Recognition*, IOSR Journal of Computing Engineering, Vol. 4, No. 6, 2012, pp. 23-30.

[36] A. Natekin and A. Knoll, *Gradient boosting machines, a tutorial*, Frontiers in Neurorobotics, Vol. 7, No. 21, 2013.

[37] M. A. Nielsen, *Neural Networks and Deep Learning*, Determination Press, 2015.

[38] R. Parikh, A. Mathai, and R. Thomas, *Understanding and using sensitivity, specificity and predictive values*, Indian Journal of Ophthalmology, Vol. 56, No. 1, 2008, pp. 45–50.

[39] M. L. Pellizzer, *Body Image-Acceptance and Action Questionnaire*, Springer Nature Singapore, 2016, pp. 1-3.

[40] I. Pop. *An Approach of the Naive Bayes Classifier for the Document Classification*, General Mathematics Vol. 14, No. 4, 2006, pp. 135-138.

[41] J. L. Puga, M. Krzywinski and N. Altman, *Points of Significance: Bayes' theorem*, Nature Methods, Vol. 12, 2015, pp. 277-278.

[42] B. Rohrer, *How to choose algorithms for Microsoft azure machine learning*, 2016. [Online]. Available: https://azure.microsoft.com/en-us/documentation/articles/machine-learning-algorithm-choice/. Accessed: Oct. 3, 2016.

[43] R. E. Schapire, *The Boosting Approach to Machine Learning An Overview*, MSRI Workshop on Nonlinear Estimation and Classification, Vol. 171, 2003, pp. 149-171.

[44] Sewagu, *https://en.wikipedia.org/wiki/Linear_regression#/media/File:Linear_regression.svg*, 2010.

[45] D. Sisodia, L. Singh, S. Sisodia and K. Saxen, *Clustering Techniques: A Brief Survey of Different Clustering Algorithms,* International Journal of Latest Trends in Engineering and Technology, Vol. 1, No. 3, 2012, pp. 82-87.

[46] L. I. Smith, *A Tutorial on Principal Components Analysis*, 2002.

[47] M. Sokolova and G. Lapalme, *A systematic analysis of performance measures for classification tasks,* Information Processing & Management, Vol. 45, No. 4, 2009, pp. 427-437.

[48] L. Su, H. Liu and Z. Song, *A New Classification Algorithm for Data Streams*, Modern Education and Computer Science, Vol. 4, 2011, pp. 32-39.

[49] P. N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. Boston, MA: Pearson Addison-Wesley, 2005.

[50] A. Trigiorgi, *Μελέτη Μεθόδων Μηχανικής και Βαθιάς Μάθησης και Εφαρμογή σε Ψυχομετρικά Δεδομένα,* Diploma Project, Department of Computer Science, University of Cyprus, 2016.

[51] P. D. Turney, *Cost-Sensitive Classification: Empirical Evaluation of a Hybrid Genetic Decision Tree Induction Algorithm,* Journal of Artificial Intelligence, Vol. 2, 1995, pp 369-403.

[52] D. R. Tveter, *Backpropagator's review*, 2004. [Online]. Available: http://www.dontveter.com/bpr/bpr.html. Accessed: Sep. 30, 2016.

[53] A. van Boxtel, *Facial EMG as a Tool for Inferring Affective States,* Proceedings of Measuring Behavior, 2010, pp. 104-108.

[54] E. W. Weisstein, *MathWorld.* [Online]. Available: http://mathworld.wolfram.com/. Accessed: Apr. 24, 2017.

[55] I. H. Witten and E. Frank, *Data mining: Practical Machine Learning Tools and Techniques*, 2nd ed. San Francisco, CA: Morgan Kaufmann Publishers In, 2005.

[56] X. Wu et al. *Top 10 algorithms in data mining,* Knowledge and Information Systems, Vol. 14, No. 1, 2007, pp. 1–37.

[57] X. Xiao, *Gene Clustering Using Self-Organizing Maps and Particle Swarm Optimization*, Proceedings of 2nd IEEE international workshop on high performance computational biology, 2003.

[58] X. Zhu, *Clustering*, 2010. [Online]. Available: http://pages.cs.wisc.edu/~jerryzhu/cs769/clustering.pdf. Accessed: Oct. 2, 2016.

[59] *Overview of principal component analysis*, in jmp. [Online]. Available: http://www.jmp.com/support/help/Overview_of_Principal_Component_Analysis.shtml. Accessed: Oct. 3, 2016.

[60] *Principal components analysis (PCA)*, in PennState Eberly College of Science, 2016. [Online]. Available: https://onlinecourses.science.psu.edu/stat505/node/49. Accessed: Oct. 3, 2016.

[61] *Self-Organizing Maps (SOMs)*. [Online]. Available: https://genome.tugraz.at/MedicalInformatics2/SOM.pdf. Accessed: Oct. 4, 2016.

[62] *Classification*, 2013. [Online]. Available: https://docs.oracle.com/cd/E11882_01/datamine.112/e16808/classify.htm#DMCON053. Accessed: Nov. 15, 2016.

[63] Electrocardiogram (ECG) and heart rate variability (HRV), 2016. [Online]. Available: http://www.megaemg.com/knowledge/heart-rate-variability-hrv/. Accessed: Nov. 20, 2016.

[64] ECG-Derived respiration, 2014. [Online]. Available: https://www.physionet.org/physiotools/edr/. Accessed: Nov. 20, 2016. (4-6)

[65] Galvactivator - Frequently Asked Questions [Online]. Available: http://www.media.mit.edu/galvactivator/faq.html. Accessed: Nov. 25, 2016.

[66] Microsoft Band Developers. [Online]. Available: https://developer.microsoftband.com/bandSDK Accessed: Aug. 2, 2016.

[67] Muse Developer. [Online] Available: http://developer.choosemuse.com/android. Accessed: Oct. 15, 2016.

[68] scikit-learn Machine Learning in Python. [Online]. Available: http://scikit-learn.org/. Accessed: May 15, 2017.