2023-12-01

Integrating Machine Learning Methods For Medical Diagnosis

Jazmin Quezada
University of Texas at El Paso

INTEGRATING MACHINE LEARNING METHODS FOR MEDICAL DIAGNOSIS

JAZMIN QUEZADA

Doctoral Program in Data Science

APPROVED:

Maria Christina Mariani, Chair, Ph.D.

Osei Tweneboah, Ph.D., Co-Chair

Joe A. Guthrie, Ph.D.

Natasha S Sharma, Ph.D.

Kristine M. Garza, Ph.D.

Stephen Crites, Ph.D.
Dean of the Graduate School

To my

Father & brother Michael, may your souls rest in peace,

and to my mother; with the greatest of love.

INTEGRATING MACHINE LEARNING METHODS FOR MEDICAL DIAGNOSIS

by

JAZMIN QUEZADA

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

Department of Mathematical Sciences

THE UNIVERSITY OF TEXAS AT EL PASO

December 2023

Table of Contents

Chapter 1

Introduction

As the prevalence of data-driven healthcare continues to grow, the role of machine learning in medical diagnosis has become increasingly pivotal. This study introduces an innovative approach dedicated to enhancing and fine-tuning predictive models for medical diagnoses through the application of various optimization techniques. The aim is to advance the accuracy and efficiency of diagnostic processes, contributing to the ongoing evolution of data-driven healthcare practices. Machine learning enables us to employ algorithms designed to mimic human learning patterns and enhance their precision. This method primarily concentrates on refining and optimizing parameters of widely used machine learning algorithms in medical diagnosis, encompassing logistic regression, support vector machines, and neural networks. Function diagnosis $= f(p_1, p_2, ..., p_{n_f})$ can be approximated by using many approximation techniques (neural networks, various nonlinear regression models, etc.). To find optimal result of the algorithm f it is possible to extrapolate values of the function f and find possible new optimal value. Through the strategic application of optimization techniques, we meticulously navigate the parameter space of these algorithms to uncover the most optimal setups. Furthermore, by representing algorithms as computational graphs and leveraging their relationships with diagnostic outcomes, we can project the ideal characteristics of existing algorithms. This has the potential to steer the development of new, exceedingly accurate diagnostic tools. The integration of machine learning and optimization models provides a systematic, data-driven framework for enhancing existing algorithms and uncovering inventive solutions, ultimately resulting in improved medical outcomes. This approach's effectiveness is showcased in a case studies involving logistic regression, where parameters like the regularization strength inverse (C) are carefully fine-tuned. The results

from the logistic regression were refined and expanded using graph neural networks, ultimately enhancing medical diagnosis. Furthermore, we applied this method to various other ML models like K-nearest neighbors (KNN), graph neural networks (GNN), Support Vector Machine (SVM), and others, all with the primary aim of advancing medical diagnosis.

Accurate medical diagnosis is the cornerstone of effective healthcare, yet diagnostic errors remain a widespread concern. In the dynamic realm of healthcare, the incorporation of state-of-the-art technologies plays a pivotal role in enhancing diagnostic accuracy. Notably, machine learning emerges as a powerful instrument with the capability to transform medical diagnoses through the utilization of data-driven algorithms. The convergence of artificial intelligence and healthcare holds significant promise, particularly in addressing the enduring issue of diagnostic precision. Every year, millions of patients worldwide receive incorrect diagnoses, resulting in significant consequences for patient well- being and healthcare expenses. These errors can stem from various factors, such as the complexity of medical conditions, variability in clinical presentations, and the limitations of conventional diagnostic techniques.[10] The National Academy of Medicine acknowledges diagnostic errors, characterized as "the failure to establish an accurate and timely explanation of a patient's health problem(s) or to communicate that explanation to the patient," these errors encompass delayed, incorrect, or overlooked diagnoses. Despite the personal and societal implications of diagnostic errors, they are frequently overlooked and not adequately addressed in patient safety initiatives and healthcare quality metrics and guidelines. To highlight the seriousness of this issue, consider situations where critical conditions, particularly in their early stages, may go unnoticed even by highly experienced healthcare professionals. These scenarios emphasize the immediate requirement for more robust, dependable, and unbiased diagnostic tools. This research delves into the integration of machine learning and optimization models, presenting a comprehensive approach to tackle this urgent challenge. Harnessing the synergies between these two domains, our objective is to elevate the precision of medical diagnoses, leading to a tangible enhancement in patient care and outcomes. Furthermore, we aim to enhance our diagnostic processes through the applica-

tion of meta-learning. This comprehensive strategy leverages the combined strengths of AI, computer science, and meta-learning to advance the frontiers of diagnostic precision, ushering in a new era of enhanced healthcare outcomes. This approach empowers software applications to assimilate insights from both data and algorithms, mimicking the learning mechanisms inherent in human cognition. Through this iterative learning process, software can continually refine its performance and accuracy, adapting to diverse inputs and changing circumstances. Meta-learning involves the ability to acquire the skill of learning itself, encompassing tasks such as adjusting hyper-parameters within established learning algorithms and effectively utilizing existing models and knowledge to address novel challenges. This meta-learning proficiency plays a crucial role in enhancing the adaptability and flexibility of current AI systems, enabling them to efficiently address previously unencountered tasks. Bridging the gap between human performance and the current capabilities of AI systems, the development of successful collaborations in this domain holds the promise of yielding innovative ideas and substantial progress. The ability of meta-learning to empower AI systems to generalize knowledge and swiftly adapt to new scenarios positions it as a cornerstone in advancing the frontier of artificial intelligence. Meta-learning algorithms have the capacity to leverage optimal predictions derived from machine learning algorithms, thereby enhancing their own predictive capabilities. The exploration and development of meta-learning concepts in computer science date back to the 1980s, gaining widespread recognition and popularity following the influential contributions of Jürgen Schmidhuber and Yoshua Bengio on the subject. The field of meta-learning, or learning-to-learn, has seen a dramatic rise in interest in recent years, along with methods as transfer learning. Contrary to conventional approaches to AI where tasks are solved from scratch using a fixed learning algorithm, meta-learning aims to improve the learning algorithm itself, given the experience of multiple learning episodes. We will expand on the topic on chapter three, where we go over our methods and materials.

The concept of the neural network can be attributed to the collaboration between logician Walter Pitts and neuroscientist Warren McCulloch in 1943, as they sought to elu-

cidate the decision-making mechanisms inherent in human cognition and neural networks [21]. Fast forward to 1957, where American psychologist Frank Rosenblatt introduced the perceptron, marking the inaugural neural network designed to emulate human thought processes. Noteworthy developments in machine learning during the 1950s were characterized by pioneering research focusing on elementary algorithms. An illustrative example is Arthur Samuel's IBM program, which, employing the alpha-beta pruning algorithm, demonstrated machine learning capabilities by playing checkers. These algorithms extrapolated from historical data to anticipate new outcomes [17]. With technological advancements in recent decades, ML has facilitated significant breakthroughs across diverse fields, encompassing image processing and autonomous vehicles. The introduction and subsequent reemphasis of backpropagation in the 1960s and 1980s respectively fine-tuned neural networks for commercial applications, uncovering hidden layers between input and output.[17] Machine learning employs mathematical and statistical techniques to train algorithms, resulting in a comprehensive grasp of data and enabling accurate classifications or predictions. This process unveils valuable insights and refines data representation. Within the expansive field of artificial intelligence (AI), machine learning, deep learning, and neural networks are intricately connected. Neural networks, in particular, constitute a subset of machine learning, and deep learning (DL) is a subset encapsulated within the broader realm of machine learning. It is noteworthy that deep learning goes beyond conventional machine learning by engaging various types of neural networks. It involves a precise process of optimizing hyper-parameters, features, and other network parameters to attain the most effective representation of the underlying database, as expounded by [13]. Artificial neural networks (ANNs) serve as the backbone of these methodologies, comprising layers of nodes that include an input layer, one or more hidden layers, and an output layer. The classification of a neural network as either a basic neural network or a deep learning algorithm depends on the number of layers it encompasses. Specifically, a neural network with more than three layers is categorized as a deep neural network, while one with three layers or fewer is deemed a basic neural network. This study integrates the theoretical foundations of neural

network methodology with the practical applications of machine learning, especially in the domain of deep learning, to scrutinize real-world data. The research seeks to introduce advancements in network optimization for machine learning models, such as the incorporation of a meta-learning model, with a specific emphasis on enhancing the precision of medical diagnoses. The proposed work aims to contribute to the ongoing refinement and advancement of accurate diagnostic processes within the medical domain.

For a concise overview of machine learning, it employs two main techniques: *Supervised learning* involves training a model on known input and output data to predict future outputs. *Unsupervised learning* focuses on discovering hidden patterns or intrinsic structures in input data and can be categorized further. Supervised learning methods encompass neural networks, linear regression, logistic regression, random forest, among others; as for unsupervised learning methods we have typically the Hierachical clustering, K-NN (k nearest neighbors) and others. Additionally, there is semi-supervised learning, striking a balance between supervised and unsupervised approaches. This method utilizes a smaller labeled dataset to guide classification and feature extraction from a larger, unlabeled dataset. In our machine learning research, we've employed diverse algorithms on historical and contemporary datasets, encompassing medical diagnoses and other domains within AI and ML. Among the prevalent ML algorithms are neural networks, mimicking the intricacies of the human brain through an extensive network of interconnected processing nodes. Renowned for their pattern recognition capabilities, neural networks find application in various domains, including speech recognition, natural language translation, and image recognition. We also looked into various machine learning algorithms employed in our dataset for classification purposes, one being *logistic regression.* This supervised learning technique predicts outcomes for categorical response variables, such as yes/no responses to questions. While applicable to various data types, logistic regression is most commonly used with cross-sectional data and has roots in the biological sciences dating back to the early twentieth century. The logistic regression model can be derived or viewed in three different ways:

Binary logistic regression: The response variable can belong to one of two categories.

Multinomial logistic regression: The response variable can belong to one, three, or more categories, with no inherent ordering among them. **Ordinal logistic regression:** The response variable can belong to one of three or more categories, and there exists a natural ordering among these categories.

In the realm of deep learning, the emergence of modern Convolutional Neural Networks (CNNs) in the 1990s was heavily influenced by the neocognitron. Yann LeCun et al., in their seminal paper "Gradient-Based Learning Applied to Document Recognition" (cited 17,588 times) [28], showcased the efficacy of a CNN model in progressively integrating simpler features into more complex ones, leading to successful handwritten character recognition, with similar success KNN.[28] We introduced the K-nearest-neighbor (KNN) model and applied our meta-learning approach to enhance KNN for medical diagnosis. Our previous research involved using a multi-layer perceptron, commonly referred to as a neural network, to identify instances of financial market crashes and predict credit card default payments for a financial institution's clients and implemented imputation techniques using ML and other range of algorithms. This exploration into neural networks sparked the idea of integrating graph neural networks into our optimization meta-learning model. Looking forward, our objective is to apply these methodologies to medical diagnosis datasets, focusing on classification objectives. We also aim to future propose numerical optimization techniques to further refine the precision of medical diagnoses.

1.1 Data Analysis

Data science and data analytics are critical components of modern businesses and organizations. They involve the systematic process of extracting valuable insights from raw data to aid in decision-making. Data science is a multidisciplinary field that encompasses various techniques, algorithms, and tools to uncover patterns, trends, and correlations within data. It incorporates elements of statistics, mathematics, programming, and domain expertise to extract meaningful insights. Data scientists are responsible for developing models,

algorithms, and conducting in-depth analyses to address complex business -task speicfic problems.

On the other hand, data analytics is a subset of data science that focuses on the process of cleaning, transforming, and processing raw data to derive actionable and pertinent information. This involves tasks such as data cleaning, data transformation, exploratory data analysis, and the application of statistical techniques to draw meaningful conclusions. The procedure aids in mitigating inherent risks in decision-making by offering valuable insights and statistics, often presented through charts, images, tables, and graphs. Initially, we delineate the type of data, which falls into two categories. *Qualitative* data is expressed in verbal or narrative form and is gathered through methods such as focus groups, interviews, open-ended questionnaire items, and less structured situations. Qualitative data is essentially information represented in the form of words. On the other hand, *Quantitative* data is numerical, with values that could be large or small, corresponding to specific categories or labels. When analyzing a given dataset within a scientific investigation, [41] and [5] recommend adhering to two steps: Exploratory Data Analysis, aiming to uncover key statistical properties through simple graphical and numerical studies. Confirmatory Data Analysis, involving the evaluation of evidence using traditional statistical tools like significance, inference, and confidence.

Following data analysis, we proceed to modeling; multivariate models having more parameters than univariate ones, introduce additional sources of error as each parameter needs estimation. Outliers can exert a greater impact on multivariate predictions. Variables not only depend on their past values but also exhibit interdependency, which is leveraged for forecasting future values. Given the gathered information, we recognize the importance of reframing datasets. This allows the application of standard linear and nonlinear machine learning algorithms to address specific challenges. In a research project at Los Alamos National Laboratory in the summer of 2019, we worked with a multivariate dataset containing missing values. Addressing these missing values was a crucial step in data preparation before proceeding with any modeling applications. Imputation methods must be applied

before moving forward with the training dataset and predictive techniques. The following section we will introduce some of the methods that we have applied to our research throughout the Los Alamos National Laboratory project.

1.2 Missing values

In our summer research of 2019, we confronted the issue of missing values. When dealing with large raw datasets, it is not unusual to encounter gaps in the information. There are two approaches to addressing incomplete data:

- omit the entire record that does not contain information.
- Impute the missing information.

Avoiding the omission of data is crucial because it can introduce bias and compromise the representativeness of results. In statistics, imputation comes into play, involving the substitution of missing data with estimated values. "Unit imputation" is employed when substituting for a data point, while "item imputation" is utilized for replacing a component of a data point. Non-time-series-specific imputation methods, such as mean, median, and mode imputation, calculate the appropriate measure and replace missing data (NA) with values.[32] Imputation serves as a technique for filling in missing values with estimates, aiming to leverage established relationships found in the valid values of the dataset to estimate the missing ones. As mentioned earlier, Mean/Mode/Median imputation stands among the most commonly used methods. This involves replacing the missing data for a specific attribute with the mean or median (for quantitative attributes) or mode (for qualitative attributes) of all known values of that variable. It's crucial to recognize different types of missing data: (MCAR) Missing Completely at Random, where the likelihood of an observation being missing is unrelated to its value or any other values in the dataset; Missing at Random (MAR), indicating that the probability of an observation being missing is related to the values of some other observed variables; and (MNAR) Missing Not at

Random, suggesting that the probability of an observation being missing is linked to its value.

Missing data can lead to three main issues: bias, increased complexity in data handling and analysis, and reduced efficiency. Imputation addresses these problems by replacing missing values with estimates based on available information. Post-imputation, the dataset can be analyzed using standard techniques, including machine learning and neural networks. It's essential to note that the goal of imputation is not to predict individual missing values, as some may mistakenly believe. This misconception may arise from the hot deck imputation method, which aims to find the best match for each missing case by replacing it with an observed response from a similar unit.

Obtaining a more accurate estimate for each missing value does not necessarily translate to a better overall estimate for the parameters of interest [32]. Rubin provides a counterexample: consider a biased coin with a known truth model A (0.6 heads, 0.4 tails) and an alternative model B (asserting two heads). Using model A for imputations results in a hit rate of 0.52, while model B yields a hit rate of 0.6. However, this does not imply that model B is superior for handling missing values. Utilizing model B leads to invalid statistical inference, predicting all future coin tosses as heads, contradicting the estimand Q (fraction of heads). Model A provides consistent estimates for such scientific estimands. Various imputation methods, ranging from simple to complex, exist. While maintaining the full sample size, advantageous for bias and precision, they may introduce different types of bias. When a single imputation strategy is employed, the standard errors of estimates tend to be underestimated. The underlying concept is that there is substantial uncertainty surrounding the missing values. Opting for a single imputation assumes knowledge of the true value with certainty, which may not accurately reflect the reality of missing data across multiple variables in an analysis. [32].

Various imputation methods, ranging from simple to complex, exist. While maintaining the full sample size, advantageous for bias and precision, they may introduce different types of bias. When a single imputation strategy is employed, the standard errors of estimates

tend to be underestimated. The underlying concept is that there is substantial uncertainty surrounding the missing values. Opting for a single imputation assumes knowledge of the true value with certainty, which may not accurately reflect the reality of missing data across multiple variables in an analysis. In such cases, establishing a model for a single partially observed variable y given a set of fully observed X variables is not sufficient. Instead, the dataset should be viewed as a multivariate outcome, where any component can be missing. A direct approach to imputing missing data in multiple variables involves fitting a multivariate model to all variables with missing values. This approach extends to allow both the outcome Y and the predictors X to be vectors. The primary challenge of this approach lies in the significant effort required to establish a reasonable multivariate regression model. The most commonly used distributions for this purpose are multivariate normal or t distributions for continuous outcomes, and a multi-normal distribution for discrete outcomes. Concluding this chapter, with our motivation for research encompasses harnessing the strengths of machine learning methodologies and optimization methods to comprehensively address the issue of misdiagnosis, aiming for a deeper understanding and improvement in medical diagnosis.

1.3 Motivation

In current times, the terminology associated with artificial intelligence (AI), machine learning (ML), and deep learning has permeated various sectors, including business, healthcare, industries, and the military. Across these domains, the importance of precise data prediction and analysis remains paramount, regardless of the volume of collected data. The pivotal role of AI initiates with the analysis of large datasets using scientific techniques, particularly within the realm of machine learning. AI and deep learning applications have revolutionized the identification of decision-making patterns, significantly reducing the need for human intervention and minimizing errors. This research is driven by the objective of elevating the accuracy of medical diagnoses in the healthcare sector, employing machine

learning and proposing optimization methods to refine existing approaches. Throughout my tenure at the University of Texas at El Paso (UTEP), I have immersed myself in the realms of machine learning and artificial intelligence, focusing on their applications in healthcare. Our research endeavors aim to present machine learning models that incorporate feature optimization, enhancing diagnostic outcomes. Additionally, we propose an extension of Graph Neural Networks as our meta-learning model, refining the algorithm to achieve optimal results. Our exploration is motivated by the persistent challenge in healthcare—diagnostic errors. These errors, whether inaccurate or delayed, have been a blind spot in healthcare, prompting the need for advancements in machine learning and AI. The subsequent narrative delves into the impact of these technological advancements on healthcare, particularly addressing concerns related to over-diagnosis. This phenomenon involves assigning diagnostic labels to certain conditions that may not significantly impact an individual's health and well-being, as highlighted in [10]. Machine learning and AI play a crucial role in uncovering hidden insights in clinical decision-making, facilitating connections between patients and resources for self-assessments. The significance of these advancements has been particularly evident in our recent experiences with the COVID-19 pandemic, showcasing the growth and potential of these technologies in the healthcare landscape.

Our research is dedicated to enhancing the precision of medical diagnoses by employing advanced machine learning techniques and optimization methods. The chapters follow a structured approach: Chapter two introduces a brief background to our research in machine learning methodologies applications. Chapter three will cover methods and material, chapter four presents research datasets, elaborated on in chapters four detailing applications and simulations of our summarizing findings/results to medical diagnosis contributions. Chapter five concludes the study by summarizing closing remarks and motivation for future work. The primary goal is to evaluate the possibility of producing valuable predictions through the application of machine learning methodologies, harnessing their diverse capabilities to enhance medical diagnosis.

Chapter 2

Background Research

2.1 Los Alamos National Laboratory Research

The prevalence of missing data poses a common challenge in statistical analysis, intelligent techniques have become increasingly integral in addressing critical and recurring issues across industrial, commercial, and academic domains. Artificial Intelligence (AI) has experienced significant dissemination and practical application in recent years, demonstrating continuous and remarkable growth. In collaboration with Claire McKay Bowen, Ph.D. (Postdoctoral Research Associate) of the Statistical Sciences Group (CCS-6) and Joanne R. Wendelberger, Ph.D., Lead of the Statistical Sciences Group (CCS-6) at Los Alamos National Laboratory (LANL), during the summer of 2019. We engaged in research that initially focused on time series forecasting within financial datasets. This exploration extended to forecasting environmental crashes and credit card defaults in multivariate data using machine learning methodologies. This collaboration led to the conceptualization of my LANL summer project titled "Analyzing the Effects of Missing Data with Machine Learning Algorithms." The identified problem revolves around educational and career datasets, which commonly exhibit missing data, particularly as data collection spans from K-12 to college and graduation, exacerbating the issue.

Benefiting from the resources generously provided by the Computational Statistical Group (CCS-6) at LANL, I successfully undertook my project. I extend my sincere appreciation to Los Alamos and, particularly, to Claire and Joanne for their invaluable support and inspiration for future research endeavors.

Method Name	Category	Software	Reference
Mean impute (`mean`)	Mean		Little and Rubin (1987)
Expectation-Maximization (`EM`)	EM		Dempster et al. (1977)
EM with Mixture of Gaussians and Multinomials	EM		Ghahramani and Jordan (1994)
EM with Bootstrapping	EM	`Amelia II`	Honaker et al. (2011)
K-Nearest Neighbors (`knn`)	K-NN	`impute`	Troyanskaya et al. (2001)
Sequential K-Nearest Neighbors	K-NN		Kim et al. (2004)
Iterative K-Nearest Neighbors	K-NN		Caruana (2001); Brás and Menezes (2007)
Support Vector Regression	SVR		Wang et al. (2006)
Predictive-Mean Matching (`pmm`)	LS	`MICE`	Buuren and Groothuis-Oudshoorn (2011)
Least Squares	LS		Bø et al. (2004)
Sequential Regression Multivariate Imputation	LS		Raghunathan et al. (2001)
Local-Least Squares	LS		Kim et al. (2005)
Sequential Local-Least Squares	LS		Zhang et al. (2008)
Iterative Local-Least Squares	LS		Cai et al. (2006)
Sequential Regression Trees	Tree	`MICE`	Burgette and Reiter (2010)
Sequential Random Forest	Tree	`missForest`	Stekhoven and Bühlmann (2012)
Singular Value Decomposition	SVD		Troyanskaya et al. (2001)
Bayesian Principal Component Analysis	SVD	`pcaMethods`	Oba et al. (2003); Mohamed et al. (2009)
Factor Analysis Model for Mixed Data	FA		Khan et al. (2010)

Figure 2.1: Table of imputation methods

2.2 Introduction to missing data

As missing data is common problem in real-world settings which has lead to significant attention in the statistical literature. There are flexible frameworks based on formal optimization to impute missing values. Research has shown that these frameworks can readily incorporate various predictive models such as K-nearest neighbors, Support Vector Machines, and Decision tree based methods, and can be adapted for multiple imputation. With missing data phenomena is arguably the most common issue encountered by machine learning practitioners when analyzing real-world data (raw data). As we know that for many statistical models and machine learning algorithms rely on complete data sets, it becomes extremely important to handle the missing values appropriately. Below you will find a table of imputation methods for deeper perception on methodologies and their references.

There are some machine learning studies done, that have shown some algorithms naturally account for missing data and there is no need for preprocessing. In particular, CART and K-means have been adapted for problems with missing data (Breiman et al.., 1987; Wagsta, 2004). But as for many other situations, missing values need to be imputed prior

to any statistical analyses on the complete data set. Going forward lets assume that we are given data,

$$X = \{x_1, ..., x_n\}$$

with missing entries $x_{id}, (i, d) \in \mathcal{M}$. The objective is to impute the values of the missing data that mirror the underlying complete data as closely as possible. From this point, one may apply pattern recognition using machine learning methods on the imputed data and results should be complementary to the complete data given.

2.3 Educational data description

On this study data we started with a subset data description,which lead us with an initial simulation of data before moving forward. Our multivariate data had some dependent variables and Independent variables, with the total of 14 variables, some discrete others continuous. Below figure for visual of data description. So before moving forward with our imputation, we needed a simulated data-set. In order to simulated a data-set, the sample data used for prediction is given by csv file of 1000 random generated students. Now for a more complex data we have simulated a data-set of 1000 students with name for columns for input given by :

- **Gender**: female=1; male=0.
- **Race**: white=1; African-American (black)=2; Asian =3; American Indian or Alaskan Native =4, Native Hawaiian or other Pacific Islander , ethnicity: non-hispanic white =1; hispanic or latino =2 ; non-hispanic Black =3; Chinese=4; European=5 ; Arab =6 ; Indigenous=7; Filipino = 8.
- Immigrant: Yes=1, No=0
- High school-age=numeric from 14-15
- Rank

- Dual-credit, Ap credit
- Combined SAT and ACT scores
- Parents highest education received
- Family gross income

The variable that will be used for forecasting will be the *output of degree completion.*

In order to test effectiveness of the methods presented in the Educational data I generated test data with different properties. It is also possible to predict missing data by using different mathematical techniques along with statistical tools. I effectively used existing statistical software for missing data (MICE, missForest, MI,). The results for methods well executed in this chapter. As research shows that in order to predict missing data it is possible to apply machine learning techniques (neural networks), this application of simulation and prediction will also be showcased below. Through the usage of neural networks a module was developed to simulate the process described above. We simulated independent paths of our model using different time steps for the data sets.

2.4 Imputation methods

In our analysis of missing data in an Educational dataset using R, the **MICE (Multivariate Imputation via Chained Equations)** package was employed. MICE is a commonly used package that generates multiple imputations, addressing uncertainty in missing values, unlike single imputation methods such as the mean. It operates under the assumption that missing data are Missing at Random (MAR), implying that the probability of data being missing depends only on observed values and can be predicted using them. MICE imputes data on a variable-by-variable basis, utilizing specific imputation models for each variable. [18]

Variable Name	Variable Type	Description or Coding
Dependent Variable		
Degree completion	Time-varying	Dichotomous variable indicating whether a student received a STEM degree by postsecondary public institution for an academic year
Independent Variables		
Student Characteristics		
Gender	Time-invariant	M = Male, F = Female
Gifted/Talented	Time-varying	00, 01
Immigrant	Time-invariant	00, 01
Race/ethnicity	Time-invariant	*New Ethnic Origin*: 1 = Hispanic or Latino origin, 2 = Not Hispanic or Latino origin, 3 = Not answered *Race*: 1 = White, 2 = African-American/Black, 4 = Asian, 5 = American Indian or Alaskan native, 6 = International, 7 = Unknow or Not Reported, 8 = Native Hawaiian or Other Pacific Islander
Family gross income	Time-invariant	Continuous variable indicating total parental income
Parent education	Time-invariant	*Highest grade level mother completed*: 1 = Elementary, 2 = High school, 3 = College or beyond, 4 = Unknown *Highest grade level father completed*: 1 = Elementary, 2 = High school, 3 = College or beyond, 4 = Unknown
Age at enrollment	Time-invariant	Continuous variable indicating age at enrollment
Academic Preparation for College		
Combined SAT (or ACT) score	Time-invariant	Combined scores from the SAT Mathematics and Evidenced-based Reading and Writing (EBRW) tests
High school rank	Time-invariant	1 = Accepted and ranked in top 10% of high school graduating class, 2 = Accepted and ranked in 11-25% of high school graduating class, 3 = Others
Dual credit	Time-invariant	Dichotomous indicator of whether a student ever earned dual credits
AP credit	Time-invariant	Dichotomous indicator of whether a student ever earned AP credits
College Experience		
First-year GPA	Time-invariant	Calculated first-year college GPA
GPA change	Time-invariant	The difference between first-year calculated GPA and last-year cumulative GPA

Figure 2.2: Data description

MICE-Results Employing the MICE imputation method, we effectively simulated educational data, resulting in a comprehensive dataset with well-balanced imputation weights. Subsequent sections showcase simulations of datasets, incorporating the application of neural networks with MICE as the chosen imputation method.

2.4.1 Amelia

We studied the method Amelia II, which is a tool for multiple imputation that employs a bootstrapping-based algorithm, delivering results comparable to standard approaches like IP or EM but with significantly faster processing and the capability to handle numerous variables. It extends existing methods by accommodating trends across observations within a cross-sectional unit and allowing for the incorporation of expert beliefs through priors on missing values.[11] The program provides diagnostics for assessing the fit of multiple imputation models.

In the context of Amelia, multiple imputation entails generating m imputed values for each missing cell in the data matrix, resulting in m completed datasets. These datasets maintain consistent observed values but differ in the imputed values, reflecting uncertainty about the missing data. After imputation, Amelia saves the m datasets, allowing the application of statistical methods to each set and subsequent combination of results.

The default imputation count (m=5) is typically sufficient unless the missingness rate is exceptionally high. In contrast to other methods like list-wise deletion or mean substitution, multiple imputation, when done properly, avoids bias or inefficiency, preserving data relationships while incorporating all observed data in partially missing rows. [19].

2.4.2 missForrest

missForest serves as a nonparametric imputation method suitable for diverse types of data. It demonstrates adaptability in handling mixed-type variables, nonlinear relationships, intricate interactions, and scenarios with high dimensionality (where the number of variables,

Table 2.1: Data for MICE

gender	race	ethnicity	immigrant	highschool_age
1	7	8	2	14.5
0	2	2	0	14
0	1	1	0	14
1	6	6	2	13.8
0	1	1	0	14
0	1	1	0	14
1	5	6	2	13.8
0	2	2	0	14
1	5	6	2	13.8
0	3	3	1	14
0	2	2	0	14
1	6	6	2	13.8
0	4	4	1	14
1	6	7	2	15
1	5	6	1	13.8
0	4	4	1	14
0	3	3	0	14
1	5	5	1	14
0	2	3	0	14
0	2	3	0	14

Table 2.2: Missing data for MICE

	gender	race	ethnicity	immigrant	highschool_age
1	1	7	8	2	14.5
2	0	2	2	0	14
3	NA	1	1	0	14
4	1	6	6	2	13.8
5	0	1	1	NA	14
6	NA	1	1	0	14
7	1	5	6	2	13.8
8	NA	2	2	0	14
9	1	5	6	2	13.8
10	0	3	3	NA	14
11	0	NA	NA	0	14
12	1	6	6	2	13.8
13	NA	4	4	1	14
14	1	6	7	2	NA
15	1	5	6	1	13.8
16	0	4	4	1	14
17	0	3	3	0	14
18	NA	5	NA	1	14
19	0	2	3	0	14
20	0	2	3	0	14

Table 2.3: Data after imputation for MICE

	X	gender	race	ethnicity	immigrant	highschool_age
1	1	1	7	8	2	14.5
2	2	0	2	2	0	14
3	3	0	1	1	0	14
4	4	1	6	6	2	13.8
5	5	0	1	1	0	14
6	6	0	1	1	0	14
7	7	1	5	6	2	13.8
8	8	0	2	2	0	14
9	9	1	5	6	2	13.8
10	10	0	3	3	0	14
11	11	0	1	1	0	14
12	12	1	6	6	2	13.8
13	13	0	4	4	1	14
14	14	1	6	7	2	15
15	15	1	5	6	1	13.8
16	16	0	4	4	1	14
17	17	0	3	3	0	14
18	18	1	5	6	1	14
19	19	0	2	3	0	14
20	20	0	2	3	0	14

Table 2.4: Data for Amelia

gender	race	ethnicity	immigrant	highschool_age
1	7	8	2	14.5
0	2	2	0	14
0	1	1	0	14
1	6	6	2	13.8
0	1	1	0	14
0	1	1	0	14
1	5	6	2	13.8
0	2	2	0	14
1	5	6	2	13.8
0	3	3	1	14
0	2	2	0	14
1	6	6	2	13.8
0	4	4	1	14
1	6	7	2	15
1	5	6	1	13.8
0	4	4	1	14
0	3	3	0	14
1	5	5	1	14
0	2	3	0	14
0	2	3	0	14

Table 2.5: Sample Missing Data for Amelia

	gender	race	ethnicity	immigrant	highschool_age
1	1	7	8	2	14.5
2	0	2	2	0	14
3	NA	1	1	0	14
4	1	6	6	2	13.8
5	0	1	1	NA	14
6	NA	1	1	0	14
7	1	5	6	2	13.8
8	NA	2	2	0	14
9	1	5	6	2	13.8
10	0	3	3	NA	14
11	0	NA	NA	0	14
12	1	6	6	2	13.8
13	NA	4	4	1	14
14	1	6	7	2	NA
15	1	5	6	1	13.8
16	0	4	4	1	14
17	0	3	3	0	14
18	NA	5	NA	1	14
19	0	2	3	0	14
20	0	2	3	0	14

Table 2.6: Data after imputation for Amelia

	X	gender	race	ethnicity	immigrant	highschool_age
1	1	1	7	8	2	14.5
2	2	0	2	2	0	14
3	3	0.399867	1	1	0	14
4	4	1	6	6	2	13.8
5	5	0	1	1	0.291427	14
6	6	0.120429	1	1	0	14
7	7	1	5	6	2	13.8
8	8	-0.01615	2	2	0	14
9	9	1	5	6	2	13.8
10	10	0	3	3	0.890275	14
11	11	0	1.805109	1.731756	0	14
12	12	1	6	6	2	13.8
13	13	0.256045	4	4	1	14
14	14	1	6	7	2	14.17264
15	15	1	5	6	1	13.8
16	16	0	4	4	1	14
17	17	0	3	3	0	14
18	18	0.702625	5	4.300979	1	14
19	19	0	2	3	0	14
20	20	0	2	3	0	14

p, greatly exceeds the number of observations, n). An essential requirement for missForest is that observations (rows of the supplied data frame) need to be pairwise independent.

This algorithm leverages the random forest approach, relying on the R implementation randomForest by Andy Liaw and Matthew Wiener. In simple terms, for each variable, missForest fits a random forest model on the observed data and predicts the missing values iteratively. This process is repeated until a stopping criterion is met or a user-specified maximum number of iterations is reached.

To provide additional insights, Stekhoven and Bühlmann's work [39] offers further details on the methodology behind missForest. It's crucial to note that missForest operates iteratively, continually updating the imputed matrix variable-wise, and evaluates its performance between iterations. This iterative nature ensures a dynamic and evolving imputation process, enhancing the accuracy and reliability of the imputed values. For a comprehensive understanding of missForest's functionality, it's recommended to refer to the user guide by Buuren [11].

Table 2.7: Description of columns

1	gender
2	race
3	ethnicity
4	immigrant
5	highschoolage
6	highschool rank
7	dual credit
8	ap credit
9	combined sat
10	parent mother
11	parent father
12	gross income
13	degree completion

Table 2.8: Sample input data

0	1	1	0	14	2	0	0	0	1	1	37368	0
0	3	4	1	14	2	0	0	1	1	1	59473	1
1	6	7	2	15	3	1	1	2	3	3	88947	2
1	7	8	2	14.5	3	1	1	2	4	4	100000	2
1	6	7	2	15	3	1	1	2	3	3	92631	2
0	4	4	1	14	2	0	0	1	1	1	63157	1
0	3	3	1	14	2	0	0	1	1	1	55789	1
0	2	3	0	14	2	0	0	0	1	1	48421	0
1	7	8	2	14.5	3	1	1	2	4	4	96315	2
0	2	2	0	14	2	0	0	0	1	1	41052	0
0	1	1	0	14	1	0	0	0	1	1	30000	0
1	5	6	1	13.8	2	1	1	1	2	1	77894	1
1	5	6	2	13.8	2	1	1	2	2	2	81578	2
0	3	3	0	14	2	0	0	0	1	1	52105	0
1	5	5	1	14	2	1	1	1	2	1	74210	1
1	4	5	1	14	2	1	1	1	1	1	70526	1
1	6	7	2	15	3	1	1	2	2	2	85263	2
0	2	2	0	14	2	0	0	0	1	1	44736	0
1	4	5	1	14	2	1	1	1	1	1	66842	1
0	1	1	0	14	1	0	0	0	1	1	33684	0

Similar results it is possible to get by using different software packages.

2.4.3 MI

The mi package in R offers a plethora of features designed to provide users with a comprehensive understanding of the imputation process and facilitate the evaluation of resulting models and imputations. These features encompass various aspects, offering flexibility and transparency in the imputation procedure.

Predictor, Model, and Transformation Options: Users have the liberty to make informed decisions by selecting predictors, models, and transformations for the chained imputation models. This empowers them to tailor the imputation process to the specific characteristics of their dataset, ensuring a more nuanced and contextually relevant imputation. Residual Plots for Model Fit Assessment: The package incorporates both standard and binned residual plots that serve as valuable tools for assessing the fit of conditional distributions used during imputation. These plots offer visual insights into the goodness of fit, allowing users to identify patterns, trends, or potential discrepancies in the imputation models. Chained Imputation Models: mi facilitates the creation of chained imputation models, a versatile approach where the imputation of missing values is performed iteratively, considering the dependencies between variables. This method enhances the accuracy of imputations by capturing complex relationships within the data. Transparent Imputation Process: The package emphasizes transparency in the imputation process, enabling users to delve into the details of the chosen predictors, models, and transformations. This transparency is crucial for building confidence in the imputed results and understanding the underlying mechanisms of the imputation models. Reference: The features and methodologies employed by mi are detailed in the reference [40], providing users with a comprehensive guide to leverage the capabilities of the package effectively. This reference serves as a valuable resource for understanding the theoretical foundations and practical applications of mi in the realm of missing data imputation. In summary, the mi package in R stands out for its user-friendly features that empower users to customize and assess the imputation pro-

Table 2.9: Sample data with missing values.

1	0	1	1	0	14	2	0	0	0	1	1	37368	0
2	0	NA	4	1	14	2	0	0	1	1	1	59473	1
3	1	6	7	2	15	3	1	1	2	3	3	88947	2
4	1	7	NA	2	14.5	3	1	1	2	4	4	100000	2
5	1	6	7	2	15	NA	1	NA	2	3	3	92631	2
6	0	4	4	1	14	2	NA	0	1	NA	1	63157	1
7	0	3	3	1	14	NA	0	0	NA	NA	1	55789	1
8	0	2	3	0	14	2	NA	0	0	NA	1	48421	0
9	1	7	8	2	14.5	3	NA	1	2	4	4	96315	2
10	0	NA	2	0	14	2	0	0	0	1	1	41052	0
11	0	1	1	0	14	1	0	0	0	1	1	30000	0
12	1	5	NA	1	13.8	2	1	1	1	2	1	77894	1
13	1	5	6	2	13.8	2	1	1	2	2	2	81578	2
14	0	3	3	0	14	NA	0	0	0	1	1	52105	0
15	1	5	5	1	NA	2	1	1	1	2	1	74210	1
16	1	4	5	1	14	NA	1	1	1	1	1	70526	1
17	1	6	7	2	15	3	1	1	2	2	2	85263	2
18	0	2	2	0	14	2	0	0	0	1	1	44736	0
19	1	4	5	1	14	NA	1	1	1	1	1	NA	1
20	0	1	1	0	14	1	NA	0	0	1	1	33684	0

Table 2.10: Data after imputation

1	1	0	1	1	0	14	2	0	0	0	1	1	37368	0
2	2	0	3.493	4	1	14	2	0	0	1	1	1	59473	1
3	3	1	6	7	2	15	3	1	1	2	3	3	88947	2
4	4	1	7	7.53	2	14.5	3	1	1	2	4	4	100000.0	2
5	5	1	6	7	2	15	2.933	1	1	2	3	3	92631	2
6	6	0	4	4	1	14	2	0.25	0	1	1.05	1	63157	1
7	7	0	3	3	1	14	1.997	0	0	0.553	1.04	1	55789	1
8	8	0	2	3	0	14	2	0	0	0	1.03	1	48421	0
9	9	1	7	8	2	14.5	3	1	1	2	4	4	96315	2
10	10	0	1.576	2	0	14	2	0	0	0	1	1	41052	0
11	11	0	1	1	0	14	1	0	0	0	1	1	30000	0
12	12	1	5	5.242	1	13.8	2	1	1	1	2	1	77894	1
13	13	1	5	6	2	13.8	2	1	1	2	2	2	81578	2
14	14	0	3	3	0	14	1.932	0	0	0	1	1	52105	0
15	15	1	5	5	1	13.941	2	1	1	1	2	1	74210	1
16	16	1	4	5	1	14	1.985	1	1	1	1	1	70526	1
17	17	1	6	7	2	15	3	1	1	2	2	2	85263	2
18	18	0	2	2	0	14	2	0	0	0	1	1	44736	0
19	19	1	4	5	1	14	1.965	1	1	1	1	1	70331.232	1
20	20	0	1	1	0	14	1	0.04	0	0	1	1	33684	0

cess. From choosing predictors and models to evaluating model fit through residual plots, mi facilitates a thorough exploration of the imputation journey, contributing to informed decision-making in handling missing data.

2.4.4 Finding Missing Data with Neural Network

Given the heightened interest in deep learning over the past decade, it becomes increasingly crucial to establish cohesive tools that empower practitioners to handle missing data seamlessly using diverse neural networks. Artificial Neural Networks (ANNs) employ nonlinear mathematical equations to iteratively establish meaningful connections between input and output variables through a learning process. In our approach, we utilized backpropagation networks, complemented by various optimization methods, for data classification. The typical structure of backpropagation networks comprises an input layer, one or more hidden layers, and an output layer, each housing numerous neurons. ANNs exhibit a remarkable ability to adeptly handle the nonlinear and interactive effects of explanatory variables. However, a notable drawback is the challenge in deriving a simple probabilistic formula for classification.[11]

One innovative aspect of our methodology involves feeding neural networks with missing data. We introduce model uncertainty regarding missing attributes through probability density functions, eliminating the necessity for direct completion (imputation) using singular values. The theoretical underpinning of this approach ensures a nuanced treatment of missing data, aligning with the complexity inherent in neural network architectures. This not only enhances the adaptability of neural networks to missing data scenarios but also aligns with the overarching goal of establishing unified tools for practitioners working with artificial neural networks in the realm of deep learning.

Table 2.11: Sample data with missing values (10%) for MI.

1	0	2	2	0	14	2	0	0	0	1	1	44482	0
2	1	7	7	2	15	3	1	1	2	3	3	92758	2
3	0	1	1	0	14	1	0	0	0	1	1	30000	0
4	0	1	1	0	14	2	0	0	0	1	1	37241	0
5	1	5	6	2	13.8	2	1	1	2	2	1	78275	2
6	1	6	7	2	15	3	1	1	2	2	2	85517	2
7	1	6	7	2	15	3	1	1	2	3	3	87931	2
8	0	2	2	0	14	2	0	0	0	1	1	42068	0
9	1	NA	6	2	13.8	2	1	1	2	2	2	83103	2
10	0	1	2	0	14	2	0	0	0	1	1	39655	0
11	0	1	1	0	14	1	0	0	0	1	1	34827	0
12	0	3	3	1	14	2	0	0	1	1	1	54137	1
13	1	5	6	2	13.8	2	1	1	2	2	2	80689	2
14	0	2	2	0	14	2	0	0	0	1	1	46896	0
15	0	4	4	1	14	2	0	0	1	1	1	63793	1
16	0	3	4	1	14	2	0	0	1	1	1	NA	1
17	1	5	6	1	13.8	2	1	1	1	2	1	75862	1
18	1	7	8	2	14.5	3	1	1	2	4	4	97586	2
19	0	4	4	1	14	2	0	0	1	1	1	61379	1
20	1	4	5	1	14	2	1	1	1	1	1	66206	1
21	1	5	5	1	14	2	1	1	1	2	1	73448	1
22	1	7	8	2	14.5	3	1	1	2	4	4	100000	2
23	0	3	3	1	14	2	0	0	1	1	1	56551	1
24	0	2	3	0	14	2	0	0	0	1	1	49310	0
25	1	4	5	1	14	2	1	1	1	1	1	71034	1

Table 2.12: Sample data after imputation (10%) for MI.

0	2	2	0	14	2	0	0	0	1	1	44482	0
1	7	7	2	15	3	1	1	2	3	3	92758	2
0	1	1	0	14	1	0	0	0	1	1	30000	0
0	1	1	0	14	2	0	0	0	1	1	37241	0
1	5	6	2	13.8	2	1	1	2	2	1	78275	2
1	6	7	2	15	3	1	1	2	2	2	85517	2
1	6	7	2	15	3	1	1	2	3	3	87931	2
0	2	2	0	14	2	0	0	0	1	1	42068	0
1	6	6	2	13.8	2	1	1	2	2	2	83103	2
0	1	2	0	14	2	0	0	0	1	1	39655	0
0	1	1	0	14	1	0	0	0	1	1	34827	0
0	3	3	1	14	2	0	0	1	1	1	54137	1
1	5	6	2	13.8	2	1	1	2	2	2	80689	2
0	2	2	0	14	2	0	0	0	1	1	46896	0
0	4	4	1	14	2	0	0	1	1	1	63793	1
0	3	4	1	14	2	0	0	1	1	1	58965	1
1	5	6	1	13.8	2	1	1	1	2	1	75862	1
1	7	8	2	14.5	3	1	1	2	4	4	97586	2
0	4	4	1	14	2	0	0	1	1	1	61379	1
1	4	5	1	14	2	1	1	1	1	1	66206	1
1	5	5	1	14	2	1	1	1	2	1	73448	1
1	7	8	2	14.5	3	1	1	2	4	4	100000	2
0	3	3	1	14	2	0	0	1	1	1	56551	1
0	2	3	0	14	2	0	0	0	1	1	49310	0
1	4	5	1	14	2	1	1	1	1	1	71034	1

Layer for processing missing data

The methodology for feeding neural networks with missing data, representation of missing data point is denoted by (x, J), where $x \in \mathbb{R}^D$ and $J \subset \{1, ..., D\}$ the set of attributes with missing values. Upon each missing point (x, J) we associate the subspace consisting all of the points which coincide with x on known coordinates $J' = \{1, ..., N\}/J$:

$$S = Aff[x, J] = x + span(e_J),$$

where $e_J = [e_j]_{j \in J}$ and e_j is the j^{th} vector in $\mathbb{R}^D$.

Assumption made is that the values that at the missing attributes come from the unknown D-dimensional probability distribution F. Then we can model the unobserved values of (x, J) by restricting F to subspace $S = Aff[x, J]$. Now it is possible that the values of incomplete data point (x, J) are described by the conditional density function [36]

$$F_S : S \rightarrow \mathbb{R}$$

given by:

$$F_S(x) = \begin{cases} \frac{1}{\int F(s)ds} F(x), & for\ x \in S, \\ 0, & otherwise. \end{cases}$$

Predicting One Missing Variable with Neural Network

To process probability density functions (representing missing data points) by neural networks, we generalize the neuron's activation function. We define the generalized response (activation) of a neuron $n : \mathbb{R}^D \rightarrow \mathbb{R}$ on F_S as the mean output:

$$n(F_S) = E[n(x)|x \sim F_S] = \int n(x) F_S(x) dx.$$

From the neurons response we move back to same step size as before first layer while the rest of network architecture can remain unchanged. Basic requirement is the ability of computing expected value with respect to F_S.

Recall that the ReLU neuron is given by,

$$ReLU_{w,b}(x) = max(w^T x + b, 0),$$

where $w \in \mathbb{R}^D$ and $b \in \mathbb{R}$ are the bias.

With TRUE vales being $= 0, 0, 1, 1, 1$. As we can see our neural network was able to forecast to the closest value of given TRUE value to data.

Summary of the accuracy of prediction is given below.

100 epochs

Number of Errors=476

Number of rows=1000

Number of correct answers in percent=52.4

Number of errors in percent=47.6

200 epochs

number of Errors=184

Number of rows=1000

Number of correct answers in percent=81.6

Number of errors in percent=18.4

500 epochs

Number of Errors=100

Number of rows=1000

Number of correct answers in percent=90.0

Number of errors in percent=10.0

1000 epochs

Number of Errors=0

Number of rows=1000

Number of correct answers in percent=100.0

Number of errors in percent=0.0

2.4.5 Classification of Credit Card Default

For a brief introduction to our dataset, regarding the default of credit card problem, this case of information of customers default payments come from Taiwan. We will us default binary result of classification - credible or not credible clients. Our indicates that payment date was on October of 2005; Taiwan bank collected a cash and credit card issuer. There is a total of 25,000 observations, 5529 observations (22.12%) are the cardholders with default payment. The dataset will be set employed as a binary variable, default payment (Yes = 1, No=0), as the response variable. Within our dataset we have used the following of 23 variables as explanatory variables denoted as:

- X1: Amount of the given credit (NT dollar): it includes both the individual consumer credit and his/her family (supplementary) credit.
- X2: Gender (1 = male; 2 = female).
- Education (1 = graduate school; 2 = university; 3= high school; 4 = others).
- X4: Marital status (1 = married; 2 = single; 3 = others).
- X5: Age (year).
- X6-X11: History of past payment. We tracked the past monthly payment records (from April to September, 2005) as follows: X6 = the repayment status in September, 2005; X7 = the repayment status in August, 2005 ; . . .; X11 = the repayment status in April, 2005. The measurement scale for the repayment status is: -1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months; . . .; 8 = payment delay for eight months; 9 = payment delay for nine months and above.
- X12-X17: Amount of bill statement (NT dollar). X12 = amount of bill statement in September, 2005; X13 = amount of bill statement in August, 2005; . . .; X17 = amount of bill statement in April, 2005.

- X18-X23: Amount of previous payment (NT dollar). X18 = amount paid in September, 2005; X19 = amount paid in August, 2005; . . .; X23 = amount paid in April, 2005.

The delinquency crisis of credit card debt increased in recent years in Taiwan. As observed on variables from data were collected year of 2005, based on Taiwan recent studies they were expecting a peak in the third quarter of year 2006 (crisis on credit card debt). For Taiwan to increase their market share, the card-issuing banks of Taiwan over issued credit cards to unqualified candidates. Another factor that would have to be dealt with is the fact that most cardholders, irrespective of their repayment ability and lead to an overused credit card, this accumulates to a heavy credit card and cash debts. This crisis (crash) leads to major importance to business and banks, on having the ability predict customers' credit risk, and reduce the damage and uncertainty. Such methods are describe to be statistical methods, which are used to classifying applicants for credit into "good" and "bad" classes. Throughout the growth of Artificial Intelligence and machine learning these types of models/methods have become increasingly important with the dramatic growth in consumer credit in the past years.

Moreover, for the default data, we use the logistic model by defaulting on the credit card debt. For example, the probability of default given *balance* may be written as:

$$P_r(default = Yes|balance),$$

we will denote for not the above probability by $p(balance)$ for convenience. Moving on, we now see how we should model this relationship between X and Y with gathered information from pervious chapters. As for the binary classification problem,

$$p(X) = Pr(Y = 1|X)$$

as for X, recall from perviously stated, that the linear regression model

$$p(X) = \beta_0 + \beta_1 X,$$

where the goal is to use the default = Yes to predict *balance.*

For a quick intuition on the modeling perspective, we copy $p(X)$ using a function that gives outputs between 0 and 1 for all values of X, where

$$p(X) = \beta_0 + \beta_1 X.$$

We use the logistic function,

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}.} \tag{2.1}$$

To fit a model in equation (2.1) we use least square method, after some manipulation of equation (2.1),

$$\frac{p(X)}{1 - p(X)} = e^{\beta_0+\beta_1 X} \tag{2.2}$$

Taking the log of equation (2.2) gives us,

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

where the left hand side is called the log-odds or logit. With further production estimating the coefficients we note that:

- β_0 and β_1 in equation (2.1) are unknown and must be estimated using the training data.

- Using the method of least squares to fit the coefficients as shown previously [2].

On the figure 2.3 a figure on the credit card clients, for data visuals.

Data was randomly divided into two groups, one for model training and the other to validate the model. The training data is based on error rates, in our research we will show how the artificial neural network (ANN) is the best models in the classification methods, along with other CNN and RNN. We have the following Mathematica code, Mathematica as other softwares, has been growing in machine learning AI community. It is able to read the data from the csv file.

Appropriate code in Mathematica is given below.

```
DataMatrix = Import["defaultofcreditcardclients.csv"];
NumberOfRows = Dimensions[DataMatrix][[1]];
TrainingSet = Table[DataMatrix[[i, 1 ;; 23 ]] -> DataMatrix[[i, 24 ]],
     {i, 1, NumberOfRows}];
Prediction = Classify[TrainingSet, Method -> "NeuralNetwork"]
Prediction[{20000, 2, 2, 1, 24, 2, 2, -1, -1, -2, -2, 3913, 3102, 689, 0, 0, 0,
   0, 689, 0, 0, 0, 0}]
Prediction[{50000, 1, 2, 1, 57, -1, 0, -1, 0, 0, 0, 8617, 5670, 35835, 20940,
  19146, 19131, 2000, 36681, 10000, 9000, 689, 679}]
```

To move forward with the application of Artificial neural network (ANN) we have the following scatter plot to represent our data prediction.

Artificial neural networks perform classification more accurately than the others. In the predictive accuracy of probability of default, artificial neural networks have shown the best performance based on R2 (0.9647, close to 1), regression intercept (0.0145, close to 0), and regression coefficient (0.9971, close to 1). The predictive default probability produced by ANN is the only one that could be used to represent real probability of default.

Table 2.13: Classification accuracy

method	$(\frac{ErrorRate}{Training})$	Validation
k-nearest neighbor	0.18	0.45
Logistic regression	0.20	0.44
Neural networks	0.19	0.54
Naïve Bayes	0.21	0.21

As the perspective from risk control, when it comes to estimating the default is more meaningful than classifying clients into binary results - risky and non-risky. Therefore, artificial neural networks should be employed to score clients instead of other data mining techniques, such as logistic regression.

Chapter two effectively demonstrated the versatility of neural networks in adapting to various imputation methods in past studies done while studying the capabilities behind machine learning, showcasing their inherent capability to perform imputation tasks based on the specific network setup. The chapter highlighted successful simulations and imputations conducted on an educational dataset, a key focus of the summer research project in collaboration with Los Alamos National Laboratory and other research projects. Building on this foundation, the research extended its exploration into the realm of machine learning methods. This progression allowed for a deeper understanding of how these methods can be applied to medical diagnosis, addressing the critical need for accurate disease classification and mitigating the urgency surrounding misdiagnosis.

In Chapter four, we will present two medical datasets used in our optimized graph meta-learning model. The chapter will also explore the application of machine learning models and optimization techniques discussed in Chapter three. The goal is to extract valuable insights from medical data, contributing to the ongoing efforts to enhance diagnostic accuracy and healthcare outcomes using machine learning.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
1		X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15	X16	X17	X18	X19	X20	X21	X22	X23	Y
2	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	PAY_5	PAY_6	BILL_AMT	BILL_AMT	BILL_AMT	BILL_AMT	BILL_AMT	BILL_AMT	PAY_AMT	PAY_AMT	PAY_AMT	PAY_AMT	PAY_AMT	PAY_AMT	default pay
3	1	20000	2	2	1	24	2	2	-1	-1	-2	-2	3913	3102	689	0	0	0	0	689	0	0	0	0	1
4	2	120000	2	2	2	26	-1	2	0	0	0	2	2682	1725	2682	3272	3455	3261	0	1000	1000	1000	0	2000	1
5	3	90000	2	2	2	34	0	0	0	0	0	0	29239	14027	13559	14331	14948	15549	1518	1500	1000	1000	1000	5000	0
6	4	50000	2	2	1	37	0	0	0	0	0	0	46990	48233	49291	28314	28959	29547	2000	2019	1200	1100	1069	1000	0
7	5	50000	1	2	1	57	-1	0	-1	0	0	0	8617	5670	35835	20940	19146	19131	2000	36681	10000	9000	689	679	0
8	6	50000	1	1	2	37	0	0	0	0	0	0	64400	57069	57608	19394	19619	20024	2500	1815	657	1000	1000	800	0
9	7	500000	1	1	2	29	0	0	0	0	0	0	367965	412023	445007	542653	483003	473944	55000	40000	38000	20239	13750	13770	0
10	8	100000	2	2	2	23	0	-1	-1	0	0	-1	11876	380	601	221	-159	567	380	601	0	581	1687	1542	0
11	9	140000	2	3	1	28	0	0	2	0	0	0	11285	14096	12108	12211	11793	3719	3329	0	432	1000	1000	1000	0
12	10	20000	1	3	2	35	-2	-2	-2	-2	-1	-1	0	0	0	0	13007	13912	0	0	0	13007	1122	0	0
13	11	200000	2	3	2	34	0	0	2	0	0	-1	11073	9787	5535	2513	1828	3731	2306	12	50	300	3738	66	0
14	12	260000	2	1	2	51	-1	-1	-1	-1	-1	2	12261	21670	9966	8517	22287	13668	21818	9966	8583	22301	0	3640	0
15	13	630000	2	2	2	41	-1	0	-1	-1	-1	-1	12137	6500	6500	6500	6500	2870	1000	6500	6500	6500	2870	0	0
16	14	70000	1	2	2	30	1	2	2	0	0	2	65802	67369	65701	66782	36137	36894	3200	0	3000	3000	1500	0	1
17	15	250000	1	1	2	29	0	0	0	0	0	0	70887	67060	63561	59696	56875	55512	3000	3000	3000	3000	3000	3000	0
18	16	50000	2	3	3	23	1	2	0	0	0	0	50614	29173	28116	28771	29531	30211	0	1500	1100	1200	1300	1100	0
19	17	20000	1	1	2	24	0	0	2	2	2	2	15376	18010	17428	18338	17905	19104	3200	0	1500	0	1650	0	1
20	18	320000	1	1	1	49	0	0	0	-1	-1	-1	253286	246536	194663	70074	5856	195599	10358	10000	75940	20000	195599	50000	0
21	19	360000	2	1	1	49	1	-2	-2	-2	-2	-2	0	0	0	0	0	0	0	0	0	0	0	0	0
22	20	180000	2	1	2	29	1	-2	-2	-2	-2	-2	0	0	0	0	0	0	0	0	0	0	0	0	0
23	21	130000	2	3	2	39	0	0	0	0	0	-1	38358	27688	24489	20616	11802	930	3000	1537	1000	2000	930	33764	0
24	22	120000	2	2	1	39	-1	-1	-1	-1	-1	-1	316	316	316	0	632	316	316	316	0	632	316	0	1
25	23	70000	2	2	2	26	2	0	0	2	2	2	41087	42445	45020	44006	46905	46012	2007	3582	0	3601	0	1820	1
26	24	450000	2	1	1	40	-2	-2	-2	-2	-2	-2	5512	19420	1473	560	0	0	19428	1473	560	0	0	1128	1
27	25	90000	1	1	2	23	0	0	0	-1	0	0	4744	7070	0	5398	6360	8292	5757	0	5398	1200	2045	2000	0
28	26	50000	1	3	2	23	0	0	0	0	0	0	47620	41810	36023	28967	29829	30046	1973	1426	1001	1432	1062	997	0
29	27	60000	1	1	2	27	1	-2	-1	-1	-1	-1	-109	-425	259	-57	127	-189	0	1000	0	500	0	1000	1
30	28	50000	2	3	2	30	0	0	0	0	0	0	22541	16138	17163	17878	18901	19617	1300	1300	1000	1500	1000	1012	0
31	29	50000	2	3	1	47	-1	-1	-1	-1	-1	-1	650	3415	3416	2040	30430	257	3415	3421	2044	30430	257	0	0
32	30	50000	1	1	2	26	0	0	0	0	0	0	15329	16575	17496	17907	18375	11400	1500	1500	1000	1000	1600	0	0
33	31	230000	2	1	2	27	-1	-1	-1	-1	-1	-1	16646	17265	13266	15339	14307	36923	17270	13281	15339	14307	37292	0	0
34	32	50000	1	2	2	33	2	0	0	0	0	0	30518	29618	22102	22734	23217	23680	1718	1500	1000	1000	1000	716	1
35	33	100000	1	1	2	32	0	0	0	0	0	0	93036	84071	82880	80958	78703	75599	3023	3511	3302	3204	3200	2504	0
36	34	500000	2	2	1	54	-2	-2	-2	-2	-2	-2	10929	4152	22722	7521	71439	8981	4152	22827	7521	71439	981	51582	0
37	35	500000	1	1	1	58	-2	-2	-2	-2	-2	-2	13709	5006	31130	3180	0	5293	5006	31178	3180	0	5293	768	0
38	36	160000	1	1	2	30	-1	-1	-2	-2	-2	-1	30265	-131	-527	-923	-1488	-1884	131	396	396	565	792	0	0
39	37	280000	1	2	1	40	0	0	0	0	0	0	186503	181328	180422	170410	173901	177413	8026	8060	6300	6400	6400	6737	0
40	38	60000	2	2	2	22	0	0	0	0	0	-1	15054	9806	11068	6026	-28335	18660	1500	1518	2043	0	47671	617	0
41	39	50000	1	1	2	25	1	-1	-1	-2	-2	-2	0	780	0	0	0	0	780	0	0	0	0	0	1
42	40	280000	1	1	2	31	-1	-1	2	-1	0	-1	498	9075	4641	9976	17976	9477	9075	0	9976	8000	9525	781	0
43	41	360000	1	1	2	33	0	0	0	0	0	0	218668	221296	206895	628699	195969	179224	10000	7000	6000	188840	28000	4000	0
44	42	70000	2	1	2	25	0	0	0	0	0	0	67521	66999	63949	63699	64718	69970	3000	4500	4042	2500	2800	2500	0

Figure 2.3: Sample data set.

Chapter 3

Methodologies

Machine learning is a vast field that intersects information technology, mathematics, statistics, probability, artificial intelligence, psychology, neurobiology, and numerous other disciplines. When presented with a sufficiently detailed dataset, AI tools excel at detecting patterns and trends. This capability empowers medical professionals to promptly and accurately reach a verdict on how to address a specific medical emergency. Neural networks, a key component of machine learning, demonstrate adaptability to changing input, enabling the generation of optimal results without the need for redesigning output criteria.[15] The application of machine learning methods for classification is driven by the intuition that machine learning can deliver intricate, non-linear models. Fundamentally, machine learning revolves around crafting algorithms that enable computers to learn extensively and adapt to evolving information. Learning is a dynamic process involving the discovery of statistical regularities and patterns within data. Certain algorithms also offer insights into the relative difficulty of learning in diverse environments [12]. Presently, a multitude of machine learning algorithms has been developed [9], continually refined and enhanced. Notably, the latest advancements in machine learning include the capacity to automatically apply intricate mathematical calculations to large datasets, resulting in significantly faster computation of results.

Adaptive programming, particularly prevalent in machine learning, allows applications to recognize patterns, learn from experience, abstract new information from data, and optimize the accuracy and efficiency of processing and output. To delve into this, let's begin by discussing sequence problems. The most straightforward machine learning problem involving a sequence is a one-to-one problem, illustrated in Figure 3.1. As we advance

through Chapter 2, we will persist in introducing machine learning methodologies applied to our medical datasets. This includes the materials employed for the optimization of predictive models. Furthermore, we will introduce the Graph Neural Networks model to augment and optimize results before delving into subsequent analyses.

3.1 Simple Neuron

Lets start with the introduction to a **neuron** with a single scalar input and no bias is shown below 3.1.

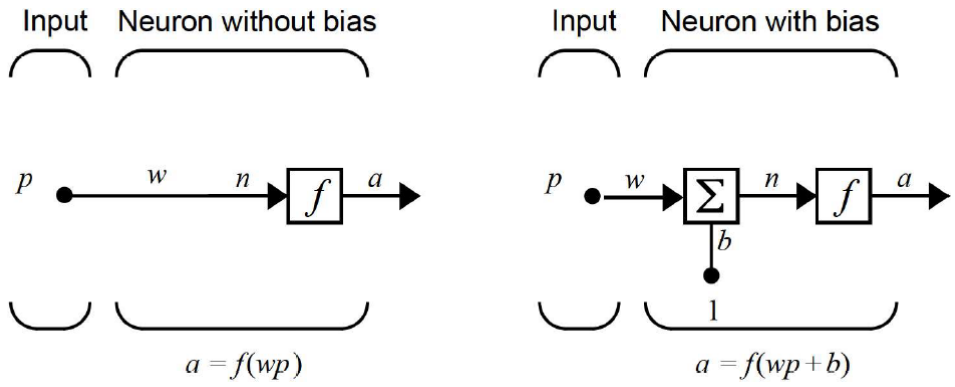

Figure 3.1: One to one system

The scalar input p undergoes transmission through a connection, where its strength is multiplied by the scalar weight w, resulting in the product $w \cdot p$, which is once again a scalar. In this context, the only parameter of the transfer function f is the weighted input $w \cdot p$, leading to the production of the scalar output a. The neuron on the right is equipped with a scalar bias, denoted as b. The bias can be conceptualized as either a simple addition to the product wp through the summing junction or as a shift of the function f to the left by an amount b. The bias functions similarly to a weight, with the distinction that it maintains a constant input of 1.

The **transfer function** net input n, again a scalar, is the sum of the weighted input wp and the bias b. This sum is the argument of the transfer function f. Three of the most commonly used functions are shown Fig 3.2 below. Sigmoid transfer function is commonly used in backpropagation networks, in part because it is differentiable [8].

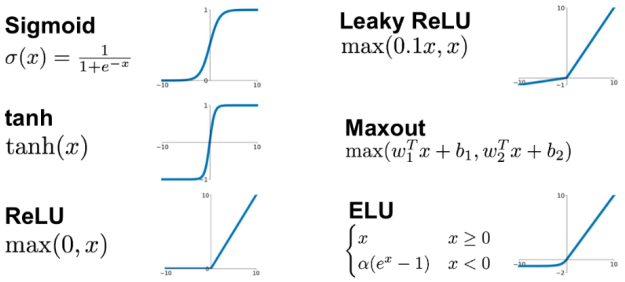

Figure 3.2: Commonly used transfer/activation functions.

3.1.1 Neuron With Vector Input

A neuron with a single R-element input vector is shown Fig 3.3 below. Here are the individual element inputs

$$p_1, p_2, ..., p_R,$$

are multiplied by the weights

$$w_{1,1}, w_{1,2}, ..., w_{1,R}$$

and the weighted values are feed to the summing junction. Their sum is simply $W \cdot p$, the dot product of the (single row) matrix W and the vector p.

The neuron has a bias b, which is summed with the weighted inputs to form the net input n. This sum, n, is the argument of the transfer function f.

$$n = w_{1,1}p_1 + w_{1,2}p_2 + ... + w_{1,R}p_R + b.$$

This expression can, of course, be written in as

$$n = W \cdot p + b$$

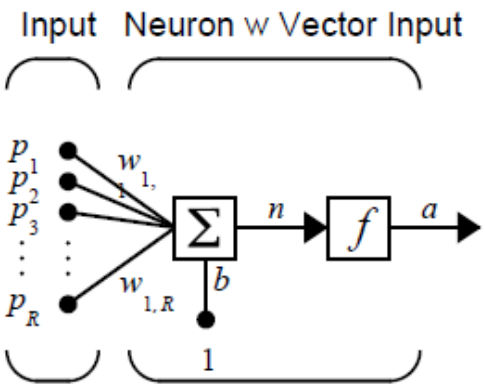

Figure 3.3: Neuron With Vector Input

3.1.2 A Layer of Neurons

Here fig 3.4 shows a one-layer network with R input elements and S neurons as follows. In

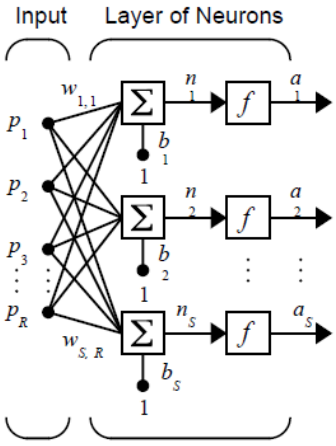

Figure 3.4: A Layer of Neurons

this network, each element of the input vector p is connected to each neuron input through the weight matrix W. The i^{th} neuron has a sum that gathers its weighted inputs and bias, to form its own scalar output $n(i)$. The various $n(i)$ taken together form an S-element net, input vector n. Finally, the neuron layer outputs form a column vector a.

The input vector elements enter the network through the weight matrix W.

$$W = \begin{bmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,R} \\ w_{2,1} & w_{2,2} & \dots & w_{2,R} \\ & & \dots & \\ w_{S,1} & w_{S,2} & \dots & w_{S,R} \end{bmatrix}$$

The row indices on the elements of matrix W indicate the destination neuron of the weight, and the column indices indicate which source is the input for that weight. Thus, the indices say that the strength of the signal from the second input element to the first (and only) neuron is. The S neuron, R input one-layer network also can be drawn in abbreviated notation. Here p is an R length input vector, W is an $S \times R$ matrix, with a and b are S length vectors. As defined previously, the neuron layer includes the weight matrix, the multiplication operations, the bias vector R, the sum and the transfer function boxes.[18]

$$a = f(Wp + b). \tag{3.1}$$

3.1.3 Multiple Layers of Neurons

Expanding into a larger network involves incorporating multiple layers, each consisting of a weight matrix W, a bias vector b, and an output vector a. To differentiate between these components for each layer, we adopt a layer notation. This layer notation is evident in the illustration of a three-layer network figure 3.5 and in the equations presented. In a multilayer network, the layers assume distinct roles, where a layer responsible for generating the network output is termed the output layer. Conversely, all other layers are designated as hidden layers. In the previously illustrated three-layer network, one layer functions as the output layer (layer 3), while two layers act as hidden layers (layer 1 and layer 2). Although some authors might consider the inputs as a fourth layer, we opt not to adopt this designation.

The same three-layer network discussed earlier can also be represented using our abbreviated notation.

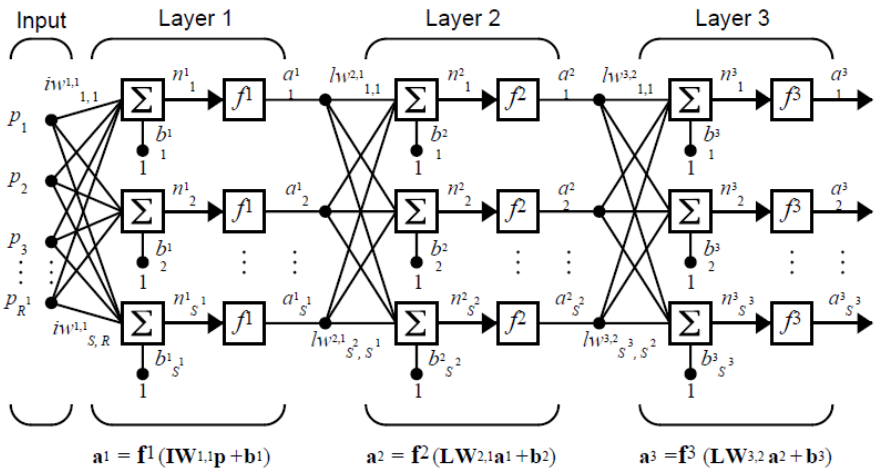

Figure 3.5: Multiple Layers of Neurons

We have the output of the neural network that can be calculated in the following way

$$y = a^3 = f^3(LW^{3,2} f^2(LW^{2,1} f^1(IW^{1,1} p + b^1) + b^2) + b^3), \tag{3.2}$$

where $IW^{1,1}$ is the weight matrix of the layer 1, $LW^{2,1}$ is the weight matrix of the layer 2, and $LW^{3,1}$ is the weight matrix of the layer 3. Biases of the layers are b^1, b^2, b^3. Multi-layer neural network are powerful models with non-convex objective functions. Although our convergence analysis does not apply to non-convex problems, we empirically found that Adam optimizer often outperforms other methods in such cases. Which later application will be shown along with a neural network model with two fully connected hidden layers (we can change number and analysis results) hidden units each and ReLU activation are used for this experiment with mini-batch.

Model Definition and Description- Neural Network

A neural network is a highly adaptable mathematical and statistical model employed across diverse tasks, encompassing both regression and classification. In regression scenarios, a single output unit, denoted as Yi, is employed when X = 1. For n-class classification tasks,

the top layer of the neural network consists of G units, with each unit (kth unit) modeling the probability of belonging to class K.

The intermediate layers of the neural network, positioned between the input and output layers, play a crucial role. These layers contain hidden units, and the derived features they calculate are known as Zm. These features are termed "hidden" because their values, denoted as Zm, are not directly observed during the training or inference process. The incorporation of hidden layers allows neural networks to capture complex relationships and patterns within the data, enabling them to learn intricate representations that contribute to improved performance in various tasks [18].

Additionally, it's worth noting that neural networks can be customized in terms of architecture, activation functions, and optimization strategies based on the specific requirements of the problem at hand. The adaptability of neural networks has contributed to their widespread use in fields such as image recognition, natural language processing, and many other domains where complex data relationships need to be learned and utilized for predictive or analytical purposes.

The derived features Z_m are created from linear combinations of the inputs, and then the target Y_k is modeled as a function of linear combinations of the Z_m as follows [18]:

$$Z_m = \sigma\left(\alpha_{0m} + \alpha_m^T X\right), m = 1, \ldots, M \tag{3.3}$$

$$T_k = \beta_{0k} + \beta_k^T Z, k = 1, \ldots, K \tag{3.4}$$

$$f_k(X) = g_k(T), k = 1, \ldots, K \tag{3.5}$$

here $Z = (Z_1, Z_2, \ldots, Z_M)$, and $T = (T_1, T_2, \ldots, T_K)$. The nonlinear function $\sigma()$ in (3.3) is called the activation function. In practice, different activation functions are used for different problems. The output function $g_k(T)$ allows a final transformation of the vector of outputs T. For regression, we used the identity function $g_k(T) = T_k$ and for K-class classification, the soft-max function defined in (3.6) is used. Figure 3.2 shows some of the most commonly used activation functions.

$$g_k(T) = \frac{e^{T_k}}{\sum_{\ell=1}^{K} e^{T_\ell}}. \tag{3.6}$$

Model Fitting - Neural Network

The neural network model comprises unknown parameters commonly referred to as weights. Our objective is to find optimal values for these weights that ensure the model fits the training data effectively. The entire set of weights is denoted by θ, encompassing[18]:

$$\begin{aligned} &\{\alpha_{0m}, \alpha_m; m = 1, 2, \ldots, M\}\, M(p+1) \text{ weights}, \\ &\{\beta_{0k}, \beta_k; k = 1, 2, \ldots, K\}\, K(M+1) \text{ weights}. \end{aligned} \tag{3.7}$$

For regression and classification, we use sum-of-squared errors and cross-entropy (deviance) as our measure of fit, respectively defined in (3.8) and (3.9):

$$R(\theta) = \sum_{k=1}^{K} \sum_{i=1}^{N} \left(y_{ik} - f_k\left(x_i\right)\right)^2 \tag{3.8}$$

$$R(\theta) = -\sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} \log f_k\left(x_i\right) \tag{3.9}$$

Typically, the solution is obtained by minimizing the function $R(\theta)$. However, this process may result in potential overfitting of the data. To mitigate this issue, regularization is introduced, either directly through a penalty term or indirectly via early stopping [18]. The conventional approach for minimizing $R(\theta)$ involves gradient descent, commonly referred to as back-propagation in this context, as discussed by Rumelhart et al. [33]. In our research, we introduce the concept of meta-learning and investigate its efficacy in minimizing functions. Specifically, we explore the capabilities of a meta-learning model in this context. In essence, a neural network is characterized by its architecture, activation functions, weights and biases, loss function, optimization algorithm, and tailored design choices adapted to the specific problem it aims to solve. The training process entails iteratively adjusting these parameters to enable the network to discern and internalize intricate patterns and relationships present within the given dataset. This adaptability and capacity for learning

make neural networks powerful tools across a spectrum of applications, from image and speech recognition to natural language processing. Figure 3.6 showcases a fully connected neural network.

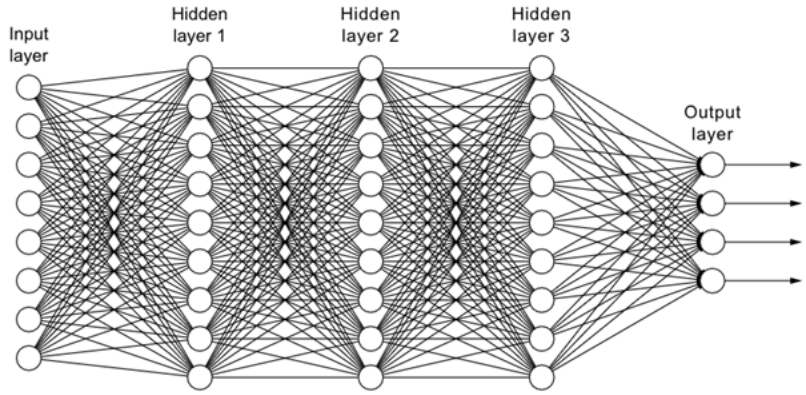

Figure 3.6: The Architecture of a Neural Network

3.1.4 Linear Regression

Linear regression is a fundamental and widely used technique in machine learning for predicting a continuous outcome variable based on one or more predictor variables. The goal of linear regression is to find the best-fitting linear relationship (a line) that minimizes the difference between the predicted and actual values. The supervised learning model used as a classification method. Start off with stating *training data:* $\{(x_1, g_1), (x_2, g_2), ..., (n_N, g_N)\}$, with the feature vector $X = (X_1, X_2, ..., X_p)$,where each variable X_j is quantitative. The response variable G is categorical. $G \in \mathcal{G} = \{1, 2, ..., K\}$, from a predictor $G(x)$ to predict G based on X. For a quick simple and most commonly used example we have email spam G has only two values, say 1 denoting a useful email and 2 denoting a junk email. X is a 57-dimensional vector, each element being the relative frequency of a word or a punctuation mark. $G(x)$ divides the input space (feature vector space) into a collection of regions, each labeled by one class.

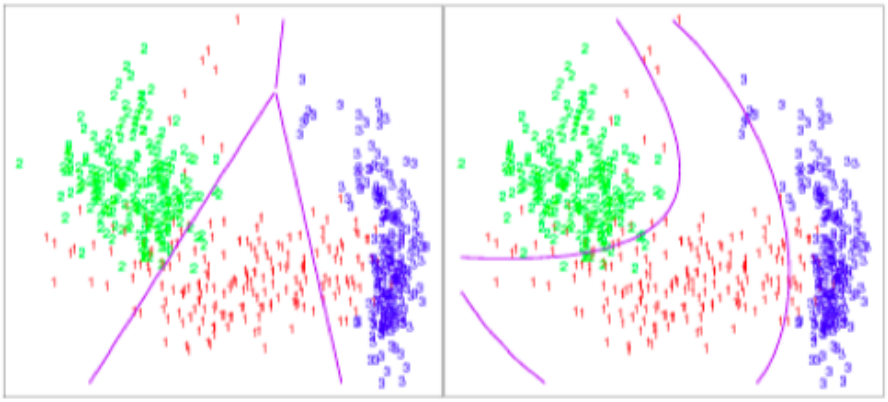

Figure 3.7: Linear Regression Classification problem.

On figure 3.7 the left plot shows some data from three classes, with linear decision boundaries found by linear discriminant analysis. The right plot shows quadratic decision boundaries. These boundaries were obtained by fining linear boundaries in the five-dimensional space $X_1, X_2, X_{12}, X_1^2, X_2^2$. Linear inequalities in this space are quadratic inequalities in the original space. For linear methods we have two class problem, the decision boundary between the two classes is a hyperplane in the feature vector space. A hyperplane in the p dimensional input space is the set:

$$\left\{ x : \alpha_0 + \sum_{j=1}^{p} \alpha_j x_j + 0 \right\}.$$

The two regions separated by a hyperplane:

$$\left\{ x : \alpha_0 + \sum_{j=1}^{p} \alpha_j x_j > 0 \right\},$$

and

$$\left\{ x : \alpha_0 + \sum_{j=1}^{p} \alpha_j x_j < 0 \right\}.$$

Linear regression finds application in diverse domains, such as predicting house prices, stock prices, and scenarios characterized by a linear relationship between input features

and the target variable. Despite its simplicity, this model is prized for its interpretability and straightforward implementation, rendering it a valuable and frequently employed tool within the machine learning toolkit.

3.1.5 Logistic Regression

Logistic regression, despite its name, is a statistical method predominantly utilized in machine learning for binary classification problems, where the outcome variable possesses two distinct classes. This method is well-suited for scenarios wherein the relationship between input features and the probability of belonging to a specific class is presumed to be approximately linear.

In logistic regression, the model assesses the impact of multiple independent variables presented concurrently to predict the membership of one of the two dependent variable categories. While the term "regression" in its name may be somewhat misleading, it stems from the fitting of a linear model to the feature space. This involves a more probabilistic perspective on classification, wherein the logistic regression model determines the likelihood of an instance belonging to a particular class.[18] Despite its nomenclature, logistic regression is fundamentally a classification technique, not a regression method, and its efficacy lies in its ability to model the probabilities associated with categorical outcomes. Quickly lets go over the different ways of expressing probability, lets consider a two-outcome probability space, where:

$$p(O_1) = p,$$

$$p(O_2) = 1 - p = q.$$

We can express probability of O_1 as:

Then we have the following functions from probability to log odds,

logit function:

$$z = log\left(\frac{p}{1-p}\right),$$

Table 3.1: Logistic regression

	notation	range equivalents
standard probability	p	0, 0.5, 1
odds	p/q	0, 1, $+\infty$
log odds (logit)	log(p/q)	$(-\infty, 0, +\infty)$

logistic function:

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

Using a logistic regression model we have a model which consists of a vector β in d-dimensional feature space. For point x in feature space, we have β to convert it into a real number z in the range $-\infty$ to $+\infty$ then,

$$z = \alpha + \beta \cdot x = \alpha + \beta_1 x_1 + ... + \beta_d x_d.$$

We map z to the range 0 to 1 using the logistic function $p = \frac{1}{(1+e^{-z})}$. Training a logistic regression model we take fourth the need to optimize β so the model gives the best possible reproduction of training set labels. This is usually done by numerical approximation of maximum likelihood, and on really large data set, we may use the gradient descent. As we can see the logistic regression can be considered a special case of linear regression models. A logistic regression model specifies that an appropriate function of the fitted probability of the event is a linear function of the observed values of the available explanatory variables. The Logistic regression model has one major advantage of being able to produce a simple probabilistic formula of classification, on the contrary a weakness is that it cannot properly deal with problems of non-linear and interactive effects of explanatory variables.

For instance, a sophisticated machine learning program could classify flowers based on photographs. Our aspiration is more modest, are going to classify an example of a Iris flowers based on the length and width measurements of their sepals and petals. The Iris genus entails about 300 species, but our program was able to simulate and classify the

following three:

- Iris Setosa
- Iris Virginica
- Iris Versicolor

This data set, iris_training.csv, is a plain text file that stores tabular data formatted as comma-separated values (CSV). Use the head -n5 command to take a peak at the first five entries:

```
120,4,setosa,versicolor,virginica
6.4,2.8,5.6,2.2,2
5.0,2.3,3.3,1.0,1
4.9,2.5,4.5,1.7,2
4.9,3.1,1.5,0.1,0
```

From this view of the data set, we notice the following:

1. The first line is a header containing information about the data set:
    - There are 120 total examples. Each example has four features and one of three possible label names.
2. Subsequent rows are data records, one example per-line, where:
    - The first four fields are features: these are characteristics of an example. Here, the fields hold float numbers representing flower measurements.
    - The last column is the label: this is the value we want to predict. For this data set we have an integer value of 0, 1, or 2 that corresponds to a flower name.

Each label is associated with string name (for example, "setosa"), but machine learning typically relies on numeric values (binary). Here we find ourselves with more than two variables to classify. The label numbers are mapped to a named representation, such as:

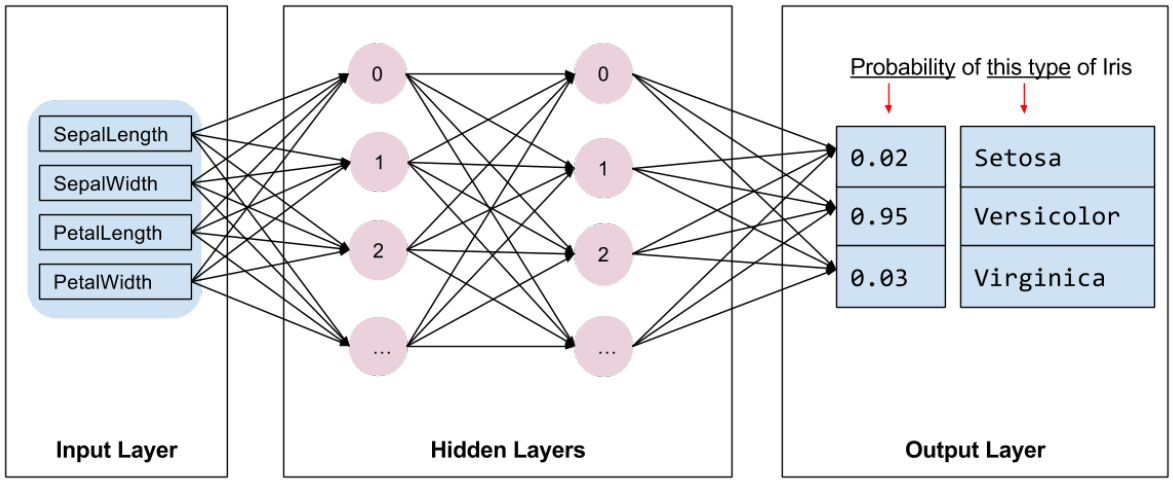

Figure 3.8: A neural network with features, hidden layers, and predictions.

- 0: Iris Setosa
- 1: Iris Versicolor
- 2: Iris Virginica

The selection of an appropriate algorithm for modeling and training is crucial in machine learning, given the multitude of algorithms designed to recognize patterns and make intelligent decisions based on input data. One of the primary challenges in machine learning lies in understanding and adapting to the behavior of inputs, which necessitates training the model with observed examples to ensure effectiveness and sensitivity in producing outputs. In the context of solving the Iris classification problem, a neural network proves to be a viable choice. Neural networks are adept at identifying intricate relationships between features and labels. They are characterized by a highly-structured graph organized into one or more hidden layers, each containing one or more neurons. This program specifically employs a dense network, also known as a fully-connected neural network, where neurons in one layer receive input connections from every neuron in the preceding layer.

Illustrated in Figure 3.8, the dense neural network implemented here comprises an input

layer, two hidden layers, and an output layer. This architecture allows the neural network to capture and process complex patterns in the Iris dataset, showcasing its capability to handle the classification task effectively.

Python code based on Tensor-flow may be found in *Appendix A*.

3.2 Convolutional Neural Networks (CNN)

The Convolutional Neural Network (CNN) is the preferred model for image classification, adopting a feed-forward architecture similar to traditional neural networks, encompassing input, hidden, and output layers. What sets CNNs apart are distinctive elements like pooling and fully connected layers. Notably, CNNs often include a greater number of hidden layers, showcasing their enhanced ability to extract features from input data. Through the simultaneous use of multiple convolutional filters, CNNs efficiently process and leverage correlations in multivariate time series data. This is achieved with fewer parameters, leading to accelerated learning and reduced data requirements. In contrast to fully connected networks, CNNs focus on small image patches, fostering a deeper understanding analogous to scrutinizing a book through a magnifying glass—one patch at a time. The foundational principle of convolutional neural networks revolves around local connectivity, where each node establishes a link to a specific region in the input, termed its receptive field. This is realized by replacing the weighted sums in traditional neural networks with convolutions.[26]

At each layer of a convolutional neural network, the input undergoes convolution with a weight matrix (filter) to produce a feature map. The weight matrix traverses the input, computing the dot product with each region. Unlike regular neural networks, all values in the resulting feature map share the same weights, indicating their collective detection of a specific pattern. This local connectivity and weight-sharing characteristic in CNNs reduce the count of trainable parameters, leading to more efficient training.

The core idea of a convolutional neural network is to learn a weight matrix in each layer

capable of extracting the necessary, translation-invariant features from the input. Overall, CNNs excel in image-related tasks by capturing hierarchical patterns through their unique architecture and parameter-sharing strategy.

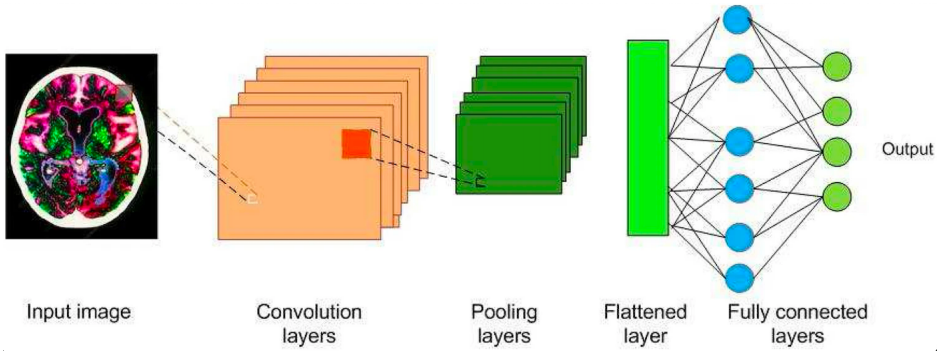

Figure 3.9: Typical CNN architecture.

CNNs, as a form of regularization, are modified versions of multi-layer perceptrons, with the term "multi-layer perceptrons" typically referring to fully connected networks where each neuron in one layer connects to all neurons in the next layer. The inherent "fully-connectedness" of these networks makes them susceptible to overfitting data, often mitigated by regularization techniques like incorporating weight magnitudes into the loss function.

In contrast, CNNs approach regularization differently by capitalizing on the hierarchical patterns in data, synthesizing more intricate patterns through smaller and simpler ones. Consequently, in terms of connectedness and complexity, CNNs lean towards the lower extreme. The inspiration for convolutional networks draws from biological processes, particularly the organization of the animal visual cortex. The connectivity pattern between neurons in CNNs mirrors the arrangement found in the visual cortex, where individual neurons respond to stimuli within a limited region known as the receptive field. These receptive fields overlap, collectively covering the entire visual field.

Lets consider a 256×256 image, CNN can efficiently scan it chunk by chunk, say a 5×5 window. The 5×5 window slides along the image (usually left to right, and

top to bottom), as shown below. Example, a stride length of 2 means the 5×5 sliding window moves by 2 pixels at a time until it spans the entire image. A convolution is a weighted sum of the pixel values of the image, as the window slides across the whole image. Turns out, this convolution process throughout an image with a weight matrix produces another image (of the same size, depending on the convention). Convolving is the process of applying a convolution, as the sliding-window methodology happens in the convolution layer of the neural network. A typical CNN has multiple convolution layers. Each convolutional layer typically generates many alternate convolutions, so the weight matrix is a tensor of $5 \times 5 \times n$, where n is the number of convolutions. For example, say we have an image which goes through a convolution layer on a weight matrix of $5 \times 5 \times 64$. It then generates 64- convolutions by sliding a 5×5 window. Therefore, this model has $5 \times 5 \times 64 (= 1,600)$ parameter, which is remarkably fewer parameters than a fully connected network, $256 \times 256 = 65,536$. The following is showcased on the following fig 3.10.

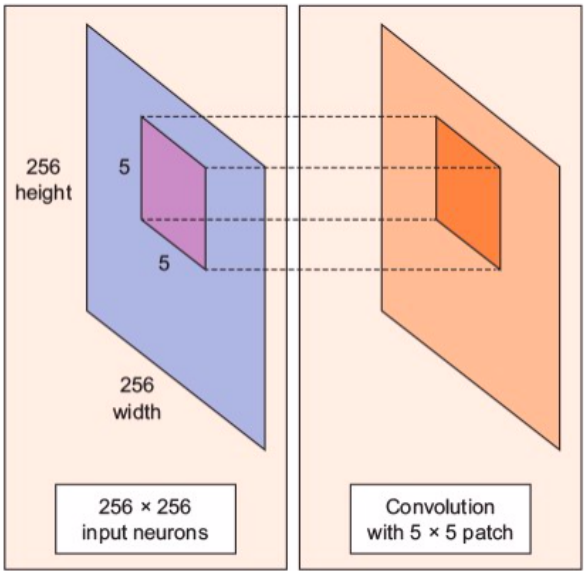

Figure 3.10: CNN classifier.

This leads us to the best part of the CNN, which is that the number of parameters is

independent of the size of the original image. You can run the same CNN on a 300×300 image, and the number of parameters will not change in the convolution layer. The CNN model will learn a function that maps a sequence of past observations as input to an output observation. As such, the sequence of observations must be transformed into multiple examples from which the model can learn. On a later chapter the application of the CNN will be displayed on our data.

In summary, CNNs are specialized neural networks designed for image-related tasks, featuring architectures that effectively capture hierarchical patterns and spatial relationships in visual data. Their success has extended to various fields beyond computer vision due to their ability to automatically learn relevant features from raw input.

3.2.1 Support Vector Machines

The support vector machine (SVM) extends the concept of the support vector classifier [18]. A SVM stands out as a potent supervised learning algorithm utilized for both classification and regression tasks. Its strength lies in its effectiveness in high-dimensional spaces, making it well-suited for scenarios where distinct margins of separation between classes exist.

The Lagrange dual function is expressed as [18]

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{i'=1}^{N} \alpha_i \alpha_{i'} y_i y_{i'} \left\langle h\left(x_i\right), h\left(x_{i'}\right) \right\rangle \tag{3.10}$$

here $h\left(x_i\right)$ is a transformed feature vector. The solution can be written as

$$f(x) = h(x)^T \beta + \beta_0$$

$$= \sum_{i=1}^{N} \alpha_i y_i \left\langle h(x), h\left(x_i\right) \right\rangle + \beta_0 \tag{3.11}$$

where α_i, β_0 can be determined by solving $y_i f\left(x_i\right) = 1$ in (13) for any x_i for which $0 < \alpha_i < C$ where C is the cost parameter which regulate the level of miss classifications

allowed. Both (3.10) and (3.11) involve $h(x)$ only through inner products, therefore the transformation $h(x)$ is not needed, only the knowledge of the kernel function [18]:

$$K\left(x, x'\right) = \langle h(x), h\left(x'\right)\rangle \tag{3.12}$$

that computes inner products in the transformed space.

SVMs are valued for their ability to handle complex decision boundaries, high-dimensional data, and their robustness in different domains. However, they may require careful tuning of parameters, and their performance can be impacted by large datasets. Overall, SVMs are a versatile and effective tool in machine learning for both classification and regression tasks.

3.2.2 k-Nearest Neighbors

The k-Nearest Neighbors (kNN) algorithm is a widely utilized learning method for supervised tasks, encompassing both classification and regression. Despite its simplicity and effectiveness, the primary challenge in practical applications lies in the sensitivity of kNN to hyper-parameter settings. These settings include the crucial choice of the number of nearest neighbors (k), the distance metric employed, and the weighting scheme applied during prediction. In essence, kNN, falling under the instance-based learning category, predicts outcomes by considering the majority class or average value of its k nearest neighbors within the feature space. Its straightforward yet powerful nature makes it a versatile tool in supervised machine learning, addressing a spectrum of tasks [9].

The procedure for kNN learning is as follows. Suppose a training dataset $\mathcal{D}(x_t, y_t)_{t=1}^{N}$ is given for a supervised learning task, where x_t and y_t are the input vector and the corresponding label vector of the $t - th$ instance. y_t is assumed to be a one-hot vecotr in the case of a classification task and a scalar value in the case of a regression task. In the training phase, the dataset $\mathcal{D}$ is just stored without any explicit learning from the dataset. In the inference phase, for each query instance x, kNN search is performed to retrieve kNN

instances $\mathcal{N}(x_t) = (x_t^{(i)}, y_t^{(i)})_{i=1}^{k}$ that are closest to x based on a distance function d. Then, the predicted label $\hat{y}$ is obtained as a weighted combination of the labels $y^{(1)}, ..., y^{(k)}$ based on a weighting function w along with the distance function d as follows:

$$\hat{y} = f(x; \mathcal{D}) = \frac{\sum_{i=1}^{k} w(d(x, x^{(i)}))\dot{y}^{(i)}}{\sum_{i=1}^{k} w(d(x, x^{(i)})).}$$

It computes the average or weighted average of the target values associated with the parameter K. The choice of the parameter k (number of neighbors) is crucial. A smaller k can lead to a more flexible model, but it may be sensitive to noise, while a larger k may provide a smoother decision boundary but might ignore local patterns. Additionally, the parameter p, which represents the power parameter employed in the Minkowski distance metric. The Minkowski distance or Minkowski metric is a metric in a normed vector space which can be considered as a generalization of both the Euclidean distance and the Manhattan distance. The Minkowski distance of order p (where p is an integer) between two points.

$$X = (x_1, x_2, ..., x_n) \; and \; Y = (y_1, y_2, ..., y_n) \in \Re^n \text{is defined as:}$$

$$D(X, Y) = (\sum_{i}^{n} |x_i - y_i|^p)^{1/p}.$$

Minkowski distance is typically used with p being 1 or 2, which correspond to the Manhattan distance and the Euclidean distance. This parameter influences the shape of the decision boundaries in the k-nearest neighbors (kNN) algorithm,[9] allowing for customization based on the specific characteristics of the data and problem at hand. First prediction is predicted with default parameters and this result is used for comparing. After that, best value of every parameters are found and are discussed their effects on result. Through GridSearch and Meta-learning algorithms are used to find best values of each parameters. So results can be compared each other in the conclusion part. To purpose of this kernel, understanding parameters of kNN Classifier algorithm and gain experience about hyper-

parameter tunings.

We introduce our approach as we have other models studied, the kNN learning approach, seamlessly integrating a graph neural network. This method adeptly learns a task-specific kNN rule from the training dataset using a graph neural network. The process involves constructing a kNN graph for each instance and its kNN counterparts, with nodes representing label information and edges conveying distance details. Subsequently, a graph neural network is deployed to leverage the kNN graph, predicting labels for each instance. The graph neural network essentially functions as a data-driven implementation of implicit weight and distance functions, significantly boosting kNN's prediction performance without the need for meticulous hyper-parameter tuning [42, 7].

Crucially, the proposed method extends its applicability to diverse supervised learning tasks, encompassing both classification and regression. Furthermore, it streamlines the prediction process for new data by eliminating the need for an additional optimization procedure, thereby enhancing computational efficiency. In summary, k-Nearest Neighbors stands out as a versatile and intuitive algorithm, well-adapted to specific data characteristics and applications. Its simplicity and lack of assumptions render it a valuable tool, particularly in scenarios where the inherent data structure is ambiguous or when interpretability holds paramount importance.

3.3 Graph Neural Networks

Graph Neural Networks (GNNs) represent a specialized class of neural networks meticulously crafted to operate on data structured as graphs. This architecture has garnered substantial attention and demonstrated its prowess across diverse domains, particularly excelling in tasks entailing relational data and structured information. Key applications of GNNs encompass node classification, link prediction, and graph classification, showcasing their versatility in handling interconnected data structures. A set of objects, and the con-

nections between them, are naturally expressed as a graph. The conceptual roots of Graph Neural Networks trace back to 2005, underscoring their enduring presence in the field of neural networks. However, it is in the last five years that they have truly gained prominence. This surge in popularity can be attributed to the growing recognition of the significance of leveraging graph structures for more effective machine learning solutions. Neural networks have been adeptly adapted to harness the inherent structure and properties embedded within graphs, facilitating more nuanced and powerful learning outcomes.[34]

In delving into the construction of a Graph Neural Network, several integral components come to the forefront. These include mechanisms for node classification, link prediction, and overall graph classification. The design choices behind these components are motivated by the inherent structure of graphs, aiming to unlock new optimization algorithms that can more effectively capture and leverage intricate relationships within the data.

As the adoption of Graph Neural Networks continues to expand, researchers and practitioners alike explore novel optimization algorithms, seeking to unravel the full potential of this architecture. The intersection of neural networks and graph theory offers a rich avenue for advancements in various fields, from social network analysis to molecular biology and recommendation systems. With ongoing research and development, the capabilities of GNNs are poised to evolve, unlocking new frontiers in the realm of machine learning and artificial intelligence. [23, 38]

Key Concepts of Graph Neural Networks:

- Graph Structure: GNNs are particularly adept at handling data that can be represented as a graph, where entities are nodes, and relationships between entities are edges. This structure allows GNNs to capture complex dependencies and patterns in relational data.

- Node Embeddings: GNNs learn embeddings for each node in the graph, representing the nodes in a continuous vector space. These embeddings encode both the node's own features and information from its neighboring nodes, enabling a holistic understanding

of the graph.

- Message Passing: GNNs employ a message-passing mechanism, where each node aggregates information from its neighbors and updates its own representation iteratively. This enables the model to capture global graph structure.
- Graph Convolutional Networks (GCNs): GCNs are a popular variant of GNNs that use a convolutional-like operation to aggregate information from neighboring nodes. They have been successful in tasks such as node classification and link prediction.
- Graph Attention Networks (GAT): GATs introduce attention mechanisms to assign different weights to neighboring nodes, allowing nodes to selectively attend to more relevant neighbors during message passing.
- Meta-Learning with GNNs: GNNs have been explored as meta-learning models, especially in few-shot learning scenarios. Meta-learning involves training models on a variety of tasks so that they can quickly adapt to new, unseen tasks with minimal examples.
-

Forecasting Information with GNNs:

- Temporal Graphs: GNNs can be extended to handle temporal graphs, where nodes and edges change over time. This makes them suitable for tasks such as temporal link prediction and forecasting evolving graph structures.
- Event Prediction: In social networks, GNNs can predict events such as friendships forming or breaking based on the evolving relationships within the graph.

Applications and Endeavors in Deep Learning:

- Drug Discovery: GNNs have shown promise in drug discovery by predicting molecular properties and understanding chemical relationships in molecular graphs.

- Recommendation Systems: GNNs can be used in recommendation systems to model user-item interactions and provide personalized recommendations based on the graph of user preferences.
- Social Network Analysis: GNNs are extensively used in social network analysis to predict user behavior, identify communities, and forecast connections between users.
- Traffic Forecasting: GNNs can be applied to traffic flow prediction in transportation networks by modeling the relationships between different locations.
- Meta-Learning:

  GNNs as meta-learning models have shown promising results in few-shot learning scenarios, where the model learns to quickly adapt to new tasks with limited examples.[35]

To start in simplest terms, a graph is a combination of nodes (or vertices) and edges. Let's establish what a graph is shown in fig. 3.11.

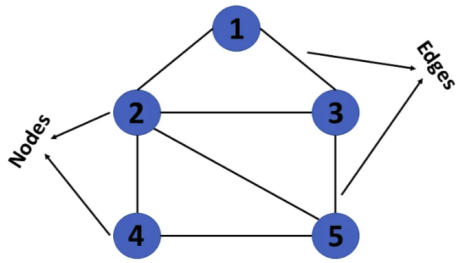

Figure 3.11: Basic Graph representation

Setup of graph we assume a graph

- $\mathbf{G}$, then define objects: *nodes*, *vertices* $\rightarrow \mathbf{N}$,
- interactions: *links* and *edges* $\rightarrow \mathbf{E}$
- system: *networks, graphs* $\rightarrow \mathbf{G}(\mathbf{N}, \mathbf{E})$.

- Assuming we have a graph **G:**
- V is the vertex set
- **A** is the adjacency matrix (assume binary)
- $X \in \mathbb{R}^{m \times |V|}$.

Graph neural network (GNN) is a deep model for graph representation of learning. One advantage of GNN is its ability to incorporate *node features* into the learning process such as social networks, biological networks and Gene expression profiles, gene functional information ,etc. One also has the abililty to include the non-features as indicator vectors (one-to-one encoding of a node(s)) and the constant vector of 1: [1,1,1,...]. Furthermore, lets explore more on why graphs? Graphs may be looked as a general language for describing and analyzing entities with relations/interactions. We most commonly have modern data bases such as images and text/speech shown in fig. 3.12.

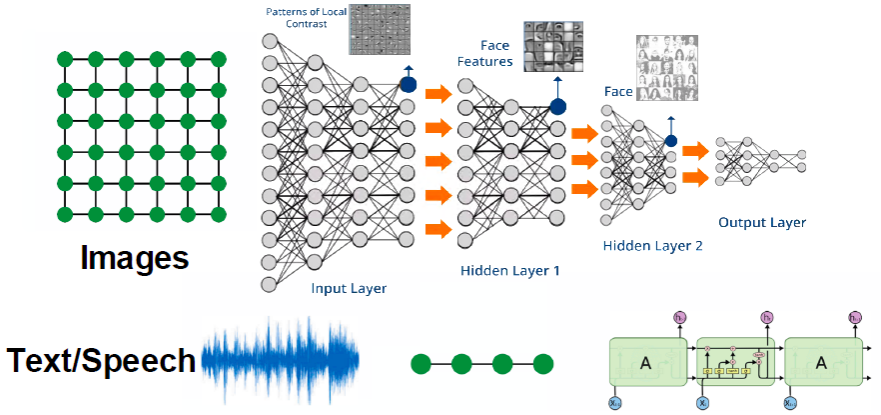

Figure 3.12: Deep learning toolbox with fixed sized grids and sequences as line graphs.

We can additionally specialize graphs by associating directionality to edges (directed, undirected). The *computational graph* is a **directed graph** that is used for expressing and evaluation mathematical expressions. Figure 3.13

An *undirected graph* is graph, i.e., a set of objects (called vertices or nodes) that are connected together, where all the edges are bidirectional, is sometimes called an undirected network. An undirected graph is when each node has a reciprocal connection. Say A is connected to B and B is connected to A, for a real world example of this is when you add a friend on Facebook. Each user now has full access to the other user's public content fig. 3.14.

One key feature of a graph network is in the Deep Graph Library (DGL), the dgl.graph function is used to create a graph object. Pre-training Graph Neural Networks (Hu et al., 2019)) This function takes various parameters to define the structure of the graph. Here are some key parameters commonly used in dgl.graph:

- num_nodes: Specifies the number of nodes in the graph.
- edges: Represents the edges of the graph. It can be a tuple of source and destination node IDs, or it can be a tensor with two columns representing the source and destination nodes.
- ntype and etype: Node type and edge type, respectively. These parameters are used when dealing with heterogeneous graphs, where nodes and edges can belong to different types.
- device: Specifies the device on which to create the graph (e.g., CPU or GPU).
- idtype: Specifies the data type of the node and edge IDs.
- readonly: If set to True, the resulting graph is read-only, meaning you cannot modify its structure.
- parent: If specified, this is another graph whose structural information is used as a scaffold to build the new graph.
- multigraph: If set to True, the graph allows multiple edges between the same pair of nodes.

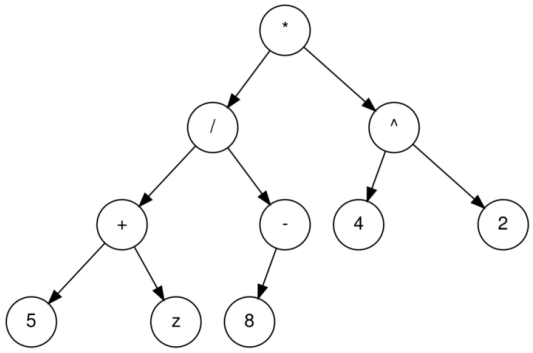

Figure 3.13: Graph which represents arbitrary computations.

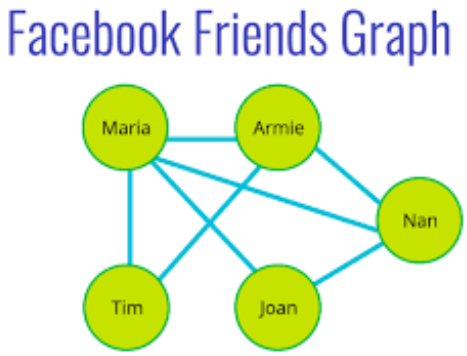

Figure 3.14: Facebook undirected graph

Node and Graph Embeddings: DGL allows for the learning of node and graph embeddings, which can capture important features of medical entities or entire medical records. These embeddings can be used as input for downstream tasks like diagnosis prediction. Our contribution to advancing medical diagnosis involves harnessing the capabilities of embeddings. We have introduced Meta-learning through a graph network and are actively researching ways to extend this approach to include Transfer Learning with Pre-trained Models. Transfer Learning in DGL facilitates transfer learning, allowing models pre-trained on large graph datasets to be fine-tuned on medical data. This is especially useful when labeled medical datasets are limited. It's important to note that while DGL provides a powerful framework, the success of applying it to medical diagnosis tasks also depends on the availability and quality of labeled medical data, ethical considerations, and collaboration with domain experts in healthcare.[22]

By amalgamating these attributes, graph neural networks emerge as a comprehensive and flexible framework for medical diagnosis, effectively navigating the intricate and interconnected landscape of healthcare data challenges. The machine learning and statistical methodologies mentioned earlier find practical application in our simulations. Following this, we embark on a succinct introduction to the optimization techniques seamlessly integrated into our machine learning models, aiming to elevate the precision of medical diagnoses.

3.4 Optimization Methods

Machine learning is an iterative endeavor wherein the acceleration of model training is vital to streamline the iterative cycle, considering the numerous parameters to fine-tune and mathematical techniques to enhance. Optimization stands at the core of machine learning, as the majority of machine learning challenges essentially boil down to optimization problems. The overarching aim in machine learning is to craft a model that excels in performance, delivering accurate predictions across a specific set of scenarios. Attaining this objective hinges on the application of optimization techniques in machine learning. These methods assume a pivotal role, facilitating models to learn from data and enhance their proficiency by pinpointing the optimal set of parameters or weights. This optimization process seeks to minimize a designated loss function, capturing the disparity between model predictions and actual outcomes. The process begins by formulating a model for the desired problem, selecting a suitable family of models, and preparing data amenable to modeling. Subsequently, the model is typically trained by addressing a core optimization problem that optimizes the variables, parameters, and features of the model in relation to the chosen loss function, possibly incorporating a regularization function. In the course of model selection and validation, the fundamental optimization problem is often iteratively solved to ascertain the most favorable results.

In summary, within the realm of machine learning, the effectiveness of an optimization algorithm is measured by certain desirable properties. These properties underscore the integral role of optimization methods in efficiently training machine learning models. The judicious selection of an appropriate optimization algorithm, coupled with meticulous tuning of hyper-parameters, profoundly influences a model's performance and its ability to generalize well to new data.

- good generalization,
- scalability to large problems

- good performance in practice in terms of execution times and memory requirements
- exploitation of problem structure
- fast convergence to an approximate solution of model, robustness and numerical stability for class of machine learning models attempted.

Therefore, optimization algorithms such as stochastic gradient descent, backpropagation (gradient descent), and the Adam optimizer, among others, play a crucial role, along with other features within. These methods are instrumental in enhancing the learning accuracy of our neural network. We will introduce our proposed optimization approach, incorporating a meta-learning model and integrating it with graph neural network.

3.4.1 Backpropagation (Gradient descent) and generalizations

Through supervised learning of an artificial neural networks using gradient descent. Its value of the neural network can be computed. (3.2). Given an artificial neural network and an error function, the method calculates the gradient of the error function with respect to the neural networks weights. Then we let x be a vector of input values and $\hat{y} = f(x, W, b)$ is a vector of the output values. For every initial values $x^{(1)}, x^{(2)}, ..., x^{(n)}$ it is possible to compare experimental output $y^{(1)}, y^{(2)}, ..., y^{(n)}$ and the values given by the neural network $\hat{y}^{(1)}, \hat{y}^{(2)}, ..., \hat{y}^{(n)}$. The goal of the learning process is minimization of the error.

$$E = \sum_{i=1}^{n} \|y^{(i)} - \hat{y}^{(i)}\|^2 = \sum_{i=1}^{n} \sum_{j=1}^{m} (y_j^{(i)} - \hat{y}_j^{(i)})^2 \tag{3.13}$$

The gradient descent is the simplest case of finding minimum error.

We use the basic formulation and structure of the gradient descent,

$$x_{i+1} = x_i - \gamma \nabla E(x_i),$$

where γ is some constant.

By iteratively traversing multiple passes and adjusting the weights in a direction that reduces the overall error between the target and predicted values, we aim to approach the local minimum on the gradient surface [18]. In the presented problem, the vector of parameters, denoted as x, encompasses the weight matrix and the bias matrix. To determine the minimum of the error E, it is essential to compute the derivatives with respect to the weight matrix W and the bias vector b.

The Partial derivatives of error with respect to weight matrix are as follows,

$$\frac{\partial E}{\partial w_{i,j}} = \sum_{i=1}^{n}\sum_{j}^{m}\frac{\partial}{\partial w_{i,j}}(y_j^{(i)} - \hat{y}_j^{(i)})^2 = \sum_{i=1}^{n}\sum_{j}^{m}2(y_j^{(i)} - \hat{y}_j^{(i)})\frac{\partial}{\partial w_{i,j}}(y_j^{(i)} - \hat{y}_j^{(i)}) =$$

$$\frac{\partial E}{\partial w_{i,j}} = -\sum_{i=1}^{n}\sum_{j}^{m}2(y_j^{(i)} - \hat{y}_j^{(i)})\frac{\partial \hat{y}_j^{(i)}}{\partial w_{i,j}}.$$

Partial derivatives of error with respect to bias,

$$\frac{\partial E}{\partial b_i} = \sum_{i=1}^{n}\sum_{j}^{m}\frac{\partial}{\partial b_i}(y_j^{(i)} - \hat{y}_j^{(i)})^2 = \sum_{i=1}^{n}\sum_{j}^{m}2(y_j^{(i)} - \hat{y}_j^{(i)})\frac{\partial}{\partial b_i}(y_j^{(i)} - \hat{y}_j^{(i)}) =$$

$$\frac{\partial E}{\partial b_i} = -\sum_{i=1}^{n}\sum_{j}^{m}2(y_j^{(i)} - \hat{y}_j^{(i)})\frac{\partial \hat{y}_j^{(i)}}{\partial b_i}.$$

Derivatives $\frac{\partial \hat{y}_j^{(i)}}{\partial w_{i,j}}$, $\frac{\partial \hat{y}_j^{(i)}}{\partial b_i}$ can be found by using the formulas below. The output values y can be calculated by using the formula (3.1)

$$p^{(k+1)} = f^{(k)}(W^{(k)}p^{(k)} + b^{(k)}),$$

where $p^{(k+1)}$ are values in the layer $k+1$, $W^{(k)}$ is the weight matrix of the layer k, $b^{(k)}$ is a bias of the layer k, $p^{(k)}$ is an input value of the layer k, and $f^{(k)}$ is an activation function of the layer k.

$$\frac{\partial p^{(k+1)}}{\partial w_{i,j}} = \frac{\partial}{\partial w_{i,j}}f^{(k)}(W^{(k)}p^{(k)} + b^{(k)}) =$$
$$= f'^{(k)}(n^{(k)})\left(\frac{\partial W^{(k)}}{\partial w_{i,j}}p^{(k)} + W^{(k)}\frac{\partial p^{(k)}}{\partial w_{i,j}}\right),$$

where $n^{(k)} = W^{(k)}p^{(k)} + b^{(k)}$.
If $w_{i,j}$ is in $W^{(k)}$ then $\dfrac{\partial p^{(k)}}{\partial w_{i,j}} = 0$ and

$$\frac{\partial p^{(k+1)}}{\partial w_{i,j}} = f'^{(k)}(n^{(k)})\frac{\partial W^{(k)}}{\partial w_{i,j}}p^{(k)}.$$

If $w_{i,j}$ is not in $W^{(k)}$ then $\dfrac{\partial W^{(k)}}{\partial w_{i,j}} = 0$ and

$$\frac{\partial p^{(k+1)}}{\partial w_{i,j}} = f'^{(k)}(n^{(k)})W^{(k)}\frac{\partial p^{(k)}}{\partial w_{i,j}}.$$

If $w_{i,j}$ is in $W^{(k-1)}$ then $\dfrac{\partial p^{(k-1)}}{\partial w_{i,j}} = 0$ and

$$\frac{\partial p^{(k)}}{\partial w_{i,j}} = f'^{(k-1)}(n^{(k-1)})\frac{\partial W^{(k-1)}}{\partial w_{i,j}}p^{(k-1)},$$

then

$$\frac{\partial p^{(k+1)}}{\partial w_{i,j}} = f'^{(k)}(n^{(k)})W^{(k)}\frac{\partial p^{(k)}}{\partial w_{i,j}} = f'^{(k)}(n^{(k)})W^{(k)}f'^{(k-1)}(n^{(k-1)})\frac{\partial W^{(k-1)}}{\partial w_{i,j}}p^{(k-1)}.$$

If $w_{i,j}$ is not in $W^{(k-1)}$, then $\dfrac{\partial W^{(k-1)}}{\partial w_{i,j}} = 0$,

$$\frac{\partial p^{(k)}}{\partial w_{i,j}} = f'^{(k-1)}(n^{(k-1)})W^{(k-1)}\frac{\partial p^{(k-1)}}{\partial w_{i,j}}.$$

Now it is necessary to find the derivative $\dfrac{\partial p^{(k-1)}}{\partial w_{i,j}}$, by using similar calculations.

Similar calculations can be applied to be bias.

$$\frac{\partial p^{(k+1)}}{\partial b_i} = \frac{\partial}{\partial b_i}f^{(k)}(W^{(k)}p^{(k)} + b^{(k)}) =$$

$$= f'^{(k)}(n^{(k)})\left(W^{(k)}\frac{\partial p^{(k)}}{\partial b_i} + \frac{\partial b^{(k)}}{\partial b_i}\right),$$

where $n^{(k)} = W^{(k)}p^{(k)} + b^{(k)}$.
If b_i is in $b^{(k)}$ then $\dfrac{\partial p^{(k)}}{\partial b_i} = 0$ and

$$\frac{\partial p^{(k+1)}}{\partial b_i} = f'^{(k)}(n^{(k)})\frac{\partial b^{(k)}}{\partial b_i}.$$

If b_i is not in $b^{(k)}$ then $\dfrac{\partial b^{(k)}}{\partial b_i} = 0$ and

$$\frac{\partial p^{(k+1)}}{\partial b_i} = f'^{(k)}(n^{(k)})W^{(k)}\frac{\partial p^{(k)}}{\partial b_i}.$$

If b_i is in $b^{(k-1)}$ then $\dfrac{\partial p^{(k-1)}}{\partial b_i} = 0$ and

$$\frac{\partial p^{(k)}}{\partial b_i} = f'^{(k-1)}(n^{(k-1)})\frac{\partial b^{(k-1)}}{\partial b_i},$$

$$\frac{\partial p^{(k+1)}}{\partial b_i} = f'^{(k)}(n^{(k)})W^{(k)}\frac{\partial p^{(k)}}{\partial b_i} = f'^{(k)}(n^{(k)})W^{(k)}f'^{(k-1)}(n^{(k-1)})\frac{\partial b^{(k-1)}}{\partial b_i}.$$

If $b_{i,}$ is not in $b^{(k-1)}$ then $\dfrac{\partial b^{(k-1)}}{\partial b_i} = 0$ and

$$\frac{\partial p^{(k)}}{\partial b_i} = f'^{(k-1)}(n^{(k-1)})W^{(k-1)}\frac{\partial p^{(k-1)}}{\partial b_i}.$$

Similarly, we can calculate the derivative of $\dfrac{\partial p^{(k-1)}}{\partial b_i}$.

3.4.2 Least square method

The least squares method is a foundational mathematical technique applied in regression analysis to determine the optimal-fitting curve or line through a set of points. It achieves this by minimizing the sum of the squares of the vertical deviations (residuals) between the observed and predicted values. Widely employed for estimating the parameters of linear models, the least squares method serves as a fundamental tool in statistical modeling and data analysis. Renowned for its simplicity and mathematical elegance, it provides a systematic approach to parameter estimation and model fitting in various fields.[18]

To apply the ordinary least squares (OLS) method, we apply the below formula to find the equation:

$$m = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2},$$

$$b = \bar{y} - m * \bar{x}.$$

Where x = independent variables and $\bar{x}$ = average of independent variables, with y= dependent variables, and $\bar{y}$ = average of dependent variables. Lets solidify the basic concepts in least squares regression, suppose we have some simple data set, $\{(x_i, y_i), i = 1, ..., n\}$, where x_i and y_i are real numbers. Say our model of y is related to x which is given by

$$y = f(x; w) + e,$$

$$f(x; w) = w^{'}\phi(x).$$

Where $\phi : \mathcal{R} \rightarrow \mathcal{R}^d$ is a specified function which maps to x to a d-dimensional ï¿œfeatureï¿œ vector, $\phi(x) = (\phi_1(x), ..., \phi_d(x))^{'}$; w is a d-dimensional parameter vector $w = (w_1, ..., w_d)^{'}$; e is the prediction error, which we do not model explicitly. We will use $w^{'}$ to denote the transpose of any vector w. In order to determine the least squared prediction error,

$$J(w) = \frac{1}{n}\sum_i (y_i - f(x_i; w))^2.$$

Solution to this problem is $\hat{w} = (X^{'}X)^{-1}X^{'}y$, where $X = (\phi(x_1), ..., \phi(x_n))^{'}$ is a $n \times d$ matrix whose first row is $\phi_1(x_1), ..., \phi_d(x_1)$ and the last row is given by $\phi_1(x_n, ..., \phi_d(x_n)$; The output vector y is defined as y is defined as $y = (y_1, y_2, ..., y_n)^{'}$. Assumption made on matrix $(X^{'}X)$ is invertible so that the problem is well-posed, (i.e. there exists a unique minimizer). This holds true for feature vectors $\phi(x_1), ..., \phi(x_n)$ associated with the training examples span the $d - dimensional$ feature space. As the feature vectors are long and the number of training points n is small. Now, for the estimate of $\hat{w}$ the resulting prediction errors $\hat{e}_i = y_i - f(x_i; \hat{w})$ should be "uncorrelated" with features:

$$\frac{1}{n}\sum_i \hat{e}_i \phi_k(x_i) = 0, k = 1, ..., d.$$

As these conditions are obtained by taking the derivative of $J(w)$ with respect to each w_i, $i = 1, .., d$, and setting them to zero. Make a note that the prediction error needs not

to be zero mean unless one of the features is a constant, i.e., say $\phi_1 = 1$ for all x, so that

$$\frac{1}{n}\sum_i \hat{e}\phi_1(x_i) = \frac{1}{n}\sum_i \hat{e}_i = 0.$$

3.4.3 Parameters and hyper-parameters of the model

In machine learning, achieving optimal accuracy and performance hinges on effectively managing both parameters and hyper-parameters. Parameters, the internal variables learned from training data, and hyper-parameters, external configuration settings set before training, collectively shape the model's capabilities. Consequently, the optimization of these elements is vital for superior model accuracy. The process of optimization encompasses a blend of manual tuning and automated methods, leveraging techniques such as grid search, random search, or sophisticated optimization libraries. A profound comprehension of the specific problem domain further enhances the effectiveness of hyper-parameter tuning, ensuring the model generalizes well to new and unseen data. In essence, the intricate interplay between parameter and hyper-parameter optimization is fundamental for maximizing the accuracy of machine learning models. In the process of fine-tuning our model, hyper-parameter optimization becomes imperative as we seek the optimal combination of values. This strategic optimization allows us to systematically reduce errors, constructing a model of unparalleled accuracy. Enhancing the model's performance necessitates the precise tuning of hyper-parameters.[18]

Following each iteration, a crucial step involves comparing the model's output with the expected results, evaluating accuracy, and, if required, refining the hyper-parameters. This iterative process can be executed manually or, alternatively, leveraging various optimization techniques, especially beneficial when dealing with substantial datasets. Let's delve into their definitions and roles:

- Parameters: Definition: Parameters are the internal variables that the model learns from the training data. They are the coefficients in linear regression, weights in neural

networks, or split points in decision trees.

  - Training: During the training phase, the model adjusts its parameters to minimize the difference between predicted and actual outcomes.
  - Impact on Accuracy: Properly tuned parameters are essential for achieving high accuracy. Optimal values ensure the model generalizes well to new, unseen data.

- Hyper-parameters: Definition: Hyper-parameters are external configuration settings that are not learned from the data but are set before the training process. Examples include the learning rate, the number of hidden layers in a neural network, or the depth of a decision tree.

  - Tuning: Hyper-parameters are set prior to training and need to be tuned to find the best configuration for a specific problem. This is often done through techniques like grid search or random search.
  - Impact on Accuracy: The choice of hyper-parameters significantly influences the model's performance. Proper tuning helps prevent overfitting or underfitting and maximizes accuracy on unseen data.

- Hyperparameter Tuning:

  - Grid Search: Involves defining a grid of hyper-parameter values and evaluating the model's performance for each combination.
  - Random Search: Randomly samples hyper-parameter combinations, offering a more efficient alternative to grid search.
  - Cross-Validation: Techniques like k-fold cross-validation help assess how well a model with a particular set of hyper-parameters generalizes to different subsets of the data.

- Impact on Model Selection: Model Selection: Different models may have different sets of hyperparameters. For instance, a random forest and a support vector machine

have distinct hyperparameters.

- Ensemble Models: In ensemble models like boosting or bagging, the choice of base models and their respective hyperparameters can significantly impact accuracy.
- Automated Hyper-parameter Tuning: Hyper-parameter Optimization: Techniques like Bayesian optimization, genetic algorithms, or specialized libraries (e.g., scikit-learn's GridSearchCV or RandomizedSearchCV) automate the hyperparameter tuning process.
- Efficiency: Automated methods are particularly useful when dealing with a large hyperparameter search space, improving efficiency in finding optimal configurations.[18]

3.4.4 GridSearchCV

One of the simplest algorithm that can be used as an example of meta-learning is GridSearchCV. GridSearchCV is the process of performing hyper-parameter tuning in order to determine the optimal values for a given model. As mentioned above, the performance of a model significantly depends on the value of hyper-parameters. Note that there is no way to know in advance the best values for hyper-parameters so ideally, we need to try all possible values to know the optimal values. Doing this manually could take a considerable amount of time and resources and thus we use GridSearchCV to automate the tuning of hyper-parameters [6]. GridSearchCV is a function that comes in Scikit-learns (or SK-learn) package. So an important point here to note is that we need to have the Scikit learn library installed on the computer. This function helps to loop through predefined hyper-parameters and fit your estimator (model) on your training set. So, in the end, we can select the best parameters from the listed hyper-parameters. In this case for given machine learning algorithm f it is possible to find a set of discrete values $\{x_0, x_1, ..., x_n\}$ (these are values of parameters of the algorithm f). Approximate value of the minimum/maximum of the function f can be find in the following way,

$$f_{max} \approx \underset{x \in \{x_0, x_1, ..., x_n\}}{\arg\max} f(x).$$

From mathematical point of view this method assumes constant approximation of function $f(x)$ in some neighborhood of points $x_0, x_1, ..., x_n$. Accuracy of this method depends of the accuracy of the approximation,

$$f_{max} \approx \underset{x \in \{x_0, x_1, ..., x_n\}}{\arg\max} f(x).$$

Where $N(x_i)$ is some neighborhood of the point x_i. It is possible to improve accuracy of presented method by improving accuracy of the approximation f_{approx,x_i}.

In summary, GridSearchCV is a powerful tool for automating the process of hyperparameter tuning, helping machine learning practitioners find the optimal configuration for their models.

3.4.5 Optimization of function by using subsequent approximations and related predictions

The concept of "Optimization of function by using subsequent approximations and related predictions" seems to describe a general approach to optimization problems, and several techniques in optimization and machine learning may align with this description. However, for a more precise and detailed understanding, it would be helpful to break down the elements of the phrase. Here's a general overview:

- Optimization of Function: Optimization involves the process of finding the best solution, often the maximum or minimum, of a given objective function. This function could represent, for example, a cost to be minimized or a utility to be maximized.

- Subsequent Approximations: The term "subsequent approximations" suggests an iterative approach to optimization. Instead of finding the optimal solution directly,

the method involves making successive approximations, refining the solution with each iteration.

- Related Predictions: The phrase "related predictions" could imply that the optimization process involves making predictions or estimations related to the function being optimized. This could be particularly relevant in machine learning contexts where models are trained to make predictions.

- Iterative Optimization: Many optimization algorithms, such as gradient descent, operate iteratively by making small adjustments to the solution in the direction that improves the objective function. This process is repeated until convergence is achieved.

- Machine Learning Context: In machine learning, optimization is a crucial step during the training of models. The objective is often to minimize a loss function, and iterative methods are commonly used to adjust the model's parameters for better predictions on the training data.

- Prediction and Modeling: The optimization process is often intertwined with the prediction aspect in machine learning models. As the model parameters are optimized, the model's predictions on new or unseen data are expected to improve.

- Example: For instance, in training a machine learning model, the iterative optimization process involves adjusting the model's weights or parameters based on the difference between the predicted outcomes and the actual outcomes. Subsequent iterations refine the model's predictions until convergence.

There are many optimization algorithms which use various approximations techniques in order to approximate minimum/maximum of function. In sequential linear programming objective function $f(x)$ is approximated by a linear approximation.

$$f_{approx,x_i}(x) = f(x_0) + f'(x_0)(x - x_0),$$

to find approximation of the maximum of function in some neighborhood $N(x_0)$ of given point x_0.

$$x_{max,N(x_0)} = \arg\max_{x \in N(x)} f(x) \approx \arg\max_{x \in N(x)} f_{approx,x_0}(x) \tag{3.14}$$
$$f_{max} \approx f(x_{max,N(x_0)}).$$

In the next iteration it is necessary to approximate function $f(x)$ in the point $x_{max,N(x_0)}$ assume that $x_0 = x_{max,N(x_0)}$ and repeat calculations.

In the sequential quadratic programming nonlinear function $f(x)$ is approximated by the quadratic Taylor's approximation,

$$f_{approx,x_0}(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2$$

and then then it is necessary to repeat the procedure [14].

In summary, the expression "Optimization of function by using subsequent approximations and related predictions" corresponds to iterative optimization strategies, notably in the realm of machine learning. This iterative process involves refining models through successive iterations to enhance their predictive capabilities. In a broader context, various approximation methods, such as neural networks, polynomials, and other machine learning algorithms, can be employed to approximate the minimum or maximum of general nonlinear functions.

3.4.6 Meta-Learning

Meta-learning, also referred to as "learning to learn," resides within the realm of statistical science and stands as a subset of the broader field of machine learning. While conventional machine learning models typically necessitate extensive training with a substantial dataset, humans exhibit a remarkable capacity too rapidly and efficiently acquire new knowledge and skills.[31] Meta-learning aids researchers in discerning which algorithms yield superior predictions or insights from databases. It is used to improve the results and performance

of the learning algorithm by changing some aspects of the learning algorithm based on the results of the experiment. Meta-learning algorithms utilize metadata from learning algorithms as their input, subsequently generating predictions and offering insights into the performance of these learning algorithms. For instance, in a learning model involving images, metadata may encompass details like size, resolution, style, creation date, and owner. Each learning algorithm operates on a set of assumptions about the data, known as its inductive bias, or sometimes referred to as the learning bias of the algorithm. Meta-Learning leverages metadata such as algorithm properties (performance metrics and accuracy) or previously derived patterns from the data. This is employed to learn from, select, modify, or combine various learning algorithms effectively for a given learning task. The fundamental goal of meta-learning is to create our own machine learning model with the ability to rapidly grasp new concepts and skills with minimal training examples. The most critical challenge in meta-learning lies in the systematic design of experiments.[31]

Formalizing Meta- Learning model

In conventional machine learning approaches, the effectiveness of the model depends on manually designed feature extraction. In contrast, deep learning combines both feature and model learning, resulting in notable performance improvements. Meta-learning within neural networks takes this a step further by integrating the learning of collective features, models, and algorithms.

In traditional supervised machine learning, we are provided with a training dataset $\mathcal{D} = \{(x_1, y_1), ..., (x_N, y_N)\}$, consisting of pairs (input image, output label). We can then train a predictive model $\hat{y} = f(\theta(x))$ with parameters θ, by solving for the optimal parameter values: $\theta^* = argmin_\theta \mathcal{L}(\mathcal{D}; \theta, \omega)$.

Meta learning algorithm objective function can be mathematically expressed as:

$$\omega^* = argmin_\omega \sum_{i=1}^{M} \mathcal{L}^{meta}(\theta^{*^i}(w), w, \mathcal{D}_{source}^{val^{(i)}})$$

$$s.t.\ \theta^{*^i}(\omega) = argmin_\theta\ \mathcal{L}^{task}(\theta, \omega, \mathcal{D}^{trian^{(i)}}_{source})$$

Where $\mathcal{L}$: a function that measures the match between true labels and those predicted by $f(\theta)$.

θ : parameter for inner algorithm

$\mathcal{D}$: the Data-set under consideration.

ω: meta-knowledge to denote dependence on θ and class of labels.

A robust meta-learning model should undergo training across a range of learning tasks and be optimized for optimal performance across a distribution of tasks, potentially encompassing unseen tasks as well.[29] We introduced our meta-learning model to a range of different learning tasks. The optimal value (max), in our case, is medical diagnosis; summary of our model Iterations: Below is the algorithm for our Meta-learning model 1.

Algorithm 1 Meta-Learning: Model-Agnostic Meta-Learning (MAML)

Require: $p(\tau)$: distribution over tasks

Require: α, β: step size hyper-parameters

1. Randomly initialize θ
2. **while** not done **do**
3. Sample batch of tasks $\tau_i\ \sim p(\tau)$
4. **for** all (τ_i) **do**
5. **Evaluate** $\nabla_\theta \mathcal{L}_{\tau_i}(f_\theta)$ with respect to K examples
6. Compute adapted parameters with gradient descent : $\theta'_i = \theta - \alpha \nabla_\theta \mathcal{L}_{\tau_i}(f_\theta)$
7. **end for**

/// Note: the meta-update is done using the test sets

8. Update $\theta \leftarrow \theta - \beta \nabla_\theta \Sigma_{\tau_i\ p(T)} \mathcal{L}_{\tau_i}(f_{\theta'_i})$
9. **end while**

Our Graph Meta- learning Algorithm

Leveraging insights from hyper-parameter tuning and inspired by the Model-Agnostic Meta-Learning (MAML) model, we embarked on the development of a bespoke meta-learning algorithm that seamlessly integrates a graph neural network. Building upon the previously mentioned dgl.graph features, our approach harnesses the power of graph structures to enhance the learning process. In the realm of hyper-parameter tuning, we meticulously fine-tuned model parameters to optimize performance. This iterative process allowed us to identify configurations that significantly contribute to the effectiveness of our meta-learning algorithm.

The incorporation of the MAML model further augmented our meta-learning framework. MAML, known for its ability to quickly adapt to new tasks with limited data, provided a solid foundation for our algorithm. We adapted its principles to accommodate the unique characteristics of graph-structured data, allowing our model to capture intricate relationships and dependencies present in medical datasets.

Central to our innovation is the utilization of the dgl.graph feature from the Deep Graph Library (DGL). This feature enabled us to represent and process medical data in a graph structure efficiently. The graph neural network within our meta-learning algorithm leverages this representation, capturing complex interactions between various medical entities. This inclusion of graph structures contributes to the algorithm's adaptability and effectiveness in handling diverse medical datasets to higher performance to unseen data.

Graph Meta-learning (our algorithm)

1. Iteration 1

   Use base-algorithm (Logistic regression, kNN, SVM), to generate initial training set $D_{m,1}$ for meta-learning algorithm

$$D_{m,1} = \{(x_{tr,1}, y_1), (x_{tr,1}, y_1), ..., (x_{tr,n_{1,tr}}, y_{n_{1,tr}})\}$$

   where $y_1 = BaseAlgorithm(x_{tr,1}, D)$,

$y_{n_{1,tr}} = BaseAlgorithm(x_{tr,n_{1,tr}}, D)$,

$n_{1,tr}$ is the number of data points (simulations),

D is the data set of the base-model,

$x_{tr,0}, x_{tr,1}, ..., x_{tr,n_{1,tr}}$ are parameters of the base model,

and $y_0, y_1, ..., y_{n_{1,tr}}$ are appropriate medical diagnosis.

Find $y_{max,1} = \max\{y_0, y_1, ..., y_{n_{1,tr}}\}$.

2. Train meta-learning algorithm (neural network, polynomial regression, graph neural networks, etc.) on the meta-learning training set $D_{m,1}$

$$W_1^* = \arg\max_W Loss(MetaAlgorithm, D_{m_1}, W)$$

where W is a matrix of parameters of the meta-learning algorithm, W_1^* is a matrix of parameters of meta-learning algorithm after training, $Loss$ is some loss function (cross-entropy loss, MSE, Huber loss, L_1 loss, etc.)

3. Find the set of parameters which corresponds to maximum predicted value.

$$x_{max,pr,1} = \arg\max_{x_{pr} \in X_{pr,1}} MetaAlgorithm(x_{pr}, W_1^*)$$

where $X_{pr,1}$ is a set of parameter used in prediction. In this step it is possible to apply many existing optimization algorithms (gradient descent, search, etc.).

4. Iteration 2

Generate new training set $D_{m,2}$ around the point $x_{max,pr,1}$.

$$D_{m,2} = \{(x_{tr,0}, y_0), (x_{tr,1}, y_1), ..., (x_{tr,n_{2,tr}}, y_{n_{2,tr}})\}$$

Update $y_{max,2}$ based on the new data.

5. Re-train meta-learning algorithm.

$$W_2^* = \arg\max_W Loss(MetaAlgorithm, D_{m_1}, D_{m_2}, W)$$

6. Find prediction of the new optimal value.

$$x_{max,pr,2} = \underset{x_{pr} \in X_{pr,2}}{\arg\max}\, MetaAlgorithm(x_{pr}, W_2^*)$$

7. Iteration 3

   Generate new training set $D_{m,3}$ around the point $x_{max,pr,2}$.

$$D_{m,3} = \{(x_{tr,0}, y_0), (x_{tr,1}, y_1), ..., (x_{tr,n_{3,tr}}, y_{n_{3,tr}})\}$$

   Update $y_{max,3}$ based on the new data.

8. Re-train meta-learning algorithm.

$$W_3^* = \underset{W}{\arg\max}\, Loss(MetaAlgorithm, D_{m_1}, D_{m_2}, D_{m_3}, W)$$

9. Find prediction of the new optimal value.

$$x_{max,pr,3} = \underset{x_{pr} \in X_{pr,3}}{\arg\max}\, MetaAlgorithm(x_{pr}, W_3^*)$$

10. Continue calculations until convergence.

$$|y_{max,i} - y_{max,i+1}| < \varepsilon$$

11. The result of the optimization process is a pair (x_{max}, y_{max}).

$$y_{max} = BaseAlgorithm(x_{max})$$

    y_{max} is maximum value generated by the base algorithm,

    x_{max} is a set of optimal parameters of the base algorithm which generate the maximum value y_{max}.

Our meta-learning algorithm, enriched by hyper-parameter tuning, MAML principles, and the expressive power of graph neural networks through dgl.graph, stands as a testament

to the synergy achieved by merging advanced machine learning techniques. This holistic approach not only optimizes the learning process but also positions our algorithm as a promising candidate for addressing the challenges posed by medical diagnosis tasks, where nuanced relationships and dependencies play a pivotal role in accurate predictions.

3.4.7 Universal approximation theorem

The Universal Approximation Theorem is a significant result in the field of artificial neural networks, particularly in the context of neural network theory. It was first formulated by George Cybenko in 1989 and later independently proven and extended by Kurt Hornik in 1991. The theorem demonstrates the universal approximation capabilities of feedforward neural networks with a single hidden layer.[20]

Here are key points about the Universal Approximation Theorem:

- Approximation Power: The theorem states that a feedforward neural network with a single hidden layer containing a sufficient number of neurons (or units) can approximate any continuous function on a compact subset of R^n, given appropriate activation functions and parameter settings.

- Single Hidden Layer: The remarkable aspect of the theorem is that it specifically applies to networks with just one hidden layer. The hidden layer is where the network learns to map input data to a higher-dimensional feature space.

- Activation Functions: The Universal Approximation Theorem doesn't prescribe a specific activation function, but it assumes the use of activation functions that are non-constant, bounded, and continuous. Commonly used activation functions like sigmoid and rectified linear unit (ReLU) satisfy these conditions.

- Sufficiency of Neurons: The theorem doesn't provide an exact formula for the number of neurons needed in the hidden layer but establishes that, theoretically, there exists a sufficient number of neurons to approximate any continuous function.

- Compact Subset: The approximation holds on a compact subset of $\Re^n$. A compact set is a closed and bounded subset of Euclidean space.

- Practical Implications: While the Universal Approximation Theorem demonstrates the theoretical capability of neural networks, it doesn't provide guidance on practical network architecture, training, or generalization to unseen data. In practice, the choice of architecture, regularization techniques, and training strategies plays a crucial role.

- Extensions and Variations: The theorem has been extended to include other architectures and variations, such as networks with multiple hidden layers (deep networks) and different types of activation functions.

- Caveats: While the theorem underscores the universality of neural networks in approximating functions, it doesn't guarantee efficiency or ease of training. In practice, network training can be challenging, especially for deep architectures.

The Universal Approximation Theorem states that neural networks have the capability to serve as a universal approximation model for a wide range of functions. This means that given a large enough neural network with appropriate parameters, it is possible to approximate a diverse set of relationships between variables, which is crucial in various optimization tasks.

Theorem 1. *Let $C(X, \mathbb{R}^m)$ denote the set of continuous functions from a subset X of a Euclidean $\mathbb{R}^n$ space to a Euclidean space $\mathbb{R}^m$.*

Let $\sigma \in C(\mathbb{R}, \mathbb{R}.)$

Note that $(\sigma \circ x)_i = \sigma(x_i)$, so $\sigma \circ x$ denotes σ applied to each component of x.

Then σ is not polynomial if and only if for every $n \in \mathbb{N}$, $m \in \mathbb{N}$, compact subspace $K \subseteq \mathbb{R}^n$, $f \in C(K, \mathbb{R}^m), \varepsilon > 0$ $\exists k \in \mathbb{N}$, $A \in \mathbb{R}^{k \times n}$, $b \in \mathbb{R}^k$, $C \in \mathbb{R}^{m \times k}$ such that $\sup_{x \in K} \|f(x) - g(x)\| < \varepsilon$. Where $g(x) = C \cdot (\sigma \circ (A \cdot x + b))$.

Furthermore, non-continuous activation functions can serve to approximate a sigmoid function, enabling the extension of the aforementioned theorem to encompass these functions. For instance, the step function is a viable choice. This revelation highlights that a perceptron network with a solitary infinitely wide hidden layer possesses the capability to approximate arbitrary functions. The same construction employed for the first layer can be applied to approximate the identity function in subsequent layers, allowing for the approximation of a broader class of functions with a network of greater depth.[20]

In summary, the Universal Approximation Theorem is a fundamental result highlighting the expressive power of neural networks with a single hidden layer. It underscores the universality of these networks in representing a wide range of functions, laying the theoretical foundation for the widespread use of neural networks in various applications.[20]

3.4.8 Epochs

In the realm of artificial neural networks, an epoch signifies a complete cycle through the entire training dataset, marking a pivotal and transformative period in the model's learning history. It represents a point in time that encapsulates significant developments and notable changes. Training a neural network typically extends beyond a few epochs, involving the repeated exposure of the network to the training data in varied patterns. In essence, the number of epochs is a hyper-parameter crucially defining how many times the learning algorithm iterates through the entire training dataset. Each epoch provides an opportunity for the neural network to update its internal model parameters, contributing to improved generalization when faced with new, unseen input (test data). While it might seem counterintuitive initially, the significance lies in the iterative exposure of the entire dataset, enhancing the network's ability to discern patterns and nuances. Here are key points about epochs:

- Definition: An epoch represents one complete cycle through the entire training dataset. During each epoch, the model updates its parameters based on the information in

the training data.

- Training Process: In machine learning, especially in deep learning tasks like training neural networks, learning occurs through an iterative process. The model processes batches of training data, makes predictions, computes the error, and updates its parameters to minimize this error.

- Multiple Epochs: Training a model typically requires multiple epochs. The number of epochs is a hyper-parameter that the user must specify before training begins. It determines how many times the model will work through the entire training dataset.

- Underfitting and Overfitting: Underfitting may occur if the model is too simple and cannot capture the underlying patterns in the data even after multiple epochs. On the other hand, overfitting may occur if the model becomes too specialized to the training data, performing poorly on new, unseen data.

- Validation Data: To monitor the model's performance during training and prevent overfitting, a portion of the dataset is often set aside as validation data. The model's performance on the validation set is evaluated after each epoch.

- Early Stopping: Early stopping is a technique where training is halted if the model's performance on the validation set stops improving. This helps prevent overfitting and can be based on criteria such as the accuracy or loss on the validation data.

- Batch Size: The concept of epochs is closely related to the batch size, which represents the number of training examples utilized in one iteration. The number of batches in one epoch is determined by the total number of training examples divided by the batch size.

In summary, epochs are a critical aspect of the training process in machine learning, providing a measure of how many times a model has seen the entire training dataset.

Properly selecting the number of epochs is essential to achieve effective model training without underfitting or overfitting.

3.5 Activation functions with Neural Networks

Activation functions stand as vital components within artificial neural networks, acting as mathematical operations applied to the output of each neuron or node. They play a pivotal role in introducing non-linearity to the network, a key factor enabling the model to grasp intricate patterns and relationships in data. Now, let's swiftly delve into the significance of activation functions in machine learning and neural networks. Serving as crucial elements, these functions contribute to the non-linear transformation applied to the input of neurons or nodes. By introducing non-linearity, they empower the network to effectively learn from and model complex patterns within the data. The following outlines key facets of activation functions in the context of neural networks.

- Linearity vs. Non-Linearity: Activation functions introduce non-linearity to the network. Without non-linear activation functions, a neural network would essentially reduce to a linear model, and the ability to model complex relationships in data would be severely limited.
- Common Activation Functions: Fig. 3.2
  - Sigmoid (Logistic): It squashes input values between 0 and 1, making it useful in binary classification problems. However, it has drawbacks like vanishing gradients.
  - Hyperbolic Tangent (tanh): Similar to the sigmoid but ranges from -1 to 1, addressing the vanishing gradients issue to some extent.
  - Rectified Linear Unit (ReLU): It outputs the input for positive values and zero for negative values, offering simplicity and often faster convergence. However, it suffers from the "dying ReLU" problem where neurons can become inactive.

  - Leaky ReLU: It addresses the dying ReLU problem by allowing a small negative slope for negative input values.
  - Parametric ReLU (PReLU): Similar to Leaky ReLU but with the negative slope as a learnable parameter.
  - Exponential Linear Unit (ELU): A variant of ReLU that smoothens the transition for negative input values.
  - Scaled Exponential Linear Unit (SELU): A self-normalizing activation function that can improve convergence in deep networks.

- Choice of Activation Function: The choice of activation function depends on the nature of the problem, network architecture, and potential challenges like vanishing gradients or dead neurons. ReLU and its variants are widely used due to their simplicity and effectiveness.

- Output Layer Activation: The choice of activation function in the output layer depends on the nature of the task:
  - Sigmoid: Used for binary classification problems, where the network outputs probabilities.
  - Softmax: Used for multi-class classification problems, converting network outputs into probability distributions.

- Vanishing and Exploding Gradients: Some activation functions, like sigmoid and tanh, can suffer from vanishing gradients, making it challenging for deep networks to learn. Others, like ReLU, may lead to exploding gradients. Techniques like weight initialization and batch normalization can help mitigate these issues.

- Adaptive Activation Functions: Some research explores adaptive activation functions that can learn the activation behavior during training, adapting to the specific characteristics of the data.

In essence, selecting the appropriate activation function for neural networks is a pivotal design choice, influencing the network's learning capacity, convergence speed, and its capability to model intricate relationships within data. Ongoing research endeavors aim to introduce novel activation functions and refine existing ones to enhance the overall performance and stability of neural networks.

3.6 Data Splitting

Classification stands as a cornerstone task in machine learning, aiming to forecast the categorical class or label of an input by leveraging its distinctive features. This essential function finds extensive application across diverse domains, including image recognition, spam detection, medical diagnosis, and sentiment analysis. Machine learning, a pivotal methodology within this realm, constitutes a primary avenue for categorizing the multifaceted sciences. It delves into the mechanisms of autonomously acquiring the ability to make precise predictions based on historical observations. In the realm of artificial neural networks, recognizing the inevitability of incomplete raw data is crucial. Given that these networks typically demand a comprehensive dataset for accurate classification, addressing the issue of incomplete data becomes imperative. Through imputation and data completion strategies, we pave the way for the effective classification of data, ensuring the artificial neural network's capability to learn and predict remains robust. The simpler model, while carrying more bias, exhibits less sensitivity to the specific instances encountered during training, mitigating the risk of overfitting. In the context of training, especially in artificial neural networks, suboptimal generalization is often manifested as over-training. To circumvent this, a prevalent technique is to employ hold-out cross-validation, effectively preventing over-training and enhancing the model's ability to generalize to unseen data. In most of the applications, simple random sampling is used. Nevertheless, there are several sophisticated statistical sampling methods suitable for various types of data sets. In the case of supervised learning, a computational model is trained to predict outputs of an un-

known target function. The target function is represented by a finite training data set T of examples of inputs and the corresponding desired outputs: $T = [\ x1,\ d1], ..., [\ xn,\ dn]$, where $n > 0$ is the number of ordered pairs of input/output samples (patterns). At the end of the training process, the final model should predict correct outputs for the input samples from T, but it should also be able to generalize well to previously unseen data. Cross-validation techniques [4] belong to conventional approaches used to ensure good generalization and to avoid over-training. K-fold cross-validation maximizes the use of the data. The basic idea is to divide the data set T into two subsets ï¿œ one subset is used for training while the other subset is left out and the performance of the final model is evaluated on it. K-fold divides data randomly into k folds (subsets) of equal size, then we train the model $k-1$ folds (i.e. use one fold for testing). Repeat this process k times so that all folds are used for testing, which then we compute the average performance on the k test sets. This effectively uses all the data for both training and testing, typically $k = 10$ is used.

Some of the methods are simple and widely used, although they suffer from high variance of the model performance. For classification problems, we we measure the performance of a model in terms of its error rate: percentage of incorrectly classified instances in the data set. Building a model, we want to use it to classify new data. Hence we are chiefly interested in model performance on new (unseen) data. The re-substitution error (error rate on the training set) is a bad predictor of performance on new data. The model was build to account for the training data, so might over-fit it, i.e., not generalize to unseen data. Moveover, the more data available, more training which equals better model. The more test data, the more accurate the error estimate.

3.7 Classification method

Classification serves as a fundamental task in machine learning, striving to predict the categorical class or label of an input by utilizing its unique features. This essential func-

tion finds extensive application across diverse domains, including image recognition, spam detection, medical diagnosis, and sentiment analysis. Machine learning, a pivotal methodology within this realm, constitutes a primary avenue for categorizing the multifaceted sciences. It delves into the mechanisms of autonomously acquiring the ability to make precise predictions based on historical observations. In the realm of artificial neural networks, recognizing the inevitability of incomplete raw data is crucial. Given that these networks typically demand a comprehensive dataset for accurate classification, addressing the issue of incomplete data becomes imperative. Through imputation and data completion strategies, we pave the way for the effective classification of data, ensuring the artificial neural network's capability to learn and predict remains robust.[18]

Classification methods based on Neural Networks, take a single neuron (processing element- PE) with inputs and outputs. Once more we have a set of training observation $(x_1, y_1), ..., (x_n, y_n)$ that we can use to build a classifier. With classifying examples into given sets of categories.

Examples of Classification problems we have the following:

- fraud detection
- optical character recognition
- machine vision (e.g., face detection)
- market segmentation (e.g., predict if customer will respond to promotion)
- bioinformatics (e.g., classify proteins according to their function)

Classification is the most widely used Machine learning technique that involves separating the data into different segments which are non-overlapping. Hence classification is the process of finding a set of models that describe and distinguish class label of the data object. Classification can be performed on structured or unstructured data. Classification

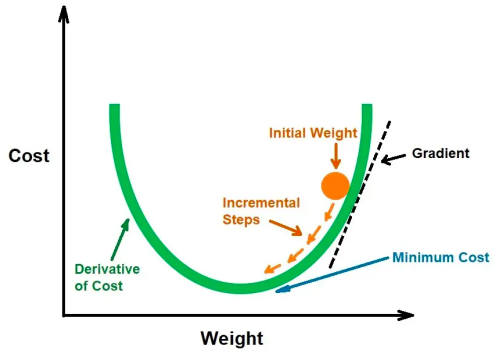

Figure 3.15: Graphical representation of the gradient decent method.

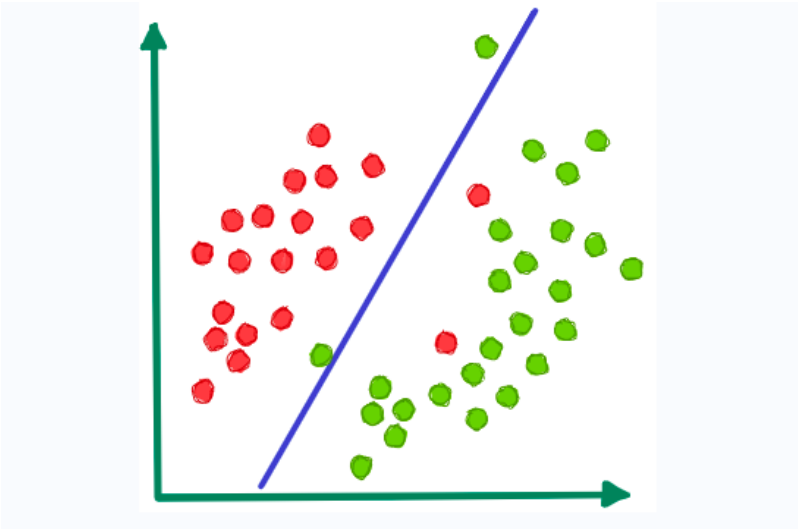

Figure 3.16: Classification.

is a technique where we categorize data into a given number of classes. The main goal of a classification problem is to identify the category/class to which a new data will fall under. Building an accurate classifier is always what we are working towards, for good test performance, we need 1) enough training examples, 2) good performance on training set and 3) classifier that is not "too ï¿œcomplex"ï¿œ (ï¿œOccamï¿œs razorï¿œ). With classifiers having to be "as simple as possible, but no simpler". We have "simplicity" closely related to prior expectation. Before moving forward a few terminologies needed when dealing with classification method.

- Classier: An algorithm that maps the input data to specific category.
- Classification model: A classification model tries to draw some conclusion from the input values given for training. It will predict the class labels/categories for the new data.
- Feature: A feature is an individual measurable property of a phenomenon being observed.
- Binary Classification: Classification task with two possible outcomes. Eg: Gender classification(Male/Female)
- Multiclass classification: Classification with more than two classes. In multi class classification each sample is assigned to one and only one target label. eg: An animal can be cat or dog but not both at the same time
- train the classifier: All classifiers in sciit-learn uses a fit (X, y) method to fit the model(training) for the given train data X and train label y.
- Predict the target: Given an unlabeled observation X, the predict(X) returns the predicted label y.

3.7.1 Regression method

Regression is a supervised learning task in machine learning that focuses on predicting a continuous numerical value or a real-valued output based on input features. The goal is to establish a relationship between the input variables and the target variable, allowing for the prediction of quantitative outcomes. Here are key aspects of regression methods in machine learning:

- 1. Objective: The primary objective of regression is to model the relationship between the independent variables (features) and the dependent variable (target) to make accurate predictions of continuous values.
- 2. Types of Regression:
  - Linear Regression: Assumes a linear relationship between the input features and the target variable. It seeks to find the best-fitting line that minimizes the sum of squared differences between predicted and actual values.
  - Multiple Regression: An extension of linear regression that involves multiple independent variables to predict a single dependent variable.
  - Polynomial Regression: Allows for capturing non-linear relationships by introducing polynomial terms of the input features.
  - Ridge Regression and Lasso Regression: Variants of linear regression with regularization terms to prevent overfitting.
  - Support Vector Regression (SVR): An extension of support vector machines for regression tasks, which aims to find a hyperplane that best represents the data.
  - Decision Tree Regression: Utilizes a decision tree structure to make predictions based on the input features.
  - Random Forest Regression: An ensemble method that combines multiple decision trees to improve accuracy and robustness.

  - Gradient Boosting Regression: Builds a sequence of weak learners (usually decision trees) to progressively correct errors made by the preceding models.
  - Neural Network Regression: Applies neural networks to regression problems, employing multiple layers of interconnected neurons to learn complex patterns.

- 3. Training and Evaluation:
  - Training: Regression models are trained on labeled datasets, where the algorithm learns to map input features to continuous target values.
  - Evaluation: Model performance is typically evaluated using metrics such as mean squared error (MSE), mean absolute error (MAE), R-squared, or other relevant metrics depending on the nature of the problem.

- 4. Handling Overfitting: Overfitting can be a concern, especially in complex models. Techniques like regularization and cross-validation are employed to mitigate overfitting and ensure better generalization.

- 5. Feature Importance: Some regression algorithms, such as decision trees and random forests, provide insights into the importance of different features in making predictions.

- 6. Interpretability: Linear regression models are often more interpretable than complex models like neural networks, making them valuable when understanding the relationship between variables is essential.

- 7. Deployment: Once trained and validated, regression models can be deployed for making predictions on new, unseen data.

- 8. Use Cases: Regression is used in various applications, including predicting stock prices, estimating house prices, forecasting sales, and modeling the relationship between variables in scientific research.

Regression methods stand as pivotal components in the realm of quantitative analysis, offering indispensable capabilities for modeling and predicting continuous outcomes across a spectrum of diverse fields. These methods, serving as foundational tools, play an instrumental role in both statistical modeling and machine learning, contributing significantly to our understanding and interpretation of relationships within datasets.

In conclusion, Chapter Two serves as a foundational exploration into the methodologies underpinning our study, with a particular focus on the analysis of medical datasets. We embarked on a journey through an introductory overview of the methods employed, laying the groundwork for a deeper understanding of our research. Additionally, a succinct introduction to the optimization techniques applied within our machine learning models was provided. This chapter establishes the framework for subsequent discussions, setting the stage for a comprehensive exploration of our study's methodologies and their implications in the realm of medical data diagnosis. Chapter three will cover briefly our starting projects in the realm of machine learning and applications across different fields; along with some appendix work and introduction to the data sets applied to our simulations on our proposed method.

Chapter 4

Medical Diagnosis

4.1 Medical Diagnosis

Medical diagnosis involves identifying the disease or condition that explains a person's symptoms and signs. This process relies on a health history, physical examination, and various tests, including blood tests, imaging tests, and biopsies, to establish an accurate diagnosis [3]. It is crucial to avoid wasting time on an incorrect course of treatment. Determining the cause of an illness is complex due to the similarity of symptoms among many diseases. Physicians interpret information from clinical interviews, physical exams, and diagnostic tests, ranging from blood tests to medical imaging such as X-rays, MRI, ultrasound, or CAT scans. Consultation with specialists may be required to interpret results and plan treatments.

Machine learning plays a vital role in enhancing and expediting the diagnosis process. It not only improves efficiency but also reduces the risk of misdiagnosis, thereby avoiding potentially severe consequences associated with incorrect diagnoses. The Society to Improve Diagnosis in Medicine (SIDM) highlights the prevalence of diagnostic errors, with over 12 million American adults receiving incorrect diagnoses annually [8]. Notably, women's heart problems are particularly challenging to detect compared to cardiac issues in men. Physicians may sometimes dismiss genuine symptoms as being "all in the patient's head" if test results do not provide an obvious answer visible to the naked eye. In light of this, medical diagnosis can be viewed as a classification problem. Classification involves determining the category or sub-population to which an observation or observations belong.

To classify data one can use the threshold function $f(x_1, ..., x_n)$ where $x = (x_1, ..., x_n)$

is a feature vector.

- If $f(x) > 0$, then $x \in C_1$.
- If $f(x) < 0$, then $x \in C_2$

C_1, C_2 are different classes (in this case diagnosis).

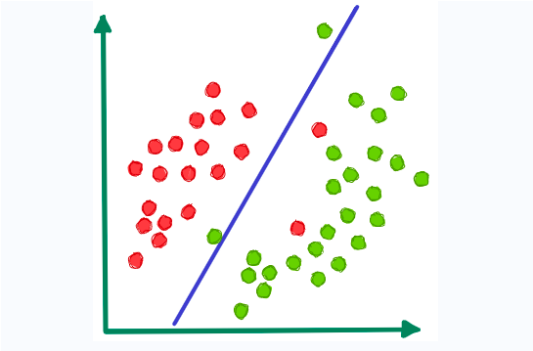

Figure 4.1: Decision boundary.

With the conceptualization of medical diagnosis as a classification problem, we can now delve into the applications of machine learning in this domain. The impact of machine learning on medical diagnosis and healthcare applications has been extensively studied. In its early stages, machine learning has demonstrated significant potential for enhancing diagnostic accuracy, resulting in notable advancements in saving time, reducing costs, and most importantly, saving lives [16]. Machine learning proves particularly influential in cancer diagnosis, leveraging data from medical imagery to detect, measure, and analyze tumors. Its computational prowess allows for faster and more efficient data and imagery analysis compared to the capabilities of individual human medical professionals. Notably, a study from China revealed that an AI system outperformed certain doctors in diagnosing common childhood diseases, emphasizing the remarkable diagnostic capabilities of machine learning in healthcare applications.

[30] The study trained a deep-learning system on 101 million data points generated from the electronic records of 1.3 million patient visits to a medical center in Guangzhou.

Studies found that the AI system was able to meet or outperform two groups of junior physicians in accurately diagnosing a range of ailments, from asthma and pneumonia, to sinusitis and mouth-related diseases. The AI was also able to meet or exceed diagnostic performance with some groups of senior physicians, for instance, in the category of upper respiratory issues.machine learning could complete screenings in less time. This could reduce referral wait times for high-risk patients. Machine learning could also broaden health care access. Some regions and populations in the United States have limited access to medical professionals. This emerging technology could automate certain tasks, which in turn could reduce clinical workloads and empower non-specialists to perform complicated tasks, such as cardiac imaging and analysis. This could allow medical professionals to reach larger segments of the population in at-home care or smaller clinical settings, and provide more patients with access to care. [23] Thus, the motivation behind this proposed work stems from observed cases wherein diagnostic accuracy has notably increased, reaching percentages as high as 90 to 95. In the subsequent sections, we will delve into our study involving a medical dataset, presenting precise medical diagnoses achieved through the application of various machine learning models. These models are implemented within our proposed optimized meta-learning framework.

4.2 Data Background - Pediatric Pneumonia Chest X-Ray Data Images

The first medical data set applied to the machine learning methodologies was for the classification of pediatric pneumonia, data provided by Kermany and Goldbaum [24]. Pneumonia, a severe lung infection, is a leading cause of mortality in young children, accounting for 14% of childhood deaths under 5 years old (Troeger et al., 2018). Sample image of a pediatric pneumonia image 4.2. For a quick overview on Pneumonia, it is an infection that inflames the air sacs in one or both lungs. The air sacs may fill with fluid or pus (pu-

rulent material), causing cough with phlegm or pus, fever, chills, and difficulty breathing (ref. Mayoclinic.org/diseases-conditions). Pneumonia can range in seriousness from mild to life-threatening. It is most serious for infants and young children, people older than age 65, and people with health problems or weakened immune systems.[23]

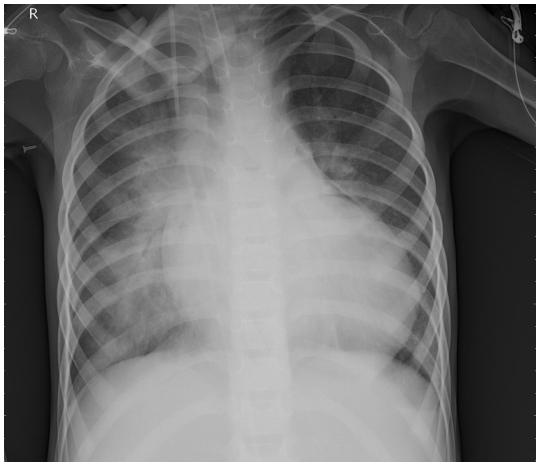

Figure 4.2: Pneumonia and lung infections.

Data initially collected was a total of 5,232 chest X-ray images from children, including 3,883 characterized as depicting pneumonia (2,538 bacterial and 1,345 viral) and 1,349 normal.[25]

The model was then tested with 40 normal images and 14 pneumonia images (8 bacterial and 6 viral) from 34 patients. After the pre-trained model of 100 epochs (iterations through the provided dataset) of the model, we were able to see good accuracy in classification. With later iterations of the model, we increase sizes on both training and testing, as validation sets.

There are two groups of pictures.

- Group 1 – healthy people
- Group 2 – sick people

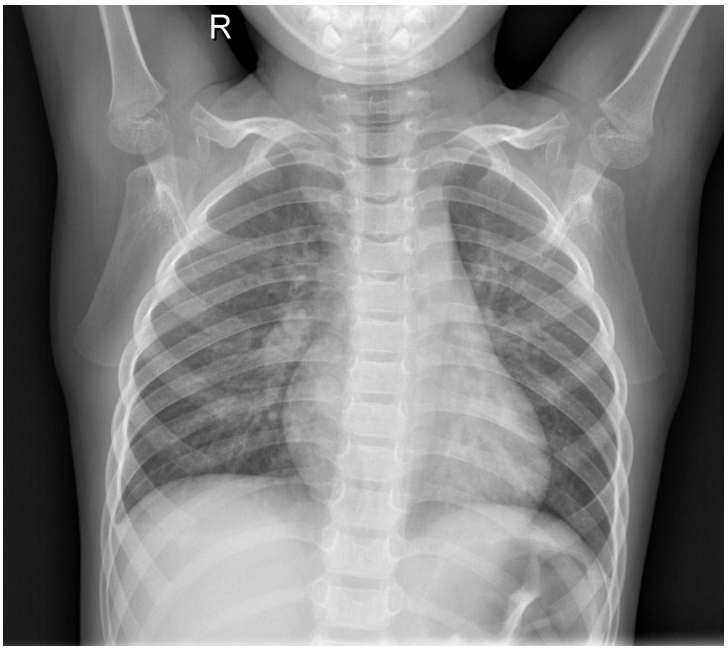

Figure 4.3: Chest X-ray image 1 healthy lung

Dataset is divided into 3 subsets.

- Training (healthy, sick)
- Testing (healthy, sick)
- Validation (healthy, sick)

In this specific framework in which we trained a neural network with the data of conventional approaches (convolutional neural network). The development of convolutional neural network layers has allowed for significant gains in the ability to classify images and detect objects in a picture [27].

```
Step 0:
loss = 0.587146103382 train acc = 0.9296875 val acc = 0.600000023842
```

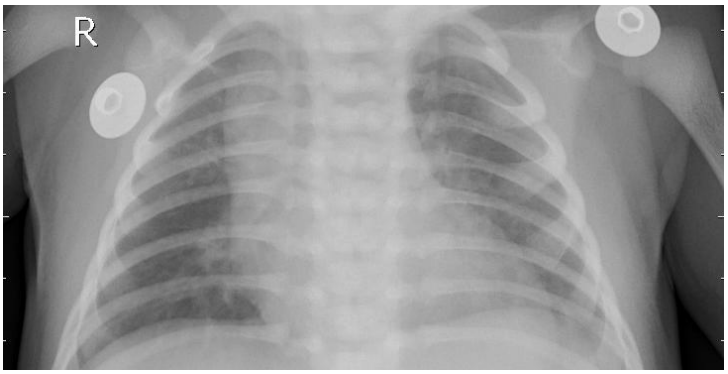

Figure 4.4: Chest X-Ray of person 378, bacteria infection

```
train acc = 0.9765625 val acc = 0.800000011921


Step 10: loss = 0.157285600901 train acc = 0.9765625
val acc = 0.800000011921
Step 20: loss = 0.0681663155556
train acc = 1.0 val acc = 0.759999990463
 loss = 0.0681663155556
train acc = 1.0 val acc = 0.759999990463
Step 30: loss = 0.0513591244817
train acc = 1.0 val acc = 0.759999990463
Step 30: loss = 0.0513591244817 train acc = 1.0 val acc = 0.759999990463
```

When the model was trained with a much smaller number of images (about 100), it retained high performance in accuracy, sensitivity, specificity, and area under the ROC curve for achieving the correct diagnosis and referral, thereby illustrating the high efficacy on using the CNN for image classifications, even with a limited training dataset.

```
Step 350: loss = 0.00186197937001
train acc = 1.0 val acc = 0.759999990463
```

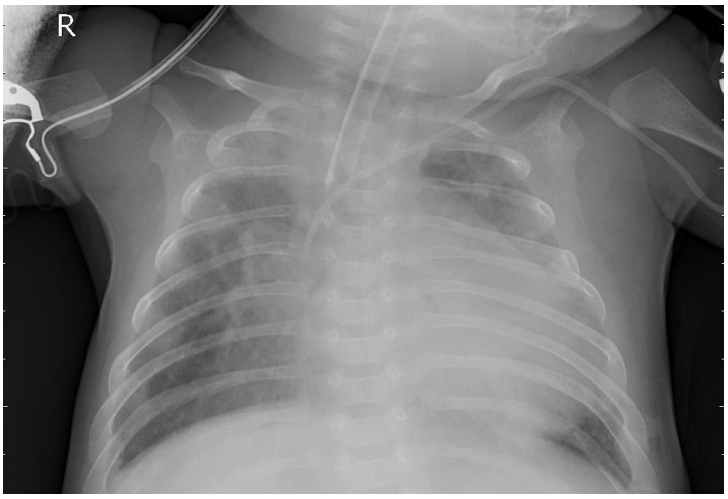

Figure 4.5: Chest X-Ray of person1, virus6 infection

```
Step 350: loss = 0.00186197937001 train acc = 1.0
val acc = 0.759999990463

Step 360: loss = 0.00185873173177
train acc = 1.0 val acc = 0.759999990463

Step 360: loss = 0.00185873173177 train acc = 1.0
val acc = 0.759999990463

Step 399: loss = 0.00168729701545 train acc = 1.0
val acc = 0.759999990463


Best validation accuracy = 80.0000011921
```

```
Final test accuracy =  88.8888895512
```

```
Total Model Runtime: 1min, 8.46sec
```

Furthermore, we have seen that the performance of this model would likely be enhanced when tested with a larger ImageNet dataset. As perhaps, with more advanced deep-learning techniques and architecture the rapid progression and development of medical image classification with the usage of convolutional neural networks has shown great results. AI beyond diagnosis or classification of images and into the realm of making treatment recommendations is a promising area of future investigation. The CNN represents a generalized platform that can potentially be applied to a wide range of medical imaging techniques (e.g., chest X-ray, MRI, computed tomography) to make a clinical diagnostic decisions, which enchances one of the concerned areas in the medical field behind image diagnoses and the accuracy behind the medical diagnoses.

4.3 Breast Cancer

The second medical dataset we studied is sourced from the UCI Machine Learning Repository, specifically the Breast Cancer Wisconsin (Diagnostic) Data. Before delving into the dataset specifics, it's essential to provide a brief overview of breast cancer. Breast cancer stands as the most prevalent cancer among women in the United States, excluding skin cancer. Statistically, approximately 30% of all newly diagnosed female cancers each year are attributed to breast cancer, illustrating its significant impact on women's health.[37]

Breast cancer is a complex and diverse disease with various subtypes, necessitating comprehensive research and diagnostic efforts. The dataset derived from the UCI Machine Learning Repository focuses on diagnostic information related to breast cancer in Wisconsin. In the upcoming section, we will explore the intricacies of this dataset and leverage different machine learning models within our optimized meta-learning framework to enhance our understanding of breast cancer diagnosis and contribute valuable insights to the

field of medical research.

The American Cancer Society's estimates for breast cnacner in the United States for 2022 alone are:

- About 287,850 new cases of invasive breast cancer (IDC) will be diagnosed in women.
- Breast cancer mainly occurs in middle-aged and older women. The median age at the time of breast cancer diagnosis is 62.
- About 43,250 women will die from breast cancer in 2022.[37]

Having the ability to accurately identifying and categorizing breast cancer subtypes is an important clinical task. With construction of an automated method(s) this may be used to save time and reduce error, again being one of many highlights on applying machine learning algorithms. Identifying significant risk factors contributing to diseases and other health conditions is a crucial breakthrough we eagerly anticipate. Through dedicated effort, we aim to make meaningful contributions to this vital area of research. The models under consideration span from simple logistic regression to kNN and SVM, encompassing the application of our meta-learning model. This diverse approach holds the potential to transform patient care, streamline administrative processes, and significantly impact healthcare goals and needs, particularly in the diagnostics of diseases like Breast cancer, Heart disease, and Lung cancer. This includes the task of developing new medical procedures, as stated before the handling of patient data and records and along with the treatment of chronic diseases.

As we strive to reduce diagnostic failure rates, which remain relatively high, the spectrum of medical AI applications encompasses diagnosis, disease screening, treatment, and prognostication. The ultimate goal is to optimize patient care by improving efficiency and enhancing areas such as imagery diagnosis, which serves as the inspiration for this research topic.

4.3.1 UCI Machine Learning Repository. Breast Cancer Wisconsin (Diagnostic) Data set.

For some description of our Breast Cancer Wisconsin (Diagnostic) data set which falls in the multivariate characteristics, with 569 number of instances and 32 attributes.

Attribute Information:

- ID number
- Diagnosis (M= malignant, B = benign)

Ten real-valued features are computed for each cell nucleus:

- radius (mean of distances from center to points on the perimeter)
- texture (standard deviation of gray-scale values)
- perimeter
- area
- smoothness
- compactness
- concavity (severity of concave portions of the contour)
- concave points (number of concave portions of the contour)
- symmetry
- fractal dimension ("coastline approximation"-1)

Prediction results of the Logistic Regression

Let's assume that f is a method for finding medical diagnosis by using logistic regression with some specific dataset [1]. Logistic regression has various parameter x. In presented example only one parameter is used $x = C$, where C is an inverse of regularization strength. The presented function was approximated using the meta-learning model implemented through a neural network, as illustrated by the extrapolated green line in Fig. 4.6. Prediction indicates that the values of the function f are increasing. In the next iteration bigger values of x will be considered; for verification purposes exact values of the function f were calculated (orange line) in the Fig.4.7). Next two iterations and related predictions are shown below, results given by the following neural network values of medical diagnosis as a function of parameter x are shown on figures 4.6 & 4.7.

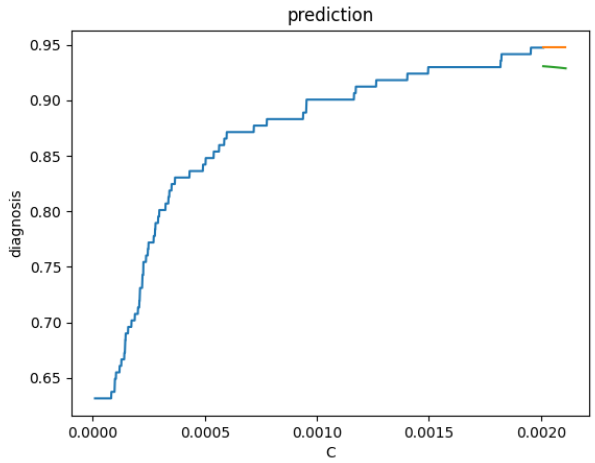

Figure 4.6: Approximation and prediction of the results of the logistic regression (iteration 1)

After every iteration the model is retrained and the accuracy of the prediction increases. The following result is the 10-iterations on the Figure 4.8.

This suggests that the model employed a quadratic equation to estimate the association between the variables. In this scenario, envision the data as naturally represented in

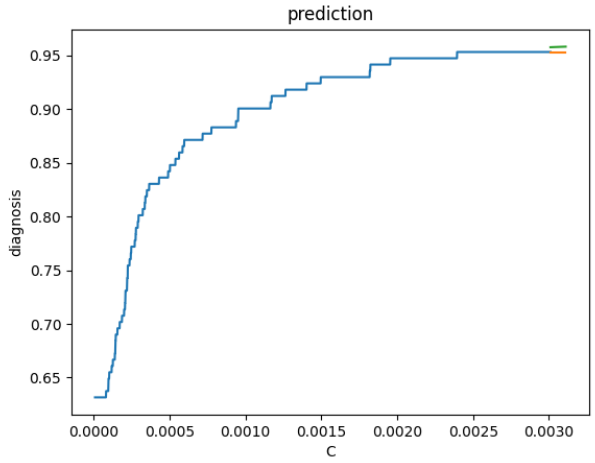

Figure 4.7: Approximation and prediction of the results of the logistic regression (iteration 2)

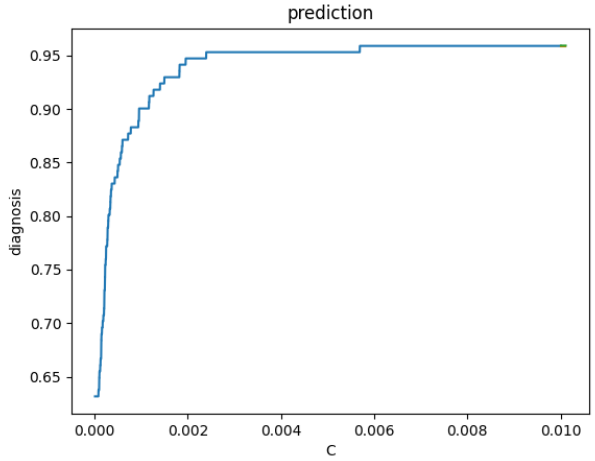

Figure 4.8: Approximation and prediction of the results of the logistic regression (iteration 10)

a graph, with nodes representing data points and edges indicating relationships or connections. Utilizing this graph structure, a graph neural network can discern how data points interact and influence one another. This led us to explore the amalgamation of polynomial

regression with a graph neural network. One is essentially allowing the model to benefit from both the polynomial features (which capture local interactions) and the global graph structure (which captures broader dependencies). This fusion of techniques can lead to more nuanced and accurate predictions.

Logistic regression, a prominent model in our study, it encompasses various parameters denoted by x. In our specific example, we focused on a singular parameter, namely $x = C$, where C represents the inverse of the regularization strength. The function we presented and approximated using a meta-learning model implemented through a graph neural network, and this process is exemplified by the extrapolation. Significantly, the model demonstrated outstanding adaptability to graph neural networks (GNN), underscoring its versatility in navigating intricate relationships within the data. Additionally, it leveraged polynomial regression as a meta-learning model for interpreting the data structure.

4.3.2 Graph representation of learning model for logistic regression

To create a graph representation of logistic regression it is possible to use the following Python code (file: 'dgl.py').

```
\begin{footnotesize}
import network as nx
import matplotlib.pyplot as plt


G = nx.Graph()


G.add_node('C', pos=(0, 2))
G.add_node('T', pos=(1, 1))
G.add_node('S', pos=(2, 1))
G.add_node('P', pos=(3, 2))
```

```
G.add_node('I', pos=(4, 1))
G.add_edge("C","T")
G.add_edge("T","S")
G.add_edge("S","P")
G.add_edge("P","I")

plt.title("Graph representation of logistic regression")
nx.draw(G, nx.get_node_attributes(G, 'pos'), with_labels=True)
plt.savefig('lr-graph.png')

A = nx.adjacency_matrix(G).todense()
print(A)
\end{footnotesize}
```

The final result is given in on the following picture 4.9.

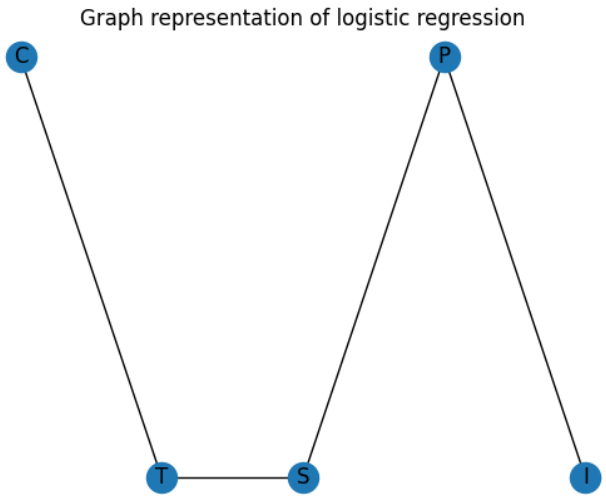

Figure 4.9: Graph represntation of logistic regression

In the graph the following nodes were included:

- C - quantitative variable, label 'C'
- tol - quantitative variable, label 'T'
- solver - cathegorical variable, label 'S'
- penalty - cathegorical variable, label 'P'
- intercept_scaling - quantitative variable, label 'I'

Preseted graph has the following adjacency matrix.

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{4.1}$$

In meta-lerning it is possible to use any parameter of the logistic regression.

- penalty'l1', 'l2', 'elasticnet', None, default='l2'
- dualbool, default=False
- tolfloat, default=1e-4
- C float, default=1.0
- fit_interceptbool, default=True
- intercept_scalingfloat, default=1
- class_weightdict or 'balanced', default=None
- random_stateint, RandomState instance, default=None
- solver'lbfgs', 'liblinear', 'newton-cg', 'newton-cholesky', 'sag', 'saga', default='lbfgs'
- max_iterint, default=100

- multi_class‘auto’, ‘ovr’, ‘multinomial’, default=’auto’
- verboseint, default=0
- warm_startbool, default=False
- n_jobsint, default=None
- l1_ratiofloat, default=None

Code for prediction of medical diagnosis by using graph neural networks

```
import dgl
import torch
import torch.nn as nn
import torch.optim as optim
import numpy as np
import scipy.sparse as sp
import networkx as nx

fileData = open('training-data-2.csv', 'r')
fileDataLineArray = fileData.readlines()

xInput = []
yInput = []
for line in fileDataLineArray:
    lineArray = line.split(',')
    number1Str = lineArray[0]
    number2Str = lineArray[1]
    number1Float = float(number1Str)
    number2Float = float(number2Str)
    xInput.append(number1Float)
    yInput.append(number2Float)
fileData.close()
```

```
# Create a set of graphs with features
num_graphs = len(xInput)
graph_list = []
numberOfFeatures = 1
for i in range(num_graphs):
    #num_nodes = np.random.randint(5, 15)
    num_nodes = 5
    graph = dgl.graph(([0, 1, 2, 3, 4], [1, 2, 3, 4,0]))
    graph.ndata['features'] = torch.tensor([[xInput[i]], [1], [1], [1], [1]])
    #graph.ndata['features'] = torch.randn(num_nodes, numberOfFeatures)  # 5-dimensional node features
    print(graph.ndata['features'])
    graph_list.append(graph)
    print(graph)
    print(graph.ndata['features'])

    # Convert the graph to a NetworkX graph
    nx_graph = graph.to_networkx()

    # Get the adjacency matrix from the NetworkX graph
    adjacency_matrix = nx.adjacency_matrix(nx_graph).todense()

    print("Adjacency Matrix:")
    print(adjacency_matrix)

# Create graph labels
#graph_labels = torch.randint(0, 2, (num_graphs,))  # Binary labels
#graph_labels = torch.tensor([1., 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1])
graph_labels = torch.tensor(yInput)
print('graph_labels')
print(graph_labels)

# Define a Graph Neural Network (GNN) model for graph classification
class GNNGraphClassification(nn.Module):
    def __init__(self, in_dim, hidden_dim, out_dim):
```

```
        super(GNNGraphClassification, self).__init__()
        self.conv1 = dgl.nn.GraphConv(in_dim, hidden_dim)
        self.conv2 = dgl.nn.GraphConv(hidden_dim, out_dim)

    def forward(self, graph_list, features_list):
        predictions = []
        for g, features in zip(graph_list, features_list):
            x = self.conv1(g, features)
            x = torch.relu(x)
            x = self.conv2(g, x)
            g.ndata['h'] = x
            hg = dgl.sum_nodes(g, 'h')
            predictions.append(hg)
        return torch.stack(predictions)

# Instantiate the GNN model
gnn_model = GNNGraphClassification(in_dim=numberOfFeatures, hidden_dim=10, out_dim=1)  # 1-dimensional

# Define loss function and optimizer
#criterion = nn.BCEWithLogitsLoss()  # Binary Cross-Entropy loss
criterion = nn.MSELoss()
optimizer = optim.Adam(gnn_model.parameters(), lr=0.01)

# Training loop
num_epochs = 500
for epoch in range(num_epochs):
    optimizer.zero_grad()
    predicted_labels = gnn_model(graph_list, [g.ndata['features'] for g in graph_list])
    predicted_labels = predicted_labels.view(-1)  # Reshape to match expected shape
    loss = criterion(predicted_labels, graph_labels.float())
    loss.backward()
    optimizer.step()

    if (epoch + 1) \% 10 == 0:
```

```
        print(f"Epoch [{epoch+1}/{num_epochs}], Loss: {loss.item()}")

# Make predictions on new graphs
new_graph = dgl.graph(([0, 1, 2, 3,4], [1, 2, 3, 4,0])) # Example new graph with 3 nodes
new_graph = dgl.add_self_loop(new_graph)  # Add self-loops to the new graph
new_graph.ndata['features'] = torch.tensor([[0.00020999999999999998], [1], [1], [1], [1]])
with torch.no_grad():
    predicted_label = gnn_model([new_graph], [new_graph.ndata['features']])

print("Predicted Label for New Graph:", predicted_label)
```

Learning Curve of meta-learning model gnn

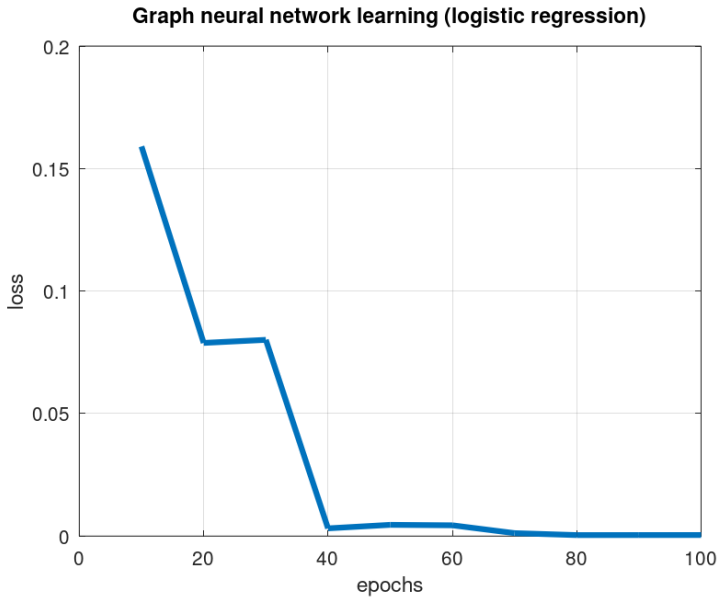

Figure 4.10: Learning curve for graph neural network

Prediction

In presented example computational graph related to logistic regression was represented by the graph 4.10. Node C represents inverse of regularization strength. Training set D_T

of the graph neural network has the following structure

$$D_T = \{(graph_1, label_1), (graph_2, label_2), ..., (graph_n, label_n)\} \tag{4.2}$$

Labels are equal to medical diagnosis.

Graph neural network can compute/predict value of medical diagnosis ($label_i = diagnosis_i$) for given computational process represented by the $graph_i$

$$diagnosis_1^p = label_1^p = GNN(graph_1, W)$$

$$diagnosis_2^p = label_2^p = GNN(graph_2, W)$$

...

$$diagnosis_n^p = label_n^p = GNN(graph_n, W)$$

Predicted labels $label_1^p, label_2^p, ..., label_n^p$ can be compared to labels from the data set $label_1, label_2, ..., label_n$ and appropriate Loss function can be constructed.

During the training process weight matrix W is computed based on the training set.

$$W^* = \arg\max_W Loss(GNN, W, D_T) \tag{4.3}$$

There are many options for the Loss function (cross-entropy, mean squared error, mean absolute error, hinge-loss, etc.)

For given computational graph $graph_s$ and to train graph neural network with known weight matrix W^* it is possible to calculate prediction $diagnosis_s$.

$$diagnosis_s = GNN(graph_s, W^*) \tag{4.4}$$

Value $diagnosis_s$ can be re-applied in meta-learning.

In presented example for computational graph with value $C = 0.000209999$ the following diagnosis was predicted 0.7035.

In presented example medical diagnosis was calculated by using logistic regression (data set [1]). Computational graphs visually represent the sequence of operations in an algorithm

or mathematical model. Each node in the graph signifies a distinct operation, and the edges delineate the flow of data or computations. Subsequent predictions were made utilizing these meta-learning representations where $x_{Input[i]}$ is a value of parameter C in logistic regression. Table 4.1 showcases observation of the relationship between epochs and loss performance (time optimization), which shows to be crucial in training machine learning models. Initially, as the model learns patterns from the training data, the loss typically decreases. However, there's a risk of overfitting if the model becomes too specialized to the training data. Monitoring loss over multiple epochs helps practitioners understand when the model has reached a suitable level of generalization and can make accurate predictions on new, unseen data; giving us a great implication for our meta-learning model.

Table 4.1: Epochs' relation on Loss Performance for Meta-learning model

Epochs	Loss
470/500	0.00038865587
480/500	0.00039655691
490/500	0.0003965569
500/500	0.0003998697

This procedure shown by used meta-learning, predicted label for new graph diagnosis([[0.6162]]), the actual predicted label of diagnosis was 0.72.51, once the model ran through the entirely epoch cycles, we showcased improvement of loss function and higher accuracy of medical diagnosis. The described procedure is a form of meta-learning, which as perviously mentioned it involves training a model on a diverse set of tasks or datasets, enabling it to quickly adapt and perform well on new, unseen tasks. In this case, the GNN is meta-trained on various computational graphs, allowing it to generalize its learning to different medical diagnoses. According to numerical results graph neural networks can be successfully applied in prediction of values of medical diagnosis.

4.3.3 Prediction of the results of the k-Nearest Neighbors (kNN)

Medical diagnosis (data set [1]) can be computed by the k-Nearest Neighbours method (kNN). In this example, $'x'$ signifies the number of neighbors, a pivotal parameter in the k-Nearest Neighbors (kNN) method used for medical diagnosis. kNN relies on assessing the similarity of cases in a dataset to generate predictions. In conjunction, we employed polynomial regression as a meta-learning model—a potent technique adept at capturing nuanced data relationships, particularly in scenarios where linear models may prove inadequate. The application of polynomial regression served as the most straightforward and effective approach for implementing both KNN and the meta-learning model.[43]

By incorporating a graph neural network as a meta-learning model, we tap into the inherent structure of medical data, leading to more precise and informed diagnoses. This amalgamation of kNN, polynomial regression, and GNNs creates a robust tool for improving medical diagnosis and treatment strategies. This comprehensive approach offers personalized care rooted in the collective knowledge embedded in the medical graph.

Figure 4.11 showcases the outcomes of these predictions, offering a visual depiction of how effectively the models forecasted medical diagnoses based on the selected parameters and methodologies.

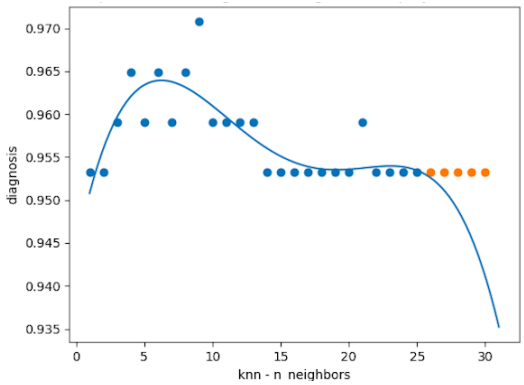

Figure 4.11: Approximation and prediction of the results of the kNN by polynomial regression meta-learning model

In the next example parameter x was equal to power parameter for the Minkowski metric Fig.4.12 distance metrics for machine learning. The case where $p = 1$ is equivalent to the Manhattan distance and the case where $p = 2$ is equivalent to the Euclidean distance.[43]

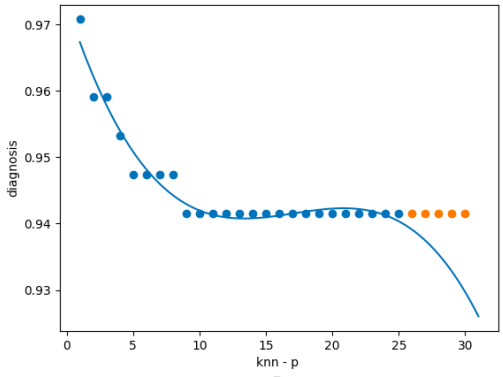

Figure 4.12: Approximation and prediction of the results of the kNN by using polynomial

kNN, with the Minkowski power parameter playing a pivotal role, exhibited intriguing results when integrated into the meta-learning model alongside the base model. This seamless integration demonstrated a smooth transition, aligning well with polynomial regression and extending into the domain of computational graphs, yielding noteworthy outcomes. Following this approach, we extended the simulation to our graph neural network meta-learning model. As illustrated in Figure 4.13, the learning curve depiction through our base model, kNN, underscores its remarkable adaptability in learning new tasks through our meta- learning model.

In the context of a computational graph with a specified value of $n = 10$, the k-nearest neighbors model provided a diagnostic prediction of 0.9582. Our meta-learning model involved training model on a variety of tasks (features), allowing it to learn how to learn effectively. In the case of the computational graph and kNN model, integrating this meta-learning component could mean refining the model's ability to adapt and make accurate predictions based on the specific characteristics and patterns present in different datasets or scenarios. The meta-learning model (MtL) essentially captures higher-level knowledge

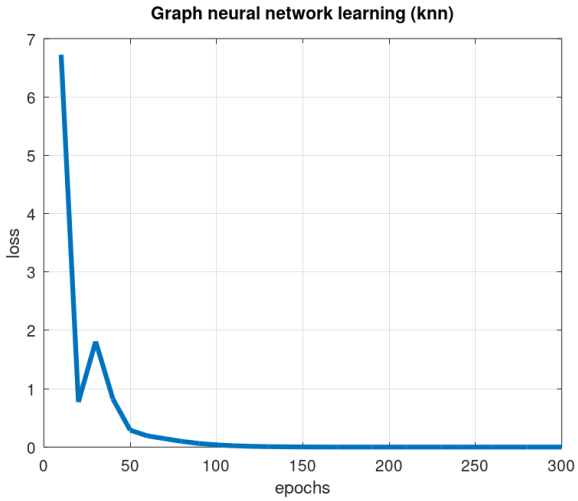

Figure 4.13: Learning curve for graph neural network (base model - kNN)

from the base model, enabling more efficient learning and prediction in diverse situations.

4.3.4 Prediction of the results of Support Vector Machine

As our study focuses on the critical impact of hyper-parameter values on the predictive performance of machine learning algorithms. We move on to our third base model applied to our meta-learning model. The importance of hyper-parameter tuning, while beneficial, often incurs high computational costs, particularly on larger datasets, and may not consistently outperform default values.[43] The research has proposed a recommender system based on meta-learning specifically designed to determine optimal scenarios for using default values versus tuning hyper-parameters for Support Vector Machine base model on new unseen datasets. The results demonstrate the system's effectiveness in accurately predicting when hyper-parameter tuning significantly enhancing model performance. Notably, SVMs are primarily influenced by four hyper-parameters: kernel function (k), its width (γ) or polynomial degree (d), and the regularized constant(C). Leveraging Meta-learning (MtL), the system proves capable of reducing the overall tuning cost without substantial loss in predictive performance.

A search consists of:

- an estimator (regressor or classifier such as sklearn.svm.SVC());
- a parameter space;
- a method for searching or sampling candidates;
- a cross-validation scheme; and
- a score function.

```
The best parameters for using this model is ('C': 1000, 'gamma': 0.01, 'kernel': 'rbf').

accuracy.score
0.9473684210526315
```

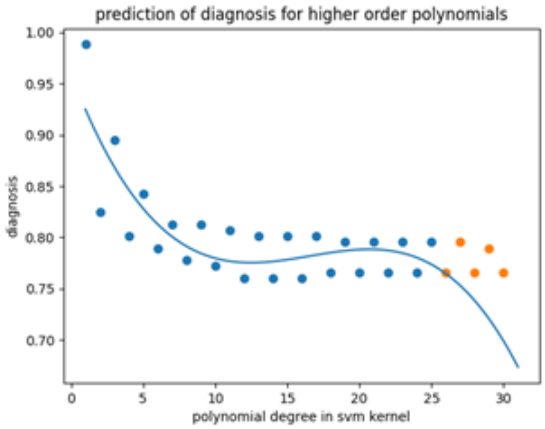

Figure 4.14: Prediction of diagnosis for higher order polynomials (SVM base model)

kernel	max.score
linear	0.875
neuralnet	0.880
rbf	0.905

Table 4.2: best GridSearch parameters SVM model

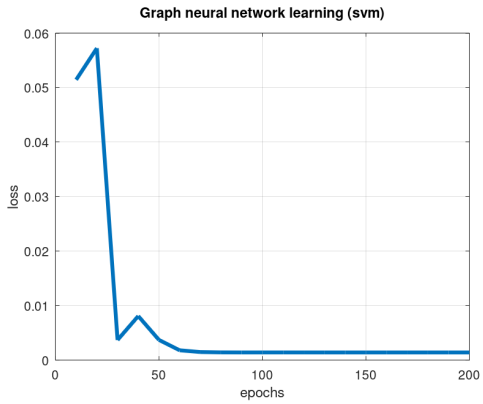

Figure 4.15: Learning curve of meta-graph learning mode (base model: SVM)

The results of medical diagnosis for Support Vector Machine (SVM) indicate that the parameter influencing the outcomes is the degree of the polynomial kernel. Notably, the highest diagnosis accuracy, reaching 0.988304, is achieved when the degree is set to 1. The application of meta-learning in this context justifies this observation as the function demonstrates a decreasing trend.

For SVM in a computational graph with a specified value of $p = 10$ (degree of the polynomial kernel), the predicted diagnosis is 0.8142. This information suggests a potential decrease in diagnosis accuracy compared to the optimal setting observed with a degree of 1. The influence of different polynomial degrees in SVM models highlights the significance of hyper-parameter tuning in achieving optimal predictive performance through our meta-learning model.

4.4 Results

In our exploration of medical diagnosis, we initially examined the performance of Support Vector Machines (SVM), k-Nearest Neighbors (KNN), and logistic regression models. While these models provided valuable insights, we recognized the potential for further optimization and improvement. The present findings encompass the performance of models prior to the implementation of meta-learning model. [24] [1] Table 4.3 showcases the exploration on the impact (time optimization) of epochs on loss function performance.

Table 4.3: Comparative Performance Metrics Across Different Base Models

	Test-Accuracy	Loss
Logistic regression	0.8138	0.3419
K-nearest neighbors	0.7797	0.4107
Support vector machine	0.6998	0.4769

To enhance our predictive modeling capabilities, we integrated advanced techniques,

including feature extraction, and applied our meta-learning models. Through the integration of feature extraction and meta-learning, our objective is to achieve superior results in medical diagnosis compared to the initial models. The optimized predictive model results are presented in Table 4.4, showcasing the advancements made through our integrated approach.

Table 4.4: Optimized Predictive models results.

	Test-Accuracy	Loss
Logistic regression	0.9415	0.03794
K-nearest neighbors	0.87076	0.20197
Support vector machine	0.8801	0.3183

In conclusion, the results observed in our study demonstrate considerable improvements in the outcomes of predictive modeling. This advancement underscores the growing importance of data-driven approaches in the healthcare domain, particularly in the context of medical diagnosis. The enhanced performance highlights the potential and value that sophisticated data analysis and modeling techniques bring to the forefront, contributing to more effective and accurate healthcare decision-making processes. As we conclude Chapter Five, we will summarize our findings and outline future avenues for exploration and improvement.

Chapter 5

Conclusions

Medical diagnosis, essentially a classification problem, can be represented as a computational graph using classification algorithms. Our investigation underscores a substantial enhancement in medical diagnosis achieved through the innovative use of a pre-training, meta-learning model approach. The demand for precise and reliable diagnostic outcomes is paramount in improving patient care, treatment strategies, and overall healthcare outcomes. In an era where technology continues to advance, the fusion of machine learning methodologies with medical databases offers a transformative opportunity to elevate diagnostic accuracy to unprecedented levels. The significance of this multifaceted approach is underscored by its potential to reshape the landscape of healthcare. From introducing innovative medical procedures to addressing the complexities of chronic disease management, this approach strives to bring about positive transformations. As we confront the challenge of high diagnostic failure rates, the integration of AI in healthcare not only aims to enhance accuracy but also to establish a comprehensive framework that positively influences patient outcomes. The ultimate aspiration is to create a healthcare environment that is not only technologically advanced but also optimized for precision, efficiency, and improved patient care. In figure 5.1 showcases the benefits of healthcare AI products across different services.

This dissertation has presented applications of neural networks and modern machine learning methods, to the problem of prediction in customer credit card default classification, and with 9 different countries financial stock market and SP500 properly forecasted. With our summer research that consisted of imputation methods and finding the missing values with the use of neural networks. These results contained some findings which are counter-intuitive and should be investigated further. Our contribution involved a thorough

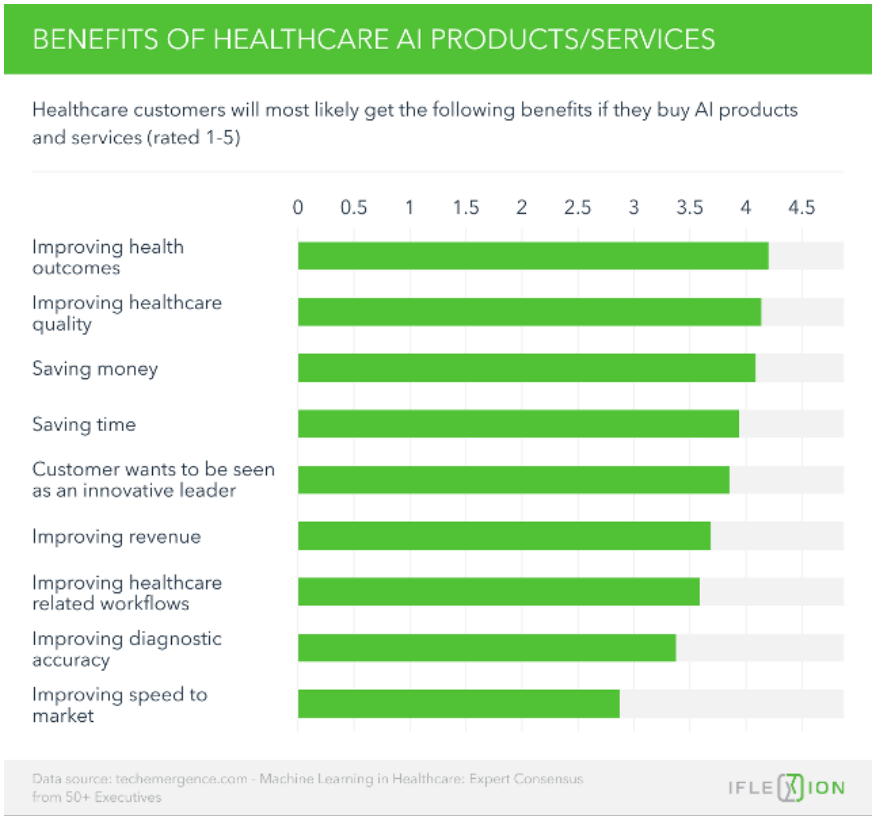

Figure 5.1: Benefits of health-care artificial intelligence products.

and careful demonstration of how a meta-learning model can be seamlessly integrated with advanced machine learning algorithms. Through a precise and skillful mathematical and statistical combination of these methods, we successfully realized our ultimate objective of improving medical diagnosis. This integration allowed us to harness the strengths of both meta-learning and advanced machine learning, resulting in a more robust and effective approach to medical diagnosis. It investigates a range of algorithms and advanced techniques for approximating functions, with the goal of finding the best configurations. Specifically, optimizing a function with n variables can be framed as optimizing a graph with n inputs and 1 output value. Our current contribution emphasizes in the crucial task of accurately identifying and categorizing breast cancer subtypes in a clinical setting. An automated method can save time and reduce errors, showcasing the application of machine learning algorithms (future plans). The investigation aims to contribute to identifying significant risk factors for diseases, enhancing medical diagnosis. This diverse approach involves various machine learning algorithms, including logistic regression, kNN, and SVM, with the integration of a meta-learning model. The significance of this approach extends to transforming patient care, streamlining administrative processes, and impacting healthcare goals on improving medical diagnosis. The incorporation of polynomial regression with a graph neural network opens new possibilities for predicting properties and enhancing diagnostic accuracy. Logistic regression, a key model, is explored with a focus on the regularization strength parameter C. The meta-learning model demonstrates adaptability to graph neural networks and leverages polynomial regression for interpreting data structure.

The study also delves into computational graphs, representing the sequence of operations in algorithms. Meta-learning representations are employed for predictions, and given illustrations of the relationship between epochs and loss performance during training (time optimization). Just as hyper-parameter tuning's showcased critical impact on predictive performance was discussed, and a recommender system based on meta-learning is proposed for Support Vector Machine models. Results from SVM models show the influence of the polynomial kernel degree on diagnosis accuracy. The meta-learning model helps reduce

tuning costs without significant performance loss. The study concludes with noteworthy improvements in predictive modeling outcomes, emphasizing the growing importance of data-driven approaches in healthcare and contributing to more effective decision-making. Future avenues for exploration and improvement are outlined.

5.1 Future plans

For future and currently exploring the idea of self creating an "autonomous" mega-meta-learning model, referring to a machine learning model that will have the capability of operating with a significant degree of independence and automation in various stages of the machine learning lifecycle. I have currently studied the autonomy is achieved through the integration of automated processes and algorithms that enable the model to perform tasks without continuous human intervention. Which is key when being applied specially to health-care field, this giving medical experts the ability to be more hands on with other high demanding medical issues that are in need of such human interaction. To briefly give some key aspects of an autonomous ML model include:

- Automated Training: The model can automatically learn from data without explicit programming. Automated machine learning (AutoML) tools are often used to optimize algorithms, hyper-parameters, and feature selection.
- Feature Engineering: The model can automatically extract relevant features from raw data, eliminating the need for manual feature engineering. This is particularly useful in scenarios with large and complex datasets.
- Adaptability: Autonomous ML models can adapt to changing data distributions and patterns over time, ensuring that they remain effective in dynamic environments.
- Model Deployment and Monitoring: These models can handle deployment to production environments automatically and continuously monitor their performance, making adjustments as needed.

- Decision-Making: Some autonomous models are designed to make decisions and take actions based on their predictions without external approval, especially in autonomous systems like self-driving cars or automated trading platforms.

The goal of autonomous ML models is to reduce the need for manual intervention, streamline the machine learning process, and improve the efficiency and effectiveness of model development and deployment. AutoML tools and frameworks play a crucial role in achieving autonomy by automating repetitive tasks and allowing data scientists and engineers to focus on higher-level aspects of the machine learning pipeline. My future plans are driven by my summer job experience in 2021 with the Office of Naval Intelligence (ONI). Initially, I applied my expertise in machine learning and mathematics to contribute to this government institute. The experience exposed me to an exceptionally diverse dataset, serving as a catalyst for pursuing my Ph.D. and focusing on a project tailored to the specific needs of the Navy branch.

The project aims to address challenges within the TAC-55 data science group, emphasizing the application of advanced mathematical and statistical techniques. The primary goal is to innovate and enhance the group's capabilities in predicting and classifying information with minimal human interaction. The project aligns with the critical need for high-level analytics and automation in naval intelligence operations. Overall, this endeavor represents a strategic contribution to advancing data science capabilities within the Navy.

Chapter 6

Timeline

Table 6.1: Timeline

Timeline	Task
Aug. – Dec. 2020	Investigate and analyze the effects of missing data with *Machine learning algorithms*. Examine and explore the Imputation methos (MICE, Ameila) for application to large dataset (seldom data). Look into Machine learning algorithms along with imputation methods. Apply methodologies to the missing Educational dataset and analyze its best fit for classification information. Completion of methodologies *missing data, machine learning*. Acknowledgements: Statistical Science Group, Los Alamos National Lab.

Jan. – May 2021	Establish a Classification model via Classification and Regression Tree (CART), a predictive algorithm used in *machine learning*. Investigate the type of classification algorithm for CART and implement to dataset on Food and Housing Insecurity among college students enrolled at a public Hispanic Serving Institution (HIS). Identify groups in the data with desirable characteristics, analyzing specific sets of variables that create a "path" leading to Food Insecurity (FI) and Housing Insecurity (HI) among current students.
Aug. – Dec. 2021	Develop a *machine learning* process via neural network. Utilization of Convolutional Neural Networks (CNN) for automatically detecting important features. Applying CNN to CT images with Keras Project for dataset on image classification of brain tumor via CT scans. Investigate applications of *machine learning techniques, neural networks*, self-supervised learning, clustering; for patients with brain tumors and with pneumonia by imagine classification. Motivation is to optimize algorithm directly.

Jan. – May 2022	Comprehensive review of currently used *machine learning* techniques for classification of medical data related to datasets. Investigate (Neural networks, Logistic regression, Random forest, Classification and Regression Tree (CART), decision tree etc.). Identification of significant parameters and algorithms which can be used in optimal configurations. Tested various approaches for finding optimal machine learning algorithm (evolutionary programming, computational graph optimization etc.).
Jun. – Aug. 2022	Investigate the Invasive ductal carcinoma (IDC) breast cancer in **UCI Machine Learning Repository. Breast Cancer Wisconsin (Diagnostic**) Data set. Apply logistic regression a significant machine learning algorithm. Assign predictor variables to dataset ('radius_mean', 'perimeter_mean', 'area_mean', 'compactness_mean' and 'concave_points_mean') with outcome variable 'diagnosis'.
Aug. – Dec. 2022	Submit dissertation proposal Aug. 30,2022. Expand logistic regression method, with addition of Multiclassifiers. Integrate Deep Convolutional neural networks and Random Forest Classifiers for classification of breast cancer. Successfully presented at NMSU/UTEP 2022 Fall workshop. Narrowing machine learning algorithms for optimization. Continue to algorithm for graph neural network optimization method.

Jan. – Mar. 2023	Applying classification models developed to new medical diagnosis, datasets (lung cancer, brain tumors, heart disease etc..). Showcase results for *Deep Convolutional neural networks and random forest classifiers.* Compare results with pervious breast cancer dataset. Record results on medical diagnosis and machine learning methodologies.
April – May 2023	Collect and revise recorded information and results. Submit draft for paper publication "Machine Learning Methodologies with Applications to Medical Diagnosis".
Jun.- July 2023	Finish paper and attend 2023 Hawaii University International Conferences Science, Technology and Engineering, Mathematics and Education Prince Waikiki Resort, Honolulu, Hawaii. -Start modeling our built in meta-learning model to machine learning methods. -Start graph meta-learning simulations.
Aug. – Oct. 2023	Application of optimization method proposed Meta-learning. Collect results and summarize observations, new title for defense: "Integrating Machine Learning and Optimization Methods for Medical Diagnosis". - Present Slides at the UTEP/NMSU 2023 Mathematical and Computational Sciences Workshop. - Gather information on self built meta-learning model. -Continue to work on dissertation defense.

Nov. 2023	–	Dec.	Finish gathering results and writing for dissertation, continue to transfer information onto slides presentation. Work on publishing new paper with current research approach of computational graphing and meta-learning. - Defend dissertation. Future work: continue to improve current built predictive model, ultimate goal is to extend our current model to an "autonomous" ML model.

References

[1] Diagnostic wisconsin breast cancer database. `https://archive.ics.uci.edu/dataset/17/breast+cancer+wisconsin+diagnostic`.

[2] Math works, introducing machine learning.

[3] National cancer institute, 2023.

[4] C. Arlot and A. Celisse. A survey of cross-validation procedures for model selection. 2010.

[5] E. Barany, M.P. Beccar Varela, I. Florescu, and I. Sengupta. Detecting market crashes by analysing longmemory effects using high-frequency data. *Quantitative Finance*, 12(4):623–634, 2012.

[6] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

[7] James Bergstra, Daniel Yamins, and David D. Cox. Practical hyperparameter optimization methods for machine learning. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, pages 1203–1211, 2013.

[8] Joseph Berkson. Application of the logistic function to bio-assay. *Journal of the American Statistical Association*, 39(227):357–365, 1944.

[9] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[10] A. Bohr and K. Memarzadeh. The rise of artificial intelligence in healthcare applications. *Artificial Intelligence in Healthcare*, 2020.

[11] S.V. Buuren and K. Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in r. *Journal of Statistical Software*, 45(3), 2011.

[12] Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the Twenty-third International Conference on Machine Learning (ICML 2006)*, 2006.

[13] Kai Chen, Chen Liang, Jiaxin Wang, and Ce Zhang. Neural databases. *arXiv preprint arXiv:2010.06973*, 2020.

[14] Ole Christensen and Khadija Laghrida Christensen. *Approximation Theory: From Taylor Polynomials to Wavelets.* Applied and Numerical Harmonic Analysis. Birkhäuser, Boston, MA, USA, 1st edition, 2004. Corr. 2nd printing 2005, Corr. 3rd printing 2006.

[15] E.F. Fama. The behavior of stock-market prices. *The Journal of Business*, 38(1):34–105, 1965.

[16] GAO. Machine learning's potential to improve medical diagnosis. *GAO Blog*, 2022.

[17] Avi Gopani. History of machine learning. *International Journal of Computer Science and Mobile Computing*, 8(6):1–7, 2019.

[18] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer, New York, NY, USA, 2nd edition, 2008.

[19] James Honaker, Gary King, and Matthew Blackwell. Amelia ii: A program for missing data.

[20] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[21] M.-P. Hosseini, A. Hosseini, and K. Ahi. A review on machine learning for eeg signal processing in bioengineering. *IEEE Rev. Biomed. Eng.*, 2020.

[22] et al. Hu. Pre-training graph neural networks. 2019.

[23] Xiaowei Hu, Yu Yan, Wenhao Ren, Hongsheng Li, Yifan Zhao, Ahmad Bayat, and Bjoern Menze. Feedback graph attention convolutional network for medical image enhancement. *arXiv preprint arXiv:2006.13863*, 2020.

[24] D. Kermany and M. Goldbaum. Labeled optical coherence tomography (oct) and chest x-ray images for classification. Mendeley Data, 2018.

[25] Daniel Kermany, Kang Zhang, and Michael Goldbaum. Labeled optical coherence tomography (oct) and chest x-ray images for classification, 2018.

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. pages 1097–1104, 2009.

[27] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25, pages 1097–1105. Curran Associates, Inc., 2012.

[28] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[29] Liu Lu, Zhou Tianyi, Long Guodong, Jiang Jing, and Zhang Chengqi. Learning to propagate for graph meta-learning. *arXiv:1909.05024*, 2019.

[30] Adam Rasmi. Ai systems in china have produced another study showing the potential of ai in medical diagnosis. *Quartz*, 14, 2019.

[31] Matthias Reif, Faisal Shafait, and Andreas Dengel. Meta-learning for evolutionary parameter optimization of classifiers. *Machine Learning*, 87:357–380, 2012.

[32] Donald B Rubin. Multiple imputation after 18+ years. *Journal of the American Statistical Association*, 91(434):473–489, 1996.

[33] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[34] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Trans. Neural Netw*, 20(1):61–80, 2008.

[35] Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. Structured sequence modeling with graph convolutional recurrent networks. In *Proc Int Conf Neural Inf Process (ICONIP)*, pages 362–373, 2018.

[36] M. Smieja, L. Struski, J. Tabor, B. Zielinski, and P. Sputnik. Processing of missing data by neural networks. In *32nd Conference on Neural Information Processing Systems*, NeurIPS 2018, Montréal, Canada, 2018.

[37] American Cancer Society. American cancer society, Accessed: 05/15/2021.

[38] Tingting Song, Wenming Zheng, Peipei Song, and Zhen Cui. Eeg emotion recognition using dynamical graph convolutional neural networks. *IEEE Trans Affect Comput*, 2018.

[39] D.J. Stekhoven and P. Buehlmann. Missforest - nonparametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118, 2012.

[40] Yu-Sung Su, Andrew Gelman, Jennifer Hill, and Masanao Yajima. Multiple imputation with diagnostics (mi) in r: Opening windows into the black box. *Journal of Statistical Software*, 45(2):1–31, 2011.

[41] John W Tukey. *Exploratory data analysis*. Addison-Wesley, 1977.

[42] Li Yang, S. S. Iyengar, Shenghuo Zhu, and Kun Fu. Hyperparameter tuning in deep learning: A review. *ACM Computing Surveys (CSUR)*, 52(1):1–38, 2019.

[43] Z. Zhang. Introduction to machine learning: K-nearest neighbors. *Ann Transl Med*, 4(11):218, 2016.

Curriculum Vitae

Jazmin Quezada an El Paso, Texas native received a Bachelor's and Master's degree in Mathematics from University of Texas at El Paso. Her scientific interests are related to Machine Learning \AI and forecasting of missing and hidden data by using different machine learning methodologies. She presented her scientific work on several scientific workshops. In June 2019, participation in a internship at the Los Alamos National Laboratories.