PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

UNIVERSITY ABBES LAGHROUR OF KHENCHELA

Faculty of Sciences and Technology

Department of Mathematics and Computer science

Dissertation

Submitted in partial fulfillment of the requirements for MASTER degree in Software engineering and distributed systems

Major: Software engineering and distributed systems

Theme

Application of swarm intelligence techniques for positive and unlabeled learning

Presented by **AGGOUN Fouad** and **KHORCHI Iskendar**

Jury:

President	HAOUASSI HICHEM	Professor	University of KHENCHELA
Examiner	LEDMI MAKHLOUF	Professor	University of KHENCHELA
Supervisor	AZIZI NABIL	Professor	University of KHENCHELA

JUNE 2023

Dedication

I dedicate this work to my parents:

May they find here the testimony of my deep gratitude and acknowledgment

I want to express my deep gratitude to my supervisor Dr. AZIZI NABIL from KHENCHELA university, for the confidence he has placed in me, through his presence always with me, by his direction, his modesty, his advice, and constructive remarks for the good progress of this work. To my brothers , my grandparents and my family who give love and liveliness.

To all those who have helped me - directly or indirectly - and those who shared with me the emotional moments during the accomplishment of this work and who warmly supported and encouraged throughout my journey.

To all my friends who have always encouraged me, and to whom I wish more success.

Thanks!

ملخص

يعد التعلم الآلي مجالا صاعدا جذب الكثير من الاهتمام في السنوات الأخيرة و لها العديد من التطبيقات ، مثل حيث تشكل كميات كبيرة من البيانات النصية غير المسماة تحديا وتستغرق وقتاً طويلا في هذه العملية التي تسمى تعلم الايجابي و الغير مصنف (PU).

يعد تحسين حشد الجسيمات خوارزمية (PSO) تعريفية مستخدمة على نطاق واسع تعتمد على سرب من الجسيمات القادرة على إيجاد التخصيص الأمثل. يمكن تطبيقه على مشاكل التحسين المختلفة في جميع أنحاء العالم. تقدم هذه الورقة لمحة عامة عن الحالة الحالية لهاته الخوارزمية.

لقد قدمنا طريقة جديدة تسمى (NBPSO-SVM) وهي (PSO) ثنائي جديد باستخدام مصنف (SVM). للتحقق من صحة هذه الطريقة ، أجرينا سلسلة من التجارب. تظهر النتائج الرائعة لتجاربنا ان الطريقة فعالة ويمكن استخدامها في مجموعة متنوعة من التطبيقات.

Abstract

Machine learning (ML) is an emerging field that has attracted much attention in recent years. ML algorithms solve the well known clustering problem have many applications, such where large amounts of unlabeled text data are a challenging and time consuming this process called PU learning.

Particle Swarm Optimization (PSO) is a widely-used meta-heuristic algorithm based on a swarm of particles that are able to find an optimal allotment. It can be applied to various optimization problems across the globe. This paper presents an overview of the current state of the art of PSO and discusses the current status of the algorithm.

We have introduced a new method called NBPSO-SVM (new binary PSO SVM) a new binary PSO using SVM classifier. To validate this method we have made a series of experiments. The admirable results of our experiments show that this method is effective and can be used in a variety of applications.

Résumé

L'apprentissage automatique (ML) est un domaine émergent qui a attiré beaucoup d'attention ces dernières années. Les algorithmes ML résolvent le problème de clustering bien connu ont de nombreuses applications, par exemple lorsque de grandes quantités de données textuelles non étiquetées sont un processus difficile et chronophage appelé apprentissage PU.

Particle Swarm Optimization (PSO) est un algorithme méta-heuristique largement utilisé basé sur un essaim de particules capables de trouver une allocation optimale. Il peut être appliqué à divers problèmes d'optimisation à travers le monde. Cet article présente un aperçu de l'état actuel de l'art de la PSO et discute de l'état actuel de l'algorithme.

Nous avons introduit une nouvelle méthode appelée NBPSO-SVM (nouveau PSO binaire SVM) un nouveau PSO binaire utilisant le classificateur SVM. Pour valider cette méthode, nous avons réalisé une série d'expériences. Les résultats admirables de nos expériences montrent que cette méthode est efficace et peut être utilisée dans une variété d'applications.

Contents

List of Figures

List of Abbreviations

AI: Artificial Intelligence

DL: Deep Learning

ML: Machine Learning

PUL: Positive Unlabeled Learning

EM: Expectation

ROC: receiver operating characteristic curve

AUC: Area under the ROC Curve

SVM: Support Vector Machines

PSO: Particle Swarm Optimization

List of Tables

General Introduction

The impressive growth that qualitative data (categorical data) and quantitative data have experienced in recent decades requires the enhancement of techniques for classifying this data. Automatic classification is one of the main methods.

Machine learning methods constitute a class of attractive techniques for accomplishing the tasks of classifying known and unknown data, which are referred to as labeled and unlabeled data. When chosen wisely, these tools can be employed to assist, and even replace the human operator.

The main problem in classification is to construct a classifier from a set of data. A large amount of data is usually required to create a classification model with a sufficiently high recognition rate. In practice, it is often easy to obtain this amount of data in certain types of applications.

Support vector machines are a set of learning techniques intended to solve problems of discrimination, i.e. deciding which class a sample belongs to, or regression, i.e. predicting the numerical value of 'a variable. The success of this method is justified by the solid theoretical bases that support it. There is indeed a direct link between the statistical learning theory and the learning algorithm of the SVM.

The method then searches for the hyperplane that separates the positive examples from the negative examples, ensuring that the margin between the nearest of the positives and the negatives is maximal. Intuitively, this guar-

antees a good level of generalization because new examples may not be too similar to those used to find the hyperplane but still be clearly located on one side or the other of the border. Another interest is the selection of Support Vectors which represent the discriminating vectors thanks to which the hyperplane is determined. The examples used during the search for the hyperplane are then no longer useful and only these support vectors are used to classify a new case.

Particle Swarm Optimization (PSO) is an optimization algorithm inspired by the behavior of social groups like bird flocks . It involves a population of particles, each representing a potential solution to an optimization problem. These particles move through a search space and adjust their positions and velocities based on personal and global best-known solution among the entire swarm. The goal is to converge towards the optimal solution by iteratively updating particle positions and velocity.

The purpose of this thesis is the combination of swarm optimization algorithms and learning methods applied to positive and unlabeled data.

Chapter 1

Machine Learning

1.1 Introduction

Since their evolution, humans have been using many types of tools to accomplish various tasks in a simpler way. The creativity of the human brain led to the invention of different machines. These machines made the human life easy by enabling people to meet various life needs, including travelling, industries, and computing. And Machine learning is the one among them. According to Arthur Samuel Machine learning is defined as the field of study that gives computers the ability to learn without being explicitly programmed. Arthur Samuel was famous for his checkers playing program. Machine learning (ML) is used to teach machines how to handle the data more efficiently. Sometimes after viewing the data, we cannot interpret the extract information from the data. In that case, we apply machine learning. With the abundance of datasets available, the demand for machine learning is in rise. Many industries apply machine learning to extract relevant data. The purpose of machine learning is to learn from the data. Many studies have been done on how to make machines learn by themselves without being explicitly programmed. Many mathematicians and programmers apply several approaches to find the solution of this problem which are having huge data sets.[1]

Machine learning is an application of Artificial Intelligence. While AI is the umbrella term given to machines emulating human abilities, machine learning is a specific branch of AI where machines are trained to learn how to process and make use of data. The objective of machine learning is not only effective data collection but also to make use of the ever-increasing amounts of data being gathered by manipulating and analyzing them without heavy human input[2]

Difference between Artificial Intelligence , Machine Learning and Deep Learning

Artificial Intelligence, Machine Learning, and Deep Learning have become the most talked-about technologies in today's commercial world as companies are using these innovations to build intelligent machines and applications. And although these terms are dominating business dialogues all over the world, many people have difficulty differentiating between them. This blog will help you gain a clear understanding of AI, machine learning, and deep learning and how they differ from one another.

Before jumping into the technicalities, let's look at what tech influencers, industry personalities, and authors have to say about these three concepts.[3]
"AI doesn't have to be evil to destroy humanity – if AI has a goal and humanity just happens in the way, it will destroy humanity as a matter of course without even thinking about it, no hard feelings. **Elon Musk**, Technology Entrepreneur, and Investor.[3]

Artificial Intelligence, deep learning, machine learning whatever you are doing if you do not understand it , learn it. Because otherwise, you're going to be a dinosaur within 3 years. **Mark Cuban**, American entrepreneur, and television personality.[3]

The three terms are often used interchangeably, but they do not quite refer to the same things.

Here is an illustration designed to help us understand the fundamental differences between artificial intelligence, machine learning, and deep learning.[3]

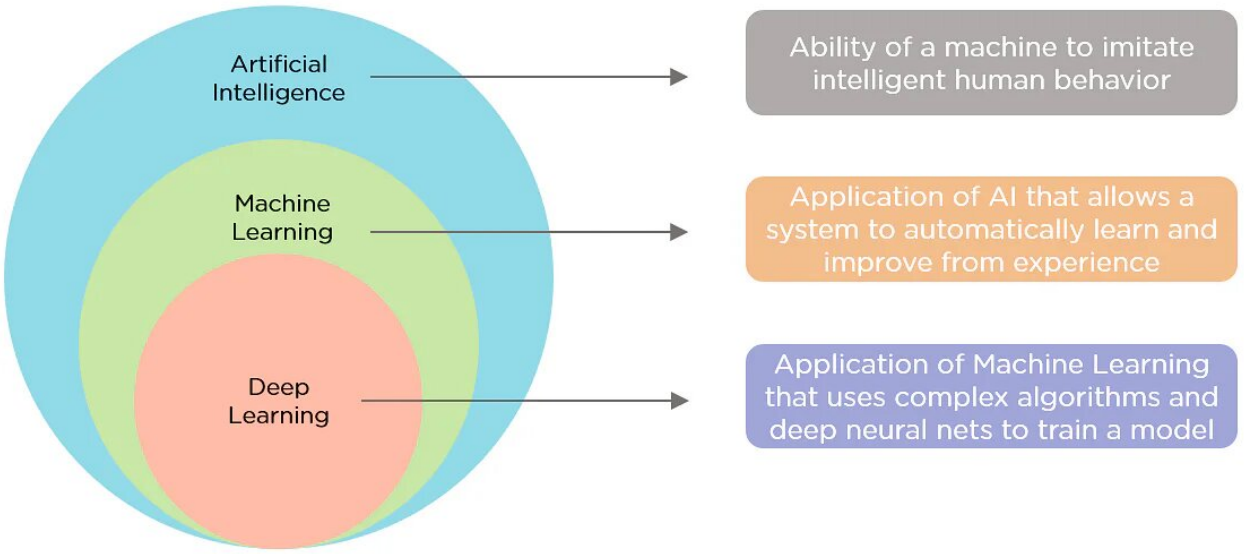


Figure 1.1: A schematically representation of the interrelation of AI, ML, and DL

Now, let's explore each of these technologies in detail.

Artificial Intelligence?

Artificial intelligence, commonly referred to as AI, is the process of imparting data, information, and human intelligence to machines. The main goal of Artificial Intelligence is to develop self-reliant machines that can think and act like humans. These machines can mimic human behavior and perform tasks by learning and problem-solving. Most of the AI systems simulate natural intelligence to solve complex problems.[3]

Machine Learning?

Machine learning is a discipline of computer science that uses computer algorithms and analytics to build predictive models that can solve business problems.

As per McKinsey and Co., machine learning is based on algorithms that

can learn from data without relying on rules-based programming.

Tom Mitchell's book on machine learning says "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."[3]

Deep Learning?

Deep learning is a subset of machine learning that deals with algorithms inspired by the structure and function of the human brain. Deep learning algorithms can work with an enormous amount of both structured and unstructured data. Deep learning's core concept lies in artificial neural networks, which enable machines to make decisions.

The major difference between deep learning vs machine learning is the way data is presented to the machine. Machine learning algorithms usually require structured data, whereas deep learning networks work on multiple layers of artificial neural networks.[3]

1.2 Machine Learning

1.2.1 Definition

Machine learning is a sub-domain of artificial intelligence The goal of machine learning is usually to understand the structure of the data and to match that data to models that can be understood and used by humans. While artificial intelligence and machine learning are often used together, they are two different concepts. AI is a broad concept – decision-making machines, learning new skills, and problem-solving in the same way for people - and machine learning is an AI set that enables intelligent systems to independently learn new things from the data.[2]

We have seen machine learning as a trendy expression for hardly any years, the meaning behind this could be the high rate of data/information creation by applications, the expansion of computation power over the years, and the development of better algorithms.[2]

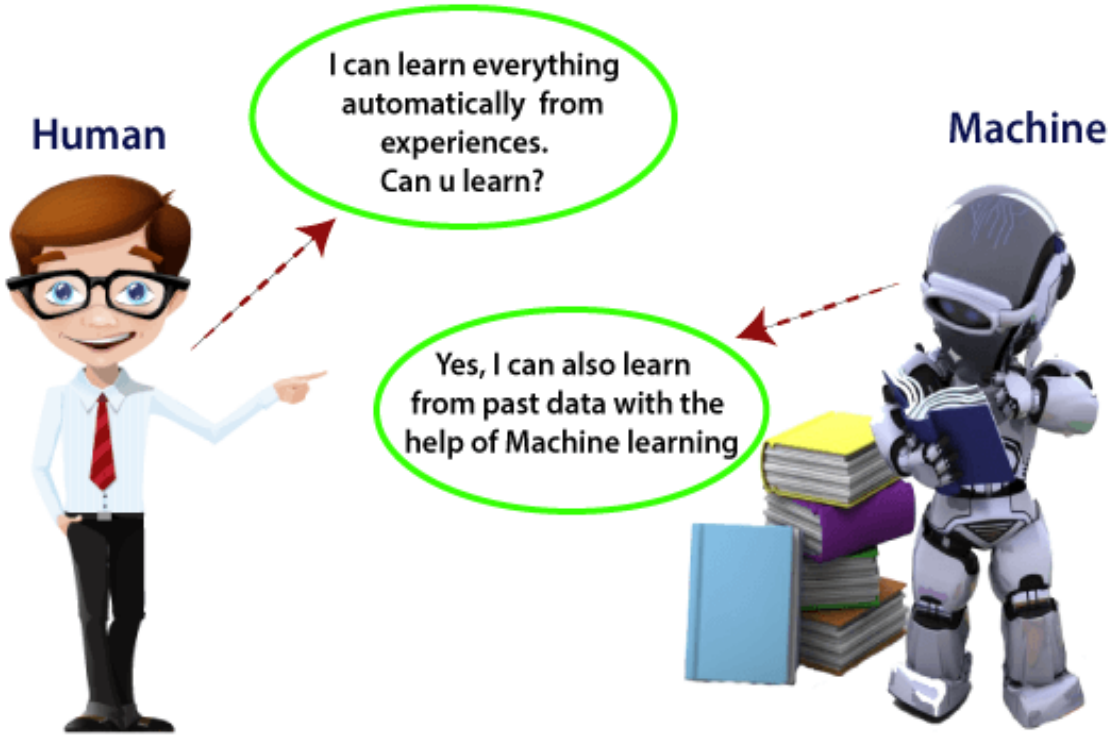


Figure 1.2: Human & Robot

The Figure shows that the human learns everything automatically from experience & the robot can also learn from previous experience data with the help of machine learning.

The name machine learning was coined by Arthur Samuel in 1959, an American pioneer in the field of computer gaming and artificial intelligence who stated that Learning gives computers the ability to learn without being explicitly Arthur Samuel created the first self-study program for playing checkers. You realize that the more the system plays, the better it performs.And in

1997, Tom Mitchell gave a mathematical and relational definition that computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience.[2]

1.2.2 Applications of Machine Learning

Machine learning is a buzzword for today's technology, and it is growing very rapidly day by day. We are using machine learning in our daily life even without knowing it such as Google Maps, Google assistant, Alexa, etc. Below are some most trending real-world applications of Machine Learning:[4]

* Image Recognition.
* Speech Recognition.
* Traffic prediction.
* Product recommendations.
* Self-driving cars.
* Email Spam and Malware Filtering.
* Virtual Personal Assistant.
* Online Fraud Detection.
* Stock Market trading.
* Medical Diagnosis .
* Automatic Language Translation.

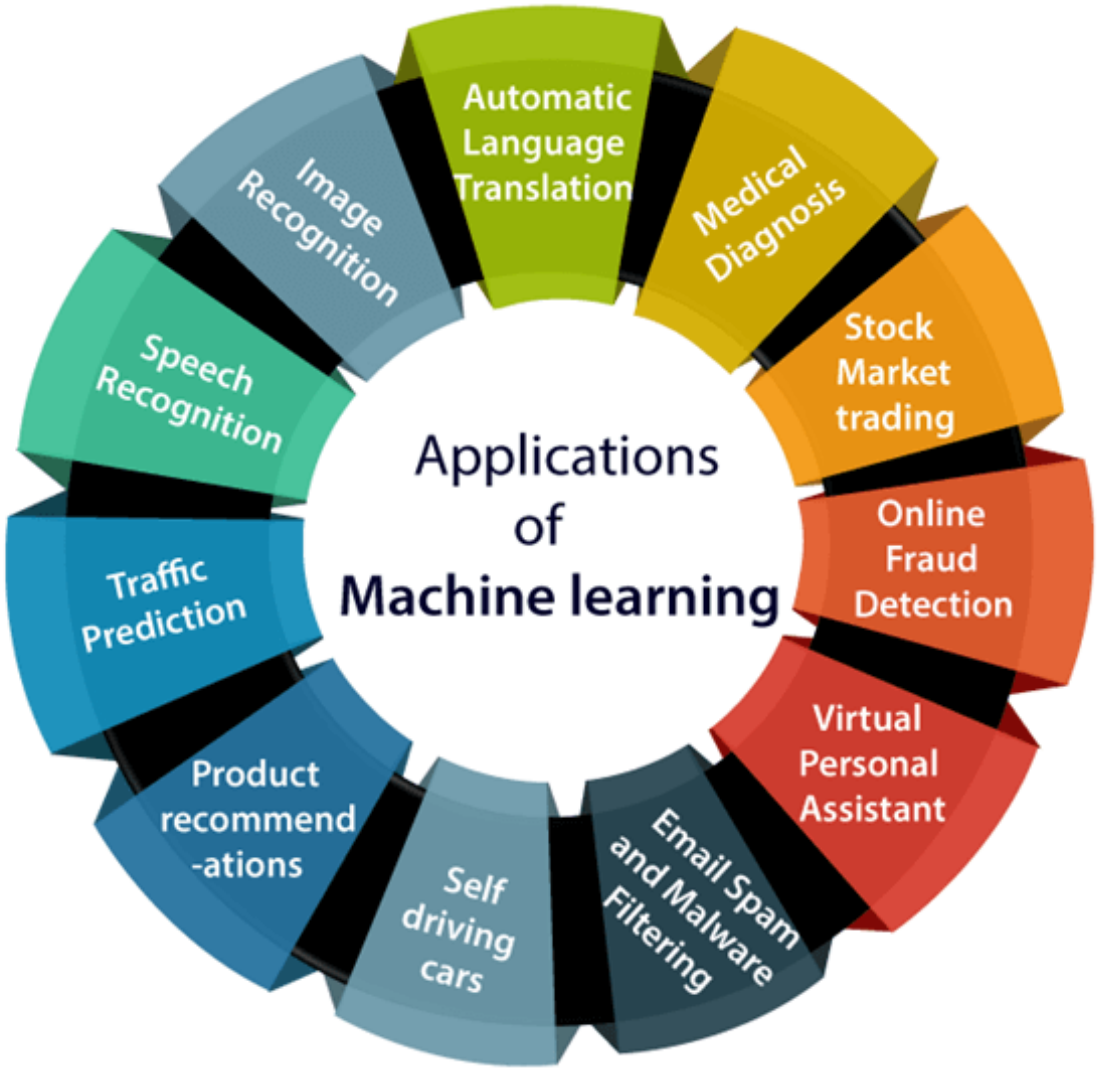


Figure 1.3: Applications of Machine Learning

1.2.3 Types of machine learning

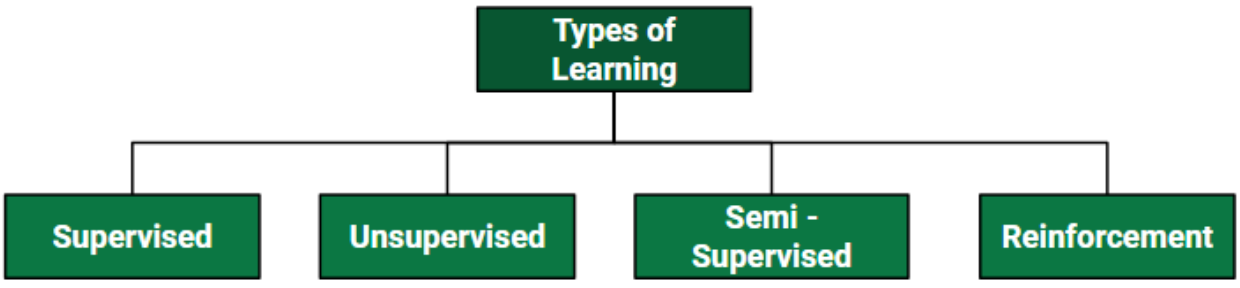


Figure 1.4: Types Of machine Learning

Let's see in detail each type:

1.2.3.1 Supervised Learning

Supervised learning is the most common technique for training neural networks and decision trees. Both of these techniques are highly dependent on the information given by the pre-determined classifications. In the case of neural networks, the classification is used to determine the error of the network and then adjust the network to minimize it, and in decision trees, the classifications are used to determine what attributes provide the most information that can be used to solve the classification puzzle. We'll look at both of these in more detail, but for now, it should be sufficient to know that both of these examples thrive on having some "supervision" in the form of pre-determined classifications.[5]

Supervised learning is an approach to creating artificial intelligence (AI), where a computer algorithm is trained on input data that has been labeled for a particular output. The model is trained until it can detect the underlying patterns and relationships between the input data and the output labels, enabling it to yield accurate labeling results when presented with never-before-seen data.[6]

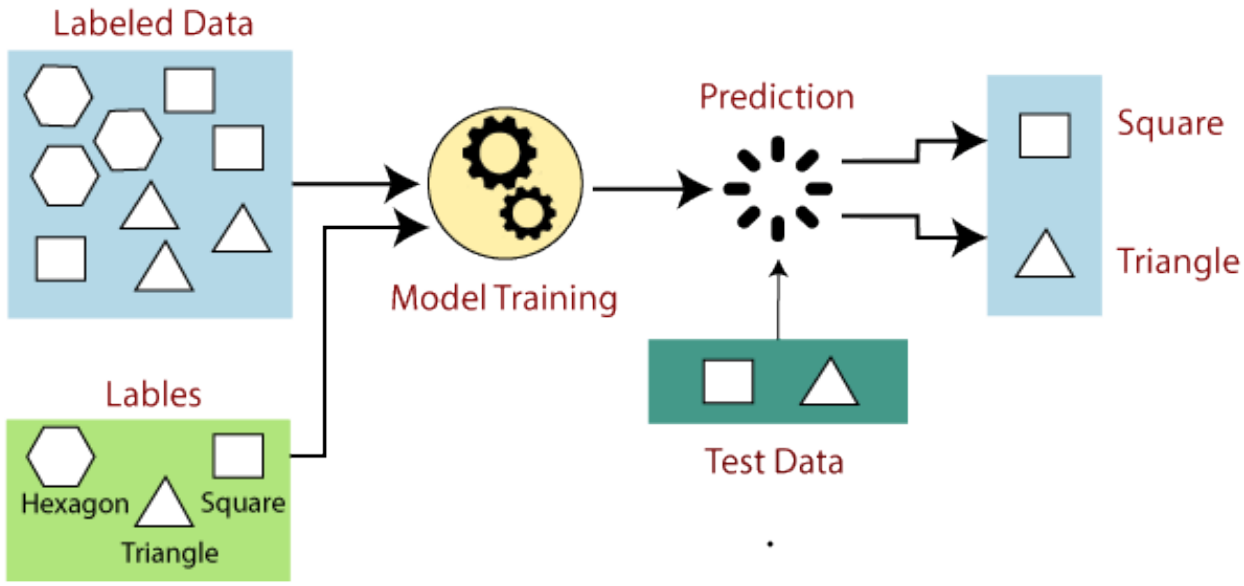


Figure 1.5: Supervised Machine Learning

There are two main types of supervised learning:

Classification: In classification, the goal is to predict a categorical or discrete output variable. The input data is labeled with a specific category or class, and the algorithm learns to classify new, unseen data based on the learned relationships between input and output variables. Some common examples of classification problems include spam detection, sentiment analysis, and image classification.

Regression: In regression, the goal is to predict a continuous output variable r Machine(SVM):Support Vector Machine(SVM) is a supervised machine learning algorithm used for both classification and regression. Though we say regression problems as well its best suited for classification. The objective of SVM algorithm is to find a hyperplane in an N-dimensional space that distinctly classifies the data points. The dimension of the hyperplane depends upon the number of features. If the number of input features is two, then the hyperplane is just a line. If the number of input features is three, then the hyperplane becomes a 2-D plane. It becomes difficult to imagine when the number of features exceeds three.[7](**(For more details See Chapter 3)**

***k-nearest Neighbor method:**

The K-Nearest Neighbors algorithm is a clear and simple supervised machine

learning algorithm that can be used to solve regression and classification problems. The K-NN algorithm assumes that the new case and existing cases are similar and places the new case in the category that is most similar to the existing categories. The K-NN algorithm stores all available data and classifies a new data point based on its similarity to the existing data. This means that when new data appears, the KNN algorithm can quickly classify it into a suitable category. K-NN is a non-parametric algorithm, which means it makes no assumptions about the data it uses. It's also known as a lazy learner algorithm because it doesn't learn from the training set right away; instead, it stores the dataset and performs an action on it when it comes time to classify it. Pattern recognition, data mining, and intrusion detection are some of the demanding applications.[2]

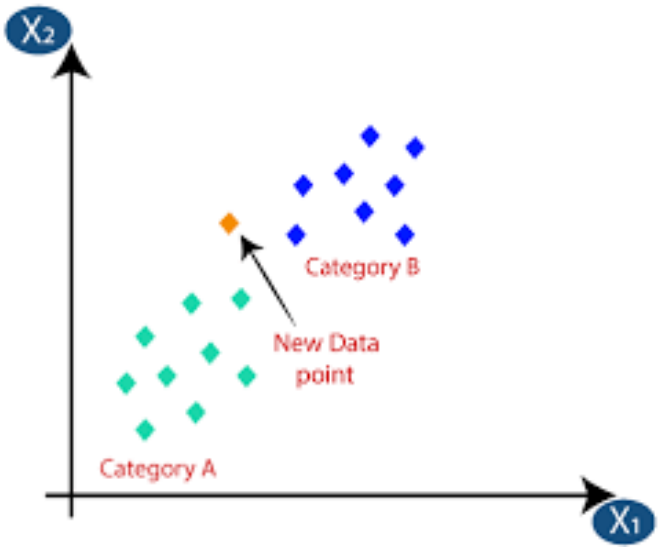


Figure 1.6: KNN example.

*Naive Bayes:

Naive Bayes is a simple learning algorithm that utilizes Bayes' rule together with a strong assumption that the attributes are conditionally independent given the class. While this independence assumption is often violated in practice, naive Bayes nonetheless often delivers competitive classification accuracy. Coupled with its computational efficiency and many other desirable features, this leads to naive Bayes being widely applied in practice.[8]

Structure of Learning System

Naive Bayes is a form of Bayesian Network Classifier based on Bayes' rule

$$\mathrm{P}(y \mid \mathbf{x}) = \mathrm{P}(y)\mathrm{P}(\mathbf{x} \mid y)/\mathrm{P}(\mathbf{x}) \tag{1.1}$$

together with an assumption that the attributes are conditionally independent given the class. For attribute-value data, this assumption entitles[8]

$$\mathrm{P}(\mathbf{x} \mid y) = \prod_{i=1}^{n} \mathrm{P}\left(x_i \mid y\right) \tag{1.2}$$

where xi is the value of the i_{th} attribute in x, and n is the number of attributes[8].

$$\mathrm{P}(\mathbf{x}) = \prod_{i=1}^{k} \mathrm{P}\left(c_i\right) \mathrm{P}\left(\mathbf{x} \mid c_i\right) \tag{1.3}$$

where k is the number of classes and ci is the ith class. Thus, (1) can be calculated by normalizing the numerators of the right-hand-side of the equation. The resulting classifier uses a linear model, equivalent to that used by logistic regression, differing only in the manner in which the parameters are chosen. For categorical attributes, the required probabilities P(y) and P(xi | y) are normally derived from frequency counts stored in arrays whose values are calculated by a single pass through the training data at training time. These arrays can be updated as new data are acquired, supporting incremental learning. Probability estimates are usually derived from the frequency counts using smoothing functions such as the Laplace estimate or an m- estimate.[8]

Decision Tree:

Decision Tree algorithm belongs to the family of supervised learning algorithms. Unlike other supervised learning algorithms, the decision tree algorithm can be used for solving regression and classification problems too.

The goal of using a Decision Tree is to create a training model that can use to predict the class or value of the target variable by learning simple decision rules inferred from prior data(training data).

In Decision Trees, for predicting a class label for a record we start from the root of the tree. We compare the values of the root attribute with the record's attribute. On the basis of comparison, we follow the branch corresponding to that value and jump to the next node.[9]

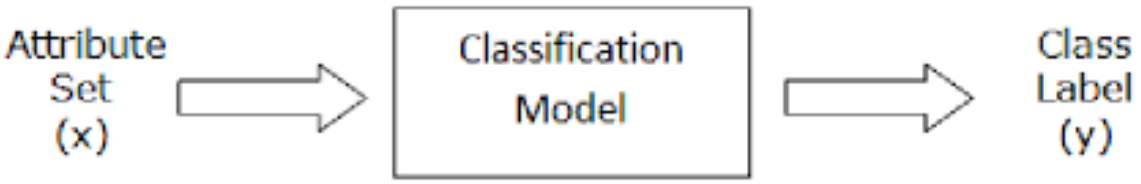


Figure 1.7: Classification of mapping attribute set (X) to its class label (Y).

Decision Tree is similar to the human decision-making process and so that it is easy to understand. It can solve in both situations whether one has discrete or continuous data as input. The example of Decision Tree is as follow:[10]

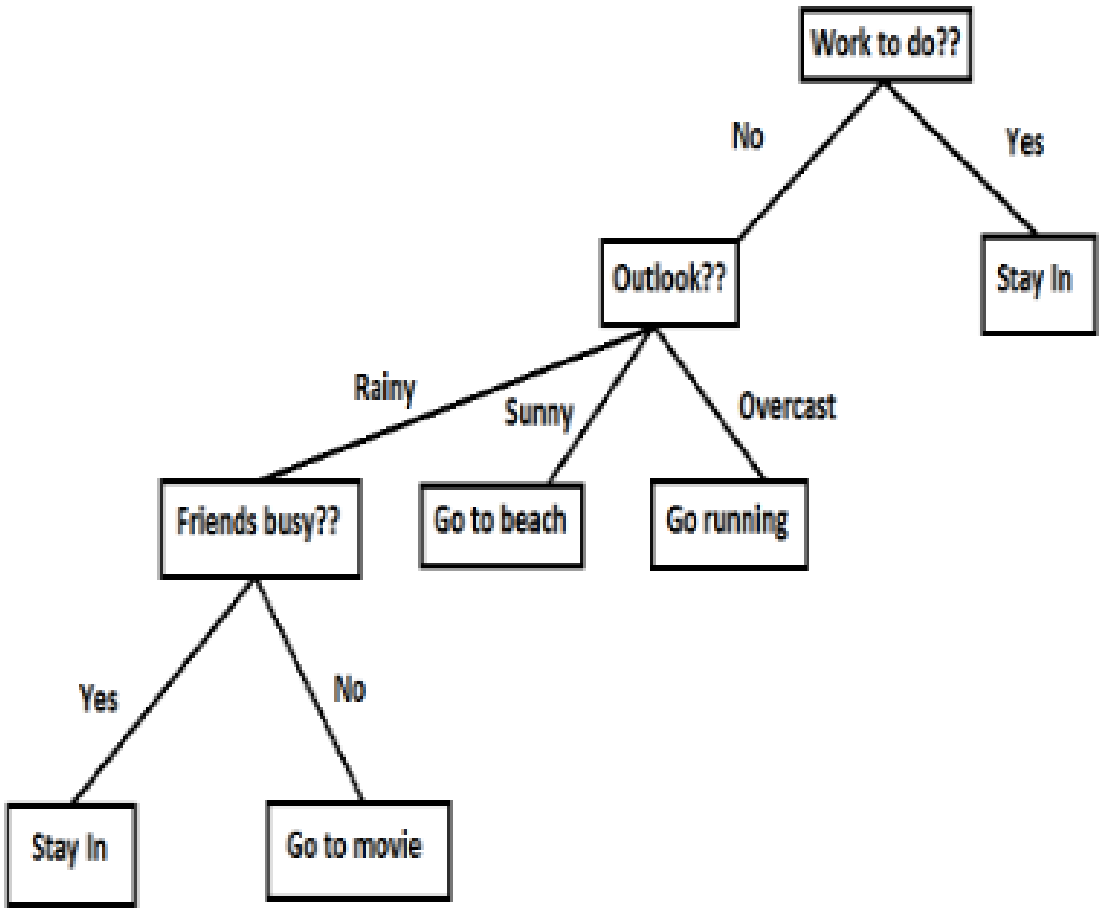


Figure 1.8: Example of Decision Tree on what to do when different situations occur in weather

When data does not offer benefits while splitting, it directly stops the execution. Try to

find one test at a time rather than optimize the whole tree together.[10]

Random forest: A random forest is a machine learning technique that's used to solve regression and classification problems. It utilizes ensemble learning, which is a technique that combines many classifiers to provide solutions to complex problems.

A random forest algorithm consists of many decision trees. The 'forest' generated by the random forest algorithm is trained through bagging or bootstrap aggregating. Bagging is an ensemble meta-algorithm that improves the accuracy of machine learning algorithms.[11]

logistic regression:

ogistic regression is a Machine Learning classification algorithm that is used to predict the probability of certain classes based on some dependent variables. In short, the logistic regression model computes a sum of the input features (in most cases, there is a bias term), and calculates the logistic of the result.

The output of logistic regression is always between (0, and 1), which is suitable for a binary classification task. The higher the value, the higher the probability that the current sample is classified as class=1;[12]

$$h_\Theta(X) = \frac{1}{1 + e^{-\Theta X}} \tag{1.4}$$

Artificial Neural Networks (ANN):

Artificial Neural Networks is inspired by the human brain and it's can be used for machine learning and artificial intelligence. With these networks, various problems can be solved computer-based. The artificial neural network (ANN) is to some extent modelled on the structure of the biological brain. It consists of an abstracted model of interconnected neurons, whose special arrangement and linking can be used to solve computer-based application problems in various fields such as statistics, technology or economics. The neural network is a research subject of Nero informatics and part of the artificial intelligence. Neural networks must be trained before they can solve problems.[13]

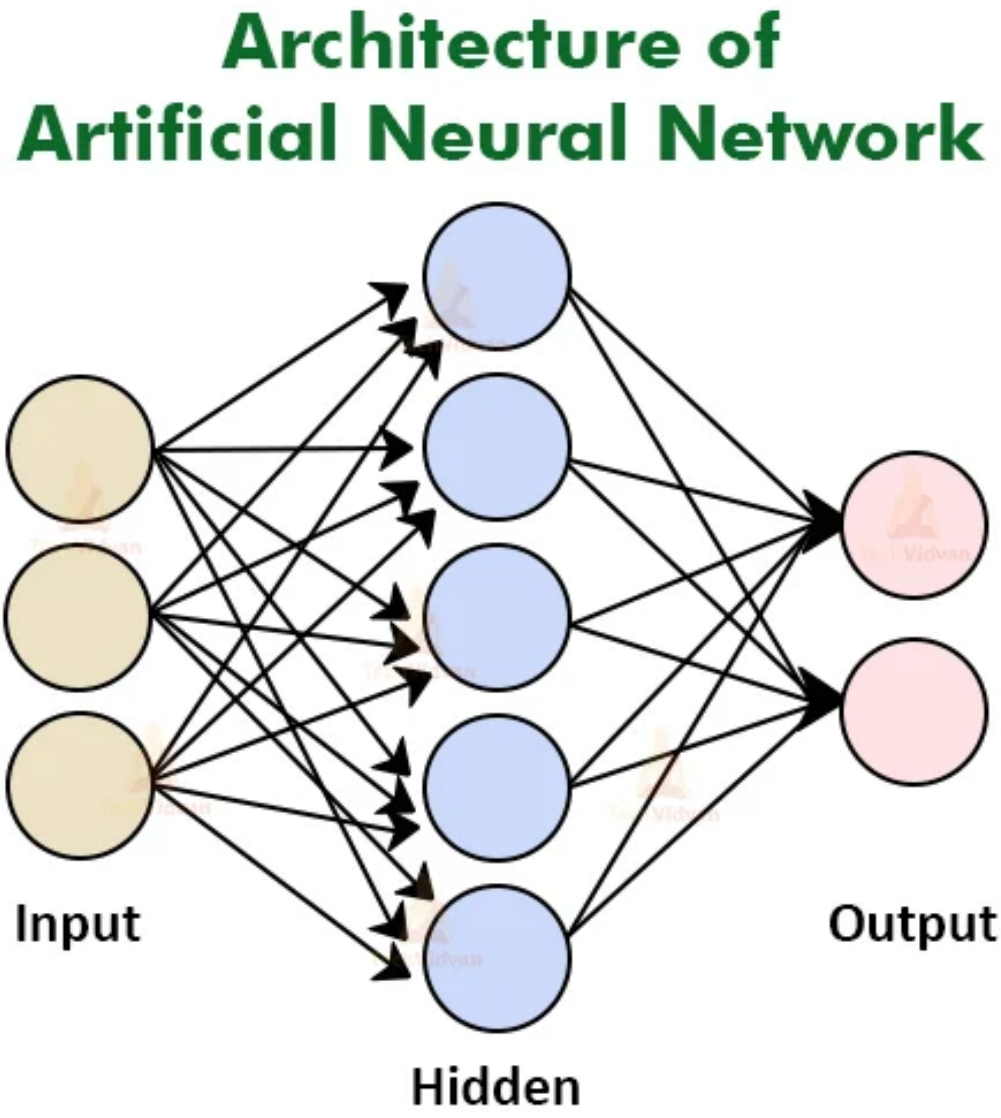


Figure 1.9: Artificial Neural Network Architecture

1.2.3.2 Unsupervised learning :

These are called unsupervised learning because unlike supervised learning above there is no correct answers and there is no teacher. Algorithms are left to their own devises to discover and present the interesting structure in the data. The unsupervised learning algorithms learn few features from the data. When new data is introduced, it uses the previously learned features to recognize the class of the data. It is mainly used for clustering and feature reduction.[14]

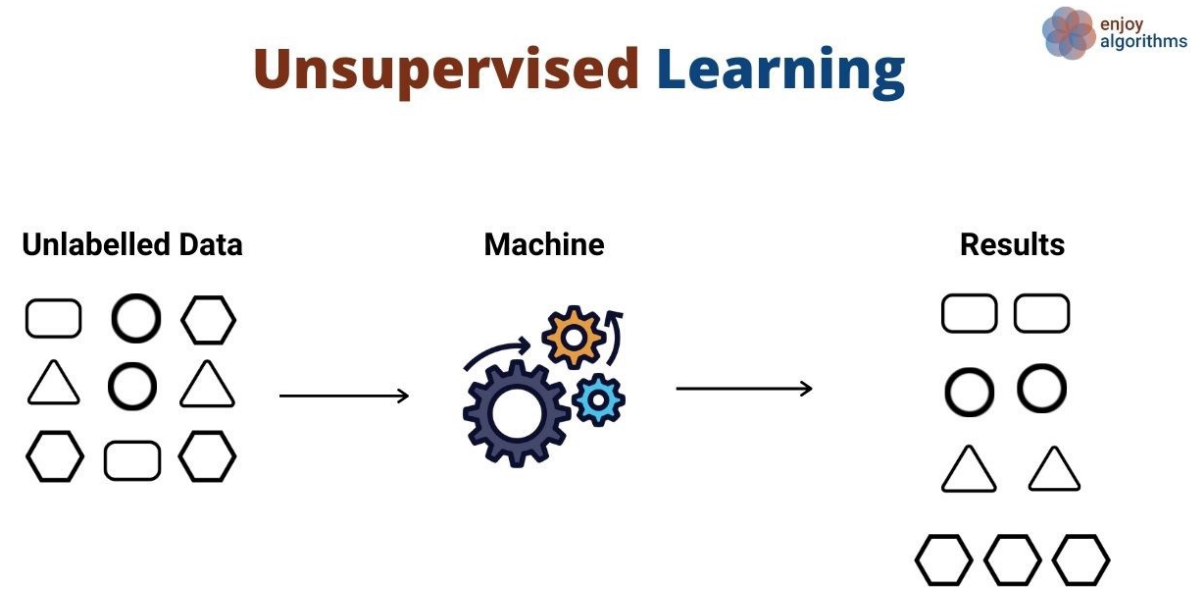


Figure 1.10: Unsupervised learning

There are two types of Unsupervised learning :

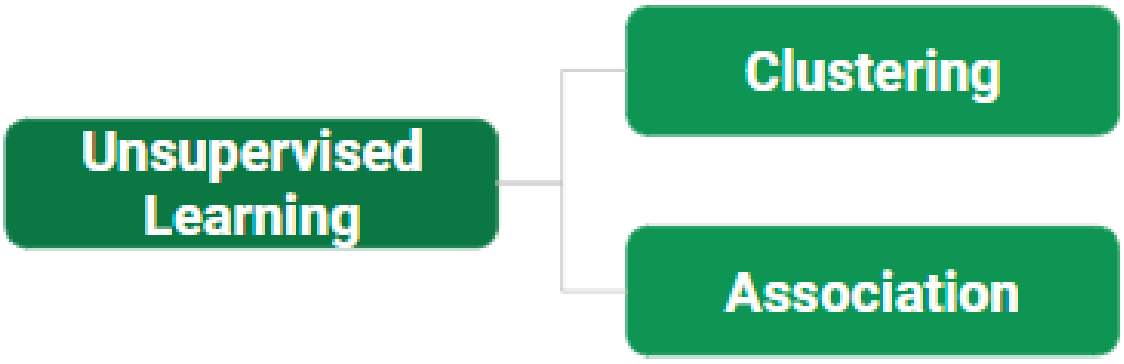


Figure 1.11: Types of Unsupervised learning

A/clustering : Clustering is an unsupervised ML technique. The task of clustering is to find patterns or structures from unstructured data. The process of clustering is to divide the data points or population into some groups, such that data points of the same groups

having similar characteristics while data points belonging to the different groups must have dissimilar properties. These natural groups are called clusters.[2]

B/Association rules: Association rule is unsupervised learning where algorithm tries to learn without a teacher as data are not labelled. Association rule is descriptive not the predictive method, generally used to discover interesting relationship hidden in large data-sets. The relationship are usually represented in form of rules or frequent item-sets[15].

The most famous algorithms used in this approach are :

1. **K-Means Clustering:** K-means is one of the simplest unsupervised learning algorithms that solve the well known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters. The main idea is to define k centers, one for each cluster. These centers should be placed in a cunning way because of different location causes different result. So, the better choice is to place them is much as possible far away from each other.[14]

2. **Principal Component Analysis:**

   Principal component analysis is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. In this the dimension of the data is reduced to make the computations faster and easier. It is used to explain the variance-covariance structure of a set of variables through linear combinations. It is often used as a dimensionality-reduction technique.[14]

3. **Apriori algorithm:** Apriori is an algorithm for frequent item set mining and association rule learning over relational databases. It proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those item sets appear sufficiently often in the database. The frequent item sets determined by Apriori can be used to determine association rules which highlight general trends in the database: this has applications in domains such as market basket analysis. [16]

1.2.3.3 semi-supervised learning

Semi-supervised learning is a machine learning approach where the model is trained using both labeled and unlabeled data. In supervised learning, a model is trained using labeled

data, where each example is associated with a target label. In unsupervised learning, a model is trained using only unlabeled data, without any target labels.

Semi-supervised learning combines these two approaches by using a small amount of labeled data, along with a large amount of unlabeled data, to improve the accuracy of the model. The idea is that the model can learn from the patterns in the unlabeled data to generalize better, even when there is limited labeled data available.

Semi-supervised learning has many applications, especially in cases where labeling data is expensive or time-consuming. For example, in natural language processing, labeling large amounts of text data is a challenging and expensive task. Semi-supervised learning can help overcome this challenge by leveraging the vast amounts of unlabeled text data available online to train more accurate models.

There are various techniques used in semi-supervised learning, including self-training, co-training, and multi-view learning. These techniques involve different ways of leveraging the unlabeled data to improve the model's performance.

1.2.3.4 reinforcement learning

Reinforcement learning is an area of Machine Learning. It is about taking suitable action to maximize reward in a particular situation. It is employed by various software and machines to find the best possible behavior or path it should take in a specific situation. Reinforcement learning differs from supervised learning in a way that in supervised learning the training data has the answer key with it so the model is trained with the correct answer itself whereas in reinforcement learning, there is no answer but the reinforcement agent decides what to do to perform the given task. In the absence of a training dataset, it is bound to learn from its experience.

Reinforcement Learning (RL) is the science of decision making. It is about learning the optimal behavior in an environment to obtain maximum reward. In RL, the data is accumulated from machine learning systems that use a trial-and-error method. Data is not part of the input that we would find in supervised or unsupervised machine learning.[17]

1.2.4 Choosing a type of machine learning

To choose the appropriate type of machine learning, you should consider the following:

1-The nature of the problem you are trying to solve: Different types of problems require different types of machine learning. For example, if you are trying to predict a numerical

value, such as the price of a house, you may want to use regression, whereas if you are trying to classify images, you may want to use convolutional neural networks.

2-The type of data you have: The type and amount of data you have available will also impact your choice of machine learning. For example, if you have labeled data, you may want to use supervised learning, whereas if you only have unlabeled data, you may want to use unsupervised learning.

3-The resources available: The choice of machine learning algorithm may also depend on the resources available, such as computational power and time. For example, some machine learning algorithms may require large amounts of data or computational resources.

4-The performance requirements: Finally, you should also consider the performance requirements for your model. Some machine learning algorithms may be more accurate but slower, while others may be faster but less accurate. You should choose a machine learning algorithm that meets your performance requirements.

The following figure provides guidance on which algorithms to try first according to the factors mentioned:

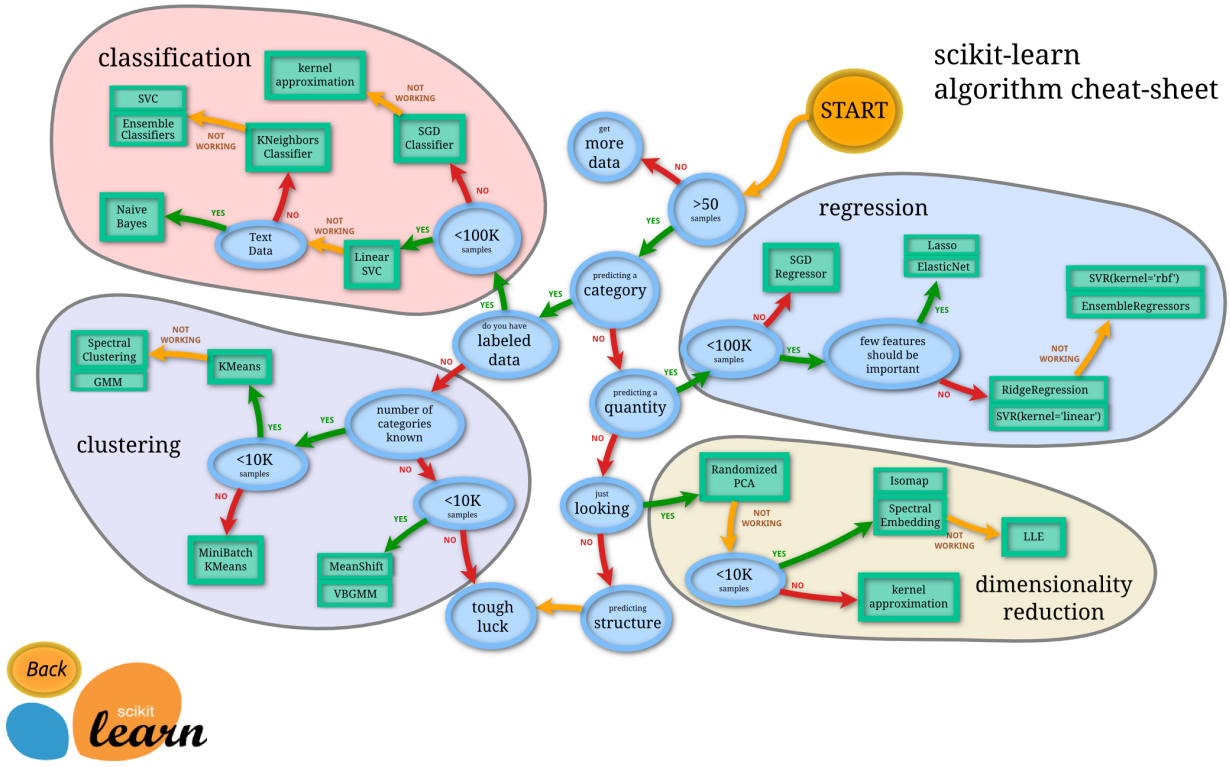


Figure 1.12: choosing a type of ML(Scikit Learn)

1.2.5 Steps of machine learning

The process of machine learning can be broken down into 7 steps as shown in Figure To illustrate the significance and function of each step, we would be using an example of a simple model. This model would be responsible for differentiating between an apple and an orange. Machine learning is capable of much for complex tasks. However, to explain the process in simplistic terms, a basic example is taken to explain the relevant concepts:[2]

Figure 1.13: Steps of Machine Learning

1.3 Conclusion

In conclusion, machine learning is an exciting and rapidly evolving field that offers many opportunities for solving complex problems and making data-driven decisions. With the ability to learn from data and make predictions or decisions without being explicitly programmed, machine learning has the potential to revolutionize the way we interact with data. By understanding the different types of machine learning, key concepts, and common algorithms, we can start to leverage this powerful technology to tackle real-world problems and drive innovation.

Chapter 2

Positive and Unlabeled Learning

2.1 Introduction

Positive Unlabeled Learning (PUL) is a machine learning paradigm where the data consists of two classes: positive examples and unlabeled examples [18]. Figure 2.1 illustrates the difference between supervised learning and PUL. In supervised learning, the data consists of both positive and negative examples, and the algorithm is trained to distinguish between the two classes. In PUL, however, only the positive examples are labeled, and the goal is to learn a classifier that can distinguish between the positive and unlabeled examples.

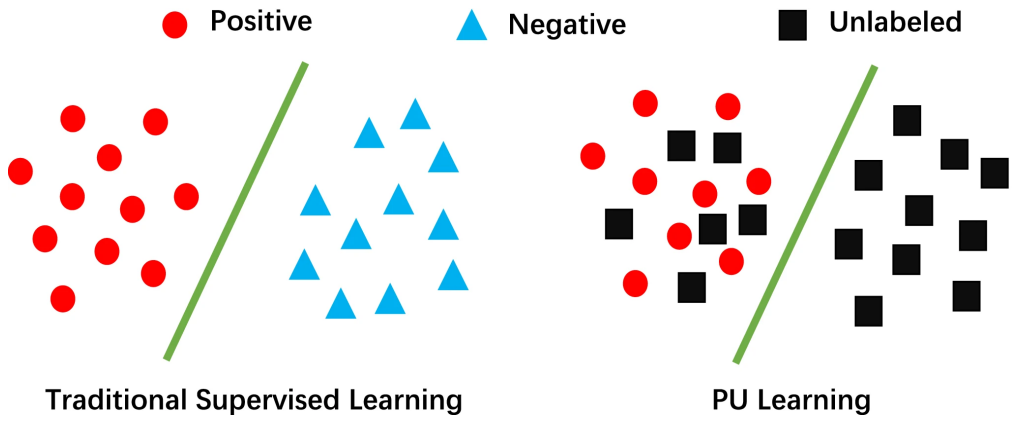


Figure 2.1: Difference between supervised learning and Positive Unlabeled Learning

PUL has various applications in domains such as fraud detection, anomaly detection, and sentiment analysis [19]. The advantage of PUL over traditional supervised learning is that it can learn from large amounts of unlabeled data, which is often more abundant than labeled data in real-world scenarios. However, PUL also presents unique challenges such as the risk of misclassifying unlabeled examples as positive, which can lead to a biased classifier [20].

2.2 Positive Unlabeled Learning

2.2.1 Definition:

Positive-unlabeled learning is a type of machine learning where there are only two classes of data, positive and unlabeled. The positive class represents the data that is of interest and the unlabeled class represents the data that is not known whether it is positive or negative. PU learning is particularly useful when the negative class is difficult or expensive to obtain, which is a common problem in many real-world applications.

The goal of PU learning is to learn a classifier that can accurately predict the positive class in new, unlabeled data. To achieve this goal, PU learning algorithms typically use the labeled positive data and the unlabeled data to learn a decision boundary that separates the positive and unlabeled data. [21] One common approach to PU learning is to use a technique called density estimation. In this approach, the positive and unlabeled data are used to estimate the underlying probability distributions of the two classes. The difference between these distributions can then be used to construct a decision boundary that separates the positive and unlabeled data.

PU learning has many applications, including fraud detection, disease diagnosis, and text classification. It is particularly useful in situations where negative data is difficult to obtain, such as in rare event detection or anomaly detection.[21]

This paper by Liu et al. proposes a new risk estimator for PU learning that is non-negative and does not require a calibration step. The paper also discusses the challenges of PU learning and compares the performance of different algorithms on various datasets. The proposed risk estimator can be used with a variety of PU learning algorithms, including those based on density estimation and those based on kernel methods.

2.2.2 characteristics of positive-unlabeled

PU learning is a form of semi-supervised learning. PU learning can be thought of as a type of semi-supervised learning where only the positive class is labeled, and the negative class is unlabeled. This makes PU learning particularly useful in situations where obtaining negative labeled data is difficult or expensive.[22]

PU learning requires careful selection of unlabeled data. Since the negative class is not labeled, the quality of the unlabeled data can greatly affect the performance of the PU learning algorithm. Careful selection of the unlabeled data can help improve the accuracy of

the learned classifier.[23]

PU learning can be used to handle imbalanced datasets. In many real-world applications, the positive class may be much smaller than the negative class. PU learning can be used to learn a classifier that is biased towards the positive class, helping to handle the class imbalance problem.[24]

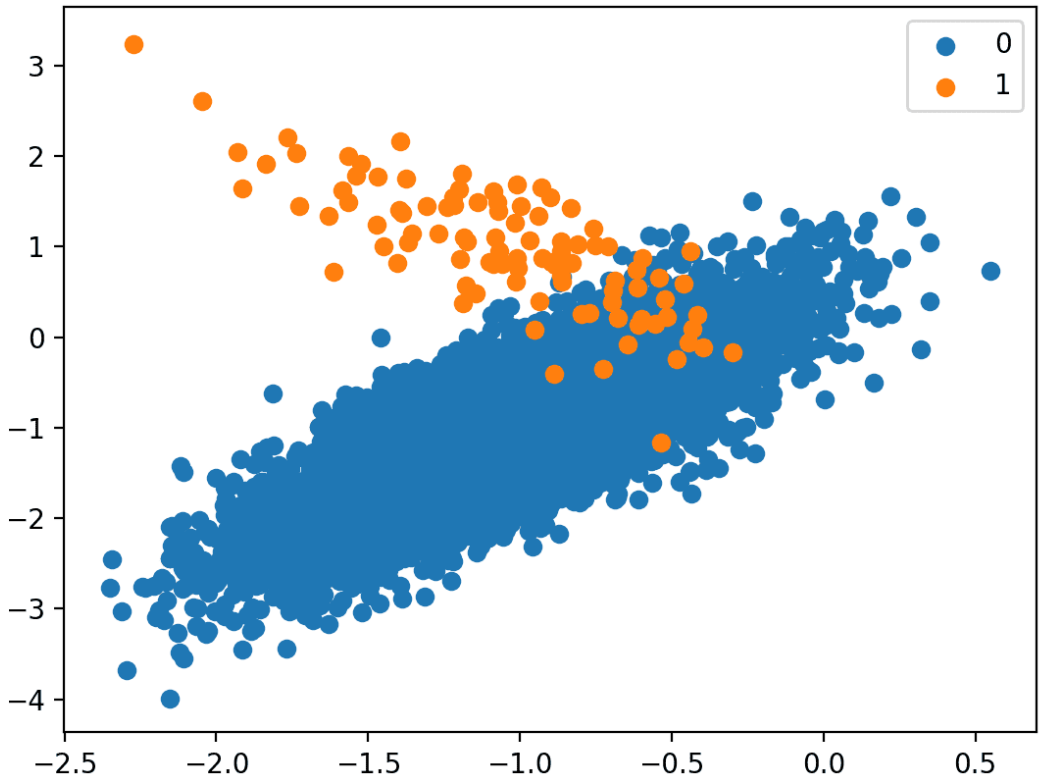


Figure 2.2: svm applied to imbalanced datasets

PU learning can be sensitive to the choice of positive examples. The performance of the PU learning algorithm can be affected by the choice of positive examples used to train the classifier. Careful selection of the positive examples can help improve the accuracy of the learned classifier.[25]

2.2.2.1 Advantages and Disadvantages of Positive unlabeled learning

Advantages:

- PU learning can effectively handle imbalanced classification problems, where the negative class is difficult to define or label.[26]
  *PU learning can be applied in scenarios where obtaining negative labeled data is expensive or impractical.[27]
- PU learning can be used to improve the performance of existing models trained on labeled data, by incorporating additional unlabeled data.[28]
- PU learning may not perform well when the positive labeled examples are noisy or unreliable.[27]

Disadvantages:

- PU learning may require careful tuning of the model parameters, such as the threshold for deciding which unlabeled examples to treat as negative.[27]
- PU learning may not work well when the class priors are highly skewed or when the positive class is rare.[28]

2.3 Methods for Positive Unlabeled Learning

1. **Self-training:** In self-training, the algorithm first uses the labeled positive data to train a model. Then, it uses the trained model to make predictions on the unlabeled data, and the most confident predictions are added to the positive data. The algorithm repeats this process iteratively, adding more data to the positive set until the model's performance reaches a certain threshold.
2. **One-Class SVM:** One-Class SVM is a type of support vector machine that can be used for PU learning. It is trained only on positive data and can be used to classify new data points as either positive or negative.
3. **Expectation-Maximization:** EM algorithm can be used for PU learning. The algorithm assumes that there is a hidden variable that determines whether a data point is positive or negative. The algorithm first estimates the parameters of the model using the positive labeled data and then uses the estimated model to classify the unlabeled data points as either positive or negative.

4. **Positive-Unlabeled Bayesian Network:** This method uses Bayesian Networks to model the relationship between the positive and negative data. The algorithm first builds a Bayesian network using the positive labeled data and then uses it to estimate the probability of a data point being positive or negative.

5. **Positive-Unlabeled Decision Trees:** This method builds a decision tree using the positive labeled data and then uses it to classify the unlabeled data. The algorithm iteratively adds the most informative negative samples to the negative set until the performance of the decision tree reaches a certain threshold.

6. **Positive-Unlabeled Random Forest:** Similar to decision trees, random forests can also be used for PU learning. The algorithm first trains a random forest using only the positive labeled data and then uses it to classify the unlabeled data points.

These are some of the methods for Positive Unlabeled learning, and the choice of method depends on the specific problem's requirements and characteristics.

2.4 Applications of Positive Unlabeled Learning

2.4.1 Fraud detection

Fraud detection is a challenging task as it involves identifying a small number of fraudulent transactions from a large number of legitimate ones. Positive Unlabeled Learning can be used to address this problem by training a binary classifier on a set of labeled positive examples (fraudulent transactions) and unlabeled examples (legitimate transactions).[29]

And this is an example (figure 2.3) The architecture of the proposed credit card fraud detection model

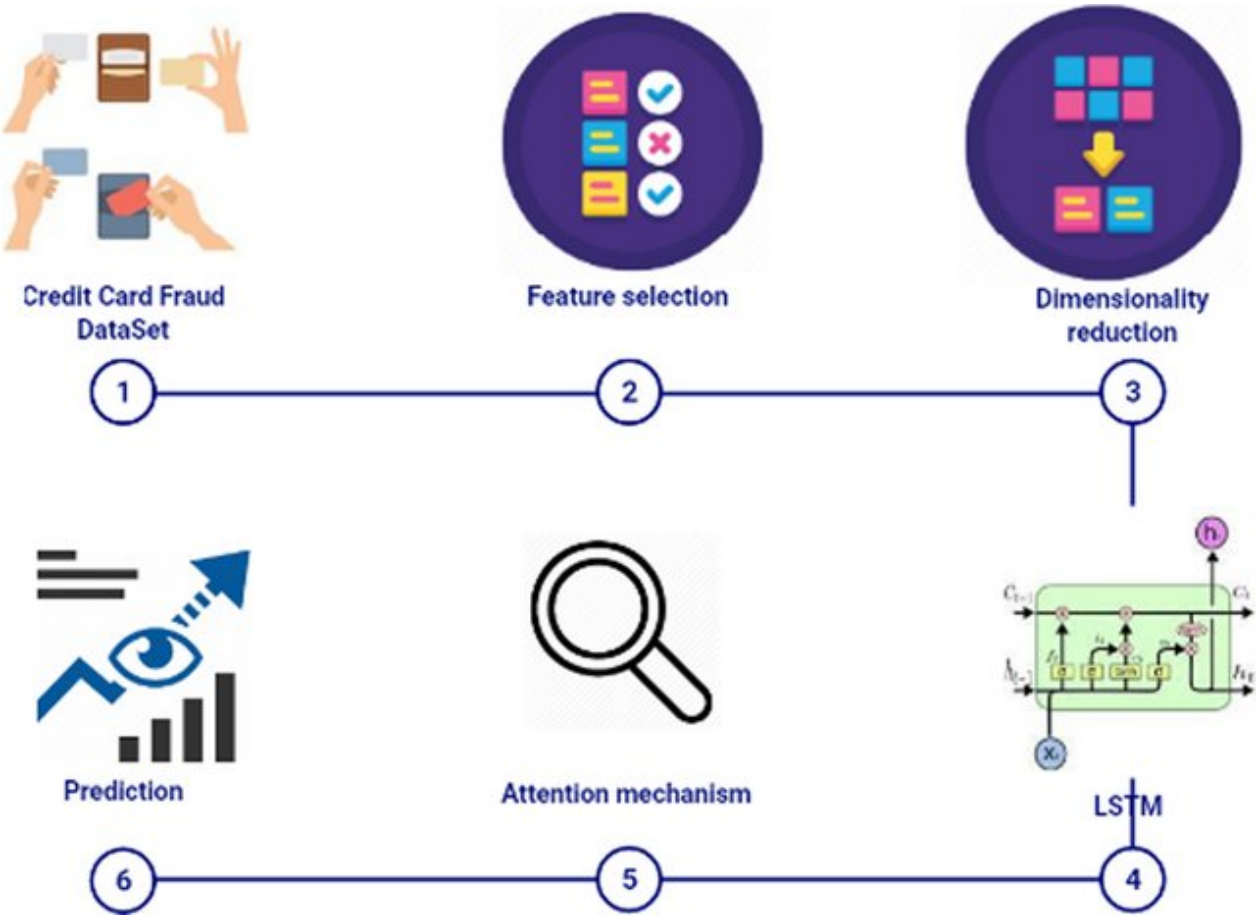


Figure 2.3: The architecture of the proposed credit card fraud detection model

2.4.2 Anomaly detection

Anomaly detection involves identifying rare events or observations that deviate from the norm or expected behavior. Anomaly detection is widely used in many fields, including finance, healthcare, network security, and fraud detection. PUL can be applied to anomaly detection by treating the normal observations as positive examples and the anomalies as the unlabeled data.[30]

One approach to using PU learning for anomaly detection is to train a classifier on a set of normal data and then use the trained model to identify anomalies in new data. This approach has been used in several studies, including:[31] In this study, the authors proposed a novel PU learning approach called PU-AD, which uses a two-stage strategy to detect anomalies. In the first stage, a classifier is trained on the normal data using PU learning. In the second stage, the trained classifier is used to classify new data, and the data points with high uncertainty scores are considered anomalies.[31]

Other studies have also used PU learning for anomaly detection in different domains, such as cybersecurity and medical imaging. Overall, PU learning shows promise as a useful approach for anomaly detection in imbalanced datasets.

2.5 Evaluation Metrics for Positive Unlabeled Learning:

2.5.1 Precision, recall, and F1 score:

Precision measures the proportion of true positives among the instances that the model predicted as positive. Recall measures the proportion of true positives that the model correctly identified among all actual positive instances. F1 score is the harmonic mean of precision and recall.[32]

Before going into the details, an overview figure is always nice:[33]

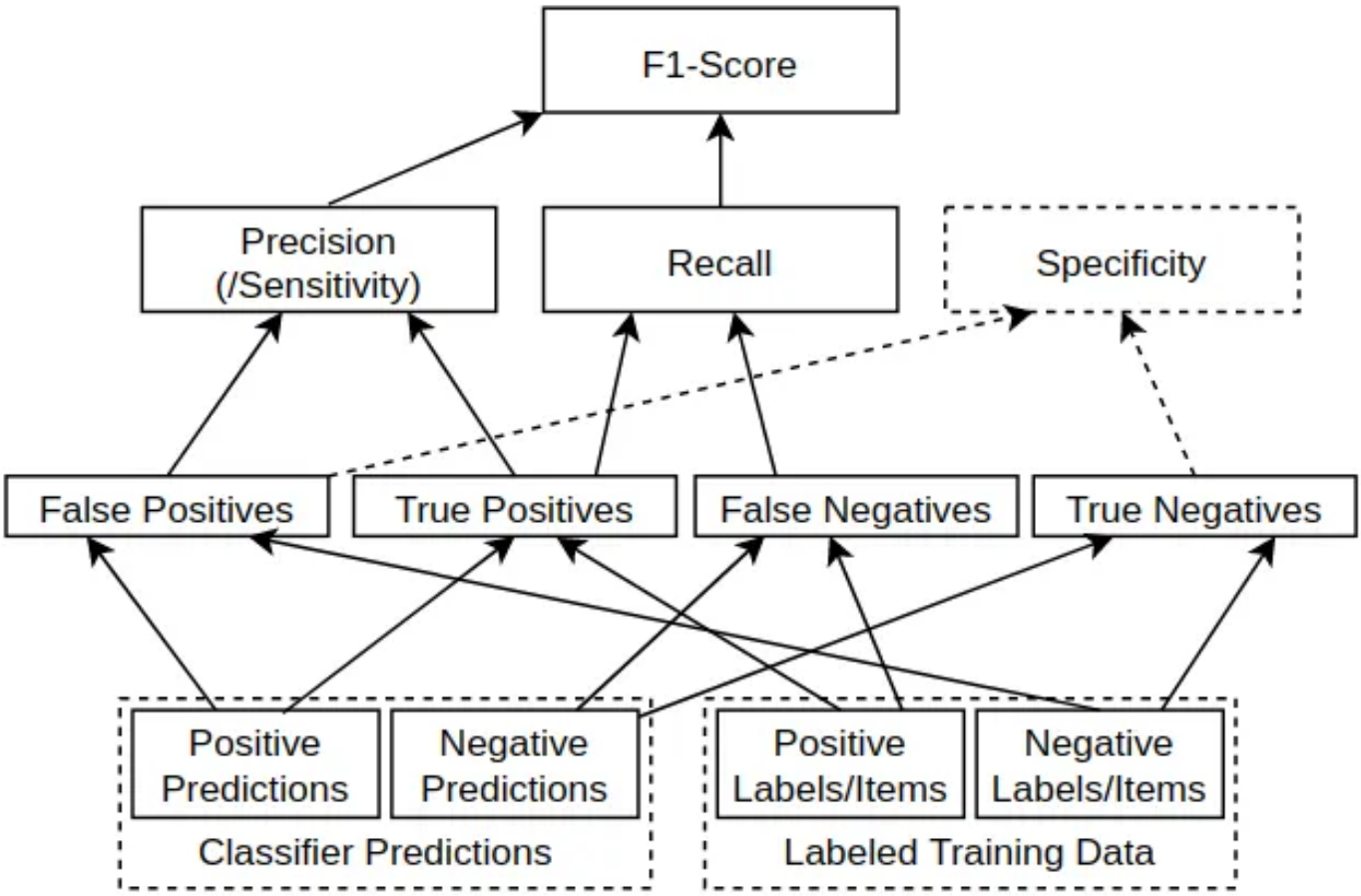


Figure 2.4: Hierarchy of Metrics from raw measurements / labeled data to F1-Score

On the first look, it is a bit of a messy web. No need to worry about the details for now, but we can look back at this during the following sections when explaining the details from the bottom up. The metrics form a hierarchy starting with the the true/false negatives/positives (at the bottom), and building up all the way to the F1-score to bind them all together. Lets build up from there.[33]

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.1}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.2}$$

Precision precision is a measure of how many of the positive predictions made are correct (true positives). The formula for it is:[33]

$$\frac{\text{True Positives}}{\text{True Positives + False Positives}} = \frac{\text{N. of Correctly Predicted Positive Instances}}{\text{N. of Total Positive Predictions you Made}} = \frac{\text{N. of Correctly Predicted People with Cancer}}{\text{N. of People you Predicted to have Cancer}}$$

Figure 2.5: Precision formulas

Recall

is a measure of how many of the positive cases the classifier correctly predicted, over all the positive cases in the data. It is sometimes also referred to as Sensitivity. [33]The formula for it is:

$$\frac{\text{True Positives}}{\text{True Positives + False Negatives}} = \frac{\text{N. of Correctly Predicted Positive Instances}}{\text{N. of Total Positive Instances in the Dataset}} = \frac{\text{N. of Correctly Predicted People with Cancer}}{\text{N. of People with Cancer in the Dataset}}$$

Figure 2.6: Recall formulas

F1-Score

F1-Score is a measure combining both precision and recall. It is generally described as the harmonic mean of the two. Harmonic mean is just another way to calculate an "average" of values, generally described as more suitable for ratios (such as precision and recall) than the traditional arithmetic mean. The formula used for F1-score in this case is:[33]

$$2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

Figure 2.7: F1 Score formulas

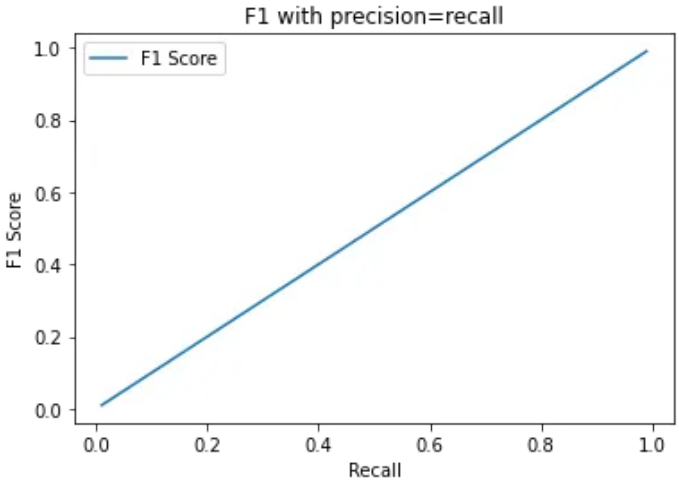
Some advantages of F1-score:

1. Very small precision or recall will result in lower overall score. Thus it helps balance the two metrics.

2. If you choose your positive class as the one with fewer samples, F1-score can help balance the metric across positive/negative samples.

3. As illustrated by the first figure in this article, it combines many of the other metrics into a single one, capturing many aspects at once.

F1-score when precision=recall[33]



	Precision	Recall	F1-Score	Difference
0	0.1	0.1	0.1	0.0
1	0.2	0.2	0.2	0.0
2	0.3	0.3	0.3	0.0
3	0.4	0.4	0.4	0.0
4	0.5	0.5	0.5	0.0
5	0.6	0.6	0.6	0.0
6	0.7	0.7	0.7	0.0
7	0.8	0.8	0.8	0.0
8	0.9	0.9	0.9	0.0
9	1.0	1.0	1.0	0.0

Figure 2.8: F1-score when precision = recall. F1-score equals precision and recall at each point when p=r.

F1-score equals precision and recall if the two input metrics (P and R) are equal. The Difference column in the table shows the difference between the smaller value (Precision/Recall) and F1-score.

in conclusion :

Precision is great to focus on if you want to minimize false positives. For example, you build a spam email classifier. You want to see as little spam as possible. But you do not want to miss any important, non-spam emails. In such cases, you may wish to aim for maximizing precision.

Recall is very important in domains such as medical (e.g., identifying cancer), where you really want to minimize the chance of missing positive cases (predicting false negatives). These are typically cases where missing a positive case has a much bigger cost than wrongly classifying something as positive.

Neither precision nor recall is necessarily useful alone, since we rather generally are interested in the overall picture. Accuracy is always good to check as one option. F1-score is another.

F1-score combines precision and recall, and works also for cases where the datasets are imbalanced as it requires both precision and recall to have a reasonable value, as demonstrated by the experiments I showed in this post. Even if you have a small number of positive cases vs negative cases, the formula will weight the metric value down if the precision or recall of the positive class is low.

Besides these, there are various other metrics and ways to explore your results. A popular and very useful approach is also use of ROC- and precision-recall curves. [33]

2.5.2 receiver operating characteristic curve ROC

An ROC curve (receiver operating characteristic curve) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters:

*True Positive Rate

*False Positive Rate

True Positive Rate (TPR) is a synonym for recall and is therefore defined as follows:

$$\text{TPR} = \frac{TP}{TP + FN} \tag{2.3}$$

False Positive Rate (FPR) is defined as follows:

$$\text{FPR} = \frac{FP}{FP + TN} \tag{2.4}$$

An ROC curve plots TPR vs. FPR at different classification thresholds. Lowering the classification threshold classifies more items as positive, thus increasing both False Positives and True Positives. The following figure shows a typical ROC curve.

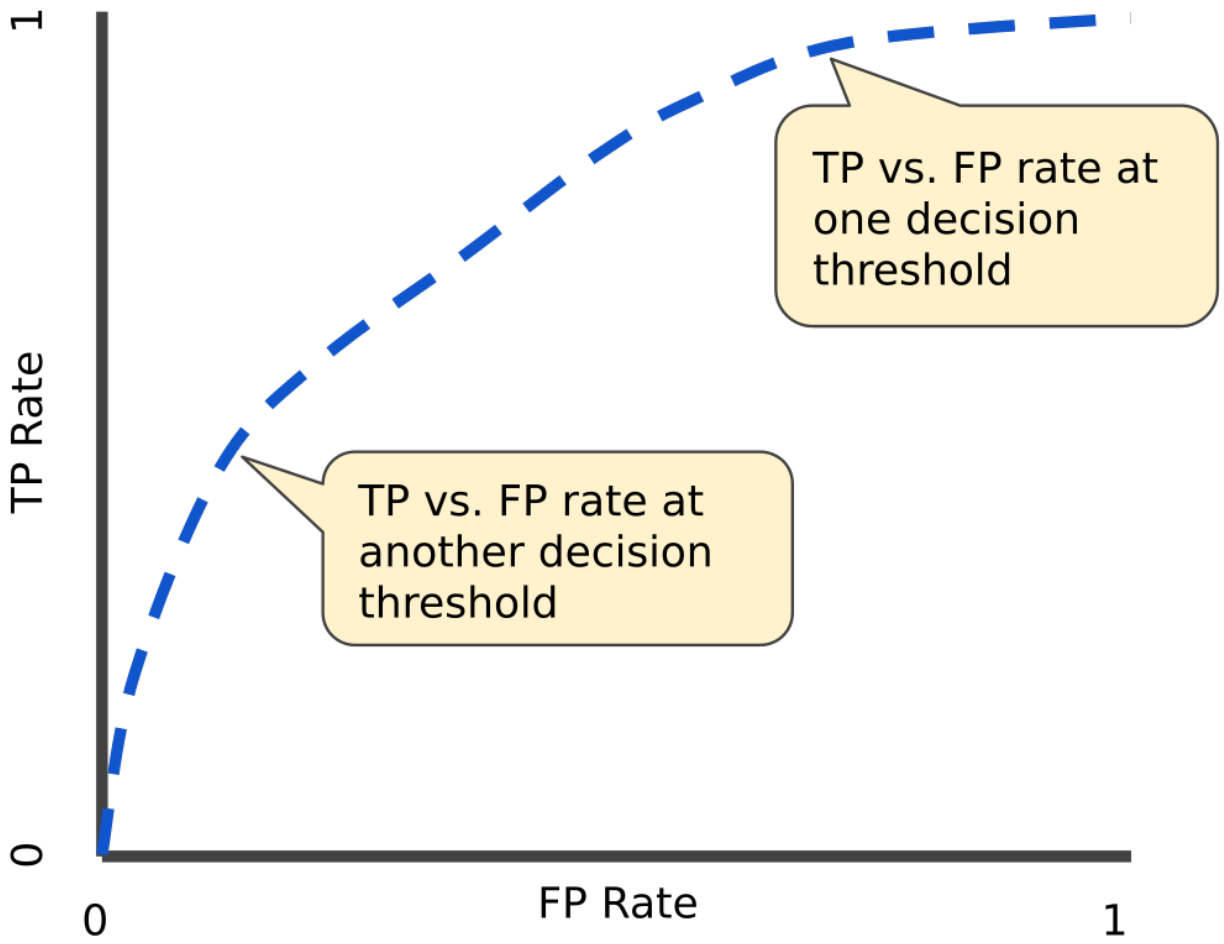


Figure 2.9: TP vs. FP rate at different classification thresholds.

To compute the points in an ROC curve, we could evaluate a logistic regression model many times with different classification thresholds, but this would be inefficient. Fortunately, there's an efficient, sorting-based algorithm that can provide this information for us, called AUC.

2.5.3 AUC: Area Under the ROC Curve

AUC: Area Under the ROC Curve AUC stands for "Area under the ROC Curve." That is, AUC measures the entire two-dimensional area underneath the entire ROC curve (think integral calculus) from (0,0) to (1,1).[34]

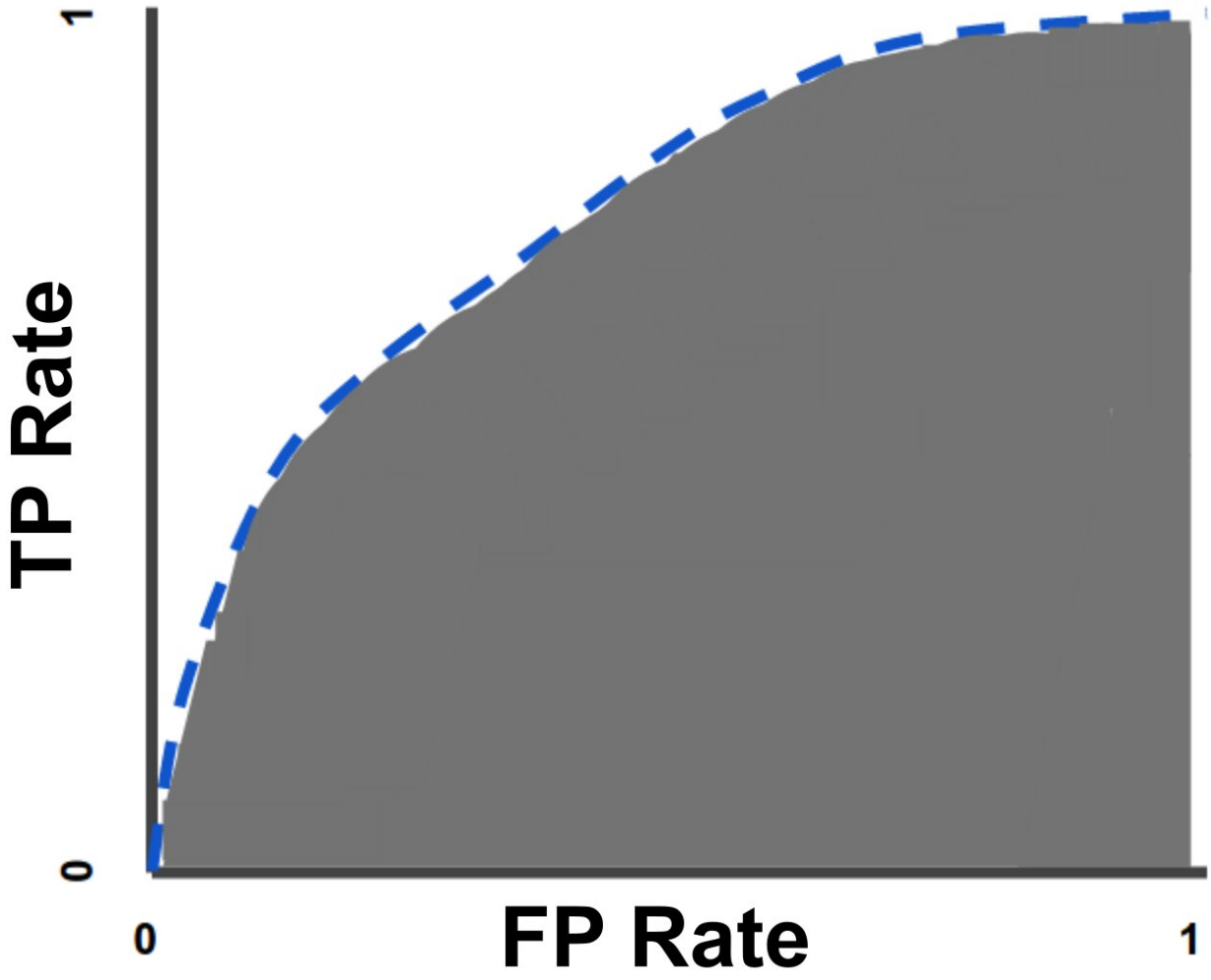


Figure 2.10: AUC (Area under the ROC Curve).

AUC provides an aggregate measure of performance across all possible classification thresholds. One way of interpreting AUC is as the probability that the model ranks a random positive example more highly than a random negative example. For example, given the following examples, which are arranged from left to right in ascending order of logistic regression predictions:

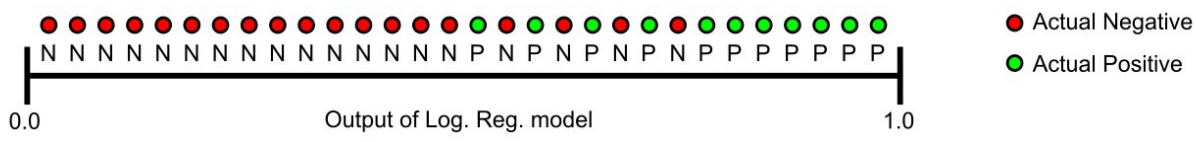


Figure 2.11: Predictions ranked in ascending order of logistic regression score.

AUC represents the probability that a random positive (green) example is positioned to the right of a random negative (red) example.

AUC ranges in value from 0 to 1. A model whose predictions are 100% wrong has an AUC of 0.0; one whose predictions are 100% correct has an AUC of 1.0.

2.6 Future Directions and Challenges

One future direction for positive unlabeled learning is the development of more advanced algorithms that can handle larger and more complex data-sets. As noted by Wang & al. (2019)[35], "there is a need for more scalable and efficient PU learning algorithms that can handle high-dimensional and large-scale data-sets" (p. 154). Another important challenge in positive unlabeled learning is the issue of data bias, which can lead to poor performance on negative examples and false positives in real-world applications (Jiang & al., 2018).[36]

One of the key future directions for positive unlabeled learning is the development of new algorithms and techniques that can handle larger and more complex data-sets. This will require advances in computational power, as well as new methods for data prepossessing, feature extraction, and model selection. Additionally, there is a need for more rigorous evaluation metrics and benchmark data-sets, to ensure that the performance of positive unlabeled learning algorithms can be compared fairly.

Another challenge in positive unlabeled learning is the issue of data bias. Because the training data contains only positive examples and unlabeled examples, there is a risk that the resulting model will be biased towards the positive class. This can lead to poor performance on negative examples, and can even result in false positives in real-world applications. To address this challenge, researchers are exploring a variety of techniques such as bias correction, active learning, and data augmentation.

Finally, there is a need for more research into the theoretical foundations of positive unlabeled learning. While many practical algorithms have been developed, there is still much to be understood about the underlying principles and mathematical properties of these methods. This will be important for developing more robust and effective algorithms, as well as for better understanding the limitations and assumptions of positive unlabeled learning in different contexts.

2.7 Conclusion

In conclusion, positive unlabeled learning is an emerging area of research that aims to address the challenges of imbalanced data-sets, where the negative class is either unknown or difficult to define. This learning paradigm has shown promising results in various domains, such as text classification, image analysis, and bio-informatics.

Several approaches have been proposed for PU learning, including EM algorithms, non-parametric methods, and deep learning models. These techniques have their strengths and limitations, and researchers are actively exploring new techniques to improve the accuracy and efficiency of PU learning.

Despite the success of PU learning, there are still challenges that need to be addressed, such as the issue of data bias, scalability of algorithms, and the need for a more theoretical understanding of the learning paradigm. Further research is needed to develop more advanced algorithms that can handle larger and more complex data-sets, and to enable the application of PU learning in a broader range of real-world applications.

Overall, positive unlabeled learning has the potential to provide significant benefits in many fields and is expected to be an active area of research for years to come. As new developments and innovations continue to emerge, we can expect to see PU learning play an increasingly important role in addressing the challenges of imbalanced data in many applications.

Chapter 3

SVM ,Swarm Intelligence, PSO(Particle Swarm Optimization)

3.1 Introduction

Machine learning has become an essential tool for solving classification and regression tasks in various domains, including computer vision, natural language processing, and bio-informatics. Among the many machine learning algorithms, SVM, (essaim), and PSO have gained popularity for their ability to achieve high accuracy and efficiency in solving complex problems.

3.2 SVM (Support Vector Machines)

3.2.1 General introduction about SVM:

Support Vector Machines (SVM) are a powerful machine learning algorithm widely used for classification and regression tasks. SVMs are known for their ability to handle high-dimensional data, separate classes with a maximal margin, and handle non-linearly separable data using the kernel trick. They have found applications in diverse fields such as image recognition, text classification, and bio-informatics.

One of the seminal works on SVMs is the paper titled "Support-Vector Networks" by Cortes and Vapnik, [37] published in 1995. The authors introduced the concept of SVMs and presented their formulation as a binary classification algorithm based on the idea of finding the hyperplane with the maximum margin. This paper laid the foundation for SVMs and their theoretical underpinnings.

3.2.2 Definition:

(SVM) Support Vector Machines is a supervised learning method ,it is a widely used machine learning algorithm for classification and regression tasks. SVM works by finding a hyperplane that separates the data into different classes with the largest possible margin. SVM has been successfully applied in various domains, including text classification, image recognition, and bio-informatics [38].

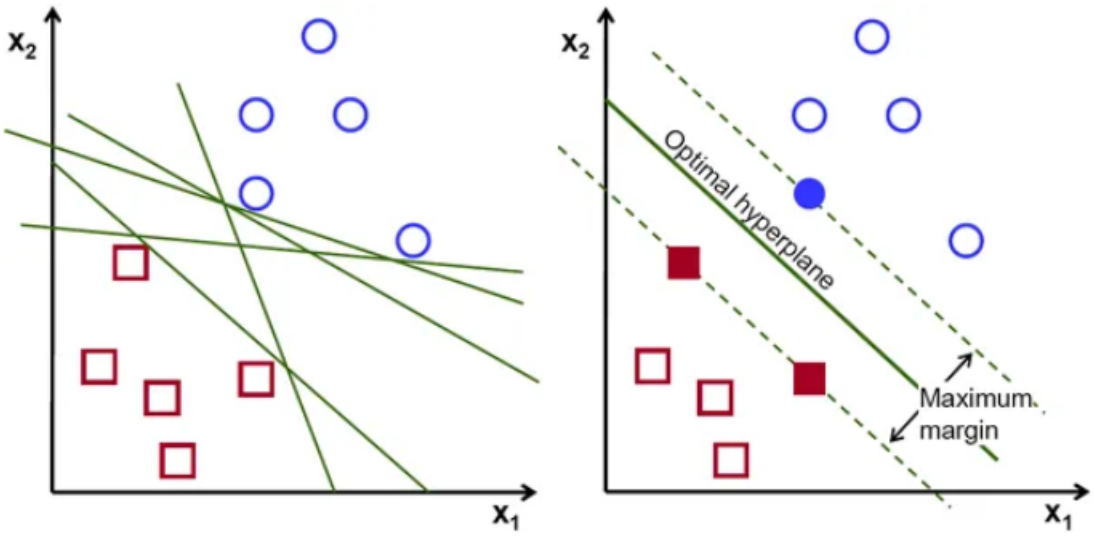


Figure 3.1: Support-Vector-Machine

Margin : Figure 3.2 shows part of the iris data-set The two classes can clearly be separated easily with a straight line. The left plot shows the decision boundaries of three possible linear classifiers. The model whose decision boundary is represented by the dashed line is so bad that it does not even separate the classes properly.
The other two models prefect on this training set, but their decision boundaries is close to the instances that these models will probably not perform as well on new instances. In contrast, the solid line in the plot on the right represents the decision boundary of this classifier. this line separates the two classes but also stays as far away from the closest training instances as possible. You can think of an SVM classifier as fitting the widest possible street (represented by the parallel dashed lines) between the classes . This is called **large margin classification**.

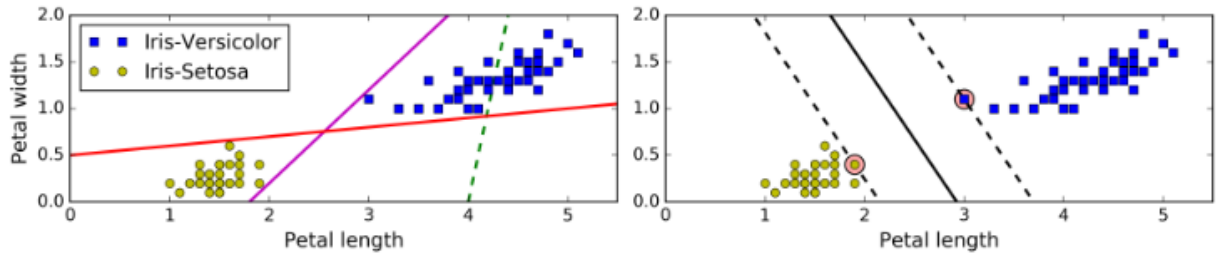


Figure 3.2: Large margin classification

Soft Margin Classification : Hard margin classification has two issues: it only works if the data is linearly separable and it is sensitive to outliers.In Figure 3.3, an additional outlier is added to the iris dataset. As a result, it becomes impossible to find a hard margin on the left, and on the right, the decision boundary looks significantly different from the one depicted in Figure 3.2, which does not have the outlier. Furthermore, this new decision boundary is less likely to generalize well to new data.

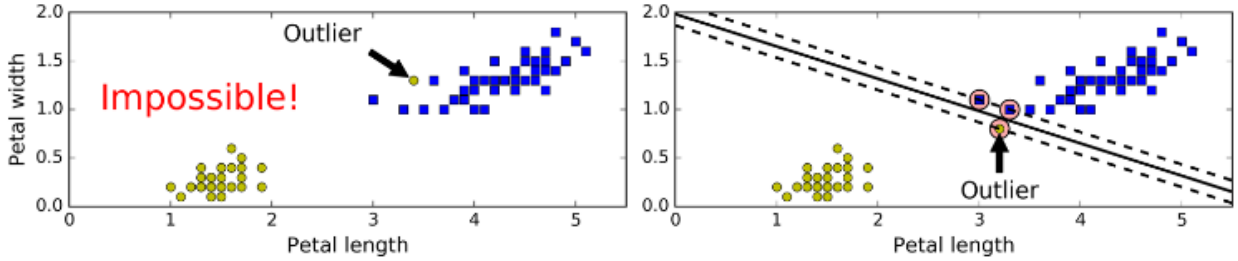


Figure 3.3: Soft margin classification

Soft margin classification is used to balance between the size of the margin and the number of margin violations, in order to avoid the issues of hard margin classification.

3.2.3 Linear SVM and Nonlinear SVM :

Linear SVM and Nonlinear SVM are two variants of the Support Vector Machine algorithm used for classification tasks.

The main difference between these two variants lies in the type of decision boundary they use to separate the data points into different classes. Linear SVM uses a linear decision boundary (a straight line or hyperplane), while Nonlinear SVM uses a nonlinear decision boundary (a curved line or surface).

3.2.3.1 Linear SVM

Linear SVM is suitable for problems where the data is linearly separable, which means that there exists a straight line or hyperplane that can completely separate the two classes. Linear SVM works well in high-dimensional space, and it is computationally efficient [39]

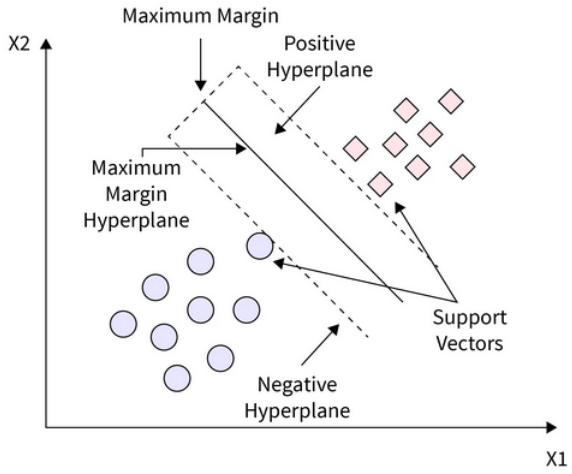


Figure 3.4: Linear SVM

Linear SVM is a variant of Support Vector Machines (SVM) that uses a linear decision boundary to separate classes in a binary classification problem. In linear SVM, the goal is to find the optimal hyperplane that maximizes the margin between the classes.[40] The decision boundary of a linear SVM is a hyperplane defined by the equation:

$$w^T x + b = 0$$

where w is the weight vector perpendicular to the hyperplane, x is the input vector, and b is the bias term. The weight vector w determines the orientation of the hyperplane, and the bias term b controls its position. The key steps involved in training a linear SVM are as follows:

1. Data Preprocessing: The input data is typically standardized or normalized to ensure that all features have similar scales. This step helps prevent any particular feature from dominating the optimization process.

2. Feature Vector Representation: Each input sample is represented as a feature vector, where each feature corresponds to a specific attribute or characteristic of the data

3. Margin Maximization: The linear SVM aims to find the hyperplane with the maximum margin that separates the two classes. The margin is defined as the distance between the hyperplane and the nearest data points from each class.

4. Optimization Problem: The optimization problem involves finding the optimal weight vector w and bias term b that maximize the margin while satisfying the constraint that all samples are correctly classified

5. Support Vectors: The data points that lie on the margin or within the margin are known as support vectors. These support vectors are crucial in defining the decision boundary of the linear SVM.

6. Soft Margin Classification: In cases where the data is not linearly separable, a soft margin classifier can be used. The soft margin allows for some misclassifications by introducing a penalty term for violating the margin constraint. This penalty term is controlled by the regularization parameter C.

Linear SVMs have several advantages, including simplicity, computational efficiency, and good generalization properties. They work well when the data can be separated by a linear decision boundary. However, if the data is not linearly separable, linear SVMs may not perform optimally, and alternative approaches such as kernel SVMs can be considered.

Overall, linear SVMs provide an effective and interpretable method for binary classification tasks, particularly when the data exhibits a clear linear separation

3.2.3.2 Nonlinear SVM

on the other hand, is used when the data is not linearly separable. In this case, a nonlinear decision boundary is required to separate the data points accurately. Nonlinear SVM uses kernel functions to transform the input features into a higher-dimensional space, where a linear decision boundary can be used to separate the data points accurately. Some common kernel functions used in Nonlinear SVM include Polynomial Kernel, Gaussian RBF Kernel, and Sigmoid Kernel.

the decision boundary is obtained by mapping the input data to a higher-dimensional feature space using a kernel function. The decision boundary in the higher-dimensional space is then defined by a hyperplane.

Mathematically, the decision boundary in a non-linear SVM can be represented as:

$$f(x) = sign(\sum_{i} \alpha_i y_i K(x_i, x) + b)$$

where:

1. $f(x)$ is the prediction for a new input vector x, α_i are the Lagrange multipliers obtained during the training process,
2. y_i represents the class labels (-1 or +1) of the training samples,
3. x_i are the support vectors,
4. $K(x_i, x)$ is the kernel function that calculates the similarity or distance between the support vectors x_i and the new input vector x,
5. b is the bias term.

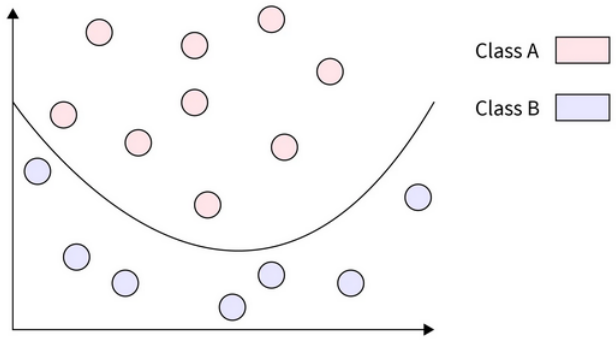


Figure 3.5: Nonlinear SVM

3.2.4 Kernel functions :

The SVM algorithm can be extended to handle non-linearly separable data by using kernel functions.

A kernel function : is a mathematical function that takes two vectors as input and returns a scalar value. The kernel function computes the similarity or distance between the two input vectors in a high-dimensional space. In other words, a kernel function maps the input vectors into a higher-dimensional space where the data may be linearly separable [41].

The most commonly used kernel functions in SVMs are:

1. **Linear Kernel:** This kernel function is used when the data is linearly separable. It computes the dot product of the input vectors in the original feature space..

$$K(x, x') = x^T x'$$

2. **Polynomial Kernel:** This kernel function is used when the data has a polynomial structure. It maps the input vectors into a higher-dimensional space and computes the dot product of the transformed vectors.

   The polynomial kernel function is defined as:

   $$K(x_i, x) = (\gamma * (x_i^T x) + r)^d$$

   where:

   - $K(x_i, x)$ is the value of the kernel function between the support vector x_i and the new input vector x.
   - γ is a parameter that controls the influence of the dot product term. It determines the similarity measure between the support vectors and the new input vector.
   - x_i and x represent the support vector and the new input vector, respectively.
   - r is an optional parameter known as the coefficient of the independent term, which allows shifting the decision boundary away from the origin.
   - d is the degree of the polynomial, which determines the complexity of the decision boundary. It represents the highest power to which the dot product term is raised.

   The polynomial kernel function calculates the similarity between vectors by taking the dot product and raising it to the power of the degree d. The γ parameter scales the influence of the dot product term, and the coefficient r allows adjusting the intercept or shifting the decision boundary. The choice of γ, r, and d depends on the characteristics of the data and the problem at hand. These parameters need to be appropriately tuned to achieve the best performance of the SVM model. This tuning process is often done through techniques such as cross-validation or grid search.

3. **Gaussian RBF Kernel:** This kernel function is used when the data has a non-linear structure. It maps the input vectors into an infinite-dimensional space using a Gaussian distribution and computes the dot product of the transformed vectors.
   The Gaussian RBF kernel function is defined as:

   $$K(x_i, x) = exp(-\gamma * ||x_i - x||^2)$$

   where:

   - $K(x_i, x)$ is the value of the kernel function between the support vector x_i and the new input vector x.

- γ is a parameter that controls the width of the RBF kernel. It determines the influence of each support vector in the decision boundary calculation.
- x_i and x represent the support vector and the new input vector, respectively.
- $||x_i - x||^2$ is the squared Euclidean distance between the support vector x_i and the new input vector x.

The Gaussian RBF kernel measures the similarity or distance between vectors based on their Euclidean distance. It computes the exponential of the negative squared Euclidean distance, resulting in a similarity measure between 0 and 1. The γ parameter controls the width of the kernel, affecting how rapidly the similarity decreases with distance. Higher values of γ lead to narrower and more localized decision boundaries, while lower values result in smoother and more global decision boundaries.
Determining the optimal value of γ is often done through techniques like cross-validation or grid search, where different values are evaluated on a validation set to select the best-performing parameter.

3.3 Swarm intelligence(essaim)

Swarm Intelligence is a branch of artificial intelligence that studies collective behavior in decentralized and self-organized systems. It has been used to solve complex problems that cannot be easily solved by traditional algorithms. The "essaim" algorithm is a specific example of a Swarm Intelligence algorithm that has been applied to various domains, including image analysis and natural language processing .[42]

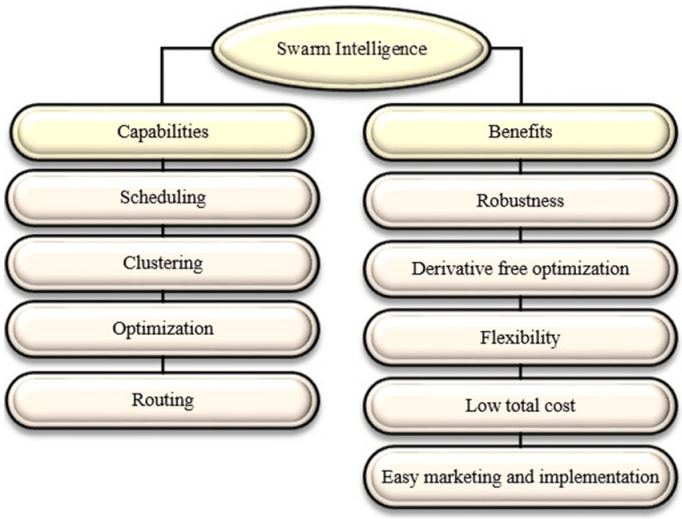


Figure 3.6: capabilities and benefits of swarm intelligence

3.3.1 Fundamentals of Swarm Intelligence:

Swarm intelligence is a computational paradigm inspired by the collective behavior of social insect colonies and other natural systems. It involves the study of how simple agents, through local interactions and decentralized decision-making, can exhibit intelligent behavior as a group. The following are the key fundamentals of swarm intelligence [43]:

1. **Emergecnt Behavior:** Collective behavior arising from interactions among individual agents.
2. **Self-Organization:** Coordination without central control or external guidance.
3. **Decentralization:** Decisions based on local information and interactions.
4. **Communication:** Sharing knowledge and coordinating actions among agents.
5. **Adaptability:** Adjusting to changing environments and robustness to failures.
6. **Exploration and Exploitation:** Balancing search for new solutions and leveraging known ones.
7. **Scalability and Flexibility:** Handling large-scale problems and dynamic environments.

These principles enable the development of swarm intelligence algorithms for diverse problem domains, offering powerful solutions for real-world challenges

3.3.2 Swarm Intelligence Algorithms

Swarm intelligence algorithms are computational methods that mimic the collective behavior and problem-solving capabilities of swarms in nature. These algorithms are designed to tackle complex problems by utilizing the principles of self-organization, decentralized decision-making, and local interactions. some popular swarm intelligence algorithms:

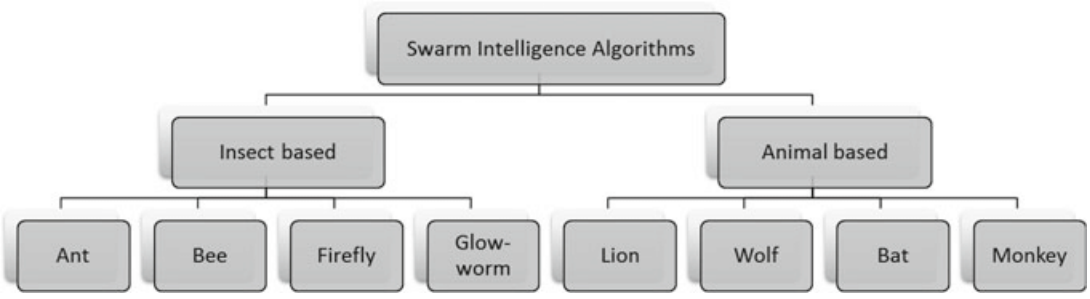


Figure 3.7: Hierarchy of swarm intelligence based algorithms as adopted in the present study

1. **Ant Colony Optimization (ACO):** ACO is inspired by the foraging behavior of ants. It uses the concept of pheromone trails to find optimal paths in combinatorial optimization problems. Ants deposit pheromones to communicate and reinforce paths based on the quality of solutions found. ACO has been successfully applied to problems such as the traveling salesman problem and vehicle routing problems.[44]

2. **Particle Swarm Optimization (PSO):** PSO is inspired by the flocking behavior of birds. It consists of a swarm of particles that move through the problem space, searching for optimal solutions. Each particle adjusts its movement based on its own best-known position and the best-known position of its neighbors. PSO has been widely used in optimization problems, function optimization, and parameter tuning.

3. **Firefly Algorithm (FA):** The Firefly Algorithm is based on the flashing patterns of fireflies. Fireflies use their bioluminescent signals to attract and communicate with each other. In the algorithm, each firefly represents a potential solution, and their attractiveness is determined by the objective function. Fireflies move towards more attractive ones, simulating the optimization process. FA has been applied to various optimization problems and function optimization tasks.[45]

4. **Bacterial Foraging Optimization (BFO):** BFO algorithm mimics the foraging behavior of bacteria to solve optimization problems. It models the chemotaxis, reproduction, and elimination-dispersal processes of bacteria. BFO has been successfully applied

to optimization problems, including function optimization, parameter estimation, and clustering.[46]

5. **Artificial Bee Colony (ABC) Algorithm:** The ABC algorithm is inspired by the foraging behavior of honey bees. It simulates the employed bees, onlooker bees, and scout bees to explore the search space and find optimal solutions. The employed bees exploit the known solutions, while the onlooker bees perform a probabilistic selection process. The scout bees search for new solutions outside the current search space. ABC has been applied to various optimization problems, including function optimization and parameter tuning.

6. **Fish School Search (FSS):** FSS algorithm takes inspiration from the collective behavior of fish schools. It simulates the individual fish movement and their interactions. Each fish represents a potential solution, and they adjust their positions based on their own experience and the influence of neighboring fish. FSS has been utilized in optimization problems, clustering, and data classification [47].

3.3.3 Application Domains of Swarm Intelligence

Swarm intelligence algorithms have found applications in various domains due to their ability to solve complex problems and their adaptability to dynamic environments. Here are some application domains where swarm intelligence has been successfully utilized:[48]

- **Optimization Problems:** Swarm intelligence algorithms have been widely applied to optimization problems, including combinatorial optimization, continuous optimization, and multi-objective optimization. Examples include solving the traveling salesman problem, vehicle routing problems, job scheduling, resource allocation, and portfolio optimization.[49]

- **Robotics and Autonomous Systems:** Swarm intelligence techniques are used in the field of robotics and autonomous systems to enhance coordination, cooperation, and decision-making among multiple robots or agents. Swarm robotics explores the collective behaviors of robot swarms to accomplish tasks such as exploration, area coverage, formation control, and object retrieval.

- **Image and Data Clustering:** Swarm intelligence algorithms are utilized in image and data clustering tasks, where the objective is to group similar data points or pat-

terns together. Swarm-based clustering techniques enable efficient and effective data partitioning, providing insights into patterns and structures in large datasets.

- **Wireless Sensor Networks:** Swarm intelligence is applied in the domain of wireless sensor networks (WSNs) to optimize network performance and energy efficiency. Swarm-based algorithms help in tasks such as node localization, routing, data aggregation, and coverage optimization, enabling effective utilization of limited resources in WSNs.

- **Traffic and Transportation Systems:** Swarm intelligence algorithms are employed to optimize traffic flow, improve transportation efficiency, and reduce congestion in urban environments. They can assist in traffic signal control, route planning, traffic flow modeling, and vehicle scheduling, leading to enhanced traffic management and reduced travel time.

- **Financial and Economic Modeling:** Swarm intelligence techniques have been applied in financial and economic modeling to predict market trends, optimize investment portfolios, and simulate economic behaviors. These algorithms assist in decision-making, risk analysis, and portfolio management by considering multiple variables and potential scenarios.

- **Social Network Analysis:** Swarm intelligence approaches are used in social network analysis to understand the dynamics, behavior, and influence within social networks. They aid in tasks such as community detection, information diffusion modeling, opinion dynamics, and recommendation systems.

- **Bioinformatics and Computational Biology:** Swarm intelligence algorithms have been employed in bioinformatics and computational biology to solve problems related to gene expression analysis, protein structure prediction, sequence alignment, and molecular docking. These techniques assist in understanding biological systems and complex biological processes.

These application domains highlight the versatility and effectiveness of swarm intelligence algorithms in solving a wide range of problems across different fields. The ability to handle complex, dynamic, and real-world scenarios makes swarm intelligence a valuable approach for addressing challenging tasks in various domains.

3.4 Particle Swarm Optimization (PSO): A Swarm Intelligence Algorithm for Optimization Problems

PSO is a population-based optimization algorithm that is inspired by the collective behavior of social organisms such as flocks of birds or schools of fish. PSO has been used to solve a wide range of optimization problems in various domains, including engineering, finance, and healthcare [50].
(PSO) mimics the collective behavior of bird flocks to solve optimization problems. This section provides an in-depth exploration of PSO as a powerful optimization technique. It covers the fundamental concepts of PSO, including the representation of solutions as particles, the concept of fitness evaluation, and the update rules for particle movement. Additionally, the section discusses the key components of PSO, such as the initialization, velocity update, and position update mechanisms. It also highlights the influence of parameters, such as cognitive and social coefficients, on the behavior of the algorithm. Furthermore, the section presents various variants and enhancements of PSO, including adaptive PSO, hybrid PSO,[51] and multi-objective PSO. Real-world applications of PSO in different domains, such as engineering, data mining, and image processing, are also discussed. Finally, the section concludes with an overview of current research trends, challenges, and future directions in PSO.

3.4.1 Fundamentals of Particle Swarm Optimization (PSO):

Particle Swarm Optimization (PSO) is a population-based metaheuristic algorithm that mimics the collective behavior of bird flocks or fish schools to solve optimization problems. PSO operates by iteratively updating a population of particles in the search space to find optimal or near-optimal solutions. the key fundamentals of PSO[52]:

- **Representation of Solutions as Particles:** PSO represents potential solutions to the optimization problem as particles in the search space. Each particle is associated with a position vector, which corresponds to a candidate solution, and a velocity vector,
- **Fitness Evaluation:** The fitness function evaluates the quality or objective value of a particle's position in the search space. The fitness function guides the search process by providing a measure of the particles' performance.
- **Particle Movement and Swarm Behavior:** In PSO, particles adjust their positions

and velocities based on their own experience and the collective information of the swarm. The movement of each particle is influenced by its previous best-known position (personal best) and the best-known position among all particles in the swarm (global best)

- **Local and Global Best Solutions:** Each particle maintains its personal best position.the swarm maintains a global best position These best positions guide the movement of particles and influence their search trajectory.

PSO's performance and behavior are influenced by several factors, including the cognitive and social coefficients that control the balance between personal and global information, the inertia weight that controls the particle's momentum, the neighborhood topology that defines the interaction among particles, and the swarm size.

3.4.2 Algorithm Description:

The PSO algorithm operates iteratively, following these fundamental steps:

3.4.2.1 Initialization:

Initialize the population of particles with random positions and velocities within the search space. Set the personal best positions of particles initially equal to their current positions.

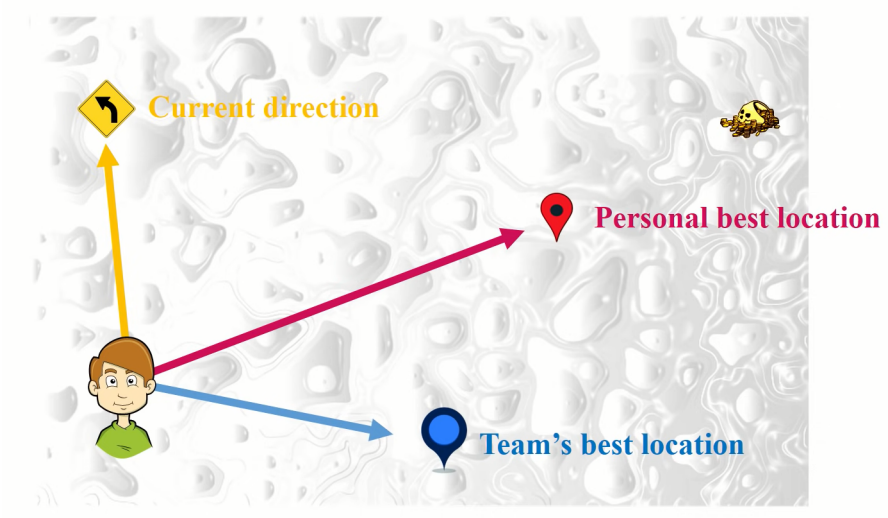


Figure 3.8: PSO Initialization

1. **Initialization of Particle Positions:**

For each particle i, the initial position is randomly generated within the search space. The position vector for particle i is denoted as

$$X_i = [x_i1, x_i2, ..., x_id]$$

where d represents the number of dimensions in the search space.

The position components x_ij for particle i are initialized as: $x_ij = lower_bound_j + (upper_bound_j - lower_bound_j) * rand()$, where lower $_bound_j$ and $upper_bound_j$ represent the lower and upper bounds of the $j-th$ dimension of the search space, and $rand()$ generates a random number between 0 and 1.

2. **Initialization of Particle Velocities:** For each particle i, the initial velocity is randomly generated within a specified range. The velocity vector for particle i is denoted as

   $$V_i = [v_i1, v_i2, ..., v_id].$$

   The velocity components v_ij for particle i are initialized as:

   $$v_ij = (v_max_j - v_min_j) * rand() + v_min_j$$

   where v_max_j and v_min_j represent the maximum and minimum velocities allowed in the $j-th$ dimension, respectively.[53]

3. **Set Personal Best Positions:** Initially, the personal best position of each particle i is set equal to its initial position X_i.

The random initialization of positions and velocities ensures the exploration of the search space by the particles, allowing them to start the optimization process from different regions. The specific range of random values used for initialization, such as the bounds of the search space and velocity limits, can be customized based on the characteristics of the problem being solved.
It's important to note that different variations of PSO may introduce additional initialization techniques or modifications to enhance exploration and exploitation during the search process.

3.4.2.2 Velocity Update:

Update the velocity of each particle based on its current velocity, personal best position, and the global best position. The velocity update equation contains cognitive and social components that control the particle's movement towards better solutions.

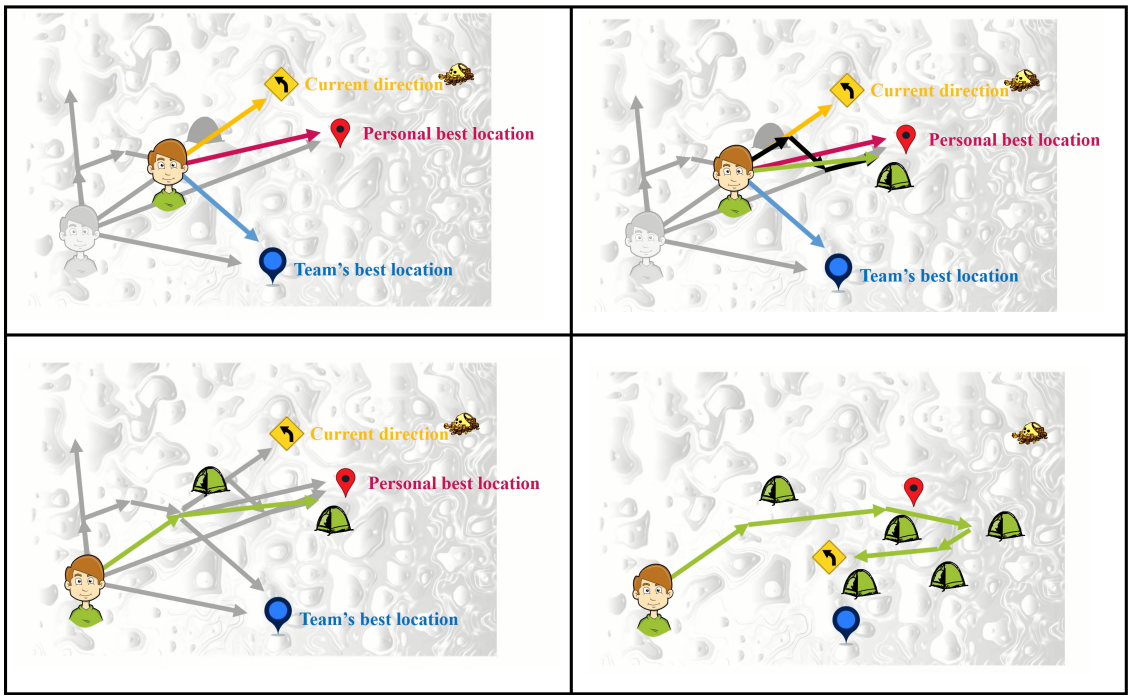


Figure 3.9: PSO Update

For each particle i:

1. **Update the particle's velocity components in each dimension j:**

   $$V_ij(t+1) = w * V_ij(t) + c1 * rand1 * (P_best_ij - X_ij(t)) + c2 * rand2 * (G_best_j - X_ij(t))$$

   where:

   $V_ij(t+1)$ is the updated velocity component of particle i in dimension j at iteration t+1.

   $V_ij(t)$ is the current velocity component of particle i in dimension j at iteration t.

   w is the inertia weight, controlling the particle's momentum and balancing exploration and exploitation. It determines the influence of the particle's previous velocity on its updated velocity.

   c1 and c2 are the cognitive and social coefficients,rand1 and rand2 are random numbers between 0 and 1 [53] .

2. **Update the particle's position components in each dimension j:**

   $$X_ij(t+1) = X_ij(t) + V_ij(t+1),$$

   where :

   $X_ij(t+1)$ is the updated position component of particle i in dimension j at iteration t+1, and $X_ij(t)$ is the current position component at iteration t.

The velocity update equation combines three components: the particle's previous velocity, the attraction towards its personal best position, and the attraction towards the global best position of the swarm. The cognitive coefficient (c1) controls the influence of the personal best position, and the social coefficient (c2) controls the influence of the global best position [53].

The inertia weight (w) affects the trade-off between exploration and exploitation. Higher values of the inertia weight allow the particle to maintain a higher momentum, promoting exploration and the search for new solutions. Lower values of the inertia weight facilitate exploitation by reducing the momentum and focusing on exploiting the already discovered promising regions.

3.4.2.3 Position Update:

Update the position of each particle by adding its velocity to its current position. This movement represents the exploration and exploitation of the search space.

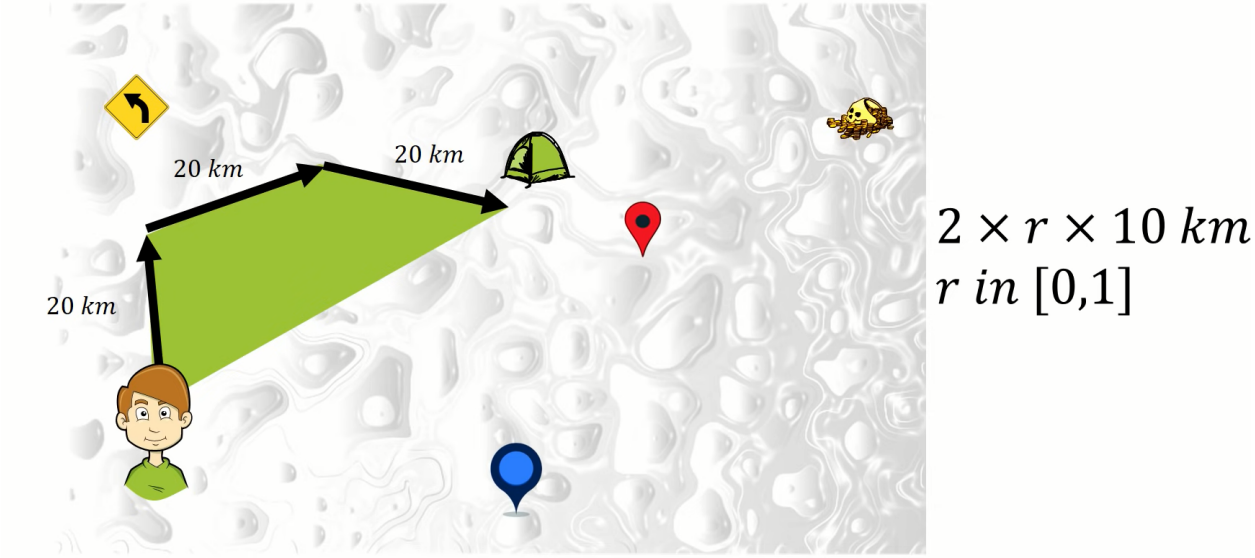


Figure 3.10: PSO position update

For each particle i:

$$X_ij(t+1) = X_ij(t) + V_ij(t+1),$$

where :

$X_ij(t+1)$ is the updated position component of particle i in dimension j at iteration t+1.

$X_ij(t)$ is the current position component of particle i in dimension j at iteration t.

$V_ij(t+1)$ is the updated velocity component of particle i in dimension j at iteration t+1. The position update equation simply adds the corresponding velocity component to the current position component of each particle in each dimension. This update represents the movement of particles in the search space based on their velocities.

3.4.2.4 Fitness Evaluation:

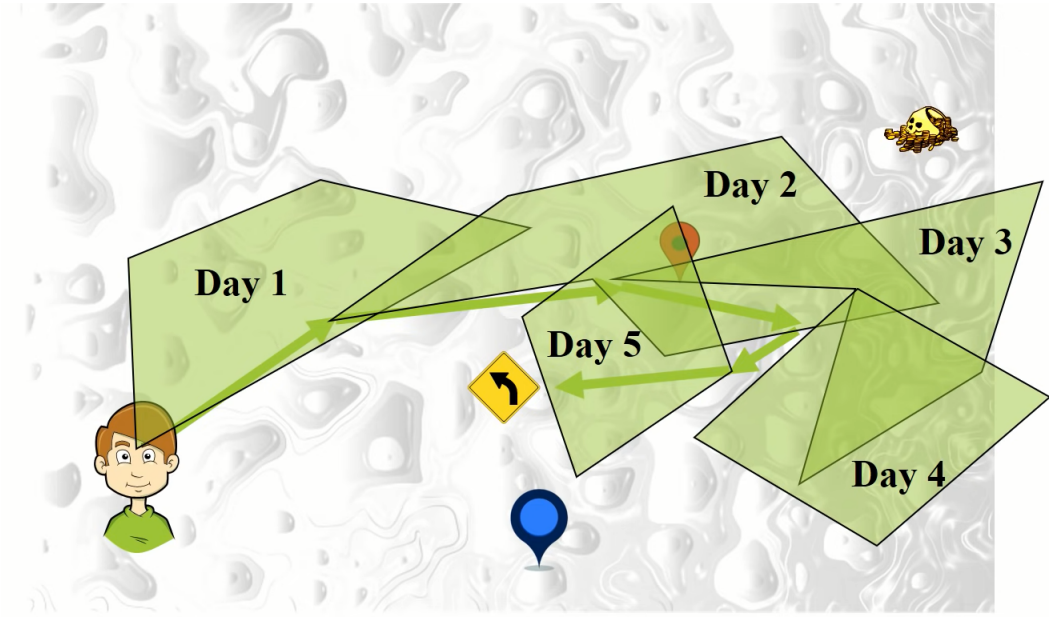


Figure 3.11: PSO keep track with location to measure fitness

Evaluate the fitness or objective value of each particle's new position using the fitness function.
The fitness evaluation step provides a quantitative measure of how well each particle is performing in the search space. It guides the search process by providing information on the quality of each particle's position. Based on the fitness values, particles adjust their velocities and positions in subsequent iterations, aiming to improve their performance.

3.4.2.5 Update Personal and Global Best:

Update the personal best positions of particles if their new positions have better fitness values. Update the global best position if any particle has found a better solution than the current global best.
the personal best and global best positions are updated based on the fitness values of the particles

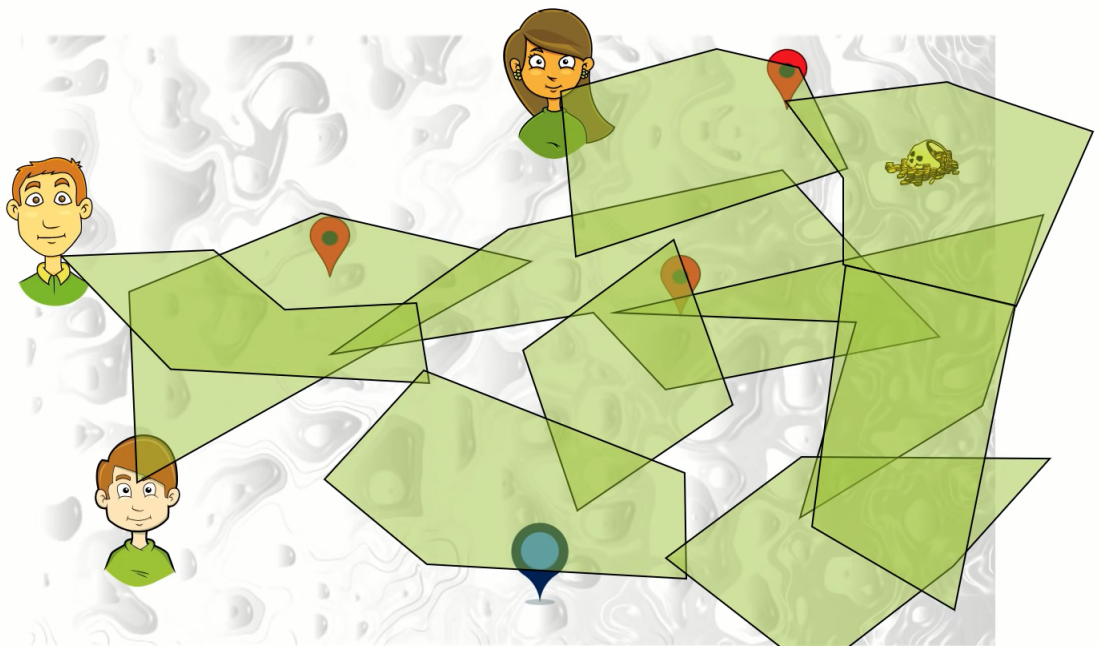


Figure 3.12: PSO Personal.best and Swarm.best(team.best)

- **Personal Best Update:**
  For each particle i, compare its current fitness value with its personal best fitness value ($PbestFitness_i$).
  If the current fitness value is better (either lower or higher, depending on the optimization problem), update the personal best fitness value and the personal best position ($PbestPosition_i$).

- **Global Best Update:**
  Compare the personal best fitness values of all particles in the swarm with the fitness value of the current global best ($GbestFitness$).
  If any particle has a better personal best fitness value than the current global best, update the global best fitness value and the global best position ($GbestPosition$).
  The global best position represents the best position found by any particle in the entire swarm.

The personal best update is performed on an individual particle level. Each particle compares its current fitness value with its personal best fitness value and updates its personal best position if necessary. This allows particles to track and remember their own best positions throughout the search process[53].

The global best update, on the other hand, considers the personal best fitness values of all particles in the swarm. It compares these values with the current global best fitness value and updates the global best position if any particle has found a better solution. The global best position represents the best solution found by any particle in the entire swarm.

By updating the personal and global best positions, PSO allows particles to share and incorporate knowledge about the best solutions found so far. This cooperative behavior facilitates exploration and exploitation of the search space, helping the swarm converge towards optimal or near-optimal solutions for the given optimization problem.
important to note the personal and global best positions are used in the velocity update step to guide the particles' movement towards more promising regions of the search space.

3.4.2.6 Termination Criteria:

Check if the termination criteria are met to stop the algorithm.
Termination criteria can include a maximum number of iterations, reaching a desired fitness threshold, or a predefined computation budget.
common termination criteria used in PSO:

- **Maximum Iterations:**
  Set a maximum number of iterations or generations that the algorithm will run.[54]
  Terminate the algorithm when the specified number of iterations is reached[50].
  This criterion ensures that the algorithm does not run indefinitely and has a fixed termination point.

- **Target Fitness Value:**
  Define a target fitness value that represents the desired quality of the solution.
  Terminate the algorithm when any particle in the swarm reaches or exceeds the target fitness value[55].

- **Convergence Criteria:** Monitor the convergence behavior of the swarm by tracking the changes in the global best fitness value over consecutive iterations.
  Define a convergence threshold or tolerance value.[56]
  Terminate the algorithm if the improvement in the global best fitness value falls below the convergence threshold for a certain number of iterations or remains unchanged.

- **Time Limit:**
  Set a maximum allowed execution time for the algorithm.
  Terminate the algorithm if the specified time limit is exceeded.
  This criterion ensures that the algorithm does not run for an excessively long duration[57].

The choice of termination criteria depends on the problem being solved and the specific requirements of the application. It is important to strike a balance between allowing the algorithm enough time to explore the search space and avoiding excessive computation time.

3.4.3 Variants and Enhancements of PSO:

Over time, Particle Swarm Optimization (PSO) has undergone various modifications and improvements to address specific challenges and enhance its performance in different scenarios. Here are several notable variations and enhancements of PSO:

Adaptive PSO:

Adaptive PSO dynamically adjusts the cognitive and social coefficients (c1 and c2) during the optimization process. The coefficients are updated based on the behavior of the swarm and the progress made in the search. This adaptive mechanism allows the algorithm to effectively balance exploration and exploitation.

Cooperative PSO (CPSO):

CPSO involves dividing the swarm into subgroups or sub-swarms, each having its own local best and global best.

Particles in different sub-swarms cooperate by sharing information periodically. CPSO enhances the exploration capabilities of the algorithm by encouraging diversity among sub-swarms. [58]

Hybrid PSO: Hybrid PSO combines the PSO algorithm with other optimization techniques or algorithms. Common hybridizations include integrating PSO with local search algorithms, evolutionary algorithms, or other metaheuristic techniques.

The goal of hybrid PSO is to leverage the strengths of different algorithms to improve solution quality and search efficiency [59].

Multi-Objective PSO (MOPSO):

MOPSO extends PSO to handle multi-objective optimization problems.

Techniques like Pareto dominance and non-dominated sorting are used to maintain a diverse set of optimal solutions (Pareto front).[60]

MOPSO enables simultaneous optimization of multiple conflicting objectives.

Quantum-behaved PSO (QPSO): QPSO incorporates concepts from quantum mechanics to enhance PSO.
It replaces classical position and velocity components with quantum-inspired counterparts. QPSO aims to improve exploration and overcome premature convergence in the search process .
Self-Adaptive PSO:
Self-adaptive PSO includes mechanisms to automatically adjust parameters during the optimization process.
Parameters such as inertia weight, cognitive and social coefficients, and neighborhood topologies are adapted based on the swarm's performance. The objective of self-adaptive PSO is to optimize the algorithm's parameters and adapt to changing problem landscapes[61].

3.4.4 Applications of PSO :

PSO has demonstrated successful application in diverse domains for optimizing a wide range of problems. Here are several noteworthy examples of PSO's applications:

- **Function Optimization:**
  PSO can be used to optimize mathematical functions by finding the global minimum or maximum. It has been applied to optimize functions in fields such as engineering, finance, operations research, and data science. [56]

- **Engineering Design and Parameter Tuning:**
  PSO is used for engineering design problems, including optimal design of structures, circuits, and systems.
  It is also employed for parameter tuning in machine learning algorithms, such as optimizing the hyperparameters of neural networks.

- **Feature Selection and Dimensionality Reduction:**
  PSO helps in selecting a subset of relevant features from high-dimensional data. It aids in reducing the dimensionality of data while preserving important information for classification or regression tasks.

- **Image and Signal Processing:**

PSO is utilized in image reconstruction, image denoising, image segmentation, and object tracking.
It is applied to optimize the parameters of image and signal processing algorithms for improved performance. [62]

- **Data Clustering:**
  PSO is employed for data clustering to group similar data points together.
  It helps in identifying patterns and structures in datasets and is used in areas such as customer segmentation, bioinformatics, and pattern recognition.

- **Neural Network Training:**
  PSO has been applied to train the weights and biases of neural networks.
  It aids in optimizing the parameters of neural networks for improved accuracy and generalization.

- **Portfolio Optimization:**
  PSO is used for financial portfolio optimization to find an optimal allocation of assets that maximizes returns and minimizes risks.
  It helps in constructing well-balanced investment portfolios based on various criteria.

- **Energy Management:**
  PSO is employed for optimizing energy consumption in smart grids, energy-efficient buildings, and renewable energy systems.
  It aids in finding optimal schedules and control strategies to minimize energy costs and improve efficiency. [63]

- **Vehicle Routing and Optimization:**
  PSO is applied to optimize vehicle routing problems, such as route planning, fleet management, and logistics optimization.
  It helps in minimizing travel distances, reducing fuel consumption, and improving transportation efficiency.

These are just a few examples of the diverse applications of PSO. The versatility of PSO allows it to be applied to various optimization problems across different industries and research domains. Its ability to handle both continuous and discrete variables, as well as its simple implementation, makes it a popular choice for many optimization tasks.

3.4.5 Challenges and Future Directions :

Challenges and future directions in PSO encompass various aspects. Key challenges include premature convergence, scalability, handling constraints and multiple objectives, dynamic and uncertain environments, incorporating domain-specific knowledge, hybridization with other algorithms, real-world applications, and explainability. Future research can focus on strategies such as adaptive parameter tuning, parallel implementations, constraint handling mechanisms, adaptive algorithms for dynamic environments, utilizing problem-specific knowledge, effective hybridization, benchmarking on real-world problems, enhancing interpretability, and visualization techniques. Addressing these challenges and exploring future directions will enhance the capabilities and applicability of PSO.

3.4.6 summary about PSO:

Particle Swarm Optimization (PSO) is a widely-used metaheuristic algorithm with applications across diverse domains. It adapts swarm behavior to solve optimization problems effectively. PSO's variants and enhancements address specific challenges. Despite its effectiveness, ongoing research aims to overcome challenges and explore new directions for improvement. PSO's simplicity and versatility make it a valuable tool for solving complex optimization problems.

3.5 Conclusion

The chapter provides a comprehensive overview of Support Vector Machines (SVM), Swarm Intelligence, and Particle Swarm Optimization (PSO). It covers the concepts, applications, and challenges of these techniques. SVM is discussed for classification and regression, Swarm Intelligence explores collective behavior, and PSO is explained as a metaheuristic algorithm. The chapter emphasizes their versatility and provides insights for future improvements. Overall, it serves as a valuable resource for researchers, practitioners, and students in the field of intelligent optimization.

Chapter 4

Realisation

4.1 Introduction

In order to take advantage of the bibliographical study presented in the previous chapters, focused on the learning methods, positive and unlabeled learning (PU learning) and the **Particle Swarm Optimization algorithm (PSO)**, this last chapter is devoted to the description of our efforts in the problem of positive and unlabeled learning.

Our efforts were mainly directed towards proposing a new method of **binarization** of the PSO algorithm so that it adapts to our problem of **PU learning**. This method is used to select the best negative elements from the unlabeled set.
Finally, we used the **SVM** learning method to create the best classification model.

4.2 NBPSO-SVM : Particle Swarm binary search algorithm for PU learning

4.2.1 Explaining the problem

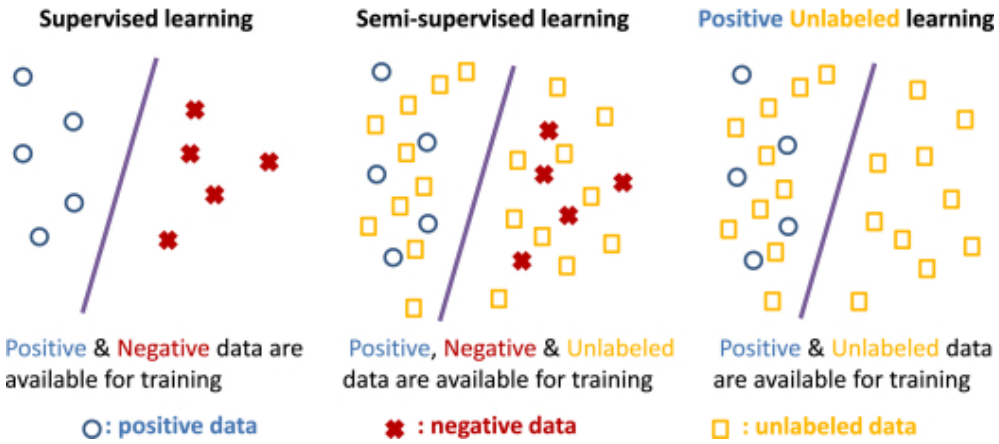


Figure 4.1: comparison between supervised, semi-supervised, PU learning

- **Supervised learning**:
  All examples of training sets are labeled. The training process is performed on positive and negative training examples, the new elements without a known class label are predicted based on the model learned.

- **Semi-supervised**:
  Set of data from a small labeled sample including positive and negative samples, and a large set of unlabeled. labeled samples are used to learn the model parameters.

- **PU learning**:
  There are no negative samples known for the training set, only a small set of labeled positive samples and a large set of unlabeled samples.

4.2.2 Presentation of the Method

As mentioned in the previous chapter, the original Particle Swarm Optimization algorithm (PSO) was primarily designed for continuous optimization problems. Our problem is considered a binary and combinatorial optimization problem. Consequently, certain modifications to the original PSO are necessary to adapt it for PU learning processing.

In this algorithm, each particle in the swarm represents a potential solution of reliable positive instance from the unlabeled set. The forthcoming section will provide detailed explanations regarding this concept.

4.3 Swarm position encoding

Particle Swarm Optimization is a scalable research technique population (Swarm) based.
In order to find the best reliable negative examples in PU learning problems, a new binary version of **PSO** is proposed called **NBPSO-SVM** New Binary PSO-SVM as a pairing algorithm with a classifier based on **SVM**.

NBPSO-SVM uses binary encoding to represent the unlabeled samples. With the following scheme,
the X_i position of each particle will simply be a vector that provides binary values describing the selection as reliable negatives among unlabeled examples with the following scheme:
the X_i position of each particle will simply be a vector of binary values describing the selection as reliable negatives among the unlabeled examples.

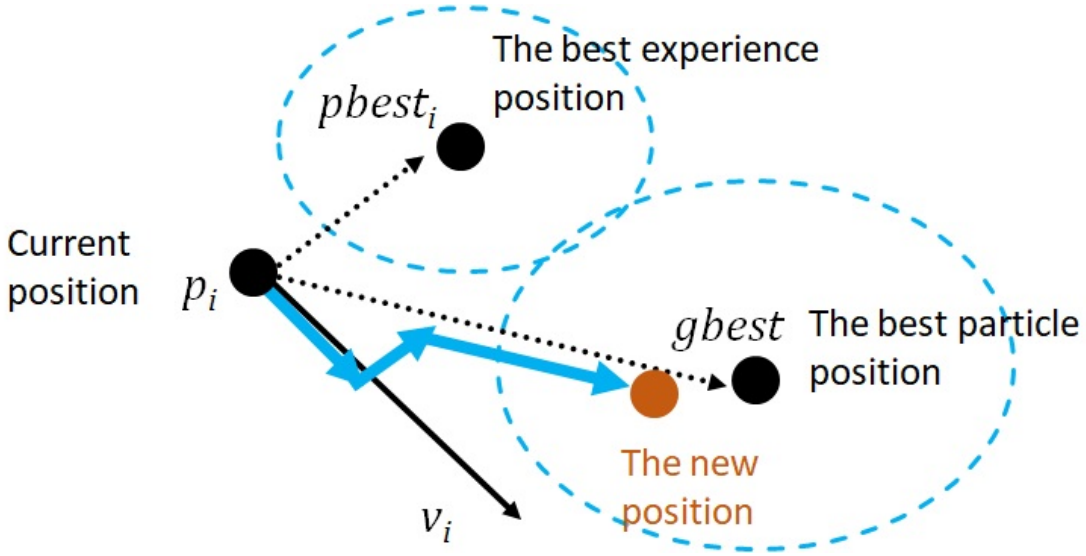


Figure 4.2: how can the velocity, current position, personal-best determine the global best

We represent the swarm as a set of N particles. Each particle is determined by its current position, velocity, and Personal Best Position which are used to calculate fitness. Given U the size of the unlabeled set and $i = 1, \ldots, N$ (N is the number of particles in the swarm). Each particle X_i is represented by a vector of U binary values $x_i^1, \ldots, x_i^U$ with $x_i^j \in \{0, 1\}$

define either the sample x_i^j $(j = 1, \ldots, U)$ is selected as reliable negative (in this case $x_i^j = 1$) or not $(x_i^j = 0)$

- $particles(n) = \{X_1, X_2, \ldots, X_N\}$: a swarm of N particles (Matrix N*U)
- $particle_i$: The Position Of a Particle $X_i = \{x_1, X_2, \ldots, x_U\}$: a binary vector
- $velocities_i$: The velocity Of Each Particle is a vector in the same dimension as a particle
- $pbest_i$: The Best position Of a particle (local)
- $gbest$: The Best position Of The Swarm (Global)
- f: Fitness Function
- t: The Iteration Number

4.3.1 Initialization :

Similar to the flock of birds looking for food, we start a number of random points on the plane (particles), The particles are randomly scattered in the binary search space looking for the best position. After a certain number of iterations, the global best is considered the overall best explored by the swarm.

number of particles = 10

At the first iteration : We have no Global-best-position and no global-best-fitness (g_best_position=[ ] , g_best_fitness= [ ]) , and for each particle the current position is its best-position (Particles[i]=best_Position[i])

```
def BinaryInitialization(num_particles, dim, ub, lb):
    # Initialize particles
    particles = np.zeros((num_particles, dim))
    velocities = np.zeros((num_particles, dim))
    best_fitnesses = []
    global_best_position = None
    global_best_fitness = 0
    best_models = []
    global_best_model = None

    for i in range(num_particles):
        particles[i, :] = (lb + (ub - lb) * np.random.rand(dim)) > 0.5
```

```
        fit, clf = fitness_func(particles[i, :])
        best_fitnesses.append(fit)
        best_models.append(clf)
        if best_fitnesses[i]>global_best_fitness:
            global_best_fitness = best_fitnesses[i]
            global_best_position = particles[i, :].copy()
            global_best_model = best_models[i]
    best_positions = particles.copy()
    return particles, velocities, best_positions ,best_fitnesses,
                                        best_models, global_best_position
                                        , global_best_fitness,
                                        global_best_model
```

After initialization we then calculate the fitness for each particle then compare it . We store the best-positions and the best-fitnesses and the global parameters

4.3.2 Fitness Function Evaluation :

Each particle is considered as solution , we use the selected solutions in our **SVM model classifier** then extract the confusion matrix.

The Confusion Matrix : A confusion matrix is a table that is used to evaluate the performance of a classification model. It summarizes the predictions made by the model and compares them to the actual ground truth values.

.	Predicted	
.	Positive	Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

- True Pxositives (TP): Instances that are correctly predicted as positive.
- False Positives (FP): Instances that are incorrectly predicted as positive.
- False Negatives (FN): Instances that are incorrectly predicted as negative.
- True Negatives (TN): Instances that are correctly predicted as negative.

Confusion Matrix allows for the calculation of various evaluation metrics such as accuracy, precision, recall (sensitivity), specificity, and F1-score. These metrics provide a comprehen-

sive understanding of the model's performance and its ability to correctly classify instances into different classes.

```
def fitness_func(solution, it=None, agt=None):
    if np.sum(solution) == 0:  # no feature is selected
        Fitness = 0
        return Fitness
    global iU
    solution = solution.astype(bool)
    ib = iU[solution == 1]
    Xb, yb, yb_orig = selection(ib, X, y, y_orig)


    clf = SVC(gamma='scale', probability=True,class_weight='balanced')
    clf.fit(Xb, yb)
    y_pred = clf.predict(Xb)

    tn, fp, fn, tp = confusion_matrix(yb,y_pred).ravel()
    tpr = tp/(tp+fn)
    Fitness = tpr
    return Fitness, clf
```

Note : $TPR = TP/(TP + FN)$ Stands for TPR (True Positive Rate), also known as **sensitivity or recall**

A higher TPR indicates that the model has a lower rate of false negatives and is better at identifying positive instances.
This how we measure the fitness of each particle.

4.3.3 Update position:

To update the position of a particle, we have to adapt the classic PSO to the new type of data that we want to treat in this case the binary vectors. Therefore, we introduced new operations that replace the ancient ones in the update position equation. This is considered as our major contribution to this problem.
The velocity of the particle is first updated in Eq 4.1, and then the new position is computed based on the updated velocity in Eq. 4.2.
The mathematical representation of the position update equation:

$$velocity_i^{t+1} = \omega \otimes velocity_i^t \oplus c_1 \cdot r_1 \otimes (pbest_i \ominus X_i^t) \oplus c_2 \cdot r_2 \otimes (gbest_d \ominus X_i^t) \tag{4.1}$$

$$Particles[i] = Particles[i] \oplus velocities[i] \tag{4.2}$$

In Eq. 4.1 and Eq. 4.2 we have introduced three new operators, namely $\ominus$, $\oplus$, $\otimes$ to deal with binary vectors.

4.3.4 Subtract Operator $\ominus$:

In the continuous version of PSO, this operator calculates the difference between the best position (global and local) and the current position of the ith particle. In this new binary version of PSO we propose the following mechanism to implement this operator.

We used the subtraction between two binary vectors to generate an intermediate solution. This operator is defined in Eq. 4.3 and Eq. 4.4 In which, we use DX_i to show the subtraction between the best position of a particle ($pbest_i^t$ for instance) and the current position of the particle X_i^t.

$$DX_i = pbest_i^t \ominus X_i^t \tag{4.3}$$

$$DX_{ij}^t = \begin{cases} 0 \text{ or } 1 \text{ randomly} & \text{if } pbest_{ij}^t = 0 \text{ and } X_{ij}^t = 1 \\ pbest_{ij}^t - X_{ij}^t & \text{otherwise} \end{cases} \tag{4.4}$$

The implementation code in python is below

```
def Bin_Sub_op(M, X):
    # binary subtraction
    DX = np.zeros(M.size, dtype=int)
    for i in range(M.size):
        if M[i] == 0 and X[i] == 1:
            DX[i] = np.random.randint(2, size=1)[0]
        else:
            DX[i] = M[i] - X[i]

    return DX
```

4.3.5 Addition operator $\oplus$:

In the same way as the previous operator, This operator is introduced to calculate the addition of two binary vectors based on Eq. 4.5 and Eq. 4.6

$$AX_i = V1_i^t \oplus V2_i^t \tag{4.5}$$

$$AX_{ij}^t = \begin{cases} 0 \text{ or } 1 \text{ randomly} & \text{if } V1_{ij}^t = V2_{ij}^t = 1 \\ V1_{ij}^t + V2_{ij}^t & \text{otherwise} \end{cases} \tag{4.6}$$

The implementation code in Python is below

```
def Bin_Add_op(op1, op2):
    # binary addition
    M = np.zeros(op2.size, dtype=int)
    for i in range(op2.size):
        if op1[i] == op2[i] and op2[i] == 1:
            M[i] = np.random.randint(2, size=1)[0]
        else:
            M[i] = op1[i] + op2[i]

    return M
```

4.3.6 Multiply operator $\otimes$:

To binarize this operator, the multiplication operator calculates the product of a binary vector by a scalar (a real number between 0 and 1). To keep the same philosophy of the original method, so depending on the value of the scalar, a set of 1s picked randomly among the 1s of the vector that will be replaced by 0s. Therefore we have 3 cases:

- if $scalar = 0$, then the product returns a null vector.
- if $scalar = 1$, then the product returns the same vector.
- if $scalar \in]0, 1[$, a set of 1s are replaced by 0s according to the scalar value.

The implementation code in Python is below

```
def mult_scalar(Scalar, V):

    nb1 = V[V == 1].size
```

```
    stp = round(nb1*(1-Scalar)) # nbr of 1 a changes randomly

    indx1 = np.argwhere(V == 1)
    randindx1 = np.random.choice( np.random.permutation(indx1.size), stp)

    V[randindx1] = 0
    P = V

    return P
```

The mult_scalar function takes an input vector V and a scalar value Scalar. It randomly changes a portion of 1s in V to 0s based on the calculated number of changes determined by Scalar. The function then returns the modified vector P.

Finaly, the new positions of the particles are updated as we said previously using Eq 4.1 and Eq 4.2 with the code python below:

```
    # Update velocity
    velocity = Bin_Add_op(Bin_Add_op(mult_scalar(inertia, velocity) ,
          mult_scalar(c1 * random(),Bin_Sub_op(best_position, particle))),
          mult_scalar(c2 * random(),Bin_Sub_op(global_best_position,
                                               particle)))
    # Update position
    particle = Bin_Add_op(particle , velocity)
```

4.3.7 Update Global-best and Personal-Best :

Before updating the Global-best and Personal-Best we must first evaluate the fitness then compare it to its previous value .

```
# Evaluate fitness
fitness , model= fitness_func(particles[i])
```

1. **Personal Best Update:** During each iteration, the particle compares its current fitness value with its personal_best_fitness.
   If the current fitness value is better than personal_best_fitness, the particle updates its personal_best_fitness to the current fitness value and sets its personal_best position to its current position.

   ```
   # Update personal best
   if fitness > best_fitnesses[i]:
   ```

```
    best_positions[i] = particles[i]
    best_fitnesses[i] = fitness
    best_models[i] = model
```

2. **Global Best Update:** After updating the personal_best for each particle, the algorithm identifies the particle with the best personal_best_fitness among the entire swarm.
   The position associated with this best personal_best_fitness is considered as the global_best position, and the personal_best_fitness becomes the global_best_fitness.

```
# Update global best
if fitness > global_best_fitness:
    global_best_fitness = fitness
    global_best_position = particles[i]
    global_best_model = model
```

Notice that we store the best classifier models because our main goal is to find the Best possible model

4.4 Qualitative Data (Categorical data) :

In the field of research and data analysis, two primary types of data are commonly used: qualitative data and quantitative data. Understanding the characteristics and differences between these two types of data is essential for conducting effective research and drawing accurate conclusions.

4.4.1 Qualitative Data:

Qualitative data refers to non-numerical data that is descriptive in nature. It focuses on qualities, characteristics, opinions, perceptions, and subjective information. Qualitative data is obtained through methods such as interviews, observations, surveys, focus groups, and open-ended questionnaires. It provides rich and detailed information, allowing researchers to explore and understand the underlying meanings and context of the data. Qualitative data analysis involves identifying themes, patterns, and categories within the data, often through coding and categorization techniques. This type of data is particularly valuable in social sciences, anthropology, psychology, and other fields where subjective insights and

interpretations play a significant role.
Qualitative Data Examples:

- Gender: Male, Female, Other
- Marital Status: Single, Married, Divorced, Widowed
- Education Level: High School, Bachelor's Degree, Master's Degree, Ph.D.
- Eye Color: Blue, Brown, Green, Hazel
- Occupation: Teacher, Engineer, Doctor, Lawyer

4.4.2 Quantitative Data:

Quantitative data, on the other hand, refers to numerical data that can be measured and analyzed using statistical methods. It deals with quantities, measurements, and objective information. Quantitative data is collected through methods such as surveys, experiments, measurements, and structured questionnaires. It allows researchers to quantify relationships, compare variables, and draw statistical inferences. Quantitative data analysis involves statistical techniques such as mean, median, standard deviation, regression analysis, and hypothesis testing. This type of data is commonly used in fields such as economics, physics, biology, and market research, where numerical measurements and statistical analysis are crucial. Qualitative Data Examples:

- Height: Numerical measurements of individuals' height in inches or centimeters.
- Temperature: Recorded temperatures in degrees Celsius or Fahrenheit.
- Test scores: Numerical values representing the scores achieved by students on an exam.
- Sales data: Quantitative information about the number of products sold, revenue generated, or customer purchases.
- Stock prices: Numerical values representing the prices of stocks at different time points.

4.4.3 Differences and Complementary Nature:

Qualitative and quantitative data differ in their nature, collection methods, and analytical approaches. While qualitative data provides detailed descriptions and subjective insights,

quantitative data focuses on numerical measurements and objective analysis. These two types of data are often seen as complementary, as they can be used together to gain a more comprehensive understanding of a research topic. The integration of qualitative and quantitative data, known as mixed methods research, allows researchers to triangulate findings, validate results, and provide a deeper understanding of complex phenomena.

4.5 Analysing the Data-sets

Our method has been applied to multiple data-sets, where we applied the **BPSO-SVM** method to each dataset. Subsequently, we assessed the performance of our approach by evaluating the performance using the commonly known metrics in machine learning, namely precision, recall, and F1-score. In order to optimize the results, we fine-tuned the hyperparameters to obtain the best possible outcomes.

4.5.1 Listing and describing the Data-sets

We use those well-chosen Data sets : audiology, breast-cancer, dermatology, hepatitis, lymph, pima, soybean, vote, chess, nursery .

This is a quick description of the used data sets. Since this is not our main goal we are not going into details

Data-set	**Features**	**% of pos**	**Nbrof Pos**	**Nbr of Unl**	**Nbr Test**
Audiology	69	30%	15	79	11
		40%	20	74	11
		50%	25	68	11
Breast-cancer	9	30%	54	203	29
		40%	72	185	29
		50%	91	166	29
Dermatology	34	30%	30	136	19
		40%	40	126	19
		50%	50	116	19
Hepatitis	19	30%	9	130	16
		40%	12	127	16
		50%	15	124	16
Lymph	18	30%	17	11	14
		40%	22	106	14
		50%	28	100	14
Soybean	35	30%	25	140	18
		40%	33	132	18
		50%	42	123	18
Pima	8	30%	135	556	77
		40%	180	511	77
		50%	225	466	77
Vote	16	30%	72	319	44
		40%	96	295	44
		50%	120	271	44
Chess	36	30%	451	2425	320
		40%	601	2275	320
		50%	751	2125	320
Nursery	8	30%	1166	6562	859
		40%	1555	6173	859
		50%	1944	5784	859

Table 4.1: Data-sets listing

data-sets are split into Train / Test Set :

1. **K-fold cross-validation** : The data-sets are divided into k equal-sized folds = **10 folds**

2. **Precision / Recall :** in order to calculate the precision and recall we split the data based on precision [30 , 40 , 50] per positives

3. **Pos / Unl / Test** : finally we save Positive samples and unlabeled samples and test samples into different file so its easier to handle

The data-sets stored in **.arff** files type .

Overview of Audiology Data-Set files

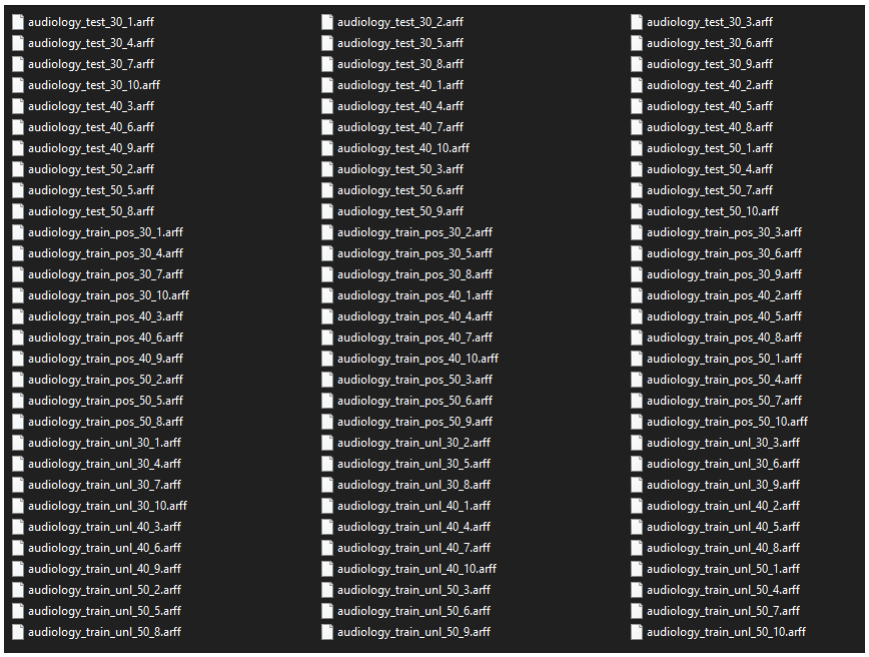


Figure 4.3: Overview of Audiology data-set files

So we can use those lines of codes to import our data-sets

```
datasets =['audiology', 'breast-cancer', 'dermatology', 'hepatitis','lymph
                                        ', 'pima', 'soybean', 'vote', 'chess'
                                        , 'nursery']

for dataset in datasets:
    for perc in ['30', '40', '50']:
        for fold in range(1, 11):
            pos_name = 'data/' + dataset + '_train_pos_' + perc + '_' +
                                                    str(fold) + '.arff'
```

```
        unl_name = 'data/' + dataset + '_train_unl_' + perc + '_' +
                                                    str(fold) + '.arff'
        test_name = 'data/' + dataset + '_test_' + perc + '_' + str(
                                                    fold) + '.arff'
```

Now our Data ready to explore

4.5.2 Binarization of the Data-sets:

In order to transfer the data to a binary values we use **One-Hot Encoder** a technique used to represent categorical variables as binary vectors .

4.5.2.1 One-Hot Encoder:

One-hot encoding is a technique used to convert categorical variables into a binary vector representation that can be used as input for machine learning algorithms. It is particularly useful when dealing with categorical data that does not have an inherent order or ranking.

4.5.2.2 The one-hot encoding process :

1. Identify Categorical Variables: Determine which variables in your data-set are categorical and need to be encoded.
2. Create Binary Columns: For each categorical variable, create a new binary (0 or 1) column for each unique category within that variable. The number of new columns created corresponds to the number of unique categories.
3. Assign Binary Values: Assign a value of 1 to the corresponding binary column for each instance that belongs to a specific category, and assign a value of 0 to all other binary columns.

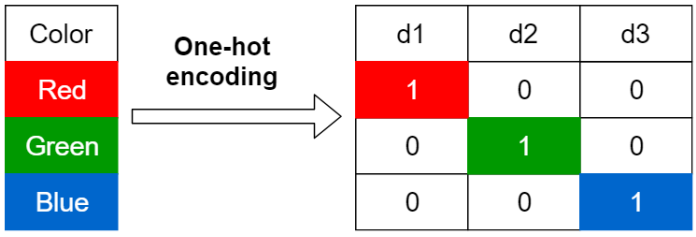


Figure 4.4: one hot encoding

```
enc = preprocessing.OneHotEncoder()  # (categorical_features='all')
all_data = np.concatenate( (train_pos_data, train_unl_data, test_data) ,
                                        axis=0)
enc.fit(all_data)

train_unl_encoded = enc.transform(train_unl['data']).toarray()
train_pos_encoded = enc.transform(train_pos['data']).toarray()
train_test_encoded = enc.transform(test['data']).toarray()
```

It converts categorical features into a binary representation that can be used in our NBPSO-SVM method.

4.6 Results :

In this section, we provide an exhaustive set of experiments to show the effectiveness of our PU Learning approach in categorical data. Experiments are performed on 30 samples derived from 10 data-sets, publicly available on the Machine Learning Repository of the **UCI** (the UCI machine Learning repository http://archive.ics.uci.edu/ml/).

For each data-set, we produce three different samples that differ from each other in the number of examples labeled as positive, respectively 30% , 40% and 50% of the positive class. The remaining positive examples plus all negative examples are considered unlabeled. We assume the majority class as positive, the other as negative. If the data-set does not describe a binary classification problem we select the two largest classes (in number of instances) to reduce the problem to a binary classification task. Finally, as an additional pre-processing, all numeric attributes are discretized into 10 bins with equal width value .
Details of the 30 samples are already seen in Table 4.1.
To evaluate the our approach, we used the the data and results mentioned in the paper [64].
F-Measure (F1 score) as a performance indicator

		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

Figure 4.5: precision and recall trade-off

F-Measure (F1 score) equal to :

$$\text{F1 score} = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

The Recall :

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

The Precision :

$$Precision = \frac{TP}{TP + FP}$$

The F-measure allows better evaluation in unbalanced data-sets; that's why we prefer to use these measurements rather than precision. F-measure values are calculated and averaged over a 10-fold cross-validation scheme. Note that unlike other partially supervised parameters, such as anomaly detection, the goal of PU learning is to build a binary classifier that should discriminate well between positive and negative examples.

Each database is divided into 10 parts (folds) on which a learning and testing process is applied 10 times. Each time two different parts are used for learning and testing. The final result will be the average of the 10 parts

This called **K-fold Cross-validation**

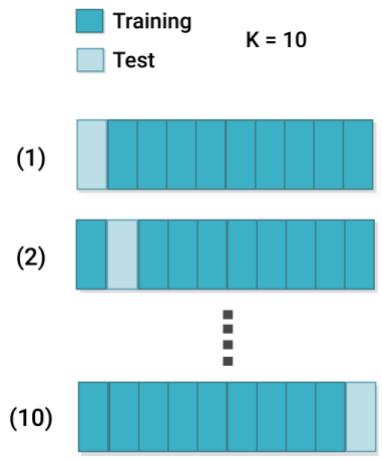


Figure 4.6: K-fold Cross-validation

The results obtained in the table 4.3 show the best performance mentioned in bold, our method outperforms the other methods 15 times out of 30, with an average F-measure of 0.797 which is the best among the set of approaches.

Dataset	**perc**	**pos**	**Precision**	**Recall**	**F1-score**
Audiology	30	15	0.8672619047619048	1.0	0.9238927738927
Audiology	40	20	0.9291666666666668	1.0	0.9587412587412
Audiology	50	26	1.0	1.0	1.0
Breast-cancer	30	54	0.3440566936503469	0.7194444444444443	0.46091893176
Breast-cancer	40	72	0.36948375970434794	0.60416666666666	0.4516500918354
Breast-cancer	50	91	0.4173076923076923	0.65972222222222	0.5083051730877
Dermatology	30	30	0.5874349833173362	1.0	0.7323585858585
Dermatology	40	40	0.6890398980104863	0.985714285714285	0.7954971139971
Dermatology	50	51	0.7764141414141414	1.0	0.8662155441886
Hepatitis	30	9	0.7917195304695304	0.8871794871794872	0.8333609111
Hepatitis	40	12	0.8285010822510822	0.8775641025641026	0.8494737033192
Hepatitis	50	15	0.8946352258852259	0.8288461538461538	0.8530044608065
Lymph	30	17	0.7409595959595959	0.9625	0.8318099169492
Lymph	40	22	0.7359324009324009	0.95	0.8261152882205
Lymph	50	28	0.7994949494949495	0.925	0.8531269349845
Lima	30	135	0.46631316783176296	0.7987179487179487	0.5880455661632
pima	40	180	0.48720974543699735	0.7841880341880342	0.5996816578175
Pima	50	225	0.5116741718057506	0.7428774928774928	0.6043495116388
Soybean	30	25	0.8411888111888111	1.0	0.9080849851902
Soybean	40	33	0.8628671328671329	0.9888888888888889	0.9162668033
Soybean	50	42	0.8738816738816739	0.9444444444444444	0.9039333841851
Vote	30	72	0.6127833946221043	0.9941176470588236	0.7522500218213
Vote	40	96	0.8072946419628341	0.9761029411764707	0.8808161635406
Vote	50	120	0.8734314954051795	0.9698529411764707	0.917120614140
Chess	30	451	0.6499261253444992	0.9300653594771242	0.7388632396604
Chess	40	601	0.6839809436858983	0.9143790849673202	0.7568131253254
Chess	50	751	0.71133177591884	0.9221706226350189	0.7876855610734
Nursery	30	1166	0.6499612942700712	1.0	0.766607815584

Table 4.2: Data-sets Results

4.7 Conclusion:

In this chapter, we have introduced a new method called **NBPSO-SVM** which is a coupling of a swarm intelligence algorithm PSO (Particle swarm Algorithm) and the machine learning SVM method. The binary nature of our problem requires the transformation of the algorithm. We have illustrated our process with an application on real databases (10×3 databases). We have described each step of our process and presented the results of our experiments, and to validate this method we have made a comparison with other learning methods.
The results obtained are very satisfactory and show the effectiveness of our approach.

Dataset	pos%	BPSO-SVM	GPU BN	GPU MT	Pulce	BPN	APNB	PTAN	APTAN
Audiology	30%	**0.923**	0.839	0.902	0.745	0.68	0.7	0.66	0.66
Audiology	40%	**0.958**	0.879	0.804	0.846	0.75	0.74	0.71	0.66
Audiology	50%	**1.0**	0.98	0.991	0.899	0.8	0.8	0.78	0.71
Breast-cancer	30%	0.461	0.475	0.45	**0.534**	0.4	0.3	0.43	0.43
Breast-cancer	40%	0.452	0.483	**0.513**	0.438	0.42	0.4	0.43	0.45
Breast-cancer	50%	**0.541**	0.517	0.535	0.443	0.42	0.41	0.44	0.44
Dermatology	30%	0.74	**1**	0.834	0.992	0.57	0.57	0.57	0.56
Dematology	40%	0.801	**1**	0.836	0.992	0.57	0.58	0.57	0.57
Dematology	50%	0.867	**0.992**	0.951	**0.992**	0.59	0.60	0.57	0.58
Hepatitis	30%	0.833	0.822	0.665	0.843	**0.87**	**0.87**	0.85	0.86
Hepatitis	40%	0.849	0.778	0.654	0.873	**0.88**	**0.88**	0.85	0.85
Hepatitis	50%	0.853	0.764	0.742	0.855	**0.88**	**0.88**	0.86	0.85
Lymph	30%	0.831	0.827	0.782	**0.851**	0.84	0.85	0.79	0.84
Lymph	40%	0.823	0.825	0.795	0.827	**0.84**	0.83	0.79	0.81
Lymph	50%	0.862	0.824	0.835	0.814	0.86	**0.87**	0.81	0.82
Chess	30%	**0.739**	0.689	0.663	0.696	0.58	0.64	0.59	0.64
Chess	40%	**0.757**	0.691	0.665	0.688	0.58	0.64	0.60	0.64
Chess	50%	**0.788**	0.773	0.650	0.655	0.58	0.64	0.60	0.64
Nursery	30%	**0.881**	0.809	0.761	0.739	0.65	0.65	0.56	0.5
Nursery	40%	0.813	**1**	0.762	0.773	0.69	0.69	0.61	0.56
Nursery	50%	0.83	**0.96**	0.779	0.807	0.69	0.7	0.74	0.44
Pima	30%	**0.589**	0.588	0.576	0.532	0.49	0.5	0.5	0.5
Pima	40%	**0.599**	0.568	0.593	0.547	0.49	0.5	0.5	0.51
Pima	50%	0.604	**0.609**	0.605	0.528	0.49	0.51	0.51	0.52
Soybean	30%	**0.908**	0.893	0.776	0.738	0.81	0.86	0.80	0.81
Soybean	40%	**0.916**	0.883	0.852	0.767	0.86	0.86	0.84	0.83
Soybean	50%	0.903	0.890	**0.923**	0.823	0.92	0.92	0.88	0.86
Vote	30%	0.752	**0.826**	0.799	0.679	0.62	0.62	0.65	0.55
Vote	40%	**0.880**	0.850	0.790	0.801	0.71	0.71	0.58	0.54
Vote	50%	**0.917**	0.844	0.829	0.829	0.77	0.77	0.61	0.56
#wins		**15**	7	3	3	3	4	0	0
Avg. f1-score		**0.797**	0.769	0.743	0.751	0.677	0.686	0.653	0.643
30		0.765	0.777	0.72	0.735	0.651	0.665	0.631	0.635
40		0.776	0.796	0.727	0.755	0.679	0.683	0.648	0.642
50		**0.816**	0.815	0.784	0.764	0.7	0.71	0.679	0.652

Table 4.3: Results Comparison with others approaches

General conclusion

The work presented in this dissertation is situated in the context of machine learning, more specifically in the framework of positive and unlabeled (PU) learning. Generally, binary classifiers take as input two sets of data, one containing positively labeled data and the other containing negatively labeled data. The goal is to learn a separator between these two sets of data. Unfortunately, it often happens that these sets only contain positively labeled data along with unlabeled data.

Our objective was to leverage Swarm Intelligence techniques to enhance the performance of positive and unlabeled learning methods. To successfully complete this dissertation, we started with a literature review to define our problem. We began with a first chapter on the concepts of machine learning and its methods. In the second chapter, we presented a state-of-the-art overview of positive and unlabeled learning by summarizing some papers. The third chapter focused on the methods used in our contribution, namely the Particle Swarm Optimization (PSO) algorithm in its continuous version and the Support Vector Machine (SVM) method.

The last chapter was dedicated to our contribution. First, we proposed a new method for selecting reliable negatives from unlabeled data, which is based on the PSO algorithm. To adapt to the binary nature of our problem, we proposed a binary version of this algorithm. Subsequently, to induce our classification model, we used the SVM method. The experiments and results obtained clearly demonstrate that our method mostly outperforms other methods used for comparison.

Perspectives:

- Considering conducting a comparative study to choose the best objective function.
- Considering the reutilization of this method, which could be applied to other types of

data for positive and unlabeled (PU) learning classification, such as text classification, web classification, etc.

- Considering the optimization of the selection step by dimensionality reduction. For example, clustering could be used to select a representative for each cluster instead of the entire population.
- Considering accelerating the movement of crows by implementing a parallel process.

Bibliography

[1] Batta Mahesh. Machine learning algorithms-a review. *International Journal of Science and Research (IJSR).[Internet]*, 9:381–386, 2020.

[2] Ritesh Kumar Jain, Dr Ruchi Doshi, Dr Kamlesh Lakhwani, and Dr Kamal Kant Hiran. *Machine learning: Master supervised and unsupervised learning algorithms with real examples*. BPB Publications, 2021.

[3] Shruti M. Differences between ai vs. machine learning vs. deep learning: Simplilearn, Feb 2023.

[4] https://www.javatpoint.com/applications-of-machine-learning.

[5] https://www.aihorizon.com/supervised$_u nsupervised_m achine_l earning$.

[6] David Petersson. what is sup-learning, Mar 2021.

[7] www.geeksforgeeks.org/support-vector-machine-algorithm, Nov 2022.

[8] Geoffrey I Webb, Eamonn Keogh, and Risto Miikkulainen. Naïve bayes. *Encyclopedia of machine learning*, 15:713–714, 2010.

[9] www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html.

[10] Harsh H Patel and Purvi Prajapati. Study and analysis of decision tree based classification algorithms. *International Journal of Computer Sciences and Engineering*, 6(10):74–78, 2018.

[11] Introduction to random forest in machine learning.

[12] How does logistic regression work?/www.kdnuggets.com/2022/07/logistic-regression-work.html.

[13] overview of neural networks - researchgate.

[14] Batta Mahesh. Machine learning algorithms-a review. *International Journal of Science and Research (IJSR).[Internet]*, 9:381–386, 2020.

[15] Manil wagle. Association rules: Unsupervised learning in retail, Mar 2020.

[16] Apriori algorithm,en.wikipedia.org/wiki/apriori$_a lgorithmcite_n ote - apriori - 1, Mar 2023$.

[17] https://www.geeksforgeeks.org/what-is-reinforcement-learning/, Mar 2023.

[18] Charles Elkan and Keith Noto. Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 213–220. ACM, 2008.

[19] Ryota Kiryo, Gang Niu, David Duvenaud, and Satoshi Hirano. Positive-unlabeled learning with non-negative risk estimator. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1818–1827. PMLR, 2017.

[20] Bing Liu and Dacheng Tao. Positive-unlabeled learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 30(10):3049–3069, 2019.

[21] Bing Liu, Wei Wang, and Sinno Jialin Pan. Positive-unlabeled learning with non-negative risk estimator. *ACM Trans. Knowl. Discov. Data*, 10(3):21:1–21:22, June 2016.

[22] Bo Du, Jianmin Wang, Wei Fan, and Zhihui Huang. Analysis of learning from positive and unlabeled data. *IEEE Transactions on Neural Networks and Learning Systems*, 25(10):1824–1836, 2014.

[23] Ryota Kiryō and Hisashi Kashima. Positive-unlabeled learning with non-negative risk estimator. *Journal of Machine Learning Research*, 18(1):5605–5642, 2017.

[24] Sohee Lee and Seungjin Lee. Positive-unlabeled learning for imbalanced data classification: A comparative study. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2018.

[25] Bo Du, Jianmin Wang, Wei Fan, and Zhihui Huang. Convex pu learning with attribute assumptions. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 3264–3270, 2018.

[26] Misha Denil, David Matheson, and Nando De Freitas. Learning from positive and unlabeled examples. In *Proceedings of the 31st International Conference on Machine Learning (ICML-2014)*, pages 992–1000, 2014.

[27] Bo Li, Xiaolin Zhang, Yong Li, Xuelong Li, and Bin Li. Positive-unlabeled learning: A survey. *Information Sciences*, 471:350–368, 2019.

[28] Bing Liu, Yan Dai, Xiaodong Li, Wee Sun Lee, and S Yu Philip. Partially supervised classification of text documents. In *Proceedings of the 19th International Conference on Machine Learning (ICML-2002)*, pages 387–394, 2002.

[29] Yan Jiang and Yi Yang. A hybrid approach for fraud detection with positive-unlabeled learning. *Expert Systems with Applications*, 62:358–369, 2016.

[30] Eleazar Eskin, Andrew Arnold, Michael Prerau, Leonid Portnoy, and Salvatore J Stolfo. Positive-unlabeled anomaly detection with kernel density estimation. In *Proceedings of the 10th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 327–336. ACM, 2006.

[31] Zihao Zhang, Tong Zhou, Xiaohui Chen, and Jing Zhan. A novel positive-unlabeled learning approach for effective anomaly detection. *Information Sciences*, 563:252–266, 2021.

[32] Yuan Meng, Wei Shen, Jie Zhang, and Jiawei Han. Positive-unlabeled learning with non-negative risk estimator. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[33] Teemu Kanstrén. A look at precision, recall, and f1-score, May 2021.

[34] Classification: Roc curve and auc nbsp;|nbsp; machine learning nbsp;|nbsp; google developers.

[35] Shanshan Wang, Heng Zhang, and Chao Chen. Positive-unlabeled learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 30(3):152–172, 2019.

[36] Bo Jiang, Yong Zhang, and Jiebo Luo. Positive-unlabeled learning with non-negative risk estimator. *IEEE Transactions on Neural Networks and Learning Systems*, 29(12):6294–6304, 2018.

[37] Vapnik Cortes. Cortes c., vapnik v. *Support-vector networks, Machine learning*, 20(3):273–297, 1995.

[38] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[39] Jianbo Lu, Shuang Xu, and Hui Wang. Comparison of svm kernels on spam email classification. In *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, pages 80–284. IEEE, 2019.

[40] Vojislav Kecman. Support vector machines–an introduction. In *Support vector machines: theory and applications*, pages 1–47. Springer, 2005.

[41] Shouwei Li, Ke Liu, Jinchang Ren, and Lu Zheng. Diagnosis of parkinson's disease based on svm-recursive feature elimination algorithm. *Frontiers in Neuroscience*, 14:107–350, 2020.

[42] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, 1999.

[43] James Kennedy. *Swarm intelligence*. Springer, 2006.

[44] Marco Dorigo and Thomas Stützle. *Ant colony optimization: overview and recent advances*. Springer, 2019.

[45] J Senthilnath, SN Omkar, and V Mani. Clustering using firefly algorithm: performance study. *Swarm and Evolutionary Computation*, 1(3):51–171, 2011.

[46] Vipul Sharma, SS Pattnaik, and Tanuj Garg. A review of bacterial foraging optimization and its applications. In *National Conference on Future Aspects of Artificial intelligence in Industrial Automation, NCFAAIIA*, pages 9–12, 2012.

[47] Carmelo JA Bastos Filho, Fernando Buarque de Lima Neto, Anthony J da CC Lins, Antônio IS Nascimento, and Marília P Lima. Fish school search. *Nature-inspired algorithms for optimisation*, 193:70–277, 2009.

[48] Christian Blum and Daniel Merkle. *Swarm intelligence: introduction and applications*. Springer Science & Business Media, 2008.

[49] Anupam Biswas and Bhaskar Biswas. Swarm intelligence techniques and their adaptive nature with applications. *Complex System Modelling and Control Through Intelligent Soft Computations*, pages 101–273, 2015.

[50] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948. IEEE, 1995.

[51] Xiaodong Li and Andries P Engelbrecht. Particle swarm optimization: an introduction and its recent developments. In *Proceedings of the 9th annual conference companion on Genetic and evolutionary computation*, pages 2000–3414, 2007.

[52] Federico Marini and Beata Walczak. Particle swarm optimization (pso). a tutorial. *Chemometrics and Intelligent Laboratory Systems*, 149:50–165, 2015.

[53] Ali Mirjalili. Particle swarm optimization (pso).

[54] M Clerc. Particle swarm optimization. *Part. Swarm Optim.*, pages 1–17, 2010.

[55] Daniel Bratton and James Kennedy. Defining a standard for particle swarm optimization. In *2007 IEEE swarm intelligence symposium*, pages 120–127. IEEE, 2007.

[56] Yuhui Shi and Russell Eberhart. A modified particle swarm optimizer. In *1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360)*, pages 69–73. IEEE, 1998.

[57] Riccardo Poli, James Kennedy, and Tim Blackwell. Particle swarm optimization: An overview. *Swarm intelligence*, 1:33–57, 2007.

[58] Yuhui Shi and Russell C Eberhart. Empirical study of particle swarm optimization. In *Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)*, volume 3, pages 1945–1950. IEEE, 1999.

[59] Konstantinos E Parsopoulos and Michael N Vrahatis. Particle swarm optimization method in multiobjective problems. In *Proceedings of the 2002 ACM symposium on Applied computing*, pages 603–607, 2002.

[60] Carlos A Coello Coello, Gary B Lamont, David A Van Veldhuizen, et al. *Evolutionary algorithms for solving multi-objective problems*, volume 5. Springer, 2007.

[61] Tianshi Chen, Jun He, Guangzhong Sun, Guoliang Chen, and Xin Yao. A new approach for analyzing average time complexity of population-based evolutionary algorithms on unimodal problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(5):1092–1106, 2009.

[62] Ya Bi, Mei Xiang, Florian Schäfer, Alan Lebwohl, and Cunfa Wang. A simplified and efficient particle swarm optimization algorithm considering particle diversity. *Cluster Computing*, 22:13273–13282, 2019.

[63] Morteza Nazari-Heris, Mehdi Abapour, and Behnam Mohammadi-Ivatloo. An updated review and outlook on electric vehicle aggregators in electric energy networks. *Sustainability*, 14(23):15747, 2022.

[64] Teresa M. A. Basile, Nicola Di Mauro, Floriana Esposito, Stefano Ferilli, and Antonio Vergari. Density estimators for positive-unlabeled learning. In Annalisa Appice, Corrado Loglisci, Giuseppe Manco, Elio Masciari, and Zbigniew W. Ras, editors, *New Frontiers in Mining Complex Patterns*, Lecture Notes in Computer Science, pages 49–64. Springer International Publishing, March 2018. New Frontiers in Mining Complex Patterns - Sixth edition of the International Workshop NFMCP in conjunction with ECML-PKDD 2017, NFMCP 2017 ; Conference date: 18-09-2017 Through 22-09-2017.

[65] Yitong Jiang, Xinyu Zhang, Xingyi Li, and Zhaoxi Li. Positive unlabeled learning for sentiment analysis. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 1669–1674. IEEE, 2020.

[66] Jinyi Yan, Yu Lu, and Hui Wang. Positive unlabeled learning for improving sentiment analysis of movie reviews. *IEEE Access*, 9:49557–49565, 2021.

[67] Yilin Zhang, Shuai Zhang, Kaili Mao, Jingsheng Lei, and Xu Sun. Pu learning for sentiment analysis with unbalanced and insufficient labeled data. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1912–1922, 2018.

[68] Jia Li, Hanqing Zhao, Tie-Yan Liu, and Xiaodong Wang. Active learning for positive-unlabeled learning. *IEEE Transactions on Knowledge and Data Engineering*, 32(3):480–492.