National
Technical
University of
Athens

Master Thesis

Covid-19 Prediction using Machine Learning on CT scans

ZERKELIDIS DIMITRIOS

SUPERVISOR:
GEORGIOS MATSOPOULOS
PROFESSOR

Athens 2021

National Technical University of Athens
Electrical Engineering and Computer Science
Master in Machine Learning and Data Science

Covid-19 Prediction using Machine Learning on CT scans

Master thesis
of
Zerkelidis Dimitrios

Supervisor: Georgios Matsopoulos

Approved from the three-member committee in April 2021.

(Signature)	*(Signature)*	*(Signature)*
....	....	....
Supervisor	Committee member 1	Committee member 2
Georgios Matsopoulos	Dimitrios Koutsouris	Panagiotis Tsanakas
Professor N.T.U.A.	Professor N.T.U.A.	Professor N.T.U.A.

Athens, April 2021

....
Zerkelidis Dimitrios
Graduate Student in Machine Learning and Data Science in N.T.U.A.

Current thesis has been devised in the context of activities of the Bio-Medical Imaging Group research team of National Technical University of Athens. https://biomig.ntua.gr/

Abstract

In 2020 our world and peoples' lives changed rapidly due to Corona Virus Disease 19 (Covid-19). The virus was able to spread very fast, forcing the World Health Organisation (WHO) to declare the new disease as a pandemic.

As a result, a vast need, for immediate detection of patients that ail from Covid-19, was created very soon. Current tests are usually slow, expensive and prone to mistake. In this endeavor, a dataset from CT scans is collected, which is DICOM image series of each patient. This dataset is consisted of people that tested either positive or negative to Covid-19.

At the very beginning, while having the DICOM image series from our patients, a lung-segmentation algorithm is implemented. The algorithm that is chosen for extracting the lung area is multiple thresholding. Also, gaussian smoothing and hole filling is used. Hence, only the Region-Of-Interest (ROI) is used which is the patient's lung area. This will work as a mask for the real image.

Later, this thesis focuses on extracting features in the Region Of Interest (ROI) of the CT scans using medical image processing algorithms. These features can be divided into the next categories: 1)First Order Statistics, 2)GLCM, 3)3D-Shape, 4) GLSZM, 5)GLRLM, 6)NGTDM and 7)GLDM. In the next stage, feature selection is applied to identify the best subset of features that can predict the existence of Covid-19 to a patient.

After feature extraction, both supervised and unsupervised machine learning algorithms are implemented for patients' binary classification is conducted. Supervised methods such as Support Vector Machines, K Nearest Neighbors, Decision Trees with bagging and boosting methods are used. On the other side, as unsupervised models, Isolation Forests, Local Outlier Factor and Auto-encoders with threshold on reconstruction error will be tested. In the unsupervised case, the problem is viewed as anomaly detection because of the high imbalance between the classes.

Our metric evaluation will be based on AUROC scores, sensitivity, specificity and F1-score, since our dataset is highly imbalanced. For that reason data augmentation techniques will be tested too. Finally, the features with the best clinical impact will be mentioned.

Keywords

Covid-19, Binary Classification, Machine Learning, Prediction, Supervised learning, Unsupervised learning, CT scans, AUROC, Anomaly Detection Feature Extraction, F1-score, Imbalanced Dataset, Data Augmentation Lung Segmentation, Region Of Interest

Περίληψη

Το 2020 ο κόσμος και οι ζωές των ανθρώπων άλλαξαν ραγδαία λόγω της ασθένειας κορωνοϊού 19(Covid-19). Ο ιός μπόρεσε να εξαπλωθεί πολύ γρήγορα, αναγκάζοντας τον Παγκόσμιο Οργανισμό Υγείας (ΠΟΥ) να κηρύξει τη νέα ασθένεια ως πανδημία.

Ως αποτέλεσμα, μια τεράστια ανάγκη, για άμεση ανίχνευση ασθενών από τον Covid-19, δημιουργήθηκε πολύ σύντομα. Οι τρέχουσες δοκιμές είναι συνήθως αργές, ακριβές και επιρρεπείς στο σφάλμα. Στην προσπάθειά αυτή, έχουν συλλεχθεί αξονικές θώρακος, οι οποίες είναι DICOM σειρές εικόνων του κάθε ασθενή. Αυτό το σύνολο δεδομένων αποτελείται από άτομα που έχουν επισημανθεί είτε θετικά είτε αρνητικά στον κορωνοϊό (Covid-19).

Στην αρχή, έχοντας ως δεδομένα τις σειρές εικόνων DICOM από τους ασθενείς μας, εφαρμόζεται ένας αλγόριθμο τμηματοποίησης πνευμόνων. Ο αλγόριθμος που επιλέχθηκε για την εξαγωγή της περιοχής των πνευμόνων από τις εικόνες ήταν με χρήση πολλαπλών κατωφλιών. Επίσης, έγινε χρήση των μεθόδων gaussian smoothing και hole filling. Ως εκ τούτου, λαμβάνονται μόνο οι περιοχές ενδιαφέροντος (ROIs) που είναι η περιοχή των πνευμόνων του ασθενούς. Η περιοχή ενδιαφέροντος θα λειτουργήσει ως μάσκα στην πραγματική εικόνα.

Στη συνέχεια, για την εξαγωγή χαρακτηριστικών στην περιοχή ενδιαφέροντος (ROI) από τις αξονικές θώρακος χρησιμοποιούνται ιατρικοί αλγόριθμοι επεξεργασίας εικόνας. Οι αλγόριθμοι αυτοί δίνουν τις εξής κατηγορίες χαρακτηριστικών: 1)First Order Statistics, 2)GLCM, 3)3D-Shape, 4) GLSZM, 5)GLRLM, 6)NGTDM και 7)GLDM. Στο επόμενο στάδιο, η επιλογή χαρακτηριστικών εφαρμόζεται για τον προσδιορισμό του καλύτερου υποσυνόλου αυτών των χαρακτηριστικών, που μπορούν να προβλέψουν την ύπαρξη του Covid-19 σε ένα ασθενή.

Έπειτα από την εξαγωγή των χαρακτηριστικών, εφαρμόζονται αλγόριθμοι, επιβλεπόμενης και μη επιβλεπόμενης μάθησης για τη δυαδική ταξινόμηση των ασθενών. Μέθοδοι επιβλεπόμενης μάθησης επιστρατεύονται όπως οι αλγόριθμοι, Support Vector Machines, K Nearest Neighbor, Decision Trees με μεθόδους bagging ή boosting. Από την άλλη πλευρά, ως μη επιβλεπόμενης μάθησης μέθοδοι αλγόριθμοι όπως, Isolation Forests, Local Outlier Factor και Αυτοκωδικοποιητές με χρήση κατωφλιού το reconstruction error θα δοκιμαστούν. Στην περίπτωση των μη επιβλεπόμενων μεθόδων, το πρόβλημα αντιμετωπίζεται ως πρόβλημα ανίχνευσης ανωμαλιών λόγω της μεγάλης ανισσότητας των κλάσεων.

Η αξιολόγηση των προγνωστικών μοντέλων θα βασιστούν στις μετρικές AUROC, ευαισθησία (sensitivity), ειδικότητα (specificity), F1-σκορ, καθώς τα δεδομένα μας χαρακτηρίζονται από μεγάλη ανισορροπία. Για το λόγο αυτό θα δοκιμαστούν κάποιες τεχνικές επαύξησης δεδομένων.

Λέξεις-Κλειδιά

Κορονοϊός, Δυαδική Ταξινόμηση, Μηχανική Μάθηση, Πρόγνωση Επιβλεπόμενη μάθηση, Μη Επιβλεπόμενη μάθηση, Αξονικές στήθους, Ανίχνευση Ανωμαλιών, AUROC, F1-σκορ, Εξαγωγή Χαρακτηριστικών, Επαύξηση Δεδομένων, Μη ισορροπομένο σύνολο δεδομένων, Τμηματοποίηση Πνευμόνων, Περιοχή Ενδιαφέροντος

I would like thank professor Georgios Matsopoulos for trusting me in devising this thesis.

In addition, I am thankful to Teodore Economopoulos for his assistance and clearing out my questions.

–ZERKELIDIS DIMITRIOS

Contents

List of Figures

List of Tables

Chapter 1

Introduction

COVID-19 pandemic, also known as the coronavirus pandemic, is an ongoing pandemic of coronavirus disease 2019 also known as COVID-19. This disease is caused by Severe Acute Respiratory Syndrome Coronavirus 2 (SARS-CoV-2), first identified in December 2019 in Wuhan, China. The World Health Organization declared the outbreak a Public Health Emergency of International Concern in January 2020 and a pandemic in March 2020. As of 28 November 2020, more than 61.8 million cases have been confirmed, with more than 1.44 million deaths attributed to COVID-19. The death rate is said to be around 2%. Although, we cannot rely on our data, since the virus spreads very fast and many cases probably will never been confirmed.

Coronavirus 19 can be spread primarily from saliva and other bodily fluids and excretions, because they can form tiny droplets able to spread with various actions. These actions can be breathing, coughing, sneezing, singing or speaking by an infected person. Common symptoms include fever, cough, fatigue, breathing difficulties, loss of smell and taste. It usually takes close to 5 to 14 days for incubation.

This pandemic has caused global social and economic disruption, including the largest global recession since the Great Depression. Events have been cancelled, educational institutions have been partially closed, misinformation has been spread and widespread supply shortages exacerbated by panic buying, agricultural disruption and food shortages.

As of October 2020, there were 321 vaccine candidates in development worldwide. In November 2020, Pfizer Inc, Moderna and University of Oxford announced positive results from interim analyses of their 3rd phase vaccine trials. Until people have access to vaccine, to prevent from COVID-19 they should get used habits such as hand-washing, avoid interaction in crowded places and wear of masks.

Governments all over the world try to find ways to identify COVID-19 cases immediately. Laboratory tests are of common use such as PCR testing or serologic testing. In addition the diagnosis from medical images by specialised doctors is possible too. The use of chest CT scans can reveal patterns useful to pulmonologists or radiologists. The same thing can be said for chest X-ray images although the prediction of the disease in that case is less accurate.

Based on the above paragraphs, it is understandable why there was a need for mobilization of all technologies and the research community in the fight against coronavirus 19. The scope of this thesis is to contribute in this fight and provide a Computer-Aided Detection (CAD) to doctors, so that they can predict COVID-19 in patients faster, cheaper and more accurately.

Machine learning can help in the design of reliable CAD systems. By using high resolution image series such as CT scans, we can extract plenty features that will be inserted into a prediction model.

1. Thesis Structure

This thesis consists of 6 chapters. The first chapter makes an introduction about COVID-19 pandemic. It points out the need for fast identification of patients and the socioeconomic impact of the disease. Also, it describes the structure of this thesis.

In the second chapter, there are references of other related works. We describe the different methods that other researchers used to approach the problem and analyze their results. Then we can compare their results with ours and make conclusions.

The third chapter is about describing the datasets that were used. In addition we describe the features that we extracted which will be passed into the machine learning model. Also, an explanation is given on the evaluation metrics that will be used.

The fourth chapter consists of the necessary theory to understand machine learning algorithms that were used. It provides a mathematical analysis of machine learning algorithms and an intuitive explanation.

The firth chapter is about the experiments and the results. Various methods will be mentioned in the experiment phase such as data augmentation and feature selection. And for our results we will use specific metric evaluations that take into imbalanced datasets.

The sixth and the last chapter condenses the results of the previous one. It formulates the basic conclusions and sets goals for feature research in this major issue.

Chapter 2

Related Work

The chapter of the bibliographic review aims at compiling the existing bibliography related to the subject of the work. The main purpose of the literature review is to provide the existing scientific background required to establish the scientific validity of the work. Firstly, methods on COVID-19 prediction will be described using CT scans. Then, the same will be applied for Chest X-RAY (CXR) images. In the end, a research on feature extraction in medical images will be mentioned.

1. Literature Review on CT and CXR datasets

This section will refer to methods used in predicting COVID-19 with CT scans.

One of the proposed methods of Wang et al. is a deep learning model named COVID-19 Detection Neural Network (COVNet) that is developed to extract visual features from volumetric chest CT scans for the detection of COVID-19 [36]. In this paper, they included in their dataset not only COVID-19 samples but also Pneumonia and Non-Pneumonia. This happened because they needed to ensure that the model distinguishes between COVID-19 and Pneumonia. To compare COVNet's performance the metrics, which Wang et al. used are AUC, sensitivity and specificity. The dataset consisted of 4,356 patients, in which 1296 were ailed with corona-Virus disease 19, the 1735 were ailed with Pneumonia and the last 1325 were in a healthy condition. The model yielded AUC of 0.96.

Another research of Kassani et al. proposed a method that combines CNN descriptors and machine learning algorithms [28]. Specifically, well known CNN deep learning models were tried such as, MobileNet, DenseNet121, DenseNet201, XCeption, InceptionV3, InceptionResNetv2, Resnet50, VGG19, NasNetLarge, Resnet152v2 and others. Using these models they extracted feature vectors that were fed as inputs in prediction models such as Decision Tree, Random Forest, XGBoost, AdaBoost, Bagging Tree Classifier and LightGBM. For the CNN models to avoid overfitting because of small and imbalanced dataset various strategies were used. These were data augmentation, transfer learning and fine-tuning. The best result that was achieved was 99.00% using DenseNet121 and Bagging Tree Classifier.

Barstuğan et al. in their paper used CT images by patches sizes of 16x16, 32x32, 48x48 and 64x64 and extracted features for each patch size by using Grey Level Co-occurence Matrix (GLCM), Local Directional Pattern (LDP), Grey Level Run Length Matrix (GLRLM), Grey Level Size Zone Matrix (GLSZM) and Discrete Wavelet Transform (DWT) [13]. Then, for classification SVM method was selected. The best result was obtained as 99.68% specificity with 10-fold cross-validation and GLSZM feature extraction method on a subset with 5912 non-infected and 6940 infected patches.

Gozes et al. developed an AI system for detecting, and tracking of Coronavirus [27]. They built 2D and 3D deep learning models using multiple international datasets. Their system is consisted of two subsystems A and B. On subsystem A, commercial software is used for nodule and opacity detection from 3D shaped lungs. In addition it can calculate measurements such as Hounsfield values. The second subsystem (B) is extracting the ROI of the image using U-net architecture. Then a ResNet-50-2D architecture is used with pre-trained weights on ImageNet[5]. Then the model is fine-tuned and trained with data augmentation to improve classification accuracy. A score of 0.996 AUC was obtained.

Narin et al. tried using pre-trained deep learning CNN models for predicting COVID-19 from Chest X-RAY images [44]. The pre-trained weights were loaded from ImageNet[5] dataset. Then three models were trained with three different datasets. The first with COVID-19 and Normal examined patients, second with COVID-19 and Bacterial Pneumonia examined patients and the last with COVID-19 and Viral Pneumonia examined patients. The best result was obtained by ResNet50 pre-trained model that using 5 fold cross validation scored 96.1% accuracy on the first dataset, 99.5% on the second dataset and 99.7% on the last dataset.

Another research from University of Patras, which was conducted by Apostolopoulos et al. used also pre-trained models using ImageNet on CXR dataset [11]. They utilised transfer learning on many CNN models with imagenet pre-trained weights and the best result was obtained from VGG19 model with 98.75% accuracy and specificity.

Heidari et al. conducted a research to develop a CAD system of Chest X-ray images to predict COVID-19. This system removes the majority of diaphragm regions, uses Histogram Equalization and Bilateral low-pass filter in the original image[31]. Then, the original image and the two filtered images are used to form a pseudo color image which is fed into three input channels of a transfer learning based CNN model. The CNN model yields specificity of 98.0% and sensitivity of 98.4% in predicting COVID-19 against Normal patient.

In the following research that was conducted by Brunese et al. VGG-16 was used with transfer learning in a three phase approach [16]. The first phase detects if in a CXR there is presence of a pneumonia. The second phase discern between COVID-19 and Pneumonia. The last step is aimed to localise the areas in the X-ray symptomatic of the COVID-19 presence using GRAD-CAM [49]. This method yield a 0.97 accuracy with 6,523 total samples. 250 with covid-19, 2,753 Pneumonia and 3,520 Healthy patients. However the high accuracy, the dataset is highly imbalanced. The F-measure that the research [16] yields is 0.89.

In the end, two additional papers will be referenced, which proposed deep learning CNN models explicitly for COVID-19 prediction utilizing Chest X-ray datasets. The first one is named COVID-Net [55]. COVID-Net was trained on a large dataset named COVIDx [2]. This model is characterised by high architectural diversity and selective long-range connectivity. Also, it yields 93.3% accuracy on Pneumonia, Covid-19 and Normal patients. The second research proposes a model named CovXNet which also trains on three class dataset (Covid-19, Normal, Pneumonia)[38]. CovXnet is a CNN based model that utilizes depth-wise convolution with varying dilation rates for efficiently extracting diversified features from CXRs. This framework uses a stacked model which is trained as follows. In the first phase, training happens with a Non-Covid database. The second phase is a transfer learning using Covid and Non-Covid data. Lastly, the third phase is the inference phase with localization using GRAD-CAM[49]. CovXNet yields an accuracy of 90.3%.

2. Literature Review on Feature Extraction in medical images

In the current section, we will mention researches that are based on feature extraction in medical images. Previously, we referred to a paper of Barstuğan et al. which actually uses many feature extraction methods such as GLCM, LDP, GLRLM, GLSZM and DWT [13].

In a research of Gletsos et al. experiments in classifying hepatic lesions using Computer Tomography (CT) images. [26]. Firstly, Regions of Interest (ROIs) are taken from computed tomography (CT) images. Then there is the feature extraction, in which the average gray level and 48 texture characteristics are derived from the spatial gray-level co-occurence matrices (GLCM), obtained from the ROIs. Then, the following characteristics are extracted using the GLCM: 1) contrast, 2) correlation, 3) angular second moment, 4)variance, 5) inverse difference moment, 6)cluster tendency, 7) entropy, 8)homogeneity. GLCM and the obtained features are related to both angular direction and pixel distance. In this work, the features were calculated for distances of 1, 2, 4, 6, 8 and 12 pixels horizontally,vertically and diagonally and for angular directions 0° , 45° , 90° and 135°. Then all these 49 features are fed into a neural network. The model yields performance 100% accuracy.

Vlachokosta et al. proposes also a methodology for hysteroscopical image classification using feature extraction methods and machine learning algorithms [54]. Firstly, preprocessing takes place which is the conversion of an RGB endometrial image into a grayscale image. Then the contrast limited adapted histogram equalization algorithm is performed in the grayscale image, to enhance vessel structures and remove noise and intensity in homogeneities. Then vessel centerline extraction is implemented. After that the feature extraction happens in a similar way using GLCM features as in [26]. Finally, the sequential floating forward search (SFFS) as presented here [12], is used for feature selection. The classifiers that Vlachokosta et al. used are Fuzzy C-means and Feed Forward Neural Networks.

Matsopoulos et al. in a paper about a WWW-based medical system, called MITIS uses image processing techniques in mammographic data [40]. The image processor provides capabilities for manipulating DICOM format images. Then there is the image enhancement tool that includes a filter for histogram equalization, unsharp masking (Laplacian or Gaussian) and the contrast limited histogram equalization. Then there is the Image segmentation tool that isolates the Regions of Interest (ROIs) both manually and automatically. The segmentation of specific ROIs of mammographic images is performed by an algorithm based on mathematical morphology [56]. After extracting the ROIs, Matsopoulos et al. [40] extract First-Order statistics features, Second-Order statistics features, features based on wavelets and features based on Law Masks and Fractal Theory.

In a research of Namita Aggarwal and R.K. Agrawal use First and Second order statistics to classify Alzheimer's disease based on T2-weighted MRI brain image [9]. To evaluate their model metrics such as sensitivity, specificity and accuracy were utilized. Their experiments show a significant advantage of First and Second order statistical features over the wavelet transformation methods. This occurred in all classifiers and all evaluation metrics. Paper [9] defines first-order histogram P(I) and then calculates mean, variance, skewness and kurtosis. The second order statistics features are calculated by GLCM matrix using measures such as ASM, contrast, correlation, Homogeneity and Entropy as mentioned also in [26].

First and Second order statistics are also used in [46]. In this research Myanmar paper currency recognition system based on First Order Statistics, Gray Level Co-occurrence Matrix (GLCM), and k-Nearest Neighbor (K-NN) is presented. Recognition rate yields 99-100%. Finally, Korda et al. used first order statistics for automatic identification of oculomotor behavior such as saccades, blinks, fixations from time series of eye's angular displacement [35]. Specifically, the features that got extracted were Max value, Min value, Mean value, standard deviation, kurtosis, skewness, energy of signal and entropy. The model yields accuracy 95.9%.

Chapter 3

Dataset and Feature Extraction

1. CT SCANS

1..1 Theory

A CT scan is using xrays in order to produce a 3D picture of human body organs such as lungs. The way CT scan consists of the following:

- ▶ Patient sits on a bed stably.
- ▶ An X-ray tube is throwing out x-rays to the patient.
- ▶ A screen, which is always opposite to the X-ray tube, interacts with these xrays that pass through the patient.
- ▶ A big turn-able table to spin around to get images from different angles.

Essentially, the x-rays are passing through the patient at different angles and a 3D picture is built with the help of computers. Hence the name Computer Tomography. Then, the computes split up organs in the body into small volumes called voxels. Every organ will have million of voxels.

A **voxel** is a cube inside of a 3D model that contains a position inside a 3D grid and a single value color. While the definition of a **pixel** is a square inside a 2D grid of an image with a single color value. For a visual explanation check 3.1

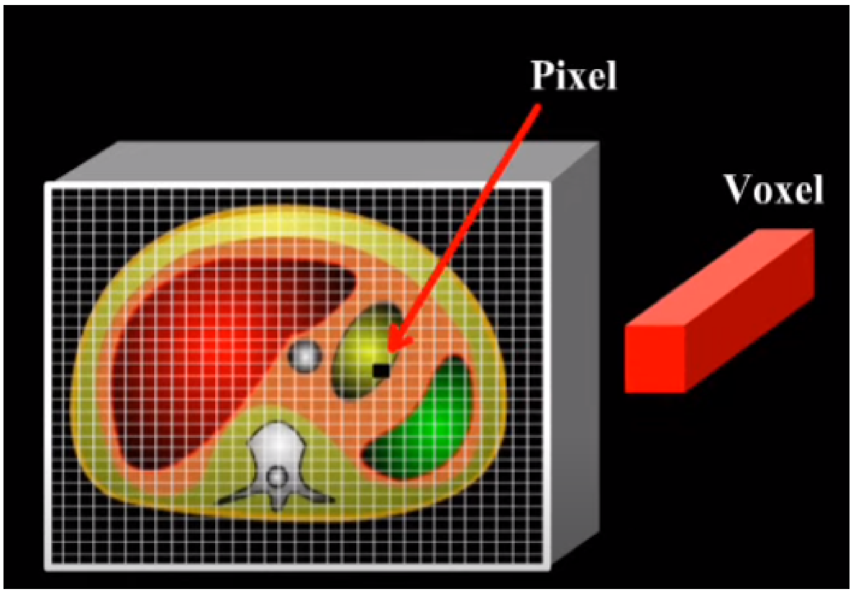


Figure 3.1: An image showing a pixel and a voxel [3]

Each voxel is given a number, which represents the extent to which x-rays are attenuated when they pass through it. For example, a piece of bone will be split up into voxels and each of these voxels will have a high number, because bone tends to attenuate x-rays, due to less intensity before passing the bone.

However, if x-rays pass through a piece of skin, it will be split up into voxels also but the number will be low. This will happen because it does not hinder the x-rays penetration. The intensity of the x-ray after it is passed through the skin, will be as high as it was before going through it.

In contrary to conventional x-ray which uses a fixed cube that sends x-rays in only one direction, a CT scanner uses a motorized x-ray source that shoots narrow beams of x-rays as it rotates around the patient. CT image slices of a patient can either be displayed individually in 2D form or stacked together to generate a 3D construct. CT scans give a more detailed view than conventional x-rays on the inside of human body.

Finally, Hounsfield Unit (HU) will be mentioned to have a complete view of CT images. HU is a measure of radiodensity, a value used to measure density on CT scans. Some reference points in HU scale are water that equals 0 HU, air that equals -1000 HU and bone which is around +1000 HU. Below we provide a mathematical definition:

$$HU = 1000 \times \frac{\mu - \mu_{water}}{\mu_{water} - \mu_{air}} \tag{3.1}$$

μ_{water} and μ_{air} are respectively the linear attenuation coefficients of water and air. While μ is the average linear attenuation coefficient in a voxel.

1..2 CT scans from Greek Hospital

The dataset is collected from a Greek hospital. There are 101 patients, who went through a CT scan. Nine of them have been diagnosed with Covid-19 and the others without Covid-19.

Covid-19 Positive	9
Covid-19 Negative	89
Uncertain	3

Table 3.1: Description of dataset

In the next figure 1..2 two images are positioned. The first one shows a slice of a patient with Covid-19 and the second one a slice of a patient without Covid-19.

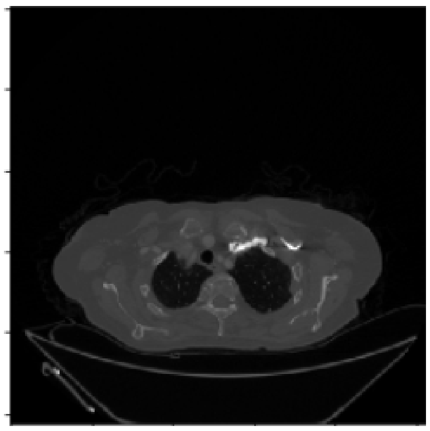


Figure 3.2: Patient's CT slice with Covid-19

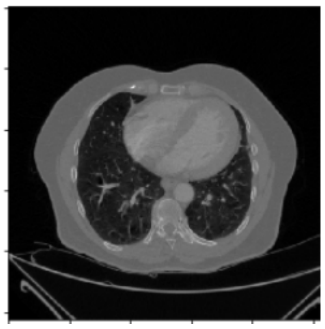


Figure 3.3: Patient's CT slice without Covid-19

An additional figure will be shown with a summarization of a patient's with Covid-19 CT series 3.4.

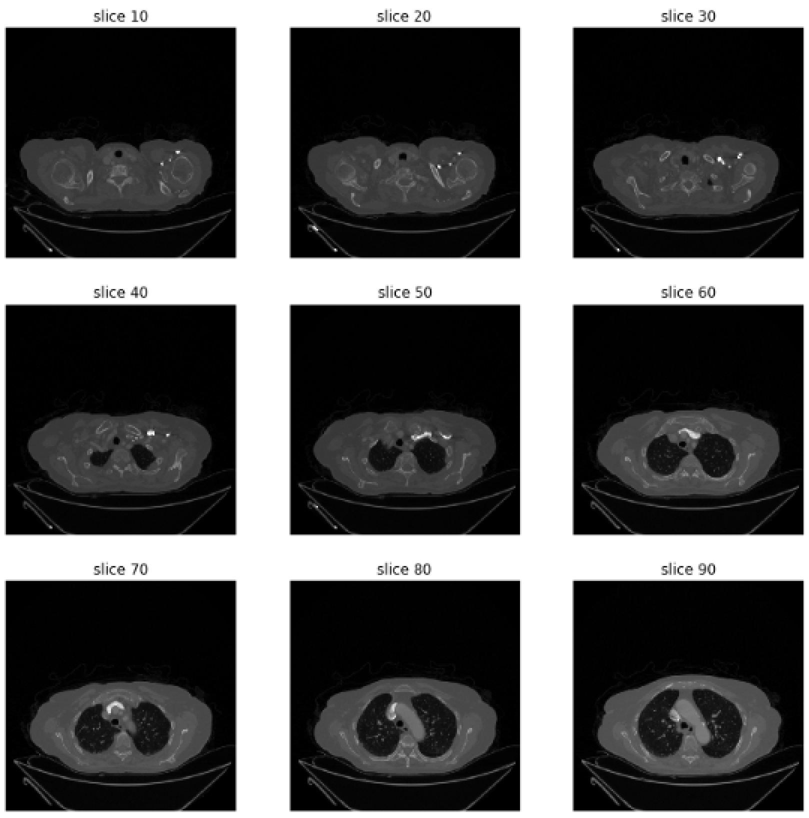


Figure 3.4: An ordered summarization of patient's CT series

Finally, for the same patient as in 3.4 a Hounsfield Units Histogram is plotted3.5. By observing this histogram it is possible to see that most of the image consists of air(many -1000 values) and from water, fat and soft tissue (many values close to 0).

2. Lung Segmentation with Thresholding

To segment lungs from the CT, multiple thresholding is used. That means, that two thesholds are used to mask the regions that we are interested in.

To obtain the two thresholds T1 = -700 and T2 = -800, a research on the dicom scans was conducted. By looking a large amount of the

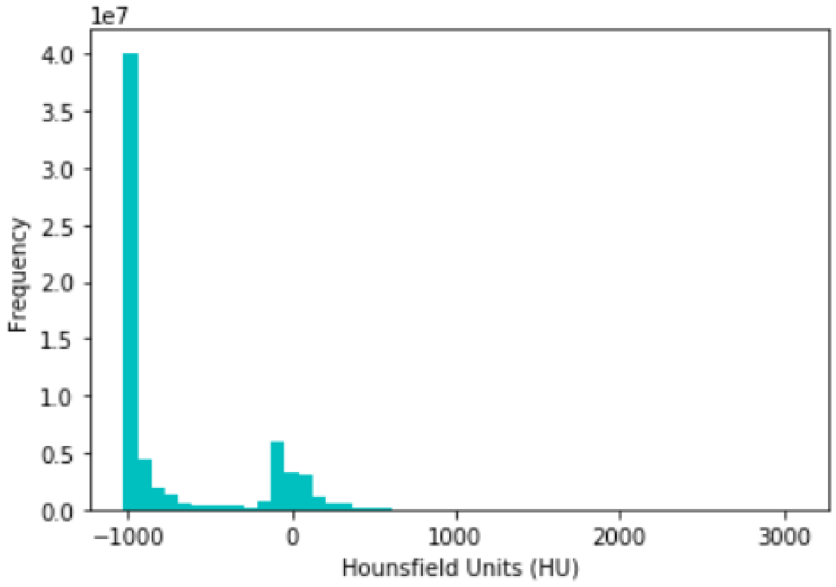


Figure 3.5: Covid-19 patient HU histogram

images, it was observed that the interested values were between (-800, -700). Usually, around -700 are the lung's pixels. In 3.2, we provide the multiple thresholding equation.

$$G(x,y) = \begin{cases} 0, & \text{If G(x,y)} \geq \text{-700.} \\ 1000, & \text{If G(x,y)} \geq \text{-800 and If G(x,y)} \leq \text{-700 .} \\ 0, & \text{If G(x,y)} \leq \text{-800.} \end{cases} \tag{3.2}$$

In the below figure 2., a CT slice will be shown together with the lung segmentation.

3. Feature Extraction and Computational Analysis

In this part of the chapter a detailed description of the feature extraction methods will be given. For that purpose pyRadiomics framework was used and in the next subsections we will describe the equations as defined in [53]. These features can be subdivided into the following classes:

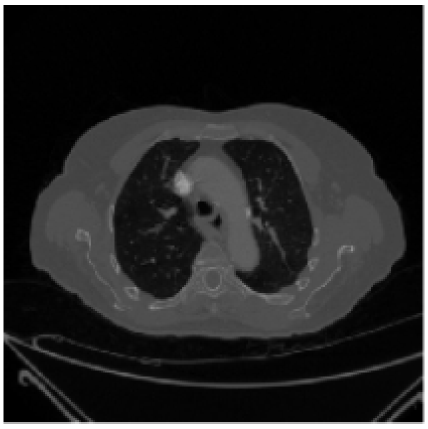


Figure 3.6: CT slice

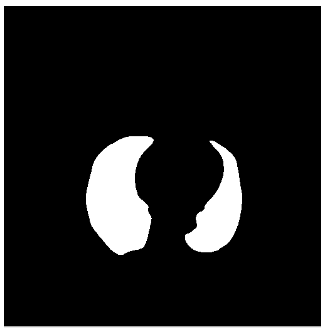


Figure 3.7: Lung Segmentation

1. First Order Statistics [9, 35, 46, 40]
2. 3D Shape-Based [37]
3. Gray Level Coccurence Matrix (GLCM) [13, 26, 54, 40]
4. Gray Level Run Length Matrix (GLRLM) [13, 51, 32, 25, 20]
5. Gray Level Size Zone Matrix (GLSZM) [13, 39]
6. Neighbouring Gray Tone Difference Matrix (NGTDM)[10]
7. Gray Level Dependence Matrix (GLDM) [50]

3..1 First Order Statistics

It is common to scientific literature to use First Order Statistics on image classification problems. Gray levels distribution can determine an image's texture. These type of features are used to describe how voxel intensities are distributed within the region of an image (using ROI-mask), through a variety of metrics.

Mathematically, First Order statistics can be defined as follows:

I represents the gray levels of an image. Then **P**(I) is defined as the first-order Histogram with the above equation.

$$\mathbf{P}(I) = \frac{\text{\#pixels with gray level I}}{\text{\#pixels in the region}} \tag{3.3}$$

Based on the definition 3.3 the Mean m_1 and the central moments μ_k of image $\mathbf{I}$ are given by:

$$m_1 = E[\mathbf{I}^1] = \sum_{\mathbf{I}=0}^{N_g-1} \mathbf{I}^1\mathbf{P(I)} \tag{3.4}$$

$$\mu_k = E[(\mathbf{I} - E[\mathbf{I})^k] = \sum_{I=0}^{N_g-1} (\mathbf{I} - m_1)^k \mathbf{P}(I), \text{ for k=2,3,4} \tag{3.5}$$

To describe the following features let:

- $\mathbf{X}$ be a set of N_p voxels in the ROI
- $\mathbf{P}$(i) be the first order histogram
- N_g are the discrete intensity levels
- p(i) be the normalized first order histogram and equal to $\mathbf{P}$(i)$/N_p$

1. Energy

Energy is a measure of the magnitude of voxel values in an image. A larger value implies a greater sum of the squares o these values.

$$\textit{energy} = \sum_{i=1}^{N_p} (\mathbf{X}(i) + c)^2 \tag{3.6}$$

In equation 3.6, c is an optional value, which shifts the intensities to prevent negative values in $\mathbf{X}$, so that voxels with very low gray values contribute the least to 3.6.

2. Total Energy

Total Energy is the value of Energy feature scaled by the volume of the voxel in cubic mm.

$$\textit{total energy} = V_{voxel} \sum_{i=1}^{N_p} (\mathbf{X}(i) + c)^2 \tag{3.7}$$

c is the same as in 3.6.

3. Entropy

Entropy averages the amount of information that is needed for encoding the image. In other words, it can specify the uncertainty or randomness of an image.

$$\textit{entropy} = -\sum_{i=1}^{N_g} p(i) \log_2 \big(p(i) + \epsilon\big) \tag{3.8}$$

ϵ is an arbitrarily small positive number ($\approx 2.2 \times 10^{-16}$).

4. Minimum

$$\textit{minimum} = \min(\mathbf{X}) \tag{3.9}$$

5. Maximum

$$\textit{maximum} = \max(\mathbf{X}) \tag{3.10}$$

6. 10th percentile

The 10th percentile of X

7. 90th percentile

The 90th percentile of X

8. Mean

The average gray level intensity within the ROI

$$mean = \frac{1}{N_p} \sum_{i=1}^{N_p} \mathbf{X}(i) \tag{3.11}$$

9. Median

The median gray level intensity within the ROI.

10. Interquartile Range

$$\textit{interquartile range} = \mathbf{P}_{75} - \mathbf{P}_{25} \tag{3.12}$$

Here $\mathbf{P}_{25}$ and $\mathbf{P}_{75}$ are the 25th and 75th percentile of the image array, respectively.

11. Range

The range of gray values in the ROI.

$$range = \max(\mathbf{X}) - \min(\mathbf{X}) \tag{3.13}$$

12. Mean Absolute Deviation (MAD)

Mean Absolute Deviation calculates the average distance of all intensity values from the mean value of the image.

$$MAD = \frac{1}{N_p} \sum_{i=1}^{N_p} |\mathbf{X}(i) - \bar{X}| \tag{3.14}$$

13. Robust Mean Absolute Deviation (rMAD)

Robust Mean Absolute Deviation is similar to Mean Absolute Deviation with the difference that now it is calculated on a subset of the image's array. This subset has values with gray levels in between or equal to the 10th and 90th percentile.

$$rMAD = \frac{1}{N_{10-90}} \sum_{i=1}^{N_{10-90}} |\mathbf{X}_{10-90}(i) - \bar{X}_{10-90}| \tag{3.15}$$

14. Root Mean Squared (RMS)

RMS is the square-root of the mean of all the squared intensity values. It is another measure of the magnitude of the image values.

$$RMS = \sqrt{\frac{1}{N_p} \sum_{i=1}^{N_p} (\mathbf{X}(i) + c)^2} \tag{3.16}$$

c is the same as in 3.6.

15. Skewness

Skewness measures the asymmetry of the distribution of values about the Mean value. Depending on where the tail is elongated and the mass of the distribution is concentrated, this value can be positive or negative.

$$skewness = \frac{\mu_3}{\sigma^3} = \frac{\frac{1}{N_p} \sum_{i=1}^{N_p} (\mathbf{X}(i) - \bar{X})^3}{\left(\sqrt{\frac{1}{N_p} \sum_{i=1}^{N_p} (\mathbf{X}(i) - \bar{X})^2}\right)^3} \tag{3.17}$$

where μ_3 is the 3rd central moment.

16. Kurtosis

Kurtosis is a measure of the 'peakedness' of the distribution of values in the image ROI. In case that most of the distribution lies towards the tail(s) rather than the mean it means that kurtosis is very high. Otherwise, if most of the distribution is concentrated near the Mean, it means that kurtosis is low.

$$kurtosis = \frac{\mu_4}{\sigma^4} = \frac{\frac{1}{N_p} \sum_{i=1}^{N_p} (\mathbf{X}(i) - \bar{X})^4}{\left(\frac{1}{N_p} \sum_{i=1}^{N_p} (\mathbf{X}(i) - \bar{X})^2\right)^2} \tag{3.18}$$

where μ_4 is the 4th central moment.

17. Variance

Variance is the the mean of the squared distances of each intensity value from the Mean value. This is a measure of the spread of the distribution about the mean.

$$\textit{variance} = \frac{1}{N_p}\sum_{i=1}^{N_p}(\mathbf{X}(i) - \bar{X})^2 \tag{3.19}$$

18. Uniformity

Uniformity calculates the sum of the squares of each intensity value. High uniformity implies smaller range of discrete intensity values(high homogeneity).

$$\textit{uniformity} = \sum_{i=1}^{N_g} p(i)^2 \tag{3.20}$$

3..2 3D Shaped-Base features

This subsection refers to features that are derived by the 3D size and shape of the Region of Interest (ROI). 3D shaped-base features have no correlation with the gray level distribution.

To implement this type of feature extraction, a triangle mesh needs to be built. Firstly, vertices are defined as points halfway on an edge between a voxel included into the ROI and one which does not belong into the ROI. Then, these two points are connected to obtain a triangle mesh.

For the generation of this triangle mesh, marching cubes algorithm[37] was enlisted. This method uses a 2x2 cube, which moves through the mask space.

To describe the next equations of the features let:

- ▶ N_v be the number of voxels included in the ROI
- ▶ N_f be the number of faces (triangles) defining the Mesh.
- ▶ V be the volume of the mesh in mm^3
- ▶ A be the surface area of the mesh in mm^2

1.Mesh Volume

$$\begin{gathered} V_i = \frac{Oa_i \cdot (Ob_i \times Oc_i)}{6} \text{ (1)} \\ V = \sum_{i=1}^{N_f} V_i \text{ (2)} \end{gathered} \tag{3.21}$$

2. Voxel Volume

$$V_{voxel} = \sum_{k=1}^{N_v} V_k \tag{3.22}$$

3. Surface Area

$$\begin{gathered} A_i = \frac{1}{2}|\mathrm{a}_i\mathrm{b}_i \times \mathrm{a}_i\mathrm{c}_i| \text{ (1)} \\ A = \sum_{i=1}^{N_f} A_i \text{ (2)} \end{gathered} \tag{3.23}$$

$\mathrm{a}_i\mathrm{b}_i$ are edges of the i_{th} triangle in the mesh, formed by vertices a_i, b_i and c_i

4. Surface Area to Volume ratio

$$\textit{surface to volume ratio} = \frac{A}{V} \tag{3.24}$$

5. Sphericity

Sphericity measures how round a shape of the tumor region is relatively to a sphere.

$$\textit{sphericity} = \frac{\sqrt[3]{36\pi V^2}}{A} \tag{3.25}$$

The value range is $0 < \textit{sphericity} \leq 1$. A value that tends to 1 means a perfect sphere.

6. Spherical Disproportion

Spherical Disproportion is the inverse of Sphericity. In other words, it defines the ratio of the surface area of the tumor region to the surface area of a sphere with the same volume as the tumor region.

$$\textit{spherical disproportion} = \frac{A}{4\pi R^2} = \frac{A}{\sqrt[3]{36\pi V^2}} \tag{3.26}$$

Where R is the radius of a sphere with the same volume as the tumor and equal to $\sqrt[3]{\frac{3V}{4\pi}}$.

7. Maximum 3D diameter (Feret Diameter)

Defines the pair with the largest Euclidean distance between tumor surface mesh vertices.

8. Maximum 2D diameter - Slice

Defines the pair with the largest Euclidean distance in row-column plane.

9. Maximum 2D diameter - Column

Defines the pair with the largest Euclidean distance in row-slice plane.

10. Maximum 2D diameter - Row

Defines the pair with the largest Euclidean distance in column-slice plane.

11. Major Axis Length

Yields the largest axis length of the ROI-enclosing ellipsoid and is calculated using the largest principal component λ_{major}

$$\textit{major axis} = 4\sqrt{\lambda_{major}} \tag{3.27}$$

12. Minor Axis Length

Yields the second-largest axis length of the ROI-enclosing ellipsoid and is calculated using the largest principal component λ_{minor}

$$\textit{minor axis} = 4\sqrt{\lambda_{minor}} \tag{3.28}$$

13. Least Axis Length

Yields the smallest axis length of the ROI-enclosing ellipsoid and is calculated using the largest principal component λ_{least}. Least Axis Length in is set to 0, when the segmentation is 2D.

$$\textit{least axis} = 4\sqrt{\lambda_{least}} \tag{3.29}$$

14. Elongation

The relationship between the two largest principal components in the ROI shape.

$$\textit{elongation} = \sqrt{\frac{\lambda_{minor}}{\lambda_{major}}} \tag{3.30}$$

Here, λ_{major} and λ_{minor} are the lengths of the largest and second largest principal component axes.

15. Flatness

The relationship between the largest and smallest principal components in the ROI shape.

$$flatness = \sqrt{\frac{\lambda_{least}}{\lambda_{major}}} \tag{3.31}$$

λ_{major} represents the largest principal component. While λ_{least} represents the smallest principal component axes.

3..3 Gray Level Co-occurence Matrix (GLCM)

The first order statistics features that were extracted as stated in the previous subsection provide information related to the gray-level distribution of the image. However there is no information in relative positions of the various gray levels within the image. That's why Second-Order features are extracted using the co-occurence matrix that will be defined in the next paragraphs.

Given a gray-level image **I**, co-occurence matrix computes how often pairs of pixels with a specific value and offset occur in the image.

- The offset, **(Dx, Dy)**, is a position operator that can be applied to any pixel in the image(ignoring edge effects): for example, **(3, 1)** could mean "three down, one right".
- An image with **c** different pixel values will produce a **c x c** co-occurence matrix, for the given offset.
- The $(i,j)^{th}$ value of the co-occurence matrix gives the number of times in the image that the i^{th} and j^{th} pixel values occure in the relation given by the offset.

For an image with **p** different pixel values, the cxc co-occurence matrix **P** is defined over an $n \times m$ image **I**, parameterized by an offset **(Dx,Dy)** as.

$$P_{Dx,Dy}(i,j) = \sum_{x=1}^{n}\sum_{y=1}^{m}\begin{cases}1, & \text{if I(x,y) = i and I(x + Dx, y + Dy) = j).}\\ 0, & \text{otherwise.}\end{cases} \tag{3.32}$$

Co-occurence matrices can also be parameterized by an angle θ.

As a two dimensional example, let the following matrix **I** represent a $5x5$ image, having 5 discrete grey levels [53]:

$$\mathbf{I} = \begin{bmatrix} 1 & 2 & 5 & 2 & 3 \\ 3 & 2 & 1 & 3 & 1 \\ 1 & 3 & 5 & 5 & 2 \\ 1 & 1 & 1 & 1 & 2 \\ 1 & 2 & 4 & 3 & 5 \end{bmatrix} \tag{3.33}$$

For distance $\delta = 1$ and angle at zero degrees, the following symmetrical GLCM is obtained:

$$\mathbf{P} = \begin{bmatrix} 6 & 4 & 3 & 0 & 0 \\ 4 & 0 & 2 & 1 & 3 \\ 3 & 2 & 0 & 1 & 2 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 3 & 2 & 0 & 2 \end{bmatrix} \tag{3.34}$$

Let:

- ϵ be an arbitrarily small positive number ($\approx 2.2 \times 10^{-16}$)
- **P**(i,j) be the co-occurence matrix for an arbitrary δ and θ
- $p(i,j)$ be the normalized co-occurence matrix and equal to $\frac{\mathbf{P}(i,j)}{\sum \mathbf{P}(i,j)}$
- N_g be the number of discrete intensity levels in the image
- $p_x(i) = \sum_{j=1}^{N_g} P(i,j)$ be the marginal row probabilities
- $p_y(j) = \sum_{i=1}^{N_g} P(i,j)$ be the marginal column probabilities
- μ_x be the mean gray level intensity of p_x and defines as $\mu_x = \sum\limits_{i=1}^{N_g} p_x(i)i$
- μ_y be the mean gray level intensity of p_y and defines as $\mu_y = \sum\limits_{i=1}^{N_g} p_y(i)i$

- σ_x be the standard deviation of p_x.
- σ_y be the standard deviation of p_y.
- $p_{x+y}(k) = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i,j)$, where $i+j=k$, and $k = 2, 3, \dots, 2N_g$
- $p_{x-y}(k) = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i,j)$, where $|i-j| = k$, and $k = 0, 1, \dots, N_g - 1$
- $HX = -\sum_{i=1}^{N_g} p_x(i) \log_2 \big(p_x(i) + \epsilon\big)$ be the entropy of p_x
- $HY = -\sum_{j=1}^{N_g} p_y(j) \log_2 \big(p_y(j) + \epsilon\big)$ be the entropy of p_y
- $HXY = -\sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i,j) \log_2 \big(p(i,j) + \epsilon\big)$
- $HXY1 = -\sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i,j) \log_2 \big(p_x(i) p_y(j) + \epsilon\big)$
- $HXY2 = -\sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p_x(i) p_y(j) \log_2 \big(p_x(i) p_y(j) + \epsilon\big)$

1. Autocorrelation

Calculates the magnitude of the fineness and coarseness of texture.

$$\textit{autocorrelation} = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i,j)ij \tag{3.35}$$

2. Joint Average

Is the mean gray level intensity of i distribution.

$$\textit{joint average} = \mu_x = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i,j)i \tag{3.36}$$

3. Cluster Prominence

Measures the skewness and asymmetry of the GLCM. If cluster prominence is high, it implies high asymmetry about the mean. In case, the feature has a low value it implies less variation of the mean.

$$\textit{cluster prominence} = \sum_{i=1}^{N_g}\sum_{j=1}^{N_g}\left(i+j-\mu_x-\mu_y\right)^4 p(i,j) \tag{3.37}$$

4. Cluster Shade

Measures the skewness and uniformity of the GLCM. A higher value indicates greater asymmetry about the mean.

$$\textit{cluster shade} = \sum_{i=1}^{N_g}\sum_{j=1}^{N_g}\left(i+j-\mu_x-\mu_y\right)^3 p(i,j) \tag{3.38}$$

5. Cluster Tendency

Measures the groupings of voxels with similar gray-level values.

$$\textit{cluster tendency} = \sum_{i=1}^{N_g}\sum_{j=1}^{N_g}\left(i+j-\mu_x-\mu_y\right)^2 p(i,j) \tag{3.39}$$

6. Contrast

Measures the local intensity variation, favoring values away from the diagonal $(i=j)$. A high value implies a greater disparity in intensity values near adjacent voxels.

$$\textit{contrast} = \sum_{i=1}^{N_g}\sum_{j=1}^{N_g}(i-j)^2 p(i,j) \tag{3.40}$$

7. Correlation

Shows the linear dependency of gray level values to their respective GLCM voxels. It ranges through 0(uncorrelated) and 1 (correlated).

$$\textit{correlation} = \frac{\sum_{i=1}^{N_g}\sum_{j=1}^{N_g} p(i,j)ij-\mu_x\mu_y}{\sigma_x(i)\sigma_y(j)} \tag{3.41}$$

8. Difference Average

Measures the relationship between occurrences of pairs with similar intensity values and occurrences of pairs with differing intensity values.

$$\textit{difference average} = \sum_{k=0}^{N_g-1} k p_{x-y}(k) \tag{3.42}$$

9. Difference Entropy

Measures the randomness/variability in adjacent intensity value differences.

$$\textit{difference entropy} = \sum_{k=0}^{N_g-1} p_{x-y}(k) \log_2 \big(p_{x-y}(k) + \epsilon\big) \tag{3.43}$$

10. Difference Variance

Measures the heterogeneity that places higher weights on differing intensity level pairs that deviate more from the mean.

$$\textit{difference variance} = \sum_{k=0}^{N_g-1} (k - DA)^2 p_{x-y}(k) \tag{3.44}$$

11. Joint Energy

Measures the homogeneous patterns in the image. A high Energy value indicates more instances of intensity value pairs in the image that neighbor each other at high frequencies.

$$\textit{joint energy} = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} \big(p(i,j)\big)^2 \tag{3.45}$$

12. Joint Entropy

Measures the randomness/variability in neighborhood intensity values.

$$\textit{joint entropy} = -\sum_{i=1}^{N_g}\sum_{j=1}^{N_g} p(i,j)\log_2\big(p(i,j)+\epsilon\big) \tag{3.46}$$

13. Informational Measure of Correlation (IMC) 1

$$\textit{IMC 1} = \frac{HXY - HXY1}{\max\{HX, HY\}} \tag{3.47}$$

Assesses the correlation between the probability distributions of i and j (quantifying the complexity of the texture), using mutual information I(x, y):

$$\begin{aligned} I(i,j) &= \sum_{i=1}^{N_g}\sum_{j=1}^{N_g} p(i,j)\log_2\big(\frac{p(i,j)}{p_x(i)p_y(j)}\big) \\ &= \sum_{i=1}^{N_g}\sum_{j=1}^{N_g} p(i,j)\big(\log_2(p(i,j)) - \log_2(p_x(i)p_y(j))\big) \\ &= \sum_{i=1}^{N_g}\sum_{j=1}^{N_g} p(i,j)\log_2\big(p(i,j)\big) - \sum_{i=1}^{N_g}\sum_{j=1}^{N_g} p(i,j)\log_2\big(p_x(i)p_y(j)\big) \\ &= -HXY + HXY1 \end{aligned} \tag{3.48}$$

14. Informational Measure of Correlation (IMC) 2

Assesses the correlation between the probability distributions of i and j (quantifying the complexity of the texture).

$$\textit{IMC 2} = \sqrt{1 - e^{-2(HXY2 - HXY)}} \tag{3.49}$$

15. Inverse Difference Moment (IDM)

Is a measurement of the local homogeneity in an image. IDM weights are the inverse of the Contrast weights (decreasing exponentially from the diagonal i=j in the GLCM).

$$IDM = \sum_{k=0}^{N_g-1} \frac{p_{x-y}(k)}{1+k^2} \tag{3.50}$$

16. Maximal Correlation Coefficient (MCC)

Measures the complexity of the texture and $0 \leq MCC \leq 1$.

$$\begin{gathered} MCC = \sqrt{\text{second largest eigenvalue of Q}} \\ Q(i,j) = \sum_{k=0}^{N_g} \frac{p(i,k)p(j,k)}{p_x(i)p_y(k)} \end{gathered} \tag{3.51}$$

17. Inverse Difference Moment Normalized (IDMN)

Measures the local homogeneity of an image. IDMN normalizes the square of the difference between neighboring intensity values by dividing over the square of the total number of discrete intensity values.

$$IDMN = \sum_{k=0}^{N_g-1} \frac{p_{x-y}(k)}{1+\left(\frac{k^2}{N_g^2}\right)} \tag{3.52}$$

18. Inverse Difference (ID)

Measures the local homogeneity of an image.

$$ID = \sum_{k=0}^{N_g-1} \frac{p_{x-y}(k)}{1+k} \tag{3.53}$$

19. Inverse Difference Normalized (IDN)

Measures the local homogeneity of an image. This feature normalizes the difference between the neighboring intensity values by dividing over the total number of discrete intensity values.

$$IDN = \sum_{k=0}^{N_g-1} \frac{p_{x-y}(k)}{1 + \left(\frac{k}{N_g}\right)} \tag{3.54}$$

20. Inverse Variance

$$\textit{inverse variance} = \sum_{k=1}^{N_g-1} \frac{p_{x-y}(k)}{k^2} \tag{3.55}$$

21. Sum Average

Sum Average measures the relationship between occurrences of pairs with lower intensity values and occurrences of pairs with higher intensity values.

$$\textit{sum average} = \sum_{k=2}^{2N_g} p_{x+y}(k)k \tag{3.56}$$

22. Sum entropy

Sum Entropy is a sum of neighborhood intensity value differences.

$$\textit{sum entropy} = \sum_{k=2}^{2N_g} p_{x+y}(k) \log_2 \left(p_{x+y}(k) + \epsilon\right) \tag{3.57}$$

23. Sum of Squares

Sum of Squares or Variance is a measure in the distribution of neighboring intensity level pairs about the mean intensity level in the GLCM.

$$\textit{sum squares} = \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} (i - \mu_x)^2 p(i,j) \tag{3.58}$$

24. Maximum Probability

Maximum Probability is occurrences of the most predominant pair of neighboring intensity values.

$$\textit{maximum probability} = \max\big(p(i,j)\big) \tag{3.59}$$

3..4 Gray Level Run Length Matrix (GLRLM)

A gray level run is a set of consecutive, collinear picture points having the same gray level value [25]. Specifically, The length of the run is the number of picture points in the run. The matrix element **P**(i,j|θ) specifies the number of times that the picture contains a run of length j, in the given direction, consisting of points having gray level i.

In other words GLRLM counts the number of pixel segments having the same intensity in a given direction. These results are represented by a matrix and the direction could be 0°, 45°, 90°, 135°and 180°. So the matrix element **P**(i,j|θ) can be defined as the number of segments of length i and gray level j.

Let:

- N_g be the number of discreet intensity values in the image
- N_r be the number of discreet run lengths in the image
- N_p be the number of voxels in the image
- $N_r(\theta)$ be the number of runs in the image along angle θ, which is equal to $\sum_{i=1}^{N_g}\sum_{j=1}^{N_r}\mathbf{P}(i,j|\theta)$ with $1 \leq N_r(\theta) \leq N_p$
- $\mathbf{P}(i,j|\theta)$ be the run length matrix for an arbitrary direction θ
- $p(i,j|\theta)$ be the normalized run length matrix, defined as $p(i,j|\theta) = \frac{\mathbf{P}(i,j|\theta)}{N_r(\theta)}$

1. Short Run Emphasis (SRE)

Is a measure of the distribution of short run lengths, with a greater value indicative of shorter run lengths and more fine textural textures.

$$SRE = \frac{\sum_{i=1}^{N_g} \sum_{j=1}^{N_r} \frac{\mathbf{P}(i,j|\theta)}{j^2}}{N_r(\theta)} \tag{3.60}$$

2. Long Run Emphasis (LRE)

Is a measure of the distribution of long run lengths, with a greater value indicative of longer run lengths and more coarse structural textures.

$$LRE = \frac{\sum_{i=1}^{N_g} \sum_{j=1}^{N_r} \mathbf{P}(i,j|\theta) j^2}{N_r(\theta)} \tag{3.61}$$

3. Gray Level Non-Uniformity (GLN)

Measures the similarity of gray-level intensity values in the image, where a lower GLN value correlates with a greater similarity in intensity values.

$$GLN = \frac{\sum_{i=1}^{N_g} \left(\sum_{j=1}^{N_r} \mathbf{P}(i,j|\theta) \right)^2}{N_r(\theta)} \tag{3.62}$$

4. Gray Level Non-Uniformity Normalized (GLNN)

Represents how similar are the gray-level intensity values in an image.

$$GLNN = \frac{\sum_{i=1}^{N_g} \left(\sum_{j=1}^{N_r} \mathbf{P}(i,j|\theta) \right)^2}{N_r(\theta)^2} \tag{3.63}$$

5. Run Length Non-Uniformity (RLN)

Similarity of run lengths throughout the image are calculated using RLN. A low RLN value means more homogeneity among image's run lengths.

$$RLN = \frac{\sum_{j=1}^{N_r} \left(\sum_{i=1}^{N_g} \mathbf{P}(i,j|\theta) \right)^2}{N_r(\theta)} \tag{3.64}$$

6. Run Length Non-Uniformity Normalized (RLNN)

Represents how similar the run lengths throughout the image are. RLNN with a low value implies more homogeneity among run lengths in the image.

$$RLNN = \frac{\sum_{j=1}^{N_r} \left(\sum_{i=1}^{N_g} \mathbf{P}(i,j|\theta) \right)^2}{N_r(\theta)^2} \tag{3.65}$$

7. Run Percentage (RP)

Measures the coarseness of the texture by taking the ratio of number of runs and number of voxels in the ROI.

$$RP = \frac{N_r(\theta)}{N_p} \tag{3.66}$$

8. Gray Level Variance (GLV)

Measures the variance in gray level intensity for the runs.

$$GLV = \sum_{i=1}^{N_g} \sum_{j=1}^{N_r} p(i,j|\theta)(i-\mu)^2 \tag{3.67}$$

with

$$\mu = \sum_{i=1}^{N_g} \sum_{j=1}^{N_r} p(i,j|\theta)i \tag{3.68}$$

9. Run Variance (RV)

Is a measure of the variance in runs for the run lengths.

$$RV = \sum_{i=1}^{N_g} \sum_{j=1}^{N_r} p(i,j|\theta)(j-\mu)^2 \tag{3.69}$$

with μ defined as in 3.68.

10. Run Entropy (RE)

A measurement for uncertainty or randomness of run lengths and gray levels distribution. High value means more heterogeneity in the image's texture.

$$RE = -\sum_{i=1}^{N_g} \sum_{j=1}^{N_r} p(i,j|\theta)\log_2(p(i,j|\theta)+\epsilon) \tag{3.70}$$

with ϵ defined as in 3.8

11. Low Gray Level Run Emphasis (LGLRE)

This feature calculates low gray-level's distribution. High LGLRE value implies greater concentration of small gray-level values in the image.

$$LGLRE = \frac{\sum_{i=1}^{N_g} \sum_{j=1}^{N_r} \frac{\mathbf{P}(i,j|\theta)}{i^2}}{N_r(\theta)} \tag{3.71}$$

12. High Gray Level Run Emphasis (HGLRE)

Measures the distribution of the higher gray-level values. HGLRE with high value is correlated with a greater concentration of high gray-level values in the image.

$$HGLRE = \frac{\sum_{i=1}^{N_g} \sum_{j=1}^{N_r} \mathbf{P}(i,j|\theta) i^2}{N_r(\theta)} \tag{3.72}$$

13. Short Run Low Gray Level Emphasis (SRLGLE)

The joint distribution of shorter run lengths with **lower** gray-level values is calculated using SRLGLE.

$$SRLGLE = \frac{\sum_{i=1}^{N_g} \sum_{j=1}^{N_r} \frac{\mathbf{P}(i,j|\theta)}{i^2 j^2}}{N_r(\theta)} \tag{3.73}$$

14. Short Run High Gray Level Emphasis (SRHGLE)

The joint distribution of shorter run lengths with **higher** gray-level values is calculated using SRHGLE.

$$SRHGLE = \frac{\sum_{i=1}^{N_g} \sum_{j=1}^{N_r} \frac{\mathbf{P}(i,j|\theta) i^2}{j^2}}{N_r(\theta)} \tag{3.74}$$

15. Long Run Low Gray Level Emphasis (LRLGLE)

The joint distribution of longer run lengths with **lower** gray-level values is calculated using LRLGLE.

$$LRLGLE = \frac{\sum_{i=1}^{N_g} \sum_{j=1}^{N_r} \frac{\mathbf{P}(i,j|\theta) j^2}{i^2}}{N_r(\theta)} \tag{3.75}$$

16. Long Run High Gray Level Emphasis (LRHGLE)

The joint distribution of longer run lengths with **higher** gray-level values is calculated using LRLGLE.

$$LRHGLE = \frac{\sum_{i=1}^{N_g} \sum_{j=1}^{N_r} \mathbf{P}(i,j|\theta) i^2 j^2}{N_r(\theta)} \tag{3.76}$$

3..5 Gray Level Size Zone Matrix (GLSZM)

Because a homogeneous texture is composed of large areas with the same intensity rather than small areas in any direction, GLSZM tries to take this into account. GLSZM quantifies the size of each area with same intensity level pixels. The $(i, j)^{th}$ element in GLSZM equals to the number of areas of size i and of gray level j. At this point, it should be noted that this matrix calculates ROI for all directions without rotating.

Let:

- N_g be the number of discreet intensity values in the image
- N_s be the number of discreet zone sizes in the image
- N_p be the number of voxels in the image
- N_z be the number of zones in the ROI,
- **P**(i,j) be the size zone matrix
- $p(i, j)$ be the normalized size zone matrix

1. Small Area Emphasis (SAE)

A measure of small size zones distribution. A higher value implies smaller size zones and finer textures.

$$SAE = \frac{\sum_{i=1}^{N_g} \sum_{j=1}^{N_s} \frac{\mathbf{P}(i,j)}{j^2}}{N_z} \tag{3.77}$$

2. Large Area Emphasis (LAE)

A measure of large area size zones distribution. A higher value implies larger size zones and coarser textures.

$$LAE = \frac{\sum_{i=1}^{N_g} \sum_{j=1}^{N_s} \mathbf{P}(i,j) j^2}{N_z} \tag{3.78}$$

3. Gray Level Non-Uniformity (GLN)

A measure of the variability of gray-level intensity values in the image. Low GLN value means more homogeneity in intensity voxel values.

$$GLN = \frac{\sum_{i=1}^{N_g} \left(\sum_{j=1}^{N_s} \mathbf{P}(i,j) \right)^2}{N_z} \tag{3.79}$$

4. Gray Level Non-Uniformity Normalized (GLNN)

The variability of gray-level intensity values of an image is calculated by GLNN feature. A low GLNN value means a greater similarity in intensity values between pixels.

$$GLNN = \frac{\sum_{i=1}^{N_g} \left(\sum_{j=1}^{N_s} \mathbf{P}(i,j) \right)^2}{N_z^2} \tag{3.80}$$

5. Size-Zone Non-Uniformity (SZN)

The variability of size zone volumes is measured by SZN. A low SZN value implies more homogeneity in size zone volumes.

$$SZN = \frac{\sum_{j=1}^{N_s} \left(\sum_{i=1}^{N_g} \mathbf{P}(i,j) \right)^2}{N_z} \tag{3.81}$$

6. Size-Zone Non-Uniformity Normalized (SZNN)

The variability of size zones volumes is represented by SZNN feature. A low SZNN value implies homogeneity between zone size volumes in the image.

$$SZNN = \frac{\sum_{j=1}^{N_s} \left(\sum_{i=1}^{N_g} \mathbf{P}(i,j) \right)^2}{N_z^2} \tag{3.82}$$

7. Zone Percentage (ZP)

A measure of the coarseness of the texture by taking the ratio of number of zones and number of voxels in the ROI.

$$ZP = \frac{N_z}{N_p} \tag{3.83}$$

8. Gray Level Variance (GLV)

A measure of the variance in gray level intensities for the zones.

$$GLV = \sum_{i=1}^{N_g} \sum_{j=1}^{N_s} p(i,j)(i-\mu)^2 \tag{3.84}$$

9. Zone Variance (ZV)

A measure of the variance in zone size volumes for the zones.

$$ZV = \sum_{i=1}^{N_g} \sum_{j=1}^{N_s} p(i,j)(j-\mu)^2 \tag{3.85}$$

10. Zone Entropy (ZE)

A measure of the randomness in the distribution of zone sizes and gray levels.

$$ZE = -\sum_{i=1}^{N_g} \sum_{j=1}^{N_s} p(i,j) \log_2(p(i,j)+\epsilon) \tag{3.86}$$

11. Low Gray Level Zone Emphasis (LGLZE)

A measure of the distribution of lower gray-level size zones.

$$LGLZE = \frac{\sum_{i=1}^{N_g} \sum_{j=1}^{N_s} \frac{\mathbf{P}(i,j)}{i^2}}{N_z} \tag{3.87}$$

12. High Gray Level Zone Emphasis (HGLZE)

A measure of the distribution of the higher gray-level values.

$$HGLZE = \frac{\sum_{i=1}^{N_g} \sum_{j=1}^{N_s} \mathbf{P}(i,j) i^2}{N_z} \tag{3.88}$$

13. Small Area Low Gray Level Emphasis (SALGLE)

A measure of the proportion in the image of the joint distribution of smaller size zones with lower gray-level values.

$$SALGLE = \frac{\sum_{i=1}^{N_g} \sum_{j=1}^{N_s} \frac{\mathbf{P}(i,j)}{i^2 j^2}}{N_z} \tag{3.89}$$

14. Small Area High Gray Level Emphasis (SAHGLE)

A measure of the proportion in the image of the joint distribution of smaller size zones with higher gray-level values.

$$SAHGLE = \frac{\sum_{i=1}^{N_g} \sum_{j=1}^{N_s} \frac{\mathbf{P}(i,j)i^2}{j^2}}{N_z} \tag{3.90}$$

15. Large Area Low Gray Level Emphasis (LALGLE)

For measuring the proportion in the image of larger size zones distribution with lower gray-level values, LALGLE feature is used.

$$LALGLE = \frac{\sum_{i=1}^{N_g} \sum_{j=1}^{N_s} \frac{\mathbf{P}(i,j)j^2}{i^2}}{N_z} \tag{3.91}$$

16. Large Area High Gray Level Emphasis (LAHGLE)

For measuring the proportion in the image of larger size zones distribution with higher gray-level values, LAHGLE feature is used.

$$LAHGLE = \frac{\sum_{i=1}^{N_g} \sum_{j=1}^{N_s} \mathbf{P}(i,j)i^2j^2}{N_z} \tag{3.92}$$

3..6 Neighbouring Gray Tone Difference Matrix (NGTDM)

Let $f(k,l)$ be the gray tone of any pixel at (k,l) having gray tone value i. Then we find the average gray-tone over a neighborhood centered at, but excluding (k,l) as follows[10]:

$$\begin{aligned} \bar{A}_i &= \bar{A}(k,l) \\ &= \frac{1}{W-1} \sum_{m=-\delta}^{\delta} \sum_{n=-\delta}^{\delta} f(k+m, l+n), \\ &\text{where } (m,n) \neq (0,0) \text{ and } w = (2d+1)^2 \end{aligned} \tag{3.93}$$

where d specifies the neighborhood size.

Then the i_{th} entry in the NGTDM is the sum of absolute differences for gray level i.

$$s(i) = \begin{cases} \sum |i - \bar{A}_i|, for \quad i \in N_i, if N_i \neq 0 \\ 0, \text{otherwise} \end{cases} \tag{3.94}$$

where N_i is the set of all pixels having gray tone i.

Let:

- n_i be the number of voxels in f(k,l) with gray level i
- $N_{v,p}$ be the total number of voxels in f(k,l)
- p_i be the gray level probability and equal to n_i/N_v
- N_g be the number of discreet gray levels
- $N_{g,p}$ be the number of gray levels where $p_i \neq 0$

1. Coarseness

The average difference between center voxel and its neighbourhood is measured by Coarseness. Additionally, it is an indication of change's spatial rate. A higher value indicates a lower spatial change rate and a locally more uniform texture.

$$Coarseness = \frac{1}{\sum_{i=1}^{N_g} p_i s_i} \tag{3.95}$$

2. Contrast

The spatial intensity change is calculated using the feature Contrast. Contrast is also dependent on the total gray level dynamic range. When dynamic range and spatial change have high values contrast follows with a high value too.

$$Contrast = \left(\frac{1}{N_{g,p}(N_{g,p}-1)} \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p_i p_j (i-j)^2\right) \left(\frac{1}{N_{v,p}} \sum_{i=1}^{N_g} s_i\right), \text{ where } p_i \neq 0, p_j \neq 0 \tag{3.96}$$

3. Busyness

A measure of the change from a pixel to its neighbour.

$$Busyness = \frac{\sum_{i=1}^{N_g} p_i s_i}{\sum_{i=1}^{N_g} \sum_{j=1}^{N_g} |i p_i - j p_j|}, \text{ where } p_i \neq 0, p_j \neq 0 \tag{3.97}$$

4. Complexity

An image is considered complex when there are many primitive components in the image.

$$Complexity = \frac{1}{N_{v,p}} \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} |i-j| \frac{p_i s_i + p_j s_j}{p_i + p_j}, \text{ where } p_i \neq 0, p_j \neq 0 \tag{3.98}$$

5. Strength

A measure of the primitives in an image.

$$Strength = \frac{\sum_{i=1}^{N_g} \sum_{j=1}^{N_g} (p_i + p_j)(i-j)^2}{\sum_{i=1}^{N_g} s_i}, \text{ where } p_i \neq 0, p_j \neq 0 \tag{3.99}$$

3..7 Gray Level Dependence Matrix (GLDM)

GLDM as the name suggests computes gray level dependencies in an image array. Suppose a GLDM **P**(i,j) and the element $(i,j)^{th}$. This value of the matrix counts the occurrences of a voxel with gray level i in the image, while having j dependent voxels in its neighbourhood.

Mathematically, the GLDM of an image **I** can be defined as:

GLDM(q,r) = # {(i,j) | **I**(i,j) = q and

#[(q,r) | dist((i,j),(k,s)) $\leq d$ and

$|\mathbf{I}(q,r) - \mathbf{I}(k,s)| \leq a]r\}$ (3.100)

Where (i,j) and (q,r) belong to G = Z_r x Z_c. The sets $Z_r = (1, ...\#rows)$ and $Z_s = (1, ...\#columns)$ are horizontal and vertical spatial domains.

K is the set of gray level values that can occur in image **I**.

Let:

- be the number of discreet intensity values in the image
- N_g
- N_d be the number of discreet dependency sizes in the image
- $N_z = \sum_{i=1}^{N_g} \sum_{j=1}^{N_d} \mathbf{P}(i,j)$ which shows the number of dependency zones of an image.
- **P(i,j)** be the dependence matrix
- p(i,j) be the normalized dependence matrix

1. Small Dependence Emphasis (SDE)

Small dependencies distribution is measured by SDE. High SDE implies less homogeneous and more independence.

$$SDE = \frac{\sum_{i=1}^{N_g} \sum_{j=1}^{N_d} \frac{\mathbf{P}(i,j)}{i^2}}{N_z} \tag{3.101}$$

2. Large Dependence Emphasis (LDE)

Large dependencies in the distribution are measured by LDE. A higher LDE value means larger dependence and more homogeneous textures.

$$LDE = \frac{\sum_{i=1}^{N_g} \sum_{j=1}^{N_d} \mathbf{P}(i,j) j^2}{N_z} \tag{3.102}$$

3. Gray Level Non-Uniformity (GLN)

The gray-level intensity values similarity is calculated by GLN. A lower GLN value is correlated with a greater similarity in intensity values in the image.

$$GLN = \frac{\sum_{i=1}^{N_g} \left(\sum_{j=1}^{N_d} \mathbf{P}(i,j) \right)^2}{N_z} \tag{3.103}$$

4. Dependence Non-Uniformity (DN)

Similarity of image dependence is measured by DN. A lower value implies more homogeneity between dependencies of the image.

$$DN = \frac{\sum_{j=1}^{N_d} \left(\sum_{i=1}^{N_g} \mathbf{P}(i,j) \right)^2}{N_z} \tag{3.104}$$

5. Dependence Non-Uniformity Normalized (DNN)

A normalized version of DN.

$$DNN = \frac{\sum_{j=1}^{N_d}\left(\sum_{i=1}^{N_g}\mathbf{P}(i,j)\right)^2}{N_z^2} \tag{3.105}$$

6. Gray Level Variance (GLV)

Measures the variance in grey level in the image.

$$GLV = \sum_{i=1}^{N_g}\sum_{j=1}^{N_d} p(i,j)(i-\mu)^2, \text{ where} \mu = \sum_{i=1}^{N_g}\sum_{j=1}^{N_d} ip(i,j) \tag{3.106}$$

7. Dependence Variance (DV)

Measures the variance in dependence size in the image.

$$DV = \sum_{i=1}^{N_g}\sum_{j=1}^{N_d} p(i,j)(j-\mu)^2, \text{ where} \mu = \sum_{i=1}^{N_g}\sum_{j=1}^{N_d} jp(i,j) \tag{3.107}$$

8. Dependence Entropy (DE)

$$DependenceEntropy = -\sum_{i=1}^{N_g}\sum_{j=1}^{N_d} p(i,j)\log_2(p(i,j)+\epsilon) \tag{3.108}$$

9. Low Gray Level Emphasis (LGLE)

This feature is used to measure lower gray-level values distribution. A higher LGLE value means a greater concentration of low gray-level values in the image.

$$LGLE = \frac{\sum_{i=1}^{N_g}\sum_{j=1}^{N_d}\frac{\mathbf{P}(i,j)}{i^2}}{N_z} \tag{3.109}$$

10. High Gray Level Emphasis (HGLE)

This feature is used to measure higher gray-level values distribution. A higher HGLE value indicates a greater concentration of high gray-level values in the image.

$$HGLE = \frac{\sum_{i=1}^{N_g} \sum_{j=1}^{N_d} \mathbf{P}(i,j) i^2}{N_z} \tag{3.110}$$

11. Small Dependence Low Gray Level Emphasis (SDLGLE)

The lower gray-level values small dependence joint distribution is measured by SDLGLE.

$$SDLGLE = \frac{\sum_{i=1}^{N_g} \sum_{j=1}^{N_d} \frac{\mathbf{P}(i,j)}{i^2 j^2}}{N_z} \tag{3.111}$$

12. Small Dependence High Gray Level Emphasis (SDHGLE)

The higher gray-level values small dependence joint distribution is measured by SDHGLE.

13. Large Dependence Low Gray Level Emphasis (LDLGLE)

The lower gray-level values large dependence joint distribution is measured by SDLGLE.

$$LDLGLE = \frac{\sum_{i=1}^{N_g} \sum_{j=1}^{N_d} \frac{\mathbf{P}(i,j) j^2}{i^2}}{N_z} \tag{3.112}$$

14. Large Dependence High Gray Level Emphasis (LDHGLE)

The higher gray-level values large dependence joint distribution is measured by SDLGLE.

$$LDHGLE = \frac{\sum_{i=1}^{N_g} \sum_{j=1}^{N_d} \mathbf{P}(i,j) i^2 j^2}{N_z} \tag{3.113}$$

Chapter 4

Machine Learning Theory

A machine learning algorithm is an algorithm that is able to learn data. At this point "learn" term should be defined. Tom Mitchell gives a very condensed definition [42]:

"A computer program is said to learn from experience **E** with respect to some class of tasks **T** and performance measure **P**, if its performance at tasks in **T**, as measured by **P**, improves with experience **E**."

Task - T

The Task **T** can be classification, regression, machine translation or anomaly detection and many more. In our case we have a classification problem.

In this type of problem, the computer needs to classify which of k categories an input sample belongs to. The learning algorithm of classification has as a goal to generate a function as follows:

$f : \mathbb{R}^n \rightarrow 1, ..., k$

The above function of the model assigns an input described by a vector $\mathbf{x}$ to a category identified by a numeric code y:

$y = f(\mathbf{x})$

Another way to approach our problem would be something similar to Anomaly Detection. That is because there are much more negative cases than the positive ones and for some cases there is uncertainty if a patient ails from covid-19. Anomaly detection is the type of task flags some samples as unusual or atypical. In other words it finds patterns in data that do not conform to expected behavior.

Performance Measure - P

Performance Measure is required to evaluate the machine learning model. Generally, **P** is related to the task **T**. In our case in which classification is the task, accuracy is common used for the model performance. However, as it was stated previously, because data is highly imbalanced, other metrics are preferred.

Experience - E

Experience is the the way that a machine learning model learns. There are plenty categories of experience such as Supervised, Unsupervised and reinforcement learning. The first two methods will be described analytically in the later sections.

1. Supervised

Supervised learning algorithms has the following characteristic. The dataset contains a certain number of features but each sample is related to a **label** or **target**. The term supervised learning has origins because a target **y** is provided by an instructor who somehow shows the machine learning model what to learn.

1..1 Support Vector Machines

Support Vector Machines [21] basic idea is to find a hyperplane which separates the d-dimensional data perfectly into its two classes. By perfectly, it is meant that the two classes have the maximum margin from the optimal hyperplane. Most of the time, the data are not linear separable. This is where the power of SVM lies. SVMs can introduce kernel methods to map features to a higher dimension where the data is separable. Typically, this mapping to the higher dimension space would cost computationally. But fortunately, SVMs do not need to calculate this higher dimension and only the formula of the dot product is needed. Hence, instead of a complexity of $O(n^2)$ we have a complexity of $O(n)$.

Linear Separation of a feature space

A hyper plane in an n-D feature space can be represented by the following equation:

$$f(\mathbf{x}) = \mathbf{x}^T\mathbf{w} + b = \sum_{i=1}^{n} x_i w_i + b = 0 \tag{4.1}$$

Dividing by $||\mathbf{w}||$, we get

$$\frac{\mathbf{x}^T\mathbf{w}}{||\mathbf{w}||} = P_{\mathbf{w}}(\mathbf{x}) = -\frac{b}{||\mathbf{w}||} \tag{4.2}$$

Then, let:

- ▶ $P_{\mathbf{w}}(\mathbf{x})$ the projection of any sample $\mathbf{x}$.
- ▶ $\mathbf{w}$ be the normal direction of the plane.
- ▶ $\frac{||b||}{||\mathbf{w}||}$ be the distance from the origin to the plane.

The n-dimensional feature space needs to be split into two areas, one for every class. Essentially, we define a mapping function that maps an input sample $\mathbf{x}$ into a target/label $\mathbf{y}$ $y = sign(f(\mathbf{x})) \in \{1, -1\}$.

$$f(\mathbf{x}) = \mathbf{x}^T\mathbf{w} + b = \begin{cases} \geq 0, & y = sign(f(\mathbf{x})) = 1,\ \mathbf{x} \in P \\ < 0, & y = sign(f(\mathbf{x})) = -1,\ \mathbf{x} \in N \end{cases} \tag{4.3}$$

Points $\mathbf{x}$ that belong to the positive side or exactly in the hyperplane will be classified as 1 otherwise as -1.

Example [6]

A straight line in 2D space $\mathbf{x} = [x_1, x_2]^T$ can be described by the following equation:

	Input $(\mathbf{x}, y)$	Output $y' = sign(f(\mathbf{x}))$	result
1	$(\mathbf{x}, y = -1)$	$y' = 1 \neq y$	incorrect
2	$(\mathbf{x}, y = 1)$	$y' = -1 \neq y$	incorrect

Table 4.1: Incorrect Cases

$$f(\mathbf{x}) = \mathbf{x}^T\mathbf{w} + b = [x_1, x_2]\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} + b = [x_1, x_2]\begin{bmatrix} 1 \\ 2 \end{bmatrix} - 1 = x_1 + 2x_2 - 1 = 0$$

This line divides the 2D plane into two halves. The distance between the origin and the line is

$$\frac{|b|}{||\mathbf{w}||} = \frac{1}{\sqrt{w_1^2 + w_2^2}} = \frac{1}{\sqrt{5}} = 0.447$$

Consider two points:

- $\mathbf{x}_1 = [1,\ 0.25]^T$, $f(\mathbf{x}_1) = 1 + 2 \times 0.25 - 1 = 0.5 > 0$, i.e., $\mathbf{x}_1$ is above the straight line;
- $\mathbf{x}_2 = [0.5,\ 0]^T$, $f(\mathbf{x}_2) = 0.5 + 2 \times 0 - 1 = -0.5 < 0$, i.e., $\mathbf{x}_2$ is below the straight line.

Learning Problem

Consider the following set of training samples with K length and also assume that the classes P and N are linearly separable.

$$\{(\mathbf{x}_k, y_k), k = 1, \cdots, K\}$$

where $y_k \in \{1, -1\}$ is the label of $\mathbf{x}_k$.

Our goal is to find a hyperplane to linearly separate P and N classes. To train the classifier we need to penalize our model every time it is mistaken. There are two cases that our model will be wrong 4.1.

For every mistake the vector $\mathbf{w}$ will be updated:

- ▶ If $(\mathbf{x}, y = -1)$ but $y' = 1 \neq y$, then

$$\mathbf{w}^{new} = \mathbf{w}^{old} + \eta y \mathbf{x} = \mathbf{w}^{old} - \eta \mathbf{x}$$

- ▶ If $(\mathbf{x}, y = 1)$ but $y' = -1 \neq y$, then

$$\mathbf{w}^{new} = \mathbf{w}^{old} + \eta y \mathbf{x} = \mathbf{w}^{old} + \eta \mathbf{x}$$

The above cases can be summed up to one learning rule as follows:

$$\text{if } \; y f(\mathbf{x}) = y(\mathbf{x}^T \mathbf{w}^{old} + b) < 0, \; \text{ then } \; \mathbf{w}^{new} = \mathbf{w}^{old} + \eta y \mathbf{x} \tag{4.4}$$

While, when the model is correct we can write:

$$y f(\mathbf{x}) = y(\mathbf{x}^T \mathbf{w} + b) \geq 0 \tag{4.5}$$

It is assumed initially $\mathbf{w} = 0$, and the K training samples are presented repeatedly, the learning law during training will yield eventually a linear combination of the training samples:

$$\mathbf{w} = \sum_{i=1}^{K} \alpha_i y_i \mathbf{x}_i \tag{4.6}$$

where $\alpha_i > 0$.

This happens because whenever a new sample $(\mathbf{x}_i, y_i)$ is received the vector $\mathbf{w}$ is updated by:

$$\text{if } \; y_i f(\mathbf{x}_i) = y_i(\mathbf{x}_i^T \mathbf{w}^{old} + b) = y_i \left(\sum_{j=1}^{m} \alpha_j y_j (\mathbf{x}_i^T \mathbf{x}_j) + b \right) < 0,$$

$$\text{then } \; \mathbf{w}^{new} = \mathbf{w}^{old} + \eta y_i \mathbf{x}_i = \sum_{j=1}^{m} \alpha_j y_j \mathbf{x}_j + \eta y_i \mathbf{x}_i, \quad \text{i.e.} \quad \alpha_i^{new} = \alpha_i^{old} + \eta$$

What it is achieved now is to describe the decision function 4.7 and the learning rule 4.8 in terms of inner products of the input vectors.

$$f(\mathbf{x}) = \mathbf{x}^T\mathbf{w} + b = \sum_{j=1}^{k} \alpha_j y_j (\mathbf{x}^T\mathbf{x}_j) + b \tag{4.7}$$

$$\text{if } \; y_i \left(\sum_{j=1}^{k} \alpha_j y_j (\mathbf{x}_i^T\mathbf{x}_j) + b \right) < 0, \quad \text{then } \; \alpha_i^{new} = \alpha_i^{old} + \eta \tag{4.8}$$

SVM hyperplane

If equation 4.5 is satisfied for all the samples that belong to either P or N class then the decision hyperplane 4.1 separates the two classes.

However, the goal is to find the optimal hyperplane that separates the two classes with the maximal margin (distance between the decision plane and the closest sample points). This hyperplane should be in the middle of the two classes.

Let's define the following hyperplanes:

- H_+ is the hyperplane closer to the points in P class
- H_- is the hyperplane closer to the points in N class
- H_0 it is the optimal hyperplane in the middle of H_+ and H_-.

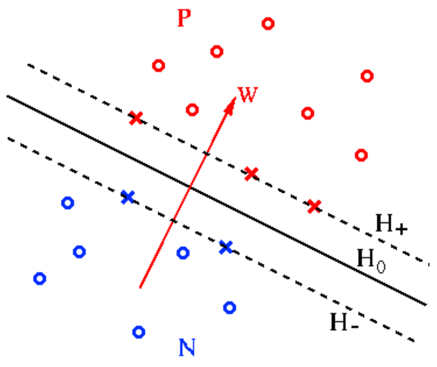


Figure 4.1: Graphical Representation of hyperplanes and samples [6]

All points $\mathbf{x}_i$ should satisfy the following inequality:

$$y_i(\mathbf{x}_i^T\mathbf{w}+b) \geq 1, \quad (i=1,\cdots,m) \tag{4.9}$$

The equality holds for those points on the planes H_+ or H_-. These points are named *support vectors*, for which the following applies.

$$\mathbf{x}_i^T\mathbf{w}+b = y_i \textbf{ or } b = y_i - \mathbf{x}_i^T\mathbf{w} = y_i - \sum_{j=1}^{m} \alpha_j y_j (\mathbf{x}_i^T\mathbf{x}_j) \tag{4.10}$$

Additionally it can be stated that the distances from the origin to the three parallel planes H_-, H_0 and H_+ are, respectively, $|b-1|/||\mathbf{w}||$, $|b|/||\mathbf{w}||$, and $|b+1|/||\mathbf{w}||$, and the distance between planes H_- and H_+ is $2/||\mathbf{w}||$.

Our goal is to maximize this distance, or, equivalently, to minimize the norm $||\mathbf{w}||$. Now the problem of finding the optimal decision plane in terms of $\mathbf{w}$ and b can be formulated as:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\mathbf{w}^T\mathbf{w} = \frac{1}{2}||\mathbf{w}||^2 \quad \text{(objective function)} \\ \text{subject to} & y_i(\mathbf{x}_i^T\mathbf{w}+b) \geq 1, \text{ or } 1 - y_i(\mathbf{x}_i^T\mathbf{w}+b) \leq 0, \quad (i=1,\cdots,m) \end{array}$$

Since the objective function is quadratic, this constrained optimization problem is called a quadratic program (QP) problem. (If the objective function is linear instead, the problem is a linear program (LP) problem). This QP problem can be solved by Lagrange multipliers method to minimize the following:

$$L_p(\mathbf{w},b,\alpha) = \frac{1}{2}||\mathbf{w}||^2 + \sum_{i=1}^{m} \alpha_i(1 - y_i(\mathbf{x}_i^T\mathbf{w}+b)) \tag{4.11}$$

with respect to $\mathbf{w}$, b and the Lagrange coefficients $\alpha_i \geq 0$ $(i = 1,\cdots,\alpha_m)$. We let

$$\frac{\partial}{\partial W}L_p(\mathbf{w},b) = 0, \quad \frac{\partial}{\partial b}L_p(\mathbf{w},b) = 0 \tag{4.12}$$

These lead, respectively, to

$$\mathbf{w} = \sum_{j=1}^{m} \alpha_j y_j \mathbf{x}_j, \quad \text{and} \quad \sum_{i=1}^{m} \alpha_i y_i = 0 \tag{4.13}$$

Substituting these two equations back into the expression of $L(\mathbf{w}, b)$, we get the *dual problem* (with respect to α_i) of the above *primal problem*:

$$\begin{aligned} &\text{maximize} \quad L_d(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T, \mathbf{x}_j \\ &\text{subject to} \quad \alpha_i \geq 0, \quad \sum_{i=1}^{m} \alpha_i y_i = 0 \end{aligned}$$

Solving this dual problem, α_i is obtained and then the optimal plane $\mathbf{w}$ can be found.

Those points $\mathbf{x}_i$ on either of the two planes H_+ and H_- (for which the equality $y_i(\mathbf{w}^T \mathbf{x}_i + b) = 1$ holds) are called *support vectors* and they correspond to positive Lagrange multipliers $\alpha_i > 0$. The training depends only on the support vectors, while all other samples away from the planes H_+ and H_- are not important.

For a support vector $\mathbf{x}_i$ (on the H_- or H_+ plane), the constraining condition is

$$y_i \left(\mathbf{x}_i^T \mathbf{w} + b \right) = 1 \quad (i \in sv) \tag{4.14}$$

sv is a set of all indices of support vectors $\mathbf{x}_i$ (corresponding to $\alpha_i > 0$). Substituting

$$\mathbf{w} = \sum_{j=1}^{m} \alpha_j y_j \mathbf{x}_j = \sum_{j \in sv} \alpha_j y_j \mathbf{x}_j \tag{4.15}$$

we get

$$y_i \Big(\sum_{j \in sv} \alpha_j y_j \mathbf{x}_i^T \mathbf{x}_j + b \Big) = 1 \tag{4.16}$$

Note that the summation only contains terms corresponding to those support vectors $\mathbf{x}_j$ with $\alpha_j > 0$, i.e.

$$y_i \sum_{j \in sv} \alpha_j y_j \mathbf{x}_i^T \mathbf{x}_j = 1 - y_i b \tag{4.17}$$

For the optimal weight vector $\mathbf{w}$ and optimal b, we have:

$$\begin{aligned} ||\mathbf{w}||^2 &= \mathbf{w}^T\mathbf{w} = \sum_{i \in sv} \alpha_i y_i \mathbf{x}_i^T \sum_{j \in sv} \alpha_j y_j \mathbf{x}_j = \sum_{i \in sv} \alpha_i y_i \sum_{j \in sv} \alpha_j y_j \mathbf{x}_i^T \mathbf{x}_j \\ &= \sum_{i \in sv} \alpha_i (1 - y_i b) = \sum_{i \in sv} \alpha_i - b \sum_{i \in sv} \alpha_i y_i \\ &= \sum_{i \in sv} \alpha_i \end{aligned}$$

The last equality is due to $\sum_{i=1}^m \alpha_i y_i = 0$ shown above. Recall that the distance between the two margin planes H_+ and H_- is $2/||\mathbf{w}||$, and the margin, the distance between H_+ (or H_-) and the optimal decision plane H_0, is

$$\frac{1}{||\mathbf{w}||} = \left(\sum_{i \in sv} \alpha_i \right)^{-1/2} \tag{4.18}$$

Soft Margin SVM

When the two classes are not linearly separable the condition for the optimal hyperplane can be relaxed by including an extra term:

$$y_i(\mathbf{x}_i^T \mathbf{w} + b) \geq 1 - \xi_i, \quad (i = 1, \cdots, m) \tag{4.19}$$

For minimum error, $\xi_i \geq 0$ should be minimized as well as $||\mathbf{w}||$, and the objective function becomes:

$$\begin{aligned} &\text{minimize} \quad \mathbf{w}^T\mathbf{w} + C \sum_{i=1}^m \xi_i^k \\ &\text{subject to} \quad y_i(\mathbf{x}_i^T \mathbf{w} + b) \geq 1 - \xi_i, \quad \text{and} \quad \xi_i \geq 0; \quad (i = 1, \cdots, m) \end{aligned}$$

Here C is a regularization parameter that controls the trade-off between maximizing the margin and minimizing the training error. Small C tends to emphasize the margin while ignoring the outliers in the training data, while large C may tend to overfit the training data.

When $k = 2$, it is called 2-norm soft margin problem:

$$\begin{aligned} &\text{minimize} \quad \mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{m} \xi_i^2 \\ &\text{subject to} \quad y_i(\mathbf{x}_i^T\mathbf{w} + b) \geq 1 - \xi_i, \quad (i = 1, \cdots, m) \end{aligned} \tag{4.20}$$

When $k = 1$, it is called 1-norm soft margin problem:

$$\begin{aligned} &\text{minimize} \quad \mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{m} \xi_i \\ &\text{subject to} \quad y_i(\mathbf{x}_i^T\mathbf{w} + b) \geq 1 - \xi_i \quad \text{and} \quad \xi_i \geq 0; \quad (i = 1, \cdots, m) \end{aligned} \tag{4.21}$$

The 1-norm is less sensitive to outliers in the training data. It is a good advice to use 1-norm SVM when there is a lot of noise in the data so that it can ignore the outliers.

Kernel Mapping

SVM converges only for linearly separable data. If the data set is not linearly separable, we can map the samples $\mathbf{x}$ into a feature space of higher dimensions:

$$\mathbf{x} \longrightarrow \phi(\mathbf{x})$$

in which the classes can be linearly separated. The decision function in the new space becomes:

$$f(\mathbf{x}) = \phi(\mathbf{x})^T\mathbf{w} + b = \sum_{j=1}^{m} \alpha_j y_j (\phi(\mathbf{x})^T\phi(\mathbf{x}_j)) + b \tag{4.22}$$

where

$$\mathbf{w} = \sum_{j=1}^{m} \alpha_j y_j \phi(\mathbf{x}_j)$$

and b are the parameters of the decision plane in the new space. As the vectors $\mathbf{x}_i$ appear only in inner products in both the decision function and the learning rule, the mapping function $\phi(\mathbf{x})$ does not need to be explicitly specified. Instead, all we need is the inner product of the vectors in the new space. The function $\phi(\mathbf{x})$ is a kernel-induced *implicit* mapping.

Definition: A kernel is a function that takes two vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ as arguments and returns the value of the inner product of their images $\phi(\mathbf{x}_i)$ and $\phi(\mathbf{x}_j)$:

$$K(\mathbf{x}_1, \mathbf{x}_2) = \phi(\mathbf{x}_1)^T \phi(\mathbf{x}_2) \tag{4.23}$$

As only the inner product of the two vectors in the new space is returned, the dimensionality of the new space is not important.

The learning algorithm in the kernel space can be obtained by replacing all inner products in the learning algorithm in the original space with the kernels:

$$f(\mathbf{x}) = \phi(\mathbf{x})^T \mathbf{w} + b = \sum_{j=1}^{m} \alpha_j y_j K(\mathbf{x}, \mathbf{x}_j) + b \tag{4.24}$$

The parameter b can be found from any support vectors $\mathbf{x}_i$:

$$b = y_i - \phi(\mathbf{x}_i)^T \mathbf{w} = y_i - \sum_{j=1}^{m} \alpha_j y_j (\phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)) = y_i - \sum_{j=1}^{m} \alpha_j y_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{4.25}$$

Example 1 [6]: linear kernel

Assume $\mathbf{x} = [x_1, \cdots, x_n]^T$, $\mathbf{z} = [z_1, \cdots, z_n]^T$,

$$K(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T \mathbf{z} = \sum_{i=1}^{n} x_1 z_1$$

Example 2 [6]: polynomial kernels

Assume $\mathbf{x} = [x_1, x_2]^T$, $\mathbf{z} = [z_1, z_2]^T$,

$$\begin{aligned} K(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^T \mathbf{z})^2 &= (x_1 z_1 + x_2 z_2)^2 = x_1^2 z_1^2 + x_2^2 z_2^2 + 2 x_1 z_1 x_2 z_2 \\ &= <(x_1^2, x_2^2, \sqrt{2} x_1 x_2), (z_1^2, z_2^2, \sqrt{2} z_1 z_2)> = \phi(\mathbf{x})^T \phi(\mathbf{z}) \end{aligned}$$

This is a mapping from a 2-D space to a 3-D space. The order can be changed from 2 to general d.

Example 3 [6]: RBF kernels

$$K(\mathbf{x},\mathbf{z}) = e^{-||\mathbf{x}-\mathbf{z}||^2/2\sigma^2}$$

1..2 K Nearest Neighbors

K nearest neighbor [22] is a non-parametric technique which can learn what the true function is on our problem but without knowing too much about that true function. Nearest Neighbor uses the term ”distance” to express the distances between the feature vectors in a n-D feature space. It has a complexity $O(dn^2)$ because every point is inspected and used to calculate the euclidean distance to a vector x. Each distance calculator is $O(d)$. The nearest neighbor for a vector x is the one which is the closest to it. The k nearest neighbors are the k vectors closest to x. To denote neighbors of a vector the following will be used:

$NN(\vec{x}, j)$ where the index j is the j^{th} nearest neighbor of $\vec{x}$.

Then, to classify a sample, we average the labels of neighbors to estimate:

$$\hat{p}(\vec{x}) = \frac{1}{k}\sum_{j=1}^{k} y_{NN(\vec{x},j)} \tag{4.26}$$

After that, $\hat{p}(\vec{x})$ is threshold-ed:

$$\hat{c}(\vec{x}) = \mathbf{1}(\hat{p}(\vec{x}) \geq 0.5) \tag{4.27}$$

Analysis of 1-Nearest-Neighbors

Let’s suppose that $y_i = c(\vec{x}_i)$ and there is no noise in the outcomes. If Nearest Neighbor algorithm cannot learn the target function without noise then the method should not be used.

When setting k= 1 and no noise the Risk of the method will be:

$$P(c(\vec{X}) \neq c(\vec{X}_{NN})) \tag{4.28}$$

We would like the Risk 4.28 to be 0 as $n \to \infty$.

Convergence of the nearest neighbor

We require the following:

$$\|\vec{x}_{NN} - \vec{x}\| \to 0 \tag{4.29}$$

Additionally, we pick a very small distance $\epsilon > 0$ and assume that all samples are Independent and Identically Distributed(IID). Then we state that the nearest neighbor is more than ϵ away from $\vec{x}$ if and only if every $\vec{x_i}$ is more than ϵ away.

$$P(\|\vec{x}_{NN} - \vec{x}\| > \epsilon) = P(\forall i, \|\vec{x}_i - \vec{x}\| > \epsilon) \tag{4.30}$$

Because of the samples being IID:

$$\begin{aligned}
P(\|\vec{X}_{NN} - \vec{x}\| > \epsilon) &= P(\forall i, \|\vec{X}_i - \vec{x}\| > \epsilon) && (4.31)\\
&= \prod_{i=1}^{n} P(\|\vec{X}_i - \vec{x}\| > \epsilon) && (4.32)\\
&= \Big(P(\|\vec{X} - \vec{x}\| > \epsilon)\Big)^n && (4.33)
\end{aligned}$$

which goes to 0 as $n \to \infty$, except $n = 1$. That means that Nearest Neighbor should not be suggested when there is zero probability and you are asked to predict a point in the middle of that region of zero probability.

Not Noise-Free function

If the true function is not noise-free then the risk for a vector x is:

$$P(Y_{new} \neq \hat{c}(\vec{x}_{new})) = P(Y_{new} \neq Y_{NN}) \tag{4.34}$$

$$= P(Y_{new} = 1, Y_{NN} = 0) + P(Y_{new} = 0, Y_{NN} = 1) \tag{4.35}$$

$$= p(\vec{x}(1 - p(\vec{x}_{NN})) + (1 - p(\vec{x}))p(\vec{x}_{NN}) \tag{4.36}$$

As $x\vec{_{NN}} \rightarrow \vec{x}$ the above equation is:

$$P(Y_{new} \neq \hat{c}(\vec{x}_{new}) = 2p(\vec{x})(1 - p(\vec{x})) \tag{4.37}$$

That means we have a risk at most twice as the risk of the optimal classifier.

Multiple Neighbors

If we use more neighbors, we will average through more data for each prediction, which reduces the variance because it averages together more noise terms (as k increases , variance decreases). In case k is decreased then it means we have less bias and we are under-fitting the model.

1..3 Decision Trees

A Decision Tree [47] can be described as a sequence of questions to approach the desired answer. It belongs to the category of non-parametric methods since it does not conform to a normal distribution. In such a tree, the root node is at the top, connected by successive branches to other nodes. The ”questions” that are asked in a decision tree can be answered with a binary value(yes or no). The classification of a particular pattern begins at the root node, which asks for the value of a particular property of the pattern. The different branches from the root node correspond to the different possible values.

The main advantage of a decision tree is the interpretability. It is fairly easy to express a decision tree outcome to logical sentences and make a meaning out of it.

CART

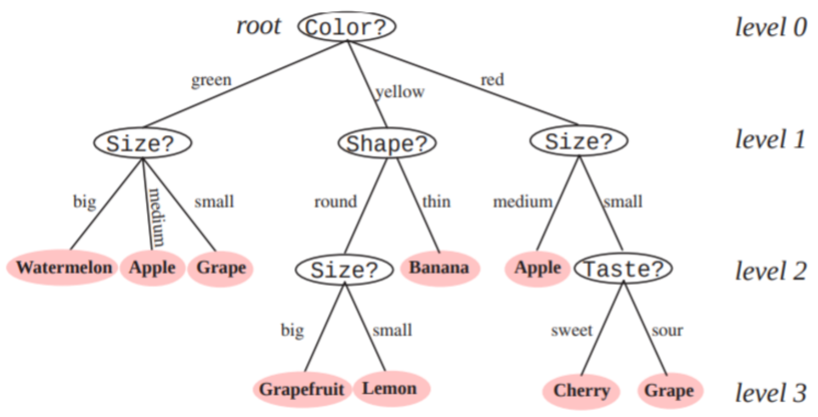


Figure 4.2: Classification in a basic decision tree.[23]

CART or else Classification and Regression Trees is a framework for tree-growing process. Given the data represented at a node, it needs to be declared if that node is going to be a leaf or split further. CART framework has six questions to answer[23]:

- Should properties be binary or multi valued?
- Which property should be tested at a node?
- When should a node be declared a leaf?
- If the tree becomes ”too large”, how can it be made smaller and simpler, i.e., pruned?
- If a leaf node is impure, how should the category label be assigned?
- How should missing data be handled?

Number of Splits

The number of splits is how many times the dataset will be split. This parameter is set by the designer of the model.

Test selection and node impurity

Most of the time when creating decision trees model is spent on what property/query test should be performed at each node. A fundamental principle as Duda and Hart state on their book [23] is to create the tree with simplicity and give an advantage to decisions that lead to a simple, compact tree with few nodes.

For that reason impurity is defined. We denote impurity of a node N as following, $i(N)$. The desired outcome is to have impurity to be zero if all of the patterns that reach the node belong to the same class, otherwise if both classes are represented equally, a large impurity is necessary.

Some impurity measures are entropy, variance, gini and misclassification.

Let:

- Entropy Impurity: $i(N) = -\sum_j P(w_j) log_2 P(w_j)$
- Gini Impurity: $i(N) = \sum_{i \neq j} P(w_i)P(w_j) = 1 - \sum_j P^2(w_j)$
- Misclassification Impurity: $i(N) = 1 - max_j P(w_j)$
- Variance Impurity: $i(N) = P(w_1)P(w_2)$

$P(w_j)$ is the fraction of patterns at a node N that are in category w_j. If all the patterns are of the same category, the impurity is 0; otherwise it gets a positive value with the greatest value occurring when the different classes are equally likely.

Gini impurity is the expected error rate at node N if the category label is selected randomly from the class distribution present at N. This metric is suggested when the probabilities are equal same as the misclassification impurity.

Choosing a value for the property test T

An heuristic that can be used is to choose the test that decreases the impurity as much as possible. The drop in impurity is defined as:

$$\Delta_i(N) = i(N) - P_L i(N_L) - (1 - P_L) i(N_R) \tag{4.38}$$

where N_L and N_R are the left and right descendants nodes, $i(N_L)$ and $i(N_R)$ their impurities, and P_L is the fraction of patterns at node N that will go to N_L when the property test is asked.

When to stop splitting

This is a very important chapter of building a decision tree because if splitting does not stop, the model will overfit until it classifies every sample correctly. We do not want that since our model will not generalize well. A classical method to stop splitting is cross-validation. Another method is to set a small threshold value in the reduction in impurity. Additionally, a statistical significance test can be used in the reduction of impurity (chi-square test).

Pruning

The previous method of stop splitting has a big disadvantage. It is influenced by the descendent nodes, which is something not wanted. It lacks of looking ahead and it is biased to growing trees which the greatest impurity reduction is near the root node.

Hence, pruning is another way to train a decision tree without overfitting. In pruning, a tree is grown fully (leaf nodes have zero impurity) and then all pairs of neighboring leaf nodes are considered for elimination. Any pair whose elimination yields a satisfactory increase in impurity is eliminated, and the common antecedent node declared a leaf.

Other Methods

Some other popular tree algorithms are ID3 and C4.5. ID3 is intended for use with nominal inputs only. If the problem involves real-valued variables, they are first binned into intervals, each interval being treated as an unordered nominal attribute.

The C4.5 algorithm is the most used in classification tasks. Real-valued variables are treated the same as in CART methodology. This method uses heuristics for pruning derived based on the statistical significance of splits.

2. Unsupervised

Unsupervised learning algorithms experience a dataset containing some features, which can be used to learn useful patterns or structures of the dataset. A task that unsupervised learning is common is clustering. Broadly speaking, these type of algorithms observe a set of samples and attempt to learn the probability distribution or some interesting properties of that probability distribution. The term unsupervised learning is given because in this case there is no instructor that shows the model how to learn.

2..1 Isolation Forest

Isolation Forest returns the anomaly score of each sample with a specific metric that is related to tree structure. It isolates observations by randomly selecting a feature and then randomly selecting a split value between maximum and minimum values of the selected features.

A tree structure can help in representing, a recursive partitioning as described in above paragraph. The number of splittings required to isolate a sample is equivalent to the path length from the root node to the terminating. The main logic of this method is that "A regular point is much harder to isolate than an anomalous point".

In order to isolate a regular point, more partitions will be needed. The number of partitions tells us if something is a regular or anomalous point. The training procedure can be described in 11.

Algorithm 1 Isolation Forest with B number of trees

```
1: Input dataset X $\in \mathbb{R}^{n \times p}$
2: for b = 1,...,B do
3:     Set tree(t) as null
4:     if nrow(X) == 1 then return t
5:     Randomly select a feature $x_i$
6:     Randomly select a split point p $\in$ (min($x_i$),max($x_i$))
7:     Add to t the node $N_{x_i,p}$
8:     Define $X_l$ and $X_r$ as the matrix composed of the samples of X
       where the variable $x_i$ is respectively larger and smaller than p.
9:     Repeat the algorithm with $X = X_l$. Link the obtained tree as the
       left child of t.
10:    Repeat the algorithm with $X = X_r$. Link the obtained tree as the
       right child of t.
11: end for
```

Then to predict if a sample is anomalous or regular, a metric needs to be defined. This metric is named "Anomaly Score" and it is denoted as S(x,m), where x denotes the sample and m the length of the dataset.

$$S(x,m) = 2^{-\frac{E(h(x))}{c(m)}} \tag{4.39}$$

The equation 4.39 intuitively says, what is the depth of finding this particular point x across all isolation trees that are constructed compared to the depth of finding an average point.

$E[h(x)]$ is the average search height for point x from the isolation trees created. While $c(m)$ is a normalization constant usually denoting the average value depth of a point x.

When the anomalous score is close to 1 the sample is flagged as anomalous while if the anomalous score is close to 0.5 the sample is flagged as normal.

Isolation Forest hyperparameters are the number of isolation trees, the sampling size (m) and the threshold of flagging a point as regular or anomalous.

2..2 Local Outlier Factor

As the name suggests, Local Outlier Factor (LOF) is a method to identify outliers. This method considers the density of the neighborhood and its performance increases when the density of the data is not similar through whole dataset.

Before writing LOF's equation, first some other terms need to be defined. One of them is **K-distance**, which is the distance between a point and its Kth nearest neighbor. The second term is the **K-neighbors** ($N_k(A)$) that includes points in a set that lie in or on the circle of radius **K-distance**.

Then, Reachability Density (RD) needs to be defined as following:

$$RD(X_i, X_j) = max(K - distance(X_j), distance(X_i, X_j)) \tag{4.40}$$

Distance can be either euclidean or manhattan or something else. The equation 4.40 basically says that if a point X_i lies withing the K-neighbors of X_j, the reachability distance will be K-distance of X_j, else it will be the distance between X_i and X_j.

After that the term Local Reachability(LRD) Density is given by the following equation.

$$LRD_k(A) = \frac{1}{\sum_{X_j \in N_k(A)} \frac{RD(A,X_j)}{||N_k(A)||}} \tag{4.41}$$

LRD is expressing how far a point is from the nearest cluster of points. If LRD is low it is implied that the closest cluster is far from the point.

Finally, the Local Outlier Factor equation is defined combining all the previous equations as:

$$LOF_k(A) = \frac{\sum_{X_j \in N_k(A) LRD_k(X_j)}}{||N_k(A)||} \times \frac{1}{LRD_k(A)} \tag{4.42}$$

Essentially, LOF is the ratio of the average LRD of the K neighbors of point A to the LRD of A. If LOF is close to 1 then the point is not an outlier. Otherwise, if LOF has a high value (higher than 1) then the point is flagged as outlier.

2..3 K-Means

In this subsection a definition for K-Means will be given. For this purpose, we will see how Christopher Bishop [14] describes this unsupervised algorithm.

Let's say we have a data set $\{x_1, x_2, ...x_N\}$. Each observation is a D-dimensional Euclidean variable x. The scope of this method is to cluster the data points into K number of clusters. At this point it is good to note that K should be a known variable a priori. But how do we define a cluster? A good way to understand it intuitively is to think about a comprised group of data points with inter-point distances very small compared with the distances to points outside of this certain cluster.

Suppose μ_k, where $k = 1, ..., K$, in which μ_k is a prototype that represents the k^{th} cluster. Usually, the prototype represents the center of a cluster or else the average of all data points inside it. Then, we would like to achieve an assignment of data points to clusters, as well as a set of vectors $\{\mu_k\}$, such that the sum of the squares of the distances of each data point to its closest center vector μ_k , is a minimum.

Now that we have given a definition to our problem, a mathematically description will be given. Let for each data point in the dataset x_i to have a corresponding set of binary indicator variables $r_{ik} \in \{0, 1\}$, where $k = 1, ..., K$. K describes the cluster that this data point belongs. Hence, if a data point x_i is assigned to cluster k then the following is true: $r_{ik} = 1$ and $r_{ij} = 0$ for $j \neq k$.

The objective function can be seen below and it represents the sum of the squares of the distances of each data point to its assigned vector μ_k, which is the center of cluster k.

$$J = \sum_{i=1}^{N} \sum_{k=1}^{K} r_{ik} ||X_i - \mu_k||^2 \tag{4.43}$$

Our goal is to minimize the above objective function by finding the right values for the $\{r_{ik}\}$ and the $\{\mu_k\}$. This can be done by utilizing an iterative procedure in which each iteration involves two successive steps corresponding to successive optimizations with respect to the values we are interested to find. This corresponds to the Expectation and Maximization steps of the EM algorithm. Let's analyze it further.

On the first step we minimize the objective function J with respect to the r_{ik} and while keeping μ_k fixed. In the second step we minimize J with respect to the μ_k while having fixed the r_{ik} term. This procedure will be repeated until the algorithm converges.

Mathematically the E-step can be expressed as:

$$r_{ik} = \begin{cases} 0 & \text{if } k = argmin_j ||X_i - \mu_j||^2 \\ 1 & \text{otherwise} \end{cases} \tag{4.44}$$

Now for the M-step, to optimize μ_k with r_{ik} fixed we need to set the objective function's derivative to zero and solve with respect to μ_k. Since J is a quadratic function we will not have any problem doing it. By solving the following equation:

$$2\sum_{i=1}^{N} r_{ik}(x_i - \mu_k) = 0 \tag{4.45}$$

with respect to μ_k we get:

$$\mu_k = \frac{\sum_i r_{ik}x_i}{\sum_i r_{ik}} \tag{4.46}$$

The denominator in this expression is equal to the number of points assigned to cluster k. This result can be interpreted as setting μ_k equal to the average of all of the data points assigned to cluster k. This procedure is repeated until there is no further change in the assignments or until some maximum number of iterations is exceeded.

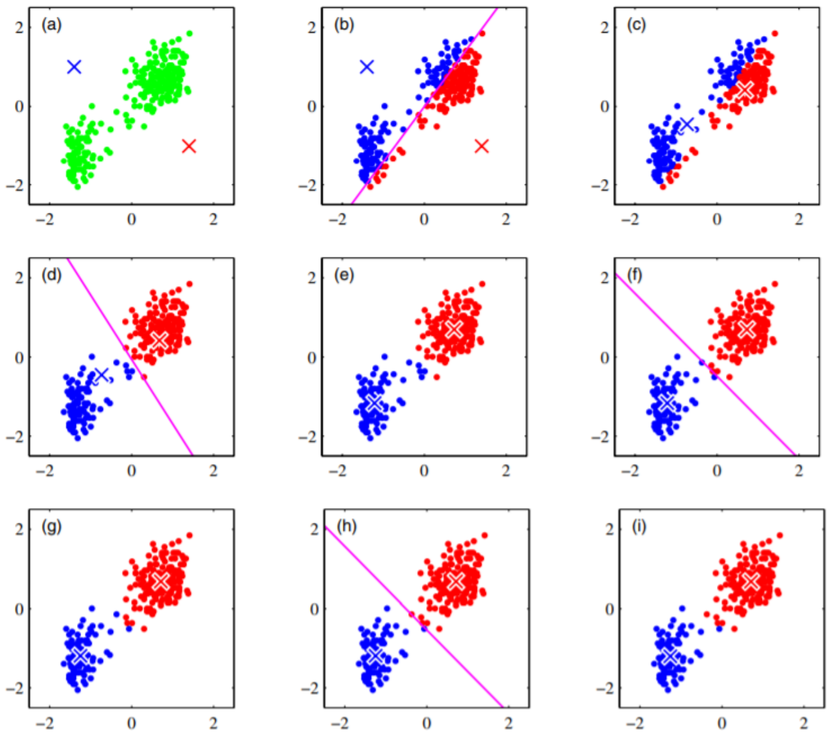


Figure 4.3: Illustration of K-Means steps.[14]

2..4 One Class SVM

Before describing the one-class SVM algorithm, a definition of what one class classification (OCC) [43] is will be given. OCC, tries to identify objects of a specific class amongst all objects, by primarily learning from a training set that only contains data of the class with the majority data points. This can help when we are interested in an anomaly detection problem, in which there is a huge data imbalance between the classes.

OCC has plenty applications such as in document classification or mostly in biomedical studies. In biomedical studies where often data from other classes can be difficult or impossible to obtain. To approach an OCC problem you can follow the three types described below:

Density Estimation

Density Estimation methods rely on estimating the density of the data points, and set the threshold. To apply this approach firstly, you should assume a distribution. Most used ones are Gaussian and Poisson.

Boundary Methods

Boundary Methods focus on setting boundaries around few set of data points. It relies on distance and hence they are not robust to scale variance. A good example of this method is one class svm or K-centers.

Reconstruction methods

Reconstruction methods use prior knowledge to build a generating model that best fits the data. Methods that follow this approach are K-Means or Self-Organizing Maps [34].

Mathematical Formulation of one-class SVM

To describe One-Class SVM mathematically we will be based on Y. Chen, Zhou and Huang formulation [19].

Let $\{x_1, .., x_n\} \in D$ and Let ϕ be a feature map so that $D \rightarrow F$. The goal is to map the data into the feature space and then try to use a hyper-sphere to describe the data in that space while putting most of the data inside the hyper-sphere. This can be seen as an optimization problem where we would like a sphere as small as possible while it includes most of the training data. Note that we only consider positive points since it is an OCC problem.

$$\begin{aligned} &min_{R\in\mathbb{R}, z\in\mathbb{R}, c\in F} \quad R^2 + \frac{\sum_i z_i}{vn} \\ &\text{subject to} \quad ||\phi(x_i) - c||^2 \leq R^2 + z_i \quad \text{and} \quad z_i \geq 0; \quad (i = 1, \cdots, n) \end{aligned} \tag{4.47}$$

Parameter $v \in [0, 1]$ is the trade-off between the radius of the hyper-sphere and the number of training samples it can hold. By using the Lagrangian's multipliers we get the following equation:

$$L(R, z, c, a, b) = R^2 + \sum_{i=1}^{n} a_i(||\phi(x_i) - c||^2 - R^2 - z_i) + \frac{\sum_{i=1}^{n} z_i}{vn} - \sum_{i=1}^{n} b_i z_i \tag{4.48}$$

After that we are taking the derivatives with respect to R , z and c to retrieve the equations below:

$$\sum_{i=1}^{n} a_i = 1 \tag{4.49}$$

$$0 \leq a_i \leq \frac{1}{vn} \tag{4.50}$$

$$c = \sum_{i=1}^{n} a_i \phi(x_i) \tag{4.51}$$

The 4.49 and 4.50 equations are constraints. The 4.51 equation shows that C can be expressed as the linear combination of $\phi(x)$ which makes it possible to express the dual form with kernel functions as we described on the supervised SVM chapter. You can use either a linear kernel or a non-linear kernel. In our case the non-linear kernel describes much better our data.

3. Ensemble Methods

Statistics and Machine Learning, use ensemble methods to obtain better predictive performance. Ensemble methods can use multiple learning algorithms (always concrete). The simplest way to construct a committee is to average the predictions of a set of individual models. From a frequentist perspective by considering the trade-off between bias and variance, which decomposes the error due to a model into the bias component that arises from differences between the model and the true function to be predicted, and the variance component that represents the sensitivity of the model to the individual data points [15].

Bagging - Bootstrap Aggregating

Bagging method uses bootstrap datasets to generate models with variability. Suppose a classification problem in which we are trying to predict the class of an input **x**. Additionally, suppose M bootstrap datasets are generated and then use each to train a separate copy $y_m(\mathbf{x})$ of a predictive model where $m = 1, ..., M$. Then, the committee prediction is given by:

$$y_{com}(\mathbf{x}) = \frac{1}{M}\sum_{m=1}^{M} y_m(\mathbf{x}) \tag{4.52}$$

As Bishop states in [15] the average error of a model can be reduced by a factor of M simply by averaging M versions of the model. But unfortunately, it depends on the **key assumption** that the errors due to the individual models are uncorrelated. In practice the errors are typically highly correlated, and the reduction in overall error is generally small.

Random Forest

Random Forest is an ensemble learning method for machine learning tasks such as classification or regression. The training algorithm for random forests applies the general technique of bagging to tree learners. Given a training set $X = x_1, ...x_n$ with responses $Y = y_1, ..., y_n$ bagging repeatedly let's say B times selects a random sample with replacement of the training set and fits trees to these samples. The algorithm looks as follows:

Algorithm 2 Bagging Trees Algorithm

```
1: for b = 1,...,B do
2:     1. Sample with replacement, n training examples from X,Y; ($X_b$,$Y_b$)
3:     2. Train a classification tree or regression tree $f_b$ on $X_b$,$Y_b$
4: end for
```

After training, the predictions for the test samples x' can be made by the majority vote in case of classification or by averaging through regression trees in case of regression.

The random forest algorithm has only one difference than the bagging tree algorithm. Random forests use a modified tree learning algorithm that selects, at each candidate split in the learning process, a random subset of the features (feature bagging). The reason for doing this is the correlation of the trees in an ordinary bootstrap sample: if one or a few features are very strong predictors for the response variable, these features will be selected in many of the B trees, causing them to correlate. As we mentioned before, this is pretty bad according to Bishop [15].

Algorithm 3 Random Forest Algorithm

```
1: randomly select "k" features from total "m" features where k«m
2: Among "k" features, calculate the node "d" using the best split point
3: Split the node into children nodes using the best split.
4: Repeat 2 and 3 steps until leaf nodes are finalized
5: Built forest by repeating steps 1 to 4 for "n" number times to create
   "n" number of trees.
```

Boosting

Boosting uses 'base' classifiers to produce a form of committee with performance significantly better than any of the 'base' classifiers. Boosting can give good results even if the base classifiers have a performance that is only slightly better than random(that is why sometimes the base classifiers are known as weak learners)[15].

The main difference between boosting and bagging is that in boosting the base classifiers are trained in sequence, and each base classifier is trained using a weighted form of the data set in which the weighting coefficient associated with each data point depends on the performance of the previous classifiers. For example for misclassified points by one of the base classifiers a greater weight is given when used to train the next classifier in the sequence. Once all the classifiers have been trained, their predictions are then combined through a weighted majority voting scheme 4.4.

Gradient Boosting

Due to boosting another type of method came out named Gradient Boosting [24] that builds models like the other boosting methods but generalizes them by allowing optimization of an arbitrary differential loss function.

Many Gradient Boosting algorithms based on trees have been proposed. Some of them are XGBoost [18], LightGBM [33] and AdaBoost [48].

Below, an analysis of XGBoost (Extreme Gradient Boosting) will be provided from the creators of the framework [18]. The objective function

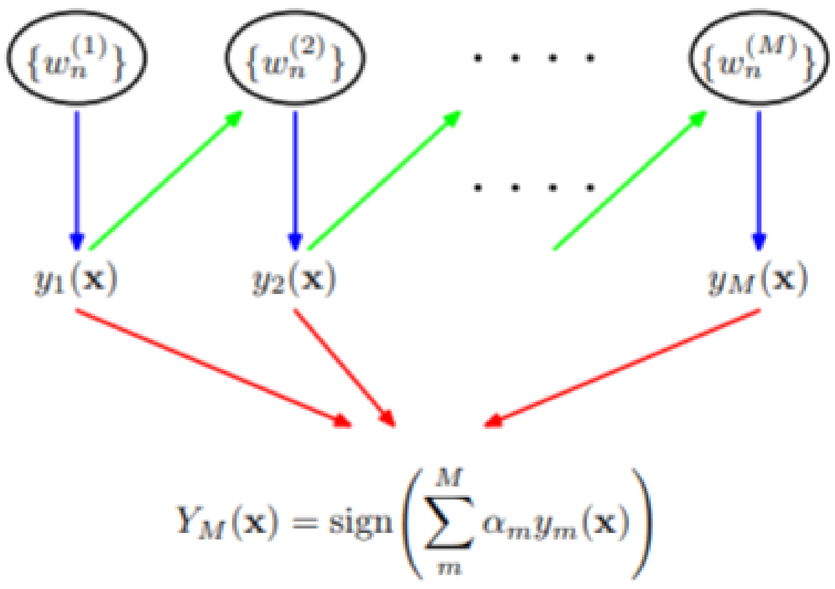


Figure 4.4: Schematic Illustration of the boosting framework [15].

consists of training loss and regularization term as follows:

$$\text{obj}(\theta) = L(\theta) + \Omega(\theta) \tag{4.53}$$

L is the training loss and Ω is the regularization term. The regularization term is used to prevent the model in becoming very complex. Hence, it avoids overfitting by penalizing variables that get too large. For classification problems usually the training loss function gets the binary-cross entropy form:

$$L(\theta) = \sum_i [y_i \ln(1 + e^{-\hat{y}_i}) + (1 - y_i) \ln(1 + e^{\hat{y}_i})] \tag{4.54}$$

Let's suppose there are K ensemble trees. The prediction scores of each individual tree are summed up to get the final score or in case of classification the majority class prevails. This can be written as:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), f_k \in \mathcal{F} \tag{4.55}$$

$\mathcal{F}$ is the set of all possible trees and f denotes an individual function of that set. Then the objective function of the ensemble K trees is the following:

$$\text{obj}(\theta) = \sum_{i}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \tag{4.56}$$

Until now, not big difference has been spotted between random forests and extreme gradient boosting. That is reasonable since the main difference is in the way of training of the ensemble trees.

The way of training gradient boosted trees is by defining an objective function and optimize it. We have already defined an objective function in equation 4.56. But it is impossible to optimize this complex model by just taking the gradient since trees' parameters are very difficult to model and use a gradient descent method to optimize. Every f_i has its own structure and leaf scores.

This problem can be surpassed by using a method called Additive Training. This method can be summed up as "Fix what you have learned and add one new tree at a time".

$$\begin{aligned}
\hat{y}_i^{(0)} &= 0 \\
\hat{y}_i^{(1)} &= f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i) \\
\hat{y}_i^{(2)} &= f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i) \\
&\dots \\
\hat{y}_i^{(t)} &= \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)
\end{aligned} \tag{4.57}$$

At this point, we need to figure out the objective function at step t. The tree that is chosen is the one that optimizes the objective function.

$$
\begin{aligned}
\text{obj}^{(t)} &= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \Omega(f_i) \\
&= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) + \text{constant}
\end{aligned}
\tag{4.58}
$$

Then the Structure Score comes into play. What it does is to re-formulate the tree model and re-write the objective value with the t_{th} tree's structure. This score will provide a decent measure of how good a tree is.

Overview of Boosting Methods

Adaboost is robust to overfitting in low noise datasets and it is easy to understand and visualize. On the other hand XGBoost uses regularized gradient boosting with both L1 and L2 regularization and allows a user to run a cross-validation at each iteration of the boosting. Also XGBoost can implement parallel processing being much faster than gradient boosting. However, XGBoost is difficult to interpret. Finally, LightGBM, uses a technique of Gradient-based one-side Sampling to filter out the data instances for finding a split value.

Voting Ensembles

Voting ensembles are ensemble machine learning models which combine predictions from multiple models for regression and classification. In regression, it calculates the average of the predictions from the models. In classification, the predictions for each label are summed and the label with the majority vote is predicted.

Voting Ensembles can be split into two categories. The methods that use soft voting and the methods that use hard voting. Hard voting predicts the class with the largest sum of votes from models while soft voting predicts the class with the largest summed probability from models.

4. Evaluation Metrics

Our dataset consists of two categories. The first is patients ailed from Covid-19 and patients that tested negative to the virus. It is worth to note that the data are heavily imbalanced. Since, positive patients are much more than negative. That means, that the accuracy does not say anything about the model performance. Actually, accuracy is only a subset of model performance.

Another methods, that should be taken into account for evaluation are F1- score, precision, recall, sensitivity, specificity and AUROC score. A sample can be either Positive or Negative when predicted and given its true label, this prediction can be True or False. Hence, it will be good to define the following terms:

Let:

- **True Positive** Patients that were diagnosed correctly with Covid-19.
- **True Negative** Patients that were diagnosed correctly for not having Covid-19.
- **False Negative** Patients that were diagnosed without Covid-19 but they were positive.
- **False Positive** Patients that were diagnosed with Covid-19 but actually they were negative.

Then, Confusion Matrix can be defined as below:

		True diagnosis		
		Positive	Negative	Total
Predicted	Positive	TP	FN	$TP + FN$
	Negative	FP	TN	$FP + TN$
	Total	$TP + FP$	$FN + TN$	N

Table 4.2: Confusion Matrix

Based on the previous variable we define the following evaluation metrics:

- **PRECISION** equals $\frac{TP}{TP+FP}$. It is a measure of the amount of accurate positives your model claims compared to the number of positives it actually claims. Precision is good when false negatives are less of a concern. For example in youtube recommendations.
- **RECALL** equals $\frac{TP}{TP+FN}$. It is the True positive rate the amount of positives your model claims compared to the actual number of positives. Recall is useful for medical diagnosis like cancer because False Negative is very important.
- **ACCURACY** equals $\frac{TP+TN}{TP+FP+TN+FN}$. it is the percent of the accurate to total predictions.
- **F1-SCORE** equals $\frac{2*precision*recall}{precision+recall}$. It is a good metric when the data are highly imbalanced.

AUC - ROC CURVE

AUC-ROC curve is a measurement for model performance in classification problems(usually binary). The term ROC is a probability curve and AUC represents a measure of separability. Essentially, it shows how much much the model is able to distinguish between the two classes. If AUC tends to 1 the model is perfect. Its values range from 0 to 1. A higher AUC means better distinguish between patients tested positive and negative in Covid-19.

A ROC curve plots TPR VS FPR at different classification thresholds. Lowering the classification threshold classifies more items as positive, thus increasing both False Positives and True Positives.

TPR is known as Recall or Sensitivity. FPR can be used to define another metric called Specificity. Specificity equals to $1 - FPR$.

Let:

- **TPR/Recall/Sensitivity** = $\frac{TP}{TP+FN}$
- **FPR** = $\frac{FP}{TN+FP}$
- **Specificity** = $\frac{TN}{TN+FP}$

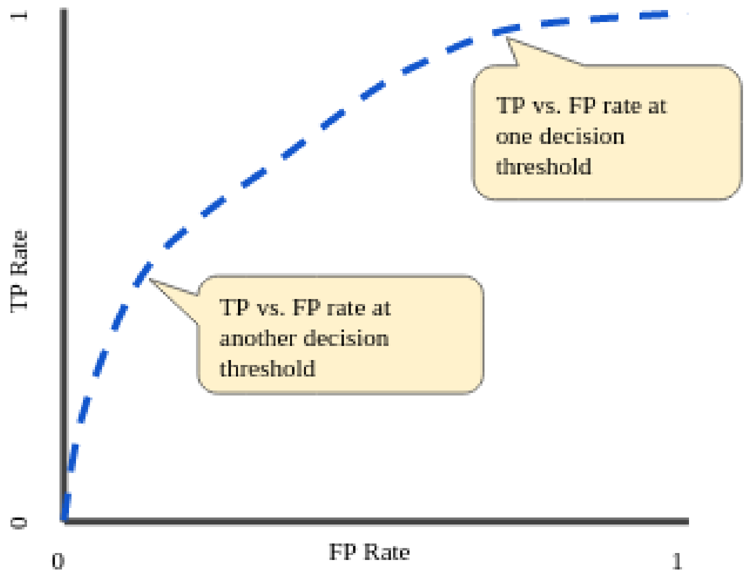


Figure 4.5: ROC CURVE [1]

Sensitivity and Specificity are inversely proportional to each other. When sensitivity is increased then specificity decreases and vice versa. Additionally, when threshold is decreased, more positive values are predicted and hence sensitivity increases and specificity decreases. If a threshold is increased then we will get more negative values thus higher specificity and lower sensitivity.

To compute the points in the ROC curve, we could evaluate a model many times with different classification thresholds. However, this would be very inefficient and slow. This problem can be solved by using a more efficient sorting-based algorithm, named AUC.

AUC means Area under the (ROC) Curve and measures the entire 2D area below the entire ROC curve 4.6.

AUC is able to provide an aggregate measure of performance across all possible classification thresholds. A way to interpret AUC is as the probability that the model ranks a random positive example higher than a random negative example 4.7.

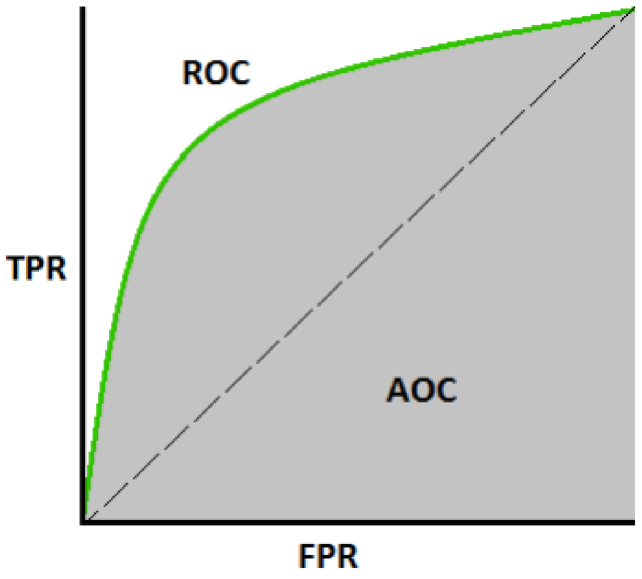


Figure 4.6: Area under the curve [7]

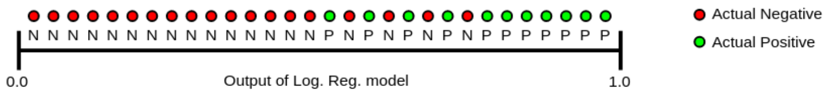

N N N N N N N N N N N N N N N P N P N P N P N P P P P P P P
0.0
Output of Log. Reg. model
1.0
Actual Negative
Actual Positive

Figure 4.7: Predictions ranked in ascending order of logistic regression score [1]

Splitting Dataset for train/test

Splitting the data into train and test groups is a very common thing to do when building machine learning models. The training set is used to train the model, and the validation/test set is to validate it on data it has never seen before. A classical approach is a 80-20 percent split or maybe 70-30 if a more robust model is needed.

When applying cross validation, the split can happen more than one times, let's say K times. Those K splits are called Folds. One method to split a dataset into K folds is to just split the data in K parts. Then we build K different models, each trained on the K-1 parts and tested on the K_{th} part. The algorithm runs K times so that every fold will be on the test fold at least once. The current paragraph can be summed up by figure 4.8.

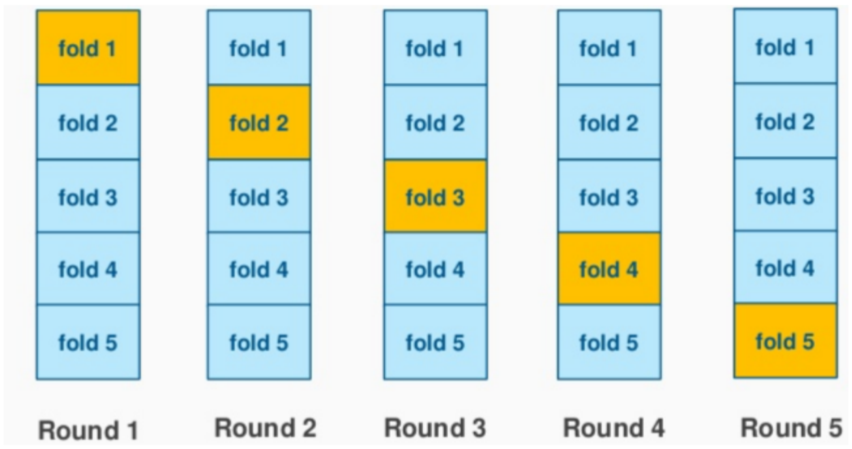


Figure 4.8: Cross Validation: K Fold [8]

In our case, we have a highly imbalanced dataset. That means that some folds may contain only one class. To fix this thing, Stratified Cross Validation can be used. Stratification seeks to ensure that each fold is representative of all strata of the data. This is done in a supervised way for classification and aims to ensure each class is equally represented across each test fold. For a graphical representation check 4.9.

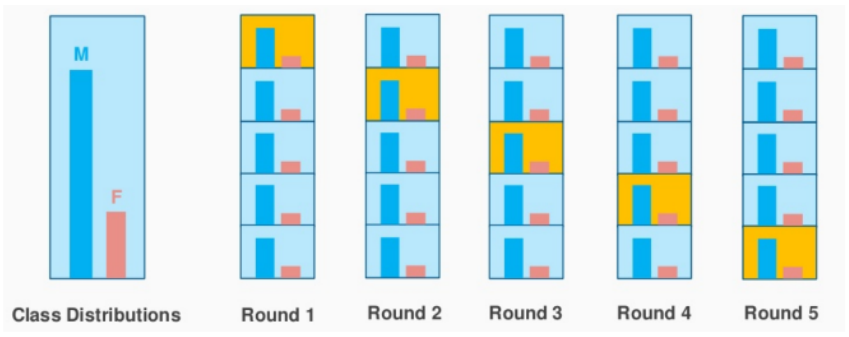


Figure 4.9: Stratified Cross Validation [8]

Stratification is not important for algorithms that weights class equally like ROC-AUC. Although, it is importance lies on measurements such as F-score or accuracy in which the over-represented class gets too much weight.

Chapter 5

Experiments and Results

1. Experiment Set Up

In this section, we are going to describe the settings of the experiments we conducted. The metrics that were used to compare the machine learning algorithms were accuracy, f1-score, recall, precision, auroc, sensitivity and specificity. A description of these metrics has been given on chapter 3.4 Evaluation Metrics. It should be mentioned that because of our data imbalance we do not consider seriously the accuracy metric since it will be biased towards the class with the majority of samples.

Additionally, we will provide how we perandom forestormed hyper-parameter tuning using grid-search in our supervised machine learning methods. Then, we should mention that the experiments were implemented with stratified train/test split using 80% of dataset as training and the other 20% as test set.

Finally, the data that were fed into machine learning models that are sensitive to distance have been scaled by Normalization which ends up with values between 0 and 1 with the following formula:

$$x_{norm} = \frac{x_i - x_{min}}{x_{max} - x_{min}} \tag{5.1}$$

2. Hyper-parameter tuning

As hyper-parameters we are defining the parameters that cannot be learnt directly from the training of the machine learning model and are passed as arguments to it. One example of this is the kernel type passed to support vector machines method. This is something that the model cannot learn while training but it is chosen by us.

A search of hyper-parameters should consist of the following properties:

1. a machine learning algorithm
2. a set of discrete parameters called parameter space
3. an evaluation metric
4. a scoring function

2..1 supervised methods

For tuning our supervised models we used Exhaustive Grid-Search. In grid-search tuning strategy all the possible combinations of the parameters we are using are being experimented. These parameters should be a discrete set. Grid search returns the model with the highest evaluation metric that has been chosen.

Support Vector Machines

In support vector machines we tuned the following parameters:

1. **Kernel type** which is chosen between linear, radial basis function and polynomial.
2. **C** which takes values from 0.1 to 10 with step 0.1
3. **Gamma** which is set to Auto or Scale in sklearn python framework.

Gamma parameter is the kernel coefficient for kernel types polynomial and radial basis function only. While **C** is the regularization parameter.

K Nearest Neighbors

In K nearest neighbors parameter space we chose to test the following settings:

1. **N_neighbors** (Number of neighbors) is set into the following values [1,3,5,7]
2. **Weights** is chosen between uniform and distance values.
3. **Metric** is chosen between euclidean and manhattan

The **Weights** parameter when is set to 'uniform' it means that all points are weighted equally in each neighborhood. While when it is set in 'distance' then the weight of the points is calculated by the inverse of their distance.

Decision Tree

The method Decision Tree was tuned with the following parameter space:

1. **Split Criterion** that takes values between gini and entropy.
2. **Max depth** of the tree which ranges from 1 to 15 with step 1.
3. **Minimum samples split** that ranges from 2 to 10 with step 1.
4. **Minimum samples leaf** that ranges from 2 to 5 with step 1.

Random Forest

1. **Mtry** number of drawn candidate variables in each split.
2. **Sample size** which is the number of observations that are drawn for each tree.
3. **Replacement** draw observations with or without bootstrap method.
4. **Node size** minimum number of observations in a terminal node.

5. **Number of Trees** number of trees in the forest which ranges from 2 to 15 with step 1.
6. **Splitting rule** which is the same to the decision tree split criterion parameter.

Xgboost

1. **Number of iterations** of the training. It is set from 2 to 5 with step 1.
2. **Learning Rate** which is set from 0.01 to 0.07 with step 0.01
3. **Max depth of trees** takes values in [3,4,5,6,7]
4. **Number of estimators** which takes values from 2 to 15 with step 1.
5. **Subsample** which takes values in [0.6, 0.8, 1.0].
6. **Colsample bytree** which takes values in [0.6, 0.8, 1.0].

Subsample means that XGBoost will receive randomly the percentage of, whatever value subsample parameter takes, training dataset so that it can avoid overandom forestitting.

Colsample bytree is an analogy between training samples and the feature columns while the tree is constructed.

2..2 Unsupervised Methods

In unsupervised methods, exhaustive grid search was not used. Instead, we followed a try and error strategy to find the optimal hyper-parameters. Next we will describe the hyper-parameters passed as arguments into the unsupervised machine learning algorithms.

K-Means

1. **Number of clusters** can be set to values 2 to 8.

2. **Init** is set to k-means++ to have a better initialization and convergence.

In K-Means anomaly detection we are trying to find the data points which are far away from the cluster centers over a certain percent for example 90 % farther.

OneClass SVM

1. **Kernel Type** which is set to radial basis function and polynomial. We avoid linear since our dataset is not linear separable.

Local Outlier Factor

In LOF method we compute the local density deviation of a given data point with respect to its neighbors. Hence, to classify a point as anomalous, this point needs to have density significantly different than the density around its neighbors.

1. **Number of neighbors** is set from 2 to 8
2. **Metric** is set to Auto so that the framework can decide the best algorithm to follow.
3. **P** is the parameter for Minkowski metric. We set p = 2 to define euclidean distance.

Isolation Forest

This method isolates each point in the dataset and splits them into anomalous and regular points. This separation considers the length of the path to separate the points. For example, if we try to split a point which is obviously a regular point, it will have many points in its round, so that it will be really difficult to isolate. While in the case of a point which is anomalous, it will be "alone" and it will be split easily between the other regular points.

1. **Number of estimators** which is the number of trees to be used. This is set to values between 3 and 10.
2. **Bootstrap** which is set to either True or False, so that replacement is used or not used respectively.

2..3 Oversampling methods to help in class Imbalance

Oversampling is used when a problem of imbalanced classification exists. Since there are is a small amount of examples in the minority class it is difficult for the model to ”draw” the decision boundary.

A solution to this is to use oversampling in the minority class. A very known algorithm to do this kind of job is called SMOTE (Synthetic Minority Oversampling Technique)[17].

What SMOTE does is choosing points close in the feature space, drawing a line between these points and then create a new sample at a point in that line. More specifically, a random data point is chosen from the minority class. After that, it finds the k nearest neighbors of that data point. Then it chooses randomly one of these neighbors to create the line between the random data point the method chose. Finally the new data point is a random point inside this line's feature space.

In addition we experimented with other variations of SMOTE for oversampling that are more selective in the creation of new data. One of this method is the Borderline-SMOTE[29]. The difference of this method is that it selects instances of the minority class that are miss-classified with a classification model as k-NN.

A very similar method to borderline-SMOTE is the borderline-SMOTE svm[45]. The difference here is that a support vector machine model is used instead of a k-NN.

Finally, an experiment with the use of Adasyn[30] oversampling method was conducted. Adasyn works by creating samples for the minority class according to their distributions. For example more synthetic data will be created for the minority class samples that are harder to learn compared to those minority samples that are easier to learn.

3. Graphical Representation of Data

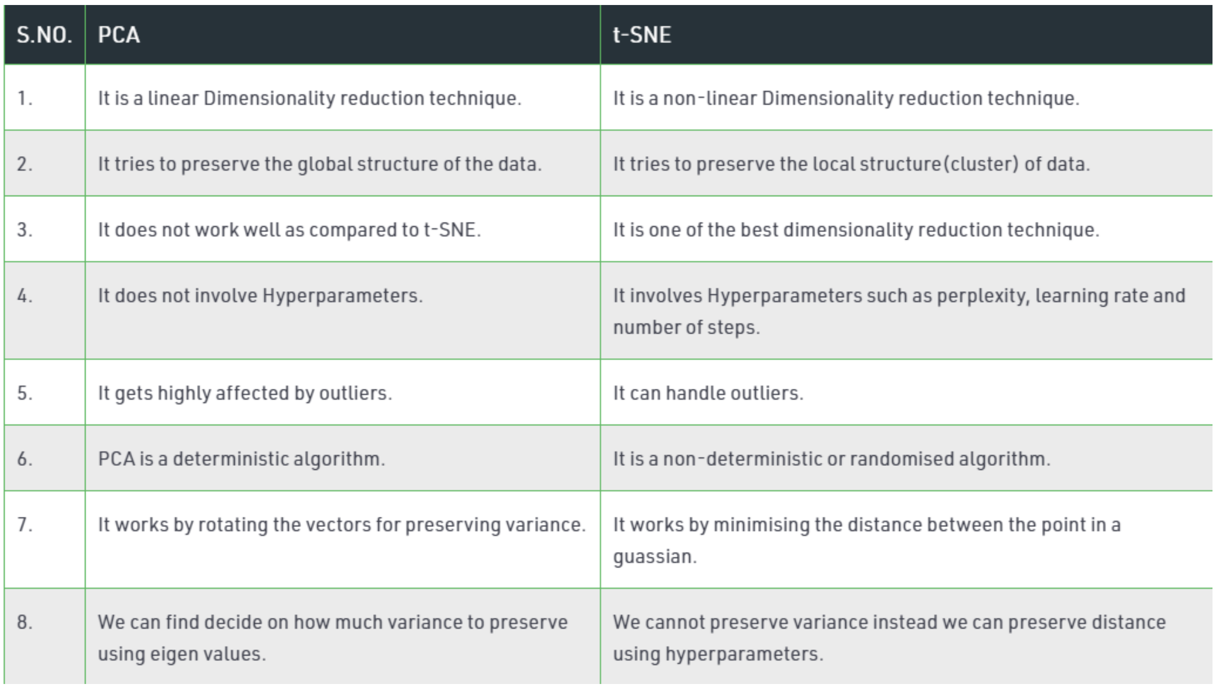
In this section we will transform our data into 2D feature space to visualize them. This will be implemented for all features together plus each feature category separately. The dimensionality reduction methods that will be used are PCA[41] and t-SNE[52].

PCA converts n-dimensions of data into k-dimensions where usually $k << n$. It tries to maintain as much information from the original dataset while transforming the data. PCA algorithm can be described in the following steps:

- Calculate the mean of each feature (column).
- Subtract the mean column value in each column.
- Calculate covariance matrix of centered data.
- Calculate eigen-decomposition of covariance

Then select the first k larger eigenvalues to represent your data in k-dimension.

On the other side t-sne method tries to embed the points from a higher dimension to a lower one trying to preserve the neighborhood of that point, by minimizing the Kullback-Leibler divergence (KL divergence) between the two distributions with respect to the locations of the points in the dataset. Differences between PCA and t-sne can be found in the figure 5.1.



S.NO.	PCA	t-SNE
1.	It is a linear Dimensionality reduction technique.	It is a non-linear Dimensionality reduction technique.
2.	It tries to preserve the global structure of the data.	It tries to preserve the local structure(cluster) of data.
3.	It does not work well as compared to t-SNE.	It is one of the best dimensionality reduction technique.
4.	It does not involve Hyperparameters.	It involves Hyperparameters such as perplexity, learning rate and number of steps.
5.	It gets highly affected by outliers.	It can handle outliers.
6.	PCA is a deterministic algorithm.	It is a non-deterministic or randomised algorithm.
7.	It works by rotating the vectors for preserving variance.	It works by minimising the distance between the point in a guassian.
8.	We can find decide on how much variance to preserve using eigen values.	We cannot preserve variance instead we can preserve distance using hyperparameters.

Figure 5.1: PCA vs t-sne [4]

After giving a brief description of PCA and t-SNE dimensionality reduction methods we will provide the plots for all the feature categories to see the patterns that are created. The red dots show the cases with covid-19 infection and the green dots the cases that the patient is not infected.

2d plot on all features

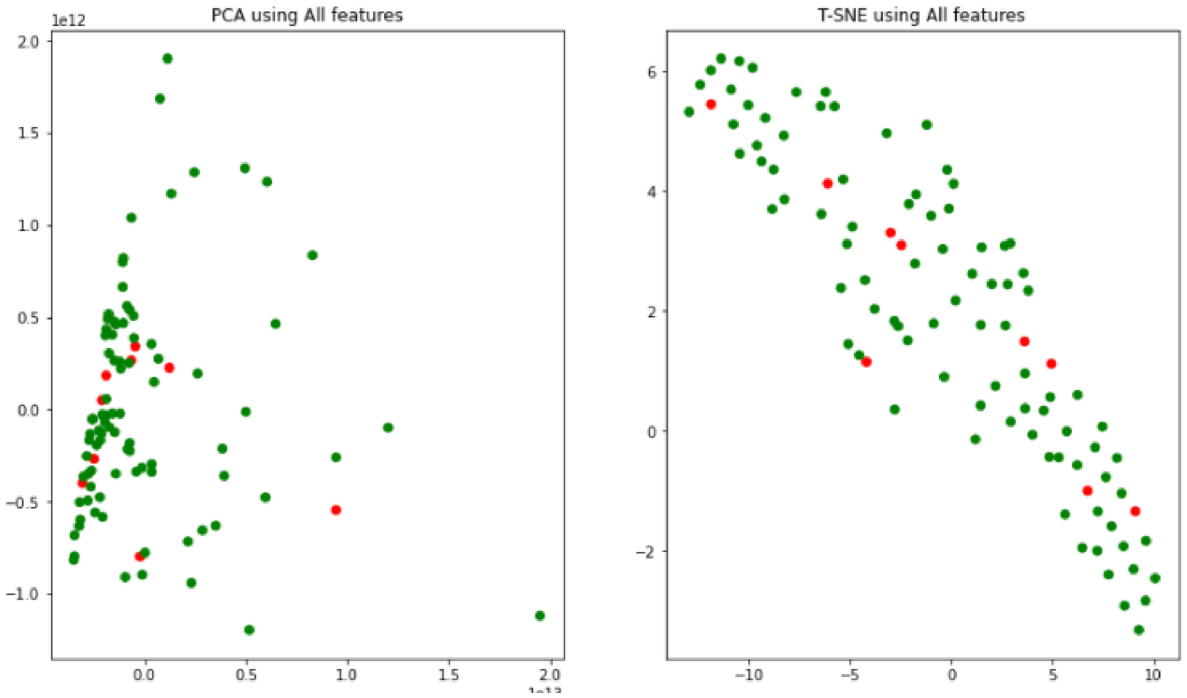


Figure 5.2: Left PCA, Right T-sne. Visualization of All features

2d plot on First Order Statistics features

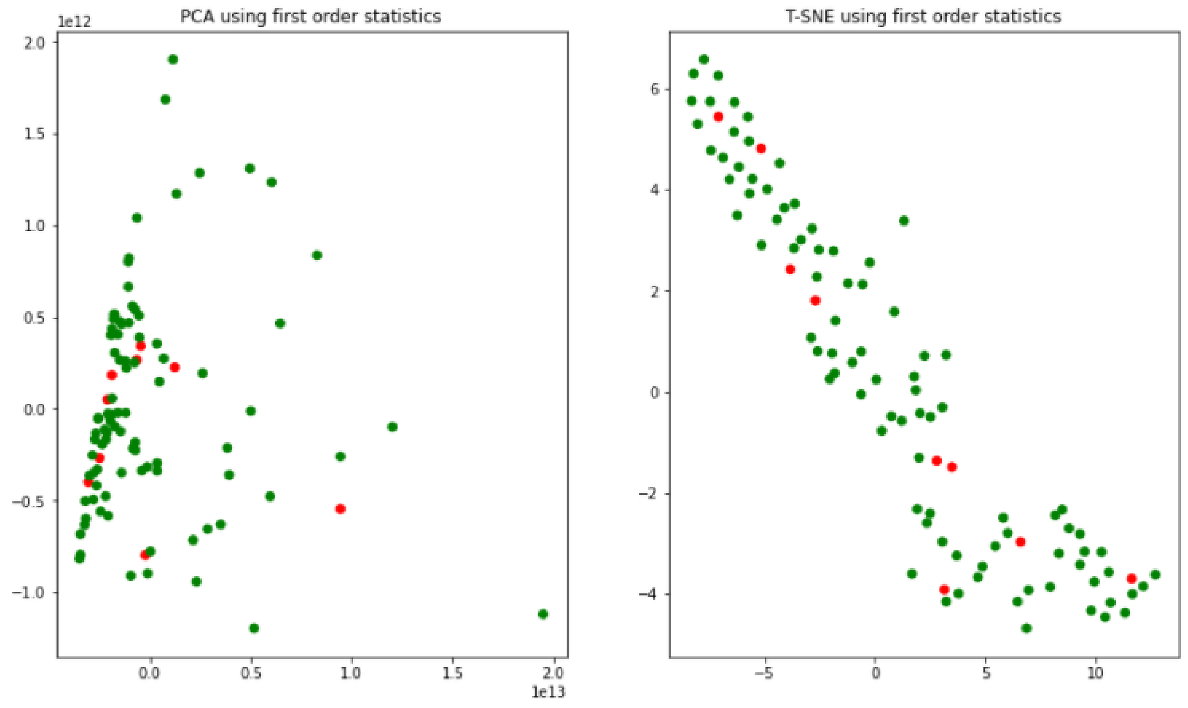


Figure 5.3: Left PCA, Right T-sne. Visualization of First Order Statistics features

2d plot on 3D shaped-base features

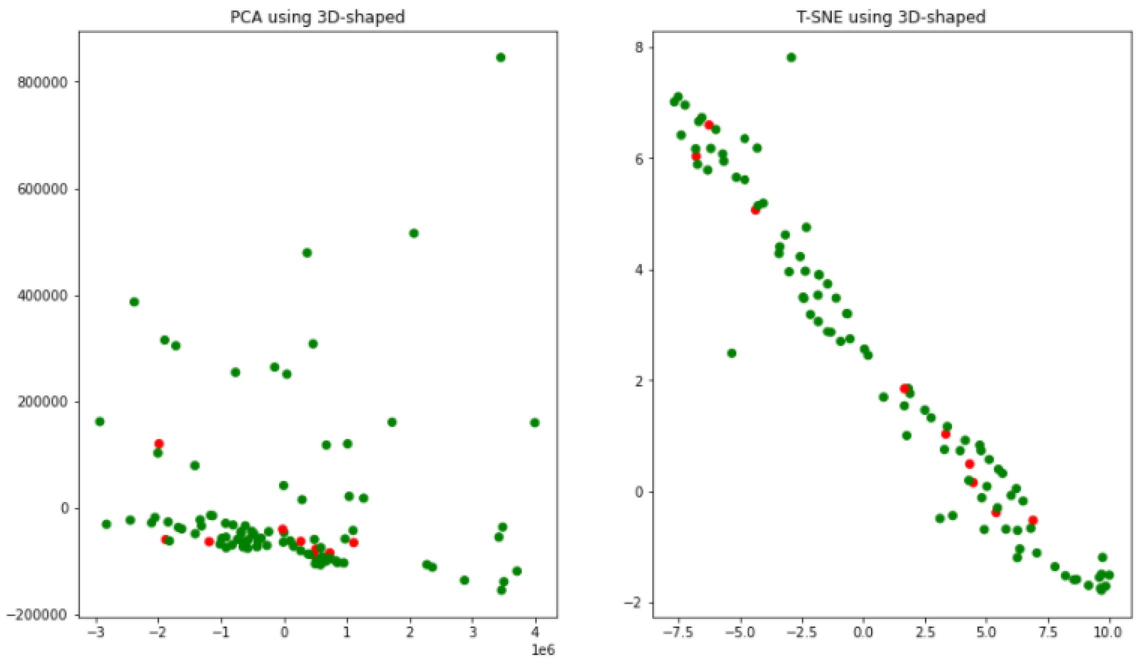


Figure 5.4: Left PCA, Right T-sne. Visualization of 3D shaped features

2d plot on GLCM features

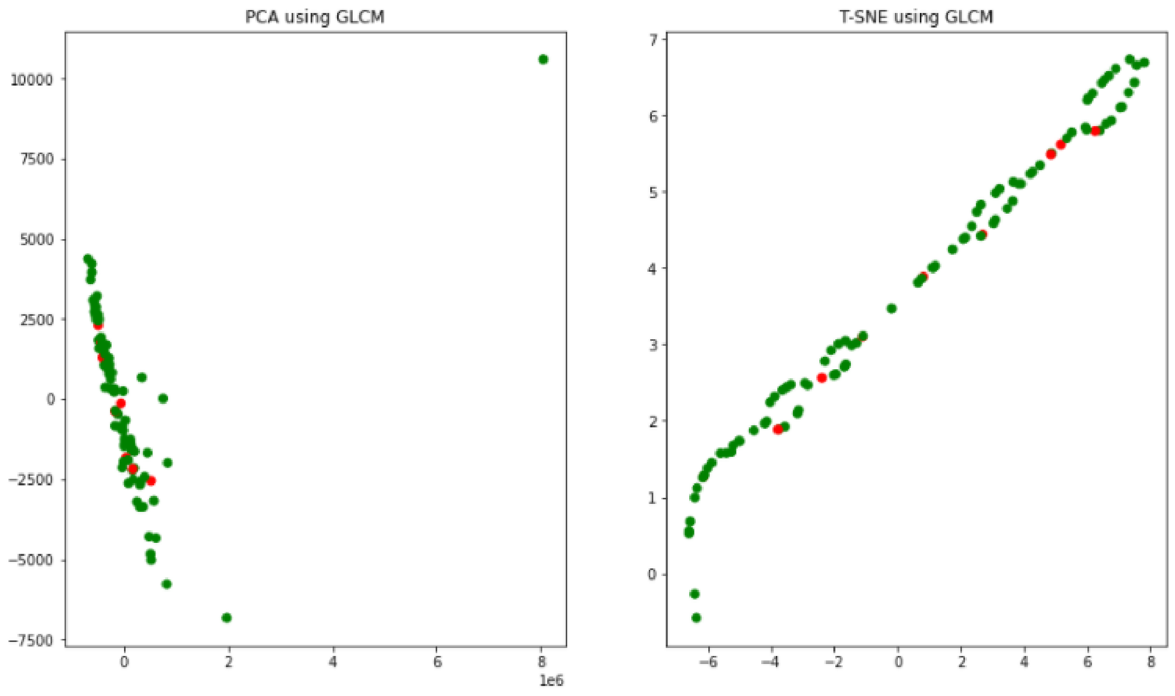


Figure 5.5: Left PCA, Right T-sne. Visualization of GLCM features

2d plot on GLRLM features

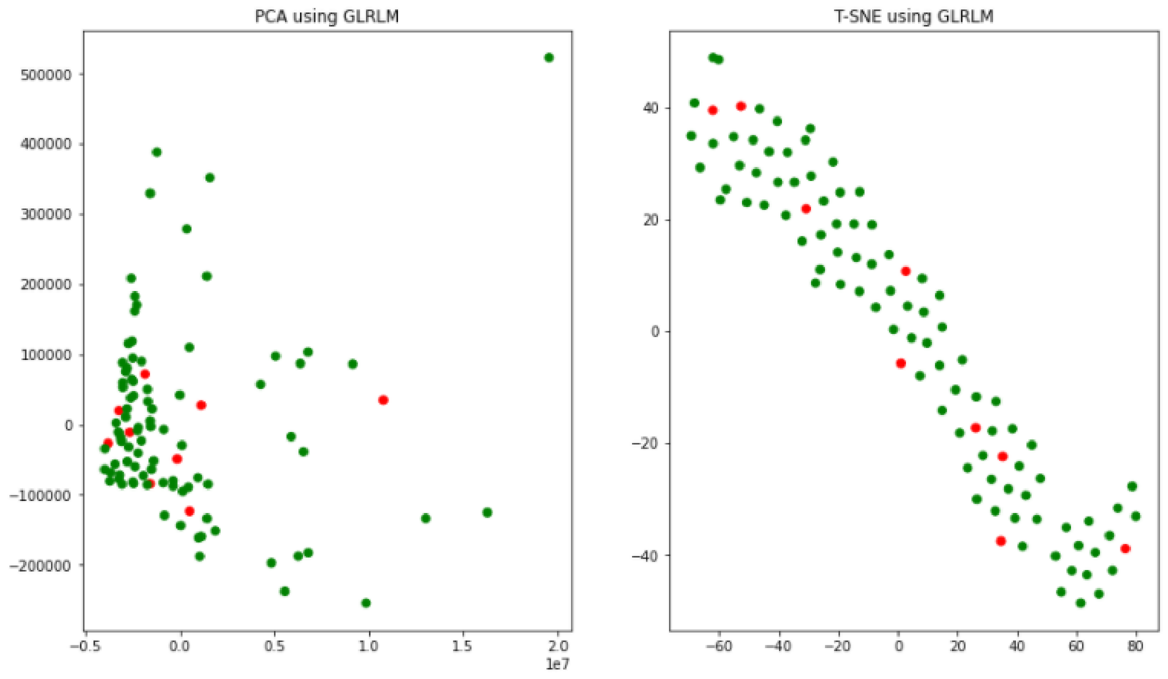


Figure 5.6: Left PCA, Right T-sne. Visualization of GLRLM features

2d plot on GLSZM features

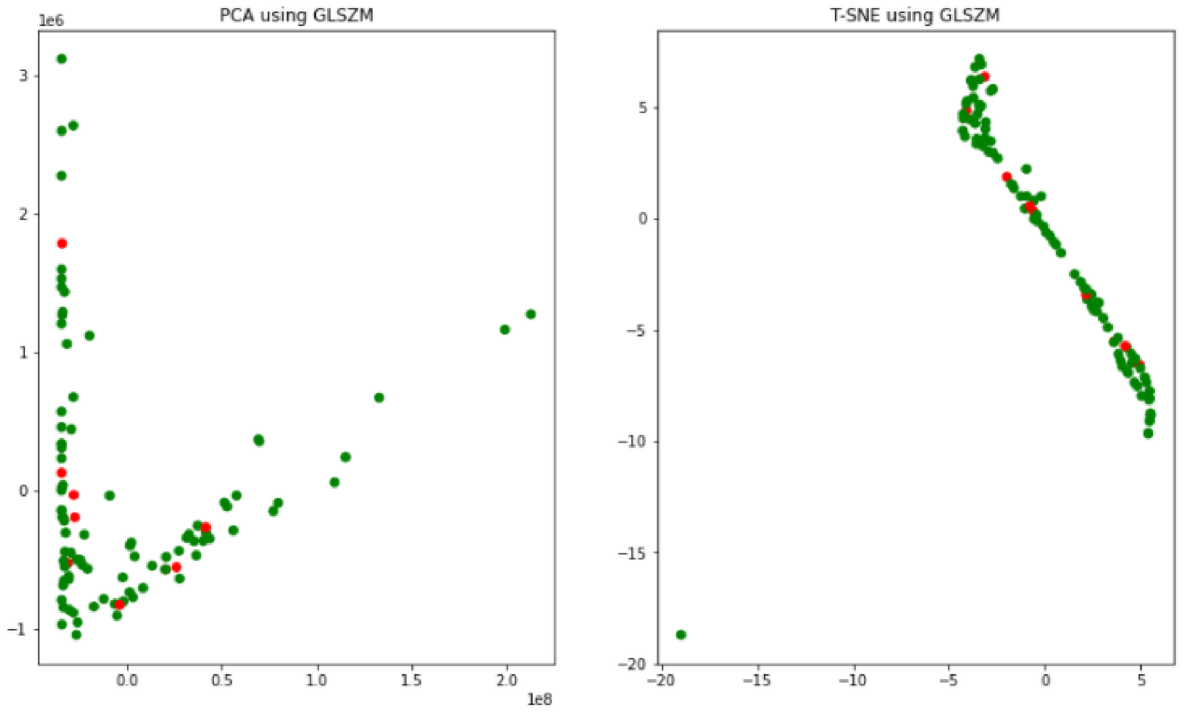


Figure 5.7: Left PCA, Right T-sne. Visualization of GLSZM features

2d plot on NGTDM features

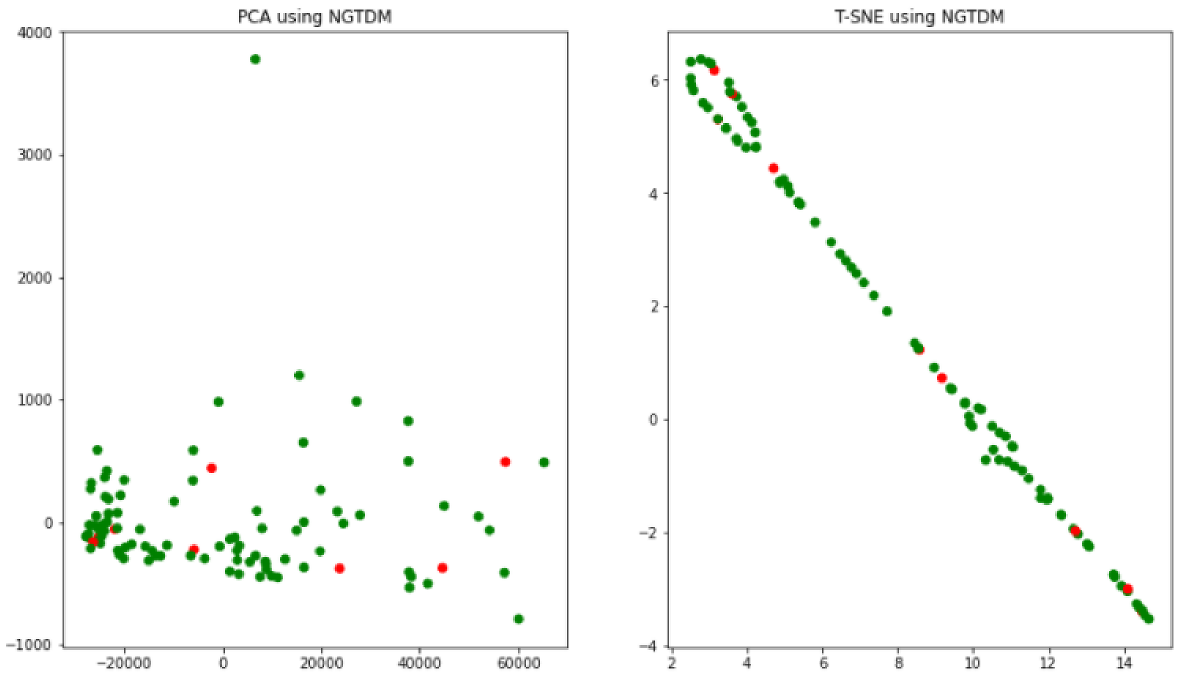


Figure 5.8: Left PCA, Right T-sne. Visualization of NGTDM features

2d plot on GLDM features

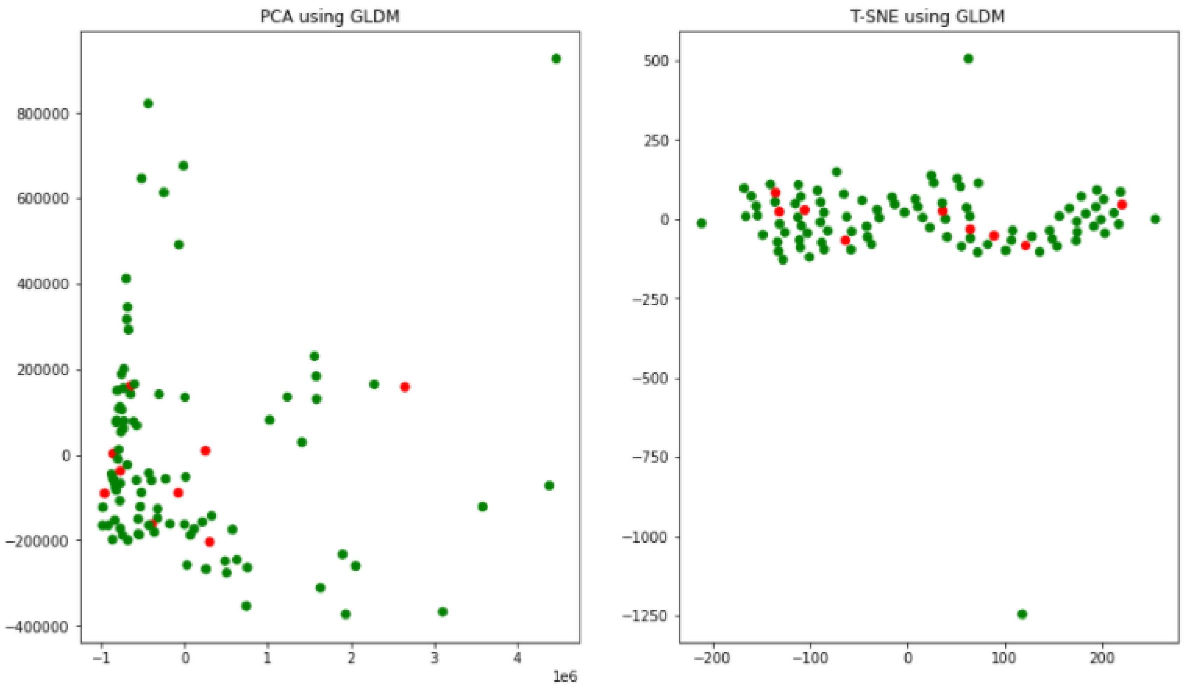


Figure 5.9: Left PCA, Right T-sne. Visualization of GLDM features

4. Classification Results

This section yields the results of our research. The algorithms that were used can be divided in two categories; supervised and unsupervised. As supervised methods the following were used: 1) SVM, 2)K-NN, 3)Random Forest, 4)XgBoost, 5)Ensemble. For unsupervised learning the methods, we experimented with 1)K-Means, 2)LOF, 3)OneClass SVM, 4)Isolation Forest.

The evaluation metrics we are providing are:

- Accuracy
- Auroc score
- Specificity
- Precision
- Recall / Sensitivity

► F1-score

It should be noted that F1, Recall and Precision are the weighted average.

4..1 Supervised Methods

First Order Statistics

Model	Acc.	Precision	Specificity	Auroc	Sensitivity	F1
svm poly	0.80	0.84	0.80	0.62	0.00	0.71
svm linear	0.80	0.84	0.8	0.62	0.00	0.71
svm rbf	0.80	0.84	0.80	0.87	0.00	0.71
decision tree	0.80	0.84	0.80	0.43	0.00	0.71
random forest	0.80	0.84	0.80	0.31	0.00	0.71
xgboost	0.80	0.84	0.80	0.43	0.00	0.71
knn	0.80	0.84	0.80	0.25	0.00	0.71
ensemble	0.80	0.84	0.80	0.31	0.00	0.71

Table 5.1: Supervised Classification Report of First Order Statistics

3D shaped-base features

Model	Acc.	Precision	Specificity	Auroc	Sensitivity	F1
svm poly	0.80	0.84	0.80	0.5	0.00	0.71
svm linear	0.80	0.84	0.80	0.12	0.00	0.71
svm rbf	0.80	0.84	0.80	0.43	0.00	0.71
decision tree	0.80	0.84	0.80	0.31	0.00	0.71
random forest	0.80	0.84	0.80	0.31	0.00	0.71
xgboost	0.80	0.84	0.80	0.43	0.0	0.71
knn	0.80	0.84	0.80	0.25	0.00	0.71
ensemble	0.80	0.84	0.80	0.31	0.00	0.71

Table 5.2: Supervised Classification Report of 3D shaped-base features

GLCM

Model	Acc.	Precision	Specificity	Auroc	Sensitivity	F1
svm poly	0.80	0.84	0.80	0.25	0.00	0.71
svm linear	0.80	0.84	0.80	0.12	0.00	0.71
svm rbf	0.80	0.84	0.80	0.25	0.00	0.71
decision tree	0.80	0.84	0.80	0.43	0.00	0.71
random forest	0.80	0.84	0.80	0.25	0.00	0.71
xgboost	0.80	0.84	0.80	0.43	0.00	0.71
knn	0.80	0.84	0.80	0.18	0.00	0.71
ensemble	0.80	0.84	0.80	0.31	0.00	0.71

Table 5.3: Supervised Classification Report of GLCM

GLRLM

Model	Acc.	Precision	Specificity	Auroc	Sensitivity	F1
svm poly	0.80	0.84	0.80	0.81	0.00	0.71
svm linear	0.80	0.84	0.80	0.30	0.00	0.71
svm rbf	0.80	0.84	0.80	0.87	0.00	0.71
decision tree	0.80	0.84	0.80	0.50	0.00	0.71
random forest	0.80	0.84	0.80	0.31	0.00	0.71
xgboost	0.80	0.84	0.80	0.43	0.00	0.71
knn	0.80	0.84	0.80	0.25	0.00	0.71
ensemble	0.80	0.84	0.80	0.31	0.00	0.71

Table 5.4: Supervised Classification Report of GLRLM

GLSZM

Model	Acc.	Precision	Specificity	Auroc	Sensitivity	F1
svm poly	0.80	0.84	0.80	0.25	0.00	0.71
svm linear	0.80	0.84	0.80	0.12	0.00	0.71
svm rbf	0.80	0.84	0.80	0.25	0.00	0.71
decision tree	0.80	0.84	0.80	0.43	0.00	0.71
random forest	0.80	0.84	0.80	0.25	0.00	0.71
xgboost	0.80	0.84	0.80	0.43	0.0	0.71
knn	0.80	0.84	0.80	0.18	0.00	0.71
ensemble	0.80	0.84	0.80	0.31	0.00	0.71

Table 5.5: Supervised Classification Report of GLSZM

NGTDM

Model	Acc.	Precision	Specificity	Auroc	Sensitivity	F1
svm poly	0.80	0.84	0.80	0.87	0.00	0.71
svm linear	0.80	0.84	0.80	0.32	0.0	0.71
svm rbf	0.80	0.84	0.80	0.25	0.00	0.71
decision tree	0.80	0.84	0.80	0.50	0.00	0.71
random forest	0.80	0.84	0.80	0.25	0.00	0.71
xgboost	0.80	0.84	0.80	0.32	0.00	0.71
knn	0.80	0.84	0.80	0.50	0.00	0.71
ensemble	0.80	0.84	0.80	0.41	0.00	0.71

Table 5.6: Supervised Classification Report of NGTDM

GLDM

Model	Acc.	Precision	Specificity	Auroc	Sensitivity	F1
svm poly	0.00	0.84	0.80	0.87	0.00	0.71
svm linear	0.80	0.84	0.80	0.50	0.00	0.71
svm rbf	0.80	0.84	0.80	0.68	0.00	0.71
decision tree	0.80	0.84	0.80	0.37	0.00	0.71
random forest	0.80	0.84	0.80	0.18	0.00	0.71
xgboost	0.80	0.84	0.80	0.43	0.00	0.71
knn	0.80	0.84	0.80	0.18	0.00	0.71
ensemble	0.80	0.84	0.80	0.31	0.00	0.71

Table 5.7: Supervised Classification Report of GLDM

Using PCA in all features

Model	Acc.	Precision	Specificity	Auroc	Sensitivity	F1
svm poly	0.80	0.84	0.80	0.50	0.00	0.71
svm linear	0.80	0.84	0.80	0.2	0.00	0.71
svm rbf	0.80	0.84	0.80	0.50	0.00	0.71
decision tree	0.80	0.84	0.80	0.18	0.0	0.71
random forest	0.80	0.84	0.80	0.25	0.00	0.71
xgboost	0.80	0.84	0.80	0.18	0.00	0.71
knn	0.80	0.84	0.80	0.43	0.00	0.71
ensemble	0.80	0.84	0.80	0.18	0.00	0.71

Table 5.8: Supervised Classification Report of PCA decomposition

Results from Supervised Methods

It is obvious that using supervised methods without a large number of samples cannot help the classifiers to learn the optimal decision boundary of our problem. Hence all the algorithms yield the same results and could not learn the distribution of the covid-infected class

4..2 Supervised Methods with Oversampling

We will provide the best results from the oversampling method with the best performance which is SMOTE - SVM.

First Order Statistics

Model	Acc.	Precision	Specificity	Auroc	Sensitivity	F1
svm poly	0.70	0.62	0.70	0.50	0.70	0.72
svm linear	0.70	0.62	0.70	0.25	0.70	0.70
svm rbf	0.80	0.90	0.80	0.87	0.80	0.82
decision tree	0.40	0.53	0.40	0.25	0.40	0.46
random forest	0.60	0.60	0.60	0.31	0.60	0.60
xgboost	0.80	0.84	0.80	0.18	0.00	0.71
knn	0.80	0.84	0.80	0.43	0.00	0.71
ensemble	0.80	0.84	0.80	0.18	0.00	0.71

Table 5.9: Supervised Classification Report with oversampling of First Order Statistics

3D shaped-base features

Model	Acc.	Precision	Specificity	Auroc	Sensitivity	F1
svm poly	0.70	0.88	0.70	0.75	0.00	0.73
svm linear	0.60	0.60	0.60	0.18	0.00	0.60
svm rbf	0.70	0.62	0.70	0.62	0.00	0.66
decision tree	0.80	0.84	0.80	0.18	0.00	0.71
random forest	0.80	0.84	0.80	0.18	0.00	0.71
xgboost	0.60	0.60	0.60	0.31	0.60	0.60
knn	0.60	0.72	0.60	0.56	0.60	0.64
ensemble	0.80	0.84	0.80	0.18	0.00	0.71

Table 5.10: Supervised Classification Report with oversampling of 3D shaped-base features

GLCM

Model	Acc.	Precision	Specificity	Auroc	Sensitivity	F1
svm poly	0.50	0.57	0.50	0.37	0.00	0.50
svm linear	0.80	0.84	0.80	0.12	0.00	0.71
svm rbf	0.70	0.88	0.70	0.62	0.70	0.71
decision tree	0.80	0.84	0.80	0.43	0.00	0.71
random forest	0.80	0.84	0.80	0.25	0.00	0.71
xgboost	0.80	0.84	0.80	0.43	0.00	0.71
knn	0.80	0.84	0.80	0.18	0.00	0.71
ensemble	0.80	0.84	0.80	0.31	0.00	0.71

Table 5.11: Supervised Classification Report with oversampling of GLCM

GLRLM

Model	Acc.	Precision	Specificity	Auroc	Sensitivity	F1
svm poly	0.80	0.84	0.80	0.50	0.25	0.71
svm linear	0.80	0.84	0.80	0.20	0.00	0.71
svm rbf	0.80	0.84	0.80	0.50	0.00	0.71
decision tree	0.80	0.84	0.80	0.43	0.25	0.71
random forest	0.80	0.84	0.80	0.25	0.25	0.71
xgboost	0.80	0.84	0.80	0.18	0.25	0.71
knn	0.80	0.84	0.80	0.18	0.50	0.71
ensemble	0.70	0.88	0.70	0.62	0.50	0.71

Table 5.12: Supervised Classification Report with oversampling of GLRLM

GLSZM

Model	Acc.	Precision	Specificity	Auroc	Sensitivity	F1
svm poly	0.80	0.90	0.80	0.87	0.80	0.82
svm linear	0.70	0.62	0.70	0.25	0.70	0.70
svm rbf	0.70	0.62	0.70	0.50	0.70	0.72
decision tree	0.40	0.53	0.40	0.25	0.40	0.46
random forest	0.80	0.84	0.80	0.43	0.00	0.71
xgboost	0.80	0.84	0.80	0.18	0.00	0.71
knn	0.60	0.60	0.60	0.31	0.00	0.60
ensemble	0.80	0.84	0.80	0.18	0.00	0.71

Table 5.13: Supervised Classification Report with oversampling of GLSZM

NGTDM

Model	Acc.	Precision	Specificity	Auroc	Sensitivity	F1
svm poly	0.80	0.90	0.80	0.87	0.80	0.82
svm linear	0.70	0.62	0.70	0.25	0.70	0.70
svm rbf	0.70	0.62	0.70	0.50	0.70	0.72
decision tree	0.40	0.53	0.40	0.25	0.40	0.46
random forest	0.80	0.84	0.80	0.18	0.00	0.71
xgboost	0.80	0.84	0.80	0.43	0.00	0.71
knn	1.00	1.00	1.00	0.91	1.00	1.00
ensemble	0.70	0.62	0.70	0.50	0.70	0.72

Table 5.14: Supervised Classification Report with oversampling of NGTDM

GLDM

Model	Acc.	Precision	Specificity	Auroc	Sensitivity	F1
svm poly	0.80	0.84	0.80	0.68	0.00	0.71
svm linear	0.80	0.84	0.80	0.18	0.00	0.71
svm rbf	0.80	0.84	0.80	0.87	0.00	0.71
decision tree	0.80	0.84	0.80	0.18	0.00	0.71
random forest	0.80	0.84	0.80	0.18	0.00	0.71
xgboost	0.80	0.84	0.80	0.31	0.00	0.71
knn	0.80	0.84	0.80	0.25	0.00	0.71
ensemble	0.80	0.84	0.80	0.43	0.00	0.71

Table 5.15: Supervised Classification Report with oversampling of GLDM

Using PCA in all features

Model	Acc.	Precision	Specificity	Auroc	Sensitivity	F1
svm poly	0.80	0.84	0.80	0.50	0.80	0.71
svm linear	0.80	0.84	0.80	0.20	0.80	0.71
svm rbf	0.80	0.84	0.80	0.50	0.80	0.71
decision tree	0.80	0.84	0.80	0.18	0.00	0.71
random forest	0.80	0.84	0.80	0.25	0.00	0.71
xgboost	0.80	0.84	0.80	0.18	0.00	0.71
knn	0.80	0.84	0.80	0.43	0.00	0.71
ensemble	0.80	0.84	0.80	0.18	0.00	0.71

Table 5.16: Supervised Classification Report with oversampling of PCA decomposition

Best Results using Supervised Methods with Oversampling

The best results were obtained using K-NN on NGTDM with oversampling SMOTE- SVM. This yield a result of 1.00 accuracy, 1.00 precision, 1.00 specificity, 0.90 score of area under the curve, 1.00 sensitivity and F1 equal to 1.00

4..3 Unsupervised Methods

First Order Statistics

Model	Acc.	Precision	Specificity	Auroc	Sensitivity	F1
OneClass SVM	0.70	0.75	0.70	0.62	0.14	0.72
LOF	0.70	0.88	0.81	0.81	0.28	0.73
K-Means	0.60	0.87	0.60	0.75	0.33	0.63
Isolation Forest	0.50	0.86	0.50	0.68	0.40	0.53

Table 5.17: Unsupervised Classification Report of First Order Statistics

3D shaped-base features

Model	Acc.	Precision	Specificity	Auroc	Sensitivity	F1
OneClass SVM	0.40	0.63	0.40	0.43	0.25	0.45
LOF	0.50	0.68	0.50	0.50	0.20	0.55
K-Means	0.50	0.86	0.50	0.68	0.40	0.53
Isolation Forest	0.70	0.75	0.70	0.62	0.14	0.72

Table 5.18: Unsupervised Classification Report of 3D shaped-base features

GLCM

Model	Acc.	Precision	Specificity	Auroc	Sensitivity	F1
OneClass SVM	0.70	0.88	0.70	0.81	0.28	0.73
LOF	0.70	0.88	0.70	0.81	0.28	0.73
K-Means	0.50	0.68	0.50	0.68	0.40	0.53
Isolation Forest	0.70	0.75	0.70	0.62	0.14	0.72

Table 5.19: Unsupervised Classification Report of GLCM

GLRLM

Model	Acc.	Precision	Specificity	Auroc	Sensitivity	F1
OneClass SVM	0.60	0.87	0.60	0.75	1.00	0.63
LOF	0.70	0.88	0.70	0.81	1.00	0.73
K-Means	0.30	0.56	0.30	0.37	0.50	0.34
Isolation Forest	0.50	0.57	0.50	0.31	0.00	0.53

Table 5.20: Unsupervised Classification Report of GLRLM

GLSZM

Model	Acc.	Precision	Specificity	Auroc	Sensitivity	F1
OneClass SVM	0.70	0.88	0.70	0.81	1.0	0.73
LOF	0.60	0.87	0.60	0.75	1.00	0.63
K-Means	0.50	0.68	0.50	0.68	0.40	0.53
Isolation Forest	0.70	0.75	0.70	0.62	0.14	0.72

Table 5.21: Unsupervised Classification Report of GLSZM

NGTDM

Model	Acc.	Precision	Specificity	Auroc	Sensitivity	F1
OneClass SVM	0.90	0.90	0.90	0.90	1.00	0.90
LOF	0.70	0.88	0.70	0.81	1.00	0.73
K-Means	0.30	0.56	0.30	0.37	0.50	0.34
Isolation Forest	0.50	0.57	0.50	0.31	0.00	0.53

Table 5.22: Unsupervised Classification Report of NGTDM

GLDM

Model	Acc.	Precision	Specificity	Auroc	Sensitivity	F1
OneClass SVM	0.60	0.87	0.60	0.75	1.00	0.63
LOF	0.70	0.88	0.70	0.81	1.00	0.73
K-Means	0.70	0.88	0.70	0.81	1.0	0.73
Isolation Forest	0.30	0.56	0.30	0.37	0.50	0.34

Table 5.23: Unsupervised Classification Report of GLDM

Using PCA in all features

Model	Acc.	Precision	Specificity	Auroc	Sensitivity	F1
OneClass SVM	0 .40	0.63	0.40	0.43	0.25	0.45
LOF	0.70	0.88	0.81	0.81	0.28	0.73
K-Means	0.50	0.86	0.50	0.68	0.4	0.53
Isolation Forest	0.70	0.75	0.70	0.62	0.14	0.72

Table 5.24: Unsupervised Classification Report of PCA decomposition

Best Results using Unsupervised Methods

The best results were obtained using OneClass SVM on NGTDM. This yield a result of 0.90 accuracy, 0.90 precision, 0.90 specificity, 0.90 score of area under the curve, 1.00 sensitivity and F1 equal to 0.90.

5. Feature Importance

In this section we are providing plots of feature importance from Random Forest classifier trained on each feature category of the dataset. Our purpose is to find the most dominant features in each category and from this derive a medical impact explanation. In the next paragraphs we provide the feature importance visualizations.

First Order Statistics

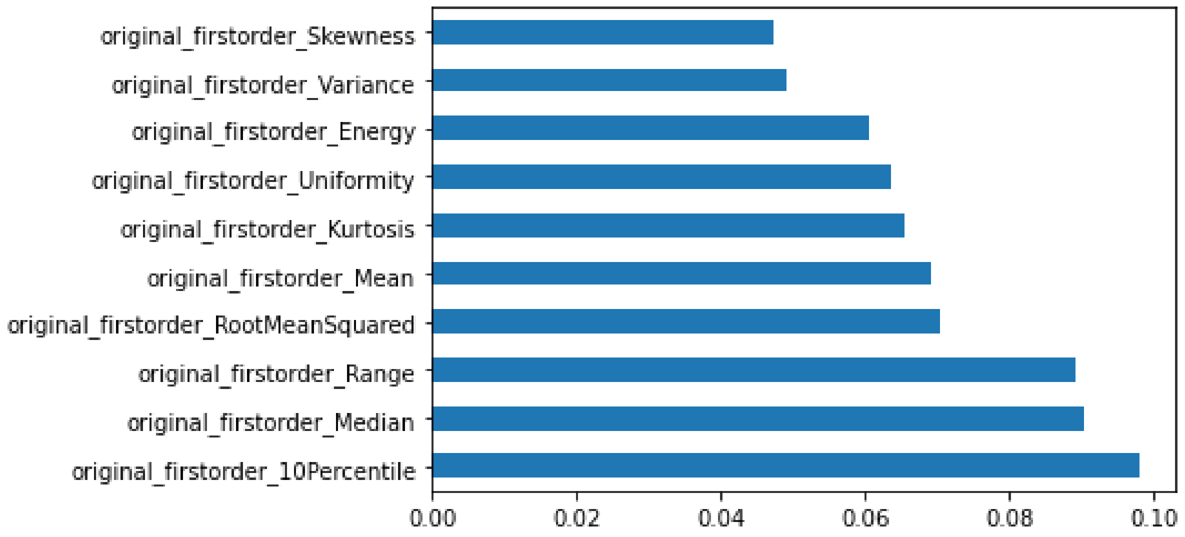


Figure 5.10: Most dominant features in first order statistics category

3D shaped-base features

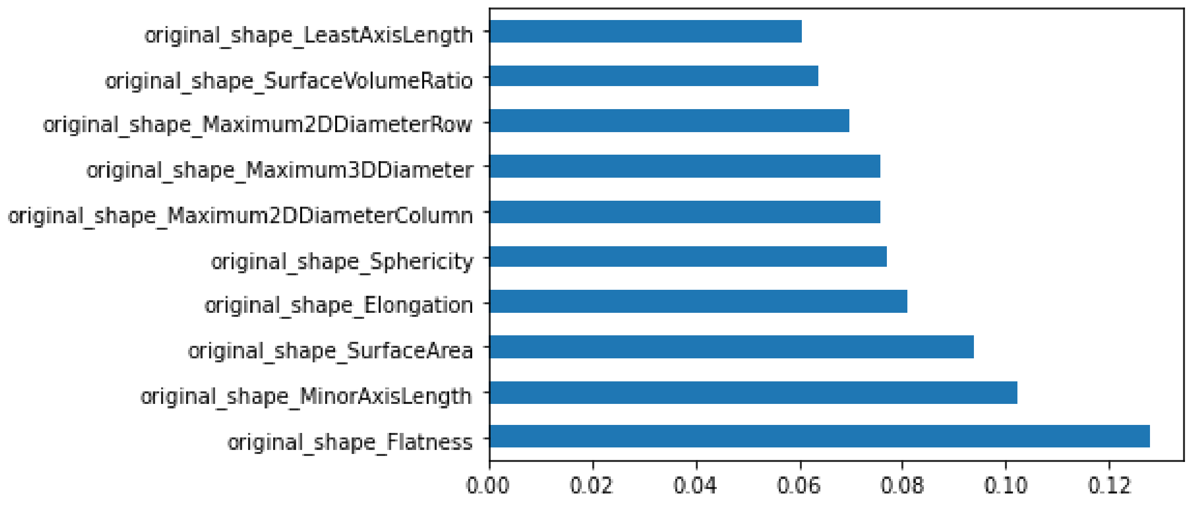


Figure 5.11: Most dominant features in 3d shaped feature category

GLCM

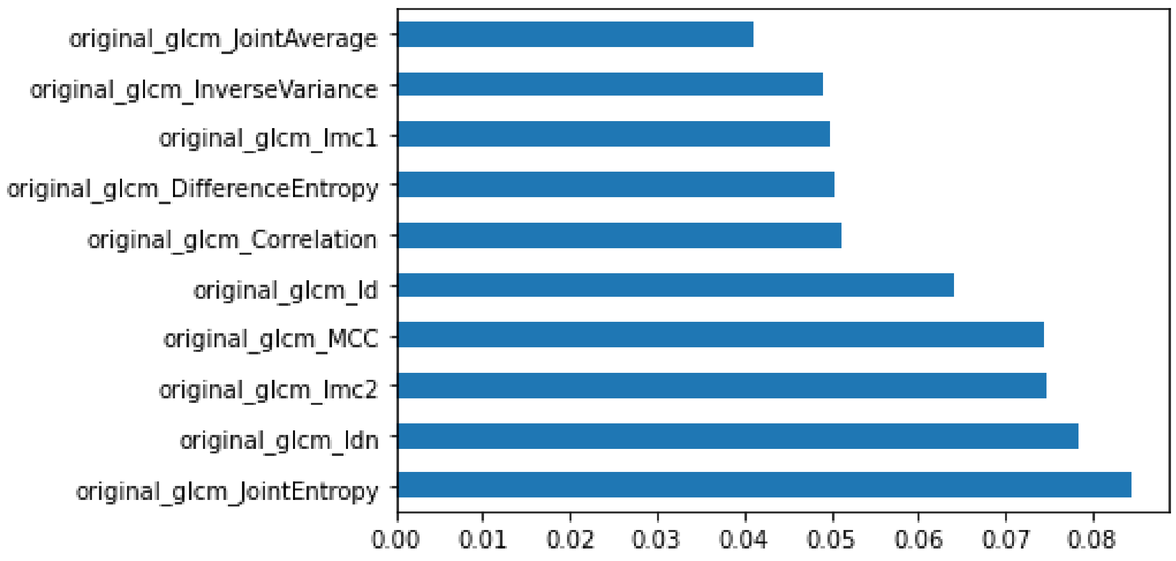


Figure 5.12: Most dominant features in GLCM category

GLRLM

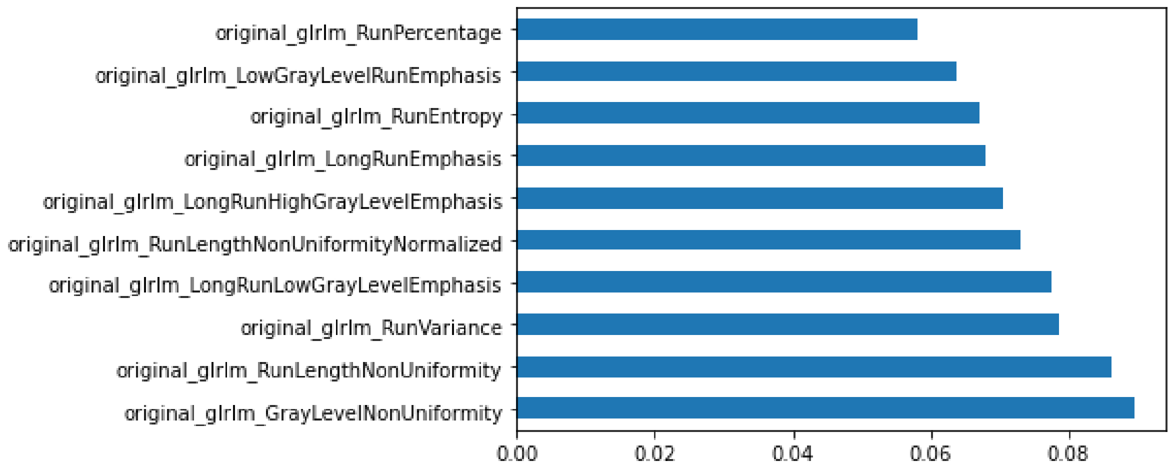


Figure 5.13: Most dominant features in GLRLM category

GLSZM

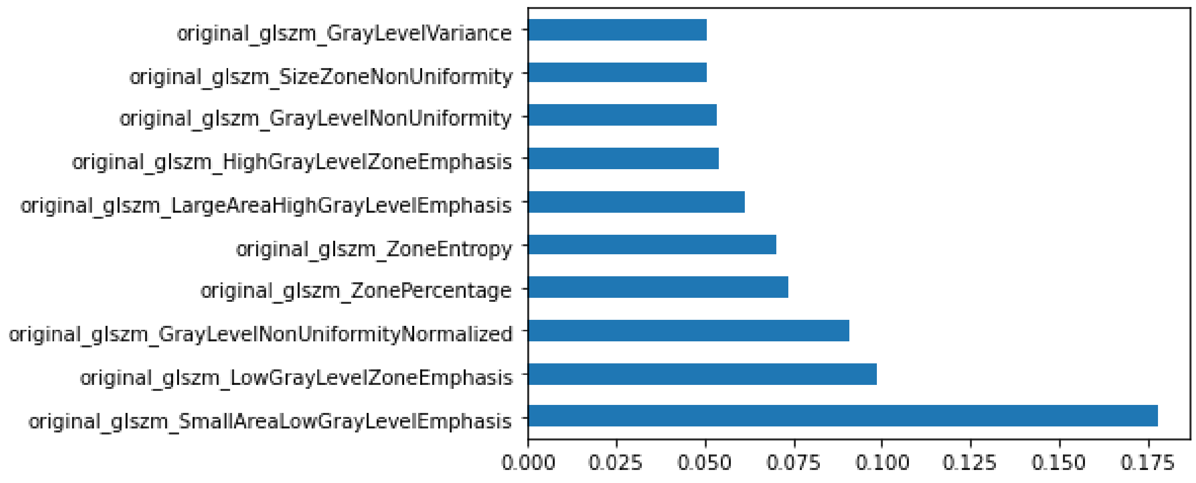


Figure 5.14: Most dominant features in GLSZM category

NGTDM

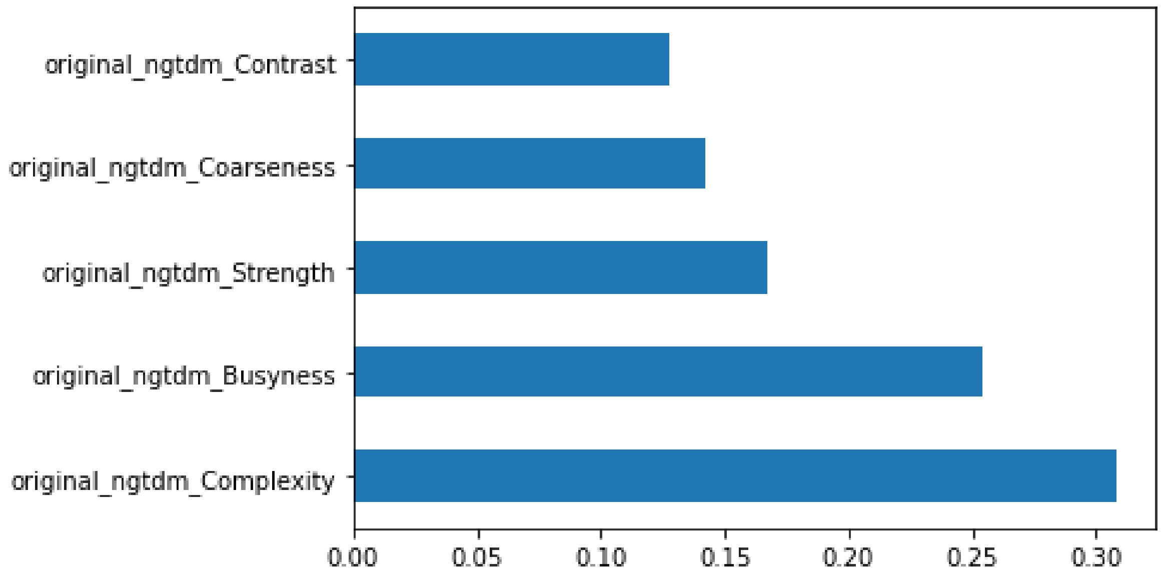


Figure 5.15: Most dominant features in NGTDM category

GLDM

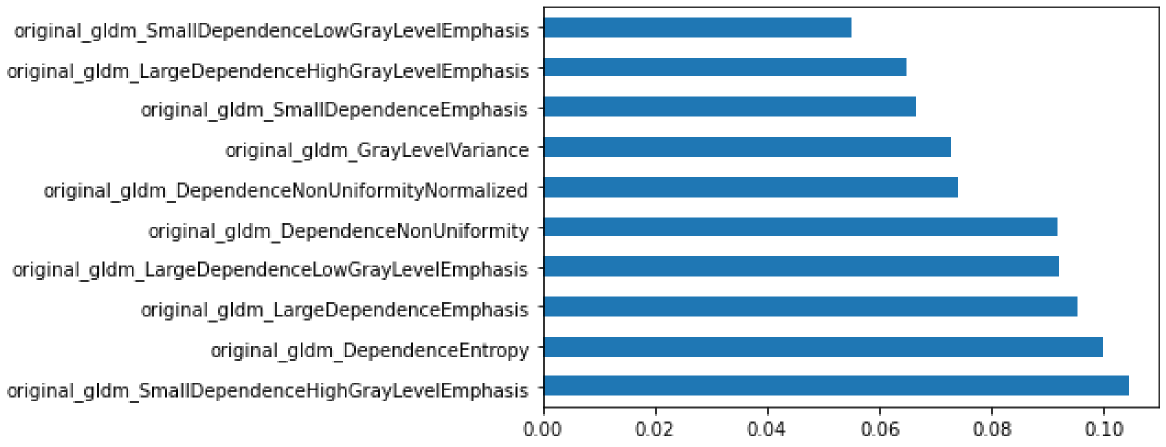


Figure 5.16: Most dominant features in GLDM category

Chapter 6

Summary

In this thesis we used raw ct scans to extract features and pass them to a machine learning model to predict if a patient has been infected by Covid19 or not. Unfortunately, the sample in this research was very small and we cannot be sure about the performance of our model.

A big problem in our experiment was class imbalance. That's why we tested oversampling methods to improve the model performance and tried anomaly detection methods. The best result was obtained from K-NN on NGTDM features with oversampling using SMOTE-SVM. Our experiments without oversampling were yielding the same result the whole time and they could not identify a covid-infected sample. So in that case oversampling helped in finding the other covid-infected cases but also identified covid-cases when it was not sometimes.

On the other hand anomaly detection - unsupervised methods could also identify covid cases but as a cost they also identify many False Positives. False Positives are not a costly mistake such as a False Negative, so we can say that unsupervised methods worked better than supervised without oversampling.

the best outcomes were obtained by the following feature category and algorithm:

- K-NN on NGTDM with oversampling SMOTE- SVM
- OneClass SVM on NGTDM (unsupervised method).

Although, because of the small amount of data we had we cannot conclude that these findings are accurate. In the future, we should increase the data and try this methods with a larger training set and compare the outcomes.

Ideally, in the future it would be very interesting to combine CT scans of both covid19 and pneumonia. By doing this, we can see if our model can distinguish between the two or can use the one label to predict the other.

Also, maybe more CT scan datasets with covid19 should be added from public sources. The problem with these datasets though is that they are not in DICOM raw format. That means, the code will need to adapt to these different files. When using public sources we can create benchmarks to see how it compares with other researches in the same data.

in addition, a new lung segmentation method could be used for the extraction of Range of Interest (RoI) to test if the model performance can be increased with that change. A good option would be U-Net architecture so that it can scale through GPU computation better because of deep learning's architecture.

Finally, the trial of deep learning methods for feature extraction (CNN descriptors) and lung segmentation combined with a supervised machine learning algorithm maybe could work very well too. Although this would need a large amount of data.

Bibliography

[1] Classification: ROC Curve and AUC developers google. https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc/.

[2] COVIDx Dataset a combination of chest x-ray datasets. https://github.com/lindawangg/COVID-Net.

[3] CT image: Pixel vs Voxel an animation created from youtube user manus mongkolsuk. https://www.youtube.com/watch?v=F0OXES7GfQ0.

[4] Difference between PCA VS t-sne comparison of pca and t-sne. https://www.geeksforgeeks.org/difference-between-pca-vs-t-sne/.

[5] ImageNet an image dataset organized according to the wordnet hierarchy. http://www.image-net.org/.

[6] SVM lecture ruye wang. http://fourier.eng.hmc.edu/e161/lectures/svm/node1.html.

[7] Understanding AUC-ROC Curve sarang narkhede . https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5/.

[8] Understanding Stratified Cross Validation the user justin lange posted two images for explaining cross validation and stratified cross validation. https://stats.stackexchange.com/questions/49540/understanding-stratified-cross-validation.

[9] Namita Aggarwal and R. Agrawal. First and second order statistics features for classification of magnetic resonance brain images. *Journal of Signal and Information Processing*, 03, 01 2012.

[10] M. Amadasun and R. King. Textural features corresponding to textural properties. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(5):1264–1274, 1989.

[11] Ioannis Apostolopoulos and Mpesiana Tzani. Covid-19: Automatic detection from x-ray images utilizing transfer learning with convolutional neural networks. *Australasian physical engineering sciences in medicine / supported by the Australasian College of Physical Scientists in Medicine and the Australasian Association of Physical Sciences in Medicine*, 43, 03 2020.

[12] P. A. Asvestas, E. Ventouras, I. Karanasiou, and G. K. Matsopoulos. Classification of event-related potentials associated with response errors in actors. In *2008 8th IEEE International Conference on BioInformatics and BioEngineering*, pages 1–6, 2008.

[13] Mücahid Barstuğan, Umut Özkaya, and Şaban Öztürk. Coronavirus (covid-19) classification using ct images by machine learning methods. 03 2020.

[14] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer-Verlag, Berlin, Heidelberg, 2006.

[15] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer, 1 edition, 2007.

[16] Luca Brunese, Francesco Mercaldo, Alfonso Reginelli, and Antonella Santone. Explainable deep learning for pulmonary disease and coronavirus covid-19 detection from x-rays. *Computer Methods and Programs in Biomedicine*, 196:105608, 06 2020.

[17] Nitesh Chawla, Kevin Bowyer, Lawrence Hall, and W. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *J. Artif. Intell. Res. (JAIR)*, 16:321–357, 06 2002.

[18] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM.

[19] Y. Chen, X. S. Zhou, and T. S. Huang. One-class svm for learning in image retrieval. pages 34–37, January 2001. IEEE International Conference on Image Processing (ICIP) 2001 ; Conference date: 07-10-2001 Through 10-10-2001.

[20] A. Chu, C.M. Sehgal, and J.F. Greenleaf. Use of gray value distribution of run lengths for texture analysis. *Pattern Recognition Letters*, 11(6):415 – 419, 1990.

[21] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

[22] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Trans. Inf. Theory*, 13:21–27, 1967.

[23] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley, New York, 2 edition, 2001.

[24] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2000.

[25] Mary M. Galloway. Texture analysis using gray level run lengths. *Computer Graphics and Image Processing*, 4(2):172 – 179, 1975.

[26] M. Gletsos, S. G. Mougiakakou, G. K. Matsopoulos, K. S. Nikita, A. S. Nikita, and D. Kelekis. A computer-aided diagnostic system to characterize ct focal liver lesions: design and optimization of a neural network classifier. *IEEE Transactions on Information Technology in Biomedicine*, 7(3):153–162, 2003.

[27] Ophir Gozes, Maayan Frid-Adar, Heather Greenspan, Patrick Browning, Huangqi Zhang, Wenbin Ji, Adam Bernheim, and Eliot Siegel. Rapid ai development cycle for the coronavirus (covid-19) pandemic: Initial results for automated detection patient monitoring using deep learning ct image analysis. 03 2020.

[28] Sara H. Kassani, Peyman Kassasni, Mike Wesolowski, Kevin Schneider, and Ralph Deters. Automatic detection of coronavirus disease (covid-19) in x-ray and ct images: A machine learning-based approach. 04 2020.

[29] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-smote: A new over-sampling method in imbalanced data sets learning. volume 3644, pages 878–887, 09 2005.

[30] Haibo He, Yang Bai, Edwardo Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. pages 1322 – 1328, 07 2008.

[31] Morteza Heidari, Seyedehnafiseh Mirniaharikandehei, Abolfazl Zargari, Gopichandh Danala, Yuchen Qiu, and Bin Zheng. Improving the performance of cnn to predict the likelihood of covid-19 using chest x-ray images with preprocessing algorithms. 06 2020.

[32] Dong hui Xu, Arati S. Kurani, Jacob D. Furst, and D. Raicu. Run-length encoding for volumetric texture. 2004.

[33] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 3146–3154. Curran Associates, Inc., 2017.

[34] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69, January 1982.

[35] Alexandra Korda, Pantelis Asvestas, George Matsopoulos, Errikos Ventouras, and Nikolaos Smyrnis. Automatic identification of oculomotor behavior using pattern recognition techniques. *Computers in Biology and Medicine*, 60, 03 2015.

[36] Lin Li, Lixin Qin, Zeguo Xu, Youbing Yin, Xin Wang, Bin Kong, Junjie Bai, Yi Lu, Zhenghan Fang, Qi Song, Kunlin Cao, Daliang Liu, Guisheng Wang, Qizhong Xu, Xisheng Fang, Shiqin Zhang, Juan Xia, and Jun Xia. Artificial intelligence distinguishes covid-19 from community acquired pneumonia on chest ct. *Radiology*, 296:200905, 03 2020.

[37] W. Lorensen and H. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH '87*, 1987.

[38] Tanvir Mahmud, Md Awsafur Rahman, and Shaikh Anowarul Fattah. Covxnet: A multi-dilation convolutional neural network for automatic covid-19 and other pneumonia detection from chest x-ray images with transferable multi-receptive feature optimization. *Computers in Biology and Medicine*, 122:103869, 06 2020.

[39] Guillaume Thibault; Bernard Fertil; Claire Navarro; Sandrine Pereira; Pierre Cau; Nicolas Levy; Jean Sequeira; Jean-Luc Mari. Texture indexes and gray level size zone matrix. application to cell nuclei classification. In *Pattern Recognition and Information Processing (PRIP): 140-145*, 2009.

[40] George K. Matsopoulos, Vassilis Kouloulias, Pantelis Asvestas, Nikolaos Mouravliansky, Kostantinos Delibasis, and Damianos Demetriades. Mitis: a www-based medical system for managing and processing gynecological–obstetrical–radiological data. *Computer Methods and Programs in Biomedicine*, 76(1):53 – 71, 2004.

[41] Sidharth Mishra, Uttam Sarkar, Subhash Taraphder, Sanjoy Datta, Devi Swain, Reshma Saikhom, Sasmita Panda, and Menalsh Laishram. Principal component analysis. *International Journal of Livestock Research*, page 1, 01 2017.

[42] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., USA, 1 edition, 1997.

[43] Mary M. Moya and Don R. Hush. Network constraints and multi-objective optimization for one-class classification. *Neural Networks*, 9(3):463–474, 1996.

[44] Ali Narin, Ceren Kaya, and Ziynet Pamuk. Automatic detection of coronavirus disease (covid-19) using x-ray images and deep convolutional neural networks. 03 2020.

[45] Hien Nguyen, Eric Cooper, and Katsuari Kamei. Borderline over-sampling for imbalanced data classification. *International Journal of Knowledge Engineering and Soft Data Paradigms*, 3:4–21, 04 2011.

[46] Khin Nyein and Anilkumar Gopalakrishnan. Myanmar paper currency recognition using glcm and k-nn. pages 67–72, 01 2016.

[47] J. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 2004.

[48] Robert E Schapire. Explaining adaboost. In *Empirical inference*, pages 37–52. Springer, 2013.

[49] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 618–626, 2017.

[50] Chengjun Sun and William G Wee. Neighboring gray level dependence matrix for texture classification. *Computer Vision, Graphics, and Image Processing*, 23(3):341 – 352, 1983.

[51] Xiaoou Tang. Texture information in run-length matrices. *Image Processing, IEEE Transactions on*, 7:1602 – 1609, 12 1998.

[52] Laurens van der Maaten and Geoffrey Hinton. Viualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 11 2008.

[53] Fedorov A. Parmar C. Hosny A. Aucoin N. Narayan V.-Beets-Tan R. G. H. Fillon-Robin J. C. Pieper S. Aerts H. J. W. L. van Griethuysen, J. J. M. Computational radiomics system to decode the radiographic phenotype. *IEEE Transactions on Systems, Man, and Cybernetics*, 77,21, 2017.

[54] Alexandra Vlachokosta, Pantelis Asvestas, Fani Gkrozou, Lazaros Lavasidis, George Matsopoulos, and Minas Paschopoulos. Classification of hysteroscopical images using texture and vessel descriptors. *Medical biological engineering computing*, 51, 03 2013.

[55] Linda Wang and Alexander Wong. Covid-net: A tailored deep convolutional neural network design for detection of covid-19 cases from chest radiography images. 03 2020.

[56] Ian Young. Image analysis and mathematical morphology, by j. serra. academic press, london, 1982, xviii + 610 p. 90.00. *Cytometry*, 4:184–185, 09 1983.

Appendix A′

Technologies - Software

Below we will provide the information about the software packages and the versions that were used in implementing this thesis.

As programming languages for feature extraction and classification Python 3.6 was used.

For matrix manipulations and data visualizations Numpy 1.19.4 and Pandas 1.1.4 modules were used.

Mainly for feature extraction the module and its version was pyradiomics 3.0.1. For reading DICOM images, pythons module pydicom 2.1.1 was used. Other useful libraries for DICOM images were pynrrd 0.4.2 and SimpleITK 2.0.1.

For machine learning methods scikit-learn 0.23.2 was preferred. Finally, xgboost 1.2.1 was used for gradient boosting trees.

In addition it should be mentioned that mDicom was used for compression of the CT scans. Then the compressed CT scans were uncompressed using imagej in Linux. The uncompressed images were fed into a c++ program that implements lung segmentation using gaussian smoothing, dilation, multiple thresholds and hole filling.

For the execution of the experiments, multiprocessing was used to make parallel the feature extraction on batches of CT scans. Each batch of CT scans was running into a single core. This really improved the computation of the feature extraction.

The system that executed the whole experiment was using 32 GB of RAM and an AMD CPU Ryzen 2GHz with 8 cores and 2 threads each.