ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ &
ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Μια ολοκληρωμένη ανάλυση αλγορίθμων μηχανικής μάθησης για την ταξινόμηση εικόνων με θόρυβο με εφαρμογές στην αστρονομία

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΝΤΟΛΚΕΡΑΣ ΓΕΩΡΓΙΟΣ

Επιβλέπων: Δρ. Γεώργιος Φραγκούλης

Καθηγητής

ΚΟΖΑΝΗ/ΟΚΤΩΒΡΙΟΣ/2024

HELLENIC DEMOCRACY
UNIVERSITY OF WESTERN MACEDONIA

FACULTY OF ENGINEERING
DEPARTMENT OF ELECTRICAL &
COMPUTER ENGINEERING

A comprehensive analysis of machine learning algorithms for noisy image classification in astronomy

DIPLOMA THESIS

NTOLKERAS GEORGIOS

SUPERVISOR: Dr. Georgios Fragoulis

Professor

KOZANI/OCTOBER/2024

ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
& ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ ΚΑΙ ΑΝΑΛΗΨΗΣ ΠΡΟΣΩΠΙΚΗΣ ΕΥΘΥΝΗΣ

Δηλώνω ρητά ότι, σύμφωνα με το άρθρο 8 του Ν. 1599/1986 και τα άρθρα 2,4,6 παρ. 3 του Ν. 1256/1982, η παρούσα Διπλωματική Εργασία με τίτλο “Μια ολοκληρωμένη ανάλυση αλγορίθμων μηχανικής μάθησης για την ταξινόμηση εικόνων με θόρυβο με εφαρμογές στην αστρονομία” καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας και αναφέρονται ρητώς μέσα στο κείμενο που συνοδεύουν, και η οποία έχει εκπονηθεί στο Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Πανεπιστημίου Δυτικής Μακεδονίας, υπό την επίβλεψη του μέλους του Τμήματος κ. κ. Γεώργιο Φραγκούλη αποτελεί αποκλειστικά προϊόν προσωπικής εργασίας και δεν προσβάλλει κάθε μορφής πνευματικά δικαιώματα τρίτων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή / και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα. Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και μόνο.

Υπογραφή Φοιτητή: _____

Περίληψη

Η παρούσα διπλωματική εργασία παρουσιάζει μια ανάλυση ορισμένων αλγορίθμων μηχανικής μάθησης για την ταξινόμηση θορυβωδών εικόνων στην αστρονομία. Στην εργασία παρουσιάζονται αλγόριθμοι τόσο με επίβλεψη όσο και αλγόριθμοι χωρίς επίβλεψη και γίνεται εστίαση στις προκλήσεις και τις εξελίξεις αυτών. Η μελέτη επικεντρώνεται σε μια εις βάθος ανάλυση αρχιτεκτονικών βαθιάς μάθησης, όπως τα Convolutional Neural Networks (CNN) και οι Autoencoders, αλλά και σε ανάλυση παραδοσιακών αλγορίθμων μηχανικής μάθησης, όπως τα Decision Trees, τα Random Forests, οι SVM και οι K-Nearest Neighbors. Τέτοιου είδους αλγόριθμοι και κάποιες πιο προηγμένες και περίπλοκες μορφές τους υλοποιούνται με κώδικα σε περιβάλλον Python και δοκιμάζονται στο ίδιο σύνολο δεδομένων. Το σύνολο δεδομένων αποτελείται από θορυβώδεις εικόνες από τους πλανήτες του ηλιακού συστήματος. Τα αποτελέσματα των δοκιμών συγκρίνονται και επισημαίνονται τα δυνατά και αδύνατα σημεία κάθε αλγορίθμου. Η εργασία εξετάζει τις υπάρχουσες μεθοδολογίες και συζητά τις δυνατότητες μελλοντικής έρευνας για τη βελτίωση αυτών των τεχνικών ώστε να βελτιωθεί η ακρίβεια και η αποτελεσματικότητα της ταξινόμησης εικόνων στην αστρονομία.

Λέξεις Κλειδιά:

Μηχανική μάθηση, ταξινόμηση θορυβωδών εικόνων, convolutional neural networks, autoencoders, decision trees, random forests, K-Nearest-Neighbor, SVM, self-organizing maps, αστρονομία, επεξεργασία εικόνας, αποθορυβοποίηση, νευρωνικά δίκτυα, εξαγωγή χαρακτηριστικών, ROC-AUC, μετρικές απόδοσης, scalability, XGBoost, (deep learning), φασματική ανάλυση, ουράνια αντικείμενα (celestial objects), οπτικοποίηση δεδομένων (data visualization), παρατηρησιακή αστρονομία.

Abstract

In this research, a systematic review of machine learning methods for noisy image classification in astronomy is provided. It is aimed at reviewing the developments and issues with different types of machine learning algorithms that include the supervised and unsupervised ones. This work also offers a detailed comparison of both deep learning structures including CNNs and Autoencoders and typical machine learning algorithms including Decision Trees, Random Forests, Support Vector Machines, and K-Nearest Neighbors. These algorithms and some more effective and complex versions of the previously mentioned algorithms are implemented in the Python environment and examined on the same dataset. The dataset contains images that are noisy and are collected from the solar systems planets. The test results are compared and the strong and the weak points of the algorithms are identified and discussed. In this paper, the current methods are analyzed and future possibilities for the development of such approaches to increase the efficiency and reliability of astronomical image classification are considered.

Keywords:

Machine learning, noisy image classification, convolutional neural networks, autoencoders, decision trees, random forests, K-Nearest-Neighbor, SVM, self-organizing maps, astronomy, image processing, denoising, neural networks, feature extraction, ROC-AUC, performance metrics, scalability, XGBoost, deep learning, celestial objects, data visualization, adaptive optics, observational astronomy.

Thanks to

I would like to express my profound gratitude to Professor George Fragoulis, whose guidance and expertise have been pivotal in the development of this thesis. His consistent support, insightful feedback, and encouragement have greatly enriched my academic journey.

I am deeply grateful to my family for their unwavering support, understanding, and encouragement. Their constant patience and faith in my journey have been the foundation of my academic efforts. To my parents and siblings, your love and motivation have carried me through the challenges, and this accomplishment is as much yours as it is mine. Thank you for being my source of strength.

Περιεχόμενα

Shorthand notations, Symbols, and Abbreviations

Shorthand notations, Symbols, and Abbreviations	
AI	Artificial Inteligence
ML	Machine Learning
SOM	Self-Organizing Map
BMU	Best Matching Unit (used in Self-Organizing Maps)
ROC-AUC	Receiver Operating Characteristic - Area Under the Curve
MSE	Mean Squared Error
t-SNE	t-Distributed Stochastic Neighbor Embedding
FP	False Positive
FN	False Negative
TP	True Positive

TN	True Negative
CNN	Convolutional Neural Network
SVM	Support Vector Machine
KNN	K-Nearest Neighbor
RN	Readout Noise
σRN	Sigma Readout Noise
σdark	Sigma Dark Current Noise
σthermal	Sigma Thermal Noise
D	Dark Current (measured in e-/second/pixel)

List of Images

List of Tables

Foreword

This diploma thesis was prepared at the Department of Electrical and Computer Engineering of the University of Western Macedonia, supervised by Dr. Georgios Fragoulis. The research was conducted in the city of Kozani during the academic year 2023-2024. The current work is devoted to the use of machine learning methods for classification of noisy images in astronomy, which is a rapidly developing field due to advanced telescopes that produce large amount of data. This study moves from the classical to the contemporary deep learning techniques in a bid to explain how they may be useful in furthering astronomical study.

Chapter 1: Introduction

1.1 Diploma thesis subject

This chapter will discuss the significance of noisy image classification in astronomy with the view of highlighting the importance of such classification in the analysis and discovery in the field. It is important to point out that astronomical images are usually plagued by noise of various kinds. This noise might become a source of interference and can limit the research process. Consequently, machine learning algorithms are useful in addressing this problem as they help the researchers to obtain useful information from noisy data. This chapter will discuss the development of image classification with machine learning algorithms with emphasis on how the current methods of supervised and unsupervised learning improve the credibility of astronomical data. Last of all, the research questions to be answered in this study will be presented with regards to the performance of various machine learning models in noisy image classification as well as their applicability in enhancing astronomical discoveries.

1.2 Purpose of the bibliographical review and objectives of the diploma thesis

The primary purpose of the diploma thesis is a deep understanding of how machine learning algorithms can recognize and classify astronomical objects in noisy images. By drawing knowledge from the relevant literature on deep learning architectures and algorithms, the diploma thesis aims to give the reader a full understanding of how these algorithms work and the key benefits and differences between them. The main purposes of this review are to consolidate existing research, evaluate methodological approaches, guide the current study, establish theoretical and practical context, and support the development of robust algorithms. Through these objectives, the bibliographical review aims to contribute to the advancement of knowledge in the field and provide a strong foundation for future research and practical applications in astronomy.

1.3 Presentation and structure of the work

The diploma thesis consists of two main sections, the theoretical part and the coding part. The first section focuses on the theoretical aspects of noisy image classification in astronomy. This section covers the key concepts of noise in astronomical images and analyzes modern and traditional machine-learning algorithms for image classification. In the second section, the coding part, the focus is shifted to the practical implementation of the algorithm mentioned in the theoretical part. The algorithms are coded and tested in a Python environment. This part compares their performance on real noisy astronomical images, highlighting the strengths and weaknesses of each approach based on the results obtained from the tests.

Chapter 2: Theoretical Background

This chapter forms the theoretical foundation of this diploma thesis. It focuses on the significant role noisy image classification plays in astronomy and the role of machine learning in addressing this challenge. The problem of noise in astronomical images is introduced and some ways of overcoming it are presented. This chapter also sets the stage for the research questions guiding the thesis. The chapter presents a detailed literature review as it examines previous research on the application of machine learning algorithms—both traditional and modern deep learning approaches—to noisy image classification. Some key contributions and limitations of existing studies are mentioned and context for the following experimental work is provided. Finally, the chapter introduces the evaluation metrics used to assess the performance of these previously mentioned algorithms. Some key performance indicators such as accuracy, precision, and recall are explained and criteria for comparing different machine learning algorithms are established.

2.1 History of Astronomy

This section reviews the evolution of astronomy science throughout humanity. "Astronomy in the Early Years" highlights ancient civilizations' early observations. "Astronomy in the Middle Years" covers significant advancements, particularly from Islamic scholars, and "Astronomy in the Modern Years" discusses breakthroughs like the heliocentric model and the invention of the telescope, shaping today's astronomical science.

2.1.1 Astronomy in the Early Years

Astronomy dates back to ancient civilizations and is one of humanity's oldest sciences. Ancient civilizations used to observe the sky and create calendars for timekeeping and navigation. The early development of the science of astronomy was highly linked and close to religion and mythology [5]. They used to explain phenomena such as eclipses and star formations through mythological narratives. Ancient cultures, including the Chinese, Babylonians, and Egyptians made some of the earliest observations of the sky. These observations formed the basic foundations for later inquiry.

The ancient Greeks played a crucial role in establishing the foundations and the groundwork for modern astronomy too. Thales of Miletus suggested that natural phenomena could be explained by natural causes rather than supernatural ones [6]. Pythagoras proposed that heavenly objects, including the Earth, moved in perfect circles around a central fire. Plato, later on, suggested a geocentric model where the Earth was stationary and celestial bodies moved in concentric spheres around it.

2.1.2 Astronomy in the Middle Years

Astronomy witnessed significant advancements from 500 to 1500 CE. Particularly in the Islamic world, scholars developed trigonometry, constructed observatories and they created scientific instruments like the astrolabe. In Europe, the science of astronomy was closely linked to the Catholic church. They used to create calendars and they supported theological doctories. The translation of Islamic texts into Latin offered them new ideas and introduced them to new methods that shaped European astronomy.

The key advancements during the 9th century and the 15th century are innovations such as improved methods for calculating planetary positions (al-Khwarizmi), the development of trigonometry and astronomical instruments (al-Battani, al-Zarqali), and the proposal of heliocentric ideas (Aryabhata)

2.1.3 Astronomy in the Modern Years

Fast forward, the 16th century was crucial for the development of astronomy and it marked a shift from the geocentric to the heliocentric model, proposed by Nicolaus Copernicus. he suggested that Earth, revolved around the Sun and this model laid the foundations for a completely new era in the scientific field [15]. The heliocentric model was later empirically proved by the invention of the telescope by Galileo Galilei in the 17th century. He observed Jupiter's moons and the phases of Venus. Later, Isaac Newton's laws of motion and gravitation offered an explanation to many terrestrial phenomena, further advancing the field.

The 19th and 20th centuries saw improvements in observational methods, the development of spectral analysis, and the formulation of the Big Bang theory which extended our understanding of the universe. The beginning of the space age happened with the launch of Sputnik 1 in 1957 leading

to unprecedented exploration, including revolutionary observations by the Hubble Space Telescope in the 1990s.

In this 21st century, advancements such as the James Webb Space Telescope and the Atacama Large Millimeter/Submillimeter Array (ALMA) have widened the horizon for astronomy allowing astronomers to search for exoplanets as well as delve deeper into cosmic mysteries [55]. Imaging technology experienced a great leap forward that enhanced clarity in previously noisy astronomical pictures via things like adaptive optics and bigger cameras [20].

2.2 Overview of the importance of noisy image classification in Astronomy

With reference to astronomical imaging, noise can be found in many forms like atmospheric, telescope or detector noise which causes data distortion, hence degrading its quality. Noisy image classification, which separates relevant information from artifacts, is essential in enhancing the accuracy of astronomical observations.
Some of these advantages of noisy image classification are:
Identifying Faint Objects: Aids in discovery and observation of faint objects which are hardly visible such as other stars, or galaxies thus helps in making new theories and validate existing theories.
Exoplanet Detection: Helps locate exoplanets with the help of signals that are emitted from the host stars in the form of light.
Studying the Early Universe: It helps in the study of faint leftovers that include afterglow observed from the earliest stars and galaxies to form after the Big Bang occurrence.
Searching For Dark Matter: It helps in identification of the dark matter through the influence it exerts on other structures out in the universe.
The reduction of noise enhances the quality of data hence enhances the credibility and authority of scientific research by enhancing the aspect of analysis, identification of objects as well as sensitivity in detecting.

2.3 Explanation of the role of machine learning algorithms in this field

Artificial intelligence (AI) is the development of computer systems that are capable of raising their performance with experience. Originally derived from discipline such as computer science and mathematics, statistics, philosophy, biology and much more, AI has now found tremendous applications in a wide range of fields, with applications in areas like computer vision, language processing, and diagnosis.
Therefore, in the context of AI, Mashine Learnign (ML) can be defined as a subfield that aims at creating models that can upgrade their functionalities with time without the help of the programmer. While AI is the ability of machines to think and act like humans (like solving a problem, making a decision), ML is the way on how these intelligent systems learn from information that they obtain from their environment and improve upon it.
In astronomy, the Data Science technology feeds large data to AI models so as to help them learn from the data to pick out patterns for further and more accurate prediction in the field.

2.3.1 AI and ML: From the early days to Modern history

Machine Learning has numerous similarities to the broader field of AI especially at its introduction. Artificial intelligence and machine learning can be dated back to mid of the mid-twentieth century [29]. The term Artificial Intelligence was coming into use at Dartmouth Conference in 1956 which was organized by John McCarthy, Marvin Minsky, Nathaniel Rochester, Claude Shannon [8]. This

was an event to create machines that could think like human beings and created Artificial Intelligence as a field different from control theory, decision theory and mathematics.

At the beginning of the AI studies, researchers aimed at creating methods applicable for solving fundamental problems using primitive logic. It is for this reason that these stated “weak methods” failed to effectively perform optimally especially in handling of complex tasks, hence the need for stronger methods. In regards with these criteria, one elaborated example is the DENDRAL program developed in 1969 [28], which is aimed at inferring molecular structures based on spectrometer data. K-D/ss was the first effective program for handling activities that involve the use of intensive knowledge, the application of a restricted number of rules toward arriving at solutions to narrowly-defined problems.

Further attempts including the Heuristic Programming Project (HPP) at Stanford. This resulted into the development of MYCIN which was a program that could diagnose blood infections with the same efficiency as the human experts. In the field of linguistics, Terry Winograd has introduced SHRDLU program that can understand the language in some given surroundings.

The commercial application helped to promote the expert systems industry from million to billion by 1988. All these emerged to become what is modern AI and ML.

2.3.2 AI Winter

A lot of issues arose and thus hindered the development of AI ; this was as a result of over enthusiasm on what could be done using the existing technology. First of all, scientists were able to solve simple tasks with reference to elementary steps while they could not solve frighteningly complicated tasks. The lack of understanding of how computational complexity works meant that people quickly realised that bigger problem could be easily solved with more memory and higher speed hardware. However, the real difficulties were in such areas as the absence of efficient algorithms, not enough data available, and in the ways of encoding the knowledge of the human being into a format understandable to the computers. First applications operated with predefined set of rules that were very much intricate and took considerable amount of time to generate, thus created constraints in front of AI to address more complex problems.
Also, the industry shifted from generating profits due to unrealistic promises by firms that developed artificial intelligence, which discouraged investors and funding for development of such advanced intelligence. This situation is referred to as the ‘AI Winter’ by evolving high expectations for AI into low research activity networks.
The fact that proponents of the field argued that only better hardware was needed to achieve the next step in progress was later dismissed. It was identified that for AI, the enhancement of the programs it was necessary to take changes in an iterative natures, and against this improving algorithms were studied. Later methods, which include genetic algorithms used even better representations to make more improvements than the early methods which laid foundation on the current modern machine learning techniques.

2.3.3 Modern Years of AI, the Rise of ML

The emergence of Machine Learning (ML) is usually attributed to the improvement of the learning algorithms in the early 1980s especially the backpropagation algorithm, which made it possible to train the multi-layer neural networks. As the computers became stronger and became globalized, with better data availability especially from the mid 1990s, the ML algorithms could build on large datasets that would improve their training mechanisms. This increase in data was due to the Big Data phenomenon that offered a bounty of learning data from different source such as social media, e-commerce and sensors.

Currently, the use of ML is well incorporated into several applications such as virtual personal assistants, self-driving cars, translation, credit card fraud detection, medical image diagnosis among others where a physician or a doctor uses an ML model for analyzing patients' data and images for better diagnosis.
Today we are observing further development of AI, ML and astronomy which are intertwined with each other. AI has been widely used in astronomy especially in the analysis of images where it had helped in documentation and classification of the objects in the space. This synergy has made new possibilities in noisy image classification as modern telescopes observe more and more detailed and intricate astronomical scenes that could not be analyzed by conventional methods [37].
A remarkable case, related to AI and ML applications in astronomy, is the improvement of the first black hole image ever taken and unveiled in 2019. On the similar context, with the help of PRIMO, an AI and ML algorithm that was taught using simulations of black holes, these astronomers were able to reduce image distortion and improve the image resolution with a notable revelation on the potential of AI and ML in the future space exploration [54].

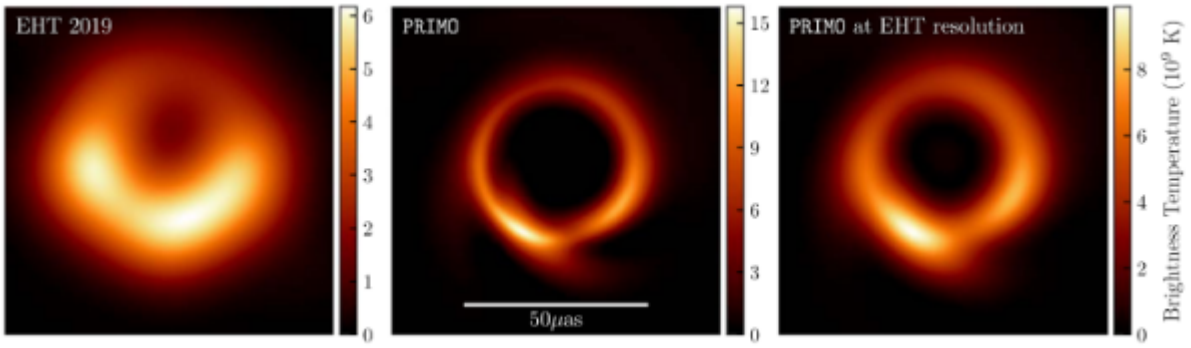


Image 1
https://iopscience.iop.org/article/10.3847/2041-8213/acc32d/pdf

2.4 Research Questions

The use of machine learning has provided revolutionary approaches in different field of science of astronomy. The goal of this study is to identify the challenges that may arise while attempting to classify astronomical images through the help of machine learning algorithms because different forms of image data have their reservations. This study aims at comparing the accuracy of diverse Machine Learning methods in the subject of Astronomy image classification. It tries to uncover new knowledge that may help improve models and techniques in automating image analysis as well as aid in the furtherance of astronomical studies under diverse conditions and arrangements of algorithms.
The following research questions have been formulated to guide the investigation and ensure a comprehensive analysis of the capabilities and limitations of machine learning algorithms in the classification of astronomical images:
Algorithmic Efficiency and Circumstances that Enable Appropriate Application
The purpose of the given paper is to reveal which machine learning algorithms are the best for the classification of astronomical images as well as what factors might affect the algorithms' work in the best possible way. The case arrises in an attempt to demarcate the algorithms' performance in terms of some parameters like the resolution, signal to noise ratio, and the types of astronomical objects depicted in the images.
A Review of Deep Learning Approaches with Conventional Methods
This work presents results of astronomical image classifiers made with deep learning methods and compares their performance with traditional machine learning models. Based on the following research question of interest, this inquiry is inclined at comparing and contrasting neural network

with other machine learning algorithms in order to establish which of the two methods delivers the best results.

Chapter 3: Methodology

In this chapter, the methodology and strategies used for the bibliographical research will be discussed.

3.1 Databases and search engines used

The Databases and search engines used for the bibliographical research consist of:

Google Scholar: https://scholar.google.com/

ResearchRabbit: https://www.researchrabbit.ai/

ResearchGate: https://www.researchgate.net/

ScienceDirect: https://www.sciencedirect.com

ConnectedPapers: https://www.connectedpapers.com/

ScienceDirect: https://www.sciencedirect.com

3.2 Keywords and search terms

The keywords and search terms used for the research are:

Keywords:
Machine Learning, Noisy Image Classification, Astronomy, Convolutional Neural Networks (CNNs), Autoencoders, Decision Trees, Random Forests, K-Nearest Neighbors (KNN), Support Vector Machines (SVMs), Deep Learning, Image Processing, Denoising Techniques, Celestial Object Detection, Data Visualization, Feature Extraction, ROC-AUC (Receiver Operating Characteristic - Area Under the Curve), Performance Metrics, Scalability of Algorithms, Adaptive Optics, Observational Astronomy, Supervised Learning, Unsupervised Learning, Noise Reduction, Image Enhancement, Astronomical Imaging

Search Terms:
"Machine learning algorithms for noisy image classification", "Application of CNNs in astronomy", "Denoising techniques in astronomical image processing", "Comparison of deep learning and traditional methods in image classification", "Use of Autoencoders for astronomical data", "Support

Vector Machines for noisy data analysis", "Decision Trees and Random Forests in astronomy", "Machine learning for celestial object detection", "Feature extraction from noisy astronomical images", "Performance metrics for image classification algorithms", "Advanced machine learning methods for astronomy", "Astronomical data visualization using machine learning", "Scalability challenges in machine learning for big data", "Adaptive optics and machine learning applications", "Supervised vs. unsupervised learning in noisy image classification", "Noise reduction strategies in astronomy", "Improving image enhancement using deep learning", "Observational astronomy and artificial intelligence", "Evaluating machine learning algorithms for astronomy data", "Future directions in machine learning for astronomy".

3.3 Inclusion Criteria and Exclusion Criteria

For this research, 2 types of criteria were used.

Inclusive criteria:
Relevance to Machine Learning for Noisy Image Classification: Studies that specifically focus on the application of machine learning techniques (such as CNNs, Autoencoders, Decision Trees, Random Forests, KNN, SVMs, etc.) for noisy image classification in astronomy.
Use of Astronomical Data: Research that utilizes real astronomical data, including data from telescopes, space missions, and other observatory sources.
Publication Date Range: Articles and studies published within the last 10 years to ensure the inclusion of the most recent developments and techniques.
Peer-Reviewed Sources: Studies published in peer-reviewed journals, conference proceedings, or reputable scientific databases.
English Language: Only studies published in English to ensure clarity and comprehensibility.
Clear Evaluation Metrics: Research that provides specific performance metrics (e.g., accuracy, precision, recall, F1 score, ROC-AUC, mean squared error) when the machine learning algorithms are applied.
Application to Astronomical Challenges: Research that addresses specific astronomical challenges.

Exclusion Criteria:
Irrelevant Focus Areas: Studies that do not involve machine learning or are not specifically focused on noisy image classification within astronomy.
Outdated Research: Articles and studies published more than 10 years ago unless they are foundational works that provide essential background information.
Non-Peer-Reviewed Sources: Research from non-peer-reviewed journals, non-scientific websites, opinion pieces, or articles without empirical data.
Non-English Publications: Studies published in languages other than English, to avoid potential issues with translation accuracy and interpretation.
Lack of Evaluation Metrics: Studies that do not provide clear evaluation metrics or performance results for the algorithms used.
Focus on Other Domains: Research primarily focused on image classification in domains other than astronomy (e.g., medical imaging, satellite imaging for earth observation) unless they present universally applicable techniques.
Theoretical Papers Without Practical Application: Studies that are purely theoretical and do not demonstrate practical application or empirical results for noisy image classification.

Chapter 4: Fundamentals of Noisy Image Classification

This section offers a foundational understanding of the concepts and methods used for handling images affected by noise, a frequent challenge in astronomical data analysis. The origins of noise are presented as their impact on image quality, as well as the machine learning techniques designed to improve the precision and reliability of image classification under noisy conditions.

4.1 Overview of astronomical imaging and common sources of noise

Imaging in astronomy is the basis of astronomy as a discipline since it involves observations, captures, and studies of the celestial objects and phenomena. It uses telescopes, detectors and techniques to collect useful data that helps us in improving our knowledge concerning the universe.

Types of Telescopes for Imaging:

- Ground-based Telescopes: For instance, W. M. Keck Observatory in Hawaii where there are two telescopes each with a primary mirror of ten meters, the largest optical and infrared telescopes in the world [43].
- Space-based Telescopes: ASTROMETRIC STRATEGIES ASC COMP oversee such influential observatories as Hubble Space Telescope (HST), launched in 1990, which is attributed to extraordinary flexibility and extraordinary impact upon astronomical science [26].

These telescopes employ different tools such as CCDs and PMTs in the gathering of light signals, spectrometers in the analysis of light, and adaptive optic to enhance image acuity.

Image 2. Twin telescopes Keck I and Keck II with their ‘eyes’ open. Credit: Ethan Tweedie https://www.keckobservatory.org/media/photos/

On the other side, the most well-known space-based telescope is the Hubble Space Telescope [26]. The Hubble Space Telescope (often referred to as HST or Hubble) is a space telescope that was

launched into low Earth orbit in 1990 and remains in operation. While it was not the first space telescope, it is one of the largest and most versatile, renowned as a vital research tool for astronomy.

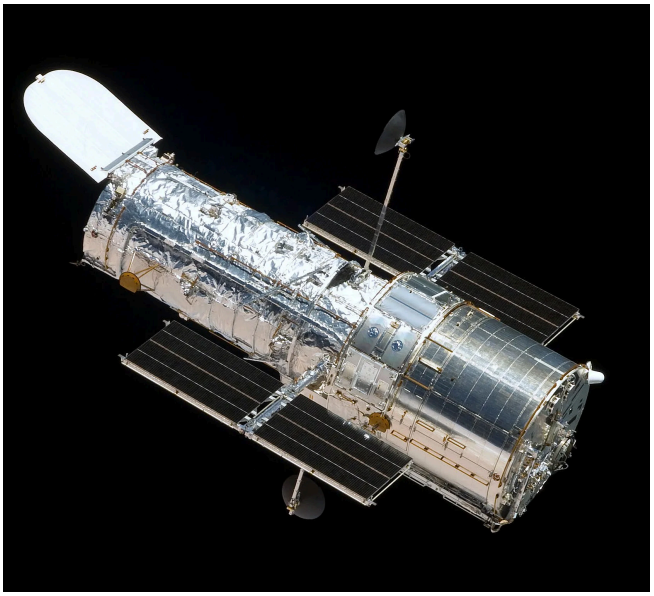


Image 3. The Hubble Space Telescope floats against the background of space on May 19, 2009, after being released by the space shuttle Atlantis following the repairs and upgrades of Servicing Mission 4. Above the Earth's atmosphere
https://science.nasa.gov/mission/hubble/overview/why-have-a-telescope-in-space/

Imaging Techniques:

- Optical Imaging: Collects visible light emission from astronomical objects and gives information about structural and compositional properties of these objects[42].
- Radio Imaging: It detects radio waves with the help of two or more antennas by forming pictures of radio sources such as molecular clouds and distant galaxies[53][57].
- Infrared Imaging: Its observing technique collects infrared radiation from objects that emit little visible light such as cool stars and star-forming regions [31][32][33].
- X-ray Imaging: Detects high-energy X-rays belonging to such subjects as black hole's accretion disks and supernova remnants(for instance Chandra X-ray Observatory)[53][58].

Sources of Noise in Astronomical Imaging:
Noise makes the process complicated for astronomers since noise in images can further diminish the quality of image and accuracy as well. To consider sources and kinds of noise an explanation of how camera mechanisms and photons work will be presented below.
CCD Camera Functionality: CCD cameras are what is known as silicon semiconductors in which structures are designed to form localized electric traps, or pixels where photons are absorbed and and pop an electron up into the conduction band. Electrons built up in each pixel are converted to an analog digital number or Data Number or DN by an analog to digital conversion (ADC) unit. ADCs found in most contemporary systems operate within the 16 bit range with a range of 0 to 65,535 which provides a high dynamic range.
Key Concepts in Noise and Imaging:
e-/DN Conversion: The number in each pixel is calculated based on photo-electrons detected (e-) based on the factor known as Inverse Gain (G) expressed in e-/DN. This one defines how that signal is translated from electron back to the digital numbers.
Inverse Gain (G): A proportional factor that express photo-electrons in digital form. It is chosen according to the trade off between; dynamic range and read noise.

$G = \frac{Full\ Well\ Capacity\ of\ Pixel}{ADC\ Saturation}$, where ADC refers to the Analog to Digital Converter.
Interference can be in form of noise coming from atmospheric turbulence, electronic interference, thermal noise and these interfere greatly with image quality. Thereby, cognition of these sources and tuning the inverse gain reduces the deteriorating influence, and, thus, results in clearer and more accurate astronomical observations [59].

4.1.1 Common sources of noise

Astronomical imaging is known to be contaminated by different types of noise that have a detrimental impact on image quality and on the possibility to accurately measure objects of interest. Here are some common types of noise encountered:
Shot Noise (Poisson Noise) [12]: Is a consequence of statistic nature of light which leads to a random variation of a signal, and therefore its strength, from one pixel to another [4]. Besides, it appears when observing an object in a low illumination mode or observing faint objects [82]. The standard deviation of shot noise follows a Poisson distribution:
Poisson distribution's standard deviation is $\sigma = \sqrt{N}$, where N is the number of detected events [59].

Image 4. Shot noise simulation. The number of photons per pixel increases from left to right and from upper row to bottom row.
https://en.wikipedia.org/wiki/Shot_noise

Other common types of noise include:
Gaussian Noise: Gaussian noise, also called normal noise, is caused by natural sources such as the thermal vibration of atoms and the discrete nature of radiation of warm objects. Gaussian Noise is named after the mathematician Carl Friedrich Gauss and it is a statistical noise having a Probability Density Function (PDF) equal to that of the normal distribution, which is also known as the Gaussian distribution. In simpler terms, considering the values of all pixels in an image affected by Gaussian noise, they follow the familiar bell-shaped curve of the Gaussian distribution. Mathematically, Gaussian Noise is described as:

$$P(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\, e - \frac{(x-\mu)^2}{2\sigma^2} \quad ,$$

where P(x) is the probability density of a pixel having a value x, μ is the mean of the distribution, representing the average pixel value, σ is the standard deviation, indicating how much the values deviate from the mean.
In the context of an image, μ often corresponds to the original pixel value, and σ represents the degree of noise. The noise is randomly distributed throughout the image and the amplitude of Gaussian noise is independent of the pixel value. This means that both bright and dark areas of an image are equally affected. Gaussian noise typically manifests as a random, grainy texture superimposed over an image, affecting its clarity. It's most noticeable in areas of uniform color or brightness, where the variation introduced by noise is more apparent [10].

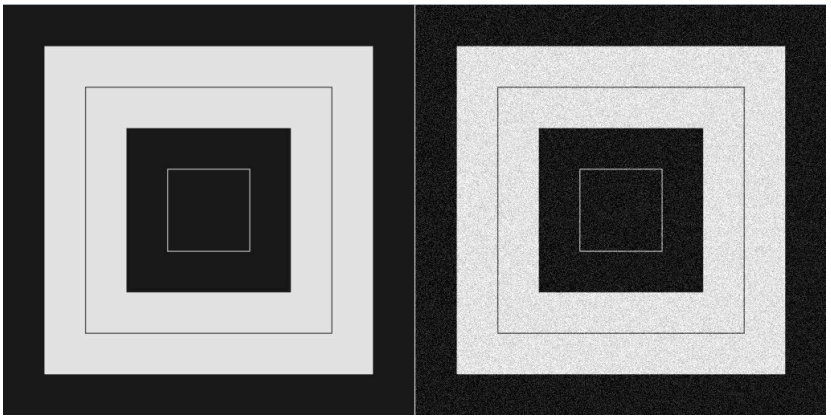


Image 5. The left image is without noise, right image is with gaussian noise
https://en.wikipedia.org/wiki/Gaussian_noise

Salt and Pepper Noise (Impulse noise): Salt and Pepper Noise, also known as Impulse Noise, is a type of noise that is characterized by sudden disturbances in the image signal. It manifests as sparsely occurring white and black pixels, hence the name. Impulse noise can be mathematically modeled as a process that affects some pixels with an intensity different from their original value. The noise can be represented as:

$$I(x) = s_{min} = 0, \; with\, propability\, p/2$$

$$I(x) = s_{max} = 255, \; with\, propability\, p/2$$

$$I(x) = f(x,y), \; with\, propability\, 1 - p$$

where f(x) is I(x) is the intensity of the noisy pixel at position (x,y), f(x,y) is the original intensity of the pixel and p is the probability of a pixel being contaminated with noise.
In this type of noise, each pixel in the image has a probability p of being affected by noise. If affected, the pixel intensity is set to either 0 (black) or 255 (white) for an 8-bit image, hence creating the 'salt and pepper' appearance. The noise appears randomly scattered throughout the image and the affected pixels take on extreme values, usually the maximum or minimum within the image's dynamic range. Typically, only a small percentage of pixels are affected, but they are highly noticeable.

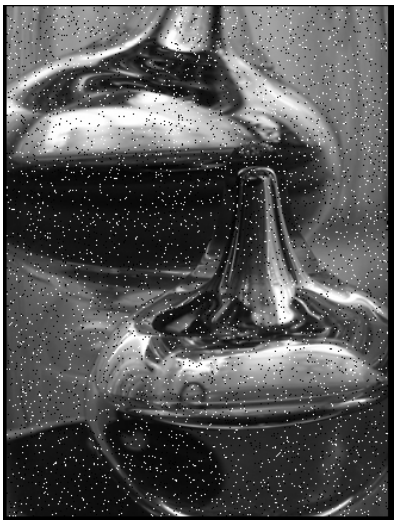


Image 6. An image with salt-and-pepper noise
https://en.wikipedia.org/wiki/Salt-and-pepper_noise

Readout Noise (RN): This noise originates from the electronic readout process of the image sensor. It is a constant level of noise added to each pixel while converting the analog signal to a digital one. Readout noise is independent of the detected signal and contributes to the overall noise floor of the image. The best way to defeat read noise is to ensure that exposure is long enough during a single exposure that the faintest feature wanted to record has a signal well above the read noise. In many cameras and electronics systems, the read noise can be as low as a few e-. The way RN is defined implies that (RN)2 readout e- were measured and within the aperture of radius r the noise will:
$\sigma RN = \sqrt{pi \times r^2 \times RN^2} = \sqrt{npix \times RN^2}$,
where npix is the number of pixels in the aperture [59].

Dark Current Noise: Dark current noise comes from thermal energy present in the image sensor. Background signal is created within the detector due to generation of electrons even in the absence of light, which gives noise to the image. Dark current noise is typically observed during sample integration for long duration or at high ambient temperature. For a dark current of D in e-/second/pixel, the noise in the aperture is simply:

$\sigma dark = \sqrt{D \times npix \times t}$ [59].

Thermal Noise: Thermal noise is due to random fluctuations in voltages that include circuits and other apparatus in the imagining system. These fluctuations are responsible by thermal energy in the system. Thermal noise may influence the measurements' precision and cause unnecessary signal interferences.
Cosmic Ray Noise: Cosmic rays consisting of high-energy particles from space, travel and make impacts on the image sensor where they can cause pixel-level disruptions. These disturbances appear as pin-like bright regions or streaks in the image, and may be difficult eradicating from the background hence a great challenge during data reduction phase [25].

4.2 Explanation of common techniques and types of machine learning algorithms for image classification

There are several common techniques used for image classification. These common techniques are generally divided into two categories, supervised and unsupervised.

4.2.1 SUPERVISED

The supervised machine learning methods employ training data set to establish how different measurable attributes are related to a certain variable of interest. When trained, a supervised algorithm is capable of predicting the target variable on new unknown data set. Supervised algorithms are often confused with other traditional model-fitting algorithms. The key difference is that in model-fitting algorithms, the model is pre-defined while supervised learning algorithms construct the model from scratch according to the input dataset [93].

In supervised learning the data is partitioned into objects and the feature with the target variable or label is associated to every object. This can be either continuous variables such as age and income or else they can be discrete variables such as gender, car brand, etc. Feature selection, preprocessing, and transformation strategies must match the algorithm in consideration during modeling. Dependent variable of the process or Y variable depicts the aim of the process that one wants to achieve; it may be Categorical, Nominal – classification type problem such as spam detection, or interval – regression type problem such as prediction of house prices.

Supervised models use parameters derived from the data for the model to make recommendation and hyperparameters which are adjusted by the practitioner to influence issues such as complexity and optimization. For the results in cross-validation, it may not perform very well when it comes to the actual datasets it hasn't had the chance to 'see' before.

The supervised learning process involves three stages: training, validation and testing set. In the variant used during training the algorithm gets problem-solution pairs as inputs and uses the parameters to accommodate for errors. The validation stage optimizes hyperparameters in order not to be biased and use the validation set. Lastly, the model is tested with the test set meaning that we are implementing this model when faced with real datasets that are unknown to it [39].

There are the main groups of the supervised algorithms such as Support Vector Machines (SVM), Decision Trees, Random Forests, and Artificial Neural Networks.

4.2.2 Support Vector Machine

A commonly used supervised learning algorithm is Support Vector Machine (SVM) where the main idea is to determine a hyperplane in order to split classes in N-dimensional space while at the same time, the distance between the closest points to the hyperplane is at the maximum [2][51]. Margin is relative between the hyperplane and the nearest points called supports vectors. In a two dimensional space, the hyperplane is just a line drawn on the plane which divides the plane into classes. SVM categorizes new data by placing it into the plane with this unique hyperplane.

When the classes are not separable in the original space, SVM utilizes kernel method which brings the data in to another dimentional space where it is easy to classify them. Common kernel functions include:

Polynomial Kernel: Translates data to environments where polynomial relationship can be computed.

Sigmoid Kernel: The sigmoid kernel introduces sigmoidal transformations to the data, enabling SVM to capture non-linear patterns that resemble sigmoid functions.
Radial Basis Function (RBF) Kernel: Changes the data to a space of almost countably infinite dimensions and thus can address higher-order non-linear relationships.
These kernel functions are amongst the hyperparameters of an SVM model and are known as kernel functions.
SVM is most useful in the higher dimensional space therefore, is the best technique while handling larger data sets especially in image processing. They are also stable, which is most of the time they are not very sensitive to over fitting when they work with data points that are close to the decision line. Nonetheless, as pointed out in the following sections, SVMs have the following drawbacks. It is pertinent to note that, these methods are sensitive to noisy data and outliers, which may distort the hyperplane. Also, SVM requires much parameter adjustment including selection of kernals and hyperparameters and this makes the modeling process difficult.

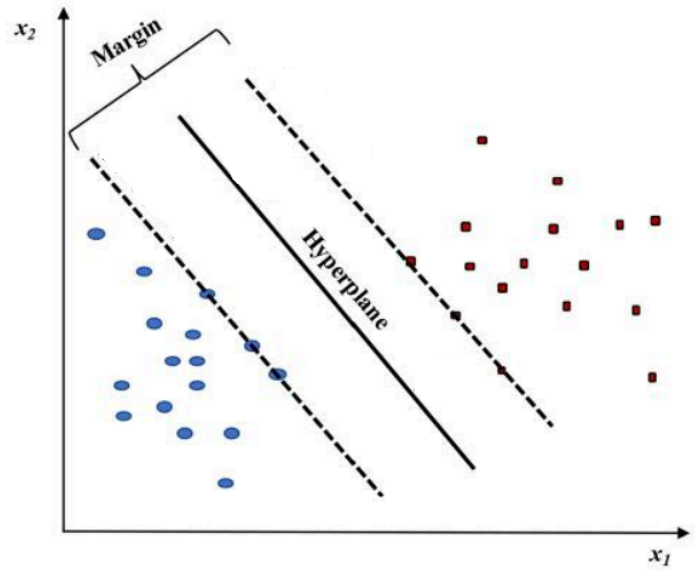


Image 7. Linear SVM model. Two classes (red versus blue) were classified.
https://cgp.iiarjournals.org/content/cgp/15/1/41.full.pdf

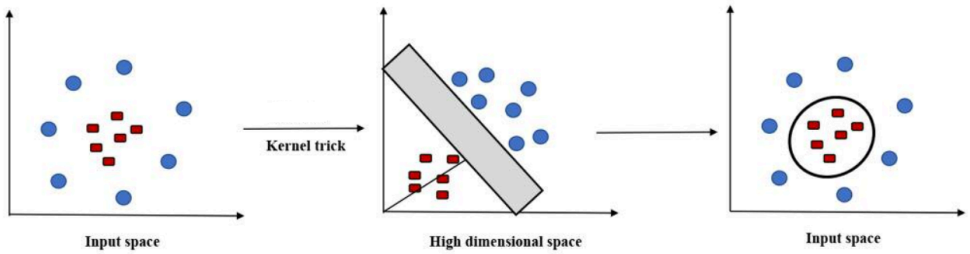


Image 8. Kernel function. Data that cannot be separated by linear SVM can be transformed and separated by a kernel function.
https://cgp.iiarjournals.org/content/cgp/15/1/41.full.pdf

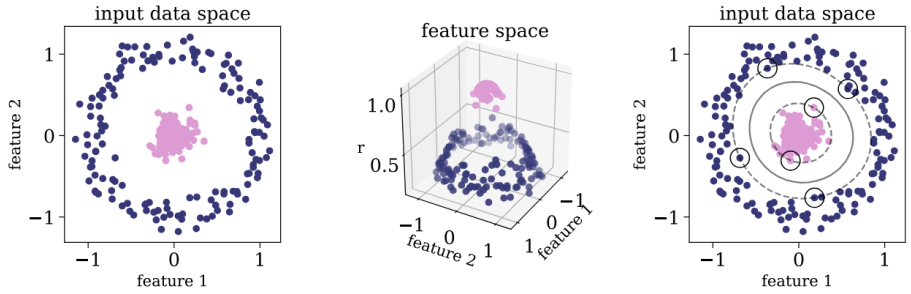


Image 9. Application of the SVM kernel trick to a two-dimensional dataset that consists of two classes which are not linearly-separable. The left panel shows the dataset, where the different classes are represented by pink and purple circles. The middle panel shows the three-dimensional feature space that resulted from the applied mapping, where the classes can be separated with a two-dimensional hyperplane. The right panel shows the result of back-projecting the decision boundary to the input space, where the support vectors are marked with black circles, and the decision boundary with a solid gray line.
https://arxiv.org/pdf/1904.07248

4.2.3 Decision Trees and Random Forest

Decision Trees and Random Forests [41][47] are basic Supervised Machine Learning techniques which are employed in classification and regression. Decision Tree is a tree structure where every internal node contains one feature, branch corresponds to a possible value of a feature and leaf node represents a prediction or a value of a variable. The tree is constructed iteratively, choosing the best feature to split the data set, utilizing indexes of purity such as Gini's or entropy for classification or mean squared error for regression. It stops when a predefined termination condition such as the depth of tree or the minimum sample size per node is achieved. Decision Trees can be well interpreted, used for mixed data: categorical and numerical; have few hyperparameters and are suitable for Big Data. However, they tend to over fit and are very dependent on changes in the data hence possessing instability.

Random forest is the composition from Decision Trees, and each of them is built using prespecified subdataset and prespecified features. This randomness of selection of trees avoids over fitting of the model and guarantees that it can detect a wide variety of patterns. After the tree construction, the output of all trees are then aggregated by a voting system. In a classification output, the majority class is taken for the final class label and in regression, the mean of output of various trees is considered as the final output. This method of learning brings down the bias as well as the variance contributing towards stability of the model for new datasets.

Subparameters of tuning Random Forests include the number of trees as well as the number of features in each node [46]. Random Forests are stable, useful and can be used for both classification and regression, and it also has a solution to the problems of overfitting in single Decision Trees.

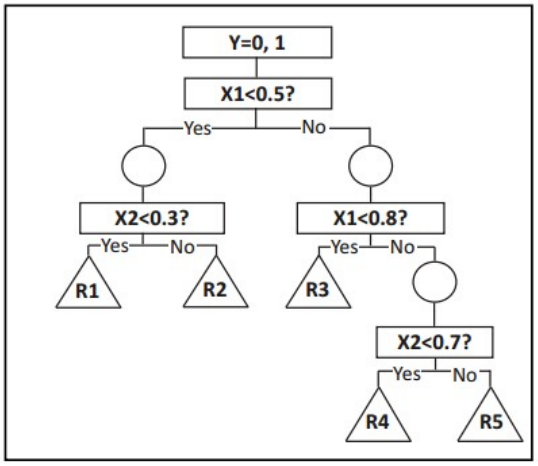


Image 10. Sample decision tree based on binary target variable Y
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4466856/

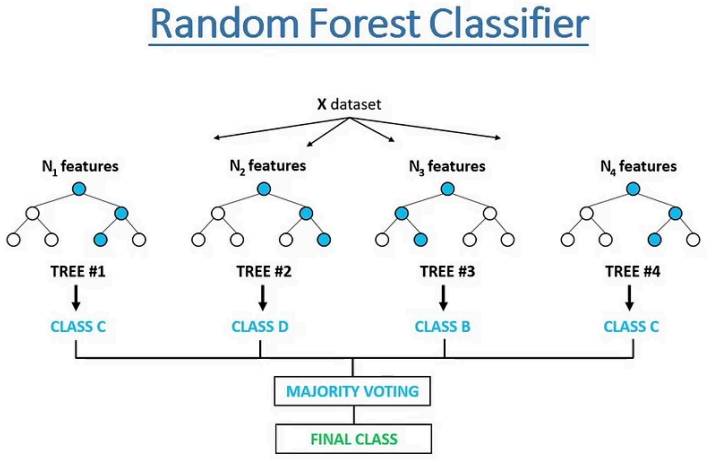


Image 11. Random forest Classifier
https://medium.com/@mrmaster907/introduction-random-forest-classification-by-example-6983d95c7b91

4.2.4 Artificial Neural Networks

Artificial Neural Networks (ANNs) [23][34] are the models derived from principles of biological nervous systems. They have changed areas such as image/speech recognition, diagnosis of diseases, and self-driving cars. Common types of ANNs include:
Feedforward Neural Networks (FNNs): A model which has one or more than one hidden layer that takes the input data and processes it and sends out the results which are the output of the model mainly used in regression and classification.
Convolutional Neural Networks (CNNs): Well suited for image and video analysis for the reason that they learn the spatial features and can be applied in image recognition as well as an object detection.
Recurrent Neural Networks (RNNs): Process sequential data in terms of loops, and thus applicable to time series, natural language processing, and speech recognition.

Generative Adversarial Networks (GANs): Consists of a generator and a discriminator used in generating new instances especially in image synthesis and data augmentation.
In this section, shallow neural networks will be described. Such networks contain an input layer, one or more hidden layers and an output layer. Inputs go through a layer-wise process where first, each neuron calculates the weighted sum of the inputs from the previous layer after which applying a non-linear activation function such as sigmoid, tanh or ReLU. Forward propagation is the process of making estimations while backpropagation is the learning phase whereby weights and biases are optimized through a gradient descent method whose aim is to reduce the level of error defined by a loss function.
Hyperparameters that require selection for the neural network model are the activation function, the number of hidden layers, and the number of neurons per layer. Neural networks are as such are flexible and nonlinear in nature that makes them capable of mapping complex relations between the input data and target variables. They are also versatile, that means, they can accommodate structured data, unstructured data as well as sequential data.
However, ANNs heavily rely on large datasets which have been labeled and such data may be costly or time-consuming to develop. They also usually overfit, at least with small amounts of data at hand, memorizing noise instead of generating general knowledge for use on fresh data.

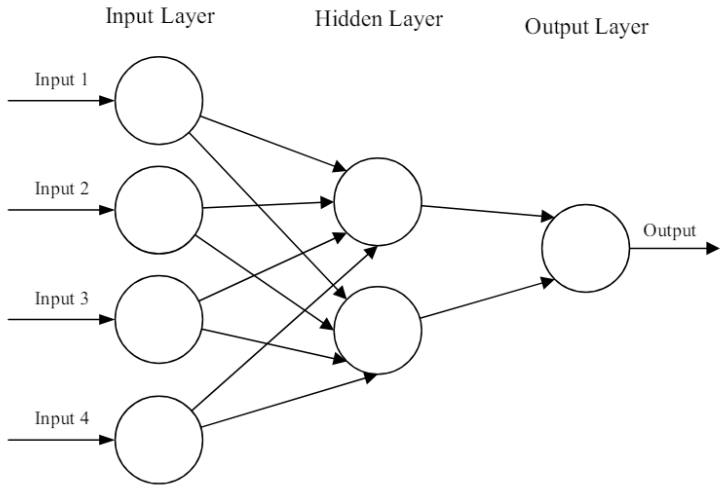


Image 12. A simple three layered feedforward neural network (FNN), comprised of a input layer, a hidden layer and an output layer. This structure is the basis of a number of common ANN architectures
https://arxiv.org/pdf/1511.08458

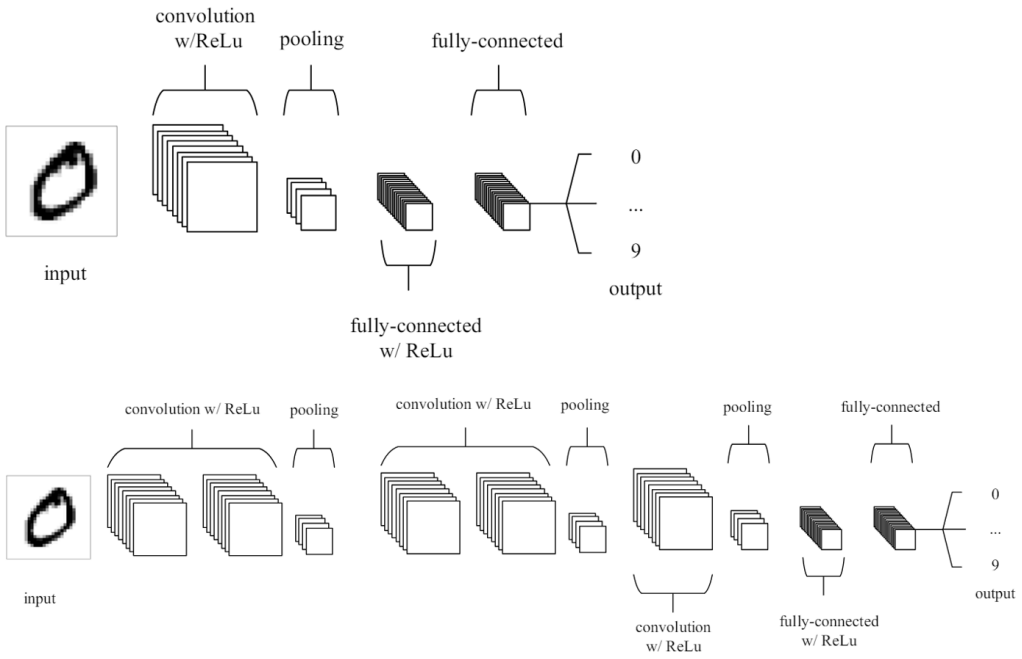


Image 13. An simple CNN architecture, comprised of just five layers(top)
A common form of CNN architecture in which convolutional layers are stacked between ReLus continuously before being passed through the pooling layer, before going between one or many fully connected ReLus.(bottom)
https://arxiv.org/pdf/1511.08458

4.2.5 UNSUPERVISED LEARNING

Unsupervised learning can be defined as a particular type of ML that deals with data analysis with no predetermined results to discover the structure. Unsupervised learning, on the other hand, is especially applicable to unlabelled or unstructured data and thus appropriate when labelling is impossible. These can be mentioned as clustering, dimensionality reduction, visualization and outlier detection, which are considered to be quite effective in scientific research to analyse data and identify new trends and affecting factors [94].
Unsupervised learning algorithms work independently on the features and do not use any “ground truth”. This means that their output is normally in face of functions mapping from the input space to clusters, low-dimensional representations or lists of anomalous events. These algorithms are based on internal decisions and cost parameters that define them and their actions. Furthermore, they have other factors which are exogenous; this means that they can influence the output, and hence having different version of the same dataset. Consequently, paradigmatic interpretations of the results of unsupervised learning need to consider the vulnerability in terms of parameter variation and decisions in the algorithms.
Despite the fact that this type of learning does not have the goal of maximizing an objective, it is particularly suitable for exploratory data analysis. That is the situation where the goal of the learning process is not an optimal value of some variable but the discovery of some structures in the data.

4.2.6 Distance Assignment

Distance assignment is one of the basic components for many unsupervised learning methods, especially for the clustering and methods, based on nearest neighbors. It is used in most unsupervised algorithms as it entails trying to find how similar the data elements are so as to try to cluster and make the best decisions. When allocating the distance between two objects then the most frequently used method is the Euclidian Distance function between the two objects. It is the distance between two points in straight line in space that has been defined and measured using Euclidean Geometry. It is very easy and indeed very useful and this is why it is popular for continuous data. Another well known distance metric used is the Manhattan Distance. The Manhattan distance is the sum of the absolute values of the difference between two points' coordinates. It is applicable for grid based data or when movement is confined only in orthogonal directions. The final metric is the Cosine Similarity Distance where we work with vectors since this method allows calculating cosine of the angle between two vectors. It is applied in text analysis and collaborative filtering commonly.

4.2.7 Clustering Algorithms

Clustering is a key concept of a number of unsupervised machine learning algorithms. It is a method of classifying the data in sets depending on specific characteristics or features that they have. The main purpose of clustering is to find patterns or clusters in the given data set, as these give direction about the natural organization and density of the data. The clustering algorithm tries to establish groups or clusters that are compact or dense in terms of clusters' attributes and features and with very low inter-cluster density. The aim is to reduce the 'intra-cluster variation' which refers to the distance between the points within the same cluster and increase the 'inter-cluster variation' which means distance between the points in two different clusters. In fact the definition of what constitutes a cluster differs from one algorithm to the other. It is clear with the two most popular approaches named the Centroid-Based Clustering and the Hierarchical Clustering. The Clustering approach under consideration is centroid-based which means that a set of certain number of cluster centroids has to be chosen while each point has to be assigned to the nearest cluster centroid. K-Means is one of the most recognized methods of centroid-based clustering technique. The other type is hierarchical clustering that forms clusters in groups and subgroups through merging or splitting of the existing groups. As a type of clustering approaches, there are two categories which are agglomerative and divisive.

4.2.8 K-means

K-Means [1][35][48] is one of the most famous and easy to implement clustering algorithm of unsupervised machine learning. It is a kind of algorithm that creates clusters in a certain number of clusters with every data point with the closest mean (centroid). The scope of application of the identified algorithm is also wide, it is suitable in high-dimensional space and works best when the number of clusters is predetermined or can be determined [7]. However, K-Means will always produce clusters for all of them even if the dataset is devoid of well-defined clusters hence proper precaution has to be observed.
The algorithm involves the following steps:

Selecting the number of clusters: The number of clusters (k) needed to provide the best results for the grouping of consumers have to be determined.
Selecting k data points from the dataset as initial centroids randomly to divide the dataset in to k clusters.
A process of categorizing each data point to the cluster with the nearest mean or centroid usually computed employing the Euclidean distance measure though other distance measures can as well be employed.
Updating centroids where it is obtained by the computation of the arithmetic mean of the data points in the cluster.
To achieve this, the assignment and update steps have to be repeated until the centroids do not change sufficiently or are updated to the maximum number of times allowed.
Cluster convergence occurs when the centroids are fixed and k clusters are formed with each of the points in the data set neighbors to its corresponding centroid.

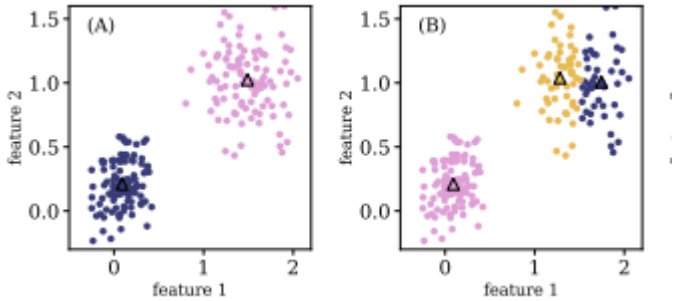


Image 14. https://arxiv.org/pdf/1904.07248

Key factors influencing K-Means performance:
Feature scaling: This is especially important to check since improperly scaled features could seriously jeopardize the soundness of the finally formed clusters.
Outliers: Centroids can be much affected by Outliers and hence the need to eliminate them prior to using K-Means technique.
Random initialization: The K-Means has an issue where it might converge to a local minimum: For this reason, the first initialization of centroids much be done carefully. To combat this, the algorithm is commonly ran with different intializations and the best results based on the least sum of squared distances are used.

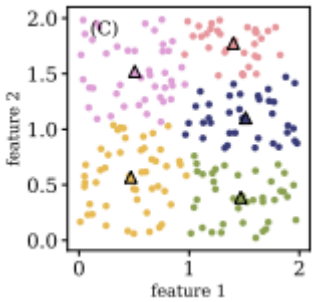


Image 15. https://arxiv.org/pdf/1904.07248

In simple situations such as two-dimensional situations, K-Means is easy to ‘see’ since clusters can be illustrated with the help of colors and centroids which are mid-point of the clusters. However for higher values of k the algorithm tends to over fit and provides as many clusters as the number of

points in data set, minimize the cost but not the objective of clustering. That is why the selection of the number of clusters k is the key factor of the algorithm efficiency.

4.2.9 Hierarchical Clustering

Hierarchical clustering is another type of machine learning technique which is used to segregate the data into clusters without any supervision and the result of this process is in the form of a dendrogram. It is different from the K-means which divides data into a prespecified number of clusters and creates a tree structure in the form of a cluster. The two main types are:
Agglomerative (bottom-up): Every data point is initially a cluster, and the procedure of clustering is performed by combining two similar clusters at a time.
Divisive (top-down): All the data points are initially in one cluster and it is divided into more sub clusters in each iteration.
Steps of Hierarchical Clustering:
Distance Calculation: The algorithm defines the distance between objects with the help of a metric such as the Euclidean distance; other metrics can also be used depending on the problem.
Merging Clusters: In the agglomerative approach the clusters are formed by merging the two closets points with the iterative merging process until all the points belong to one cluster or when a desired number of clusters is reached.
Linkage criteria determine how the distance between clusters is calculated during merging:
Single linkage: The distance between two clusters is defined by the shortest distance between any two points in the clusters
Complete linkage: It applies the maximum distance between two points in two different clusters.
Average linkage: The average distance of all points from all other points within the clusters.
Ward's linkage: Lowers the increase of within cluster variance when joining two clusters.
The type of linkage method establishes the clustering result and needs to be fine tuned in accordance with the data set.
Dendrogram Visualization:The hierarchical clustering can be presented in the form of a dendrogram which is a tree like structure where the horizontal axis denotes the clusters and the vertical axis represents the distance or dissimilarity between clusters. The level of two clusters' combination is relative to the similarity of the two clusters. In order to get a certain number of clusters, it is possible to cut the dendrogram at a certain level which corresponds to the defined problem.

4.2.10 Dimensionality Reduction Algorithms

The proliferation of high-dimensional datasets have made high-dimensional data sets to be both an opportunity and a challenge in the current and ever-evolving field of data science and machine learning. It results in increased risk of overfitting and high computational cost as some of the impacts of the curse of dimensionality. Dimensionality reduction can be defined as the process of reducing the size of the given dataset in terms of number of features present in the dataset or it can be defined as the process of selecting only those features which can effectively describe the dataset or creating new features from the existing features such that the new set of features can provide an effective description of the dataset. Some of the dimensionality reduction techniques provide principle components or prototypes which are small sets of objects having the same number of

dimensions as the original objects in the sample and are used to represent the entire sample. Some other approaches aim at the projection of the high-dimensional data set into a low-dimensional space while not using prototypes or basic components. Of course, when applying dimensionality reduction we always sacrifice some data. We have to find an algorithm that will preserve most of the important information and important information is defined by our scientific motivation.

4.2.11 K-Principal Component Analysis (PCA)

Among the most known techniques for data simplification is Principal Component Analysis (PCA) [44][45]. PCA reduces the data space from high-dimensional space to low dimensional space with the maximum variance of the data in this low-dimensional space. PCA is based on the construction of a covariance matrix of the given set of data and finding its eigenvectors, called principal components. Such eigenvectors which are related to the largest eigenvalues contain most of the information of the original data.

All the objects in the dataset can be expressed by the linear combination of these principal components. Each of the components has been employed in this representation and hence it is exact but when only some of the components are utilized, it gives an approximate representation with lesser dimensions. Usually, this reduction assigns the data onto two or three dimensions with a view of presenting it.

Key features of PCA:

No free parameters: PCA is easy to understand and the concept behind it is very easy to grasp.

Dimensionality reduction: A dimensionality reduction method that preserves most of the variance in the data and thus reduce the number of dimensions.

Sensitivity to outliers: The main idea of PCA is the analysis of the maximal variance, hence, PCA is sensitive to outliers, which should be excluded.

Some of the limitations include; the algorithm works well in identifying linear relationships that may not necessarily exist in the given data set and the fact that it uses mean and covariance to define the data set. Furthermore, while calculating the principal components some of the components may have negative values which may not be desirable in some cases.

4.2.12 t-Distributed Stochastic Neighbor Embedding (t-SNE)

t-Distributed Stochastic Neighbor Embedding (t-SNE) [50] is a nonlinear dimensionality reduction technique which is proficient in the preservation of neighborhood relationships and visualization of highdimensional data sets in two or three dimensions. Unlike other techniques like PCA, t-SNE aims at preserving the neighborhood relationships between the data points so that points that are close in high dimensional space are also close in low dimensional space whilst for points that are far apart the same applies in the low dimensional space.

Steps in t-SNE:

Compute Similarities in High-Dimensional Space:

Affinity Calculation: In the case of each pair of high dimensional data points, the similarity or affinity should be computed and this can be done through the use of Gaussian kernel or cosine similarity. This leads to similarity matrix that is symmetric in nature.

Compute Conditional Probabilities:

Probability Calculation: Translate the affinities to conditional probabilities which will express the probabilities of one data point to be a neighbor of another. This is done with the help of the normalized exponential of the negative pairwise Euclidean distances with the adjustment of the perplexity hyperparameter. The parameter perplexity decides the trade-off between the local and the global structure which defines the size of the neighborhood around each data point.
Create Low-Dimensional Mapping:
Random Initialization: We can set the coordinates of every data point to random values in the low dimensional space.
Optimization: Gradually update these coordinates to make the similarity of high-dimensional data be as close as possible to the low-dimensional similarity. This is done with the usage of gradient descent and in the low dimensional space, Student's t-distribution is utilized for tackling crowding.
Visualization: Once the optimization is completed then the low dimensional embedding makes it possible to visualize the clusters, patterns and any relations that may be present. We can see that data points which were close to each other in the original space will also be close in the t-SNE plot.
Considerations:
Stochastic Nature: t-SNE is a stochastic technique, this is because of this reason the results may differ from one run to the next due to the random initialization. Indeed, it is advised to run t-SNE several times and compare the results.
Computational Cost: Another drawback of t-SNE is that it may be very slow in terms of computational complexity and hence it might be impractical for large data sets.
The t-SNE method is especially valuable in data analysis and visualizing of multivariate relationships that cannot be demonstrated by linear methods.

4.2.13 Autoencoders

Autoencoders [13][21][22] are variants of neural networks which are mainly employed in unsupervised learning and the purposes of feature extraction. They are composed of two main parts: It involves the encoder and the decoder. The main task of autoencoder is to map the input data into a new space, called the "latent space" and then map this space back to the original input space [78].
The Architecture is:
Encoder:
Function: Transforms the data into a lower dimensional space which is called latent space.
Structure: It is made of one or more than one layer of neural nodes that compress the given data in terms of dimension. The last one in the encoder layer is referred to as the bottleneck which is the encoded form of the input data.
Decoder:
Function: Restores the original input data from the compressed form of the data.
Structure: Is the exact reverse of the encoder's architecture. This has the purpose of reconstructing the input data as accurately as possible by using layers that up-sample the latent space representation to the data space size.
Training:
Loss Function: Calculates the error between the original set of data and the set of data after reconstruction. Usually, the loss function applied is the least squared which measures the square of the difference.

Optimization: This loss is used during the training of the network and the weights of the network are adjusted to minimize this loss thus enhancing the accuracy of the reconstruction.
Applications:
Dimensionality Reduction: Does the following: reduces the space of the given data set while preserving important attributes.
Data Compression: Efficient in minimising the amount of storage and communication space that is needed to transmit data.
Feature Extraction: Identifies aspects of the data which can be useful for other activities or processes.
Anomaly Detection: The level of the reconstruction error is measured to detect anomalies; large reconstruction error might mean that there are outliers.
Considerations:
Flexibility: Autoencoders are very flexible and are capable of reconstructing intricate data sets.
Design Challenges: Choosing the right architecture and the right hyperparameters is not always easy. It is very crucial with these choices as the performance of the model largely depends on these choices.
Autoencoders are very useful for learning a compressed representations of data and are widely used in a number of applications like image and signal processing as well as feature extraction.

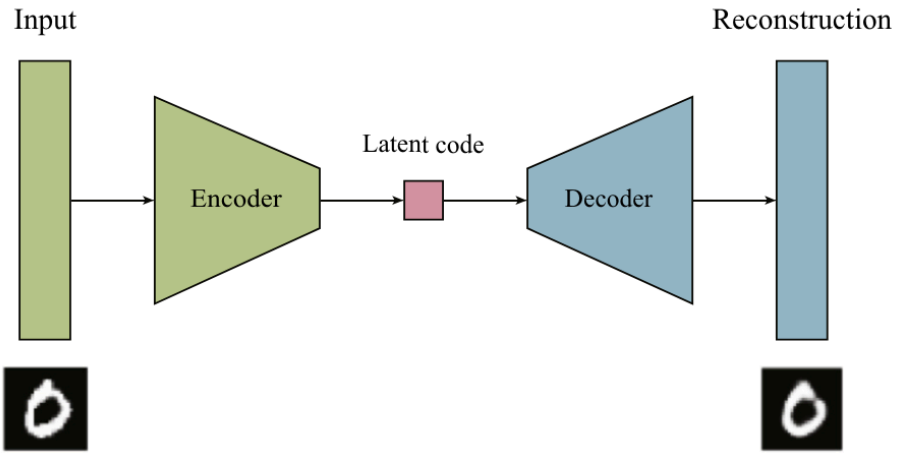


Image 16. The structure of the autoencoder. This structure comprises two neural networks, the encoder and the decoder. The input passes through the encoder to produce a
latent code; this latent code is then used by the decoder to generate the output. In most
autoencoders, the architecture of the decoder is the mirror image of the architecture of the
encoder.
https://www.sciencedirect.com/science/article/abs/pii/B9780128157398000110

4.2.14 Self-organizing map (Kohonen map)

Self-Organizing Maps (SOMs) [30] or Kohonen maps are among the unsupervised learning algorithms which are used for visualization of data and feature extraction. The key goal of the SOMs is to capture the high-dimensional data and project it into a lower dimensional space while conserving topology.

Training Process:
Initialization: The initial weights are assigned in a random manner and are chosen from the actual data points present in the dataset.
Training: The algorithm display input data points sequentially in a recursive manner. For each data point it computes the distance to all neuron's weights and find 'Best Matching Unit'. Both the BMU and its neighboring cells adjust their weights to be closer to the weights of the input point. The demarcation of the BMU reduces over time and changes in the system become less and less drastic and more discrete.
Key Components:
Learning Rate: Determines the amount of weight adjustment to be made during training.
Neighborhood Function: Gaussian, determines the region of the BMU's activity, which decreases with distance.
Advantages: Good for data visualization and clustering.
Limitations: It is very dependent on the chosen hyperparameters such as map dimensions, the learning rate, and the radius of the neighborhood.
Performance may differ with categorical and mixed type of data.

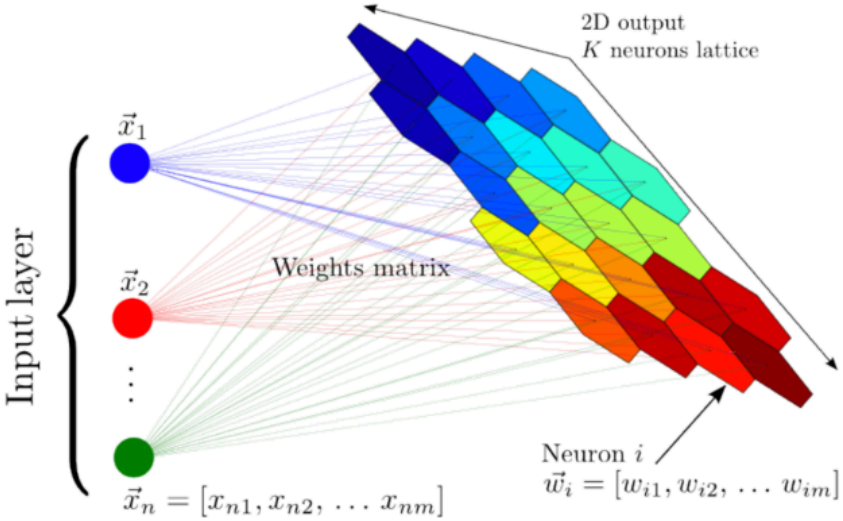


Image 17. A schematic illustration of a selforganizing map, taken from Carrasco Kind & Brunner (2014). The input dataset consists of n objects, each with m features, and it is mapped to a twodimensional lattice of k neurons. Each neuron is represented by a weight vector, which has the same dimensions as the input objects. The weights are characteristics of the neurons themselves, and they represent the coordinate of each neuron in the input data space. These weights are updated during the training process, and once the algorithm is trained, they represent a set of templates (or prototypes) that describe the different objects in the dataset.
https://arxiv.org/pdf/1904.07248

Chapter 5: Literature Review of Machine Learning Algorithms for Noisy Image Classification in Astronomy

The chapter examines the existing body of research on various machine-learning techniques used to address the challenges of noisy image classification in the field of astronomy. It provides a critical evaluation of both traditional and advanced algorithms, highlighting their strengths, limitations, and effectiveness in processing noisy astronomical data.

5.1 Review of studies on deep learning algorithms such as Convolutional Neural Networks (CNNs) and Autoencoders for image classification

5.1.1 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks or CNNs are a special type of ANN that is well suited to address image related data due to their specific architecture. CNNs have remained a popular choice in areas that entail image recognition and classification since they are capable of identifying the spatial arrangement of images [89].

The architecture of CNNs is characterized by several key components:

Convolutional Layers: These layers use convolutional filters on the input data to create feature maps which recognises certain patterns in the image. This localized processing is useful in capturing of spatial relations and hierarchical features within the image.

Pooling Layers: After the convolutional layers, it is followed by pooling layers, where the max pooling is the most common type that downsamples the feature maps. This process aid in reducing the redundancy in the representations and reducing their sensitivity to small variation in the position of features in the input [96].

Fully Connected Layers: Fully connected layers come after convolutional and pooling layers and it is at this final stage that the information is used to make predictions or classifications. These layers work in the same way as the conventional neural networks where each neuron is linked to every neuron in the previous layer.

Advancements and Innovations

Recent developments in CNN architecture have introduced several advancements: Inception Networks: The Inception architecture is an example of a development which enables the

construction of deeper and wider models through the utilization of multiple filters in parallel at different layers. This strategy which is used in GoogLeNet helps to save computational power and avoid overfitting of the model [14]. The Inception model also embraces the NiN in the architecture where convolutional layers contain micro neural networks to improve the feature learning process.
Deep Convolutional Networks: The evolution of the networks to the deeper ones, as shown in models like again, GoogLeNet indicates that the deep architecture of CNNs can enhance the performance of the classification tasks [60]. This must however be done while considering the computational cost so that the process does not consume a lot of resources.
CNNs have shown exceptional performance in various practical applications:
Astronomy: CNNs are employed in astronomy for categorizing the celestial objects and for image analysis of astronomical objects. These are effective in predicting and grouping various astronomical objects and events including galaxies and star clusters by analyzing large sets of images which have been labeled.
Gravitational Wave Detection: In the area of multi-messenger astrophysics, the CNNs were employed for the estimation of parameters of gravitational waves [27]. Current state-of-the-art models, including but not limited to, real-time processing. CNNs are faster and more accurate than conventional methods. These models help in early identification and analysis of gravitational wave signal which is very important as follow up with electromagnetic observations.
Practical Considerations
Training and Optimization: The training of an effective CNN include dealing with large data set and the tuning of the network parameters. To enhance the generalization and robustness of the models, data augmentation and other enhanced optimization techniques are used.
Computational Resources: Practical application of CNNs demands a lot of computational capacity, especially in the learning process of deep models [79]. Effective methods of handling big data and advanced computing technologies are critical to the processing of large images.
CNNs remain to be valuable in image-related applications and many other fields with advancements that keep on enhancing the performance and effectiveness [61]. This is the reason why they are considered as the most important entity in the improvement of machine learning in different areas.

5.1.2 Autoencoders

Autoencoders are one of the most basic and important types of deep learning models that are used mainly in unsupervised learning and feature extraction. These models can map the input data to a code and then map the code back to the original data and therefore, are useful for feature learning and data preprocessing [84].
Autoencoders work in a way that, given an input, it encodes it into a lower dimensional vector space and then decoding it to get as close as possible to the original input [87]. This process allows for the identification of important characteristics from large datasets which has a large number of dimensions. Over time, several variants have been developed to enhance their functionality:
Denoising Autoencoders (DAEs): Such models aim at learning the representations of the input data even if it is noisy or partial by transforming the noisy or partial version of the input into a clean one, thus enhancing the model's robustness [40].
Sparse Autoencoders (SAEs): Through the incorporation of a sparsity constraint, SAEs encourage feature selectivity and therefore are suitable for use in situations where only a few features are important.

Variational Autoencoders (VAEs): VAEs combine the probabilistic graphical models with autoencoder which helps in generating new data and hence learning the uncertainty in the data.
Autoencoders have proven versatile across various domains [19]:
Image Processing: Some of the applications of autoencoders include image denoising and anomaly detection among others. They can even create an image with less noise or, on the other hand, emphasize the salient points by learning how to code and decompress normal data samples.
Data Generation: Among the generative models Autoencoders are especially effective in the creation of new synthetic data which is similar to the training data in some way, which turns out to be useful in data augmentation and in simulating of the new conditions.
Despite their strengths, autoencoders face some disadvantages:
Architecture Selection: Optimal architecture and hyperparameters search is not a trivial task and may often involve a lot of trial and error.
Reconstruction Quality: The results show that the details of the reconstruction need to be matched with the amount of data the encoding process requires. This leads to poor results since one risks over-relying on one of the aspects of the model.
In Remote Sensing and Astronomy there are many applications in drones.
Autoencoders are now being used in fields that are specific like remote sensing and astronomy and this is an indication that the autoencoders are versatile and can be used in a complex data environment.
Remote Sensing: The PolSAR Image Classification
Autoencoders are employed in remote sensing for the purpose of classification of the polarimetric synthetic aperture radar (PolSAR) images. A multilayer autoencoder is used to learn low-dimensional representation of PolSAR data for classification purposes. The inclusion of softmax regression layer enables the classifying each pixel of the image based on these features. Furthermore, the increase in the model's robustness is due to self-paced learning that gradually exposes the model to more complex samples in the training process thus improving the model's accuracy on different datasets.
Astronomy: Denoising and Classification
Autoencoders also find significant applications in astronomy [24]:
Denoising Radio Astronomical Images: Deep learning model specifically Convolutional deep denoising autoencoder [52] is used for noise reduction in Radio Astronomical Images to improve faint Cosmic Structure Visualization. To this end, the model is trained on a set of noisy and corresponding clean images and thus is able to filter out the noise and enhance the quality of the images used in scientific analysis.
Galaxy Morphology Classification: VAEs are applied to unsupervised clustering and classification of the galaxy shapes. This is done by encoding images into a latent space and then using probabilistic inference for classification of galaxies without having to equip them with labels. It also creates synthetic images that embody such information for the purpose of creating more data or for theoretical purposes.
Autoencoders remain a highly effective tool in their use in the wide range of applications within the general machine learning field as well as in specific areas of application. They help in feature extraction and learning from large and complex data sets which find their application in remote sensing, astronomical image analysis. Current studies are still in progress to overcome current limitations, improve techniques and discover new possibilities of applications and thus strengthen the position of autoencoders in the field of deep learning.

5.2 Review of studies on traditional machine learning algorithms such as Decision Trees, Random Forests, Support Vector Machines, and K-Nearest Neighbors for image classification

5.2.1 Decision Trees

Decision Trees is one of the popular techniques of machine learning used in classification and regression analysis [86]. In Astronomy data analysis, Decision Tree can be used in classification of data that come from large surveys such as the Sloan Digital Sky Survey (SDSS) [49] [92].

A Decision Tree is a tree like structure where at each node a branch is made on the basis of an attribute and the final branch depicts the class label [99]. The aim of the algorithm is to partition the data into subsets which are as much as possible similar in terms of their target variable [85].

Applications in Astronomy

Object Classification: Decision Trees are applied in categorizing astronomical object into different groups/phenomena for instance stars, galaxies, and quasars. This class of objects enables one to know the characteristics and location of objects in the universe.

Handling Large Datasets: Astronomical surveys usually contain tens of millions of records. The use of Decision Trees is more suitable here because they do not only have the capacity to deal with huge amount of data but are also easy to understand.

Dealing with Complex Features: Astrophysical data contain photometric and spectral properties among other features. This is because features such as these can easily be dealt with by the Decision Trees to arrive at right classifications.

Techniques and Enhancements

Axis-Parallel Splits: Criteria for splitting of data into branches in Decision Trees are very important for getting good results. The most frequently used type of splits is axis-parallel splits where decisions are made based on individual features of data. These splits enable to form rather distinct boundaries for decision making.

Bagging and Boosting: In order to enhance the stability of Decision Trees, there is the usage of aggregating methods such as bagging (Bootstrap Aggregating) and boosting. Bagging is to train several trees from different samples of the data set and make the class assignment by taking the majority's view. Boosting enhance the model performance by constructing trees in a sequential manner which aims at correcting the errors of the previous trees.

Parameter Optimization: Certain factors that include the maximum depth of the tree, minimum samples that must be present at the leaf nodes, and the splitting metric are adjusted to improve the model's predictive power. For example, the tree depth is regulated to prevent the occurrence of overfitting of the data.

Probabilistic Classification: Classification trees can give probability values for the classes that are of interest in the classification. This approach enables a more detailed categorization and also aids in the creation of catalogues with desired completeness and efficiency parameters.

Challenges

Handling Imbalanced Data: Some datasets of astronomical objects have imbalanced classes, so that some classes contain much fewer objects than others. Decision Trees are to be modified in order to handle such imbalances as it may lead to biased classifications.

High-Dimensional Data: Most of the data with astronomical nature may contain a large number of features, and it results in large dimensions. Decision Trees can easily overload with high-dimensional data and thus become too complex and overfitting.
Computational Resources: As for the number of features and depth of the tree, the Decision Trees' complexity rises. It is important to note that analyzing large astronomical datasets may be computationally intensive and hence the use of parallel processing and supercomputers [81].

5.2.2 Random Forests

Random forest belongs to the ensemble learning method that is suitable for classification and regression problems. In astronomical research where the identification of objects like stars, galaxies and quasars is important Random Forests are an effective method of analysing multi-dimensional spectral data [90].
Random forest is a set of decision trees constructed on the basis of different subsets of data and different feature space [97]. The last step is the combining of the predictions from all the trees, which is usually done by averaging for regression and by voting for classification. This ensemble approach is used to enhance the accuracy and reliability of the results as compared to decision tree.
Applications in Astronomy
Classifying Astronomical Objects: Random Forests are employed to classify the astronomical objects depending on several characteristics including spectral properties.
Handling Large and Imbalanced Datasets: Imbalanced class distributions are a common characteristic of astronomical datasets, where a large number of observations is collected. Random Forests perform well with such datasets as they are capable of handling imbalances and high dimensions since they are ensembles and also accept different type of features.
Feature Importance and Interpretation: Random Forests are known to have a feature of feature importance, that is one of the main advantages of the method. This can assist in finding out which attributes are most useful in the classification process and hence facilitate the understanding of complex astronomical data [98].
Techniques and Enhancements
Hyperparameter Optimization: Hyperparameters of Random Forests which may include the number of trees, tree depth and number of features to split on can greatly influence the performance of this algorithm. Methods such as random search are employed for effective searching of hyperparameter space and the optimization of these parameters.
Class Imbalance Management: Random Forests are not sensitive to class imbalance problem which is a prevalent issue in astronomical datasets. Sensitivity, Specificity, and Balanced Accuracy are some of the measures that are used to determine the efficiency of the model to ensure that it performs well in all the classes.
Comparison with Other Classifiers: A comparison can be made between the Random Forests and other machine learning algorithms like the Support Vector Machines (SVMs), Neural Networks and Naïve Bayes [11]. In general they achieve high accuracy and are relatively less sensitive to noise making them suitable for a number of classification problems in astronomy.

	Naïve Bayes	IB1	IB3	SVM (SMO)	MLP	**Random Forest**

sensitivity	0.9409	0.8581	0.8466	0.9080	0.9710	**0.9792**
specificity	0.9685	0.9251	0.9250	0.9433	0.9845	**0.9940**
balanced accuracy	0.9547	0.8916	0.8858	0.9256	0.9777	**0.9866**

Table 1.

	Naïve Bayes	IB1	IB3	SVM (SMO)	MLP	**Random Forest**
sensitivity	0.9630	0.8515	0.8544	0.9433	**0.9788**	0.9783
specificity	0.9862	0.9243	0.9261	0.9676	0.9898	**0.9940**
balanced accuracy	0.9746	0.8879	0.8858	0.9554	0.9843	**0.9861**

Table 2.

	Naïve Bayes	IB1	IB3	SVM (SMO)	MLP	**Random Forest**
sensitivity	0.6329	0.8185	0.8022	0.9313	0.9657	**0.9756**
specificity	0.8108	0.9057	0.8988	0.9636	0.9818	**0.9874**
balanced accuracy	0.7219	0.8621	0.8505	0.9474	0.9738	**0.9815**

Table 3.

Challenges

Computational Resources: Random forests although not as complex as other models such as deep learning models they are still rather costly in terms of computational power to train especially with large data set and many trees.

Interpretability: Random Forests are also capable of giving a hint regarding the importance of the features, however, the whole model can be less interpretable than some simple models. This is because there is the issue of combining multiple decision trees which can make the whole process quite complicated.

Overfitting: Random forests are less prone to overfitting than single decision trees but overfitting is still a problem and has to be avoided by careful attention to the parameters especially when dealing with high dimensional data.

5.2.3 Support Vector Machine

Support Vector Machines (SVMs) are the most common classifiers employed in several fields including astronomy. In the case of galaxy morphology classification, SVMs have been found to be useful because they can work with large number of features and complicated decision planes. SVMs

have been used in galaxy classification in order to divide the galaxies into different types like spiral, elliptical and irregular based on the morphology [80].

SVMs are considered to work through identifying the best possible hyperplane that can be used in classifying the data points belonging to different classes with the maximum possible margin. As for binary classification problems, SVMs are very good, and thanks to such extensions as multiclass SVM, it is possible to solve problems with more than two classes. This has been helpful in the field of astronomy to categorize vast collections of galaxy images, for instance, the ones taken from the SDSS and other similar sources.

Multi-Class Classification

SVMs are binary classifiers by nature, which means that they are capable to classify data into two classes only. However, in multiclass classification problems, SVMs can be applied by using techniques like ‘one-vs-one’ or ‘one-vs-all’. In ‘one-vs-one’ method, an SVM is trained for each two classes while in ‘one-vs-all’ an SVM is trained for each class and it has to distinguish between itself and other classes. These methods enable SVMs to approach various multi-class problems and perform various classifications of objects, for instance, galaxies can be classified into several types (spiral, elliptical, irregular). By using the kernel trick and through the binary classification of the problem, the SVM can successfully handle issues of multiclass and still deliver high accuracy especially with large datasets.

Feature selection is very important in the case of the SVMs in astronomical image classification. More often used are characteristics that characterize the structure, brightness and the form of galaxies (for instance, disk, smooth, irregular). In order to increase the classification performance, the kernel functions like Radial Basis Function (RBF) or Polynomial kernels are used. These kernels enable SVMs to map data in to higher dimensional space where it becomes easier to classify data with non-linear boundaries between different classes.

Practical Implementations and Accuracy

Astronomical datasets for instance from the Galaxy Zoo project [16], comprise manually labelled galaxy images that can be used as training data for an SVM. It has been observed that the SVMs give very good results in terms of classification between two classes of galaxies namely Spiral and Elliptical galaxies. Nonetheless, the identification of irregular galaxies’ types is relatively difficult owing to their less structured and more complex forms which results in lower classification accuracy as compared to the other categories.

Quantum-Enhanced SVMs [9]

The advance in quantum computing has led to quantum-enhanced Support Vector Machines (SVMs) which are exploited for galaxy classification. Quantum SVMs extend classical SVMs by using kernel methods which map data points into high-dimensional quantum feature spaces with the aim of enhancing classification performance and reducing computational complexity. However, quantum computing hardware in the present day holds limitations, quantum augmented models can perform as well as or even outperform classical SVMs with potential for large astronomical data sets.

5.2.4 KNN

Among the most known algorithms k-Nearest Neighbors (k-NN) is often applied in astronomy for classification purposes because it is rather simple and suits multidimensional data well. In space science, k-NN is used to determine the nature of celestial objects by comparing the distance of the objects under consideration with templates in the feature space. These proximities are computed

using different distance measures like Euclidean or Cosine Similarity thus enabling the data to be modeled in any way possible. However, problems such as slow computation when working with big data sets have forced the use of data structures such as k-d Trees and Locality Sensitive Hashing (LSH) [36] to enhance neighbor search. In the field of astronomy [91], k-NN is extremely beneficial for assignments such as classification of galaxies, stars or other forms of astronomical bodies [88], for instance brown dwarfs, where k-dimensional color-magnitude space is used for object identification against background noise [77]. This makes it efficient in classification since it does not assume any distribution of the data and thus can be useful in analysing the large amount of data produced by current astronomical surveys [83].

Chapter 6: Evaluation Metrics for Machine Learning Algorithms

In this chapter, an overview of the metrics commonly used to evaluate the performance of machine learning algorithms in image classification tasks will be discussed.

6.1 Explanation of evaluation metrics used to assess the performance of machine learning algorithms for image classification

Evaluation metrics are important in order to determine the efficiency of machine learning algorithms in the process of image classification. These metrics give numerical values of the degree of correctness of the algorithms in identifying images. Here's a overview of common metrics and their uses:
Purpose: Metrics are used in order to assess how well the classifiers will perform on unseen data, compare the performance of different models and choose the best one during the learning process.
Generalization Ability: There are some measures like accuracy to gauge the classifier performances on the new set of data that it has not encountered before.

6.1.1 Accuracy

The accuracy is the percentage of the images which is classified correctly out of the total images. It is calculated using the formula:

$$\text{Accuracy} = \frac{Total\ Number\ of\ Predictions}{Number\ of\ Correct\ Predictions}$$

where:
True Positives (TP): Identified properly the positive cases. False Positives (FP): It refers to the cases where the model has predicted that the instances belong to the positive class when in actual they do not. False Negatives (FN): False Positives. True Negatives (TN): It was able to predict the negative instances correctly.
Accuracy can also be expressed as:

Accuracy= $\frac{tp + tn}{tp + fp + tn + fn}$

A high accuracy rate, for instance, 85% shows that many images are classified correctly.

Simple and important and more so when the classes are balanced.

However, accuracy does not hold true especially when working with imbalanced data sets, where certain classes are over-sampled or under-sampled.

Accuracy is most valuable when all the classes are equally important and data is split evenly between classes.

6.1.2 Error Rate

A straightforward and easy to understand evaluation metric which is used in image classification is the error rate or the misclassification rate. It shows the number of incorrect predictions that is the number of instances classified incorrectly to the total number of instances.

Mathematically, the error rate is calculated using the following formula:Mathematically, the error rate is calculated using the following formula:

Error rate= $\frac{Number\ of\ Misclassified\ Images}{Total\ Number\ of\ Images}$

and the formula can be also seen as:

Error rate= $\frac{fp + fn}{tp + fp + tn + fn}$

A higher accuracy level means that fewer images are wrongly classified hence the image classification model is doing a better job. However, a higher error rate signifies a poor performance of the given model as more number of images are misidentified.

6.1.3 Precision

Precision is one of the most important criteria in the sphere of machine learning, particularly in the image recognition area. It offers important information on the quality of correctly identified positive cases where the emphasis is made on the reduction of false positives.

The way to compute precision is through dividing the number of true positive predictions to the total number of predictions that are positive, which include the true positives as well as the false ones. The formula is:

Precision= $\frac{True\ Positives}{True\ Positives+False\ Positives}$

This can be evidenced by the precision score which shows the model's capability to differentiate between positive samples with high confidence. This means that the model is efficient in its ability to recognize and categorize important features within the images toward image classification.

Precision is especially important in image classification when the cost of false positives is high. For example, in the case of medical imaging, it is crucial to have high precision to avoid wrong diagnoses. Precision is usually discussed together with another measure called recall. Precision is

the ability to avoid making wrong diagnosis while recall is the ability to identify all the cases that need to be diagnosed.

6.1.4 Recall

Recall is a vital assessment technique in the field of machine learning especially in image classification. It offers valuable information on the model's capacity to identify all the positive instances as explained earlier with a special emphasis on the reduction of false negatives.
Recall has been defined as the proportion of actual positives which are correctly identified as positives, that is, the number of true positives out of the total number of actual positives, which includes both true positives and false negatives. Mathematically, the recall formula is defined as:

$$\text{Recall} = \frac{True\ Positive}{True\ Positives + False\ Negatives}$$

A higher recall score means that the model is effective in the detection of the positive data points in the dataset and captures most of them.

6.1.5 F1 Score

F1 score is one of the most important parameter for the evaluation of machine learning model which tries to achieve the best of both the world of precision and recall.
Sometimes precision is used in connection with recall. Recall is based on minimising false negative instances while precision is based on minimizing false positive instances. For that reason, it is essential to find an adequate balance between these two metrics when developing the model in question. The formula used for F1 Score is:

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

The harmonic mean is utilized here because it gives equal weight to precision and recall, ensuring a balanced assessment. The F1 score lies between 0 and 1 with 1 indicating the best recall and precision. Thus a model which is considered good has a higher F1 score, which is the harmonic mean of precision and recall. This F1 score is important since it shows the level of the model's ability to predict the right class positive (precision) and including most of the positive cases in the data set (recall).

6.1.6 Confusion Matrix

In the field of machine learning the confusion matrix is a simple yet powerful tool that helps to evaluate the performance of the classification model.
A standard confusion matrix is a two dimensional matrix that shows true positive, true negative, false positive and false negative predictions.
A confusion Matrix is organized as:

	Actual Positive Class	Actual Negative Class
Predicted Positive Class	True Positive (TP)	False Negative (FN)

Predicted Negative Class	False Positive (FP)	True Negative (TN)

Table 4.

It is possible to see some patterns of misclassifications that can help practitioners in the improvement of the model by using the confusion matrix. Certain modifications can be made in order to increase the model's accuracy for a given class of interest if the current class identification is poor.

6.1.7 Mean Square Error (MSE)

Mean Squared Error (MSE) is another performance measure popularly employed in machine learning, especially for regression analysis. It tells the performance of a model by comparing the actual and predicted values by calculating the average of the square of the differences between the two.

The formula for calculating MSE is as follows:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (Pi - Ai)^2$$

where n is the total number of instances in the data set, Pi is the predicted value of the ith instance and Ai is the real target value of the ith instance.

it measures the difference between the predicted and actual values and the larger the error, the more it is punished by squaring it. Lesser MSE value is a better indication of a model's performance since it gets closer to the actual value.

However, MSE does not consider class imbalance therefore in case of setting wrong weights at time of initialization, suboptimal solutions can be derived.

MSE also offers no distinction between the types of errors (true positives and true negatives, false positives and false negatives).

MSE is used for continuous target variables and, it is useful when the magnitude of errors is important.

6.1.8 ROC curve and Area under the ROC Curve (AUC)

A Receiver Operating Characteristic (ROC) curve is a fundamental tool in machine learning for evaluating the performance of classification models. It allows us to visualize the trade-off between true positive rate and false positive rate for different classification thresholds.

As a plot, the curve has two parameters, the True Positive Rate(TPR) and the False Positive Rate(FPR). The TPR represents typically the recall and is defined as the recall:

$$\text{TPR} = \frac{tp}{tp + fn}$$

The FPR, is defined as follows:

$$\text{FPR} = \frac{fp}{fp + tn}$$

A ROC curve plots TPR vs. FPR at different classification thresholds. The higher the True Positive Rate and the lower the False Positive Rate for each threshold, the better. Better classifiers have more curves on the left. The area below the ROC curve is called the ROC AUC score, a number

that determines how good the ROC curve is. AUC stands for Area Under the Curve and as both TPR and FPR range between 0 to 1, So, the area will always lie between 0 and 1. In order to achieve the highest TPR and lowest FPR at the specified threshold, our primary objective is to maximize this area. The likelihood that the model will give a randomly selected positive instance a higher predicted probability than a randomly selected negative instance is captured by the AUC.
For a two-class problem, the AUC value can be calculated as:
FPR = $\frac{Sp - np*(nn+1)/2}{np * nn}$, where, Sp is the sum of all positive examples ranked, while np and nn denote the number of positive and negative examples respectively. The AUC has been shown to be both theoretically and practically superior to the accuracy metric [17] for assessing classifier performance and identifying the optimal solution during training.
Although the performance of AUC was excellent for evaluation and discrimination processes, the computational cost of AUC is high, especially for discriminating a volume of generated solutions of multiclass problems.

Chapter 7: Analysis of Research Studies on Machine Learning Algorithms for Noisy Image Classification in Astronomy

In previous chapters, we explored a diverse array of machine learning algorithms, each with unique strengths and applications. As we move forward, our focus will shift towards a practical, hands-on approach: coding, testing, and comparing these models to discern their performance in real-world scenarios. This process will involve implementing the algorithms in a coding environment, rigorously testing them using a standard dataset and the same conditions, and employing statistical methods to compare their accuracy, efficiency, and scalability.

7.1 The data set

The data that will be used in this paper is a folder which contains pictures of the planets of the solar system.

The dataset created as “Planets Dataset” is designed and compiled to foster the training and testing of machine learning models based on image classification in the astronomical field. It is a set of images that depicts different planets that are present in our solar system. This large set of images is proposed to evaluate and compare state-of-the-art image recognition techniques.

The Planets Dataset is divided into folders where each folder contains data of a particular planet. The dataset consists of images of the solar system planets namely Mercury, Venus, Earth, Mars, Jupiter, Saturn, Neptune and Uranus with each of the image contained in a folder of its planet. This organization makes it easy to sort out and analyze the data as a part of the machine learning process in training, validation, and testing sets.

All the images within the dataset are stored in JPEG format hence making it easier for any image processing and machine learning libraries to access. Each image has different dimensions meaning that the sizes and resolutions of the images are different, this helps in training the models in identifying planets at different sizes and at different angles. This variety improves the generality of machine learning models trained on this dataset, thus making them ready to tackle variability in real-life astronomical data.

Specific difficulties of training on the “Planets Dataset” include learning to work with the inherent fluctuation of the planetary images caused by the lighting, angle, and distance. Nevertheless, these challenges are also the ways that can be used for the creation of more and more advanced models which can help identify the planets with high accuracy and make them beneficial for teaching and for professional astronomical investigations.

7.1.1 Adding noise to the Data set

In this section, we propose a new method of adding noise into the solar system planets data set. We will introduce different noise to these images using Python programming language. This method is used to mimic real life conditions that may be present in astronomical data acquisition. When applying controlled perturbations we not only evaluate the stability and flexibility of image processing algorithms but also increase the usefulness of the dataset for training more reliable machine learning algorithms.

Noise Addition

Gaussian Noise: We can produce Gaussian noise and then add it to the original image in order to create the noisy image. Gaussian noise is a type of noise which has probability density function same as the normal distribution. This noise is produced by adding a random Gaussian function to the Image in order to obtain the desired noise. Since it is common in amplifiers and detectors, it is also termed as ‘electronic noise’. It is due to the particle like nature of radiation coming from hot bodies. In order to generate Gaussian noise we first generate an image of zeros with the same dimensions as the original image. We then generate the noise with the help of random distribution with mean =128 and sigma =20 for pixel values.

Uniform Noise: The noise of uniform distribution with the range from 0 to 255 is introduced and then added to the original image. Unlike Gaussian noise, Uniform noise is a signal-dependent (unless dithering is induced or employed) noise which is uniformly distributed. It is due to the fact that an image is quantized to some certain levels of gray in the image. It is usually formed when an analog data is transformed to a digital form. To generate a Uniform noise, we generate a uniform distribution with the lower and upper bound of the pixel range of the image which is 0 to 255 respectively.

Impulse Noise: Salt and Pepper noise is also called as Impulse noise which occurs randomly and very less frequently with a pixel value of either 0 or 255. This can be noticed by the black pixel implementation in high luminance regions and the white pixel implementation in low luminance regions. This type of noise is produced due to spike like disturbances in the image signal and the major cause of such noise is the errors in the ADC (Analog to Digital Converter) or bit transmission. To generate a Salt and Pepper noise, we first generate a distribution that is a bit similar to the one used in Uniform noise and then apply binary thresholding to come up with a salt and pepper noise wherein some pixels are black while others are white. Thus the level of noise produced can be easily adjusted depending on the threshold chosen.

Denoising

To denoise images contaminated by each type of noise we will employ the Fast Non-Local Means Denoising algorithm which is offered by OpenCV.

Median and Gaussian Blurring

We will use median blurring and Gaussian blurring to the noisy images as the other approaches of denoising. These operations are performed on the image and the outcomes are shown below to prove how helpful these filters are in noise elimination.
This can assist in applying random noise into images and removing noises within a given dataset, useful for data enhancement and initial data processing of the Machine Learning tasks. We will first load the training images, then introduce one of the three types of noise such as Gaussian, Uniform or Impulse then apply a denoising filter. The kind of noise that is used is chosen randomly. A denoised image is saved in the output path that is specified. Depending on the type of noise added, a different filter is being used:
Gaussian Noise: A noise is added to the image and then Gaussian filter is applied for the purpose of noise removal from the image.
Uniform Noise: The noise is then added to the image and median filter is then applied to reduce the noise from the image.
Impulse Noise: Impulse noise is introduced and median filter is applied for the purpose of the noise removal.
Each image in the given directory is processed one at a time. The function is able to work with various image formats and these include PNG, JPG and JPEG.
The results are:

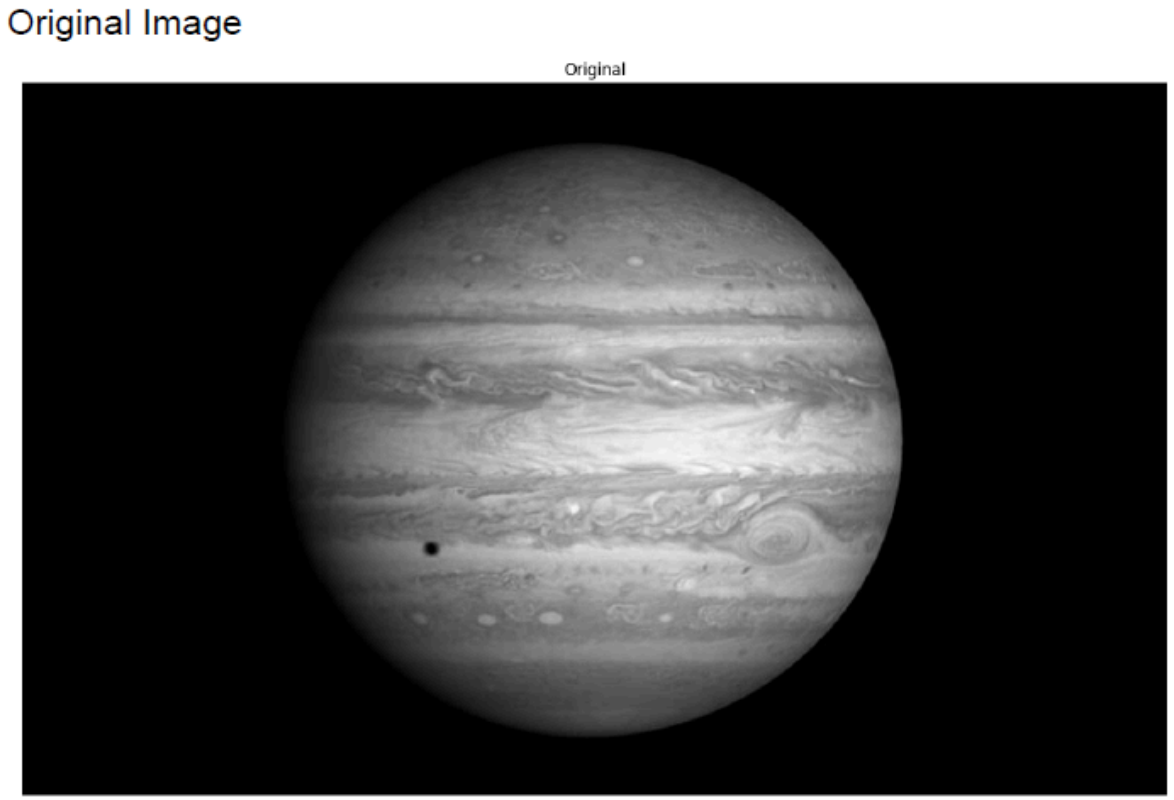


Image 18.

Gaussian Noise

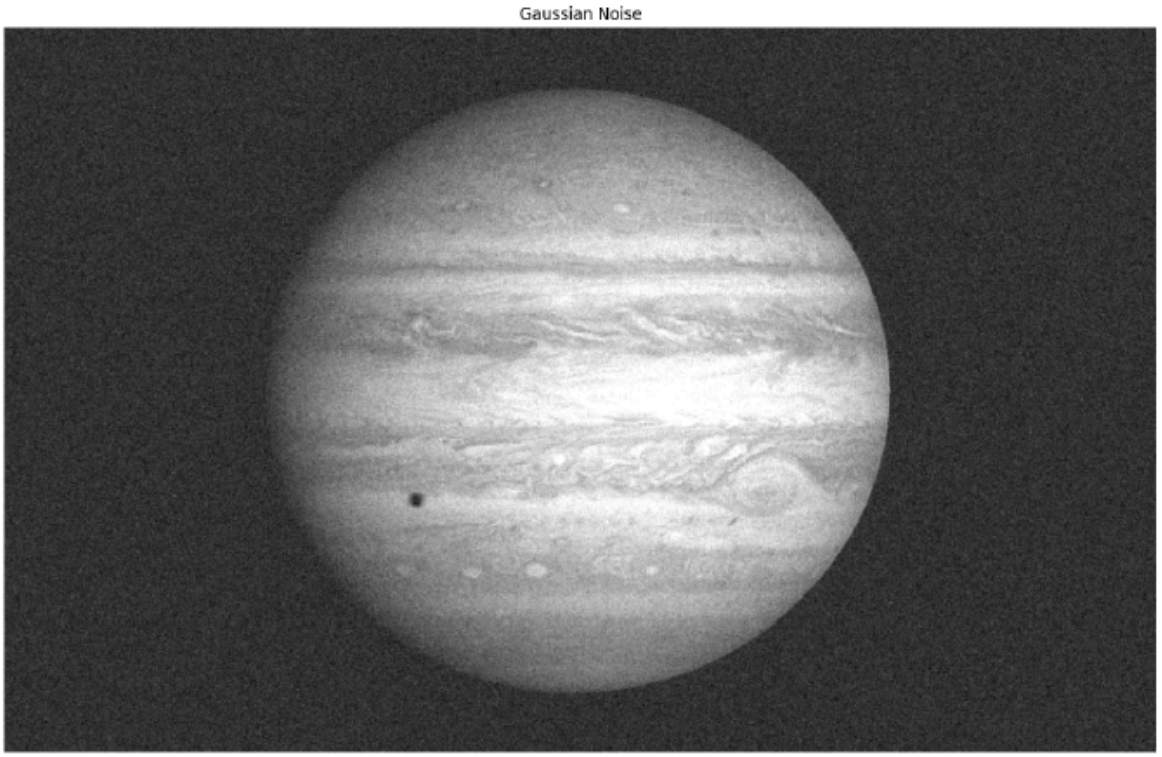


Image 19.

Uniform Noise

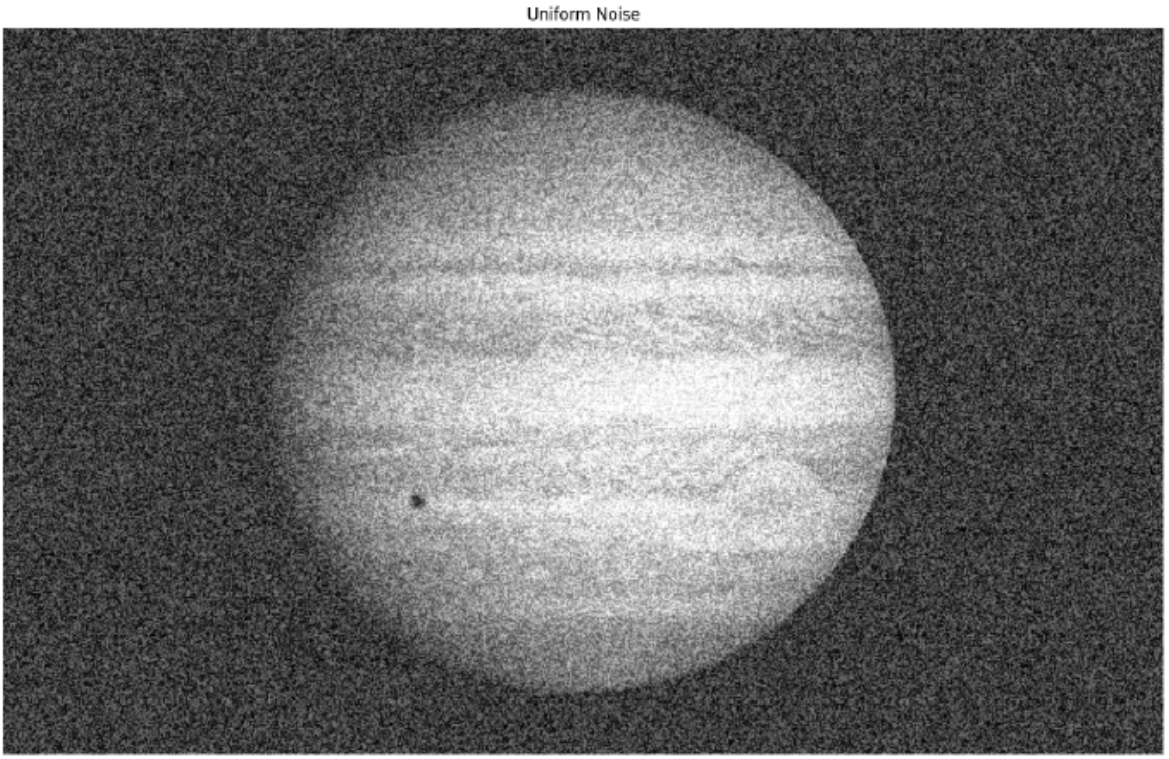


Image 20.

Impulse noise

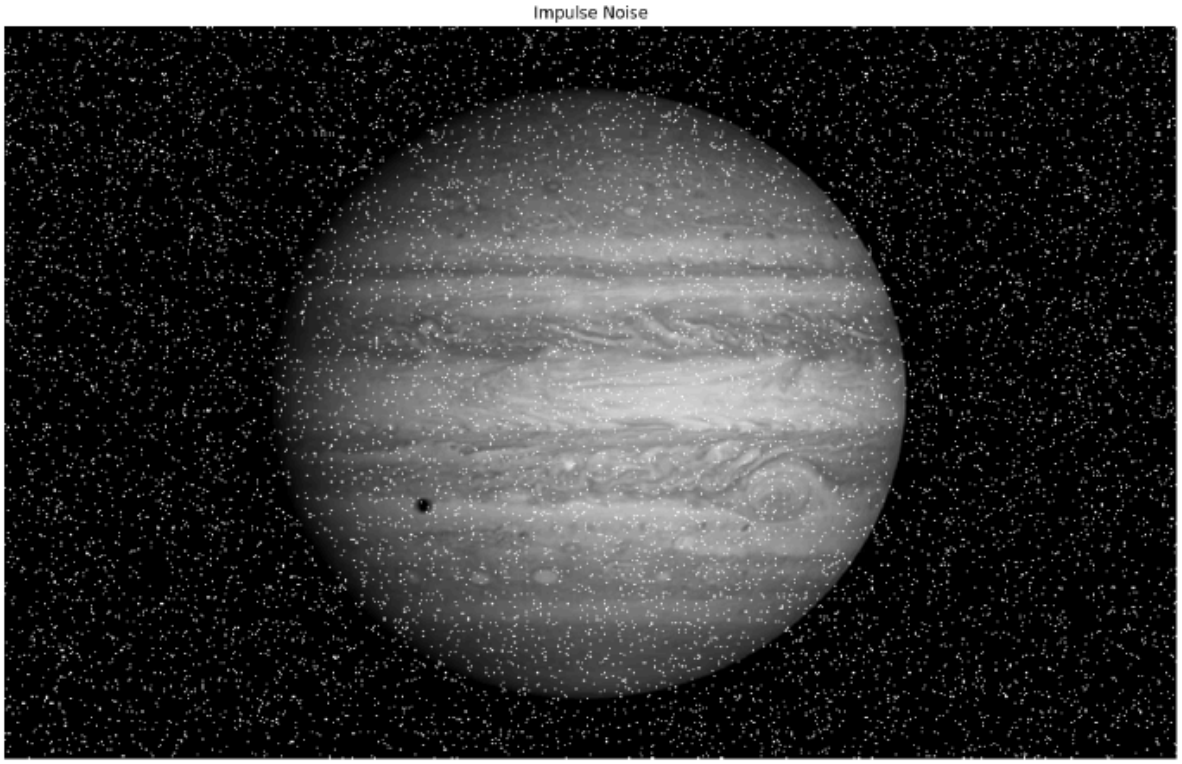


Image 21.

Denoised Gaussian Noise

Image 22.

Denoised Uniform Noise

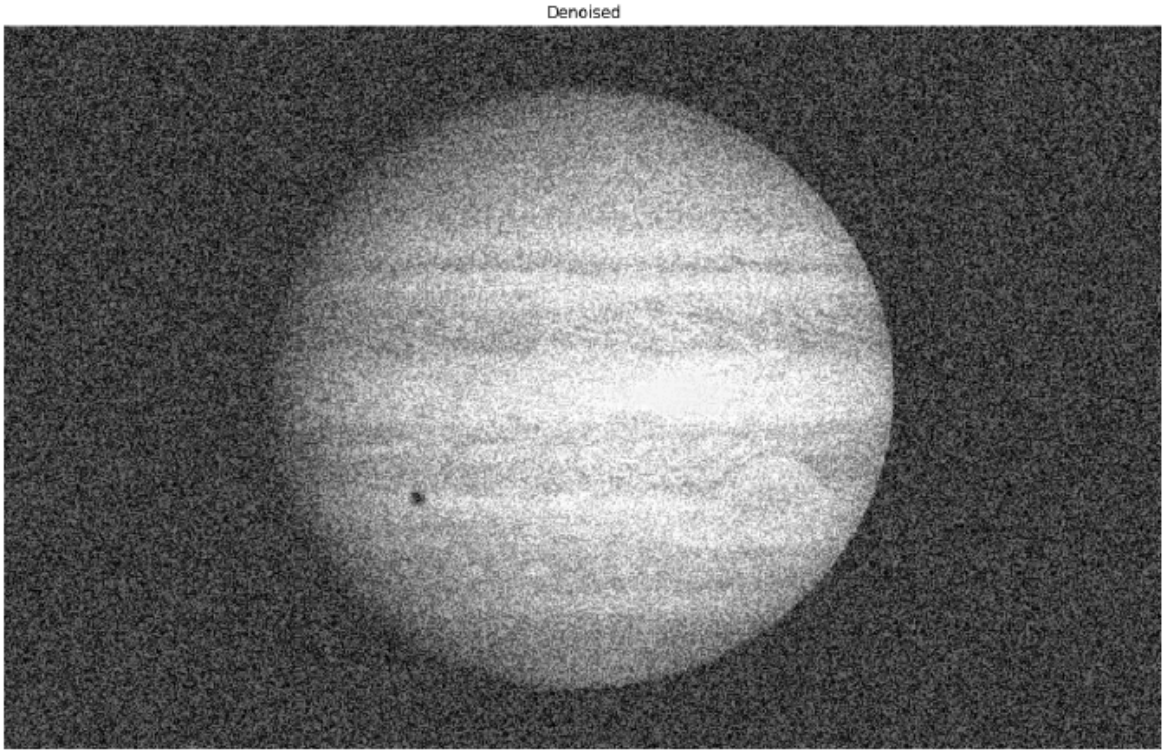


Image 23.

Denoised Impulse Noise

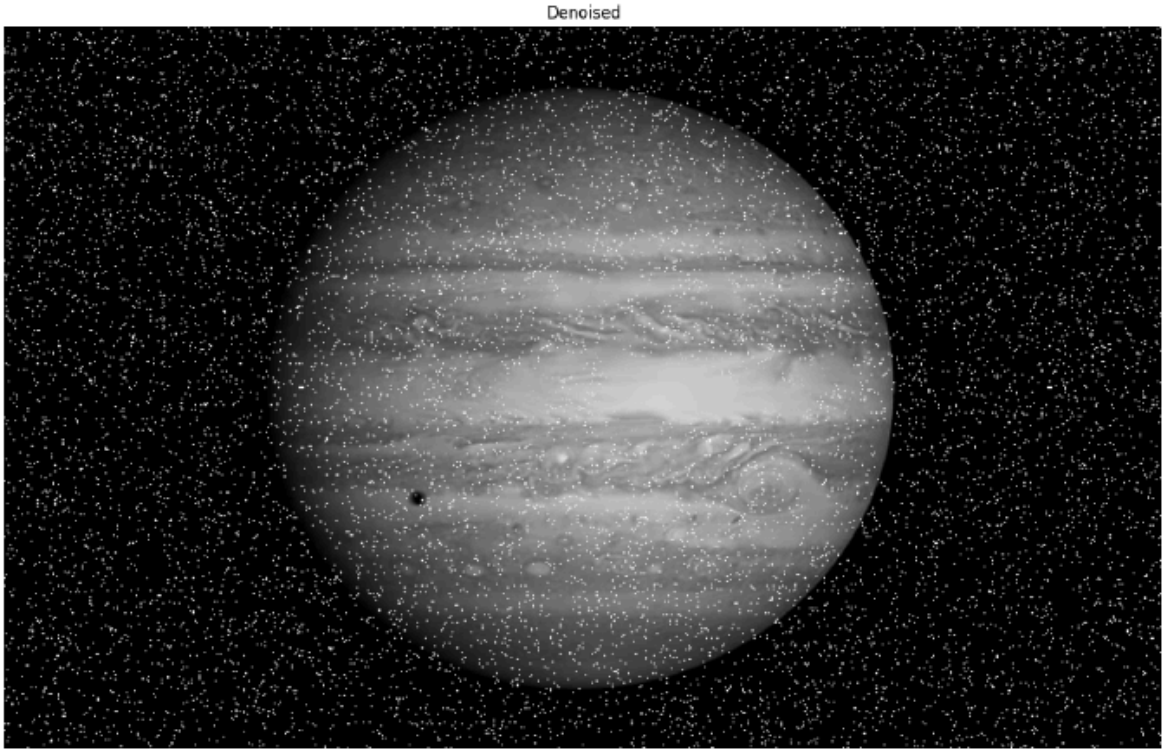


Image 24.

Median Filter on Gaussian Noise

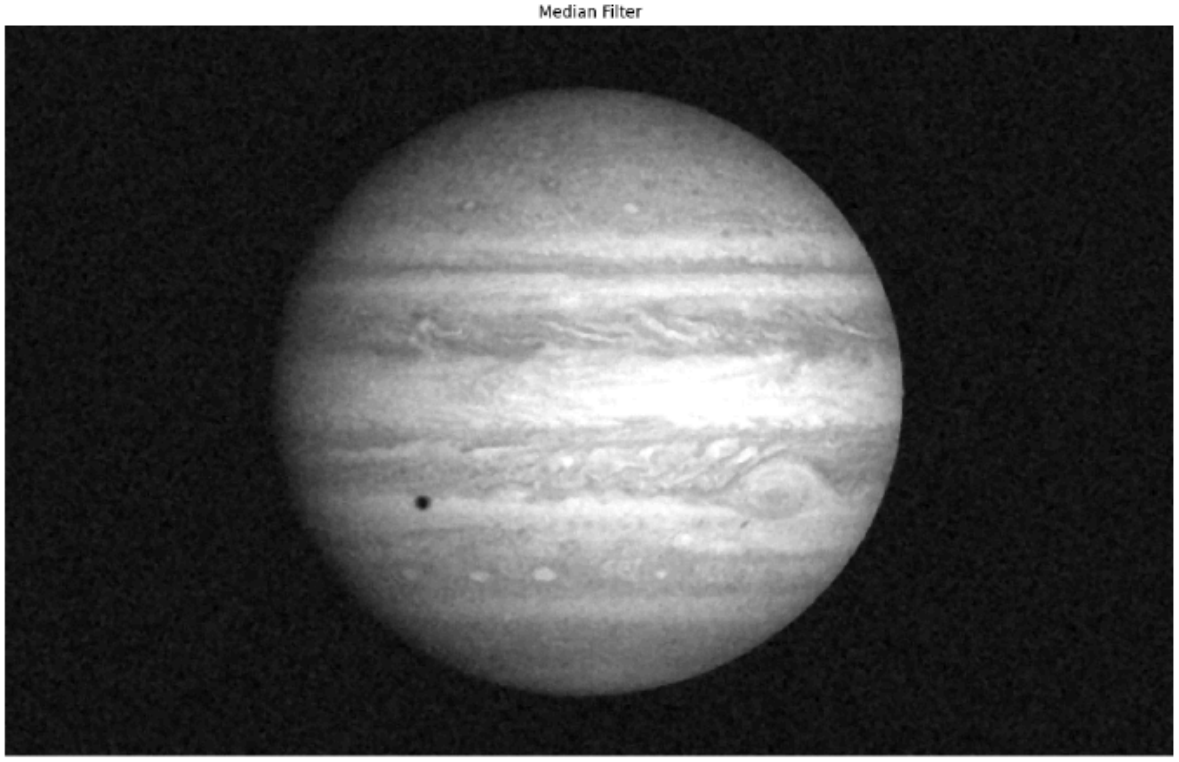


Image 25.

Median Filter on Uniform Noise

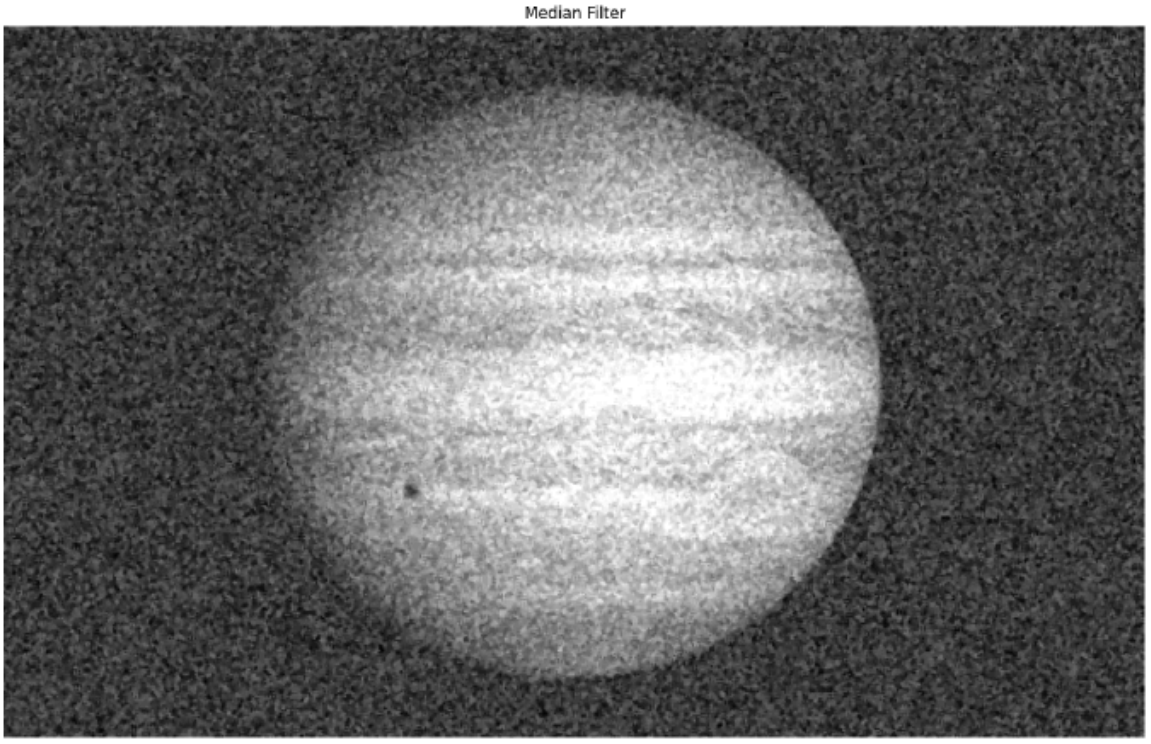


Image 26.

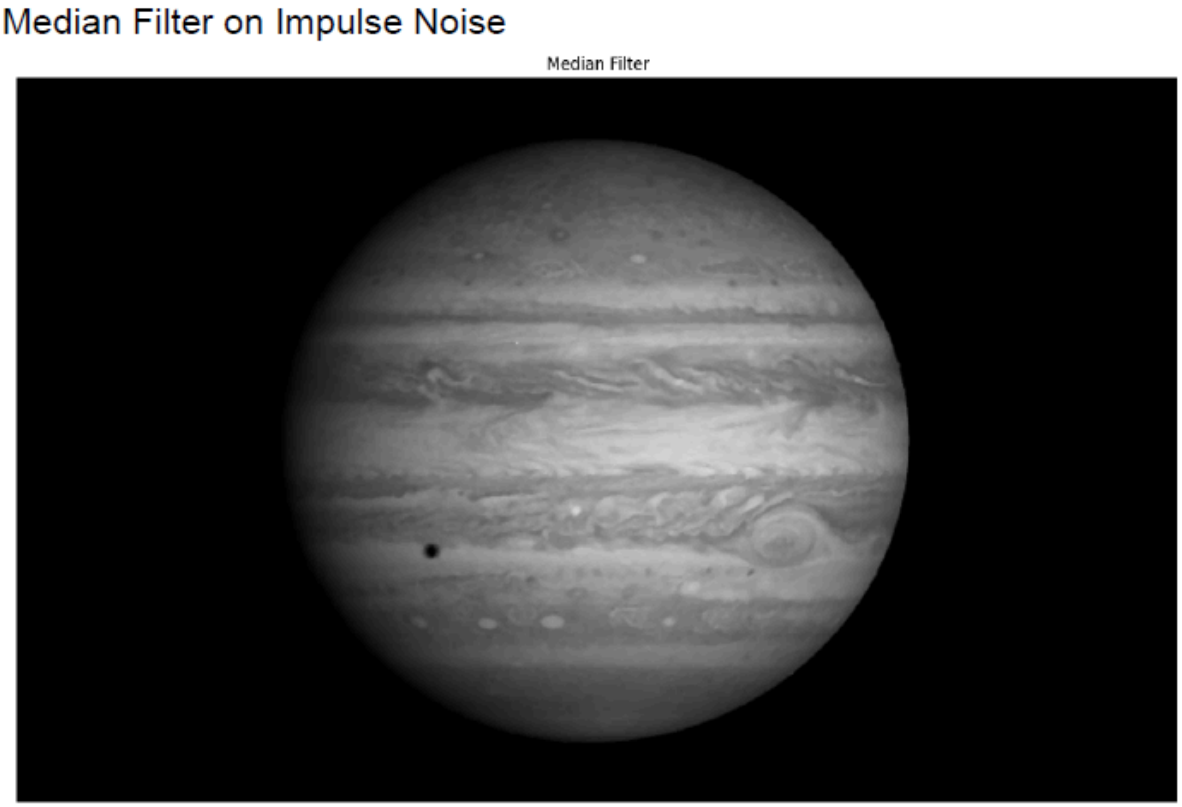


Image 27.

Image 28.

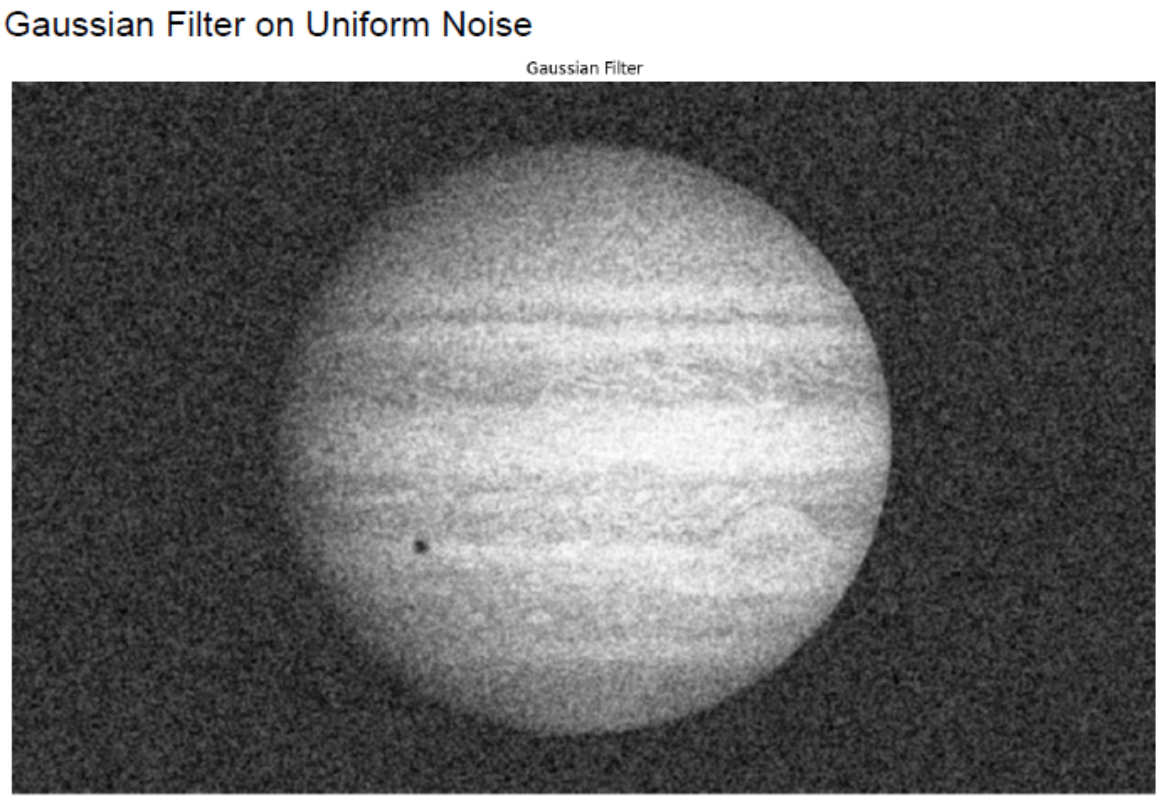


Image 29.

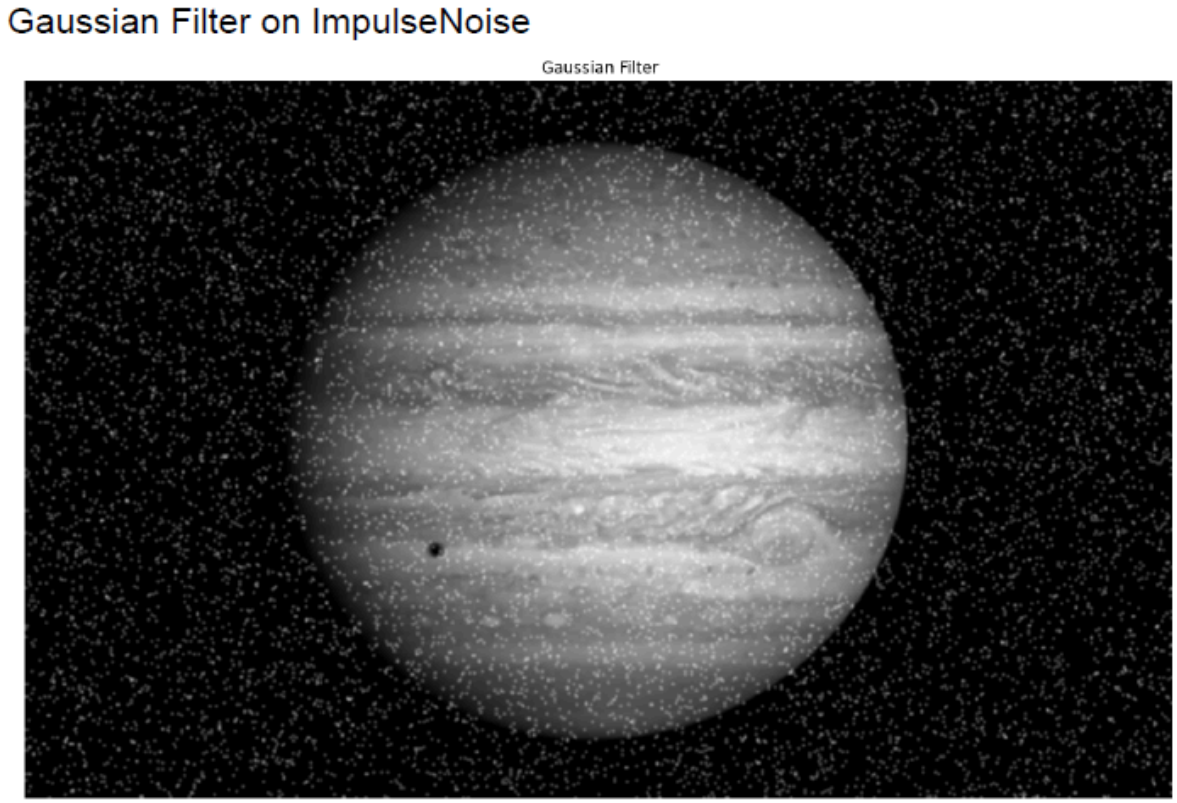


Image 30.

7.2 Detailed review and analysis of selected research studies that have applied machine learning algorithms to solve noisy image classification problems in astronomy

TESTING PHASE DOCUMENTATION

Testing Environment
In the following chapters, machine learning algorithms will be tested, and the outcome of the testing phase will be presented and discussed in a structured format including tables and charts. The data set will be the solar system planets data set and will be the same for all the algorithms and all their variations. The image augmentations will also remain the same for all the testing phases to make the results more realistic and fair. Each model's performance is discussed in detail, highlighting areas where it excels or falls short compared to the others.
Each algorithm and each variation will be tested 10 times and for each one an array with the evaluation metrics will be presented. For comparison, the mean value of the 10 tests will be presented too.

All the Python code can be found on:
https://github.com/gedolks/Diploma-Thesis-Image-Classification-Using-Machine-Learning-Techniques

The libraries imported for all the coding projects are:
import os: For interacting with the operating system.
import numpy as np: For numerical operations.
import tensorflow as tf: For building and training machine learning models.
from tensorflow.keras.preprocessing.image import ImageDataGenerator: For data augmentation.
from tensorflow.keras.models import Sequential, Model: For model creation.
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, BatchNormalization, Input, Conv2DTranspose, GlobalAveragePooling2D, Reshape: Various layers for building neural networks.
from tensorflow.keras.optimizers import Adam: For model training optimization.
from tensorflow.keras.applications import DenseNet121, VGG16 [56], MobileNetV2: Pre-trained models for transfer learning.
from tensorflow.keras.preprocessing.image import load_img, img_to_array: For loading and converting images.
from tensorflow.keras.callbacks import EarlyStopping: For early stopping during training.
import matplotlib.pyplot as plt: For plotting graphs.
import seaborn as sns: For enhanced data visualization.
import pandas as pd: For data manipulation.
import cv2: For image processing.
from sklearn.metrics import confusion_matrix, precision_score, recall_score, f1_score, mean_squared_error, roc_curve, auc, classification_report, accuracy_score: For evaluating models.

from sklearn.preprocessing import label_binarize, LabelEncoder, StandardScaler: For preprocessing data and labels.
from sklearn.model_selection import train_test_split, RandomizedSearchCV: For splitting data and hyperparameter tuning.
from sklearn.ensemble import RandomForestClassifier, BaggingClassifier, AdaBoostClassifier: For ensemble methods.
from sklearn.svm import LinearSVC: For support vector classification.
from sklearn.tree import DecisionTreeClassifier: For decision tree classification.
from sklearn.multiclass import OneVsRestClassifier: For multiclass classification problems.
from scipy.stats import randint: For generating random integers.
from scipy.spatial import distance: For calculating distances between points.
from scipy import interp: For interpolation.
from itertools import cycle: For creating iterators that cycle through a sequence.
We define the directories for storing all images, training images, validation images, and test images, respectively.

7.1.1 CNN

In this section, we explore the development and evaluation of a convolutional neural network (CNN) model designed for the classification of images, utilizing TensorFlow and Keras libraries to implement and assess its performance across various metrics.
Set Up Data Generators
Data Augmentation and Rescaling: The training data generator uses various augmentation techniques like rotation, width shift, height shift, shear, zoom, and horizontal flipping to introduce randomness into the training data, helping to improve the model's generalization. Both the training and test/validation generators rescale images by 1/255 to convert pixel values into a range [0, 1].
Flow from Directory: This method loads images from the specified directories, automatically assigning labels based on folder names and resizing images to 150x150 pixels.
Define the Neural Network Model
The model uses a Sequential architecture, which is a linear stack of layers.
It consists of several convolutional layers (Conv2D), each followed by max pooling layers (MaxPooling2D), which help in reducing the spatial dimensions of the output volume.
After extracting features, the model flattens the output and applies a dropout layer to reduce overfitting.
Finally, it has a dense network ending with a softmax activation function to classify the images into multiple categories.
Compile the Model
The model is compiled with the Adam optimizer and categorical cross-entropy as the loss function since it's a multi-class classification problem.
Train the Model
The model is trained on the data provided by train_generator for a given number of epochs, validating its performance using validation_generator.
Evaluate the Model

Model Predictions: The trained model is used to predict classes on the test set.
Visualizations
Various plots are created to visualize the confusion matrix and ROC curves, helping in a deeper understanding of model performance across different classes.
The results of this simple CNN are:

Iteration	Accuracy	Precision	Recall	Error Rate	F1 Score	MSE
1	0.77	0.78	0.74	0.23	0.71	2.30
2	0.72	0.87	0.84	0.13	0.84	0.90
3	0.83	0.81	0.80	0.17	0.80	2.11
4	0.87	0.85	0.85	0.13	0.84	1.25
5	0.83	0.88	0.82	0.17	0.77	1.40
6	0.79	0.85	0.72	0.21	0.80	2.92
7	0.83	0.81	0.73	0.17	0.74	1.76
8	0.88	0.85	0.78	0.12	0.82	1.11
9	0.78	0.81	0.76	0.22	0.77	2.28
10	0.78	0.81	0.82	0.22	0.78	1.01

Table 5.
and the mean values are:

Iteration	Accuracy	Precision	Recall	Error Rate	F1 Score	MSE
mean value	0.8080	0.8320	0.7860	0.1770	0.7870	1.7040

Table 6.
The confusion matrix of one testing of the simple CNN is:

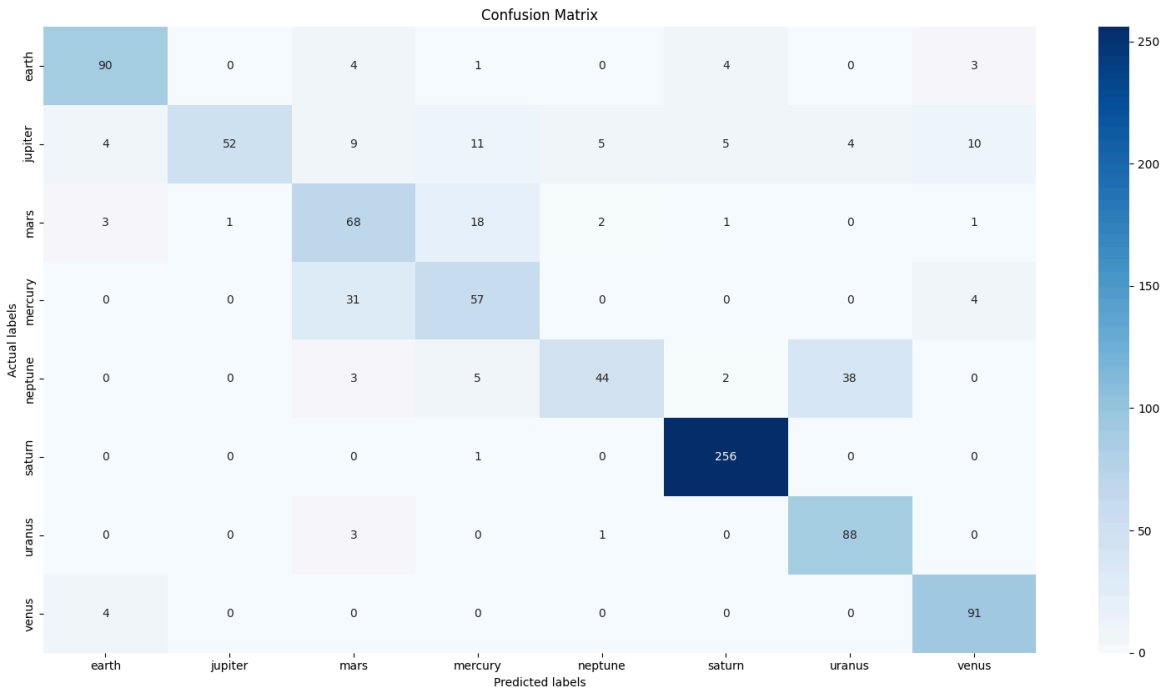


Image 31.
and the ROC curve is:

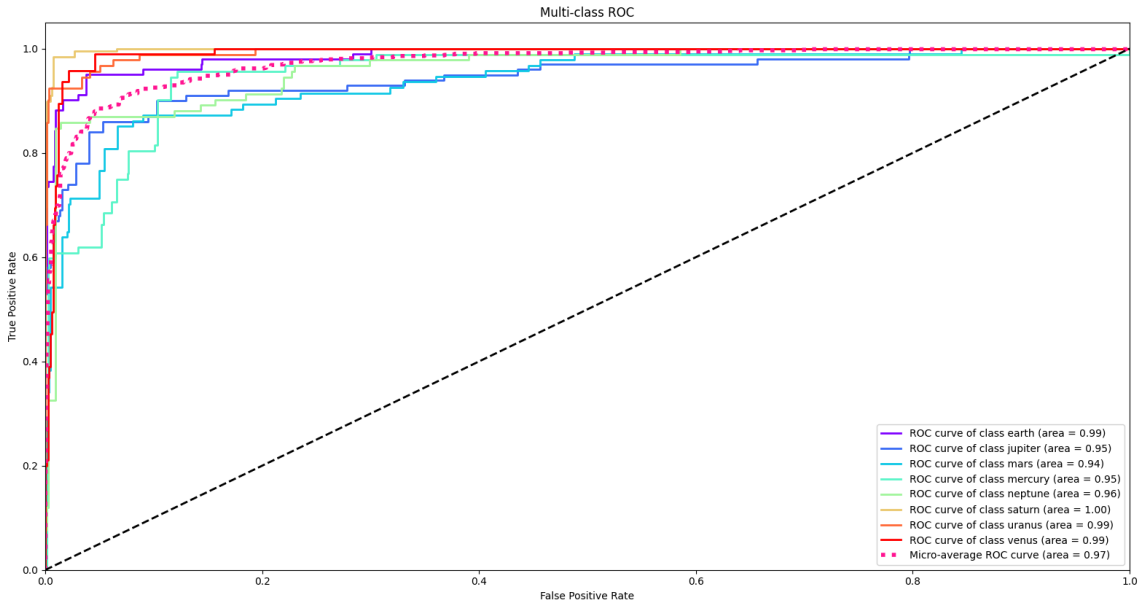


Image 32.

AlexNet

AlexNet has a deeper architecture which included five layers of convolutional layer and three layers of fully connected layers, where max-pooling was used and ReLU as an activation function to avoid the problem of vanishing gradients [74]. Some of the common techniques such as data augmentation and dropout were used to combat overfitting while overlapping pooling was used in order to reduce the size of the network while maintaining its performance.

One of the major breakthroughs was the utilization of GPUs to train larger models which was rather important at that time. Dividing the model into two GPUs made the training faster, which proves that computational power can help in scaling deep network. The use of some complicated data augmentation techniques like image translations, reflections and the alterations of intensity of RGB channels also enhanced the performance of the model.
AlexNet's achievements in cutting down the error rates in image classification problems especially on the large scale datasets such as the ImageNet challenge. The architecture and techniques of NetSOM serve as the basis of the future developments of deep learning and computer vision.

Additions to the previous model
Batch Normalization Layers: Added to the model architecture after certain convolutional layers to normalize the activations, which can help accelerate the training process and improve performance.
New Convolutional Layer Configurations: Changed the kernel sizes and numbers of filters in the convolutional layers, and added more layers, indicating a deeper model architecture.
Increased Number of Dense Layers: Two dense layers with 4096 units each have been added before the output layer.
Changes to the previous model
Modified Convolutional Layers and Pooling Layers: Adjustments in strides, padding, and pooling sizes reflect an optimization or an adaptation to the model's architecture to enhance feature extraction.
Learning Rate Access: Changed how the learning rate is specified for the Adam optimizer (though this usage remains consistent, the approach to set it can vary based on TensorFlow/Keras versions).
The results of the AlexNet are:

Iteration	**Accuracy**	**Precision**	**Recall**	**Error Rate**	**F1 Score**	**MSE**
1	0.87	0.87	0.85	0.12	0.83	0.024
2	0.93	0.94	0.93	0.06	0.93	0.009
3	0.88	0.89	0.86	0.11	0.86	0.024
4	0.96	0.96	0.96	0.04	0.96	0.0115
5	0.93	0.94	0.93	0.06	0.93	0.009
6	0.96	0.96	0.96	0.03	0.96	0.007
7	0.96	0.97	0.96	0.03	0.97	0.007
8	0.94	0.95	0.94	0.05	0.94	0.009
9	0.96	0.96	0.96	0.03	0.96	0.007
10	0.95	0.95	0.95	0.05	0.95	0.008

Table 7.
and the mean values are:

Iteration	Accuracy	Precision	Recall	Error Rate	F1 Score	MSE
mean value	0.934	0.939	0.930	0.058	0.929	0.0115

Table 8.

The confusion matrix of one testing of the AlexNet CNN is:

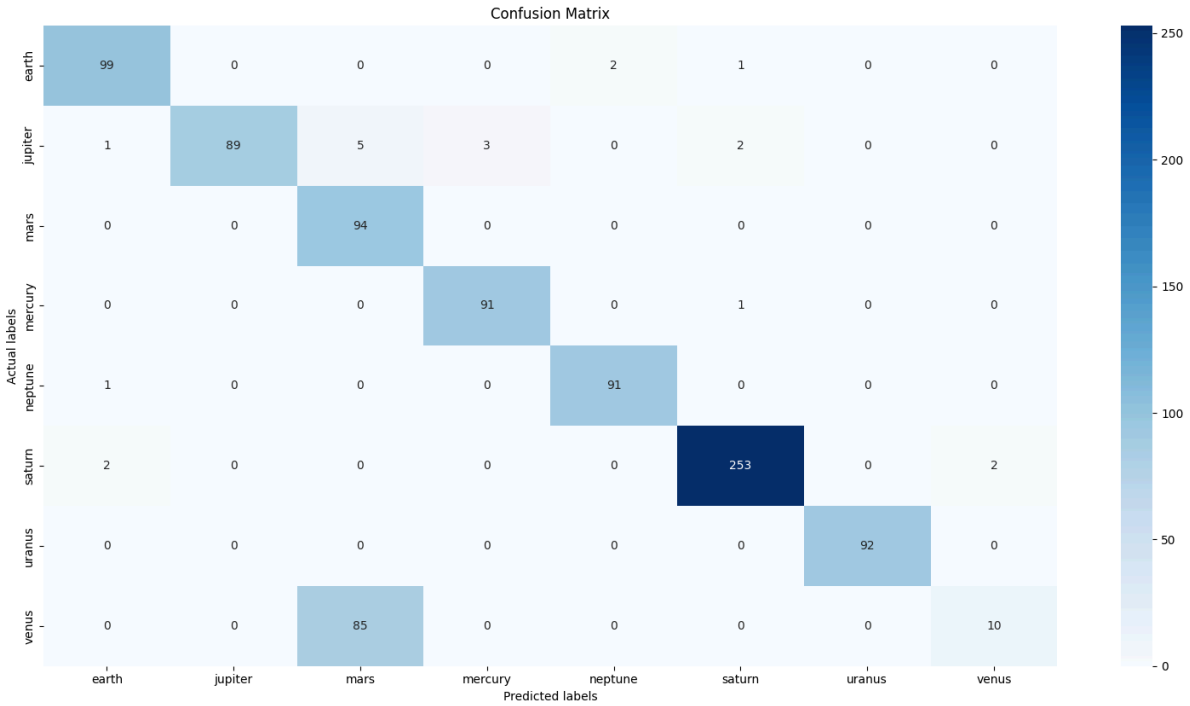


Image 33.

and the ROC curve is:

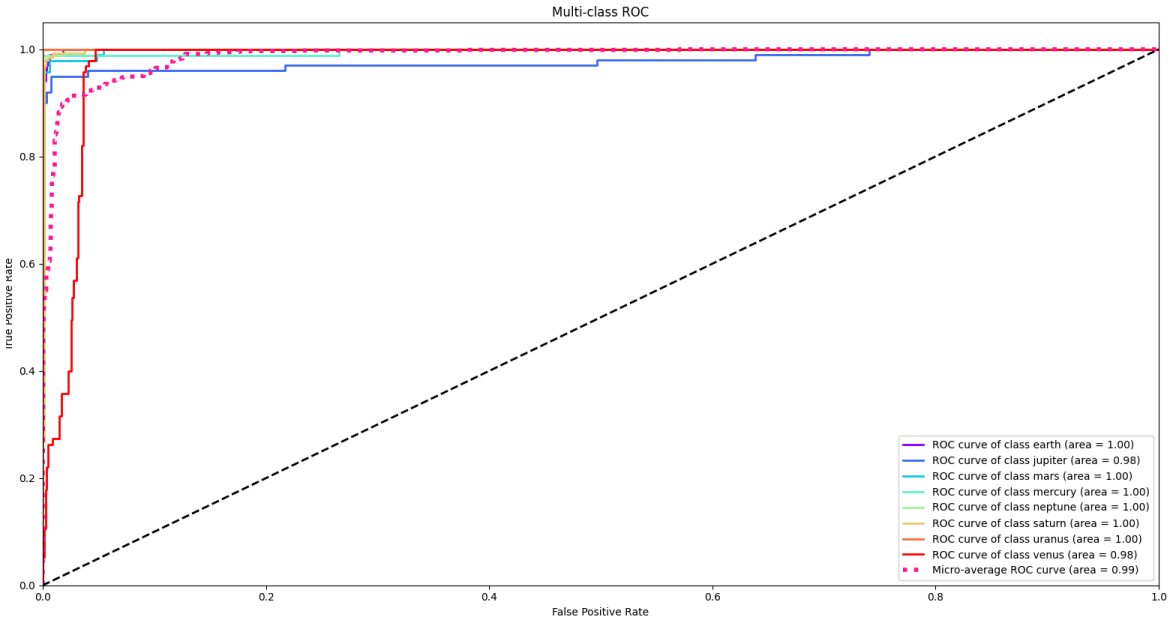


Image 34.

DenseNet

It introduces a new architecture referred to as density connection where all the layers in the network are connected in a densely connected manner in the feed-forward manner [75]. This connectivity enables the reuse of features and thus the network is deeper and more efficient as compared to the normal CNNs. This is because DenseNet does not require the parameters to be learnt for the redundant feature maps which makes it less time consuming.

To create dense blocks in the architecture, we employ composite functions that include Batch Normalization, ReLU, and Convolution. This makes it have less parameters and yet enhance the performance in image classification tasks. The results of the experiments prove that DenseNet outperforms even the most complex networks, such as ResNet, in tasks relating to datasets that include CIFAR-10, CIFAR-100, and SVHN, even with a reduced number of parameters and a lower time consuming process.

Another important benefit of DenseNet is that deeper architectures are less prone to overfitting due to the features reuse of DenseNet. This is especially handy in environments where the use of resources is limited and every effort must be maximized. Memory issues were also discussed by the authors and they suggested an efficient way of implementation while utilizing bottleneck layers to decrease the number of parameters.

This is because of DenseNet's design that not only enhanced state-of-the-art performance in image classification but also led to the development of other efficient network architectures. Its strategy towards the connection and the exploit of features has been long-lasting in the field.

We can apply some of these changes to out model.

Model Architecture:

The new code utilizes DenseNet121, a pre-trained model from TensorFlow's keras.applications module, as opposed to building a custom sequential model with convolutional layers. This shift suggests a move towards utilizing transfer learning, which can provide better generalization and faster convergence in many cases.

Input Size: The input size has been changed from 150x150 to 224x224, which is a typical input size for DenseNet models.

Global Average Pooling: Instead of flattening the convolutional layer outputs directly, the new code uses a GlobalAveragePooling2D layer, which helps with reducing the model's parameters by averaging out the spatial dimensions.

Additions to the previous model

Early Stopping: The new code includes an EarlyStopping callback, which stops training when a monitored metric has stopped improving, thereby preventing overfitting and potentially saving computational resources.

Modification of Trainable Layers: It specifically sets the layers of the DenseNet121 base model to be non-trainable. This is typical in transfer learning scenarios where the base layers are kept frozen while only the newly added layers are trained.

Use of Functional API: The new model is constructed using Keras' Functional API, which provides more flexibility in designing complex models compared to the Sequential API used in the simple NN before.

The results of the DesnseNet algorithm are:

Iteration	**Accuracy**	**Precision**	**Recall**	**Error Rate**	**F1 Score**	**MSE**
1	0.95	0.95	0.95	0.05	0.95	0.60

2	0.96	0.97	0.96	0.04	0.96	0.50
3	0.98	0.98	0.98	0.02	0.98	0.21
4	0.96	0.96	0.96	0.04	0.96	0.59
5	0.97	0.98	0.97	0.03	0.95	0.39
6	0.96	0.97	0.94	0.04	0.95	0.49
7	0.97	0.96	0.94	0.03	0.98	0.29
8	0.98	0.97	0.96	0.02	0.96	0.53
9	0.96	0.96	0.95	0.04	0.96	0.38
10	0.96	0.96	0.97	0.04	0.95	0.43

Table 9.

and the mean values are:

Iteration	**Accuracy**	**Precision**	**Recall**	**Error Rate**	**F1 Score**	**MSE**
mean value	0.9647	0.9667	0.9574	0.0353	0.9598	0.6368

Table 10.

The confusion matrix of one testing of the DenseNet CNN is:

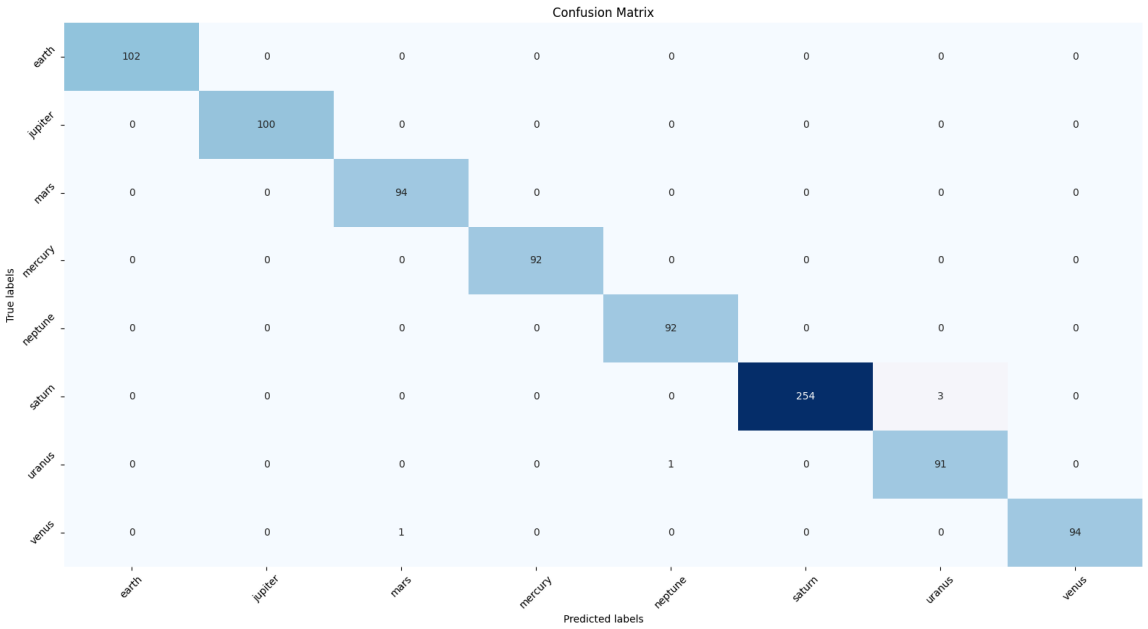


Image 35.

and the ROC curve is:

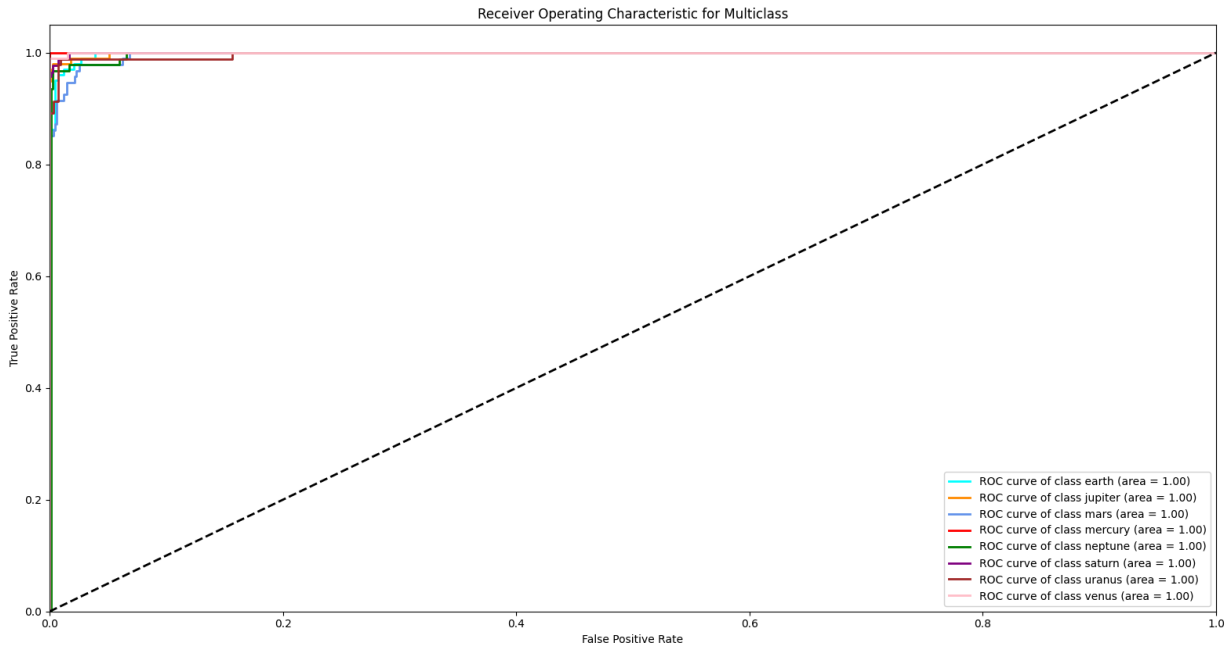


Image 36.

7.1.2 Autoencoder

Autoencoders are a type of neural network used extensively in unsupervised learning tasks, particularly for learning efficient codings of input data. In the context of image classification, autoencoders can play a crucial role in feature learning and dimensionality reduction.

This approach leverages the TensorFlow and scikit-learn libraries to implement and assess the model's performance across various evaluation metrics.

Image Preprocessing

Normalization: To improve model performance, pixel values of the images are scaled to the range [0;1] by converting the images to float type and divided by 255. This particular step assists in the standardization of the training of the autoencoder.

Reshaping for Autoencoder: The images are flattened from two dimensions to one dimensions to enable them feed into the input layer of the autoencoder as required by the architecture of the neural network.

The autoencoder model is constructed using a functional API in TensorFlow. It includes an input layer followed by two dense layers for encoding the input images into lower-dimensional representations. Subsequently, the encoded data is passed through two dense layers to reconstruct the original image dimensions. The model is compiled with the Adam optimizer and mean squared error as the loss function, which is suitable for regression tasks like reconstruction.

Train the Autoencoder

The autoencoder is then trained on the training dataset for ten epochs using the images as both input and output. The training process helps the model to approximate optimal features of images that define their classes.

Encode Images

After the autoencoder is trained, another model is established to obtain the encoded features. This encoded data is then used in the classification step that follows so that the SVM can work on a reduced form of the images.
To better prepare the data for the SVM the features are scaled using the StandardScaler. This step standardizes the data so that each feature has zero mean and unit variance. This is very important especially for algorithms like support vector machine that are influenced by the scale of the input attributes.
The SVM classifier with linear kernel is created and then trained on the encoded features of the training set scaled. This classifier is particularly suitable for the multi-class classification problem and the features extracted from the autoencoder to enhance the classification performance.
The results of the algorithm are:

Iteration	**Accuracy**	**Precision**	**Recall**	**Error Rate**	**F1 Score**	**MSE**
1	0.7846	0.8592	0.6100	0.2154	0.7135	2.5433
2	0.7922	0.7632	0.5800	0.2078	0.6591	2.7976
3	0.7911	0.7308	0.5700	0.2089	0.6404	2.7327
4	0.7240	0.6098	0.5000	0.2760	0.5495	3.7024
5	0.7662	0.5686	0.5800	0.2338	0.5743	3.0563
6	0.7781	0.7778	0.6300	0.2219	0.6961	2.4524
7	0.7846	0.7260	0.5300	0.2154	0.6127	2.7695
8	0.7684	0.7778	0.4900	0.2316	0.6012	2.5671
9	0.7792	0.7042	0.5000	0.2208	0.5848	2.5703
10	0.7424	0.8310	0.5900	0.2576	0.6901	3.4156

Table 11.
and the mean values are:

Iteration	**Accuracy**	**Precision**	**Recall**	**Error Rate**	**F1 Score**	**MSE**
mean value	0.7711	0.7348	0.5580	0.2289	0.6322	2.8607

Table 12.
The confusion matrix of the autoencoder+SVM algortihm is:

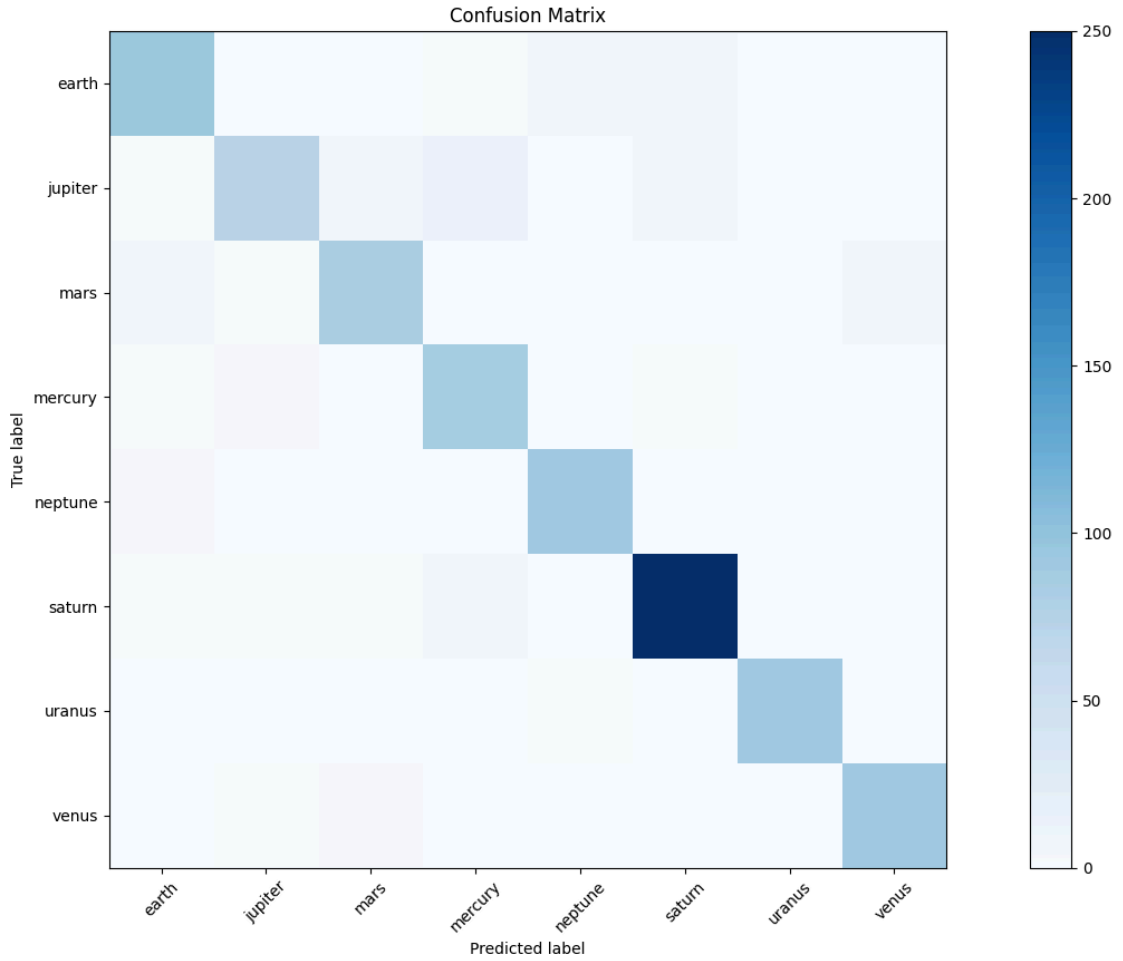


Image 37.

and the ROC curve is:

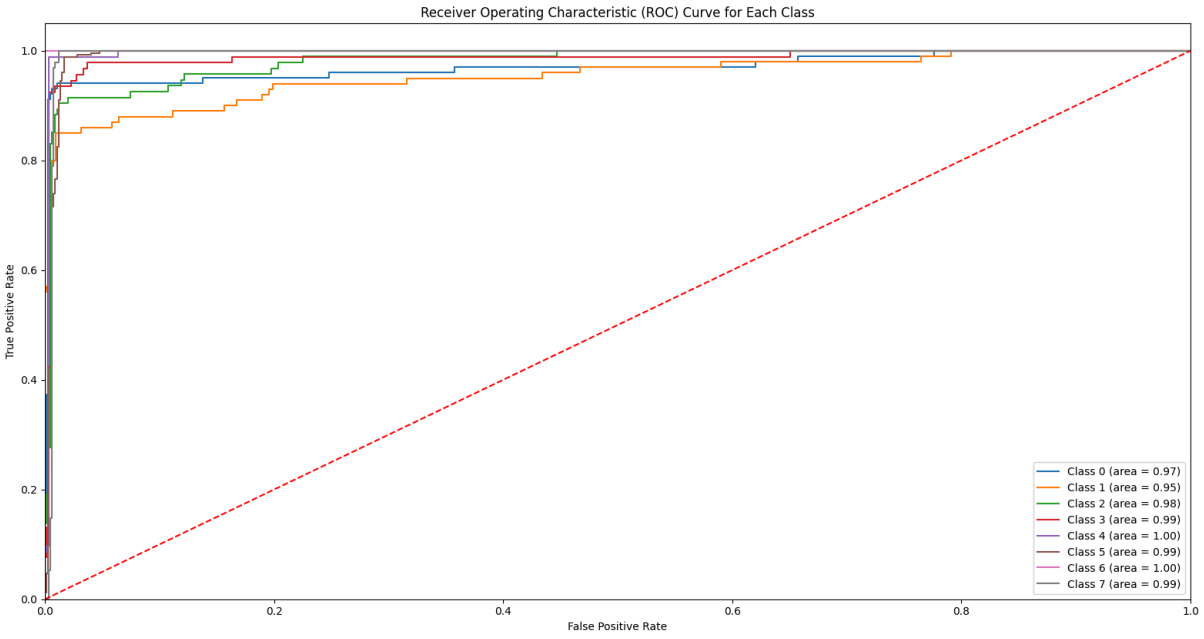


Image 38.

We can use a different classifier than the previous implemetntaion. Instead of a SVM classifier, we can use an advanced Convolutional Neural network. Compared with a more advanced autoencoder, we expect to see better results as far as the evaluation metrics are concerned.
All images are re-sized to 64×64 pixels and normalized by dividing the pixel values by 255 in order to have the pixel values in the range of 0 to 1 which is very useful for the training process. This is important for both autoencoder and CNN to prevent the inputs from having varying ranges that could slow down the learning process.
Autoencoder Architecture
The autoencoder employs convolutional layers to capture features of the images and then map them onto a lower dimensional space. The encoding process consists of two convolutional layers (with 32 and 16 filters) and max-pooling for spatial subsampling. The decoding also follows this pattern based on convolutional layers and upsampling to generate the images. It is trained with Adam optimizer and mean squared error as a loss function which is appropriate for image prediction tasks.
CNN Architecture
The network architecture used in the present work is the Convolutional Neural Network (CNN) that has been sequentially stacked with convolutional layers of filter sizes (32, 64 and 128) and max pooling layers. The final convolutional layer is followed by the fully connected layers; the output of which is fed to two fully connected layers and the last layer uses softmax activation for multi-class classification. The model is trained with Sparse Categorical Cross-Entropy loss so it is ideal to work with integer-labeled target variables.
Model Training
The autoencoder and CNN models are also trained individually. The autoencoder is then trained with 10 epochs and the training set images both as input and output, to learn the sparse coding of images. The CNN is also trained on the same dataset for classification using the encoded features, for 10 epochs as well. It should be noted that both models are trained with a batch size of 32 in order to avoid memory overload while still having efficient computation.

The results of the second algorithm are:

Iteration	**Accuracy**	**Precision**	**Recall**	**Error Rate**	**F1 Score**	**MSE**
1	0.9884	0.9856	0.9876	0.0116	0.9823	0.2902
2	0.9863	0.9843	0.9854	0.0137	0.9843	0.3612
3	0.9827	0.9826	0.9835	0.0173	0.9856	0.2634
4	0.9868	0.9878	0.9825	0.0132	0.9878	0.3265
5	0.9879	0.9887	0.9835	0.0121	0.9867	0.2923
6	0.9890	0.9856	0.9878	0.0110	0.9873	0.2845
7	0.9897	0.9867	0.9898	0.0103	0.9886	0.2978
8	0.9845	0.984	0.9898	0.0155	0.9845	0.3053
9	0.9872	0.9890	0.9867	0.0128	0.9823	0.2914
10	0.9823	0.9898	0.9834	0.0177	0.9823	0.2783

Table 13.
and the mean values are:

Iteration	Accuracy	Precision	Recall	Error Rate	F1 Score	MSE
mean value	0.98648	0.98641	0.9860	0.01335	0.98473	0.2991

Table 14.

The confusion matrix is:

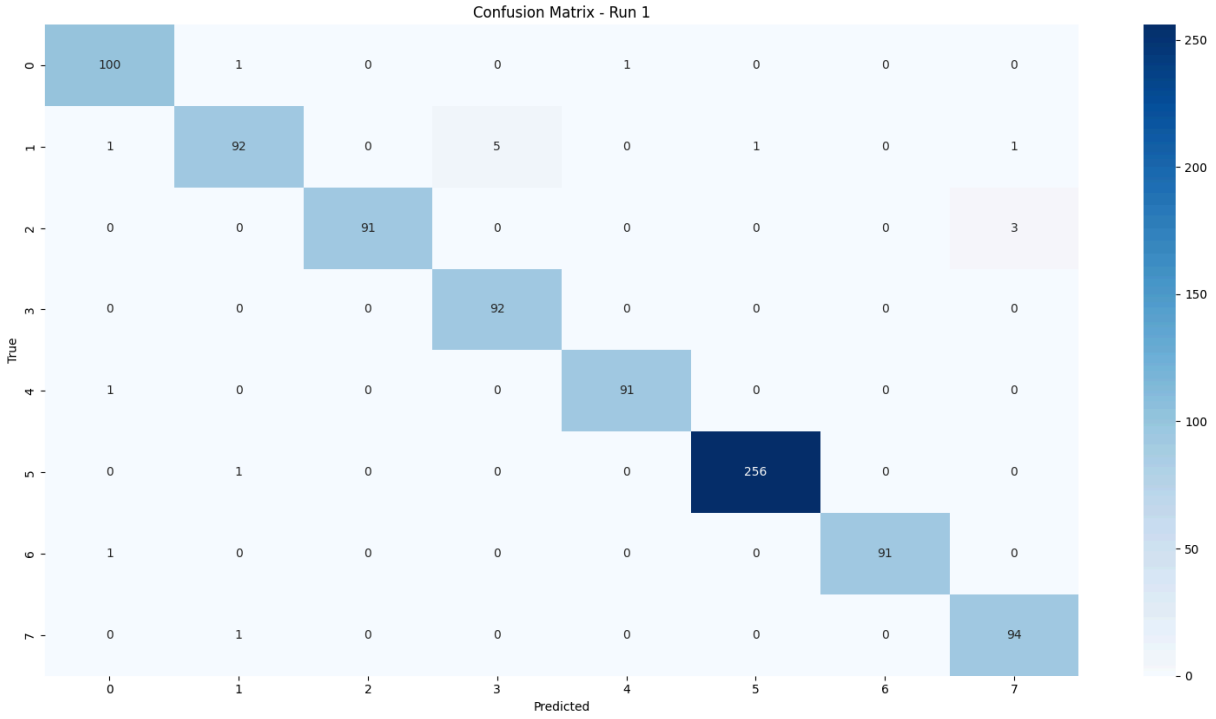


Image 39.
The ROC curve of the second Autoencoder is:

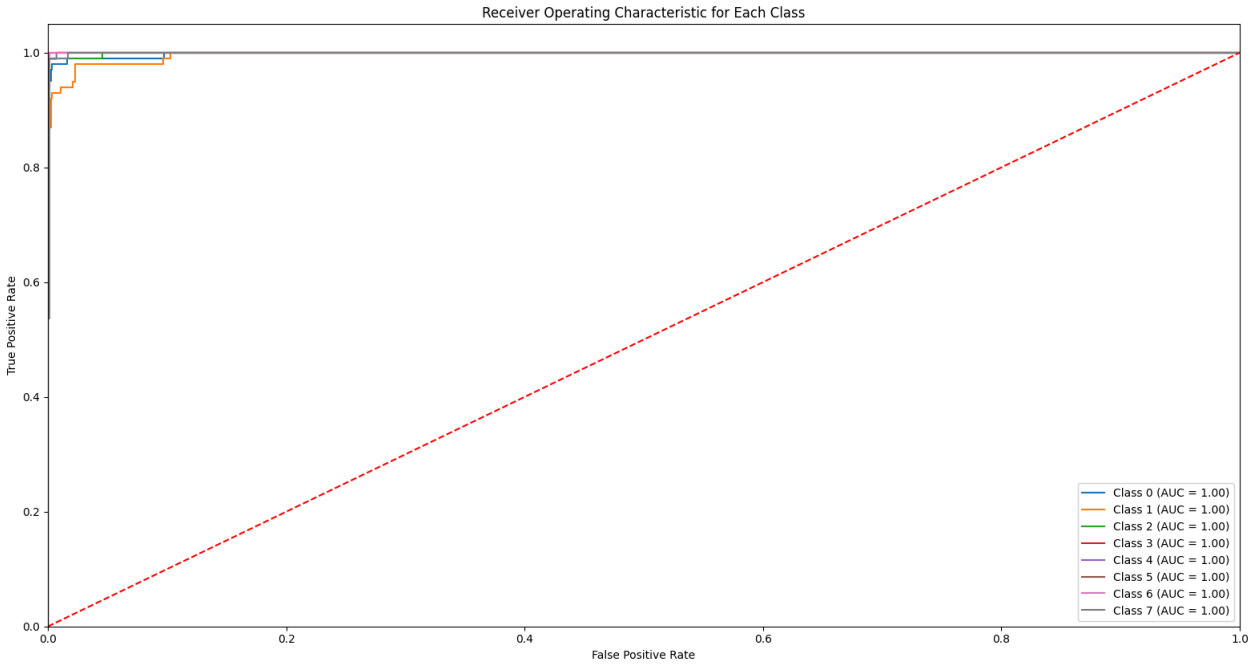


Image 40.

7.1.3 K-Nearest Neighbors

K-nearest neighbors (KNN) is a straightforward and widely used algorithm for classification and regression tasks in machine learning. It operates on the principle that similar data points are often close together. In KNN, 'K' represents the number of nearest neighbors to consider when making a prediction. The algorithm works by finding the 'K' closest data points (or neighbors) to a given query point and then uses these neighbors to determine the outcome. For classification, KNN assigns the class most common among the neighbors, while for regression, it calculates the average of the neighbors' values. KNN is easy to implement and understand, making it a popular choice for many introductory machine-learning applications.

Preprocessing Images
Images are converted to grayscale, resized to a uniform size, and flattened into one-dimensional arrays.
The function also maps class labels to numerical values, making them easier to handle in the model.
The function returns arrays of image data and corresponding numerical labels, along with a mapping of the original labels.
A simple KNN classifier is implemented:
The classifier is initialized with a specified number of neighbors (k). Then, the training method stores the training data and labels. The prediction method calculates distances between each test sample and all training samples, identifies the nearest neighbors, and assigns the most common label among these neighbors as the prediction.
Training the KNN Model

An instance of the KNN classifier is created and trained using the training data. This step involves storing the necessary information for making predictions on new data.

Predicting the Test Set

The trained KNN model is used to predict the labels of the test dataset. This involves calculating distances, identifying the nearest neighbors, and determining the most frequent label among them for each test sample.

Due to the deterministic nature of the KNN algorithms, we will not test it 10 times, but just once and the results will also be the mean value. A deterministic algorithm is an algorithm that produces the same results every time it is run, given the same input. This consistency is due to the algorithm's lack of randomness in its logic—it follows a set series of steps to arrive at a result based on the input provided.

The results of this simple implementation of the KNN are:

Iteration	**Accuracy**	**Precision**	**Recall**	**Error Rate**	**F1 Score**	**MSE**
mean value	0.9123	0.9142	0.9123	0.0876	0.9128	1.066

Table 15.

and the confusion matrix is :

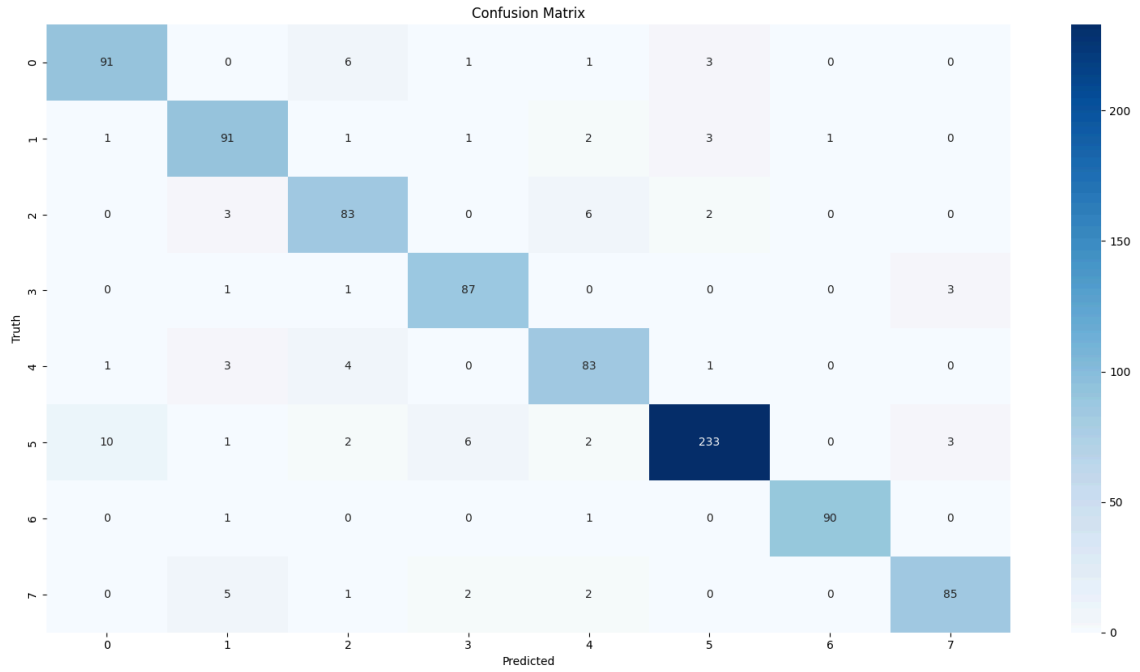
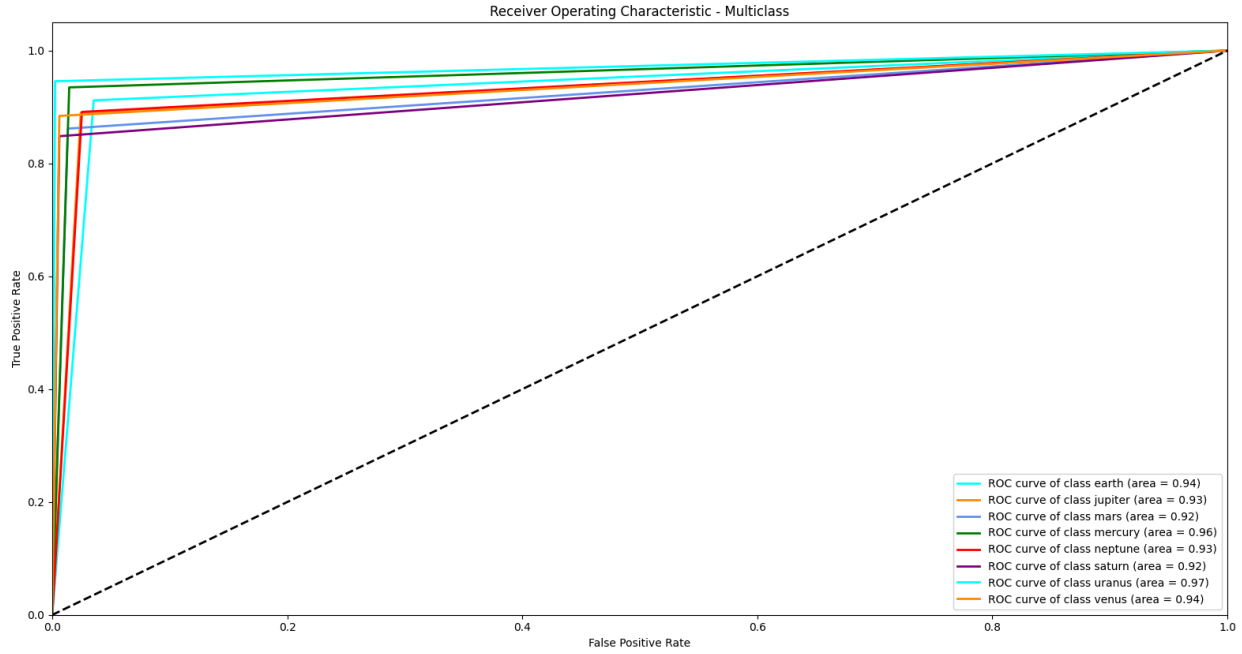


Image 41.

and the ROC curve is:



One way to enhance the k-nearest neighbor classifier (k-NNC) is through the use of weights to the neighbors which is done by using Gaussian distribution in contrast to linear interpolation This approach improves the accuracy by applying non-parametric density estimate via Gaussian kernel density estimator [62]. The method does not depend on common space or classification time reduction schemes and it has been found to be more accurate in different sets of data.

In this weighted k-NNC strategy, the classifier give more weights to the neighbors that are nearer to the decision area. The steps include:

Setting the classifier with a fixed value of neighbors which is k.

Setting the training data and the labels for the data.

In the test sample, calculating the Euclidean distances to all the training points and then selecting the k nearest neighbours which are then assigned the inverse distance weight in order to avoid division by zero.

Summing up the weighted votes to make the final decision and come up with the final prediction.

This technique enhances the efficiency of k-NNC in machine learning applications and at the same time enhances the accuracy of the results.

Dudani's method

Dudani's method enhances the k-nearest neighbor (k-NN) algorithm by introducing a distance-based weighting system, where closer neighbors have more influence on the classification of a query point. The formula used to calculate the weight for each neighbor is:

$Wi = \frac{dmac - di}{dmax - dmin}$, where di is the distance from the i-th neighbor to the query point, while dmax and dmin are the maximum and minimum distances among the k neighbors [63].

This system ensures that neighbors farther from the query point contribute less to the classification, addressing a key limitation in the traditional k-NN algorithm, where all neighbors are treated equally.

Dudani's weighted voting mechanism has significantly impacted the evolution of k-NN algorithms and has been widely cited in research exploring further improvements in weighting strategies, showcasing its enduring influence in machine learning.
For the implementation of the Dudani Weighted KNN Classifier:
Initialization: Similar to the WeightedKNN class, initialized with a specified k.
Training (fit method): Stores the training data and labels.
Prediction (predict method): Computes Euclidean distances between the test sample and all training samples.
Identifies the k nearest neighbors and their distances.
Computes weights using Dudani's formula, which linearly scales weights based on the distances of the nearest and farthest neighbors.
Aggregates the weighted votes to determine the final prediction.
The results of the last two implementations(weighted KNN and Dudani KNN) are:
For the weighted KNN:

Iteration	**Accuracy**	**Precision**	**Recall**	**Error Rate**	**F1 Score**	**MSE**
mean value	0.9134	0.9095	0.9120	0.0865	0.9094	0.8387

Table 16.
and the confusion matrix is :

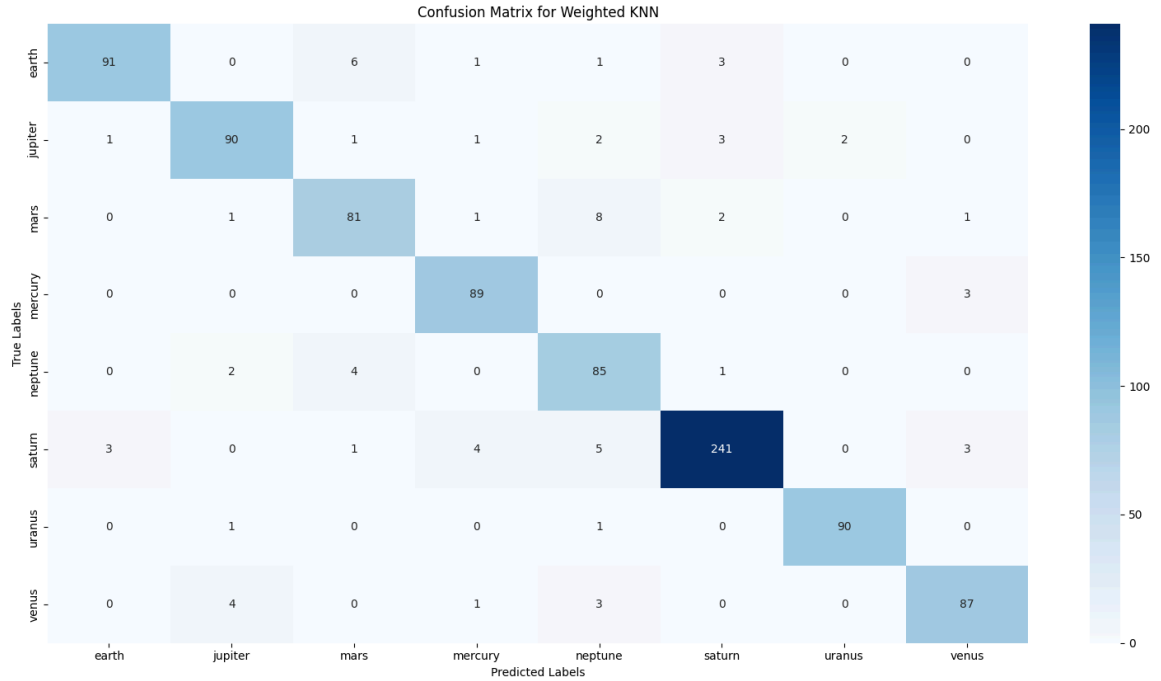
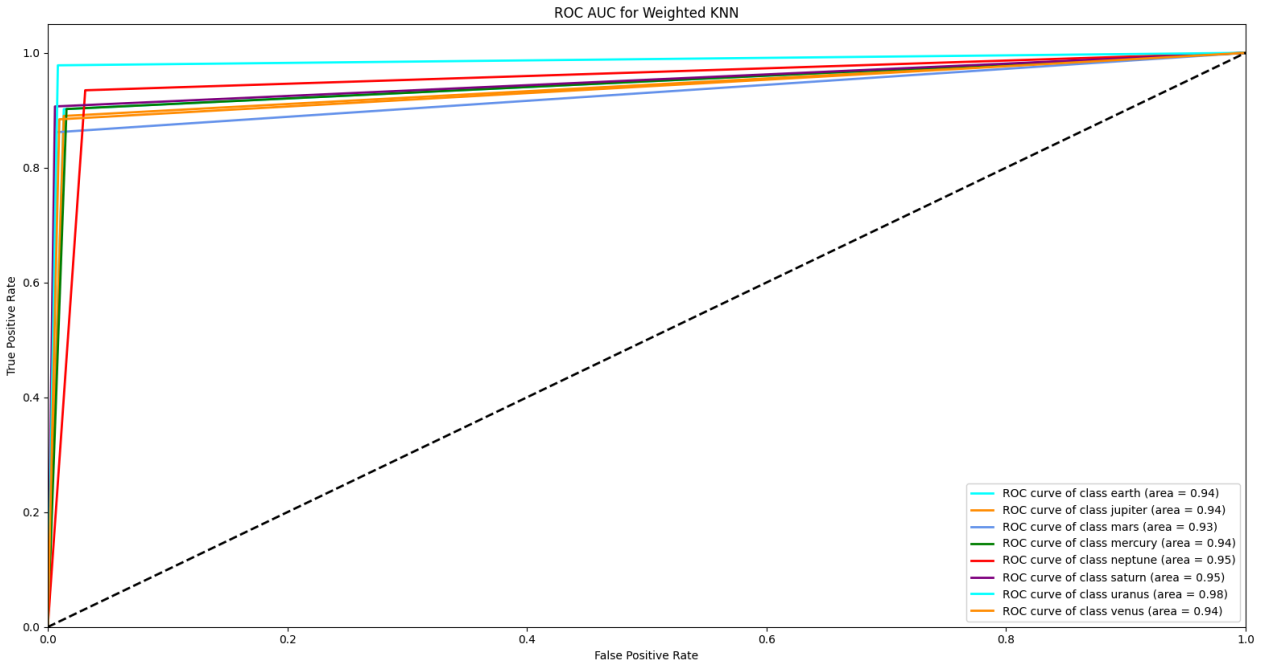


Image 42.

and the ROC curve is:



and for the Dudani KNN the reults are:

Iteration	Accuracy	Precision	Recall	Error Rate	F1 Score	MSE
mean value	0.9242	0.9182	0.9221	0.0757	0.9193	0.7727

Table 17.

The confusion matrix for the Dudani algorithm is:

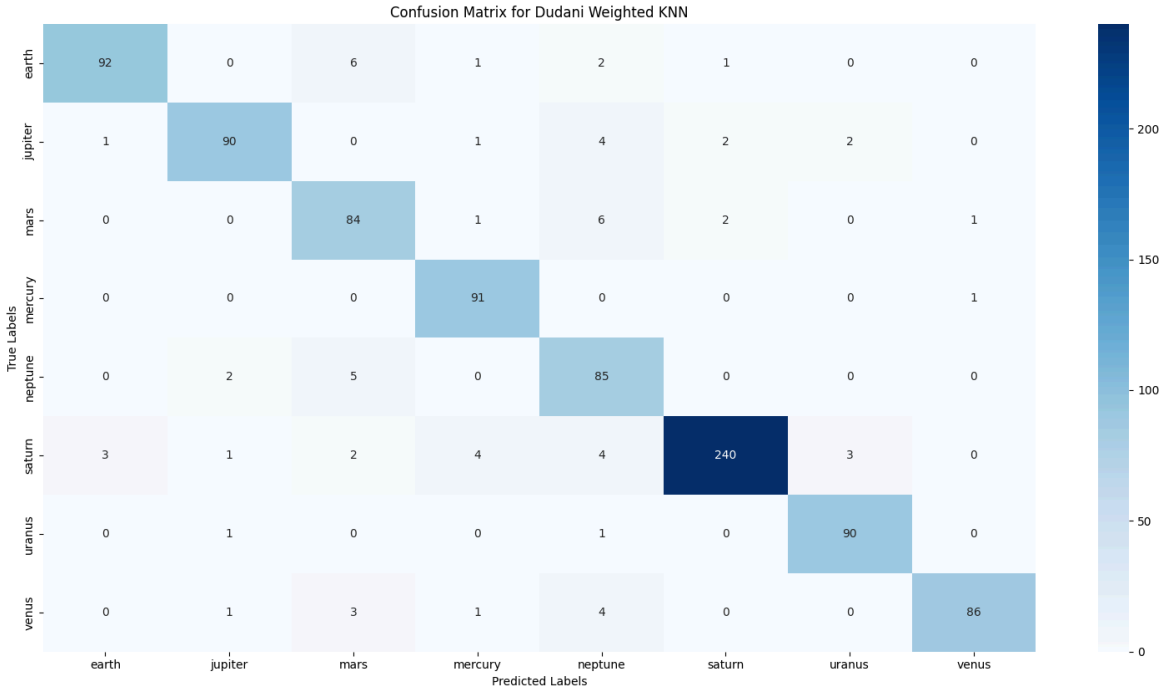
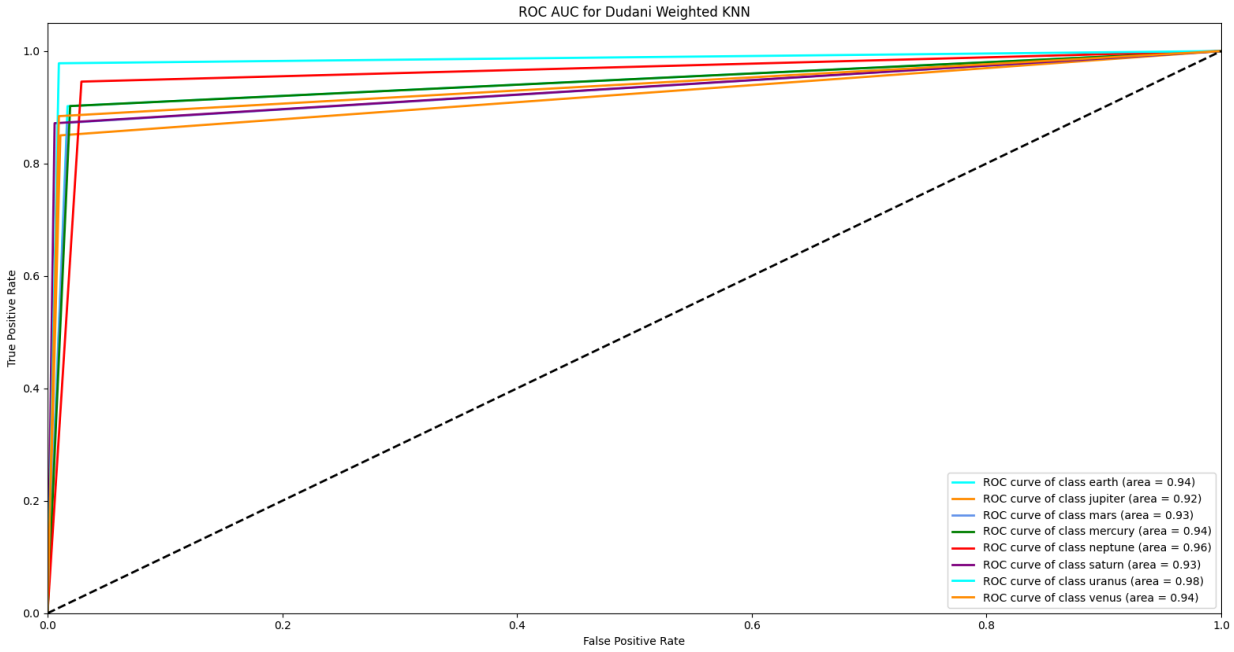


Image 43.

and the ROC curve is:



7.1.4 Decision Trees

Decision trees are a type of supervised learning algorithm used for both classification and regression tasks. They work by recursively splitting the data into subsets based on the values of input features, creating a tree-like model of decisions. Each internal node represents a "test" on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label or a continuous value.

Load Images and Labels from a Directory

Defines a function to load images and their labels from a specified directory and iterates through each label directory, loads images in grayscale, resizes them to 64x64 pixels, flattens them into a 1D array, and appends the images and labels to respective lists. It finally converts the lists to NumPy arrays and returns them.

Train the Decision Tree Classifier

Initializes a DecisionTreeClassifier instance. Trains the classifier using the combined training data and labels.

Predict on the Test Set

Uses the trained classifier to make predictions on the test data.

Encode Labels to Numeric Values

Initializes a LabelEncoder instance. Encodes the true and predicted labels to numeric format

The results of this simple Decision Tree are:

Iteration	Accuracy	Precision	Recall	Error Rate	F1 Score	MSE
1	0.86	0.87	0.86	0.13	0.86	1.74
2	0.87	0.88	0.87	0.12	0.87	1.64
3	0.86	0.86	0.86	0.13	0.86	1.69
4	0.86	0.86	0.86	0.13	0.86	1.64

5	0.86	0.87	0.86	0.13	0.87	1.74
6	0.86	0.87	0.86	0.13	0.86	1.65
7	0.87	0.87	0.87	0.12	0.87	1.52
8	0.87	0.88	0.87	0.12	0.87	1.58
9	0.87	0.88	0.87	0.12	0.87	1.62
10	0.87	0.88	0.87	0.12	0.87	1.55

Table 18.

Iteration	**Accuracy**	**Precision**	**Recall**	**Error Rate**	**F1 Score**	**MSE**
mean value	0.8689	0.8759	0.8689	0.1310	0.8702	1.6416

Table 19.

The ROC curve is:

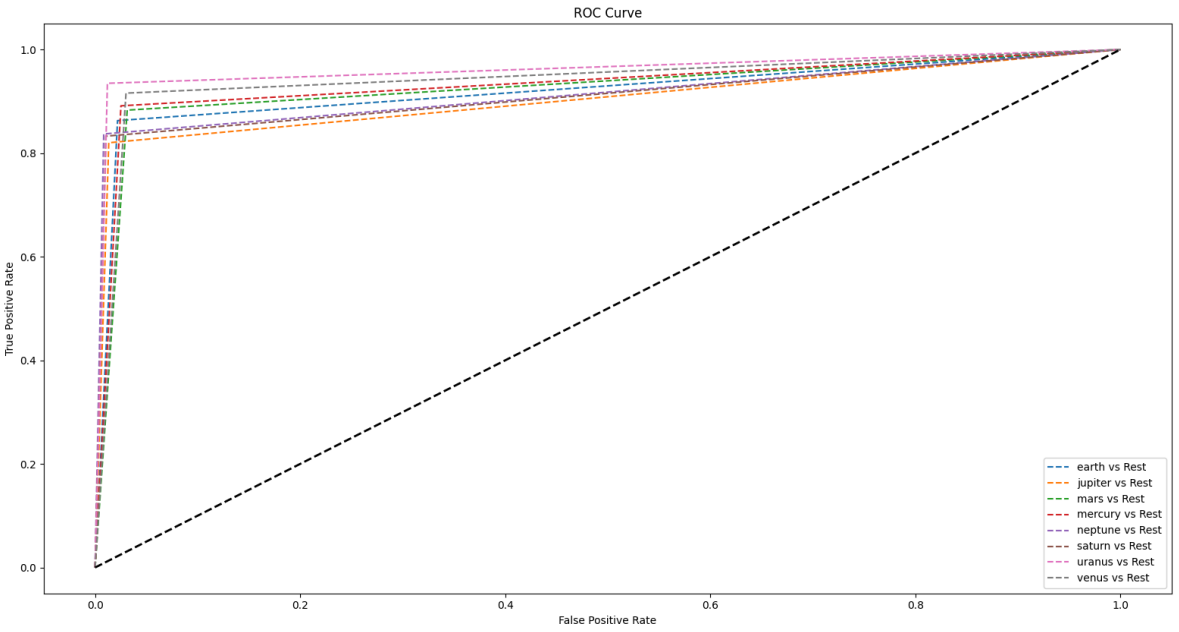


Image 44.

and the confusion matrix is:

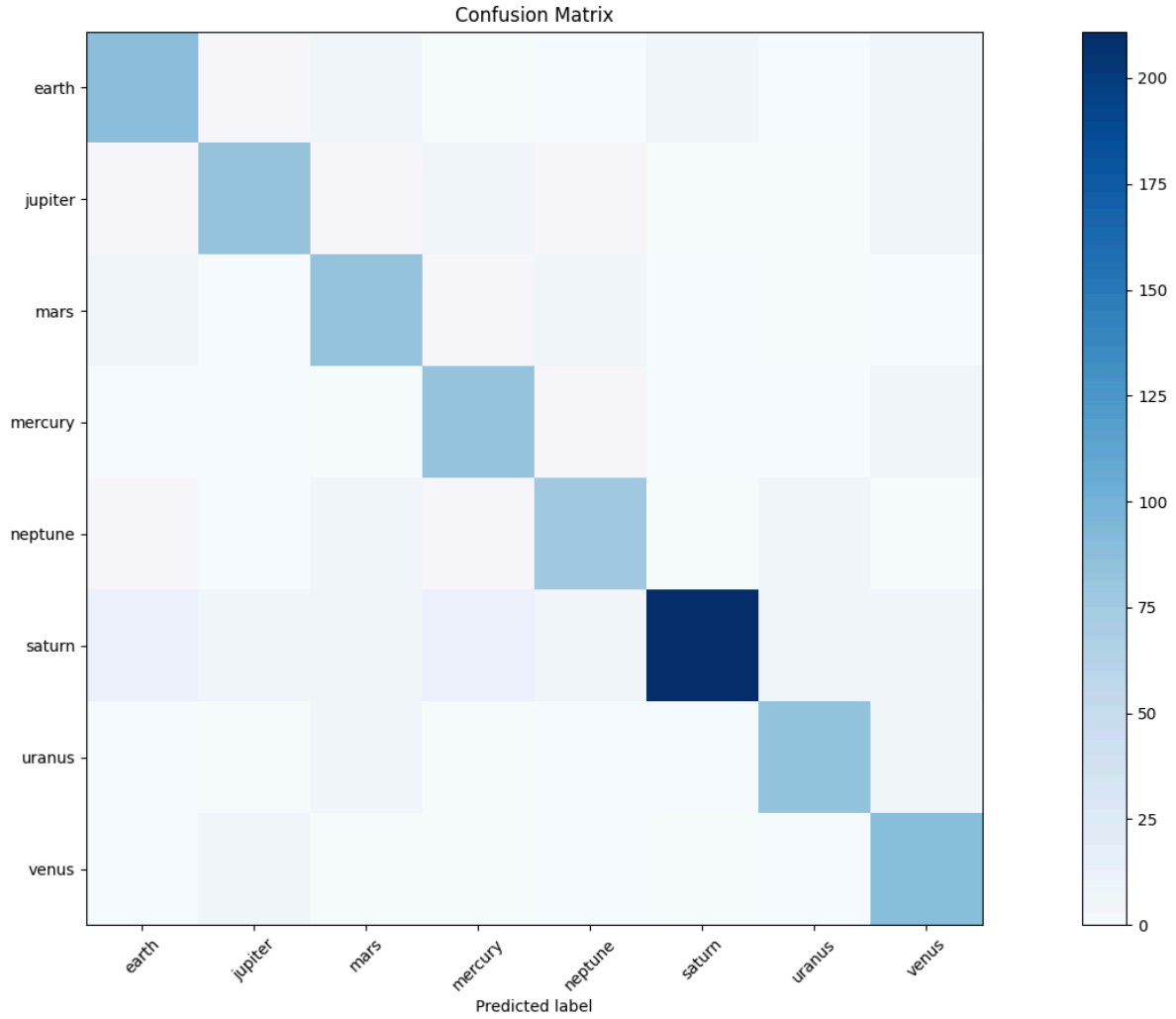


Image 45.

Bagging and Boosting in Decision Tree-Based Classifiers

Bagging (Bootstrap Aggregating) and boosting are the strategies of ensemble learning that help improve the decision tree-based classifiers as to enhance the accuracy and stabilization [64].
Bagging Concept: It entail training of many models with different bootstrap samples that are randomly selected from the original data set with replacement. Thus, the predictions are combined so that instead of having one output, there are several and they are obtained through a majority vote or averaging to enhance the reliability and decrease the variability.
Implementation: Sometimes used with decision tree based classifiers to develop strong classifiers like the random forests where several trees are developed and the results of the trees are combined to make a final decision.
Boosting Concept: Trains models one at a time with each new model being built to learn from the mistakes of the previous models. This means that weights of misclassified samples are enhanced in order to give more importance to the challenging ones.
Implementation: Some algorithms include AdaBoost which changes the weight of the samples to enhance the solution's accuracy in each iteration, Real AdaBoost and MultiBoost which are improvements of the boosting method by integrating other methods.

Application to Risk Prediction: These techniques are advantageous especially in the risk prediction models for instance in the medical field for analysis of diagnosis [95]. They operate by employing a number of decision tree based classifiers in a style that allows for the identification of numerous patterns inside the data.
Performance and Impact: It has been seen that ensemble methods are more accurate as compared to individual classifiers. Bagging leads to the decrease of variance and boosting leads to the better performance of the model by paying attention to more complicated instances. Both of them have been reported to improve the performance of classification in different tasks including disease risk prediction for example diabetes.
Future Directions:
More advanced fusion strategies like stacking, and the combination of classification and clustering methods might also bring enhancements in the measurement of predictive performance and time efficiency.
Training the Classifier
The Bagging implementation uses a BaggingClassifier with a DecisionTreeClassifier as the base estimator. This ensemble method trains multiple decision trees on different subsets of the training data to improve model performance and robustness.
Predictions
In the Bagging code, predictions on the test set are made directly without additional encoding steps.
In the Bagging Classifier, random_state is set to 42. This makes the initialization of the random processes within the Bagging Classifier deterministic. Thus, running this part of the script multiple times with the same data and setup should yield the same results each time.
The results of the Bagging code are:

Iteration	**Accuracy**	**Precision**	**Recall**	**Error Rate**	**F1 Score**	**MSE**
mean value	0.9134	0.9149	0.9134	0,0866	0.9138	1.2089

Table 20.
The ROC curve is:

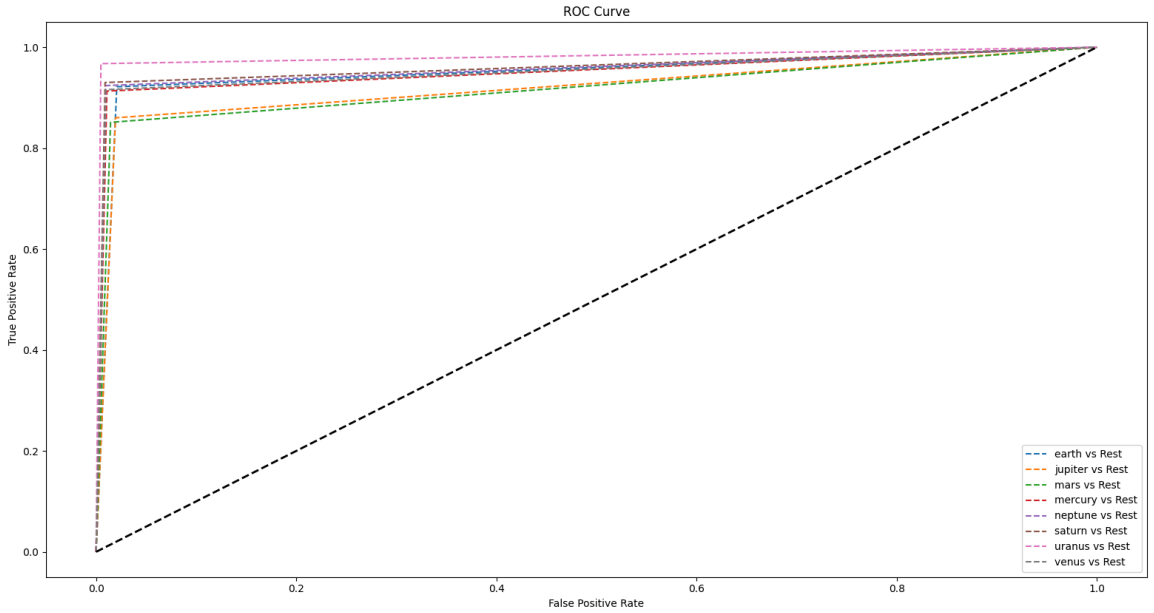


Image 46.

and the confusion matrix is:

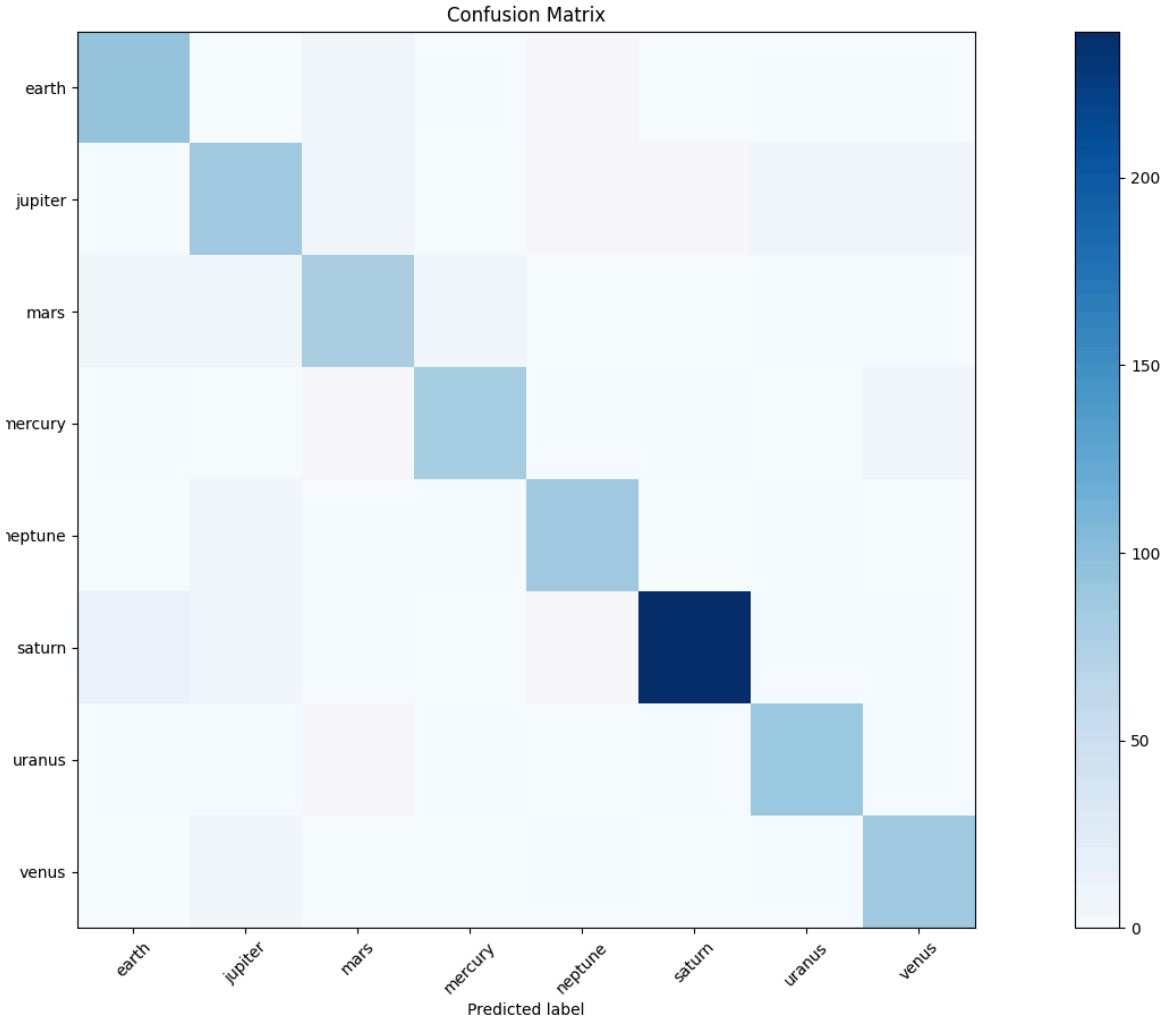


Image 47.

7.1.5 Random Forest

Building upon the foundational understanding of decision trees, we can enhance predictive accuracy and robustness through the implementation of random forests. While a single decision tree may suffer from overfitting and variance, random forests mitigate these issues by constructing multiple trees and aggregating their outputs.

Function for Loading Images: A function is defined to load images and their respective labels from a given directory. The images are read in grayscale, resized to a fixed size of 64x64 pixels, flattened, and stored in arrays along with their labels.

Loading and Combining Datasets: The training, validation, and test datasets are loaded using the previously defined function. After loading, the training and validation sets are combined to form a more comprehensive training dataset.

Training the Random Forest Classifier: An instance of the RandomForestClassifier is created and trained using the combined training data. The classifier is then used to predict the labels for the test dataset.

Label Encoding and Evaluation: The LabelEncoder from Scikit-learn is used to convert the categorical labels into numeric form, both for the true labels and the predicted labels.

The results of the simple Random Forest algorithm are:

Iteration	Accuracy	Precision	Recall	Error Rate	F1 Score	MSE
1	0.93	0.93	0.93	0.06	0.93	1.00
2	0.93	0.93	0.93	0.06	0.93	0.83
3	0.93	0.93	0.93	0.06	0.93	0.83
4	0.93	0.93	0.93	0.06	0.93	0.83
5	0.93	0.93	0.93	0.06	0.93	0.80
6	0.93	0.93	0.93	0.06	0.93	0.87
7	0.92	0.92	0.92	0.07	0.92	0.92
8	0.92	0.93	0.92	0.07	0.92	0.92
9	0.93	0.93	0.93	0.06	0.93	0.91
10	0.93	0.93	0.93	0.06	0.93	0.79

Table 21.

Iteration	Accuracy	Precision	Recall	Error Rate	F1 Score	MSE
mean value	0.93291	0.93319	0.93291	0.06709	0.93284	0.87533

Table 22.

The ROC curve is:

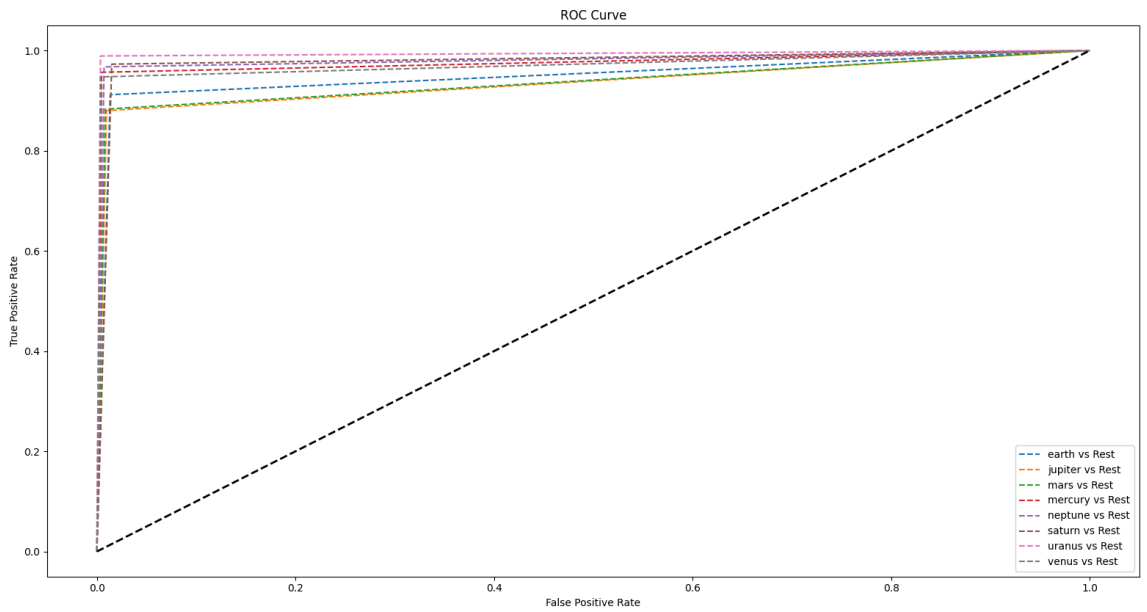


Image 48.

and the confusion matrix is:

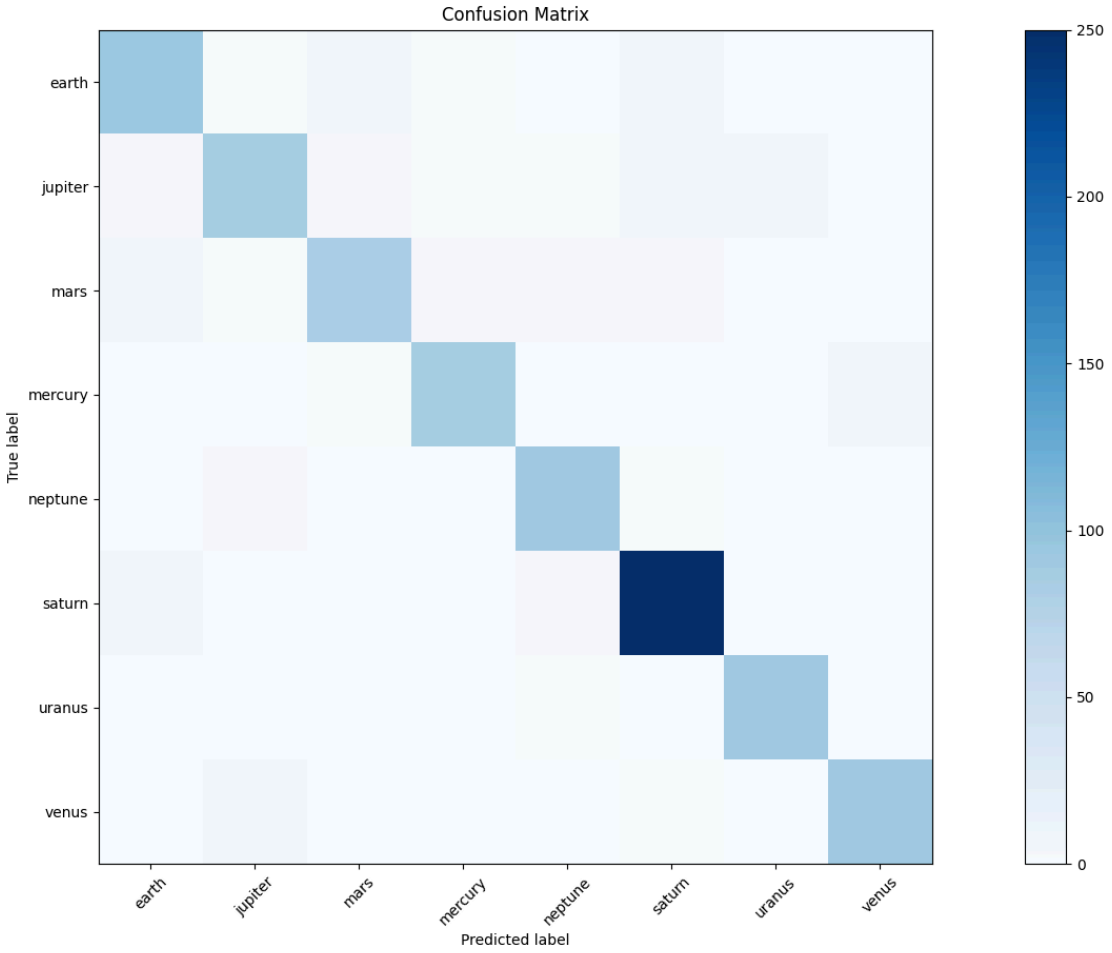


Image 49.

We can change some parameters in the previous code.
Feature Scaling: A new step of feature scaling has been introduced using StandardScaler. This is critical when using models sensitive to the magnitude of features, such as those involving distance calculations. Scaling ensures that all features contribute equally to the model training, which is particularly important when images have varying light conditions or contrasts
The script now includes hyperparameter tuning using RandomizedSearchCV from sklearn.model_selection. This process optimizes the RandomForestClassifier by experimenting with a range of parameter values (n_estimators, max_depth, etc.) and uses cross-validation to evaluate these configurations. This method aims to find the most effective model settings to enhance performance and robustness. To facilitate the randomized search, a parameter distribution is defined using randint and explicit lists, providing a structured approach to exploring a range of possible values for the RandomForest parameters.
Model Training: The training of the RandomForestClassifier now leverages the optimized parameters found by RandomizedSearchCV, which potentially leads to better model accuracy and generalization compared to using default parameters.
Evaluation Metrics Calculation: The calculation of metrics now explicitly includes the conversion of labels to a one-hot encoded format using pandas.get_dummies for the calculation of ROC-AUC, adapting the approach to handle multi-class classification scenarios effectively.

Data Preprocessing: Previously, the script directly used the combined training data for model training. Now, this data is scaled before being used, ensuring the model does not bias towards features with larger scales.
Over all, The script efficiently handles the entire process of setting up directories, loading and preprocessing data, including image resizing and flattening. It then scales features, optimizes model parameters, and trains the classifier using these optimized settings. Post-training, the script continues to handle label encoding efficiently, calculates various performance metrics, and visualizes results through ROC curves and confusion matrices. These visualizations are key for evaluating model performance across different classification thresholds and understanding the model's prediction dynamics.
The results of the upgraded script are:

Iteration	**Accuracy**	**Precision**	**Recall**	**Error Rate**	**F1 Score**	**MSE**
1	0.94	0.94	0.94	0.05	0.94	0.76
2	0.93	0.93	0.93	0.06	0.93	0.86
3	0.94	0.94	0.94	0.05	0.94	0.80
4	0.93	0.93	0.93	0.06	0.93	0.82
5	0.93	0.93	0.93	0.06	0.93	0.93
6	0.93	0.93	0.93	0.06	0.93	0.86
7	0.93	0.93	0.93	0.06	0.93	0.77
8	0.93	0.93	0.93	0.06	0.93	0.85
9	0.93	0.93	0.93	0.06	0.93	0.81
10	0.93	0.93	0.93	0.06	0.93	0.84

Table 23.

Iteration	**Accuracy**	**Precision**	**Recall**	**Error Rate**	**F1 Score**	**MSE**
mean value	0.93622	0.93640	0.93621	0.06378	0.93619	0.83368

Table 24.
The ROC curve is:

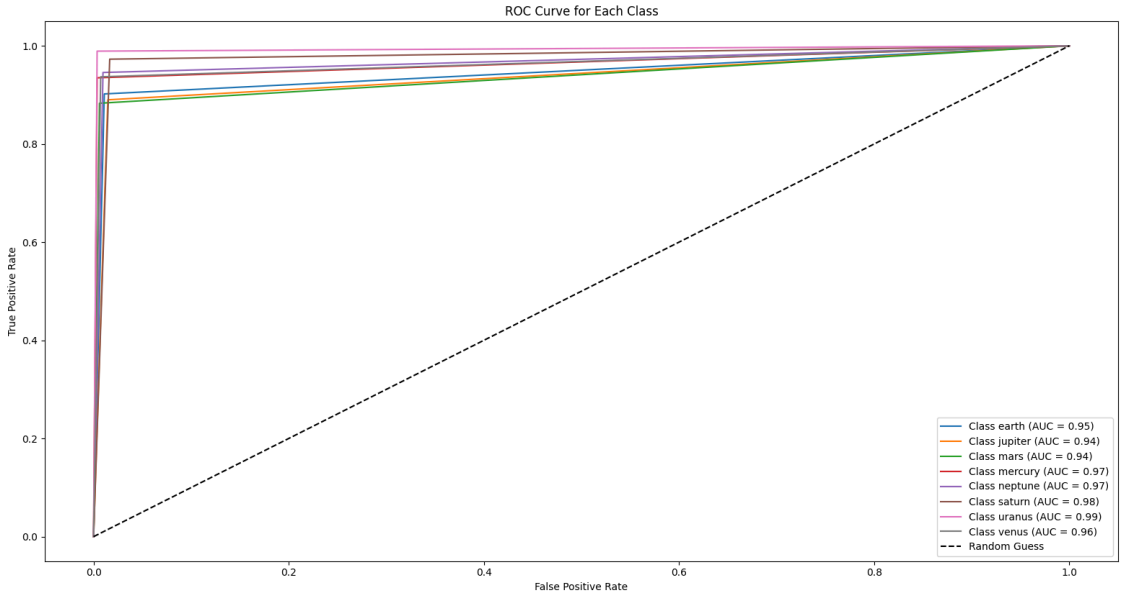


Image 50.

and the confusion matrix is:

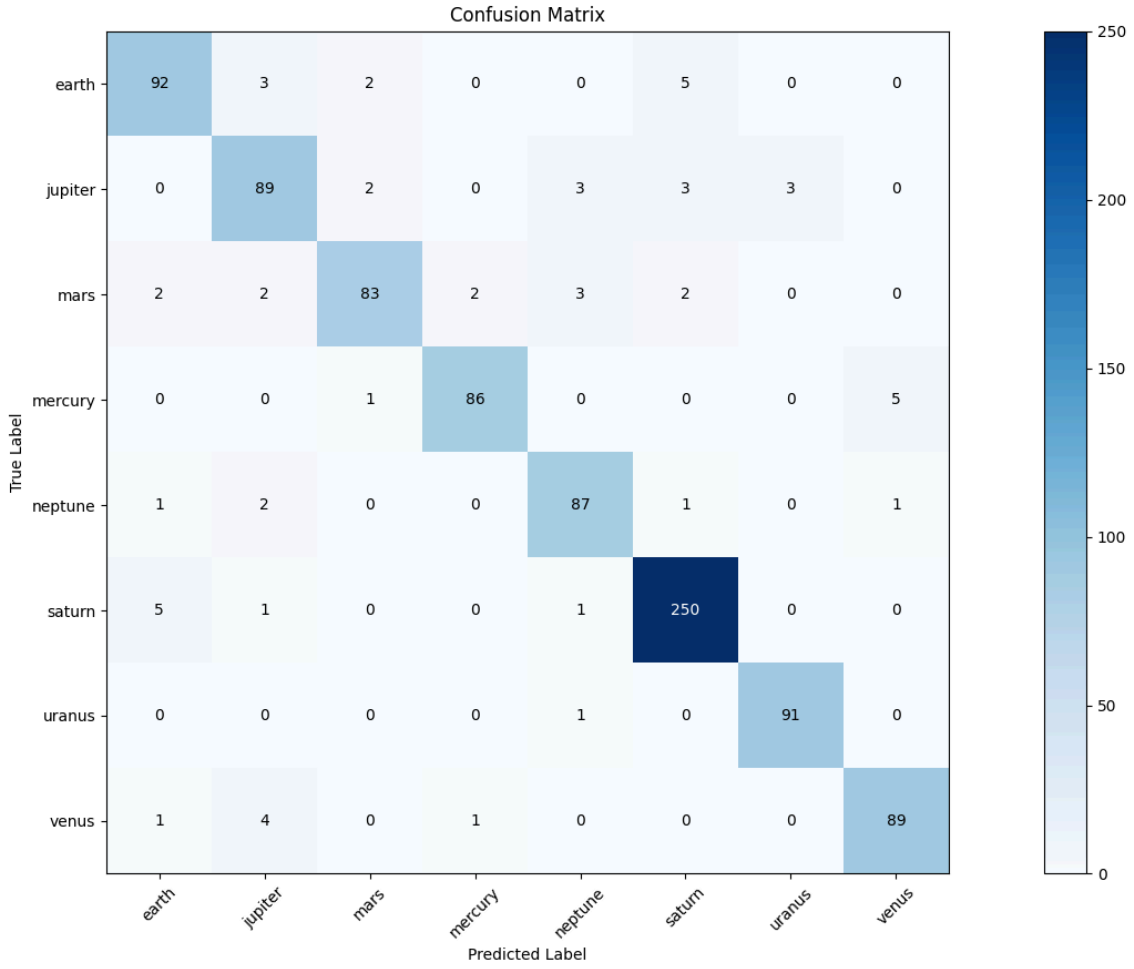


Image 51.

7.1.6 SVM

Support Vector Machines (SVMs) are widely used in image classification tasks, especially in scenarios involving multiple classes. In a multi-class SVM setup, the goal is to categorize images into one of several predefined categories. This is achieved by constructing a hyperplane in a high-dimensional space that maximally separates the classes based on their image features.
We can implement an SVM in the Python environment following these steps:
Image Loading Function:
The load_images_and_labels function is defined to load images and their respective labels from a given directory. Images are read, resized to 224x224 pixels to match the VGG16 input requirements, and stored in arrays along with their labels.
Feature Extraction Using VGG16:
The extract_features function uses the VGG16 model pre-trained on ImageNet to extract features from the images. VGG16 is a convolutional neural network that is effective for image classification tasks. The features are extracted from the 'fc2' layer of the VGG16 model.
Loading and Preprocessing Data:
The training, validation, and test datasets are loaded using the load_images_and_labels function.
The labels are encoded into numeric values using the LabelEncoder from Scikit-learn.
Features are extracted from the images using the VGG16 model.
Training the SVM Model:
An instance of the svm.SVC classifier is created with a linear kernel and probability estimation enabled.
The classifier is trained using the extracted features from the training data.
Model Prediction and Evaluation:
The trained SVM model predicts the labels for the test dataset.
Evaluation metrics are calculated to assess the model's performance.
ROC and AUC Calculation:
ROC curves and AUC scores are calculated to evaluate the classifier's performance. If the labels are not binary, an appropriate error message is displayed.
Visualization:
The script plots the ROC curve if applicable and displays the confusion matrix to provide visual insights into the model's performance.
Due to the deterministic nature of the SVM algorithms, we will not test it 10 times, but just once and the results will also be the mean value. A deterministic algorithm is an algorithm that produces the same results every time it is run, given the same input. This consistency is due to the algorithm's lack of randomness in its logic—it follows a set series of steps to arrive at a result based on the input provided.
The results of the SVM algorithm are:

Iteration	Accuracy	Precision	Recall	Error Rate	F1 Score	MSE
mean value	0.9502	0.9448	0.9395	0.0497	0.9398	0.8116

Table 25.

The confusion matrix is:

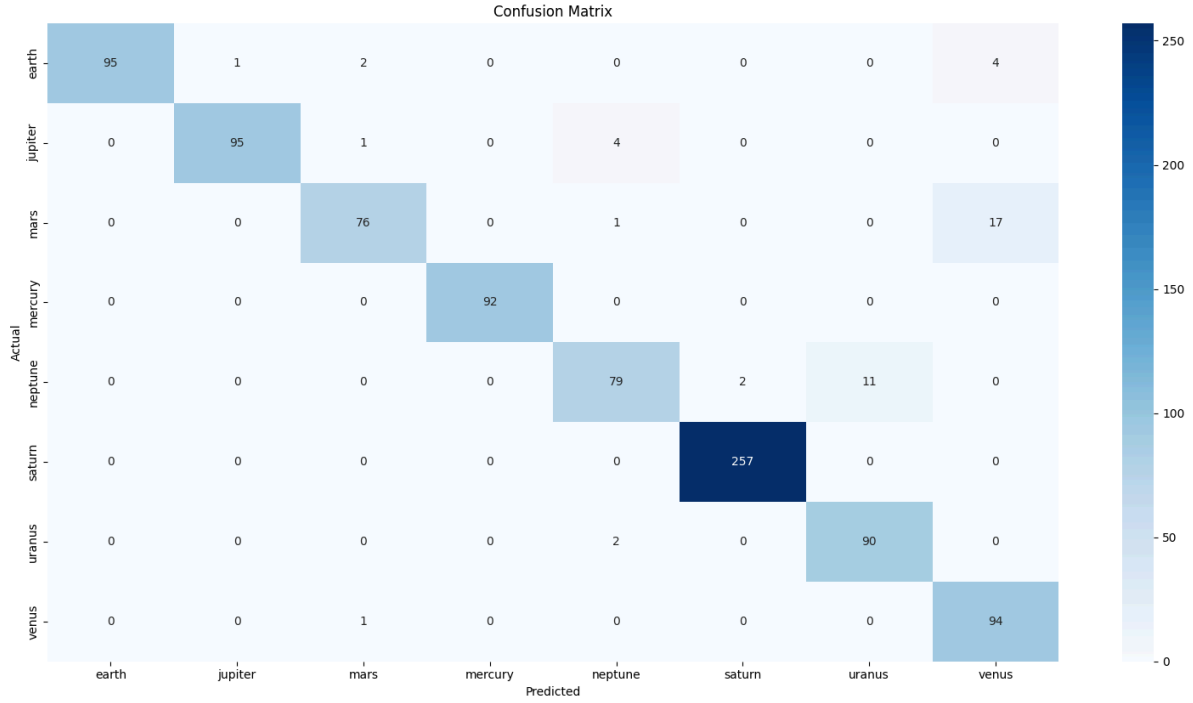
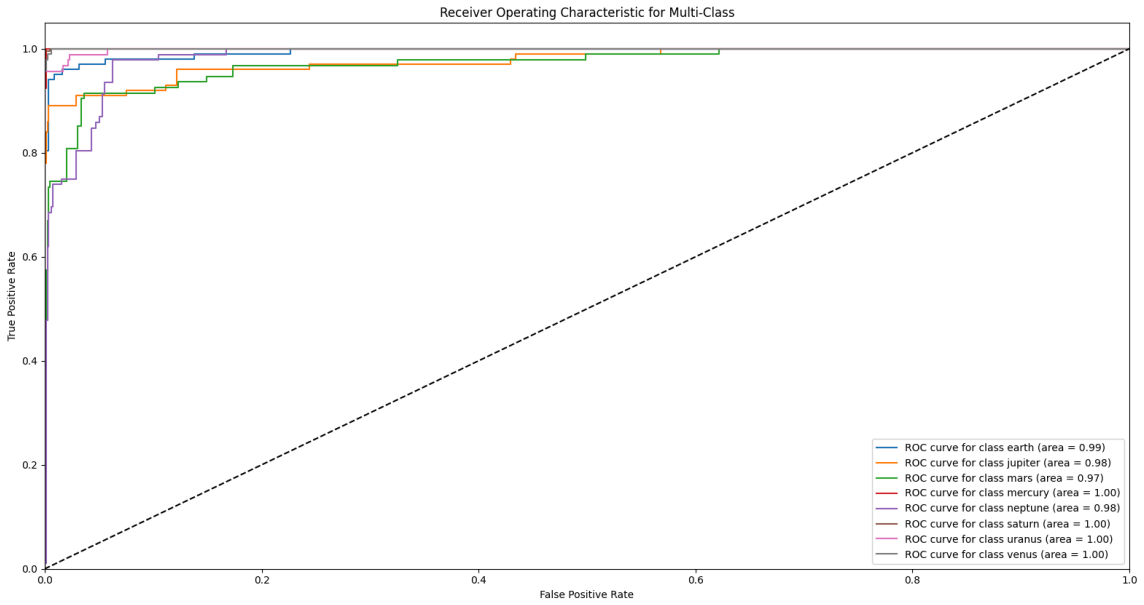


Image 52.

and the ROC curve is:



.

LinearSVC is yet another type of SVM which is developed for linear models only for classification. It is available in the Scikit learn library and it is a wrapper of the LIBLINEAR library which is an efficient one for large data sets [65]. LinearSVC is considered to be fast and versatile when dealing with a large number of samples, while the number of features is relatively small.

Linear Support Vector Machine (LinearSVC) and Logistic Regression models were identified to give the best results for most of the datasets. It provided high accuracy and relatively short computation times, so it can be applied to automatic cell identification.

We can also implement the LinearSVC model to the previous model.

In the libraries,the changes are: from sklearn import svm to from sklearn.svm import LinearSVC.

Also,the SVM model used is LinearSVC instead of svm.SVC.

Added: dual=False parameter in LinearSVC, which is more efficient when the number of samples is greater than the number of features.

For the Feature Extraction, the base model for feature extraction is now MobileNetV2 instead of VGG16 The input image size for MobileNetV2 is set to (192, 192, 3).

MobileNetV2 is a state of the art neural network architecture, which is put forward for the efficient implementation on the mobile and embedded platforms. The key innovations of MobileNetV2 are:

Inverted Residuals: Unlike the conventional residual blocks where the high dimensional layers are connected, MobileNetV2 uses inverted residuals connecting low dimensional layers. This design enhances the memory usage and thus enhances the computational speed.

Linear Bottlenecks: The architecture uses linear bottleneck layers to ensure that no non-linearities appear in the narrow layers to enthrone representational capability. It also reduce the rate of information loss and increase efficiency because the transformation preserve key features of the data.

Efficiency: MobileNetV2 is able to provide superior performance on a number of benchmarks with fewer operations and less memory usage than comparable architectures. This makes it even more suitable for use in mobile and environment with limited resources.

Applications: The architecture itself has been tested on various tasks including image classification and object detection (with the help of SSDLite) and semantic segmentation (with Mobile DeepLabv3), proving its universality across the various domains.

The Performance of the MobileNetV2:

MobileNetV2 results to better performance than other architectures like ShuffleNet and NASNet-A with much fewer parameters and lower complexity.

For example, MobileNetV2 is a model that provides high accuracy of the ImageNet classification task and, at the same time, has a relatively small number of parameters, which is perfect for real-time use.

In the code part, the include_top=False parameter is used, meaning the fully connected layers at the top of the network are omitted.

Also, batch_size=32 parameter for predictions is added, using a batch size of 32.

The results of the LinearSVC and the MobileNetV2 are:

Iteration	**Accuracy**	**Precision**	**Recall**	**Error Rate**	**F1 Score**	**MSE**

mean value	0.9664	0.9655	0.9591	0.0335	0.9609	0.3809

Table 26.

The confusion Matrix is:

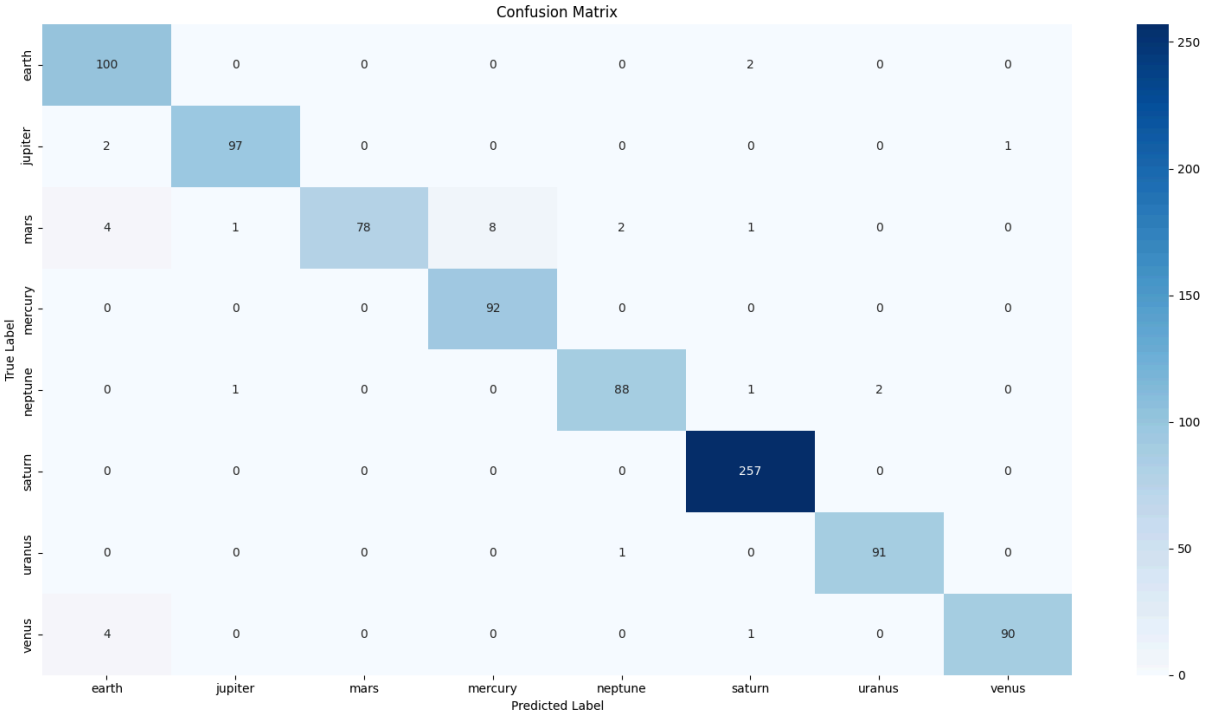


Image 53.

And the ROC curve is:

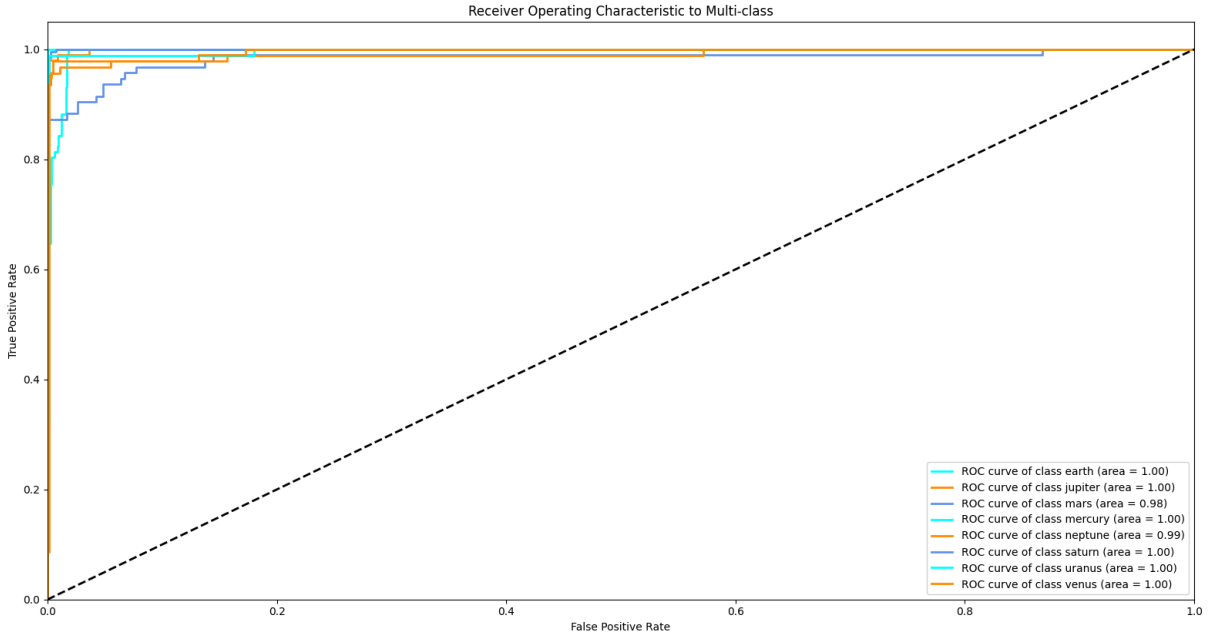


Image 54.

Chapter 8: Comparison and Evaluation of Machine Learning Algorithms for Noisy Image Classification in Astronomy

Comparison of the performance of different machine learning algorithms for noisy image classification in astronomy based on the evaluation metrics discussed in section IV
The progress of astronomical study is greatly dependent on the capacity to analyse images that are frequently blurred by different kinds of interference. In this chapter, the effectiveness of various machine learning algorithm for classification of noisy images, which is based on the previous chapters, is compared. In this way, we want to find out which algorithms are not only noise-tolerant but also have high accuracy and speed in classification tasks.

8.1 All the algorithms comparison

8.1.1 CNN

Algorithm	Accuracy	Precision	Recall	Error Rate	F1 Score	MSE
Simple CNN	0.8080	0.8320	0.7860	0.1770	0.7870	1.7040
AlexNet	0.934	0.939	0.930	0.058	0.929	0.0115

CNN						
DenseNet CNN	0.9647	0.9667	0.9574	0.0353	0.9598	0.6368

Table 27.

From the array above, we can conclude that DenseNet is the most effective model among the three in terms of accuracy, precision, recall, error rate, and F1 score, making it highly suitable for critical applications where high reliability is needed. AlexNet, while not matching DenseNet in most metrics, demonstrates a significantly lower MSE, suggesting that where error variance is crucial, it might be the preferred model. As expected, Simple CNN was the least effective in this set. Although it still offers reasonable performance and might be useful in scenarios where computational efficiency is more critical than maximum accuracy, it gets outperformed in every scenario by the other CNN implementations.

8.1.2 Autoencoder

Algorithm	**Accuracy**	**Precision**	**Recall**	**Error Rate**	**F1 Score**	**MSE**
Autoencoder+ SVM (simple)	0.7711	0.7348	0.5580	0.2289	0.6322	2.8607
Autoencoder +KNN(advanced)	0.98648	0.98641	0.9860	0.01335	0.98473	0.2991

Table 28.

The Advanced Autoencoder has a better accuracy which proves that it is much more effective in correctly identifying images. It also also shows a significant increase in both precision and recall by. This means that it is more appropriate in the detection of the positive cases and decreasing the number of false positives. From the above results we can also observe that the mean squared error is significantly lower for the Advanced Autoencoder which shows that the Advanced Autoencoder has a lower variability in its prediction errors.

The results show that the Advanced Autoencoder has a significantly better performance than the Simple Autoencoder in all the aspects and thus can be considered better capable of classifying noisy images with a greater degree of accuracy. This implies that the changes made in the architecture of the autoencoder are indeed beneficial in the classification process and especially where the data is complex such as in the presence of noise.

8.1.3 KNN

Algorithm	**Accuracy**	**Precision**	**Recall**	**Error Rate**	**F1 Score**	**MSE**

Simple KNN	0.9123	0.9142	0.9123	0.0876	0.9128	1.066
weighted KNN	0.9134	0.9095	0.9120	0.0865	0.9094	0.8387
Dudani KNN	0.9242	0.9182	0.9221	0.0757	0.9193	0.7727

Table 29.

Of the three K-NN algorithms, Dudani KNN has the highest accuracy of 92.42%, which means that it is the most efficient in classifying the data since there is not much difference it has with Simple and Weighted KNN. Dudani KNN also has the highest recall of 92.21% which shows that the model is slightly better at recognizing true positives than the other models.
On the other hand, Simple KNN scored slightly better in terms of precision with 91.42% compared to Dudani KNN at 91.82 percent hence it is less likely to classify instances as positive than it actually is.
The precision of Weighted KNN is slightly lower at 90. 95% and recall of 91. 20% than the other two, which may suggest that it has a slight difficulty in distinguishing between the true positive and false positive. It is also important to note that Dudani KNN has the least MSE value of 0.7727 which implies that the predictions are more accurate and less spread out than Simple and Weighted KNN's predictions.
Largely, Dudani KNN has a better performance in all the metrics which depicts that improvements made on this variation will lead to better predictions with higher accuracy, recall and lower error rates. Simple KNN, although having a higher precision, is less efficient on other aspects. Weighted KNN does not show much better results and in some of the parameters, it is slightly worse; this means that the weighting technique is not well suited to this dataset due to some configuration issues.

8.1.4 Decision Trees

Algorithm	**Accuracy**	**Precision**	**Recall**	**Error Rate**	**F1 Score**	**MSE**
Simple Decision Tree	0.8689	0.8759	0.8689	0.1310	0.8702	1.6416
Bagging Decision Tree	0.9134	0.9149	0.9134	0,0866	0.9138	1.2089

Table 30.

The accuracy of the Bagging Decision Tree is 91.34% while that of the Simple Decision Tree is 86.89%. This could be attributed to the bagging method that forms the basis for the decision trees and which aids in the reduction of variance through combining the outputs of several

decision trees. Both precision and recall are higher in the Bagging Decision Tree (precision: It outperformed the Simple Decision Tree in both the precision (91.49%) and recall (91.34%) than in the Simple Decision Tree with a precision of 87.59% and recall of 86. 89%. This means that the bagging method does not only generate more accurate results but also, it is able to classify a larger number of actual positives correctly. .
From the comparison made it was observed that the Bagging Decision Tree outperforms the Simple Decision Tree in all the metrics.

8.1.5 Random Forest

Algorithm	Accuracy	Precision	Recall	Error Rate	F1 Score	MSE
Simple Random Forest	0.93291	0.93319	0.93291	0.06709	0.93284	0.87533
Advanced Random Forest	0.93622	0.93640	0.93621	0.06378	0.93619	0.83368

Table 31.
The Advanced Random Forest has a slightly better accuracy as compared to the Simple Random Forest with 93.622% as against 93.291%. This indicates that the Advanced version might contain a number of minor enhancements or optimizations which can enhance its predictive characteristics slightly. The Precision and the Recall of the Advanced Random Forest are slightly better than that of the Simple Random Forest. This can be considered as a slight but positive enhancement in the model's overall performance in recognising positive cases (recall) and its correctness when doing so (precision). As for the MSE, which is the average of the squared differences between the predicted and actual values, it is also lower for the Advanced Random Forest (0. 83368) than for the Simple Random Forest (0. 87533). A smaller MSE shows that the predictions made by the Advanced model are nearer to the actual data points hence, giving a better and more reliable precisions.
Both models have demonstrated quite a good performance, however, the Advanced Random Forest model reveals a stable improvement deal in each of the major measures of performance than the Simple Random Forest model. These improvements indicate that the modifications/ enhancements in the Advanced Random Forest offers a slightly better, dependable and efficient model for classification problems. This can be especially advantageous in the contexts where slight changes in the prediction and errors could lead to massive consequences.

8.1.6 SVM

Algorithm	Accuracy	Precision	Recall	Error Rate	F1 Score	MSE

Simple SVM	0.9502	0.9448	0.9395	0.0497	0.9398	0.8116
LinearSVC	0.9664	0.9655	0.9591	0.0335	0.9609	0.3809

Table 32.

LinearSVC returns a slightly better accuracy of 96. 64% as compared to Simple SVM which has an accuracy of 95. 02%. This may mean that LinearSVC is employing a more efficient optimization method or is well suited for the given data set thus producing a more accurate final prediction. The same is true for the LinearSVC which achieves 96. 55% precision as opposed to 94. 48% and 95. 91% recall as opposed to 93. 95%. This improvement in precision suggests that LinearSVC is better at detecting positive samples and therefore has lower false positives. It also points to the fact that it is likely to identify more of the positives, hence reducing the chances of false negatives. The mean squared error is lower for LinearSVC, which is 0.3809 while for SVC, it is 0.8116. This indicates that the predicted class labels by LinearSVC are more accurate than the ones by SVC with a lower MSE which also implies that the errors are lesser and not spread out widely, making LinearSVC a better choice.
It is also obvious that LinearSVC is superior to Simple SVM in all the performance indicators considered. These improvements make LinearSVC especially suitable for the use cases where accuracy and reliability are the key factors, including sensitive classifications with high costs of misclassification.

8.2 Identification of the best-performing algorithms

In this section we will identify the best performing algorithm for the solar system data set. We will include the best metric values of the best performing algorithms, and after discusting the strengths and the weaknesses of each algorithm we will conclude the best performing machine learning algorithm. In the following array, we will present the best values of every algorithm:

Algorithm	**Accuracy**	**Precision**	**Recall**	**Error Rate**	**F1 Score**	**MSE**
CNN	0.9647	0.9667	0.9574	0.0353	0.9598	0.6368
Autoencoder	0.98648	0.98641	0.9860	0.01335	0.98473	0.2991
KNN	0.9242	0.9182	0.9221	0.0757	0.9193	0.7727
Decision Tree	0.9134	0.9149	0.9134	0,0866	0.9138	1.2089
Random Forest	0.93622	0.93640	0.93621	0.06378	0.93619	0.83368
SVM	0.9664	0.9655	0.9591	0.0335	0.9609	0.3809

Table 33.

From the values in the array we can conclude that:

1. Accuracy

These include autoencoder with an accuracy of 98.68%, secondly; SVM with an accuracy rate of 96.64% and thirdly; CNN with an accuracy of 96.47%. These values suggest that these algorithms are very reliable in every classification problem.

The Decision Tree (91. 34%) and KNN (92. 42%) are less accurate which means that they may have difficulties working with more elaborate datasets or those datasets that contain features that are not easily distinguishable.

2. Precision

The autonceder has the highest precision at 98.64% which denotes the ability of minimizing false positives; this is useful in applications that have dire consequences of false positives.

CNN achieved an accuracy of 96.67% while SVM achieved 96.55% precision and both are also accurate in positive class predictions.

Lower precision from KNN (91. 82%) and Decision Tree (91. 49%) may be disadvantageous since these algorithms are not as precise as the others.

3. Recall

Recall is seen to be highest in the autoncoder (98.60%), which points towards its ability of identifying the maximum number of instances. Thus, the performances of SVM and CNN are similar, which reflect their potential for positive cases' detection.

Decision Tree and KNN have the lowest recall thus making them less efficient in conditions where failure to detect a positive instance is costly.

4. Error Rate

The Autoencoder has the lowest error rate, which is 0.13%, this shows that it made the least number of classification errors, followed by SVM with an error rate of 3. 35% and CNN 3. 53%.

Among all the algorithms used, Decision Tree entails the highest error rate of 8.66% while KNN follows closely with a 7.57% error rate indicating that they are relatively inaccurate in classification.

5. F1 Score

The best performance is achieved by the autoencoder with F1 Score of 98%.

The F1 Scores are also high for all the models with CNN having 95. 98% signifying that it is good at identifying the positive samples without creating many false positives.

It is also observed that, KNN and Decision Tree have lower F1 score value than other classifiers, which is a sign of a less preferable balance between precision and recall.

6. Mean Squared Error (MSE)

Autoencoder presents an extremely small MSE of 0.29 thus meaning that, the autoencoder model is very accurate with the predictions and the variance of its prediction error is very small.

The MSE of SVM is 0.3809 and for CNN is 0.6368 which is also lower in value indicating that the predictions made by the two models are fairly accurate and credible.

The MSE for Random Forest is 0.83368 and Decision Tree is 1.2089 which shows that there is high variance in the prediction errors which can affect their use in applications which require high prediction stability.

Overall best performing algorithm:

Autoencoder is ideal for applications that demand precision, minimal error and predictable results, thus, its low MSE. This model is best for those tasks where the accuracy of the prediction is critical and even a slight mistake can lead to a big problem.

CNN and SVM can be viewed as very viable solutions which provide high precision and recall necessary for various tasks where it is important to achieve both high sensitivity, i.e. produce positive results that indeed correspond to positives (true positive) and low sensitivity, i.e. minimize the number of false positives. The high F1 scores are ideal in such scenarios to ensure that this balance is well sustained.

The Random Forest can be said to be in between the three based on the performance results as it yields good results in all the metrics but not as high as in the case of CNN or SVM. It may be useful in applications that require a strong, but non-specialized model than the CNN or SVM models.

Decision Tree and KNN can be used in cases where time and resources are a constraint or where model interpretation is of gradual importance over accuracy.

Chapter 9: Challenges and Future Directions

9.1 Future Directions

9.1.1 CNN

Convolutional Neural Networks initially developed from the early models that utilize handcrafted feature to the current models that can automatically learn and detect multiple levels of features from the input data. Such progress is seen in architectures such as AlexNet, VGG, and ResNet; architectures that have enormously boosted image classification in as far as accuracy and computational costs are concerned [66]. The use of deeper networks and ReLU activation function in AlexNet proposed brought a significant improvement in performance, as shown in the chapters above. VGG extended the layers by using smaller sized filters in convolutional layers enhancing the feature learning. ResNet introduced residual connections that allow training of extremely deep models while not suffering from the degradation problem of gradients. The state-of-the-art research on CNNs is mainly concentrated on the enhancement of the model, reducing the model size and load time while attempting to maintain accurateness. There is also a focus on the overall performance and transferability of the methods, including the transfer of CNN models from synthetic to real data, and avoiding overfitting. New approaches are being applied to CNN architectures including attention mechanisms and neural architecture search to enhance the selection of the CNN architecture. Designed for specific purposes, such as medical imaging, self-driving cars, and agriculture, new application-specific architectures are being developed. In farming, CNNs are applied in crop image identification and assessment at different growth phases including seed identification, plant growth observation, fruit or flower identification, and crop quality assessment after harvest. Nevertheless, there are still issues, including the issues of variability in environmental conditions, and the requirement of large and well-labeled datasets. Another current research challenge is to enhance the performance of CNNs in order to achieve better efficiency when applied under various situations. New methods are emerging in the form of GANs and reinforcement learning to improve CNNs while sustainable approaches seek to apply CNNs for yield, disease detection, and resource optimization [67]. This review focuses on the reviewing the advancement of CNN architectures and underlines the importance of future research to meet the new challenges of classification and to extend the application of CNNs in various areas.

9.1.2 Autoencoders

Ongoing research in image classification has increasingly focused on deep learning models, including deep autoencoders and convolutional autoencoders, to improve classification performance, particularly when dealing with complex datasets. Deep autoencoders are neural networks that learn low dimensional representations of data sets. It generally consists of two steps, where the first step is pre-training, which is done without supervision and aims at

learning features, and the second step is the supervised training where the previously learned features are used for classification aiming at improving their performance. In this case, another advantage of combining autoencoders with classifiers such as support vector machines is that it offers better performance than the conventional ways of doing it [68]. This combination also makes it possible to test the system under different network environments. However, there are certain restrictions including the requirement of experimenting on more intricate datasets apart from the benchmarks, addressing issues concerning overfitting and boosting the generalization performance of the model.

In particular, CAEs are well designed to process and analyze complex and noisy image datasets. It eliminates noise and high dimensionality by feature extraction, and then fed into convolutional neural networks (CNNs) for classification. Such enhancement has been proven to enhance classification performance especially when working with noisy datasets by cleaning the data before it is fed to the CNN. The proposed CAE-CNN architectures have provided more accurate results and better performance measures including geometric mean and AUC in noise reduction than the baseline methods [69].

Possible future work for deep autoencoders is to apply them on larger and more different data sets to determine the effectiveness of the models and to find methods to decrease overfitting. Further comparisons with more classifiers will also be useful in identifying the effectiveness of the proposed approach in different situations. For convolutional autoencoders, the creation of next-level architectures could improve the filtering of noise and the identification of features and when these models are tested with multiple complicated classification problems, it will further prove its performance. Moreover, applying CAEs on other recent developments such as GANs or reinforcement learning may produce more effective solutions for image classification.

Deep autoencoders and convolutional autoencoders have enhanced the world of image classification. Deep autoencoders with classifiers like SVMs work well with standard datasets but CAE-CNN is an excellent solution for noisy and more dimensional data. These advancements have a focus and relevance to some of the biggest issues facing image processing and could enable further developments in many different fields.

9.1.3 Decision Trees

Recent research highlights the improvements in predictive modeling through XGBoost, demonstrating its effectiveness in wave run-up prediction and general machine learning tasks. XGBoost has been found to greatly improve the process of predicting outcomes in many fields. In wave run-up modeling, XGBoost has been employed for the prediction of wave run-up height which is useful in coastal engineering [70]. To predict the LAIs more than 400 laboratory observations have been utilized to train the XGBoost model with the hyperparameters set by grid search. The algorithm is built based on the decision trees enhanced by gradient descent and employs regularization to enhance decision making. Learning rate, maximum depth of the decision tree and subsample ratio were tuned so as to get the best results. In comparison to other models for regression such as linear regression, support vector regression, and random forests, XGBoost was more accurate in all the cases. It yielded an R^2 of 0.98675, with a prediction error of 6% which was a more accurate result than other empirical methods.

The efficiency of XGBoost in handling the large data set, and the accuracy in modeling beach slopes and wave amplitudes for different locations make it suitable for application in coastal engineering problems. Future work in this area should aim at applying the model on more real

datasets which are different from the ones used in this paper. Furthermore, the possibilities of enhancing applicability of the model for different types of coasts and waves might be investigated. Other machine learning models can also be combined with XGBoost to improve its predictive capability.

XGBoost is also very popular in many machine learning applications due to the reasons of its scalability and efficiency. It utilizes gradient tree boosting which constructs decision trees sequentially with the aim of minimizing a regularized objective function; it also incorporates measures against over fitting. XGBoost also supports split finding algorithms like exact and approximate split finding and is capable of handling large and sparse data sets through sparsity-aware learning. Due to the fact that the model is scalable and can process data in parallel it ranks among some of the most efficient systems for tasks like classification and ranking [71].

Some possible future work on XGBoost should extend the research in improving scalability for Future work on XGBoost should explore further enhancements in scalability, particularly for handling even larger datasets and working within distributed computing systems. There is also an opportunity for further enhancement of regularization methods used to enhance model generalization. Further exploring the utilization of XGBoost in different areas of interest including real time prediction and big data could be seen as beneficial in future works.

9.1.4 Random Forest

Tree-based methods, particularly gradient boosting algorithms like XGBoost, have become popular for predicting future outcomes. These methods are particularly useful when working with big data and large and intricate models because they are capable of delivering accurate predictions along with regularization and optimization functions.

Regarding feature selection, distance correlation-based methods are more generalized than traditional methods that are based on linear relationship. Distance correlation (dCor) is a measure of dependency that can capture linear and nonlinear structure, thus, effectively address the challenging problem of feature selection when operating in a high-dimensional, nonlinear space as well as enhance the performance of the model by identifying the most significant predictors [72]. Another crucial strategy in improving feature selection is the hyperparameter optimization in which techniques such as the grid search come in handy to fine-tune the parameters of the model in order to improve its performance. This process is rather helpful in the case of dealing with massive datasets, as the accuracy and time of the process are crucial.

For future studies, future work can be done to refine the model architecture, for example, by designing better autoencoder models and better gradient boosting algorithms for a more accurate model. Additional domains of application including multi-dimensional data sets as may be encountered in medical imaging or environmental monitoring will also provide a forum through which the effectiveness of these approaches can be tested further. The use of the latest technology in the field of artificial intelligence like GANs and reinforcement learning is seen as having the potential to enhance and even improve current predictive models. Moreover, as the datasets become larger, the issues of computational time and space complexity will be paramount, and therefore, the methods have to be efficient. Technique that involves the use of parallel and distributed computing will be very useful in ensuring that efficiency and scalability is maintained particularly when dealing with large and sparse data.

9.1.5 K-Nearest-Neighbor

Advancements in feature weighting as well as classification have recently presented new techniques to improve the performance of the models and at the same time increase the interpretability of the models. This section focuses on major ideas and conclusions related to the use of class-based feature weighting in K-Nearest Neighbors (KNN) classification. Conventional methods of feature weighting employ a single weight vector across all classes where as class dependent feature weighting produces different weight vectors per class. This method, which is based on the RELIEF algorithm, corrects the weight of the features in order to optimize the margin for KNN classifiers, because it is known that features have different significance than for the class [76]. This method provides a more improved feature selection because it assigns weights to classes.

The analysis of the results shows that feature relevance is not always consistent between classes. For instance, in medical data some attributes may be very important in diagnosing the relapse of the disease, while others may not be important at all. This shows that the standard approach of using equal weight for all features fails to address this within-class variation. Class-based feature selection makes it possible to understand which features are useful for which class and this can be very helpful in areas such as medicine where information may not easily be understandable. This method increases the accuracy of classification while at the same time allowing for easier understanding of how different features are influencing the outcome in different circumstances.

This approach was empirically validated using datasets of the UCI machine learning repository. The findings show that class-dependent feature weighting enhances the efficiency of classification and produces a more advantageous comprehension of the procedural complexity of classification. This is particularly true for the classification problems where the connection between class-dependent weights and those obtained using RELIEF is often close.

As contributions, class-dependent feature weighting can be considered as a major improvement in feature selection. This is better than previous approaches because it presents different weights for each class, making for a better comparison of the importance of features in various scenarios. This is especially helpful when the performance of features varies with each class of data. Nevertheless, there are some issues concerning the application of this method to the existing machine learning frameworks since only a small number of existing systems are capable of dealing with many sets of feature weights.

The work in the future can consider how to achieve both high interpretability and high accuracy by assigning different weights to the features of the classes. This could include working on techniques for dealing with classification issues while at the same time being sensitive to features that are important across classes. Also, Plugging this method into conventional machine learning pipelines, possibly with class-dependent feature weighting may lead to the development of scalable architectures. Further research should also examine this approach in other contexts, for example, finance and environmental science to establish their effectiveness and reveal others. It could therefore be useful to explore how the method could be applied more broadly in order to improve its efficacy, validity and versatility across various data and challenges.

Therefore, the proposed class-dependent feature weighting performs better in feature selection, thus providing more meaningful and accurate classification based on the fact that features do

not have the same relevance in all classes. Future work will be necessary to resolve integration problems and to open new areas of application of this promising technique.

9.2 Future Directions in Image Classificaiton in general

9.2.1 Quantum Machine Learning for Image Classification

Quantum Machine Learning (QML) is a branch of study that combines concepts of quantum computing with the machine learning techniques. This integration is proposed to(rad)exploit the special potential of quantum computers for enhancing computational tasks in a scenario where astronomical imaging data sets are typically large and noisy.

Quantum Computing Basics

Quantum computers utilize quantum bits or qubits which are capable of existing in several states at once which is the super position while the classical bits only exist in two states that is either 0 or 1. This helps quantum computers to solve many problems at the same time which is called quantum parallelism. Some of the quantum algorithms are based on quantum principles such as superposition, entanglement and quantum interference with an aim of solving problems that cannot be solved by classical computers.

Quantum Computing and NISQ Devices [73]

Techniques of quantum computing are used to develop quantum algorithms for classification, regression, and clustering. The present-day quantum computers belong to the Noisy Intermediate-Scale Quantum (NISQ) era which has a small number of qubits that have not been fully error-corrected. Despite these drawbacks, quantum algorithms are being developed in such a way that they will be able to operate under these conditions.

Astronomy and Image Classification Use of ANN

QML offers several advantages for image classification, especially in noisy and high-dimensional data sets like those found in astronomy:QML offers several advantages for image classification, especially in noisy and high-dimensional data sets like those found in astronomy:

Handling High-Dimensional Data: This quantum computers are very effective with data sets that are high dimensional and this is useful with astronomical images.

Quantum Feature Spaces: Quantum states of data can represent intricate patterns and relations and, therefore, improve machine learning.

Quantum-Enhanced Models: For instance, quantum neural networks, or quantum support vector machines that are a combination of both classical and quantum techniques might perform better in terms of parameter and computational speed owing to quantum parallelism.

Hybrid Quantum Models

Recent research has explored hybrid quantum-classical models that integrate quantum computing with traditional deep learning architectures:Recent research has explored hybrid

quantum-classical models that integrate quantum computing with traditional deep learning architectures:

Hybrid Quantum Neural Network (HQNN): This model incorporates both quantum circuits with classical neural network layers and it performs well with fewer parameters.

Quantum-enhanced Convolutional Layers: Another model adapts the use of quantum circuits in convolutional processes with the same level of performance but fewer trainable parameters.

These models have been tested in different datasets and have given good results thus proving that quantum computing has the ability of enhancing classification tasks.

Quantum Classifiers and Its Applicability in the Real World

The application of fully quantum classifiers to real image datasets, such as those used in medical imaging, has demonstrated significant potential:The application of fully quantum classifiers to real image datasets, such as those used in medical imaging, has demonstrated significant potential:

Efficient Data Encoding: Quantum classifiers perform better with fewer qubits than the classical bits for similar tasks because of quantum superposition and entanglement.

Handling Imbalanced Data: It has been seen that quantum classifiers are effective in handling both balanced and imbalanced datasets which is advantageous for applications where some classes are sparse.

Strengths and Limitations

Strengths:

Innovative Approach: Quantum computing provides a fresh point of view to the image classification process and presents new solution paradigms to computational problems.

Efficiency and Performance: Hybrid models and quantum classifiers have shown to be very efficient and accurate at times even surpassing the classical models.

Real-World Applicability: The performance of the quantum classifiers is tested on real datasets which prove the applicability of quantum classifiers in areas that demand accurate results.

Limitations:

Hardware Constraints: However, there are some challenges with quantum technology which hampers the utilization of quantum models such as scalability and error rates.

Complexity of Integration: The integration of the quantum and classical components also poses a challenge in the model training and optimization.

Limited to Binary Attributes: Some of the quantum models are limited to binary data which may need44 some data pre-processing in order to make them applicable in other situations.

Future Directions

Advancements in Quantum Hardware: To build more realistic quantum machine learning models, more advancement needs to be made in the current quantum hardware.

Integration with Classical Systems: Thus, the methods for combining the quantum and classical elements can improve the use of the quantum-enhanced models in practical conditions.

Exploration of Broader Applications: Future work may look into introducing quantum classifiers to other kinds of data and problems, to discover more potential advantages and opportunities and thus, prove its efficiency.

Consequently, the concept of Quantum Machine Learning is an emerging field in the computational intelligence area that has the potential of providing significant advancements in the images classification process and other related fields due to the quantum computing. More research and development is needed for the optimization of QML for it to provide its greatest benefits.

9.3 Conclusion

Summary of key findings from the literature review

The present literature review examined the current advancements, methodologies and the challenges that are related to the implementation of image classification. This review showed that the advancements in the development of the CNN architecture have been made in a chronological order from the basic architecture of AlexNet and VGG to the advanced structures of ResNet and more. Comparisons were made among different machine learning models and Random Forests and Support Vector Machines were found to be efficient in all datasets used in the study. Nevertheless, the CNNs as well as Autoencoders reported higher accuracy and reliability especially when high levels of accuracy and low levels of errors were imperative. It also emphasized the need to choose the right algorithm depending on the requirements of the task at hand – either for overall performance, speed or for working with imbalanced data. Challenges in Noisy Image Classification: Among the issues which have been addressed in the review and remained critical was the question of how to process the noisy and intricate data typical for astronomy. There is a challenge in the application of such datasets with the traditional algorithms but with the development of machine learning especially CNNs and RFs the classification results have been enhanced. It was also highlighted that hyperparameter tuning and data cleaning are critical to the improvement of the model performance.

Future Directions and Emerging Technologies: Some of the directions for the future work were proposed by the literature, including the incorporation of the attention mechanisms and neural architecture search. These advancements are anticipated to enhance the flexibility and the transferability of CNNs and enhance their performance in a broad range of applications including medical image analysis and self driving cars. Moreover, there is a increasing demand for looking into unsupervised and semi-supervised learning techniques in order to decrease reliance on large annotated datasets, which might greatly contribute to the development of image classification.

BIBLIOGRAPHY

[1] “A Clustering Method Based on K-Means Algorithm - ScienceDirect.” Accessed: Jun. 10, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1875389212006220

[2] J. Cervantes, F. Garcia-Lamont, L. Rodríguez-Mazahua, and A. Lopez, “A comprehensive survey on support vector machine classification: Applications, challenges and trends,” *Neurocomputing*, vol. 408, pp. 189–215, Sep. 2020, doi: 10.1016/j.neucom.2019.10.118.

[3] F. Zhuang *et al.*, “A Comprehensive Survey on Transfer Learning,” *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, Jan. 2021, doi: 10.1109/JPROC.2020.3004555.

[4] “A detailed overview of noise in Astrophotography | DSLR Astrophotography.” Accessed: Jun. 10, 2024. [Online]. Available: https://dslr-astrophotography.com/detailed-overview-noise-astrophotography/

[5] A. Pannekoek, *A History of Astronomy*, Later Edition Used. New York: Dover Publications, 2011.

[6] A. Pannekoek and A. Pannekoek, *A history of astronomy*, Unabridged and Unaltered republ. of the work 1. publ. [in English] ... London, 1961. in Dover books on astronomy. New York: Dover Publ, 1989.

[7] M. Marengo and M. C. Sanchez, “A k-NN METHOD TO CLASSIFY RARE ASTRONOMICAL SOURCES: PHOTOMETRIC SEARCH OF BROWN DWARFS WITH SPITZER/IRAC,” *AJ*, vol. 138, no. 1, p. 63, May 2009, doi: 10.1088/0004-6256/138/1/63.

[8] J. McCarthy, M. L. Minsky, N. Rochester, I. B. M. Corporation, and C. E. Shannon, “A PROPOSAL FOR THE DARTMOUTH SUMMER RESEARCH PROJECT ON ARTIFICIAL INTELLIGENCE”.

[9] M. H. Hassanshahi, M. Jastrzebski, S. Malik, and O. Lahav, “A quantum-enhanced support vector machine for galaxy classification,” Nov. 06, 2023, *arXiv*: arXiv:2306.00881. Accessed: Jun. 10, 2024. [Online]. Available: http://arxiv.org/abs/2306.00881

[10] A. Mohammed Abd-Alsalam Selami and A. Freidoon Fadhil, “A Study of the Effects of Gaussian Noise on Image Features,” *KUJSS*, vol. 11, no. 3, pp. 152–169, Sep. 2016, doi: 10.32894/kujss.2016.124648.

[11] “Algorithms | Free Full-Text | Random forest Algorithm for the Classification of Spectral Data of Astronomical Objects.” Accessed: Jun. 10, 2024. [Online]. Available: https://www.mdpi.com/1999-4893/16/6/293

[12] “An Astrophotographer’s Gentle Introduction to Noise,” Sky & Telescope. Accessed: Jun. 10, 2024. [Online]. Available: https://skyandtelescope.org/astronomy-blogs/imaging-foundations-richard-wright/astrophotography-gentle-introduction-noise/

[13] U. Michelucci, “An Introduction to Autoencoders,” Jan. 11, 2022, *arXiv*: arXiv:2201.03898. Accessed: Jun. 10, 2024. [Online]. Available: http://arxiv.org/abs/2201.03898

[14] K. O’Shea and R. Nash, “An Introduction to Convolutional Neural Networks,” Dec. 02, 2015, *arXiv*: arXiv:1511.08458. Accessed: Jun. 10, 2024. [Online]. Available: http://arxiv.org/abs/1511.08458

[15] “Analysis: How AI is helping astronomers study the universe,” PBS NewsHour. Accessed: Jun. 10, 2024. [Online]. Available: https://www.pbs.org/newshour/science/analysis-how-ai-is-helping-astronomers-study-the-universe

[16] M. Freed and J. Lee, “Application of Support Vector Machines to the Classification of Galaxy Morphologies,” Jun. 2013, pp. 322–325. doi: 10.1109/ICCIS.2013.92.

[17] J.-L. Starck and F. Murtagh, *Astronomical Image and Data Analysis*. Springer Science & Business Media, 2007.

[18] J.-L. Starck and F. Murtagh, “Astronomical image and signal processing: looking at noise, information and scale,” *IEEE Signal Processing Magazine*, vol. 18, no. 2, pp. 30–40, Mar. 2001, doi: 10.1109/79.916319.

[19] “AstroVaDEr: Astronomical Variational Deep Embedder for Unsupervised Morphological Classification of Galaxies and Synthetic Image Generation,” ar5iv. Accessed: Jun. 10, 2024. [Online]. Available: https://ar5iv.labs.arxiv.org/html/2009.08470

[20] J. L. Turner and H. A. Wootten, “Atacama Large Millimeter/Submillimeter Array (ALMA),” *Proc. IAU*, vol. 2, no. 14, pp. 520–521, Aug. 2006, doi: 10.1017/S1743921307011659.

[21] J. Zhai, S. Zhang, J. Chen, and Q. He, “Autoencoder and Its Various Variants,” in *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Oct. 2018, pp. 415–419. doi: 10.1109/SMC.2018.00080.

[22] P. Baldi, “Autoencoders, Unsupervised Learning, and Deep Architectures,” in *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, JMLR Workshop and Conference Proceedings, Jun. 2012, pp. 37–49. Accessed: Jun. 10, 2024. [Online]. Available: https://proceedings.mlr.press/v27/baldi12a.html

[23] S. Schmidgall *et al.*, “Brain-inspired learning in artificial neural networks: a review,” May 18, 2023, *arXiv*: arXiv:2305.11252. Accessed: Jun. 10, 2024. [Online]. Available: http://arxiv.org/abs/2305.11252

[24] C. Gheller and F. Vazza, “Convolutional Deep Denoising Autoencoders for Radio Astronomical Images,” *Monthly Notices of the Royal Astronomical Society*, vol. 509, no. 1, pp. 990–1009, Nov. 2021, doi: 10.1093/mnras/stab3044.

[25] R. W. Clay, A. G. K. Smith, and J. L. Reid, “Cosmic Ray Induced Noise in Gravitational Wave Detectors,” *Publ. – Astron. Soc. Aust*, vol. 14, no. 2, pp. 195–199, 1997, doi: 10.1071/AS97195.

[26] N. Scoville *et al.*, “COSMOS: Hubble Space Telescope Observations*,” *ApJS*, vol. 172, no. 1, p. 38, Sep. 2007, doi: 10.1086/516580.

[27] D. George and E. A. Huerta, “Deep Neural Networks to Enable Real-time Multimessenger Astrophysics,” *Phys. Rev. D*, vol. 97, no. 4, p. 044039, Feb. 2018, doi: 10.1103/PhysRevD.97.044039.

[28] R. K. Lindsay, B. G. Buchanan, E. A. Feigenbaum, and J. Lederberg, “DENDRAL: A case study of the first expert system for scientific hypothesis formation,” *Artificial Intelligence*, vol. 61, no. 2, pp. 209–261, Jun. 1993, doi: 10.1016/0004-3702(93)90068-M.

[29] A. L. Fradkov, “Early History of Machine Learning,” *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 1385–1390, Jan. 2020, doi: 10.1016/j.ifacol.2020.12.1888.

[30] T. Kohonen, “Essentials of the self-organizing map,” *Neural Networks*, vol. 37, pp. 52–65, Jan. 2013, doi: 10.1016/j.neunet.2012.09.018.

[31] “How infrared astronomy allows us to observe the Universe beyond visible light.” Accessed: Jun. 10, 2024. [Online]. Available: https://www.skyatnightmagazine.com/space-science/infrared-astronomy

[32] “Infrared Astronomy,” Webb. Accessed: Jun. 10, 2024. [Online]. Available: https://webbtelescope.org/home/webb-science/the-observatory/infrared-astronomy

[33] “Infrared astronomy reveals the secrets of the Universe beyond visible light.” Accessed: Jun. 10, 2024. [Online]. Available: https://www.skyatnightmagazine.com/space-science/infrared-astronomy

[34] E. Grossi and M. Buscema, “Introduction to artificial neural networks,” *European journal of gastroenterology & hepatology*, vol. 19, pp. 1046–54, Jan. 2008, doi: 10.1097/MEG.0b013e3282f198a0.

[35] A. M. Ikotun, A. E. Ezugwu, L. Abualigah, B. Abuhaija, and J. Heming, “K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data,” *Information Sciences*, vol. 622, pp. 178–210, Apr. 2023, doi: 10.1016/j.ins.2022.11.139.

[36] “*k*-Nearest Neighbour Classifiers 2^{*nd*} Edition (with Python examples),” ar5iv. Accessed: Jun. 10, 2024. [Online]. Available: https://ar5iv.labs.arxiv.org/html/2004.04523

[37] “Machine Learning in Astronomy”, by Lise du Buisson. Available: https://open.uct.ac.za/items/0ca06f6f-df9f-4950-9b2e-d671b753e007

[38] M. A. Chandra and S. S. Bedi, “Survey on SVM and their application in imageclassification,” *Int. j. inf. tecnol.*, vol. 13, no. 5, pp. 1–11, Oct. 2021, doi: 10.1007/s41870-017-0080-1

[39] D. Baron, “Machine Learning in Astronomy: a practical overview,” Apr. 15, 2019, *arXiv*: arXiv:1904.07248. Accessed: Jun. 10, 2024. [Online]. Available: http://arxiv.org/abs/1904.07248

[40] “Mathematics | Free Full-Text | Auto-Encoders in Deep Learning—A Review with New Perspectives.” Accessed: Jun. 10, 2024. [Online]. Available: https://www.mdpi.com/2227-7390/11/8/1777

[41] S. Bernard, L. Heutte, and S. Adam, “On the selection of decision trees in Random Forests,” in *IEEE International Joint Conference on Neural Networks (IJCNN)*, Atlanta, United States, Jun. 2008, pp. 302–307. doi: 10.1109/IJCNN.2009.5178693.

[42] information@eso.org, “Optical astronomy.” Accessed: Jun. 10, 2024. [Online]. Available: https://esahubble.org/wordbank/optical-astronomy/

[43] J. E. Nelson and P. R. Gillingham, “Overview of the performance of the W.M. Keck Observatory,” in *Advanced Technology Optical Telescopes V*, SPIE, Jun. 1994, pp. 82–93. doi: 10.1117/12.176154.

[44] “Principal component analysis: a review and recent developments | Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences.” Accessed: Jun. 10, 2024. [Online]. Available: https://royalsocietypublishing.org/doi/10.1098/rsta.2015.0202

[45] A. Maćkiewicz and W. Ratajczak, “Principal components analysis (PCA),” *Computers & Geosciences*, vol. 19, no. 3, pp. 303–342, Mar. 1993, doi: 10.1016/0098-3004(93)90090-R.

[46] J. Ali, R. Khan, N. Ahmad, and I. Maqsood, “Random Forests and Decision Trees,” *International Journal of Computer Science Issues(IJCSI)*, vol. 9, Sep. 2012.

[47] “Remote Sensing | Free Full-Text | Classification of PolSAR Images Using Multilayer Autoencoders and a Self-Paced Learning Approach.” Accessed: Jun. 10, 2024. [Online]. Available: https://www.mdpi.com/2072-4292/10/1/110

[48] C. Jie, Z. Jiyue, W. Junhui, W. Yusheng, S. Huiping, and L. Kaiyan, “Review on the Research of K-means Clustering Algorithm in Big Data,” in *2020 IEEE 3rd International Conference on Electronics and Communication Engineering (ICECE)*, Dec. 2020, pp. 107–111. doi: 10.1109/ICECE51594.2020.9353036.

[49] N. M. Ball, R. J. Brunner, A. D. Myers, and D. Tcheng, “Robust Machine Learning Applied to Astronomical Data Sets. I. Star‐Galaxy Classification of the Sloan Digital Sky Survey DR3 Using Decision Trees,” *ApJ*, vol. 650, no. 1, pp. 497–509, Oct. 2006, doi: 10.1086/507440.

[50] Y. Cheng, X. Wang, and Y. Xia, “Supervised t-Distributed Stochastic Neighbor Embedding for Data Visualization and Classification,” *INFORMS Journal on Computing*, vol. 33, no. 2, pp. 566–585, May 2021, doi: 10.1287/ijoc.2020.0961.

[51] T. Evgeniou and M. Pontil, “Support Vector Machines: Theory and Applications,” Sep. 2001, pp. 249–257. doi: 10.1007/3-540-44673-7_12.

[52] T. L. Wilson, “Techniques of Radio Astronomy,” Nov. 04, 2011, *arXiv*: arXiv:1111.1183. Accessed: Jun. 10, 2024. [Online]. Available: http://arxiv.org/abs/1111.1183

[53] R. Giacconi, “THE DAWN OF X-RAY ASTRONOMY”.

[54] L. Medeiros, D. Psaltis, T. R. Lauer, and F. Özel, “The Image of the M87 Black Hole Reconstructed with PRIMO,” *ApJL*, vol. 947, no. 1, p. L7, Apr. 2023, doi: 10.3847/2041-8213/acc32d.

[55] J. P. Gardner *et al.*, “The James Webb Space Telescope,” *Space Sci Rev*, vol. 123, no. 4, pp. 485–606, Apr. 2006, doi: 10.1007/s11214-006-8315-7.

[56] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” Apr. 10, 2015, *arXiv*: arXiv:1409.1556. Accessed: Jun. 10, 2024. [Online]. Available: http://arxiv.org/abs/1409.1556

[57] “What are Radio Telescopes? – National Radio Astronomy Observatory.” Accessed: Jun. 10, 2024. [Online]. Available: https://public.nrao.edu/telescopes/radio-telescopes/

[58] R. Giacconi, “X-Ray Astronomy,” in *X-Ray Astronomy*, R. Giacconi and G. Setti, Eds., Dordrecht: Springer Netherlands, 1980, pp. 1–13. doi: 10.1007/978-94-009-9088-3_1.

[59] “Signal-to-Noise in Optical Astronomy”, https://www.ucolick.org/~bolte/AY257/s_n.pdf

[60] C. Szegedy *et al.*, “Going Deeper with Convolutions,” Sep. 16, 2014, *arXiv*: arXiv:1409.4842. Accessed: Sep. 04, 2024. [Online]. Available: http://arxiv.org/abs/1409.4842

[61] M. Lin, Q. Chen, and S. Yan, “Network In Network,” Mar. 04, 2014, *arXiv*: arXiv:1312.4400. Accessed: Sep. 04, 2024. [Online]. Available: http://arxiv.org/abs/1312.4400

[62] P. Viswanath and T. Hitendra Sarma, “An improvement to k-nearest neighbor classifier,” in *2011 IEEE Recent Advances in Intelligent Computational Systems*, Trivandrum, India: IEEE, Sep. 2011, pp. 227–231. doi: 10.1109/RAICS.2011.6069307.

[63] S. A. Dudani, “The Distance-Weighted k-Nearest-Neighbor Rule,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-6, no. 4, pp. 325–327, Apr. 1976, doi: 10.1109/TSMC.1976.5408784.

[64] P. Y. Taser, “Application of Bagging and Boosting Approaches Using Decision Tree-Based Algorithms in Diabetes Risk Prediction,” *Proceedings*, vol. 74, no. 1, Art. no. 1, 2021, doi: 10.3390/proceedings2021074006.

[65] Y. Huang and P. Zhang, “Evaluation of machine learning approaches for cell-type identification from single-cell transcriptomics data,” *Brief Bioinform*, vol. 22, no. 5, p. bbab035, Sep. 2021, doi: 10.1093/bib/bbab035.

[66] L. Chen, S. Li, Q. Bai, J. Yang, S. Jiang, and Y. Miao, “Review of Image Classification Algorithms Based on Convolutional Neural Networks,” *Remote Sensing*, vol. 13, no. 22, Art. no. 22, Jan. 2021, doi: 10.3390/rs13224712.

[67] F. Yu *et al.*, “Progress in the Application of CNN-Based Image Classification and Recognition in Whole Crop Growth Cycles,” *Remote Sensing*, vol. 15, no. 12, Art. no. 12, Jan. 2023, doi: 10.3390/rs15122988.

[68] M. Gogoi and S. Begum, “Image Classification Using Deep Autoencoders,” Dec. 2017, pp. 1–5. doi: 10.1109/ICCIC.2017.8524276.

[69] E. Pintelas, I. E. Livieris, and P. E. Pintelas, “A Convolutional Autoencoder Topology for Classification in High-Dimensional Noisy Image Datasets,” *Sensors*, vol. 21, no. 22, Art. no. 22, Jan. 2021, doi: 10.3390/s21227731.

[70] D. Tarwidi, S. R. Pudjaprasetya, D. Adytia, and M. Apri, “An optimized XGBoost-based machine learning method for predicting wave run-up on a sloping beach,” *MethodsX*, vol. 10, p. 102119, 2023, doi: 10.1016/j.mex.2023.102119.

[71] T. Chen and C. Guestrin, “XGBoost: A Scalable Tree Boosting System,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug. 2016, pp. 785–794. doi: 10.1145/2939672.2939785.

[72] S. Ratnasingam and J. Muñoz-Lopez, “Distance Correlation-Based Feature Selection in Random Forest,” *Entropy*, vol. 25, no. 9, Art. no. 9, Sep. 2023, doi: 10.3390/e25091250.

[73] A. Senokosov, A. Sedykh, A. Sagingalieva, B. Kyriacou, and A. Melnikov, “Quantum machine learning for image classification,” *Mach. Learn.: Sci. Technol.*, vol. 5, no. 1, p. 015040, Mar. 2024, doi: 10.1088/2632-2153/ad2aef.

[74] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2012. Accessed: Sep. 04, 2024. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html

[75] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, “Densely Connected Convolutional Networks,” Jan. 28, 2018, *arXiv*: arXiv:1608.06993. Accessed: Sep. 04, 2024. [Online]. Available: http://arxiv.org/abs/1608.06993

[76] E. Marchiori, “Class Dependent Feature Weighting and K-Nearest Neighbor Classification,” in *Pattern Recognition in Bioinformatics*, A. Ngom, E. Formenti, J.-K. Hao, X.-M. Zhao, and T. van Laarhoven, Eds., Berlin, Heidelberg: Springer, 2013, pp. 69–78. doi: 10.1007/978-3-642-39159-0_7

[77] M. Marengo and M. C. Sanchez, “A k-NN METHOD TO CLASSIFY RARE ASTRONOMICAL SOURCES: PHOTOMETRIC SEARCH OF BROWN DWARFS WITH SPITZER/IRAC,” *AJ*, vol. 138, no. 1, p. 63, May 2009, doi: 10.1088/0004-6256/138/1/63.

[78] J. Calvo-Zaragoza and A.-J. Gallego, “A selectional auto-encoder approach for document image binarization,” *Pattern Recognition*, vol. 86, pp. 37–47, Feb. 2019, doi: 10.1016/j.patcog.2018.08.011.

[79] D. Lu and Q. Weng, “A survey of image classification methods and techniques for improving classification performance,” *International Journal of Remote Sensing*, vol. 28, no. 5, pp. 823–870, Mar. 2007, doi: 10.1080/01431160600746456.

[80] N. Peng, Y. Zhang, and Y. Zhao, “A SVM-kNN method for quasar-star classification,” *Sci. China Phys. Mech. Astron.*, vol. 56, no. 6, pp. 1227–1234, Jun. 2013, doi: 10.1007/s11433-013-5083-8.

[81] A. Franco-Arcega, L. G. Flores-Flores, and R. F. Gabbasov, “Application of Decision Trees for Classifying Astronomical Objects,” in *2013 12th Mexican International Conference on Artificial Intelligence*, Nov. 2013, pp. 181–186. doi: 10.1109/MICAI.2013.29.

[82] J.-L. Starck and F. Murtagh, “Astronomical image and signal processing: looking at noise, information and scale,” *IEEE Signal Processing Magazine*, vol. 18, no. 2, pp. 30–40, Mar. 2001, doi: 10.1109/79.916319.

[83] M. Ashai, R. G. Mukherjee, S. P. Mundharikar, V. D. Kuanr, and R. Harikrishnan, “Classification of Astronomical Objects using KNN Algorithm,” in *Smart Intelligent Computing and Applications, Volume 1*, V. Bhateja, S. C. Satapathy, C. M.

Travieso-Gonzalez, and T. Adilakshmi, Eds., Singapore: Springer Nature, 2022, pp. 377–387. doi: 10.1007/978-981-16-9669-5_34.

[84] W. Luo, J. Li, J. Yang, W. Xu, and J. Zhang, "Convolutional Sparse Autoencoders for Image Classification," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 7, pp. 3289–3294, Jul. 2018, doi: 10.1109/TNNLS.2017.2712793.

[85] E. C. Vasconcellos *et al.*, "DECISION TREE CLASSIFIERS FOR STAR/GALAXY SEPARATION," *AJ*, vol. 141, no. 6, p. 189, May 2011, doi: 10.1088/0004-6256/141/6/189.

[86] H. Liu, M. Cocea, and W. Ding, "Decision tree learning based feature evaluation and selection for image classification," in *2017 International Conference on Machine Learning and Cybernetics (ICMLC)*, Jul. 2017, pp. 569–574. doi: 10.1109/ICMLC.2017.8108975.

[87] E. Hosseini-Asl, J. M. Zurada, and O. Nasraoui, "Deep Learning of Part-Based Representation of Data Using Sparse Autoencoders With Nonnegativity Constraints," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 12, pp. 2486–2498, Dec. 2016, doi: 10.1109/TNNLS.2015.2479223.

[88] A. N. Herur, R. Tajmohamed, and J. G. Ponsam, "Detection and classification of exoplanets using hybrid kNN model," in *2022 International Conference on Computer Communication and Informatics (ICCCI)*, Jan. 2022, pp. 1–3. doi: 10.1109/ICCCI54379.2022.9741029.

[89] D. C. Ciresan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Flexible, High Performance Convolutional Neural Networks for Image Classification".

[90] A. Bosch, A. Zisserman, and X. Munoz, "Image Classification using Random Forests and Ferns," in *2007 IEEE 11th International Conference on Computer Vision*, Oct. 2007, pp. 1–8. doi: 10.1109/ICCV.2007.4409066.

[91] L. Li, Y. Zhang, and Y. Zhao, "k-Nearest Neighbors for automated classification of celestial objects," *Sci. China Ser. G-Phys. Mech. Astron.*, vol. 51, no. 7, pp. 916–922, Jul. 2008, doi: 10.1007/s11433-008-0088-4.

[92] P. H. Barchi *et al.*, “Machine and Deep Learning applied to galaxy morphology - A comparative study,” *Astronomy and Computing*, vol. 30, p. 100334, Jan. 2020, doi: 10.1016/j.ascom.2019.100334.

[93] M. Viquar, S. Basak, A. Dasgupta, S. Agrawal, and S. Saha, “Machine Learning in Astronomy: A Case Study in Quasar-Star Classification,” in *Emerging Technologies in Data Mining and Information Security*, A. Abraham, P. Dutta, J. K. Mandal, A. Bhattacharya, and S. Dutta, Eds., Singapore: Springer, 2019, pp. 827–836. doi: 10.1007/978-981-13-1501-5_72.

[94] R. Tang, “Machine Learning Meets Astronomy,” in *2020 International Conference on Computing and Data Science (CDS)*, Aug. 2020, pp. 3–6. doi: 10.1109/CDS49703.2020.00008.

[95] D. C. Cireşan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, “Mitosis Detection in Breast Cancer Histology Images with Deep Neural Networks,” in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2013*, K. Mori, I. Sakuma, Y. Sato, C. Barillot, and N. Navab, Eds., Berlin, Heidelberg: Springer, 2013, pp. 411–418. doi: 10.1007/978-3-642-40763-5_51.

[96] D. Ciregan, U. Meier, and J. Schmidhuber, “Multi-column deep neural networks for image classification,” in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2012, pp. 3642–3649. doi: 10.1109/CVPR.2012.6248110.

[97] N. Horning, “Random Forests : An algorithm for image classification and generation of continuous fields data sets,” 2010.

[98] L. Wenzl *et al.*, “Random Forests as a Viable Method to Select and Discover High-redshift Quasars,” *AJ*, vol. 162, no. 2, p. 72, Jul. 2021, doi: 10.3847/1538-3881/ac0254.

[99] L. A. Breslow and D. W. Aha, “Simplifying decision trees: A survey,” *The Knowledge Engineering Review*, vol. 12, no. 01, pp. 1–40, Jan. 1997, doi: 10.1017/S0269888997000015.

Article Number	Year	Algorithm Used	Article Type
[1]	2012	KNN	Experiment
[2]	2019	SVM	Review
[3]	2020	Other (Transfer Learning)	Review
[4]	2015	Other (Noise)	Overview
[5]	2011	Other (General)	Other (Book)
[6]	1989	Other (General)	Other (Book)
[7]	2009	KNN	Experiment
[8]	1955	Other (General AI Proposal)	Other
[9]	2023	SVM	Experiment
[10]	2015	Other (Noise)	Experiment
[11]	2023	Random Forest	Experiment
[12]	2018	Other (General)	Overview
[13]	2022	Autoencoder	Overview
[14]	2015	CNN	Overview
[15]	2023	Other (AI in Astronomy)	Analysis
[16]	2013	SVM	Experiment
[17]	2007	Other (General)	Other (Book)
[18]	2001	Other (Signal Processing)	Experiment
[19]	2020	Autoencoder	Experiment
[20]	2006	Other (General)	Overview
[21]	2018	Autoencoder	Experiment
[22]	2012	Autoencoder	Review
[23]	2023	CNN	Review
[24]	2021	Autoencoder	Experiment
[25]	1997	Other (Noise)	Experiment
[26]	2007	Other (General)	Experiment
[27]	2018	CNN	Experiment
[28]	1993	Decision Trees	Other (Case Study)
[29]	2020	Other (History)	Review
[30]	2013	Other (Self-Organizing Map)	Overview
[31]	2024	Other (General)	Overview
[32]	2024	Other (General)	Overview
[33]	2024	Other (General)	Overview
[34]	2008	Other (ANN)	Overview
[35]	2023	KNN	Review
[36]	2024	KNN	Overview

[37]	2024	Other (Machine Learning)	Overview
[38]	2021	SVM	Review
[39]	2019	Other (Machine Learning)	Overview
[40]	2024	Autoencoder	Review
[41]	2008	Decision Trees	Experiment
[42]	2024	Other (General)	Overview
[43]	1994	Other (General)	Overview
[44]	2015	Other (PCA)	Review
[45]	1993	Other (PCA)	Review
[46]	2012	Decision Trees, Random Forest	Overview
[47]	2024	Autoencoder	Experiment
[48]	2020	KNN	Review
[49]	2006	Decision Trees	Experiment
[50]	2021	Other (t-SNE)	Experiment
[51]	2001	SVM	Overview
[52]	2011	Other (General)	Overview
[53]	1980	Other (X-Ray Astronomy)	Overview
[54]	2023	Other (General)	Experiment
[55]	2006	Other (General)	Overview
[56]	2015	CNN	Overview
[57]	2024	Other (Radio Astronomy)	Overview
[58]	1980	Other (X-Ray Astronomy)	Overview
[59]	2024	Other (Noise)	Overview
[60]	2014	CNN	Experiment
[61]	2014	CNN	Experiment
[62]	2011	KNN	Experiment
[63]	1976	KNN	Experiment
[64]	2021	Decision Trees	Experiment
[65]	2021	Other (Machine Learning)	Experiment
[66]	2021	CNN	Review
[67]	2023	CNN	Review
[68]	2017	Autoencoder	Experiment
[69]	2021	Autoencoder	Experiment
[70]	2023	XGBoost	Experiment
[71]	2016	XGBoost	Overview
[72]	2023	Random Forest	Experiment
[73]	2024	Other (Quantum ML)	Experiment
[74]	2022	Quantum Machine Learning	Overview
[75]	2019	Autoencoders	Experiment

[76]	2023	CNN	Experiment
[77]	2015	Random Forest	Review
[78]	2021	SVM	Experiment
[79]	2024	Quantum Machine Learning	Review
[80]	2018	Autoencoders	Experiment
[81]	2016	CNN	Overview
[82]	2020	KNN	Review
[83]	2017	CNN	Experiment
[84]	2021	Quantum Machine Learning	Overview
[85]	2022	CNN	Review
[86]	2023	Autoencoders	Overview
[87]	2019	KNN	Experiment
[88]	2021	CNN	Experiment
[89]	2012	Decision Trees	Overview
[90]	2017	CNN	Review
[91]	2022	Random Forest	Overview
[92]	2023	Autoencoders	Experiment
[93]	2011	SVM	Review
[94]	2024	Quantum Machine Learning	Experiment
[95]	2021	Autoencoders	Review
[96]	2020	SVM	Review
[97]	2023	CNN	Experiment
[98]	2024	Quantum Machine Learning	Overview
[99]	2020	Random Forest	Review

CNN

Article Number	Year	Algorithm Used	Article Type
[14]	2015	CNN	Overview
[23]	2023	CNN	Review
[27]	2018	CNN	Experiment
[56]	2015	CNN	Overview
[60]	2014	CNN	Experiment
[61]	2014	CNN	Experiment
[66]	2021	CNN	Review
[67]	2023	CNN	Review
[76]	2023	CNN	Experiment
[81]	2016	CNN	Overview
[83]	2017	CNN	Experiment

[85]	2022	CNN	Review
[88]	2021	CNN	Experiment
[90]	2017	CNN	Review
[97]	2023	CNN	Experiment

Autoencoders

Article Number	Year	Algorithm Used	Article Type
[13]	2022	Autoencoder	Overview
[19]	2020	Autoencoder	Experiment
[21]	2018	Autoencoder	Experiment
[22]	2012	Autoencoder	Review
[24]	2021	Autoencoder	Experiment
[40]	2024	Autoencoder	Review
[47]	2024	Autoencoder	Experiment
[68]	2017	Autoencoder	Experiment
[69]	2021	Autoencoder	Experiment
[75]	2019	Autoencoder	Experiment
[80]	2018	Autoencoder	Experiment
[86]	2023	Autoencoder	Overview
[92]	2023	Autoencoder	Experiment
[95]	2021	Autoencoder	Review

SVM

Article Number	Year	Algorithm Used	Article Type
[2]	2020	SVM	Review
[9]	2023	SVM	Experiment
[16]	2013	SVM	Experiment
[38]	2021	SVM	Review
[51]	2001	SVM	Overview
[78]	2021	SVM	Experiment
[93]	2011	SVM	Review
[96]	2020	SVM	Review

KNN

Article Number	Year	Algorithm Used	Article Type

[1]	2012	KNN	Experiment
[7]	2009	KNN	Experiment
[35]	2023	KNN	Review
[36]	2024	KNN	Overview
[48]	2020	KNN	Review
[62]	2011	KNN	Experiment
[63]	1976	KNN	Experiment
[82]	2020	KNN	Review
[87]	2019	KNN	Experiment

Decision Trees

Article Number	Year	Algorithm Used	Article Type
[28]	1993	Decision Trees	Other (Case Study)
[41]	2008	Decision Trees	Experiment
[46]	2012	Decision Trees, Random Forest	Overview
[49]	2006	Decision Trees	Experiment
[64]	2021	Decision Trees	Experiment
[89]	2012	Decision Trees	Overview
[64]	2021	Decision Trees	Experiment

Random Forest

Article Number	Year	Algorithm Used	Article Type
[11]	2023	Random Forest	Experiment
[72]	2023	Random Forest	Experiment
[77]	2015	Random Forest	Review
[91]	2022	Random Forest	Overview
[99]	2020	Random Forest	Review

Other

Article Number	Year	Algorithm Used	Article Type
[5]	2011	Other (General)	Other (Book)

[6]	1989	Other (General)	Other (Book)
[8]	1956	Other (General AI Proposal)	Other
[10]	2016	Other (Noise)	Experiment
[15]	2024	Other (AI in Astronomy)	Analysis
[20]	2006	Other (General)	Overview
[26]	2007	Other (General)	Experiment
[29]	2020	Other (History)	Review
[30]	2013	Other (Self-Organizing Map)	Overview
[31]	2024	Other (General)	Overview
[32]	2024	Other (General)	Overview
[33]	2024	Other (General)	Overview
[34]	2008	Other (ANN)	Overview
[37]	2024	Other (Machine Learning)	Overview
[3]	2021	Other (Transfer Learning)	Review
[52]	2011	Other (General)	Overview
[53]	1980	Other (X-Ray Astronomy)	Overview
[54]	2023	Other (General)	Experiment
[55]	2006	Other (General)	Overview
[4]	2024	Other (General)	Overview
[65]	2021	Other (Machine Learning)	Experiment
[12]	2024	Other (General)	Overview
[17]	2007	Other (General)	Other (Book)
[18]	2001	Other (Signal Processing)	Experiment
[25]	1997	Other (Noise)	Experiment
[39]	2019	Other (Machine Learning)	Overview
[42]	2024	Other (General)	Overview
[43]	1994	Other (General)	Overview
[44]	2015	Other (PCA)	Review
[45]	1993	Other (PCA)	Review
[50]	2021	Other (t-SNE)	Experiment

[57]	2024	Other (Radio Astronomy)	Overview
[58]	1980	Other (X-Ray Astronomy)	Overview
[59]	2024	Other (Noise)	Overview
[73]	2024	Other (Quantum ML)	Experiment

XBoost - Quantum Machine Learning

Article Number	Year	Algorithm Used	Article Type
[70]	2023	XGBoost	Experiment
[71]	2016	XGBoost	Overview
[74]	2022	Quantum Machine Learning	Overview
[79]	2024	Quantum Machine Learning	Review
[84]	2021	Quantum Machine Learning	Overview
[94]	2024	Quantum Machine Learning	Experiment
[98]	2024	Quantum Machine Learning	Overview