MLT-01 Summary Document

Overview

This document summarizes the lecture MLT-01 on Machine Learning. This lecture gives an introduction to various algorithms used in machine learning in an intuitive way. It covers some of the most widely used ML algorithms in trading.

The lecture covers the following topics:

-
- Introduction to Machine Learning
- Its methods and applications
- Supervised vs Unsupervised learning
- Intuitive understanding of transfer learning
- Decision trees and random forests
- Comparing Different Criteria
- Logistic regression
- Overview of feature selection methods
- Python Implementation

Introduction to Machine Learning

In today's world, we see an increase in computing speed and data creation. These changes are being driven by technological advances in computing and learning algorithms.

Machine learning, a set of learning algorithms, is based on statistics and, with the help of data, can solve specialised problems as a human or even better than us. It is a subtopic of artificial intelligence.

Methods and Applications

- Classification: This type of algorithm can predict the labelled output. For example, it can predict if a bank customer is eligible for a loan or not (target) based on some inputs/features such as marital status, number of children, annual salary, etc. In our domain, such algorithms can predict whether a market will go up or down in the next trading session.
- Clustering: Such algorithms can cluster stocks/assets based on certain criteria.
- Neural networks: This set of algorithms mimics how biological neurons interact and create networks. They can be used for classification and regression problems. For example, scan images of hand-written numbers and identify which numbers are they.
- Deep learning: A huge interconnected set of neural networks.
- Natural language processing: This set of algorithms can process unstructured data and enable us to summarise them so that informed decisions can be made. For example, it enables us to analyze financial news to create a buy or sell signal for a stock.

- Computer vision: These algorithms are extensively used to interpret moving objects and try to find patterns in them. For example, they are used a lot in self-driving cars.

Supervised vs Unsupervised Learning

Supervised Learning: It maps the *features* with the *target* in the training dataset to build predictive models. -

Supervised learning algorithms have a clear set of features and a target variable.

Target variable: The target variable is a field/label/column in a dataset that the user is interested to predict. For example:

- If the user wants to know the direction of an instrument then he will be interested in the close price for the next day.
- Another use case may be to come up with stock weights for allocation in a portfolio.

Features: The set of information/inputs required to predict the target variable is called features. For example, one may use price information (open, high, low, close), technical indicators, statistical parameters and so on to predict the target variable.

Supervised learning can be further divided into
- classification and
- regression

The target is used to supervise the learning process.

Unsupervised learning algorithm: There is no target supervision. Algorithms try to find patterns in the data based on - the features we provide.

Forecasting is about predicting the future. Prediction is about predicting something not related to a future variable. For example, predict if we have a dog or a cat. Forecasting is a subtopic of prediction.

Classification algorithms are models whose targets are categorical or nominal values.

Transfer Learning

A model which has taken a huge amount of effort to get trained and when applied to a new context, very little amount of training is necessary to make the model plausible and useful for this new context. This process is called transfer learning.

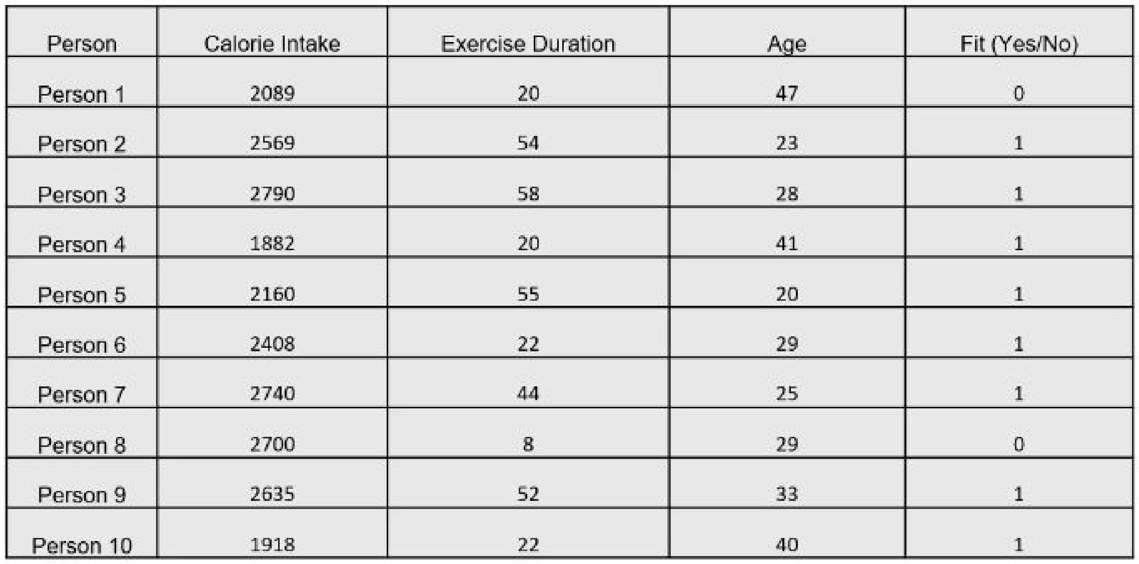
Decision Trees

Decision trees are supervised learning algorithms that can be used for both: classification and regression problems.

Classification algorithms can classify the data into labels. For example:

- Whether a credit card transaction is legitimate or fraudulentWhether a tumour is benign or malignant

Let’s explain the Decision Trees algorithm for classification with an example:

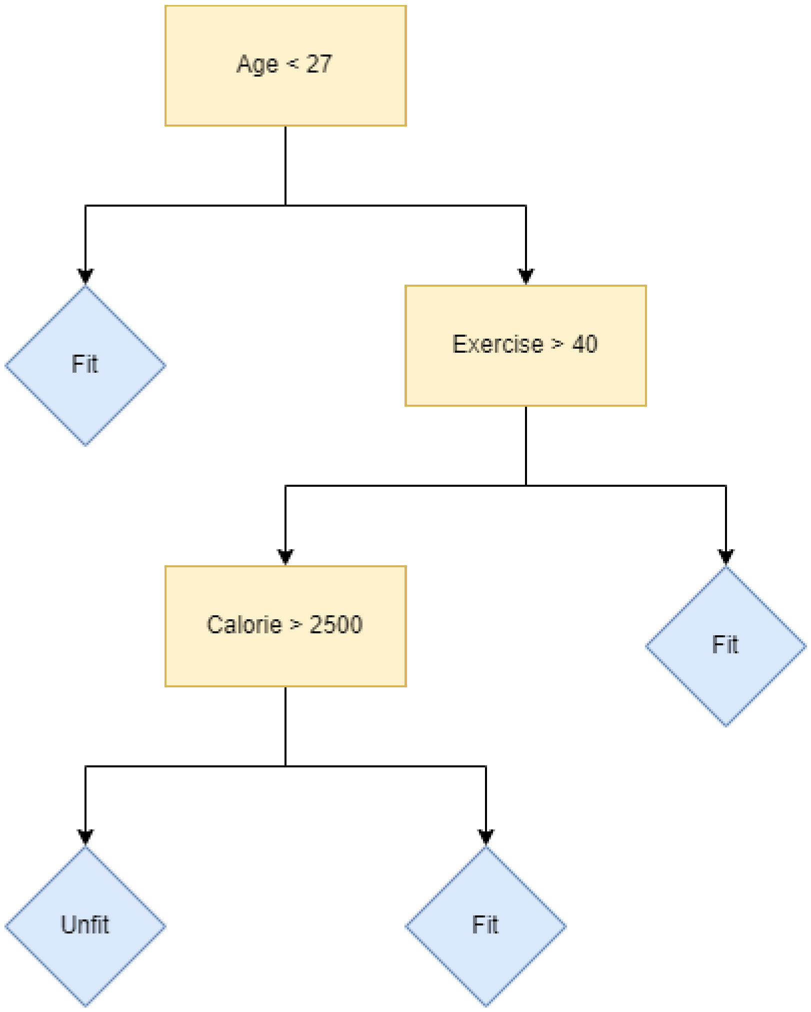


Person	Calorie Intake	Exercise Duration	Age	Fit (Yes/No)
Person 1	2089	20	47	0
Person 2	2569	54	23	1
Person 3	2790	58	28	1
Person 4	1882	20	41	1
Person 5	2160	55	20	1
Person 6	2408	22	29	1
Person 7	2740	44	25	1
Person 8	2700	8	29	0
Person 9	2635	52	33	1
Person 10	1918	22	40	1

Imagine we have the above dataset. Calorie Intake, Exercise Duration and Age will be our X1, X2 and X3 features, respectively. Y will be the Fit variable.

Let’s create a cut-off. The first node of our tree will be Age. Whenever the Age is less than 27, we say the person is fit, otherwise, he’s not fit. As you can see in the Age column, persons 2, 5 and 7 aren’t fit. For these people, let’s create another node. For people who have exercised for more than 40 minutes, we say they are fit, otherwise, they’re not fit. Lastly, for the people who are not fit, we create our last node and say: For people that had a calorie intake higher than 2500, we say they’re unfit, otherwise, they’re fit.

Let’s see the corresponding diagram:



The yellow rectangles are the nodes, the blue rhombus are the leaves.

To create a decision tree algorithm, we need to answer the following questions:

1. Which attribute to choose at each node?

   Answer: We chose Age, Exercise Duration and Calorie Intake as our first, second and third attributes, respectively.
2. What cut-off to choose for the attribute?

   We chose Age < 27, Exercise > 40 and Calorie > 2500 as cut-offs. These 2 first answers resolved the creation of nodes.
3. How to split the attribute?

   Answer: We split the attribute into 2 parts
4. What is the depth of the tree?

   Answer: We have 3 decision levels: Age < 27, Exercise > 40 and Calorie > 2500. Thus, the depth is 3.

Gini impurity function helps us to make a precise split at each level of the decision tree.

The formula is given by:

$Gini(t) = 1 - \sum_j \quad p_j^2$, where p is the proportion of people in class j.

You will need to change the cut-off of each decision level to optimize the GINI index, and consequently, obtain a decision tree fitted on the data. You have to choose the features, the decision (e.g.: age higher or less than some number) and the threshold (e.g.: the number 27 in the decision Age < 27).

The classification error measure impurity at level of the tree.

Let's define this type of impurity as:

$Error(t) = 1 - max\, p_j$, at note t.
Example, if we have 2 features:

$$Error(t) = 1 - max(\, p_1, p_2)$$

Comparing Different Criteria

There are 3 different criterias:

- The entropyThe GINIThe missclassification error

In order to choose the best model, we should follow the Occam's Razor rule: Between 2 or more models of similar generalisation errors, one should prefer the model which is the simplest.

Example:

If 2 models both have a similar accuracy, but model 1 has depth 3 and model 2 has depth 5, then you should choose model 1 as your preferred model because it is much simpler.

You want to optimize the model such that it does not overfit. To ensure that the model does not overfit, you can split the dataset into train and test datasets. This will allow, once you fit the model with the train dataset, to see how well the model will perform in the test, or unseen, dataset. You want a model that fits sufficiently well to have good results not only on the train dataset, but also on the test dataset. If not, you fall in the overfitting trap.

You can also address overfitting in decision trees by:

- Pre-pruning: Stop the algorithm when the tree becomes large.Post-pruning: Trim the nodes in a bottom-up manner.

Random Forests

It is well known that the decision tree algorithm tends to overfit the data. That's why random forests appeared as a solution to this problem.

Random forests are about creating more than one tree (as in the decision tree algorithm) and then choosing randomly which trees are best suited to fit the data. This process is done with a voting rule.

One voting rule can be the majority rule. So, i.e., if more than 50% of the randomly selected trees say that person 1 is unfit, then you assign 0 (or unfit) to the person target variable.

The decision tree model depends highly on the selected features. However, the random forest algorithm selects features randomly. So, as you might guess, this randomness in the feature selection process makes the model generalize over the data in a better way. Last but not least, the accuracy can be improved with this model compared to the decision tree algorithm.

Logistic Regression

Logistic regression estimates the probability of an outcome, such as a person voting for the Republican or the Democratic party in the US, based on a given dataset of independent variables, such as income level, wealth, education, sex, etc.

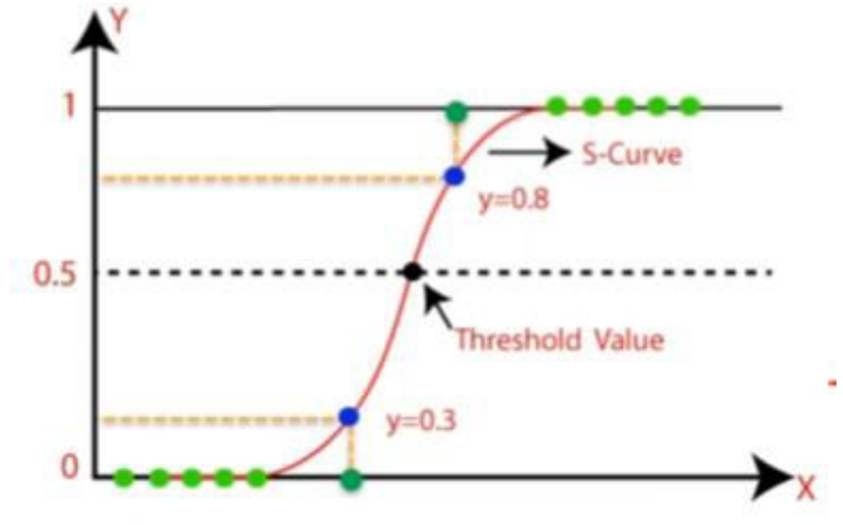
Differences between logistic regression and random forest/decision trees (RF/DF):

- Logistic regression uses a linear equation to predict the target variable with the independent/feature variables. RF/DF bases the classification of the target variable on a top-down induction analysis using the features' conditions.
- Logistic regression has some assumptions, e.g. the probability function. There is no need for assumptions for RF/DF.
- Overfitting might be a problem for logistic regression. It is not for the random forest.

Logistic regression lets you predict the probability of events with the use of independent variables. Whenever you face a categorical target variable, logistic regression can help you to determine the probability of the categorical variable outcomes per each observation in the dataset.

Commonly used to model the credit risk of individuals or enterprises.
Instead of a classification variable, the model returns a probability of an outcome.
This probability will be between 0 and 1.

For example, imagine we have a university student pass or fail an exam. Pass will be 1 and fail will be 0. An S-shape curve will describe the behaviour of the target data points, as in the following image:



The threshold in the figure is 50%, i.e., if a student has a probability of less than 50% to pass the exam, then we assign him a 0 target value; otherwise, we assign him 1.

The probability function is given by:

$$Probability = \frac{1}{1+e^{-z}}$$

Where:

$$z = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots$$

And X are the features.

<u>Overview of feature selection methods</u>

Some features might not be necessary to fit a model with a dataset. Thus, it's really important to select the appropriate features in order to fit the model appropriately with the data. Here we show a list of some feature selection methods:

- Correlation coefficient: Correlated features are dropped from the features dataset.
- Fisher's score: Score features to let you know which one explain better the target variable.
- Forward feature selection: Fit multiples regressions and on each iteration it increases the number of features up to find the best model.
- Backward feature selection: You run a regression and then drop the unnecessary independent variables.
- Best subset feature selection: Creates an iteration to create consecutively a subset of features to get, in the end, the best subset of features that can predict the prediction feature.
- Lasso (l1) regularization: Drop independent variables based on a penalisation in the regression equation.

We end the session by spending some time on the code and looking at the implementation of the concepts we learned in the lecture.