UNIVERSITY OF THESSALY

Deep learning based prediction in multivariate time series - Πρόβλεψη με Βάση τη Βαθιά Μάθηση σε Πολυμεταβλητές Χρονοσειρές

by

Konstantinos Christoforidis - 2661

A thesis submitted in part fulfilment of the requirements for the degree of Master of Science in Electrical and Computer Engineering of the University of Thessaly

in the
School of Engineering
Department of Electrical and Computer Engineers

Supervisor Professor: Dr. Dimitrios Katsaros

Date: X/X/2023

Declaration of Authorship

I, Konstantinos Christoforidis, declare that this thesis titled, 'Deep learning based prediction in multivariate time series' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at the University of Thessaly for Electrical and Computer Engineers.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at the University of Thessaly for Electrical and Computer Engineers or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

Approved by the three-member committee of inquiry:

(Signature)	*(Signature)*	*(Signature)*
……..	……..	……..
Dr. Dimitrios Katsaros	Dr. Dimitrios Rafailidis	Dr. Georgios Thanos

Declaration of Responsibility

I certify that I am the author of this thesis, and that any assistance which I have had in its preparation is fully acknowledged and reported in the thesis. I have also cited any sources from which I have made use of data, ideas or words, whether quoted verbatim or paraphrased. Also, I certify that this thesis was prepared by me personally specifically for the the requirements of the curriculum of the Department of Electrical and Computer Engineering of Thessaly.

(Signature)

.....

Konstantinos Christoforidis

Sometimes it is the people no one can imagine anything of who do the things no one can imagine.

Alan Mathison Turing

I have had my results for a long time; but I do not yet know how I am to arrive at them.

Carl Friedrich Gauss

Invert. Always invert.

Carl Gustav Jacob Jacobi

Περίληψη

Καθώς ο τομέας των Μεγάλων Δεδομένων και της Επιστήμης των Δεδομένων (ΕΔ) αναπτύσσεται τόσο στην οικονομική όσο και στην επιστημονική κοινότητα, αυξάνεται και η εφαρμογή τους στους αντίστοιχους τομείς. Μαζί με την επέκταση αυτών των τομέων, ο τομέας της μηχανικής μάθησης (ΜΜ) γεννά πολλές εφαρμογές που αλλάζουν τον τρόπο με τον οποίο αντιλαμβανόμαστε και αναλύουμε τα δεδομένα, ταχύτερα και με μεγαλύτερη ακρίβεια. Μια τέτοια εφαρμογή αντιμετωπίζει το πρόβλημα της πρόβλεψης της επιστημονικής επιτυχίας των συγγραφέων μέσω του αριθμού των αναφορών στις εργασίες τους. Αν και τα τελευταία χρόνια έχουν προταθεί πολλές λύσεις, η εξέλιξη της επιστήμης των δεδομένων και της μηχανικής μάθησης έχει προσφέρει στην κοινότητα νέα σύνολα εργαλείων προς την κατεύθυνση της εξεύρεσης καλύτερων και ακριβέστερων λύσεων. Σκοπός της παρούσας διπλωματικής εργασίας είναι να εξηγήσει θεωρητικά τις βασικές έννοιες στα τμήματα της ΕΔ και της ΜΜ, να παρέχει μια γρήγορη εισαγωγή σε ορισμένες από τις πιο κοινές μεθόδους που χρησιμοποιούνται και, τέλος, να παρουσιάσει μια εφαρμογή της Βαθιάς Μάθησης (ΒΜ) με τη χρήση Τεχνητών Νευρωνικών Δικτύων (ΤΝΔ) προκειμένου να δημιουργηθεί ένα ακριβές μοντέλο πρόβλεψης των αναφορών που λαμβάνουν επιστήμονες με τις δημοσιεύσεις τους.

Abstract

As the field of Big Data (BD) and Data Science (DS) grows both in the financial and scientific communities, their application in their respective fields grows as well. Along with the expansion of these fields, the field of Machine Learning (ML) gives birth to many applications that changes the way we perceive and analyze data, faster and with more accuracy. One such application tackles the problem of predicting scientific success of authors through the number of citations of their papers. Though throughout the last years many solutions have been proposed, the evolution of DS and machine learning (ML) has provided the community with new sets of tools towards finding better, more accurate solutions. The purpose of this thesis is to theoretically explain the basic concepts in the departments of DS and ML, provide a quick insight in some of the most common methods used and, finally, introduce one application of Deep Learning (DL) with the use of Artificial Neural Networks (ANNs) in order to create an accurate citation predicting model.

Keywords: Big Data, Data Science, Machine Learning, Deep Learning, Un/Supervised Learning, Artificial Neural Networks, Python, Jupyter Notebook, Infometrics, Bibliometrics, Scientometrics, Citations

Acknowledgements

I would like to express gratitude and appreciation to my thesis supervisor professor Dr. Dimitrios Katsaros of the Department of Electrical and Computer Engineering at University of Thessaly. Thanks to his guidance and resources and tools he provided me, I reached my goal of fulfilling the present research work.

I would also like to thank the rest of the commitee of my thesis: the professors Dr. Dimitrios Rafailidis and Dr. Georgios Thanos, without the participation and approval of whom, I would have not completed my thesis.

Lastly, I would like to thank my family and friends whose support for my work helped me complete the necessary tasks required to fulfill the requirements of this thesis.

Contents

List of Figures

List of Tables

Abbreviations

ACM	**A**ssociation for **C**omputing **M**achinery
AIF	**A**uthor **I**mpact **F**actor
ANOVA	**A**nalysis **O**f **V**ariance
AI	**A**rtificial **I**ntelligence
ANN	**A**rtificial **N**eural **N**etwork
APS	**A**merican **P**hysical **S**ociety
AUC	**A**rea **U**nder **C**urve
BA	**B**ivariate **A**nalysis
BD	**B**ig **D**ata
BMJ	**B**ritish **M**edical **J**ournal
BP	**B**ack **P**ropagation
CL	**C**onvolutional **L**ayer
CNN	**C**onvolutional **N**eural **N**etwork
CPU	**C**entral **P**rocessing **U**nit
DB	**D**ata **B**ase
DL	**D**eep **L**earning
DOI	**D**igital **O**bject **I**dentifier
DS	**D**ata **S**cience
DT	**D**ecision **T**ree
EDA	**E**xploratory **D**ata **A**nalysis
FC	**F**ully **C**onnected (**L**ayer)
FFNN	**F**eed**f**orward **N**eural **N**etwork
FN	**F**alse **N**egative
FNN	**F**eedback **N**eural **N**etwork
FNN	**F**eedforward **N**eural **N**etwork

FP	**F**alse **P**ositive
FP	**F**ront **P**ropagation
FRNN	**F**ully **R**ecurrent **N**eural **N**etwork
GA	**G**radient **A**scent
GD	**G**radient **D**escent
GNB	**G**aussian **N**aive **B**ayes
GPU	**G**raphics **P**rocessor **U**nit
GRU	**G**ated **R**ecurrent **U**nit
HD	**H**ard **D**rive
HD	**H**ard **D**isk **D**rive
HTML	**H**yper **T**ext **M**arkup **L**anguage
HTTP	**H**yper **T**ext **T**ransfer **P**rotocol
INN	**I**nteractive **N**eural **N**etwork
ISBN	**I**nternational **S**tandard **B**ook **N**umber
ISSN	**I**nternational **S**tandard **S**erial **N**umber
JCR	**J**ournal **C**itation **R**eports
JIF	**J**ournal **I**mpact **F**actor
JN	**J**upyter **N**otebook
kNN	**k**- **N**earest **N**eighbor
LDA	**L**inear **D**iscriminant **A**nalysis
LS	**L**east **S**quares
LSTM	**L**ong **S**hort **T**erm **M**emory
MA	**M**ultivariate **A**nalysis
MAP	**M**aximum **A** **P**riori (Estimation)
ML	**M**achine **L**earning
MLE	**M**aximum **L**ikelihood **E**stimation
MLP	**M**ulti-(ple) **L**ayer **P**erceptron
MSD	**M**ean **S**quared **D**eviation
MSE	**M**ean **S**quared **E**rror
NDA	**N**ormal **D**iscriminant **A**nalysis
NN	**N**eural **N**etwork
OLS	**O**rdinary **L**east **S**quares
OS	**O**perating **S**ystem

PCA	**P**rincipal **C**omponent **A**nalysis
PCIe	**P**eripheral **C**omponent **I**nterconnect **E**xpress
PL	**P**ooling **L**ayer
QDA	**Q**uadratic **D**iscriminant **A**nalysis
RAM	**R**andom **A**ccess **M**emory
ReLU	**Re**ctified **L**inear **U**nit
RF	**R**andom **F**orest
RL	**R**einforcement **L**earning
RNN	**R**ecurrent **N**eural **N**etwork
ROC	**R**eceiver **O**perating **C**urve
RSS	**R**esidual (of) **S**um **S**quares
sd	**s**tandard **d**eviation
SD	**S**teepest **D**escent
SIAM	**S**ociety for **I**ndustrial and **A**pplied **M**athematics
SL	**S**upervised **L**earning
SLP	**S**ingle **L**ayer **P**erceptron
SNN	**S**imulated **N**eural **N**etwork
SSD	**S**olid-**s**tate **D**rive
SSE	**S**um (of) **S**quares **E**rror
SSR	**S**um (of) **S**quares **R**egression
SST	**S**um (of) **S**quares **T**otal
std	**st**andard **d**eviation
SVM	**S**upport **V**ector **M**achine
TN	**T**rue **N**egative
TP	**T**rue **P**ositive
TS	**T**ime **S**eries
UA	**U**nivariate **A**nalysis
UL	**U**nsupervised **L**earning
URL	**U**niform **R**esource **L**ocator
UTH	**U**niversity of **Th**essaly
var	**var**iation
WoS	**W**eb **o**f **S**cience

I would like to dedicate this paper to my parents and two brothers, whose love and support over the years of my studies kept me moving forward and trying to strive for bigger goals.

Thank you for everything.

Chapter 1

Introduction

1.1 Machine Learning and Deep Learning

Along with the improvement of modern computer systems and the reduction of costs of such systems, machine learning has evolved and made a great impact in several financial and scientific fields.

Scientific fields in particular have been greatly benefited by machine learning's field's acceleration of training process and its many emerged methods, that lead to its improvement of accuracy and precision in its applications.

One such method in particular is called deep learning and involves artificial neural networks, which if tested and modified properly can be improved to such a level that far surpasses previously dominating methods in terms of either speed, accuracy and precision or even both.

Based on these, the aim of this thesis is to provide a quick look in the concepts involved in the field of machine and deep learning and to also provide an example of their use in a scientific problem. Specifically, the prediction of scientific papers' citations. In this problem we will demonstrate and compare the use of the most common machine learning methods and that of deep learning which is the use of artificial neural networks.

It should be noted that besides the academic and scientific part of this thesis regarding Big Data, Data Science, Machine Learning and Deep Learning, as well as the implementation of these themes and the code that was written for this purpose, I had to look at previous work regarding citation predicting methods that contained information about them and the various ways the scientific community has already studied them. These sources are included in the bibliography.

1.2 Research Questions

In this thesis, machine learning and deep learning models are used in order to predict papers' citations using data from **APS**[1]. The purpose of this research is to give an insight in the capabilities of such models and compare their results.

Therefore, the main research questions of the present thesis is:

- Is it possible to predict the number of an author's citations using machine learning methods?
- How well can machine learning techniques perform in the task of citation prediction and how do their applications' results vary?
- Can deep learning methods (Neural Networks) outperform the most commonly used machine learning methods?

[1] Americal Physical Society: https://journals.aps.org/

Chapter 2

Concepts and Literature Review

2.1 Basic Concepts

In order to go deeper into the analysis of this paper, we have to first give an introduction to some key concepts and methods of the departments of Data Science, Machine and Deep Learning.

- **Big Data (BD - exactly as defined by IBM):** can be defined as data sets whose size or type is beyond the ability of traditional relational databases to capture, manage and process the data with low latency. Characteristics of big data include high volume, high velocity and high variety and the sources of the data are becoming more complex than those for traditional data.
- **Data Science (DS) [1][7][11]:** combines math and statistics, specialized programming, advanced analytics, artificial intelligence (AI), and machine learning with specific subject matter expertise to uncover actionable insights hidden in an organization's data. These insights can be used to guide decision making and strategic planning. DS methods have been proven especially useful in the past two decades with the explosive emergence of BD.
- **Machine Learning (ML) [1][7][11][14]:** is a branch of AI and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.
- **Unsupervised Learning (UL) [1][7][11]:** uses ML algorithms to analyze and cluster unlabeled datasets. These algorithms discover hidden patterns or data groupings without the need for human intervention. Its ability to discover similarities and differences in

information make it the ideal solution for exploratory data analysis, cross-selling strategies, customer segmentation, and image recognition. Some very well known unsupervised machine learning methods are **clustering**, with hierarchical clustering, k-means, DBSCAN and others, and some less known methods such as **anomaly detection** and **latent variable models**. We will not get further into the topic of unsupervised learning, since our focus will be methods implementing supervised learning, except from some mutual concepts between unsupervised and supervised learning. For further information you can refer to [4][7].

- **Supervised Learning (SL) [1][7][11]:** uses a training set to teach models to yield the desired output. This training dataset includes inputs and correct outputs, which allow the model to learn over time. The algorithm measures its accuracy through the loss function, adjusting until the error has been sufficiently minimized.

- **Reinforcement Learning (RL) [1][7][11]:** is similar to supervised learning, but the algorithm is not trained before its use. This type of learning follows a sequence of trials and errors. A sequence of successful outcomes is reinforced to determine the best course of actions. RL usually works with agents and is typically divided into three (3) approaches: **value-based, policy-based and model-based.** RL will not be discussed in this paper but further information can be found in the bibliography.

- **Deep Learning (DL) [1][7][11]:** is a subset of ML, which is essentially a neural network with three or more layers. These neural networks attempt to simulate the behavior of the human brain allowing it to "learn" from large amounts of data. While a neural network with a single layer can still make approximate predictions, additional hidden layers can help to optimize and refine for accuracy.

- **Neural Networks (NNs) [1][2][4][7][8][11][12][14][16][21][22]:**, also known as artificial neural networks (ANNs) or simulated neural networks (SNNs), are a subset of ML and are at the heart of DL methods. Their name and structure are inspired by the human brain, mimicking the way that biological neurons signal to one another.

2.2 Literature Review

Regarding the sources for the technical aspects of BD, DS, ML, DL and NNs: the huge rise of interest about these topics shows by the abundance of sites discussing them and analyzing them both by a mathematical and programming point of view. Marking should be made for IBMs official site, TowardsDataScience and MachineLearningMastery. Books are also a great source of information and their plethora is certainly of great assistance to everyone

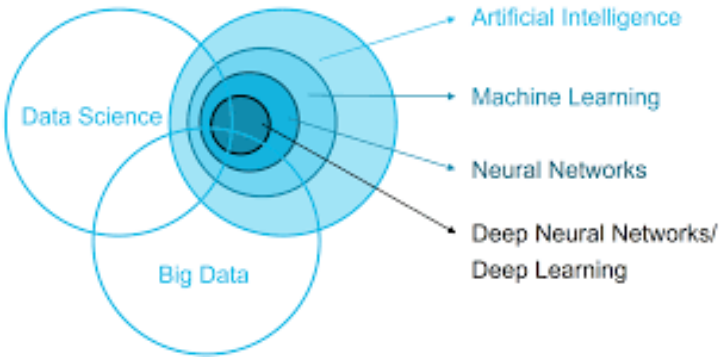

FIGURE 2.1: Summary of relationships between Big Data, Data Science, Machine learning and Deep Learning. Src: LinkedIn

who aspires to learn and work in these fields. I personally want to underline a few of them in particular: *The Hundred-page Machine Learning book [15]*, which is a very powerful tool that can quickly introduce anyone to the concepts discussed in this thesis. *Optimization for Machine Learning: Finding Function Optima with Python [16]*, from which I learned a lot about optimizing the methods and algorithms I discuss later on. *Deep Learning [9]*, which contains the basics and beyond regarding neural networks. Lastly, O'Reily's (David Beazley and Brian K. Jones) *Python Cookbook [25]*, which not only assisted me with ML methods and NNs but also everything regarding python and the tools it has for data and statistical analysis and guides on how to use them.

Regarding citation predicting, what techniques have been used and what I chose to focus on: a lot of work regarding citation predicting actually doesn't focus much on predicting the future citations of papers (and their respective authors) as much as focus on indicators of future performance such as the h-index. In other words most of the work done until now has been about evaluating the current success of authors rather than predicting their future success.

Over the last 20 years there has been a growing number of researchers conducting experiments using ML and DL techniques for citation predicting. Still most of these researches have utilized and put to test only basic (mostly regression) ML models and only a few have explored the true capabilities of DL models and NNs.

Finally, before continuing further with the fundamentals of ML and DL, a few relevant definitions to the theme of citation analysis:

- **Informetrics:** the study of the quantitative aspects of information resources and of the communication of information.
- **Scientometrics:** the field of study which concerns itself with measuring and analysing scholarly literature.

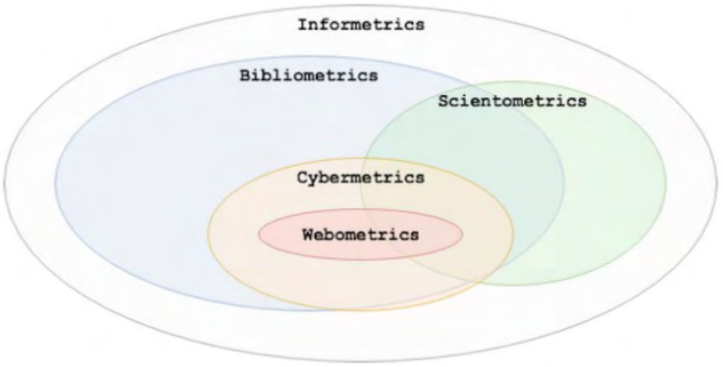

FIGURE 2.2: Relationship between scientific metrics. Src: [55]

- **Bibliometrics:** the use of statistical methods to analyse books, articles and other publications, especially in scientific contents.
- **Cybermetrics:** concerns the study of the quantitative aspects of construction and use information resources, structures and technologies, across the Internet, based on the bibliometric and information approaches.
- **Webometrics:** (also cybermetrics) tries to measure the World Wide Web to get knowledge about the number and types of hyperlinks, structure of the World Wide Web and the use of patterns.

and some metrics used for the assessment of papers' and authors' success and the prediction of citations:

- **Altmetrics:** non-traditional bibliometrics proposed as an alternative or complement to more traditional citation impact metrics, such as impact factor and h-index.
- **Impact factor (IF) or journal impact factor (JIF):** of an academic journal is a scientometric index calculated by Clarivate[1] that reflects the yearly mean number of citations of articles published in the last two years in a given journal, as indexed by Clarivate's Web of Science.
- **h-index or Hirsch index (from its conceptual creator):** an author-level metric that measures both the productivity and citation impact of the publications, initially used for an individual scientist or scholar. An author has an h-index h if h of her/his papers have at least h citations and the rest of the papers have less than h citations each.
- **Contemporary h-index (h_c-index):** corrects for the recentness of the citations, with recent citations carrying more weight. It was proposed by Antonis Sidiropoulos, Dimitrios Katsaros, and Yannis Manolopoulos in their paper *Generalized h-index for disclosing*

[1] https://clarivate.com/

latent facts in citation networks, arXiv:cs.DL/0607066 v1 13 Jul 2006. It adds an age-related weighting to each cited article, giving less weight to older articles (by default this depends on the parameterization).

- h_5**-index:** the h-index for articles published in the last five (5) complete years.
- **g-index:** an author-level metric calculated based on the distribution of citations received by a given researcher's publications, such that given a set of articles ranked in decreasing order of the number of citations that they received, the g-index is the unique largest number such that the top g articles received together at least g^2 citations.
- i_{10}**-index:** created by Google Scholar as an index to rank author impact. Simply, it is the number of publications a researcher has written that have at least 10 citations.
- **PageRank (PR):** is an algorithm used by Google Search to rank web pages in their search engine results. PageRank is a way of measuring the importance of website pages.
- **Eigenfactor score (Metrics Eigenfactor Project - 2008):** measures the number of times articles from the journal published in the past five years have been cited in the Journal Citation Reports (JCR) year.
- **Author Impact Factor (AIF):** the extension of the IF to authors. The AIF of an author A in year t is the average number of citations given by papers published in year t to papers published by A in a period of Δt years before year t.

Chapter 3

Notation & Fundamentals

3.1 Statistics & Machine Learning (ML)

3.1.1 Statistical & Analytical Tools & Definitions

A quick explanation of the various types of variables one might encounter building an ML model or conducting a statistical analysis of a set of data is needed. There are two kinds of variables: **categorical and numeric**.

Categorical variables refer to characteristics that are not quantifiable. It is divided into **Nominal**, variables describing names/labels/categories without order (like gender or types of vehicles), and **Ordinal**, variables whose values are defined by an order relationship (like the rating of applications or the year of attendance in a university). This type of variables usually require **encoding**, in the pre-process stage of development, in order to be used in the mathematical calculations conducted by the machine learning techniques.

Numeric Variables refer to quantifiable characteristics and is divided into **Continuous** and **Discrete**[1]. **Normalization** may be need for this variables to be used if their values are in someway extreme (big variations, extremely low or extremely high values, positive and negative values, outliers).

Data Normalization (or Standardization) [9]: is the act of feature (variable) transformation in order to have values of similar scale[2].

[1]For further information you can refer to https://www.statcan.gc.ca/en/start

[2]Return-to-Scale is a popular method in this regard. Further information can be found at **arXiv:2105.05347 [cs.LG]**

I shall now make a quick mention of the **three (3) types of basic analysis of data**[3] but we will not dive too deep into them because, while some amount of digging into some of these types was conducted on the data that is used in this thesis in the form of **EDA, PCA or preprocessing** (we will look deeper into these later), I focused mostly on the ML aspect of the process and only the most important and usual parts of data analytics needed to proceed in order to not diverge to much from the goal:

- **Univariate Analysis**[4] **(UA) [13]** is the most basic form of data analysis. It assumes a set of data has only one variable a focuses mostly on describing the data while ignoring relationships. In this particular paper the only part of UA that takes place in our code is in the beginning at the initial overview of the data.
- **Bivariate Analysis**[5] **(BA) [13]** is the analysis of bivariate data, used to find the relationship between two sets of values (e.g. X and Y).
- **Multivariate Analysis (MA) [13]** is used to study more complex sets of data than what UA or BA methods can handle (and methods done by hand of course - in this paper no multivariate analysis was conducted since the dataset consists of one feature variable and one outcome variable).

Exploratory Data Analysis (EDA) is the process of performing initial investigations on data in order to identify patterns and anomalies (outliers) and to test hypothesis using summary statistics and graphical representations (which in this case were performed using python's seaborn and matplotlib libraries).

The **arithmetic mean**[6] **(also known as average)** is a measure of central tendency of a finite set of numbers (sum of the values divided by the number of values). The denotation of a set of numbers $x_1, x_2, ..., x_n$ is, typically, $\overline{x}$, also called a **sample mean**. The mean of a series of observations obtained by sampling from a statistical population, is typically denoted as μ or μ_x, and is called **population mean**:

$$\overline{\boldsymbol{x}} = \overline{\boldsymbol{X}} = \frac{\sum_{i=1}^{\boldsymbol{n}} \boldsymbol{x_i}}{\boldsymbol{n}} \tag{3.1}$$

$$\boldsymbol{\mu} = \frac{\sum_{i=1}^{\boldsymbol{N}} \boldsymbol{x_i}}{\boldsymbol{N}} \tag{3.2}$$

[3] More advanced work for this type of job is usually conducted in SAS and IBM's SPSS, and less advanced work but a lot quicker can be done using R, Julia and other programming languages.

[4] 'Uni' translates to 'one' (1).

[5] 'Bi' stems from binary as in 1 (one) and zero (0) or simply: two (2) values.

[6] The mean and the median (50th percentile) of a set of values are called Measures of Central Tendency.

Variance (or var or σ^2) [1][8] is a statistical measurement of the spread between numbers in a data set. Variance measures how far each number in the set is from the mean (average), and thus from every other number in the set:

$$\boldsymbol{\sigma^2 = \frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{N}} \tag{3.3}$$

where:
$\boldsymbol{x_i}$: each value in the dataset
$\overline{\boldsymbol{x}}$: mean of all the values in the dataset
N: number of values in the dataset

Standard Deviation[7] **(or sd or st.d.) [1][8]** is a statistic that measures the dispersion of a dataset relative to its mean and is calculated as the square root of the variance:

$$\textbf{Standard Deviation} = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{n-1}} \tag{3.4}$$

where:
$\boldsymbol{x_i}$: value of the $\boldsymbol{i^{th}}$ point in the dataset
$\overline{\boldsymbol{x}}$: mean value of the dataset
N: number of data points in the dataset

3.1.2 Data Preprocessing

It is said that the **data preprocessing phase** is the most time consuming and challenging part of data science but it is also the most important part. That's because carefully and correctly conducted data cleaning and preparation can help give insights for the data, avoid making mistakes further down the processing line and achieve better and clearer results. That said:

Data Pre-processing [4][5][12] is part of the exploratory data analysis and involves a large number of data processing that should precede the application of machine learning methods in order to assist in aiming for higher scores of accuracy and precision. Such processes are

[7] Low st.d. indicates values tend to be close to the mean. High st.d. indicates the opposite.

data profiling, data cleansing/cleaning, dimensionality reduction, data transformation, data enrichment and validation. These operations deal with duplicate, NULL or NaN (Not-A-Number) or NA (Not-Available) values (missing values in other words) and other issues.

Data Profiling is the process of sifting through the data to determine its legitimacy and quality. In our case no data profiling was needed since the data came from a validated source (the American Physical Society).

Data Cleansing/Cleaning is the process of finding and/or removing duplicate and missing values.

Data Enrichment is the process of combining first party data from internal sources with disparate data from other internal systems or third party data from external sources (this process will not concern us).

Data Validation processes check for the validity of the data. Using a set of rules, it checks whether the data is within the acceptable values defined for the field or not (similar to data profiling).

Dimensionality [11] is the number of columns of data meaning their features/variables. **Dimensionality reduction, or simply dimension reduction [11]** is the process of transforming the data from a high-dimension space into a low-dimension space which is easier to be used while maintaining most of the information of the higher-plane. This part of data preprocessing is essential when dealing with dataset with many feature and outcome variables though in our case was not necessary since the data consist of only one feature and outcome variable. That being said, a note should be made for one of the most common and most effective dimensionality reduction methods, the **Principal Component Analysis (PCA)** which can be broken down to five (5) steps:

1. Normalization of continuous features.
2. Computation of the covariance matrix (in order to identify the correlations between the features of the dataset).
3. Computation of the eigenvectors and eigenvalues of the covariance matrix (step 2) (in order to identify the principal components).
4. Creation of feature vector and elimination of unnecessary principal components.
5. Apply data along the principal component axes.

Finally, **Data Transformation** is the process of converting, cleansing, and structuring data into a usable format that can be analyzed to support decision making processes, and to propel

the growth of an organization. The two most common forms of this type of process is **encoding** and **scaling**.

Encoding is a technique of converting categorical variables into numerical values so that it could be easily fitted to a machine learning model (this type of data transformation can be used but is not required for this particular dataset - depends on the approach taken). There are five (5) main types of encoding:

One-hot/Dummy encoding: creates as many binary dummy variables as the number of different values of a categorical variable. For example a variable named colour with values red, green, blue can be turned into 3 variables name red, green, blue with values zero (0) and one (1).

Label/Ordinal encoding: turns categorical variables to numerical such as the names of players of a football team to certain numbers.

Binary encoding: converts categorical data to numerical and then converting them to binary variables.

Count and Frequency encoding: categorical data are represented by the count of categories. Frequency encoding is the normalized version of count encoding.

Target encoding: encodes the categorical values of the features by using the target value, with the idea that if a feature is important it should be closer to that value.

Feature hashing: represents high-dimensional data (arrays) to smaller (fixed-size) data using hashing functions.

Data Scaling and data normalization are a loose terms that refer to data transformation activities that aim to improve the informational content of the data by adjusting an existing data set so that it conforms with a set of requirements.

- **Scaling:** changing the range of data.
- **Normalization:** changing the shape of the distribution of data.

The most common python/sklearn technique for this kind of data processing are:

- the **MinMaxScaler:** transform features by scaling each feature to a given range and
- the **StandardScaler:** standardize features by removing the mean and scaling to unit variance.

One last quick note should be made for **vectorization**. In the context of high-level languages like Python, Matlab, and R, the term vectorization describes the use of optimized, pre-compiled code written in a low-level language (e.g. C) to perform mathematical operations over a sequence of data. This is done in place of an explicit iteration written in the native language code (e.g. a "for-loop" written in Python). Vectorization allows the elimination of the for-loops in python code. It is especially important in Deep learning as we are dealing with large numbers of datasets. So, it allows the code to run quickly and helps train the algorithms faster. Vectorization is incredibly useful for Natural Language Processing (NLP) which takes an essential, albeit not very big part, in the following experiments.

Now that the basics have been covered, next comes explaining how ML techniques are evaluated, what measures are used to update (and/or penalize) their parameters and upgrade their performance.

3.1.3 Machine Learning Evaluation Tools & Metrics

First, the classified output (result of classification operations - classification and/or prediction of class of elements) is divided into four categories: **True Positives(TP), True Negatives(TN), False Positives(FP), False Negatives(FN)**.

True Positives are the output data that is **correctly** classified and marked as **True (or binary one(1))**.

False Positives are the output data that is **falsely** classified and marked as True (or binary one(1)) when in fact they are supposed to be classified as False (or binary zero(0)).

True Negatives are the output data that is **correctly** classified and marked as **False (or binary zero(0))**.

False Negatives are the output data that is **falsely** classified and marked as False (or binary zero(0)) when in fact they are supposed to be classified as True (or binary one(1)).

These four (4) categories are used in the below metrics:

- **Accuracy [4][5]:** the base metric (most commonly) used for model evaluation, describing the number of correct predictions (TPs, TNs) over all predictions:

$$\textbf{Accuracy} \equiv \frac{\boldsymbol{TP + TN}}{\boldsymbol{TP + FP + TN + FN}} \tag{3.5}$$

- **Precision [4][5]:** a measure of how many of the positive predictions made are correct (it represents reliability):

$$\textbf{Precision} \equiv \frac{\boldsymbol{TP}}{\boldsymbol{TP + FP}} \tag{3.6}$$

- **Recall or Sensitivity [4][5]:** a measure of how many of the positive cases the classifier correctly predicted, over all the positive cases in the data (it represents completeness):

$$\textbf{Recall} \equiv \frac{\boldsymbol{TP}}{\boldsymbol{TP + FN}} \tag{3.7}$$

- **Specificity [4][5]:** a measure of how many negative predictions made are correct (this measure is particularly important for high stakes operations in fields like banking etc.):

$$\textbf{Specificity} \equiv \frac{\boldsymbol{TN}}{\boldsymbol{TN + FP}} \tag{3.8}$$

- **F1-Score: [4][5]** 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. F1-score is, basically, the weighted average of precision and recall[8]:

$$\textbf{F1-score} \equiv \boldsymbol{2 * \frac{Precision * Recall}{Precision + Recall}} \tag{3.9}$$

- **Receiver Operating Curve (ROC) [6]:** a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied ($\boldsymbol{y'y}$ represents TPs rate, $\boldsymbol{x'x}$ represents FPs rate).

- **Area Under the ROC Curve (AUC) [6]:** measures **the entire two-dimensional area underneath the ROC curve (think integral calculus) from (0,0) to (1,1)**. It provides an aggregated measure of performance across all possible classification thresholds. AUC ranges in value from 0 to 1. A model whose predictions are 100% wrong has an AUC of 0, while one whose predictions are 100% correct has an AUC of 1.

[8] Precision and recall are inversely proportional amounts/metrics and thus f1-score is a better overall evaluation metric

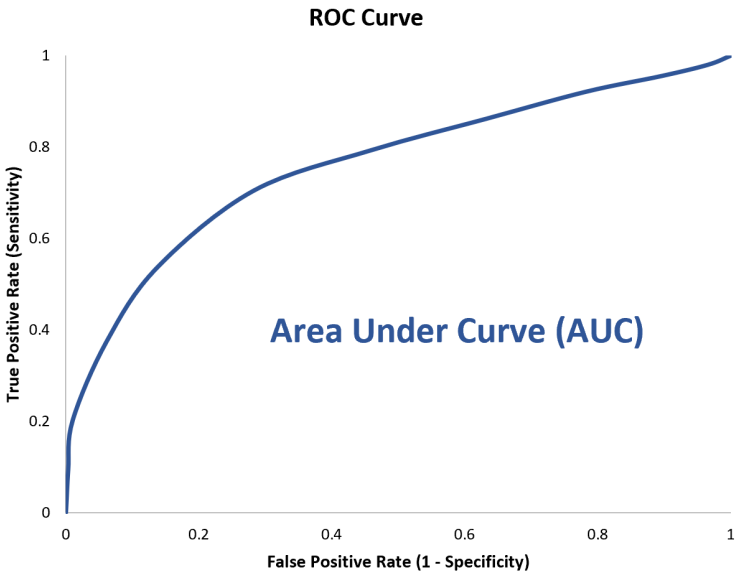

FIGURE 3.1: ROC Curve and AUC example. Src: Google

Now follow the pure arithmetic measures (because they are basically algebraic calculations), that are used more usually for regression (prediction of values - the kind that fit the parameters of this papers theme) than for classification problems:

- **Sum of Squares Total (SST) [3][4]:** the squared differences between the observed dependent variable and its mean:

$$\mathbf{SST} \equiv \sum_{i=1}^{n} (y_i - \overline{y})^2 \tag{3.10}$$

- **Sum of Squares Regression (SSR) [3][4]:** the sum of the differences between the predicted values and the mean of the dependent variable:

$$\mathbf{SSR} \equiv \sum_{i=1}^{n} (\hat{y}_i - \overline{y})^2 \tag{3.11}$$

- **Sum of Squares Error (SSE) or Residual Sum of Squares (RSS) [3][4]:** the difference between the observed value and the predicted value:

$$\mathbf{SSE} \equiv \sum_{i=1}^{n} e_i^2 \tag{3.12}$$

- **Mean Squared Error (MSE) or Mean Squared Deviation (MSD) [7]:** measures the average of the squares of the errors (the average squared difference between the estimated values and the actual value) (it is a **risk function**:

$$\mathbf{MSE} \equiv 1/n * \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{3.13}$$

  or in **matrix notation**:

$$\mathbf{MSE} \equiv \frac{1}{n} \sum_{i=1}^{n} e_i^2 = \frac{1}{n} e^T e \tag{3.14}$$

- **Root Mean Squared Error (RMSE)**: the square root of the MSE:

$$\mathbf{RMSE} \equiv \sqrt{\frac{1}{n} \sum_{i=1}^{n} e_i^2} \tag{3.15}$$

- **Mean Absolute Error (MAE)**: the mean of the absolute value of the error:

$$\mathbf{MAE} \equiv \frac{1}{n} \sum_{i=1}^{n} |\, e_i| \tag{3.16}$$

- **Mean Absolute Percentage Error (MAPE)**: the percentage version of the

$$\mathbf{MAPE} \equiv \frac{1}{n} \sum_{i=1}^{n} \frac{|\, e_i|}{y_i} \tag{3.17}$$

- **Symmetric Mean Absolute Percentage Error (SMAPE)**:

$$\mathbf{SMAPE} \equiv \frac{1}{n} \sum_{i=1}^{n} \frac{|\, e_i|}{\frac{|\, y_i| + |\, \hat{y}_i|}{2}} \tag{3.18}$$

- **Coefficient of Determination or R-squared (R^2):** the **proportion of the variation** of the dependent variable which is predictable from the independent variable:

$$R^2 \equiv 1 - \frac{RSS}{SST} \tag{3.19}$$

3.2 Supervised Learning (SL)

In this part, the most basic and commonly used ML algorithms, some of which are implemented in the experimentation section, are explained.

3.2.1 Linear Regression (LR)

Linear Regression (LR) [1][7]: used to predict the value of a variable based on the value of another variable. The variable you want to predict is called the **dependent (or output or outcome) variable (y)**. The variable you are using to predict the other variable's value is called the **independent (or input or feature) variable (x)**. Even though linear regression is one of the most fundamental methods used in the department of machine learning a quick explanation of its methods of operation will be given.

The goal of this ML algorithm is to build a model that uses a vector $\boldsymbol{x} \in \mathbb{R}^n$ as **input** to predict the value of a scalar $\boldsymbol{y} \in \mathbb{R}$ **(output)**. We denote $\hat{\boldsymbol{y}}$ to be the value predicted for y by said model. The output is calculated as follows[9]:

$$\hat{\boldsymbol{y}} = \boldsymbol{w}^T \boldsymbol{x} \tag{3.20}$$

where $\boldsymbol{w} \in \mathbb{R}^n$ is a **vector of parameters**. We will later refer to them also as **weights**, called that because each parameter $\boldsymbol{w_i}$ is multiplied with a feature $\boldsymbol{x_i}$ before summing up all the results so in a way the parameters determine how much each feature affects the prediction.

Even though the above equation is the basis of the method, the most commonly used form is the one below[10]:

[9] Notice the resemblance between this equation and the simple linear equation: $\boldsymbol{y} = \boldsymbol{ax}$ from simple calculus problems.

[10] Again notice the resemblance between this equation and: $\boldsymbol{y} = \boldsymbol{ax} + \boldsymbol{b}$.

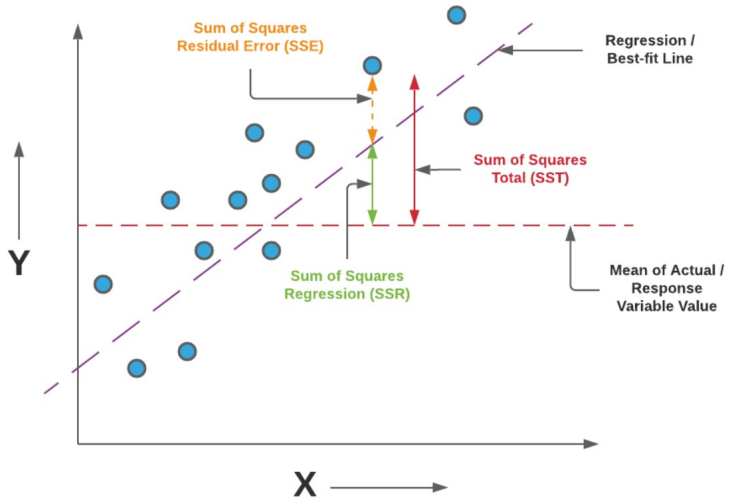

FIGURE 3.2: Linear Regression visualized. Src: VitalFlux - Data Analytics

$$\hat{y} = w^T x + b \tag{3.21}$$

where b is the intercept of the $y'y$ axis, also called bias. With bias, the plot of the model's predictions is still a line but it does not necessarily pass through the origin of the axes.

Three (3) cases can occur when altering the **capacity**[11] of the model.

1. **Under-fitting**: occurs when the model is unable to obtain a sufficiently low error value on the training set. The model is unable to predict (or classify later) values close to that of the dependent variable.

2. **Over-fitting**: occurs when the difference between the training and testing error is too large. While the model is able to predict values very close to the desired outcome it is unable to generalize (meaning if we were to test its accuracy at an unknown set of values of the same dependent variable it will fail almost immediately).

3. **Appropriate Capacity:** the desired case where the model, while not perfectly, is able to output results close to those of the dependent variable and is also able to achieve likewise results if tested in an unknown (not trained at yet) dataset of the same variables.

Both linear and logistic regression require optimization to find the optimal coefficients that result in great accuracy and precision. Next we will discuss one of the most common methods of optimization for the linear regression.

[11]The 'Capacity' of a model is an informal term used to describe the complexity of the model, where complexity refers to the relationships between the variables of the model, the neural net's neurons etc. .

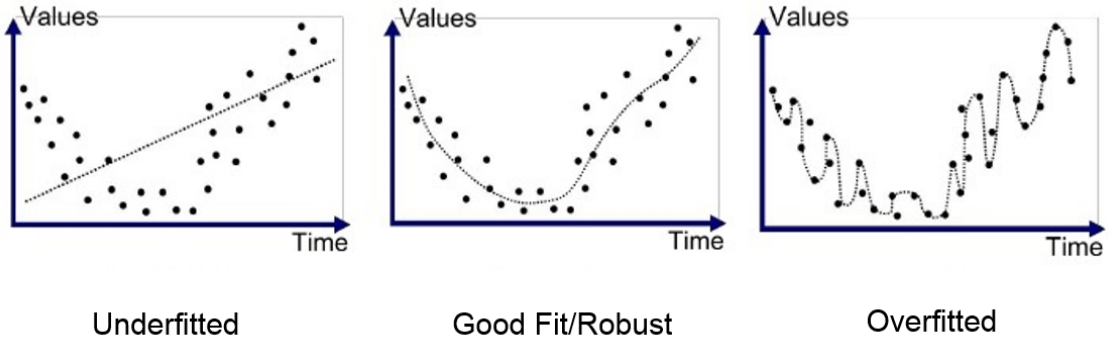

FIGURE 3.3: Fitting based on different types of model capacity. Src: Google

3.2.2 K-Nearest Neighbors (kNN)

The k-Nearest Neighbors (kNN) [1][5][11][14][16]: is a non-parametric classifier, which uses proximity (of variables' values) as a metric to make predictions or classifications. It works with the notion that similar points/nodes/values can be found close to each other.

For regression problems, the output of kNN regression is **the property value of the object**. That is the average of the values of the k-nearest neighbors.

For classification problems, the output is a **class membership**. That is, the popularity of the vote of the k-nearest neighbors is assigned to the object opted to be classified.

Whatever the problem is, in order to compute the kNN one must choose from a variety of distance metrics:

Euclidean Distance: the most commonly used distance metric, limited to vectors of real values. It measures a straight line between two points (useful for continuous variables - commonly used for regression problems).

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2} \tag{3.22}$$

Manhattan Distance (or taxicab or city block distance): another metric that measures the absolute value between two points (also commonly used for regression problems).

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i| \tag{3.23}$$

Minkowski Distance: a generalized form of the above two metrics (Euclidean and Minkowski distances - also commonly used for regression problems).

$$d(x, y) = (\sum_{i=1}^{n} |x_i - y_i|)^{\frac{1}{p}} \tag{3.24}$$

where **p**: parameter that allows for the creation of other distance metrics (if $p = 2$, $Minkowski \equiv Euclidean$, if $p = 1$, $Minkowski \equiv Manhattan$).

Hamming Distance (or overlap metric): a metric that can be used with discrete values (Boolean, string vector etc. - commonly used for classification problems).

$$D_H = \sum_{i=1}^{k} |x_i - y_i| \tag{3.25}$$

where:
$x = y \rightarrow D = 0$
$x \neq y \rightarrow D \neq 1$

Of course, one of the most important parts of implementing kNN is **defining k**. For **k=1**, the algorithm will simply assign to an object the same class as its closest neighbor. As k increases there is a trade off between high variance and low bias and the opposite. If the data used contain many outliers or noise then it is generally prefered to choose higher values of k. With cross-validation techniques one can find the optimal k. As for classification problems in particular, it is almost necessary to use odd values for k in order to avoid ties.

3.2.3 Decision Trees (DT)

Decision Trees (DT) [5][7][11][14][16]: are drawn upside down with the root at the top. A DT consists of three (3) types of nodes. Its acmes, where the tree splits, are called **branches/edges** and they represent the **chance nodes**. The conditions on which the tree splits are called **conditions/internal nodes** and they represent the **decision nodes**. Finally, the end of the branches, where the splitting stops are called **decisions/leafs** and they belong to the last type of nodes, the **end nodes**.

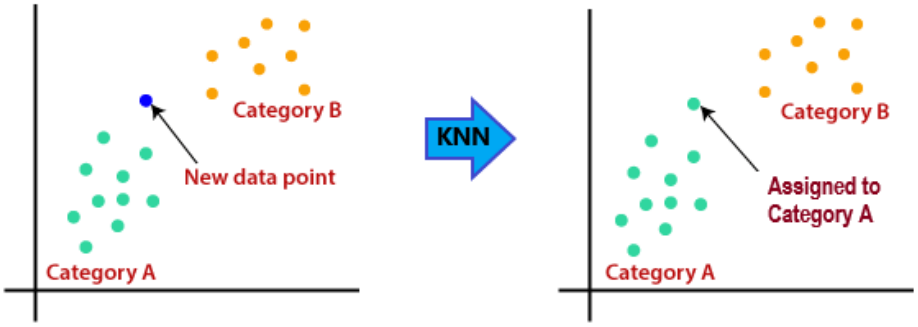

FIGURE 3.4: Illustration of kNN. Src: Medium

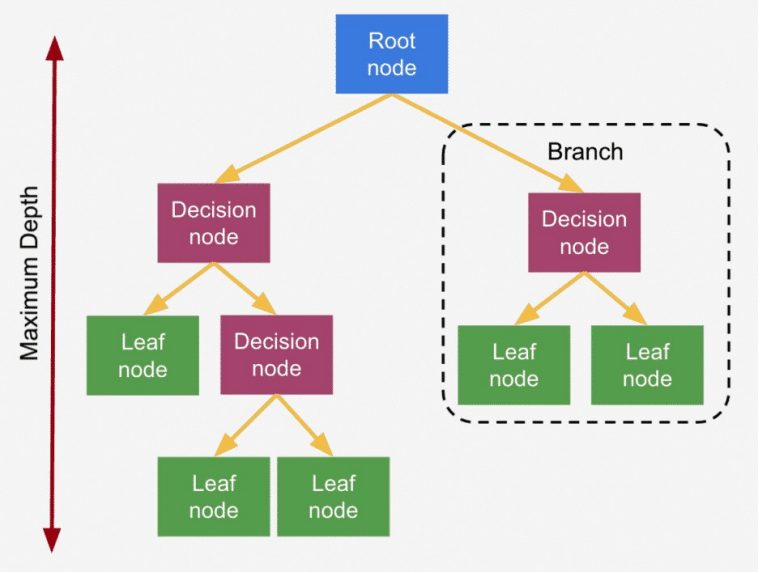

FIGURE 3.5: Illustration of a DT's structure. Src: JC Chouinard - Google

Regression Trees predict continuous values like house prices. **Classification Trees** aim to classify objects, like passengers of the titanic as deceased or survived (a well known example used in DTs and ML in general).

In a DT, the data are divided into groups. Before each split, we calculate the cost of the accuracy each split causes. This is done using cost functions. That means that for each split the rule that has the lowest cause is chosen (meaning the root node is the best predictor/classifier).

Thus it becomes clear choosing a good splitting function is important in order to acquire good results. The two most commonly used functions are **information gain (I_g)** and the **phi-function (Φ)**.

Information gain (I_g) is known as a measure of "reduction in entropy":

$$I_g(s) = H(t) - H(s,t) \tag{3.26}$$

Entropy - Shannon entropy: of a random variable is the average level of information/surprise/uncertainty inherent to the variable's possible outcomes:

$$H(X) \equiv -\sum_{x \in X} p(x) \log p(x) \tag{3.27}$$

where:
X: a discrete random variable which is distributed according to: $p : X \rightarrow [0, 1]$

Phi-function (Φ) is known as a measure of 'goodness' of a candidate split at a node:

$$\Phi(s,t) = (2 * P_L * P_R) * Q(s|t) \tag{3.28}$$

DTs have many advantages like their simplicity and its "white box" model structure but it also has a few disadvantages like their instability (meaning a small change in the data can lead to a large change in their structure) and their tendency to overfit. This last problem finds its solution in the next model (Random Forests[12]).

3.2.4 Random (Decision) Forests (RF)

Random (Decision) Forests (RF) [5][11][14]: is basically an ensemble[13] learning method for regression and classification that constructs multiple decision trees at the same time of training. For regression problems, the output is derived from the average/mean of the individual trees. For classification problems, the output is derived by vote majority from the DTs. RFs solve some of DTs' problems (although that does not mean that it always does so with greater accuracy, as stated previously). However, data characteristics play a major rule in their overall outcomes.

[12] While RFs can perform better and solve some of DTs' problems their accuracy tend to be lower than **Gradient Boosted Trees**. Further information about gradient boosted trees can be found at most of the sources related with DTs.

[13] Ensemble methods are techniques that create multiple models and then combine them to produce improved results.

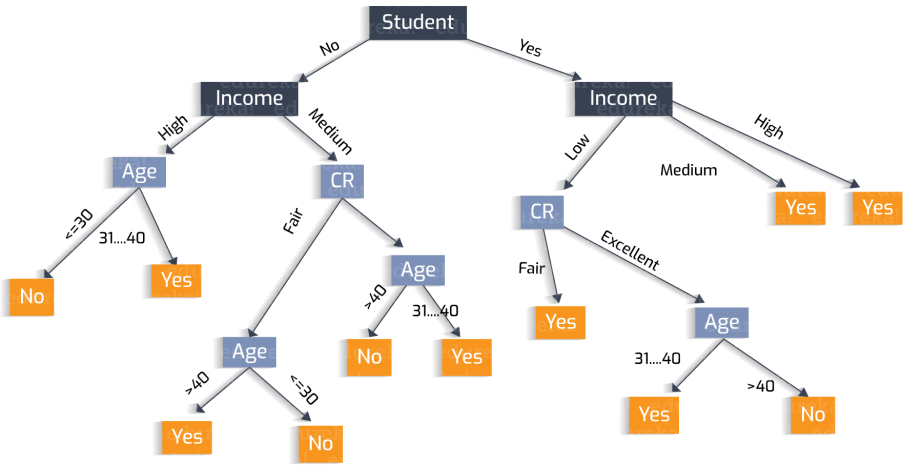

FIGURE 3.6: Illustration of an exemplary DT. Src: AquaVistaHotels - Google

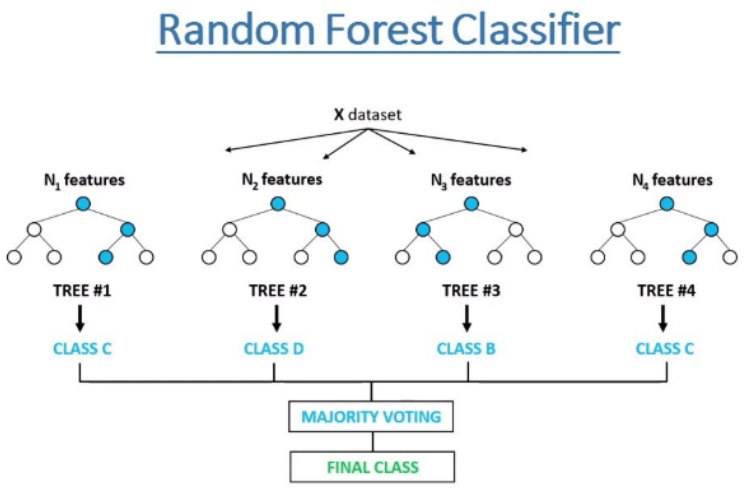

FIGURE 3.7: Illustration of a Random Forest. Src: freeCodeCamp

RFs' greatest advantage used against the nature of DTs is **bagging (bootstrap aggregation)**. Bagging is the solution against DTs' high sensitivity to data characteristics. Bagging allows each individual tree (of the forest) choose a sample of data randomly with replacement which enables the diversity of each tree.

As we can see in figure [3.7], the RF's tree number one (1) 'sees' features two (2) and three (3), while number two (2) 'sees' features one (1) and three (3). This assists the diversity of the decisions of the trees since, not only the trees are trained on different random datasets but also use different features to make decisions.

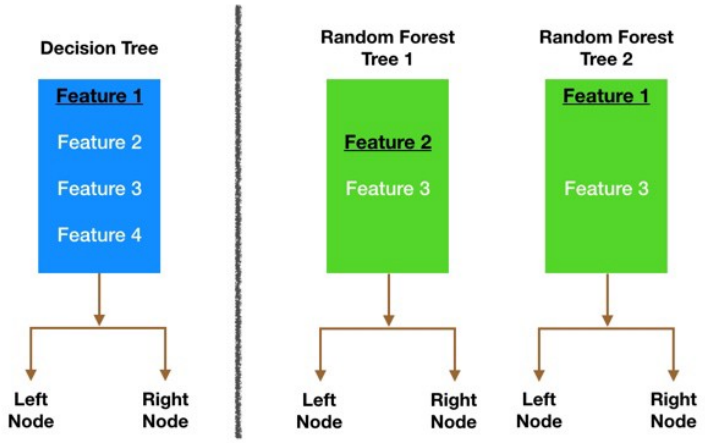

FIGURE 3.8: Example of RF's diversity. Src: TowardsDataScience

3.2.5 Least Squares (LS)

The **Least Squares (LS)** method is a statistical procedure (a form of regression) used to find the **line of best fit**. Each data point in said line represents the relationship between the independent(s) and the dependent variable.

- The line of best fit is the line that minimizes the vertical distance from the data points to the regression line.
- The term least square stems from the smallest sum of squares error (SSE).
- **Ordinary Least Squares (OLS)** is the most well known type of least squares methods.

OLS is best described as a common technique for estimating coefficients of linear regression equations which describe the relationship between one or more independent quantitative variables and a dependent variable (simple or multiple linear regression). As such, it is closely related but is not the same as linear regression (though for this particular paper it was not implemented since I chose to use only linear regression as a representational algorithm for both).

The rest of the algorithms analysed aren't used in the experimentational part of this paper since they are either used for solving classification problems (Logistic Regression, GNB) or were too slow to be used (SVM). That being said since one of the purposes of this paper is studying the basics of DS and ML, I thought it was necessary to make a note of this algorithms too.

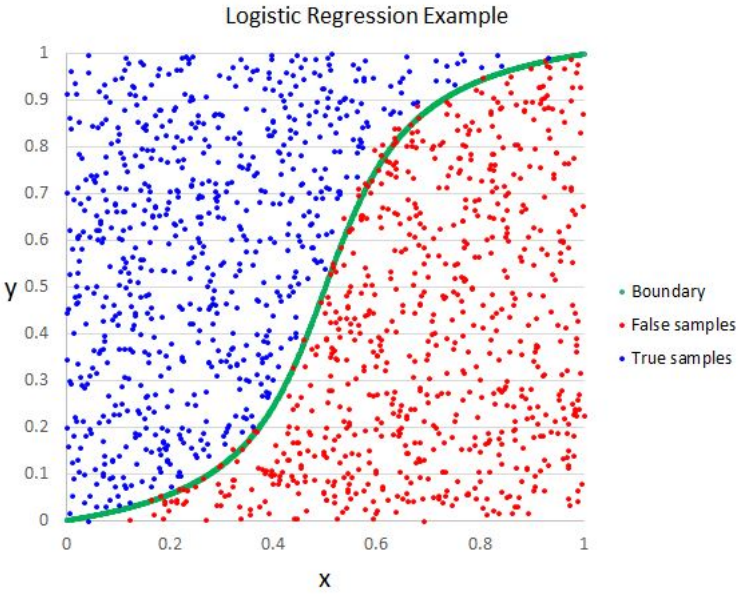

FIGURE 3.9: Logistic Regression visualized. Src: AI wiki

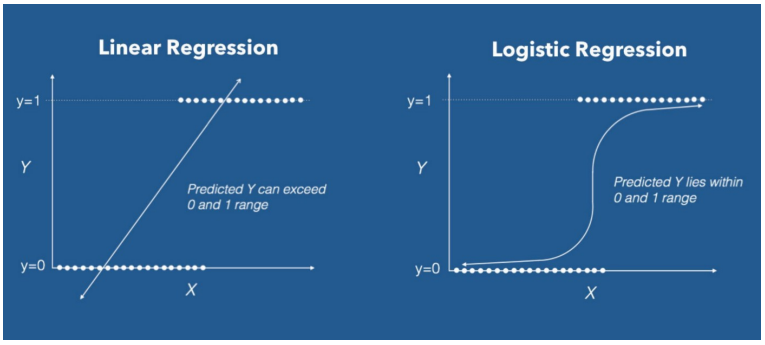

FIGURE 3.10: Fundamental difference of Linear and Logistic Regression. Src: Medium

3.2.6 Logistic Regression

Logistic Regression (also called LR) [1][14][16]: is a type of statistical model (also known as logit model) often used for classification and predictive analytics. Logistic regression estimates the probability of an event occurring based on a given dataset of independent variables. Since the outcome is a probability, the **dependent variable is bounded between 0 and 1**. In logistic regression, a logit transformation is applied on the odds—that is, the probability of success divided by the probability of failure. This is also commonly known as the log odds or the natural logarithm of odds.

Quick note about the name of this method: Logistic regression is not a regression algorithm but a classification one. Its name came from the fact that the mathematical formulation of this method is a lot similar to that of linear regression's.

The seed for the idea of this algorithm came from the fact that scientist that performed manual calculations (not computer assisted), wanted to find a linear model for classification models. That led to the realization that one could assign, for a binary problem, zero (0) as a negative label and one (1) for a positive one (meaning 0 equals that an object is not classified as a particular class and 1 equals an object is classified as that class). As a continuation of that, if one was to divide the domain of (0,1) to two (2) classes, then values closer to zero (0) would mean that they would be classified as 'A' and values closer to one (1) would be classified as 'B' (this means we don't really need absolute 0 or 1). One very famous function that does exactly that is the **(standard) logistic function or sigmoid function**:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.29}$$

where:
e: the base of the the natural logarithm or Euler's number

If we apply the equation [3.21] from linear regression in the above formula, we get:

$$f_{w,b}(x) = \frac{1}{1 + e^{-(wx+b)}} \tag{3.30}$$

Of course, just like most of the methods analyzed in this thesis, it also requires optimization in order to find the best **w** and **b**. For linear regression the goal of the optimization was to minimize the mean squared error (MSE). In the case of logistic regression the optimization criterion is **Maximum Likelihood**[14], where instead of minimization problem of the average loss, we come across a maximization problem of the likelihood of the training data in one model:

$$L_{w,b} = \prod_{i=1\ldots N} f_{w,b}(x_i)^{y_i}(1 - f_{w,b}(x_i))^{(1-y_i)} \tag{3.31}$$

Now, because of the e^{-x} or *exp(-x)* function in the model it is proven to be of better convenience the maximization of the **log-likelihood**:

[14] If $y = 1$ then $L_{w,b} = f_{w,b}(x_i)$ and if $y = 0$ then $L_{w,b} = 1 - f_{w,b}(x_i)$.

$$\boldsymbol{LogL_{w,b} = ln(L_{w,b}(x)) = \sum_{i=1}^{N} y_i ln(f_{w,b}(x)) + (1 - y_i) ln(1 - f_{w,b}(x))} \quad (3.32)$$

In contrast with linear regression, logistic regression has no closed form solution, meaning there are several different ways to approach the above optimization problem (usually gradient descent is chosen but more on that later).

This algorithm was not used in the experimentation section.

3.2.7 Gaussian Naive Bayes (GNB)

Naive Bayes classifiers [5][11][18] are a group of ML classification algorithms based on the **Bayes Theorem** that are especially useful when dealing with input data of high dimensionality.

Bayes Theorem (or Law or Rule) [11][18]:

$$\boldsymbol{P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B|A)P(A)}{P(B)}} \quad (3.33)$$

where **A,B are events and $P(B) \neq 0$** and:
P(A): the probability of A occuring (marginal or prior probability)
P(B): the probability of B occuring (marginal or prior probability)
P($A|B$)[15]: the probability of A given B (conditional or posterior probability)
P($B|A$)[16]: the probability of B given A
P($A \cap B$): the probability of A and B both occuring

A theoretical view of the above equation can be written as follows:

$$\textbf{Posterior} = \frac{\textbf{Likelihood} * \textbf{Prior}}{\textbf{Evidence}} \quad (3.34)$$

[15]It is also called the likelihood of B given a fixed A, P($A|B$) = L($B|A$).
[16]It is also called the likelihood of A given a fixed B, P($B|A$) = L($A|B$).

When dealing with continuous data, there is usually the assumption that the values follow the **normal (Gaussian) distribution** (even when that is not the case, normalization of the data usually takes place). Conforming with this idea, the **probability density of a certain/specific objervation v** is assumed to be:

$$p(x = v|C_k) = \frac{1}{\sqrt{2\pi\sigma_k}} e^{-\frac{(v-\mu_k)^2}{2\sigma_k^2}} \tag{3.35}$$

where:
x: continuous feature
μ_k: mean of values associated with class C_k
σ_k^2: Bessel[17] corrected variance in x associated with class C_k

or in simpler terms, the probability of observing point x_i given parameters θ is:

$$P(x_i|\theta) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}} \tag{3.36}$$

and for observing all points:

$$P(X|\theta) = \prod_{i=1}^{n} P(x_i|\theta) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}} \tag{3.37}$$

Now using **Maximum Likelihood Estimation (MLE)** we want to find the parameters $\theta = (\mu, \sigma)$ that maximize the probability P($X|\theta$). The **Likelihood** function is:

$$L(\theta) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}} \tag{3.38}$$

or as we previously similarly mentioned, the **Log-Likelihood** function is:

[17] uses n-1 instead of n in its formula, where n is the number of objervations in a sample.

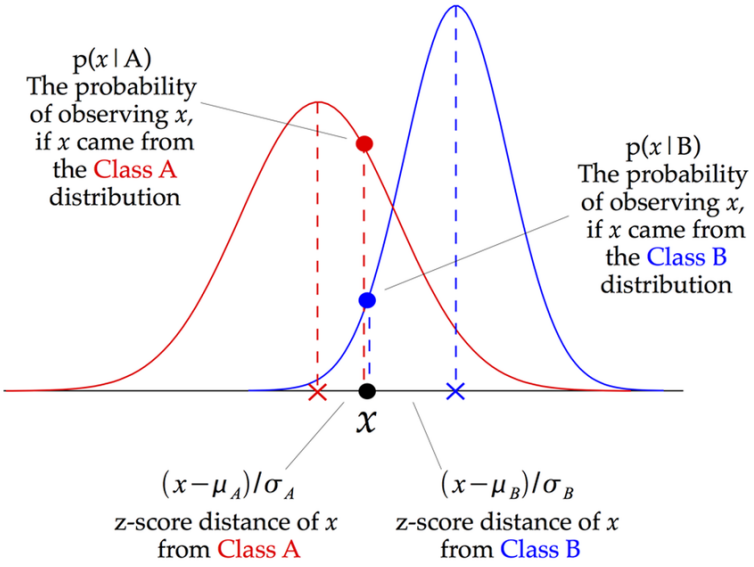

FIGURE 3.11: Illustration of how a GNB Classifier works. Src: ResearchGate

$$LL(\theta) = -\sum_{i=1}^{n} \frac{(x_i - \mu)^2}{2\sigma^2} - \frac{1}{2} n \log 2\pi - n \log \sigma \tag{3.39}$$

We want to find the parameters μ, σ that maximize the above LL(θ) equation.

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i = \mu_x \quad \textbf{(Sample Mean}^{18}\textbf{)} \quad \sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2 = \sigma_x^2 \quad \textbf{(Sample Variance)} \tag{3.40}$$

Alternatively, you can use another method like the **Maximum A Priory (MAP) Estimation**:

$$P(\theta|x) = \frac{P(x|\theta)P(\theta)}{P(x)} = \frac{P(x|\theta)P(\theta)}{\sum_{\Theta} P(x|\theta)P(\theta)} \tag{3.41}$$

where we want to determine θ that maximizes P($\theta|x$):

$$argmax_{\theta} P(\theta|x) = argmax_{\theta} \frac{P(x|\theta)P(\theta)}{\sum_{\Theta} P(x|\theta)P(\theta)} \tag{3.42}$$

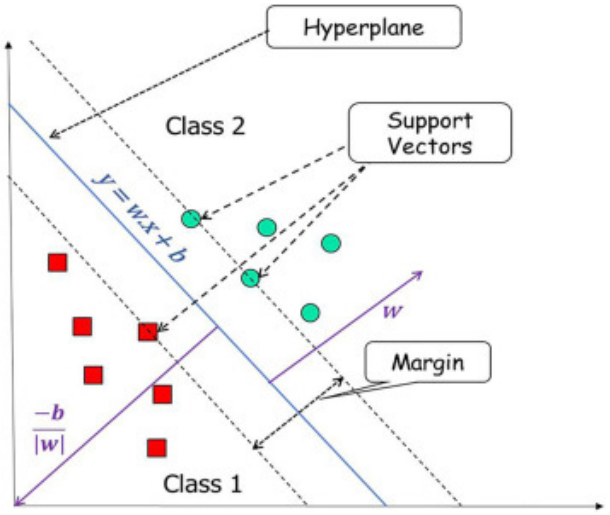

FIGURE 3.12: Support-vector Machine illustration. Src: ScienceDirect

This algorithm was not used in the experimentation section.

3.2.8 Support-Vector Machine (SVM)

Support-Vector Machines (SVMs, support-vector networks) [4][5][7][11][14][19]: are models of supervised learning used to assist in data classification and regression analysis[19]. SVMs belong to the category of algorithms which model the classification rule directly.

SVMs' objective is to find a **hyperplane** in an N-dimensional space, where N equals the number of features, that distinctly classifies the data points. For a 2-dimensional problem, meaning a classification problem that deals with two (2) features, the hyperplane would be a line. For a 3-dimensional problem, with three (3) features the hyperplane would be a plane. And so forth, with 4 or more features (though that hyperplane would be difficult to imagine).

Linear SVM: for a given a dataset of n points $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$ where y equals one (1) or minus one (-1) (in this algorithm the division of classes is in the [-1, 1] space instead of [0, 1]), indicating which class x_i belongs to. Each x_i is a $\mathbb{R}^p$ vector and the goal is to find the **maximum-margin hyperplane (seek figure [])** that divides the groups of points x_i for which $y_i = -1$ or $y_i = 1$. Any hyperplane can be written as:

$$w^T x - b = 0 \tag{3.43}$$

[19]For unlabeled data, SVM models must adapt to the unsupervised learning kind of approach. This leads to another type of model suited for this kind of data (the idea for which came in part from the lead creator of original SVM model): **Support-vector clustering**

where:
w: the normal vector to the hyperplane and
the parameter $\frac{b}{\|w\|}$ determines the offset of the hyperplane

Or in more detail:

Equation for a 2-D hyperplane:

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 = 0 \tag{3.44}$$

Equation for a p-D hyperplane:

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p = 0 \tag{3.45}$$

where $x = [x_1, x_2, ..., x_p]^T$ is the feature vector and the classification can be made considering the classes are defined by:

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p > 0 \tag{3.46}$$

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_p x_p < 0 \tag{3.47}$$

For linear classifications the use of this algorithm is pretty easy. In order to perform non-linear classifications though, SVMs use what is called the **Kernel Trick [5][11]**.

The Kernel trick: its name came from the use of kernel functions[20], which enables algorithms to operate in a **high-dimensional, implicit feature space**. That means, instead of computing the coordinates of the data, it simply calculates the **inner products** between all the pairs of the data in the input space (which make the 'trick' significantly cheaper).

"In machine learning, a kernel refers to a method that allows us to apply linear classifiers to non-linear problems by mapping non-linear data into a higher-dimensional space without

[20]There is not really a scientific consensus for which kernel function performs better overall or even in specific types of data input.

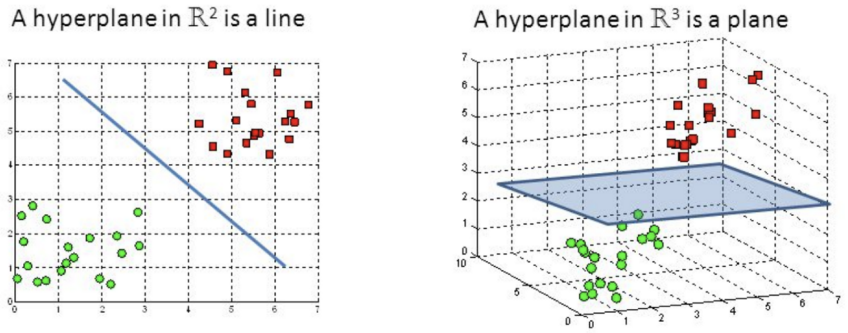

FIGURE 3.13: Example of dimensional space difference for SVMs. Src: TowardsDataScience

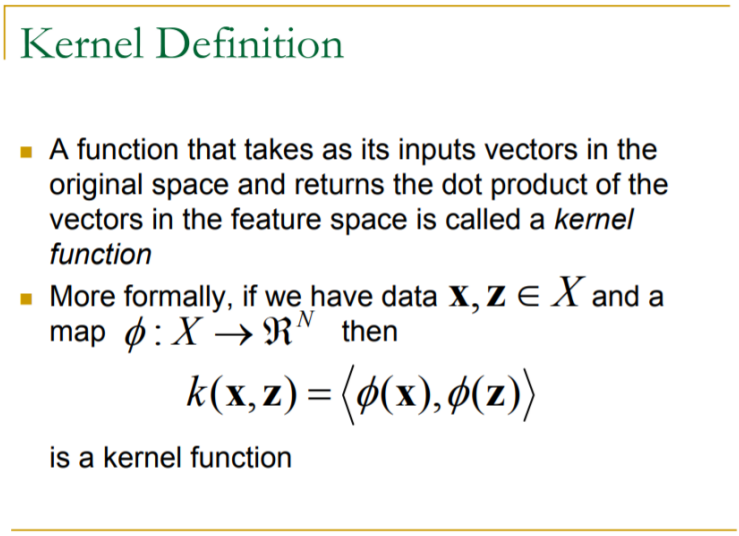

FIGURE 3.14: Definition of Kernel functions. Src: TowardsDataScience

the need to visit or understand that higher-dimensional space." **-Programmathically.com [20]**

Support Vector Regression (SVR): is a type of machine learning algorithm used for regression analysis. The goal of SVR is to find a function that approximates the relationship between the input variables and a continuous target variable, while minimizing the prediction error. Unlike Support Vector Machines (SVMs) used for classification tasks, SVR seeks to find a hyperplane that best fits the data points in a continuous space. This is achieved by mapping the input variables to a high-dimensional feature space and finding the hyperplane that maximizes the margin (distance) between the hyperplane and the closest data points, while also minimizing the prediction error.

This algorithm (SVR) was not used in the experimentation section because it was too slow. Now that the basics of ML and SL in particular have been discussed it's time we dive further into ML, in what is called DL.

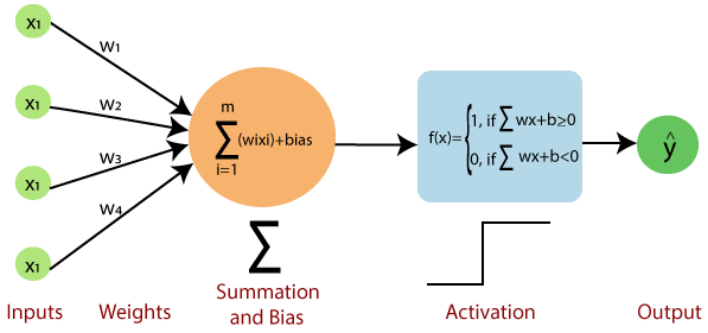

FIGURE 3.15: Illustation of a perceptron. Src: Medium

3.3 Artificial Neural Networks (ANNs)

I will now make a theoretical introduction of the element that is at the center of this papers theme and which is the foundational building block in the field of DL. That is, the **Artifical Neural Networks (ANNs).**

3.3.1 Feedforward Neural Network (FNN) & Single-layer Perceptron (SLP)

Feedforward Neural Network (FNN or FFNN): the simplest form of ANN where the connections between the nodes of all layers do not form a circle (the information flows in only one (1) direction).

The simplest kind of FNN is the **single-layer perceptron (network)** in which the inputs are fed directly to a single layer of output nodes via a set of weights. The sum of the products of the inputs and weights **(net input)** is calculated in each node and is then passed through an activation function (typically there is also bias (b) involved). This process is called **forward propagation**. Of course, SLPs can be useful but they have one big limitation, they cannot be applied to non-linear data.

Activation or Transfer functions define the output of the nodes given an (or a set of) input(s). They are categorized[21] as **ridge functions, radial (basis) functions (RBFs) and fold functions**.

Some of the most widely known and used functions are:

(Binary) step:

[21] I could deepen further into the topic of these categories but wikipedia (wikipedia.org/wiki/) has very comprehensive explanations for the categories as well as summaries for the most well known activation functions.

$$f(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases} \tag{3.48}$$

Logistic/sigmoid/logistic sigmoid function (log-sig/log-sigmoid):

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.49}$$

Hyperbolic tangent (tanh - helps with overcoming the vanishing gradient problem):

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{3.50}$$

Rectified linear unit (ReLU - Ramp function - used most often in the hidden layers of a NN):

$$f(x) = x^{+} = max(0, x) = \begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases} \tag{3.51}$$

As well as **Identity (f(x)=x), Gaussian, Softmax, Softplus, Leaky ReLU function and more**.

3.3.2 Multi-layer Perceptron (MLP)

Multi-layer perceptrons (MLPs) tackle the SLPs' limitation of linear-only use (this type is the basis of the model created for this paper). MLPs have input and output layers as well as **hidden layers (layer between input and output layers/nodes)** of neurons stacked together. Another advantage the single-layer perceptrons lack is that while they are limited to activation functions with thresholds (sigmoid, ReLU etc.), MLPs can use any arbitrary activation function instead.

Now, one could ask: since MLPs belong under the 'umbrella' of FNNs, how do they 'adapt' ('learn'). The answer is **backpropagation**.

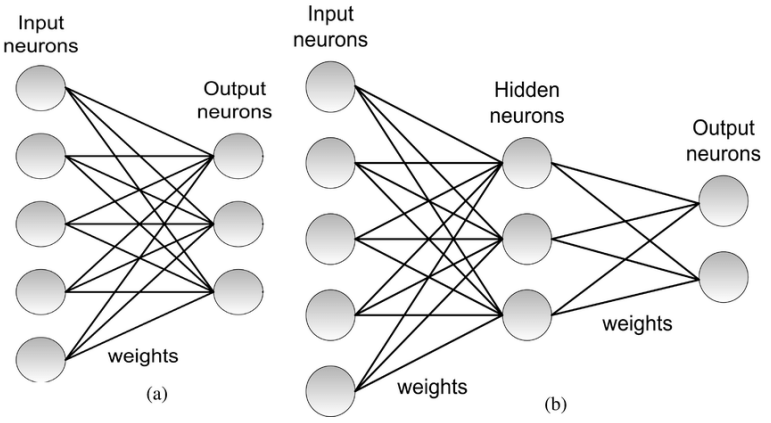

FIGURE 3.16: Demonstration of difference between single- (a) and multi-layer (b) perceptrons. Src: ResearchGate

Backpropagation (BP) [2][4][5][7][11][14][16][22]: is the mechanism MLPs use in order to adjust the weights in the network through iterations and whose goal is to minimize the cost function. BP has a take-it-or-leave-it requirement. That is that the functions that combine inputs and weights (e.g. weighted sum) and the threshold functions used (e.g. ReLU) must be differentiable. That's because in most MLPs' the optimization function used is **Gradient Descent** which requires from the aforementioned functions to have a **bounded derivative**.

In theory the actual method used is the **method of steepest descent or saddle-point method**, which is an extension of Laplace's method where one deforms a contour integral in the complex plane to pass near a stationary point (saddle point), in roughly the direction of steepest descent or stationary phase. In reality, though, we use the gradient descent method (which also sometimes is called by its parent method) which is an optimized algorithm.

Gradient Descent (GD) or Steepest Descent (SD) is a first-order iterative optimization algorithm for finding a local minimum of a differentiable function. The basic idea is to follow the opposite direction of the gradient of the function at a given current point, through a series of repeated steps. The opposite of this concept is called **Gradient Ascent (GA)** and is a technique used to find a local maximum of the same function. GD's equation can be written as follows:

$$\boldsymbol{a_{n+1} = a_n - \gamma \nabla (F(a_n))} \tag{3.52}$$

where:

$\boldsymbol{a}$: a point

$\boldsymbol{F(a)}$: a differential multi-variable function

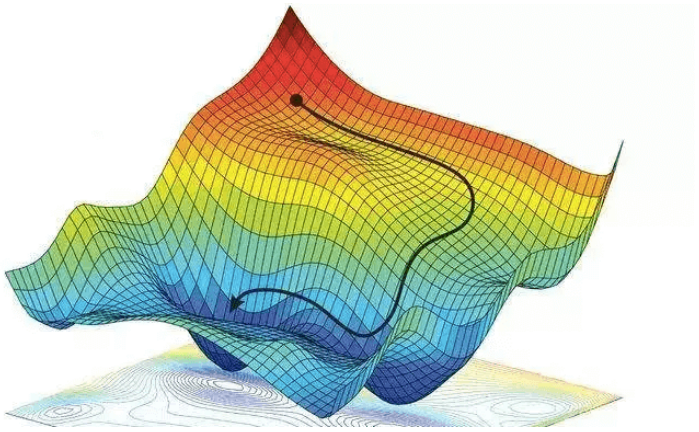

FIGURE 3.17: Visualization of GD in action. Src: easyai.tech - Google

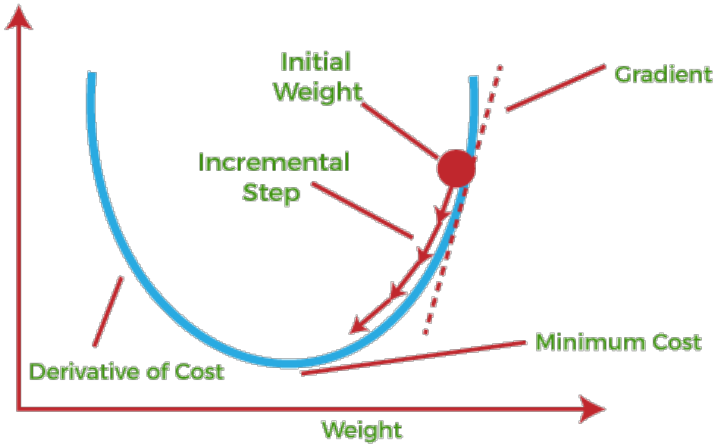

FIGURE 3.18: Graphical representation of GD. Src: Javatpoint - Google

γ: the learning rate[22] (the step we take toward the opposite direction of the local maximum

The above can be translated to:

$$x_{n+1} = x_n - \alpha_n \nabla(F(x_n)) \tag{3.53}$$

where $n \geq 0$:

x_0: an initial guess for a local minimum of F

$F(x)$: same as above

α: the learning rate which is computed by:

[22]Learning rate is a **hyper-parameter**, a configuration variable that is external to the model and whose value is not estimated by the given data.

$$a_n = \gamma_n = \frac{|(x_n - x_{n-1})^T[\nabla F(x_n) - \nabla F(x_{n-1}]|}{\|\nabla F(x_n) - \nabla F(x_{n-1})\|^2} \tag{3.54}$$

This equation provides a monotonic sequence where:

$$F(x_0) \geq F(x_1) \geq F(x_2) \geq ... \tag{3.55}$$

Using the formulas above, the weights then are update as follows:

$$\Delta_w(t) = -b\frac{\partial E}{\partial w_{(t)}} + \alpha\Delta_{w(t-1)} \tag{3.56}$$

where:
b[23]: bias
E: Error
α: learning rate
$\Delta_w(t)$: gradient at current iteration
$w_{(t)}$: weight vector
$\Delta_{w(t-1)}$: gradient at previous iteration

Choosing the right learning rate can have significant impacts both at the outcome of GD and the time it takes to execute it.

But a question now arises: **how does backpropagation propagate its calculations through the multiple layers of a NN?** The answer comes from the **chain rule [4][5][7][11][12][14][16][23]**. The chain rule's formula expresses the derivative of the composition of two (2) (of course it's also used for more than two (2)) differentiable functions. For example, for $h(x) = f(g(x))$, where f, g are differentiable functions $h(x)$ can be written as:

$$h'(x) = f'(g(x))g'(x) \tag{3.57}$$

[23]Remember: bias is decided by the designer of the NN, it is usually a small number used to add the element of randomness to that design.

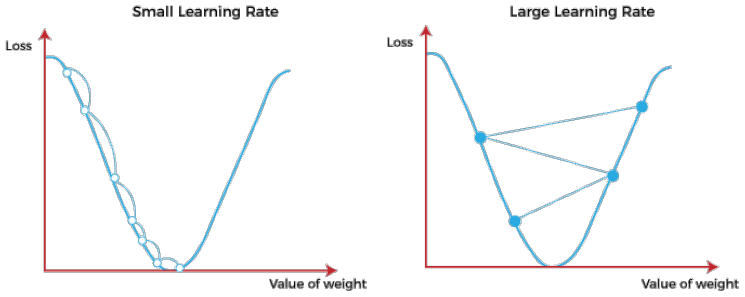

FIGURE 3.19: Visual presentation of the impact of choosing different learning rates. Src: Javatpoint - Google

This, in turn, can be translated for using instead of three (3) functions three (3) variables, one (1) independent (x) and two (2) dependent (y, z - where y depends on x, z depends on y and z depends on x via the intermidiate y). The chain rule can now be written as follows:

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx} \tag{3.58}$$

These two (2) formulas are applied to the two (2) formulas of backpropagation, allowing the NN to propagate its layers' results through its network.

3.3.3 Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) or Feedback Neural Networks (also called FNNs) or Interactive Neural Networks (INNs): are NNs designed especially for analyzing time series data, event history or temporal ordering.

RNNs have connections between the nodes of their structure, creating a cycle which allows output from some of these nodes to affect the next input of the same nodes (of themselves or others). They are based on the FFNNs[24] for which we talked about earlier. **Backpropagation through time or BPTT** is a gradient-based technique for training certain types of RNNs.

For an activation function f, one can predict the output at time step k+1 if she/he unfolds the network for k time steps:

$$h_{t+1} = f(x_t, h_t, w_x, w_h, b_h) = f(w_x x_t + w_h h_t + b_h) \tag{3.59}$$

[24]Both feedforward and feedback neural networks use the abbreviation 'FNN' so for the first kind we will use 'FFNN'. They are distinguished from the context in which the abbreviation is used.

The output y at time t:

$$y_t = f(h_t, w_y) = f(w_y \cdot h_t + b_y) \tag{3.60}$$

where:
x_t: input at time step t
y_t: output at time step t
h_t: vector that stores the values of the hidden units at time t (initial value $h_0 = 0$)
w_x: weights of inputs in recurrent layer
w_h: weights of hidden units in recurrent layer
w_y: weights of hidden to output units
b_h: bias of recurrent layer
b_y: bias of feedforward layer

Some of the types of RNNs are **fully Recurrent neural networks (FRNNs), Elman networks and Jordan networks, Long short-term memory networks (LSTMs), Gated Recurrent Units (GRUs) and Hopfield networks**. I will not get into further details about them, but one can find information regarding them in the *bibliography*.

3.3.4 Convolutional Neural Networks (CNNs):

Convolutional Neural Networks (CNNs) [1][2][4][7]: are distinguished from other neural networks by their superior performance with image, speech, or audio signal inputs. A convolutional neural network is a multilayered perceptron designed to recognize two (2) dimensional shapes with high degree of invariance and various forms of distortion (like translation, scaling skewing etc.).

Each CNN has a structure that abides by the following constraints (you can find more information in the corresponding bibliography):

1. **Feature Extraction:** each neuron of the CNN takes its input from its designated receptive field in the previous layer thus extracting the 'local' features
2. **Feature Mapping:** each computational layer of the network is composed of multiple feature maps in which the neurons are constrained to share the same weights. This benefits **shift invariance** and the **reduction of free parameters**.

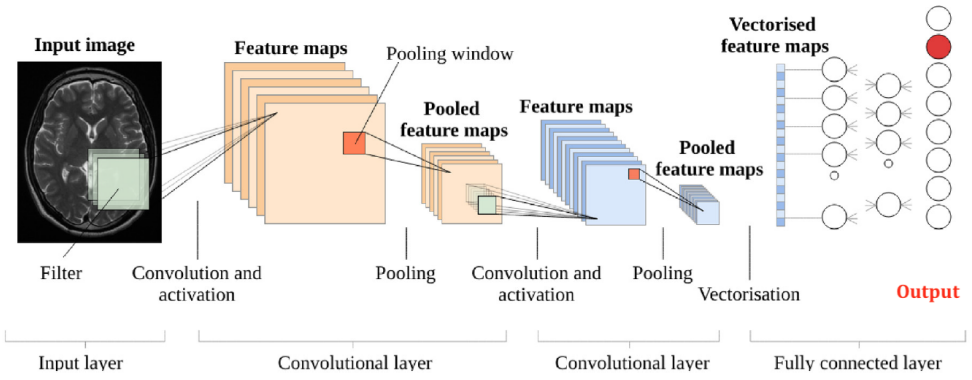

FIGURE 3.20: Example of a CNN's structure. Src: mriquestions.com

3. **Subsampling:** each computational layer performs **local averaging** and **subsampling** which lead to the decrement of resolution of the feature map. This, then, leads to the decrement of sensitivity of the feature map to various forms of distortion.

They have **three (3) main types of layers [14]**, which are:

- **Convolutional layer**: the main building block of a CNN. It contains a set of **filters (or kernels)**, the parameters of which are to be learned throughout the training and the size of which is usually smaller than the actual image (or other type of input). Each filter 'convolves' with the image and creates an **activation map**. For convolution the filter slides across the height and width of the image and the dot product between every element of the filter and the input is calculated at every spatial position.
- **Pooling layer**: usually incorporated between two (2) successive convolutional layers, the pooling layer reduces the number of parameters and computation by down-sampling the representation. The pooling function can be max or mean/average. **Max-pooling** is commonly used as it is proven that it works better.
- **Fully-connected (FC) layer**: the neurons in this layer apply a linear transformation to the input vector through a weight matrix. A non-linear transformation is then applied to the product through a non-linear activation function.

With the basics covered, we can proceed with the **'framework'** of the case study.

Chapter 4

Basic Principles and Implementation Framework

In this chapter, a walkthrough of the case study's experimentational process is given.

4.1 Topic, Data Source & Goal

The topic of this paper's case study belongs in the informetrics (bibliometrics & scientometrics) category **(see Background section)**. More specifically, I developed ML and DL (ANN) models that predict the **number of citations** an author will get in other people's work, for his contribution in scientific papers, magazines, articles etc..

Goal of the experiments is to prove that ANNs are capable of handling tasks such as the one stated above, they are the better choice amongst ML models, as well as show their potential for further improvement.

The data used in this thesis came from **APS**[1] **(American Physical Society)** and more specifically the **APS Data Sets for Research**[2] **of the Physical Review Journals**[3] (the data were provided to me after submitting a request along with the supervisor professor Dr. D. Katsaros from the official web-page of the APS). The data was in the form of a list with citing and cited DOIs (meaning pairs of articles that did the citation and the articles that they cited).

[1] aps.org
[2] journals.aps.org/datasets
[3] *Published by the American Physical Society.*

	citing_doi	cited_doi
0	10.1103/PhysRevSeriesI.11.215	10.1103/PhysRevSeriesI.1.1
1	10.1103/PhysRevSeriesI.12.121	10.1103/PhysRevSeriesI.1.166
2	10.1103/PhysRevSeriesI.7.93	10.1103/PhysRevSeriesI.1.166
3	10.1103/PhysRevSeriesI.16.267	10.1103/PhysRevSeriesI.2.35
4	10.1103/PhysRevSeriesI.17.65	10.1103/PhysRevSeriesI.2.112

	citing_doi	cited_doi
0	PhysRevSeriesI.11.215	PhysRevSeriesI.1.1
1	PhysRevSeriesI.12.121	PhysRevSeriesI.1.166
2	PhysRevSeriesI.7.93	PhysRevSeriesI.1.166
3	PhysRevSeriesI.16.267	PhysRevSeriesI.2.35
4	PhysRevSeriesI.17.65	PhysRevSeriesI.2.112

FIGURE 4.1: Initial data form.

4.2 Data Import, EDA & Data Preparation

The initial step of the process is importing the data, checking the relations and statistics of the features and outputs and visualizing them in order to get a better clearer picture of the dependencies between the different parts of the components of the analysis.

So first, I imported the data using Pandas and converted it into a more easy-to-be-processed form.

I, then, printed basic information about the dataset like the shape (8850333 rows, 2 columns), data types (object) and memory usage, checked for NULL (missing) and duplicated values (none), counted the number of unique citing_doi and cited_doi and found the most frequent (and their frequency) citing and cited_doi (606, 12012).

Still, since both columns are in a string form further processing was needed in order for them to be used. In order to do that I counted the number of citations each DOI has received and printed further statistics and some graphs to better get a notion of the datasets values. As one can see, the values in the dataset vary a lot (there are a lot of outliers).

But the feature (input) of the dataset, meaning the first column ('doi), still has string format. To deal with that, **vectorization**[4] (using sklearn) was used with a vectorizer of **max_features** set to 1000. This produced a sparse matrix with small values between one (1) and zero (0), meaning no scaling was necessary.

[4] *Vectorization is the process of transforming a scalar operation acting on individual data elements (Single Instruction Single Data—SISD) to an operation where a single instruction operates concurrently on multiple data elements (SIMD).* - ScienceDirect

	doi	num_citations_received
0	PhysRevLett.77.3865	12012
1	PhysRevB.54.11169	8331
2	PhysRev.140.A1133	8048
3	PhysRev.136.B864	6482
4	PhysRevB.50.17953	6155
...	...	...
586579	PhysRevB.84.075214	1
586580	PhysRevB.72.184105	1
586581	PhysRevE.80.026401	1
586582	RevModPhys.25.34	1
586583	PhysRevLett.63.1023	1

586584 rows × 2 columns

FIGURE 4.2: DOIs and number of citations received.

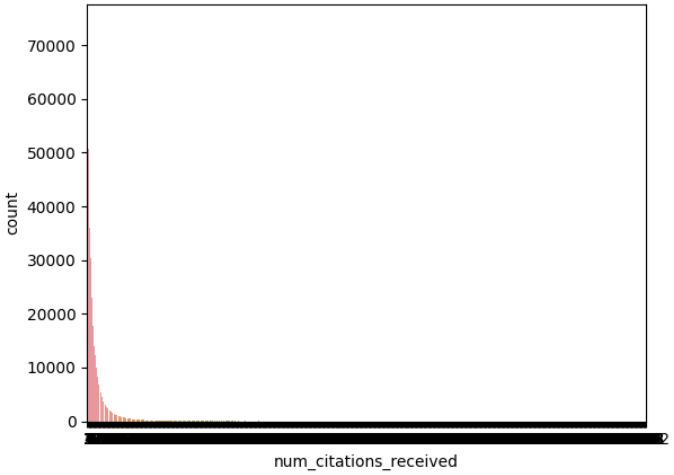

FIGURE 4.3: Countplot.

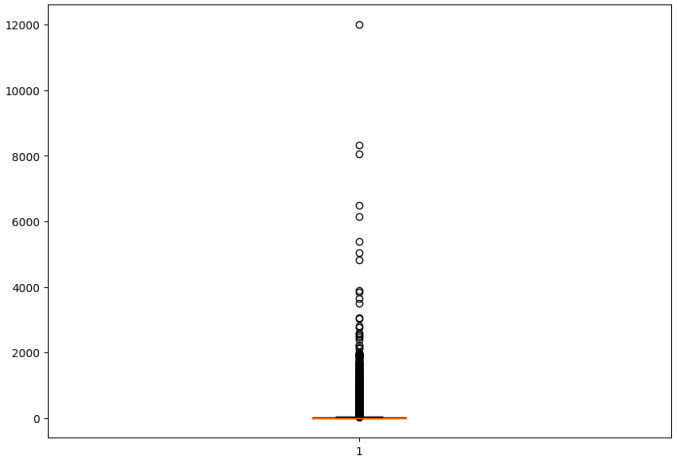

FIGURE 4.4: Boxplot.

Finally, for the output variable (num_citations_received) scaling was required since the the values of this column have large differences. For this purpose **StandardScaler**[5] **(sklearn)** was used. This was done in order to achieve unit variance.

4.3 Building ML

For the testing of all models the same ratio of training and testing percentage of the dataset was used **(80%-20% - sklearn's train_test_split was used)**.

The first ML model created used linear regression (with default parameters). Not only is linear regression a good match for this specific problem with this specific data to work with but it was, also, used for setting the baseline for the score (test error) for the models that follow. Next, I built a (single) decision tree model (DT regressor, with default parameters) and a random forest ensemble model (random forest regressor, with default parameters). Finally, three (3) k-nearest neighbors models to evaluate them with different distance functions (minkowski, euclidean, manhattan), each with the value of k in the range of [1, 40] (for these specific models I had to test each model by breaking the sets of values of k in [1, 10], [11, 20], [21, 30] and [31, 40] due to RAM problems).

Note that for all models MSE was used for evaluation. Logistic regression (classification oriented), (gaussian) naive bayes (classification oriented) and SVM (SVR, too much time) were not used.

4.4 Building DL

After building several ML models in order to set the baselines for the performance, next came designing the DL one, the **neural network**, and testing several values of hyperparameters to find the optimal one.

First, I built a NN with only one (1) hidden layer (the number of neurons for the input, hidden and output layer were 64-32-1), with ReLU as the activation function (the most basic in linear regression type of problems), adam as the optimizer, MSE as the assessment error function and set the number of epochs and batch size at 10 and 32 accordingly. After that, I tested each of these hyperparameters to find the optimal combination for the NN in the order that follows:

- **Batch size and number of epochs:** bs:32, 64, 128, 256 - epochs:10, 50, 100

[5] MinMaxScaler could also be used but after testing and comparing the models' differences are relatively the same (meaning exampleModel1 still performed better than exampleModel2 etc..

- **Training optimization algorithm:** optim_algo: SGD, RMSprop, Adagrad, Adadelta, Adam, Adamax, Nadam
- **Learning rate and momentum:** learning_rate: 0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.2, 0.3, 0.5, 0.9, 0.99, 1 - momentum: Adamax (the previous winner) does not have a momentum parameter
- **Weight initialization:** init_mode: uniform, lecun_uniform, normal, zero, glorot_normal, glorot_uniform, he_normal, he_uniform, constant, one, orthogonal, truncated_normal, ones
- **Activation function:** activation: softmax, softplus, softsign, relu, tanh, sigmoid, hard_sigmoid, linear
- **Dropout regularization:** weight_constraint:1.0, 2.0, 3.0, 4.0, 5.0 - dropout_rate: 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9
- **Number of neurons (and layers[6]):** [64, 32, 1], [64, 32, 16, 1], [64, 128, 32, 1], [64, 128, 64, 32, 1], [64, 128, 64, 32, 16, 1], [128, 64, 32, 16, 1], [128, 64, 1], [16 64 32 1], [16 64 32 16 1], [16 64 16 1], [16 32 16 1], [16 1], [16 32 1], [16 64 1], [256 128 1], [256 128 64 1], [256 182 64 32 1], [256 128 64 32 16 1]

Should be pointed out that, while the testing of the above could be done with **Grid Search**, a technique that tests every combination of the above in a single run, it was impossible to do so without running out of RAM and effectively crashing each run. So I decided to split the parameters to mini tests, each time finding the optimal value for one and testing the next one while keeping the first. While this technique has some obvious flaws, the results still are not compromised enough to prohibit drawing conclusions on the overall performance of the NN. Also, before I move on to the experiment section, a quick listing of the hardware and software used for the case study is given in the next part.

[6] In the sets of numbers that follow the first and last correspond to the input and output layers accordingly.

Chapter 5

Implementation and Core Components

This chapter includes an overview of the technical aspects of the experimentation.

5.1 Hardware

All experiments and trials conducted for this paper were done using a Lenovo Thinkpad™ notebook[1] with the following specs:

- **CPU**: AMD Ryzen™ 7 5825U with integrated Radeon™ Graphics (GPU),
- **RAM**: 16.0 GB (14.8 GB usable - 2x8GB DDR4 3200),
- **GPU**: no dedicated GPU (integrated),
- **OS**: Windows 11 Pro 64 (pre-installed) and
- **Hard Drive**: 1 TB SSD PCIe.

5.2 Software

Python 3.11.5 was used for writing the scripts for the experiment section. Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. It is dynamically typed and garbage-collected.

[1] I am not associated or affiliated in any way with Lenovo or any other company that manufactures any of the individual components (be it hardware or software) of the aforementioned notebook.

FIGURE 5.1: Visualized summary of hardware and software components used for this paper.

It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It, also, is the leading language in the field of building and testing AI (ML and DL) associated tasks and projects since it has really well developed dedicated languages for this specific purpose, making it incredibly easy to use and learn from.

Development of the project was done using **Jupyter Notebook (JN - formerly IPython Notebook)**. It is a web-based interactive computational environment for creating notebook documents. Jupyter Notebook is built using several open-source libraries, including IPython, ZeroMQ, Tornado, jQuery, Bootstrap, and MathJax. A Jupyter Notebook application is a browser-based REPL containing an ordered list of input/output cells which can contain code, text (using Github Flavored Markdown), mathematics, plots and rich media. Jupyter Notebook is part of **Project Jupyter**[2].

Of course, all the work took place in a **Microsoft's Windows 11 (ver. Pro 64)** environment (OS) but since, I think, Windows is recognized worldwide I will not extend further into the details.

5.3 Python Implementation

For the implementation of the theory explained and testing on a real world case, I used **Anaconda Navigator's** UI (User Interface) platform with JN where I coded the aforementioned

[2] *Project Jupyter is a non-profit, open-source project, born out of the IPython Project in 2014 as it evolved to support interactive data science and scientific computing across all programming languages. Jupyter will always be 100% open-source software, free for all to use and released under the liberal terms of the modified BSD license. - jupyter.org*

machine learning techniques using Python, as it provides simplicity along with a huge variety of libraries and frameworks related to ML and DL. For the standard data manipulation, arithmetic operations and visuals I used:

- **Pandas (pd)** with which I managed to handle the data. It is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.
- **NumPy (np) and SciPy (sp)** which provide assistance with arithmetic analysis and array manipulation and calculations. NumPy brings the computational power of languages like C and Fortran to Python, a language much easier to learn and use. SciPy provides algorithms for optimization, integration, interpolation, eigenvalue problems, algebraic equations, differential equations, statistics and other classes of problems. It extends NumPy by providing additional tools for array computing and provides specialized data structures, such as sparse matrices and k-dimensional trees.
- **Matplotlib (plt) and Seaborn (sns)**. The first one is the standard Python library for visualizations. It is a comprehensive library for creating static, animated, and interactive visualizations in Python. The second one is a data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

For the ML (LR, DT, KNN etc.) aspect of the project I used the most well-known and developed library possible, **Scikit-learn** or **Sklearn (sk)**. Sklearn provides simple and efficient tools for predictive data analysis. It is accessible to everybody and reusable in various contexts and it is built on NumPy, SciPy, and matplotlib. It, too, is open source and commercially usable (BSD license[3]).

It should be noted that since the case study of this project includes some part of **informetrics, bibliometrics** and **scientometrics**, one might think that some sort of **natural language processing (NLP)** took place. While that may be true in a very small degree (only in the data processing stage), no significant NLP took place, so libraries such as **nltk** (one of the most powerfull NLP python libraries), though considered, were not used.

For the DL part of the project **Keras** and **Tensorflow** was used:

- **Keras** is an API designed for human-friendly and easily operated interface in order to build neural networks capable of running on top of other frameworks such as Tensorflow.

[3] *BSD licenses are a family of permissive free software licenses, imposing minimal restrictions on the use and distribution of covered software. This is in contrast to copyleft licenses, which have share-alike requirements. The original BSD license was used for its namesake, the Berkeley Software Distribution (BSD), a Unix-like operating system. The original version has since been revised, and its descendants are referred to as modified BSD licenses.* - wikipedia

Keras offers consistent & simple APIs, it minimizes the number of user actions required for common use cases and it provides clear & actionable error messages.

- **TensorFlow** is an end-to-end open source platform for machine (and deep) learning. TensorFlow is a rich system for managing all aspects of a machine learning system. It is an open source project by the Google Brain Team with flexible architecture appropriate for platforms such as CPUs and GPUs (although for this thesis all experiments were carried out with the CPU-only version of Tensorflow), which replaced **Theano**[4]. Keras is the high-level API of the TensorFlow platform.

Different options could be used for the DL aspect of the project such as **Pytorch (torch), d2l and Theano**. That being said Tensorflow was chosen due to its recognition and ease of use.

Other libraries were used (or not) only when needed. Such are **math, statsmodel, mpl_toolkits, graphviz, sys, os, time, warnings etc.**.

Note: for the development of similar projects other tools, languages, Python frameworks and libraries could be used to do similar work. A few examples are **SAS, IBM's SPSS, Microsoft's CNTK, Julia, R** or most of any of the lower-level programming languages such as **Java or C++** (but with higher difficulty).

[4]Theano is a Python library that allows you to define, optimize, and efficiently evaluate mathematical expressions involving multi-dimensional arrays. It is built on top of NumPy.

Chapter 6

Experimentation & Validation

Now that every theoretical and technical aspect of the case study has been covered, we are ready to go through the experimentation process.

The **linear regression (LR)** starts this phase with default parameters and sets the baseline for the performance **(MSE)** of the experiment at **1.56501** (1.5650129677126559). It is a very basic model that leaves no significant room for improvement. Logistic regression (and Naive Bayes were not used for reasons mentioned in the **Fundamentals** and **Framework chapters**).

The **decision tree (DT) regressor** and the **random forest (RF) regressor (ensemble type of model)** resulted in **1.75973** (1.7597319945669845) and **1.60112** (1.6011283054357313) accordingly. Default parameters were used since I didn't have any specific indicators that set guidelines for trying specific values for them. Both models predicted worse than the LR model.

At this point I tried testing a **support vector regressor (SVM - SVR)** model with different values for its kernel and degree parameters. But for reasons explained in the **Fundamentals** and **Framework chapters** I did not go through with it.

Next came the **k-nearest neighbors regressor (KNN)** model, using three (3) different distance functions and testing several different values for k (namely in the range of [1,40]):

- Using the **Euclidean distance function:** the worst and best pairs of k and performance accordingly were (1, 1.87740) and (40, 1.57625) (declining trend with exceptions while k increases)
- Using the **Manhattan distance function:** the worst and best pairs of k and performance accordingly were (1, 1.89110) and (40, 1.57019) (declining trend with exceptions while k increases)

- Using the **Minkowski distance function:** the worst and best pairs of k and performance accordingly were (1, 1.98271) and (40, 1.58293) (declining trend with exceptions while k increases)

Still the results indicate that the simple LR model dominates the ML part of the experiment. That being said it should be noted that while LR, DT and RF were all relatively fast (LR and DT taking approximately **under 1 hour** to run and RF approximately **1.5 hours**), the KNN models required approximately **18, 8.5 and 18.5 hours** accordingly.

These all show the superiority (in the context of this thesis' experiment) of the LR model.

Last came the DL model or ANN. As mentioned previously, for assessing the true potential of the NN model a type of hyperparameter optimization was necessary. So starting with ReLU as the activation function and Adam as the optimizer (each test resulted in keeping tested parameter(s) and moving to the next test with this (these) value(s)):

- the best combination of batch size and number of epochs was: 128 and 10 with MSE of **1.5704** (1.570473551750183 - worse than LR)
- the best training optimization algorithm was Adamax with MSE of **1.5585** (1.5585778951644897 - better than LR)
- the best combination of learning rate and momentum was: 0.001 and no momentum since Adamax was picked as the training optimization algorithm with MSE of **1.5584** (1.5584787130355835 - improvement)
- the best network weight initialization mode was: he_normal with MSE of **1.5581** (1.5581960678100586 - improvement)
- the best activation function was: ReLU with MSE of **1.5589** (1.558955430984497 - no improvement)
- the best combination for dropout regularization was: model__dropout_rate=0.1 and model__weight_constraint=MaxNorm(5) with MSE of **1.5558** (1.5558226108551025 - improvement)
- the best combination of number of neurons in hidden layer were: (64,32), (256,128) and (64,128,32) with MSEs of **1.5597**, **1.5586**, **1.5578**. For this particular hyperparameter no single best combination was chosen due to the fact that since the number of hidden layers can be any positive number these three (3) results were chosen as the best.

The above results indicate that the NN model outperforms all the ML models.

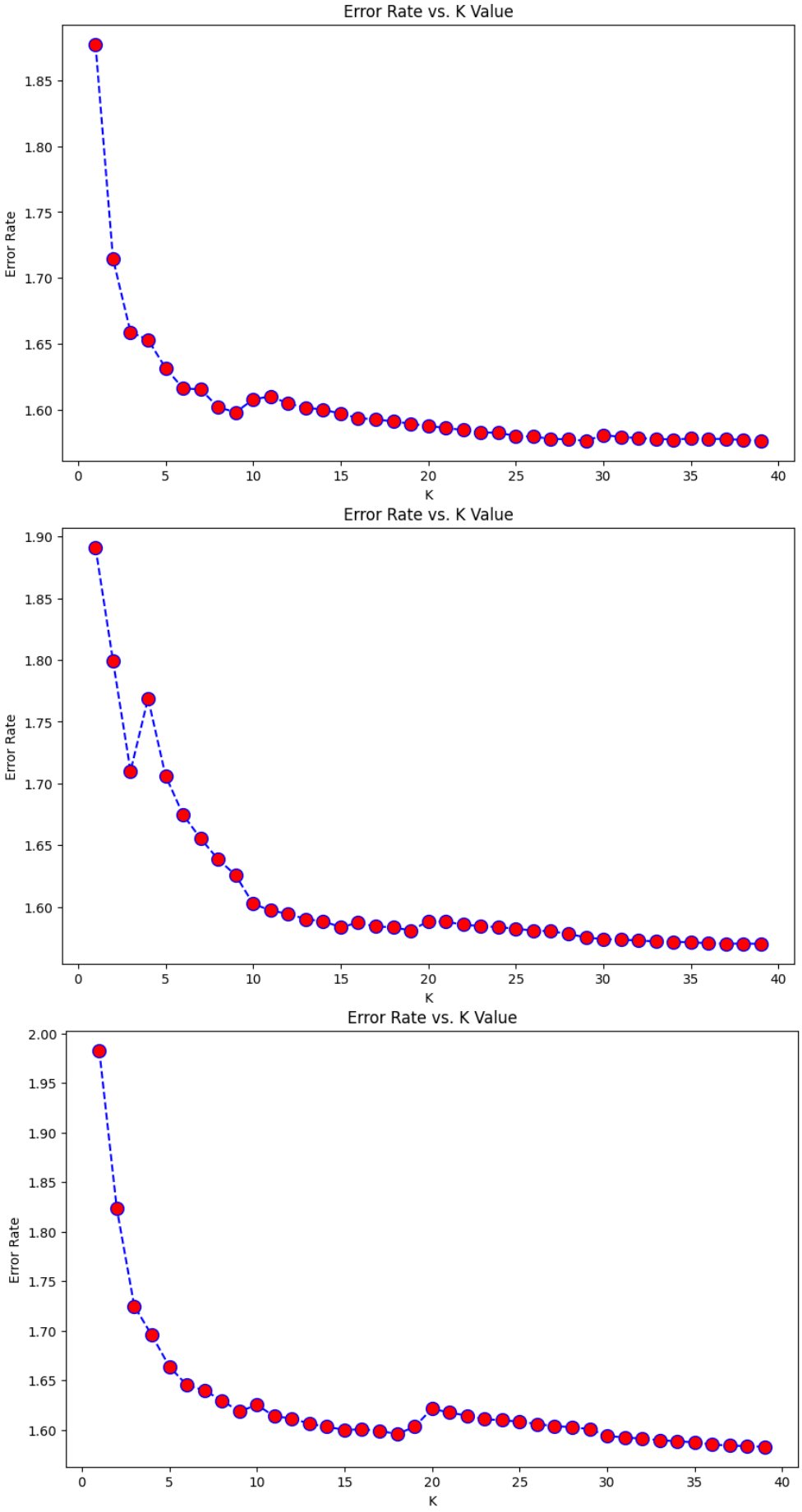

FIGURE 6.1: Results for MSE relative to k for KNN using the Euclidean, Manhattan and Minkowski distance function (in this order).

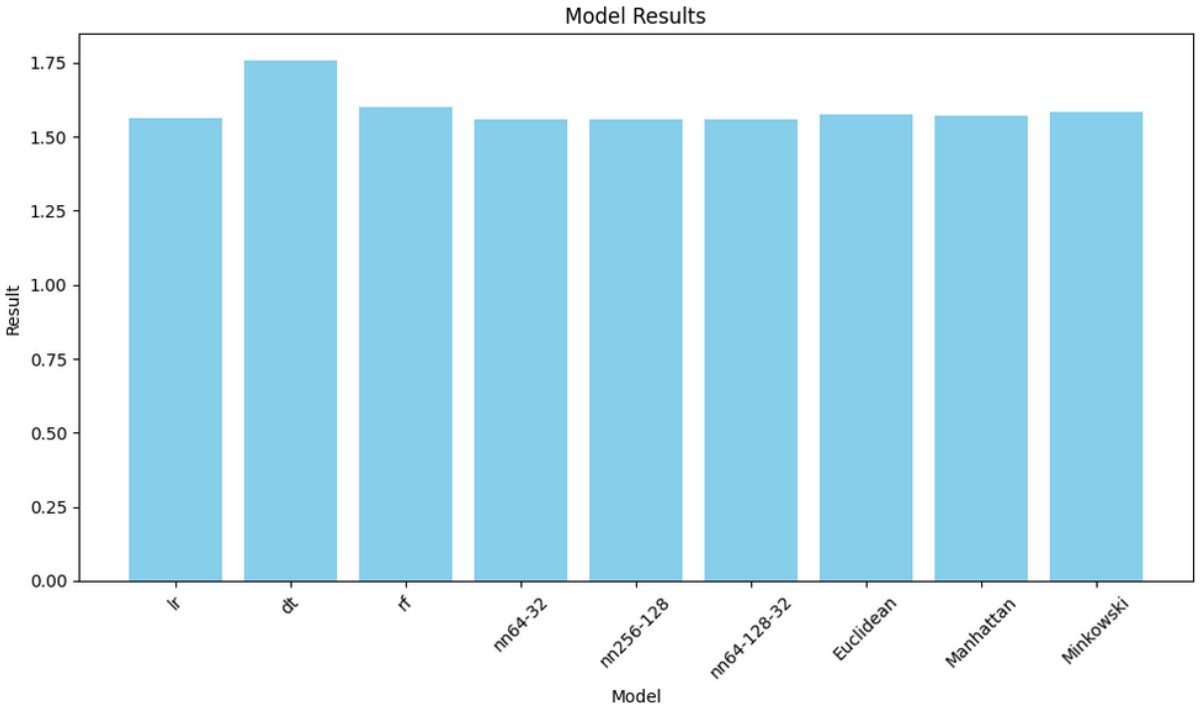

FIGURE 6.2: Concentrated results of all the ML (k=40 for every distance function in KNN ones) and DL models. All the NN models outperform the ML models with the best being nn64-128-32. The ML models are dominated by the LR model.

Chapter 7

Conclusions & Future Work

The aims of this project were to prove whether or not ML and DL can be used for informetrics related problems such as predicting the number of an author's citations based on her/his publications. This was proven in the early stage of the experimentation section. Furthermore, they were to compare the different ML and DL models, test how well they performed and finally prove whether DL methods can outperform the most commonly used ML methods. This took place at the second half of the experimentation section, in which there was not only proof that ML can clearly handle informetrics problem sufficiently well, based on the results, but it was also proven that DL models can easily outperform basic ML models both with their accuracy as well as their speed.

One can see that as early as only the second parameter's tuning, the NN model performed better and only continued to improve. That in combination with the fact that on average running the NN model required only a handful of minutes (in contrast with all the ML models) lead to the conclusion that DL and consequently NNs are superior both in terms of performance as well as speed to conventional ML and show potential for further improvement.

Through the writing of this paper I reached a better understanding of and had a deeper experience with working with ML and DL both theoretically and practically. Not only that but I, also, was introduced to the world of informetrics. Personal key takeaways from this project are the fact that ML and DL (especially the latter) has great potential for problems of this thematic as well as problems of other kinds of topic.

ANNs are a fascinating subject and are the key to improving and/or solving many known (explored) and unknown problems. Since the start of this thesis I have been more and more intrigued by them. One idea in particular is the use of convolutional NNs and more advanced work with natural language processing. In the future I would like to further explore these and

the use of ANNs in general and deepen further into their parameters and mechanics as well as problems regarding informetrics and their subcategories like cyber- and webometrics.

Bibliography

1. https://www.ibm.com/cloud/learn/data-science-introduction
2. D. Katsaros, E. Fragkou, D. Papakostas, *Backpropagation for Convolutional Neural Networks*
3. D. Katsaros, E. Fragkou, V. Lygnos (2023), *Transfer Learning for Convolutional Neural Networks in Tiny Deep Learning Environments*, https://dl.acm.org/doi/abs/10.1145/3575879.3575984, DOI: doi=10.1145/3575879.3575984
4. Nikolaos Nanas, Manolis Vavalis, Elias Houstis (2010), *Personalised news and scientific literature aggregation*, https://www.sciencedirect.com/science/article/pii/S0306457309000855, DOI: doi=https://doi.org/10.1016/j.ipm.2009.07.005
5. https://365datascience.com/tutorials/statistics-tutorials/
6. Simon Haykin (2009), *Neural Networks and Learning Machines, Third Edition*, ISBN-13: 9780133002553
7. https://towardsdatascience.com/
8. https://developers.google.com/machine-learning/
9. Ian Goodfellow, Yoshua Bengio, Aaron Courville (2016), *Deep Learning*, https://www.deeplearningbook.org/, MIT Press
10. https://www.investopedia.com
11. https://builtin.com/data-science/step-step-explanation-principal-component-analysis
12. https://en.wikipedia.org/wiki/
13. Martin T. Hagan, Howard B. Demuth, Mark Hudson Beale, Orlando De Jesus, *Neural Network Design, 2nd Edition*, https://hagan.okstate.edu/NNDesign.pdf, ISBN: 0971732116, 9780971732117
14. https://www.statisticshowto.com/

15. Andriy Burkov (2019), *The Hundred Page Machine Learning Book*, ISBN: 1999579518, 9781999579517

16. https://www.sciencedirect.com

17. Jason Brownlee (2021), *Optimization for Machine Learning: Finding Function Optima with Python*, Edition: v1.00, https://books.google.gr/books?id=tW1HEAAAQBAJ, Machine-LearningMastery.com

18. https://www.geeksforgeeks.org

19. https://iq.opengenus.org

20. FLETCHER, Tristan. Support vector machines explained. Tutorial paper, 2009, 1-19., www.cs.ucl.ac.uk/staff/T.Fletcher/

21. https://programmathically.com

22. https://news.mit.edu

23. www.analyticsvidhya.com/

24. https://machinelearningmastery.com

25. David Beazley, Brian K. Jones (2013), *Python Cookbook, 3rd Edition*, https://www.oreilly.com/library/view/python-cookbook-3rd/9781449357337/, O'Reilly Media, Inc., ISBN: 9781449357351

26. APS Dataset for Research: https://journals.aps.org/datasets

27. Daniel E. Acuna, Stefano Allesina and Konrad P. Kording (2012), *Predicting Scientific Success*, https://www.nature.com/articles/489201a, DOI: https://doi.org/10.1038/489201a

28. Yuhao Zhou, Ruijie Wang, An Zeng (2022), *Predicting the impact and publication date of individual scientists' future papers*, DOI: https://doi.org/10.1007/s11192-022-04286-w

29. Abrishami, A., Aliakbary, S. (2019). *Predicting citation counts based on deep neural network learning techniques.*, Journal of Informetrics 13(2), 485–499. https://www.sciencedirect.com/science/article/pii/S1751157718303821, DOI: https://doi.org/10.1016/j.joi.2019.02.011

30. Acuna, D. E., Allesina, S., Kording, K. P. (2012). *Predicting scientifc success.* Nature 489(7415), 201–202.

31. García-Pérez, M. A. (2013). *Limited validity of equations to predict the future h index.* Scientometrics 96(3), 901–909.

32. Mistele, T., Price, T., Hossenfelder, S. (2019). *Predicting authors' citation counts and h-indices with a neural network.* Scientometrics 120(1), 87–104. https://link.springer.com/article/10.1007/s11192-019-03110-2, DOI: https://doi.org/10.1007/s11192-019-03110-2

33. Nezhadbiglari, M., Gonçalves, M. A., Almeida, J. M. (2016). *Early prediction of scholar popularity.* In Proceedings of the 16th ACM/IEEE-CS on Joint Conference on Digital Libraries, pp 181–190.

34. Ruan, X., Zhu, Y., Li, J., & Cheng, Y. (2020). *Predicting the citation counts of individual papers via a bp neural network.* Journal of Informetrics, 14(3), 101039. https://www.sciencedirect.com/science/article/pii/S1751157719303979, DOI: https://doi.org/10.1016/j.joi.2020.101039

35. Yu, T., Yu, G., Li, P. Y., Wang, L. (2014). *Citation impact prediction for scientifc papers using stepwise regression analysis.* Scientometrics 101(2), 1233–1252.

36. Palmucci A, Liao H, Napoletano A, Zaccaria A (2020) *Where is your field going? A machine learning approach to study the relative motion of the domains of physics.* PLoS ONE 15 (6): e0233997. https://doi.org/10.1371/journal.pone.0233997

37. Murali Krishna Enduri, V Udaya Sankar, Koduru Hajarathaiah (2022), *Empirical Study on Citation Count Prediction of Research Articles*, DOI: 10.5530/jscires.11.2.17

38. William L. Croft, Jörg-Rüdiger Sack (2020), *Predicting the citation count and CiteScore of journals one year in advance*, https://www.sciencedirect.com/science/article/pii/S1751157722001018, DOI: https://doi.org/10.1016/j.joi.2022.101349

39. Cynthia Lokker, K Ann McKibbon, R James McKinlay, Nancy L Wilczynski, R Brian Haynes (2008), *Prediction of citation counts for clinical articles at two years using data available within three weeks of publication: retrospective cohort study*, https://www.bmj.com/content/336/7645/655.short, DOI: https://doi.org/10.1136/bmj.39482.526713.BE

40. Naoki Shibata, Yuya Kajikawa, Ichiro Sakata (2011), *Link prediction in citation networks*, https://onlinelibrary.wiley.com/doi/full/10.1002/asi.21664, DOI: https://doi.org/10.1002/asi.21664

41. Liyang Yao, Tian Wei, An Zeng, Ying Fan, Zengru Di (2014), *Ranking scientific publications: the effect of nonlinearity*, https://www.nature.com/articles/srep06663, DOI: https://doi.org/10.1038/srep06663

42. Osborne, Francesco; Peroni, Silvio and Motta, Enrico (2014). Clustering citation distributions for semantic categorization and citation prediction. In: I4th Workshop on Linked Science 2014— Making Sense Out of Data (LISC2014) , 19-23 October 2014, Riva Del Garda, Trentino, Italy (Forthcoming).

43. Xiao Yu, Quanquan Gu, Mianwei Zhou, Jiawei Han (2012), *Citation Prediction in Heterogeneous Bibliographic Networks*, https://citeseerx.ist.psu.edu/doc/10.1.1.220.2490, https://epubs.siam.org/doi/abs/10.1137/1.9781611972825.96, DOI: doi=10.1.1.220.2490

44. Laura Dietz, Steffen Bickel, Tobias Scheffer (2007), *Unsupervised Prediction of Citation Influences*, https://citeseerx.ist.psu.edu/doc/10.1.1.474.9148, https://dl.acm.org/doi/abs/10.1145/1273496.1273526, DOI: doi=10.1.1.474.9148

45. Qihang Zhao, Xiadong Feng (2022), *Utilizing citation network structure to predict paper citation counts: A Deep learning approach*, https://www.sciencedirect.com/science/article/pii/S1751157721001061, DOI: https://doi.org/10.1016/j.joi.2021.101235

46. Lutz Bornmann, Loet Leydesdorff, Jian Wang (2014), *How to improve the prediction based on citation impact percentiles for years shortly after the publication date?*, https://www.sciencedirect.com/science/article/pii/S1751157713001065, DOI: https://doi.org/10.1016/j.joi.2013.11.005

47. Xiaomei Bai, Fuli Zhang, Ivan Lee (2019), *Predicting the citations of scholarly paper*, https://www.sciencedirect.com/science/article/pii/S1751157718301767, DOI: https://doi.org/10.1016/j.joi.2019.01.010

48. Rui Yan, Jie Tang, Xiaobing Liu, Dongdong Shan, Xiaoming Li (2011), *Citation count prediction: learning to estimate future citations for literature*, https://dl.acm.org/doi/abs/10.1145/2063576.2063757, DOI: doi=10.1145/2063576.2063757

49. Yuxiao Dong, Reid A. Johnson, Nitesh V. Chawla (2015), *Will This Paper Increase Your h-index?: Scientific Impact Prediction*, https://dl.acm.org/doi/abs/10.1145/2684822.2685314, DOI: doi=10.1145/2684822.2685314

50. Shengzhi Huang, Yong Huang, Yi Bu, Wei Lu, Jiajia Qian, Dan Wang (2022), *Fine-grained citation count prediction via a transformer-based model with among-attention mechanism*, https://www.sciencedirect.com/science/article/pii/S0306457321002776, DOI: https://doi.org/10.1016/j.ipm.2021.102799

51. Akhil Pandey Akella, Hamed Alhoori, Pavan Ravikanth Kondamudi, Cole Freeman, Haiming Zhou (2021), *Early indicators of scientific impact: Predicting citations with altmetrics*, https://www.sciencedirect.com/science/article/pii/S1751157720306453, DOI: https://doi.org/10.1016/j.joi.2020.101128

52. Yu T., Yu G., Li PY. et al. Citation impact prediction for scientific papers using stepwise regression analysis. Scientometrics 101, 1233–1252 (2014). https://doi.org/10.1007/s11192-014-1279-6

53. Pobiedina, N., Ichise, R. Citation count prediction as a link prediction problem. Appl Intell 44, 252–268 (2016). https://doi.org/10.1007/s10489-015-0657-y

54. Klimek, P., S. Jovanovic, A., Egloff, R. et al. Successful fish go with the flow: citation impact prediction based on centrality measures for term–document networks. Scientometrics 107, 1265–1282 (2016). https://doi.org/10.1007/s11192-016-1926-1

55. Giovanni Abramo, Ciriaco Andrea D'Angelo, Giovanni Felici (2019), *Predicting publication long-term impact through a combination of early citations and journal impact factor*, https://www.sciencedirect.com/science/article/pii/S1751157718300208, DOI: https://doi.org/10.1016/j.joi.2018.11.003

56. Rajmund Klemiński, Przemyslaw Kazienko, Tomasz Kajdanowicz (2021), *Where should I publish? Heterogeneous, networks-based prediction of paper's citation success*, https://www.sciencedirect.com/science/article/pii/S1751157721000717, DOI: https://doi.org/10.1016/j.joi.2021.101200

57. Dalibor Fiala, Gabriel Tutoky (2017), *PageRank-based prediction of award-winning researchers and the impact of citations*, https://www.sciencedirect.com/science/article/pii/S175115771730038X, DOI: https://doi.org/10.1016/j.joi.2017.09.008

58. Eleni Fragkiadaki (2016), *Bibliographic database analysis: citation graphs and indirect indicators*, https://www.didaktorika.gr/eadd/handle/10442/38595, DOI: doi=10.12681/eadd/38595

59. Antonis Sidiropoulos (2006), *Ranking and clustering methods for bibliographic and web data*, https://www.didaktorika.gr/eadd/handle/10442/15367, DOI: doi=10.12681/eadd/15367

60. Eleftherios D. Mparmpagiannis (2016), *Complex network analysis for ranking Scientists*, https://ir.lib.uth.gr/xmlui/handle/11615/48323, URI: http://hdl.handle.net/11615/48323

61. Tryfon Tsakiris (2018), *Deep Learning Frameworks' Effectiveness and Efficiency for Time Series Prediction*, https://ir.lib.uth.gr/xmlui/handle/11615/49476, URI: http://hdl.handle.net/11615/49476

62. Maria Markou (2015), *Content-based scientometrics*, https://ir.lib.uth.gr/xmlui/handle/11615/46183, URI: http://hdl.handle.net/11615/46183, http://dx.doi.org/10.26253/heal.uth.1851

63. ArnetMiner: http://arnetminer.org/citation, http://arnetminer.org, http://arnetminer.org/AMinerNetwork

64. https://www.kaggle.com/datasets/Cornell-University/arxiv

65. ArXiv Open Access: https://arxiv.org/

66. CSSCI (Chinese Social Sciences Citation Index): https://lib.sustech.edu.cn/2021/0601/c530a3594/page.html

67. AcademicTree: https://academictree.org/

68. Scopus-Elsevier: https://www.elsevier.com/solutions/scopus

69. https://www.kaggle.com

70. https://github.com

Appendix A

I would like to mention a few types of analysis with a close relation and high degree of similarity to PCA, the types of regression and the (Gaussian) Naive Bayes classifier. Understanding of these types of analysis and classifiers may assist in a better understanding of the methods and algorithms used in this paper and further studying the subjects of DS, ML and DL.

Analysis of Variance (ANOVA) [4][11][13]: a collection of statistical models used to analyze the differences among means. It provides a statistical test of whether two (2) (or more) means are equal (it is a generalization of the t-test[1][2]). ANOVA uses categorical independent variables and a continuous dependent variable.

Linear Discriminant Analysis (LDA) or Normal Discriminant Analysis (NDA) or discriminant function analysis [4][11][13]: a method used to a linear combination of features that characterizes two (2) or more classes of objects. LDA uses continuous independent variables and a categorical dependent variable. LDA assumes the covariance of all classes is identical. Logistic regression has more similarities with LDA than ANOVA.

Quadratic Discriminant Analysis (QDA) [4][11][13]: a statistical classifier that uses quadratic decision surface to separate measurements of two or more classes of objects or events. Unlike LDA, QDA does not assume equal variance for all classes.

Independent Component Analysis (ICA) [4][11][13][17]: an ML technique that separates independent sources from mixed signals. Unlike PCA, it focuses on the independence of the components.

[1] **(Student's) t-test** is a statistical hypothesis test that where the statistic under question follows the student's t-distribution under the null hypothesis.

[2] **Null hypothesis or H_o (inferential statistics)**: the hypothesis that two possibilities are the same and that the observed difference is caused purely by chance.

Appendix B

- Difference between models and algorithms in machine learning

ML algorithms are procedures that are implemented in code and run on data. **ML models** are output by algorithms and are comprised of model data and a prediction or a classification algorithm. **-machinelearningmastery.com**

- Difference between model-based and instance-based algorithms in machine learning

Model-based algorithms use the training data to create a model whose parameters are computed using the training data. After the training procedure is complete, the (training) data can be discarded. E.g. Regression. **Instance-based algorithms** use the entirety of the data to build a model. E.g. k-Nearest Neighbors, Decision Trees.

Note: while SVM is considered by many a model-based algorithm, it belongs to the general family of kernel machines which are widely known to be instance-based algorithms.