UNIVERSITY OF WITWATERSRAND

MASTERS THESIS

Applying Machine Learning To Classify Disease Status For Selected Notifiable Medical Conditions In South Africa.

Student:
Innocent Lino ERONE
Student Number: 1075688

Supervisor(s):
Mr. Michael T. MAPUNDU
Dr. Trevor Graham BELL

A Research Report Submitted to the Faculty of Health Sciences in partial fulfilment of the requirements for the degree of Masters of Science in Epidemiology - Public Health Informatics

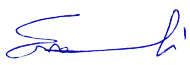
26 October, 2021

Declaration of Authorship

I, Innocent Lino ERONE, declare that this thesis titled, "Applying Machine Learning To Classify Disease Status For Selected Notifiable Medical Conditions In South Africa." and the work presented in it are my own. I confirm that:

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

Signed:



Date: 26 October, 2021

UNIVERSITY OF WITWATERSRAND

Abstract

Faculty of Health Sciences

School of Public Health
Division of Epidemiology and Biostatistics

Masters of Science in Epidemiology - Public Health Informatics

Applying Machine Learning To Classify Disease Status For Selected Notifiable Medical Conditions In South Africa.

by Innocent Lino ERONE

Introduction:
There is a change in disease profiles. Environmental variabilities continue to alter morphological appearances of species necessitating enhancement in diagnostic methods used to detect diseases. The deterministic approaches applied in the current diagnosis methods for Malaria and COVID-19 have presented challenges of low sensitivity and specificity. In this study, we described data structures and disease profiles for Malaria and COVID-19 surveillance data at the National Health Laboratory Services (NHLS), South Africa. We also explored the application of supervised Machine Learning (ML) to classify and predict clinical outcomes for Malaria and COVID-19.

Methods:
The COVID-19 surveillance data comprised of 35,202 observations from a unit dataset. The Malaria data was made up of three files; a demographics file, a laboratory results file and a travel-treatment history file of which 40,094 observations were deduced. These datasets were divided into two portions, 75% for model specification and the 25% designated as out-of-sample testing. We compared three supervised ML classifiers: Support Vector Machine (SVM), the K-Nearest Neighbor (KNN), Random Forests (RF) with their variant novelty approaches Isolation Forest (iForest) and One-Class Support Vector Machines (OCSVM) to predict clinical outcomes for Malaria and COVID-19. To account for severe label imbalances, the data with majority class labels was under-sampled to obtain an equal class balance in the target. Novelty detection approaches with iForest and One-Class Support Vector Machines (OCSVM) were also used in classifying and predicting Malaria and COVID-19 clinical outcomes.

Results:
Malaria surveillance data was characterized by large proportions of missing data for demographic, syndromic and environmental characteristics. Though complete, compared to Malaria, COVID-19 surveillance data did not follow tidy-data principles. In evaluating classifier predictive power using out-of-sample data with equal representation of clinical outcomes, RF yielded the best predictive power with Area Under Curve (AUC) scores (98%) from Malaria out-of-sample data accounting for distribution weight of clinical outcome. Though not comparable to scores from Malaria data, the RF still scored better than the SVM and KNN classifiers from out-of-sample evaluation over COVID-19 data. Generally, lower classifier performance was observed across all models when subjected to COVID-19 out-of-sample data, where the KNN classifier registered the highest number of false-positive results. There were significantly higher numbers of False-Negative predictions with the SVM classifiers compared to the RF and KNN. However, the RF performed slightly better in predicting True-Negative observations. By categorizing data with minority clinical outcome representation as outliers, OCSVM predicted more negative observation compared to the iForest.

Conclusions:
This study showed the impact of data quality in disease surveillance with respect to predictive modeling for Malaria and COVID-19 medical conditions. The data were characterized by large proportions of incompleteness. Individual demographic characteristics, reported and recorded signs and symptoms among other attributes that hold vital information for syndromic disease surveillance were lacking. While supervised ML classifiers performed well with Malaria out-of-sample data, the same methods produced suboptimal results with similar surveillance COVID-19 data. Future studies could explore unsupervised ML approaches on the same surveillance data.

Acknowledgements

Firstly, I would like to thank my academic supervisors Mr. Michael T. MAPUNDU and Dr. Trevor Graham BELL for your immense support and insight throughout the project. Your stimulating discussions informed the direction of this research. Furthermore, I wish to express my gratitude towards Brenda Nansereko whose thorough peer-review helped me write a better thesis. Special thanks to the African Union Center for Disease Control, your support is incomparable. Finally, I would like to acknowledge the academic research team at the National Institute for Communicable Diseases - South Africa, you made this research possible!

Innocent Lino ERONE
26 October, 2021

Contents

List of Figures

List of Tables

List of Abbreviations

AUC	Area Under Curve
CV	Cross Validation
DoH	Department of Health
ICD	International Classification of Disease
iForest	Isolation Forest
KNN	K-Nearest Neighbor
MCC	Mathews Correlation Coefficient
ML	Machine Learning
NHLS	National Health Laboratory Services
NICD	National Institute for Communicable Diseases
NMC	Notifiable Medical Conditions
OCSVM	One-Class Support Vector Machines
PCA	Principal Component Analysis
PPV	Positive Predictive Value
PR	Precision-Recall
RDT	Rapid Diagnostic Test
RF	Random Forests
RIM	Rule Interestingness Measures
ROC	Receiver Operator Characteristic
SVC	Support Vector Classifier
SVM	Support Vector Machine
TPR	True Positive Rate
WHO	World Health Organization

1 Introduction

1.1 Epidemiology

Although there is a global decline in incident cases[1], Malaria is still the most common disease in Africa and globally [1] with the World Health Organization (WHO) estimating 405,000 deaths from 228 million clinical episodes in 2018 alone [2]. There have been global efforts to accelerate the elimination of Malaria through improved diagnostic testing and treatment especially in the WHO low and medium-income countries which have reduced the incidence rates of Malaria [3]. However, the rates of decrease of Malaria incidence and mortality are still low in countries with low resourced health systems and limited ability for system improvements [3]. The Global Technical Strategy for Malaria (2016 to 2030) adopted by the World Health Assembly in 2015 aims at reducing Malaria-attributable cases and deaths by ninety percent by 2030 through integrating active surveillance by interventions [1]. Surveillance systems are effective in the elimination of parasites through tracking Malaria transmission and pathways which focus on diagnosis, treatment, and prevention resources [4].

COVID-19 is a highly transmitted disease that was first reported in Wuhan China in December 2019. This disease is caused by a zoonotic virus which was named severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) [5]. Coronaviruses belong to the Coronaviridae family under the Coronavirinae subfamily, and they have been known to cause several other infections in humans since the 1960s [6]. Globally, the disease has imposed a great public health burden in many countries across the world due to its high transmission rate, with the WHO reporting over 113,000,000 cumulative confirmed cases and over 2,500,00 cumulative deaths as of March 2021 [7, 8]. Recent statistics indicate Africa is one of the least affected continents with over 2.8 million cumulative confirmed cases and over 70,000 deaths, contributing 3% of the global cumulative COVID-19 related deaths [7] with South Africa as the COVID-19 epicenter in Africa [9].

As part of the road map to eliminate diseases such as Malaria and COVID 19, countries need to ensure improved testing and follow-up on infection rates [4]. As Basu and Sahi

[1] From 71 to 57 cases per 1000 population at risk between 2010 and 2018

[10] argue, early diagnosis and treatment reduce mortality rates and morbidity. Over the past decades, there has been an evolution in the diagnostic testing techniques for Malaria disease. Though widely adopted, diagnostic systems such as light microscopy and Rapid Diagnostic Tests (RDTs) are dependent on biomarkers. However, RDTs and microscopy are reported to have low sensitivity[2] and specificity[3] for Malaria [11]. However, research still highlights significant challenges in the diagnosis of Malaria. Acting as parasite reservoirs [11], asymptomatic individuals fuel resurgence of the disease years after reported treatment.

The current diagnosis methods for COVID 19 are laboratory-based and rely on biomarkers. Nevertheless, challenges associated with the diagnosis techniques exist [12]. These include shortages of test kits and long waiting times for results among others. Much as research continues to support the association of epidemiological profiles of disease with environmental variability, the current COVID-19 diagnostic techniques do not account for factors such as demographic information in diagnostic procedures. On the other hand, diagnostic techniques that assess the patient's signs[4] and symptoms[5] are reported to have poor diagnostic properties especially among asymptomatic patients [13].

Subjective diagnosis of disease from symptoms is a vital component of disease surveillance. Therefore, to aid clinical diagnosis, there is research and innovation in the use of self-learning approaches to cope with these changing patterns. Commonly known as Machine Learning (ML), self-learning has been used to predict previously complex conditions like cardiovascular diseases [14], obstructive pulmonary disease [15] among others. These stochastic methodologies use a wide array of features to identify hidden patterns in the data to predict disease outcomes. For example, by incorporating low-level features such as texture to digitized human blood smear from slides, Khan et al. [16] used K-means clustering to identify Malaria parasites with 95% accuracy. In the same way, using computer vision, Molina et al. [17] used unsupervised ML to identify Malaria parasites. The approach yielded a sensitivity score of 100% with specificity at 90%. In a similar approach, a 2018 systematic review by Poostchi et al. [18], the authors suggested that a well-defined predictor should incorporate several factors such as characteristic of the microscope, type of staining, slide preparation to image analysis in Malaria predictive approaches.

[2]The ability of a test to identify people with a disease usually expressed as a proportion

[3]The ability of a test to correctly identify people without a disease; the proportion of negatives that are correctly identified.

[4]Any objective evidence of disease

[5]Subjective evidence of disease

Disease profiles are constantly changing and environmental variability is altering the accepted morphological appearances of species [19]. Therefore, for successful control and eventual elimination of Malaria and COVID-19, more sensitive detection methodologies that incorporate symptomatic information with laboratory markers are needed. As of the year 2020, Malaria and COVID-19 accounted for the highest volumes of surveillance data at the South Africa National Institute for Communicable Diseases (NICD)[6]. Nevertheless, the institution uses deterministic approaches to predict classes/labels for Notifiable Medical Conditions (NMC). This strategy implements pre-determined rule-sets upon selected laboratory scores; an approach that is sensitive logically and rapidly becomes complex with the addition of features. Therefore, in this research, we explore stochastic discriminative approaches as an alternative to deterministic methods in predicting disease labels for Malaria and COVID-19. We constructed classifiers to accurately discriminate *Positive* and *Negative* Malaria and COVID-19 cases from demographic, symptom, and laboratory data. These classifiers could then be used for discriminative analysis to segregate probable cases of Malaria and COVID-19 with new data.

1.2 Statement of the Problem

Growing data dimensionality is a real threat to deterministic/rule-set[7] classifiers. Yet to enhance clinical diagnosis, it is necessary to look at a broad spectrum of data points/features not only from the laboratory but also from non-laboratory markers such as symptoms. As with most legacy systems, self-learning[8] in rule-set classifiers is absent. In light of changing rule-sets, continuous learning and domain expertise become mandatory to keep such classifiers relevant. To cope with changes in data structures, self-learning approaches in ML become necessary.
We chose COVID-19 and Malaria for this research because of the high volume of data readily available to experiment with this ML models.

1.3 Research Question

The questions of interest are:

1. *Are supervised ML techniques better than rule-set approaches in the classification of Malaria and COVID-19 at the NICD?*

[6] A national public health institute located in Johannesburg, South Africa
[7] A set of human-crafted conditions to trigger a decision or choice. In computer science, knowledge is presented and handled as logical rules implemented by an inference engine.
[8] The ability to recognize patterns, learn from data, and become more intelligent over time

2. *What can be said of the current deterministic approaches that categorize Malaria and COVID-19 in respect to current surveillance data profiles?*

To answer these questions, in this research, we explore stochastic self-learning classifiers using supervised ML prediction techniques in predicting disease status using Malaria and COVID-19 surveillance data from the NICD. We also perform a comparative analysis of current deterministic approaches and how it performs compared to novel ML classification approaches to achieve a maximum reward. Therefore, the work of this project aims to:

1. Describe the current surveillance data structures and profiles for Malaria and COVID-19 in South Africa.
2. Identify optimal ML algorithms that can be utilized to classify and predict Malaria and COVID-19 clinical outcomes from available data structures.
3. Evaluate the performance of selected ML algorithms against the conventional rule-set methods used to categorize disease status for Malaria and COVID-19.

1.4 Thesis Structure

The rest of this report is organized as follows: In Chapter 2, we explore general concepts in Malaria and COVID-19 NMCs from case identification, clinical manifestations to treatment. We also explore the current NMC surveillance approach at NICD. In Chapter 3, we describe the various materials and methods used in this research with empirical data (results) presented in Chapter 4. Lastly, a discussion of the research findings is presented in Chapter 5 along with conclusions from the research in Chapter 6.

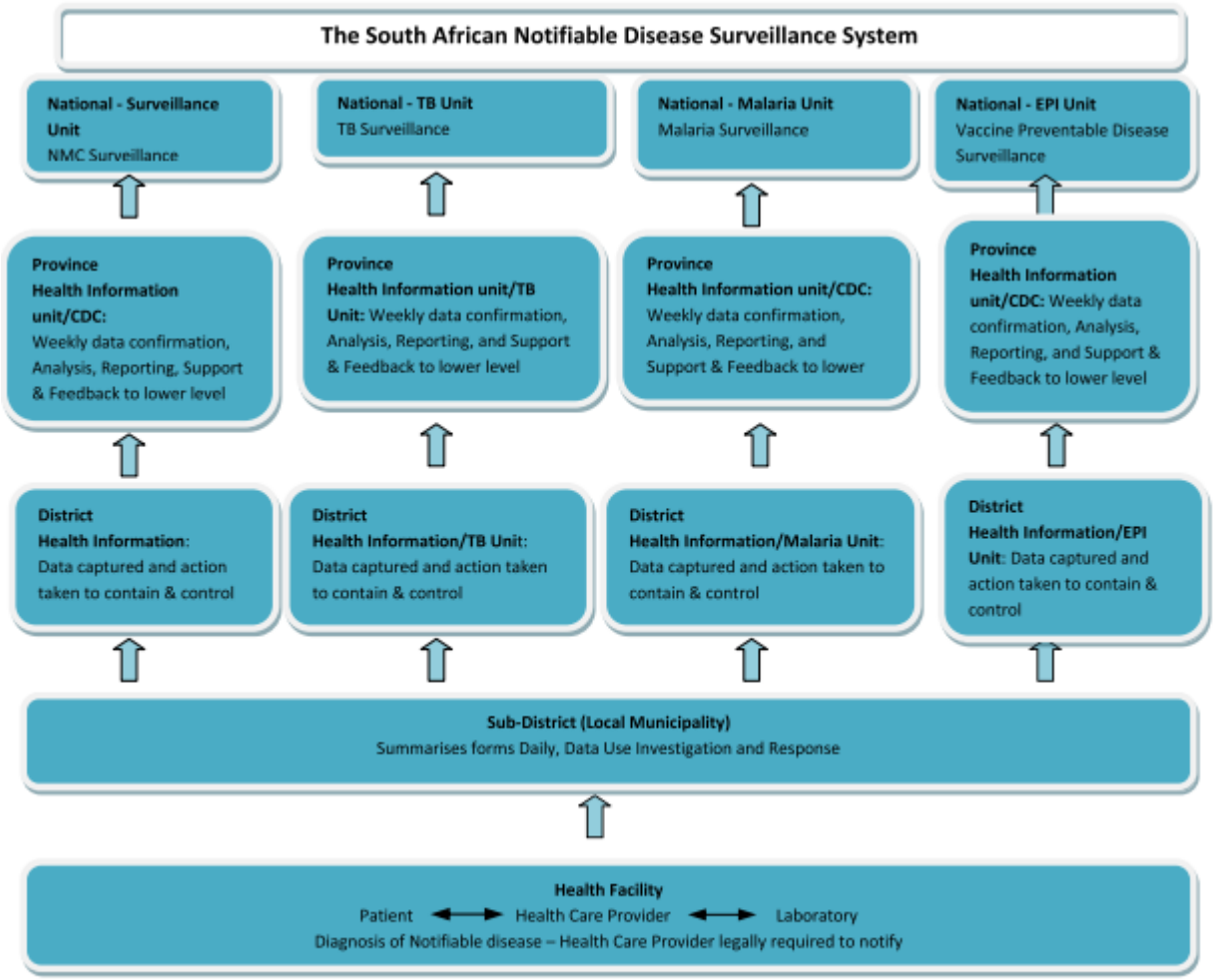
2 Theoretical Background

In this section, firstly we briefly look at Malaria, in the context of disease manifestation and management while highlighting ideal parameters to aid clinical diagnosis. Secondly, we give an overview of the NMC surveillance process as conducted by the NICD. Herein, we also covers specific key concepts and constructs that inform the direction of this research. Some of the questions addressed include; what approaches are available to address the objectives stated in Chapter 1 and what classifiers are available for this task and how to determine optimal classifiers. To answer these questions, we begin by exploring classification approaches relating to this research and what alternatives are available to answer these questions. The last section explains the performance metrics that are available to aid model selection.

2.1 Overview of NMC Surveillance in South Africa

Globally, Monitoring and Evaluation (M&E) programs are used to collect health-related data [20]. These data are then used to track progress towards targets, assess the impact of current health interventions and the WHO goals of morbidity control and elimination [21, 22]. NMC surveillance is a vital process in providing necessary information to timely and accurately detect public health threats. The National Department of Health (DoH) of South Africa defines NMC as diseases that are of public health importance [23] because of the risks they pose. As illustrated in Figure 2.1, this reporting follows an upward cascade starting at the Health establishment level, Sub-District, or District level then to the national system.

It is a legal obligation for all health practitioners to report diseases classified as NMC to the DoH. At the NICD, NMC reporting timelines vary depending on severity. The National Guidelines for the Treatment of Malaria in South Africa 2019 require all Category-1 conditions be reported within 24 hours of first diagnosis irrespective of laboratory confirmation [24]. For Category-2, within 7 days of receipt of laboratory confirmation; within 7 days of diagnosis for Category-3 [23] and up to one month of diagnosis for Category-4.



NMC = Notifiable Medical Condition, EPI = Expanded Programme on Immunisation, TB = Tuberculosis, CDC = Communicable Disease Control
Source: National Department of Health Surveillance Unit (http://www.nmc.gov.za/Docs/Notifiable_medical_conditions.pdf), National Department of Health (TB Unit, Malaria Unit and EPI Unit)

Figure 2.1: NMC Reporting Cascade

2.2 Disease Manifestation and Management

2.2.1 Overview

In South Africa, Malaria is regarded as a category-1 NMC and therefore, must be reported within 24 hours of first diagnosis irrespective of laboratory confirmation [23]. With Light Microscopy using Giemsa-stained thick/thin blood smears as the yardstick [25, 26], several diagnostic methods have been adopted to support these global efforts to reduce and eventually eliminate Malaria [27, 28]. However, not all standards are ideal especially in Malaria-endemic areas and affordable Point-of-Care diagnostics (Rapid Diagnostic Tests) have been reported to have differing sensitivity and specificity [29, 30, 31]. It is argued that in South Africa, malaria is mainly transmitted along the border areas with some parts of South Africa's nine provinces (Limpopo, Mpumalanga and KwaZulu-Natal) endemic for malaria[32]. In the Figure 2.2 is an illustration of the disease severity in South Africa.

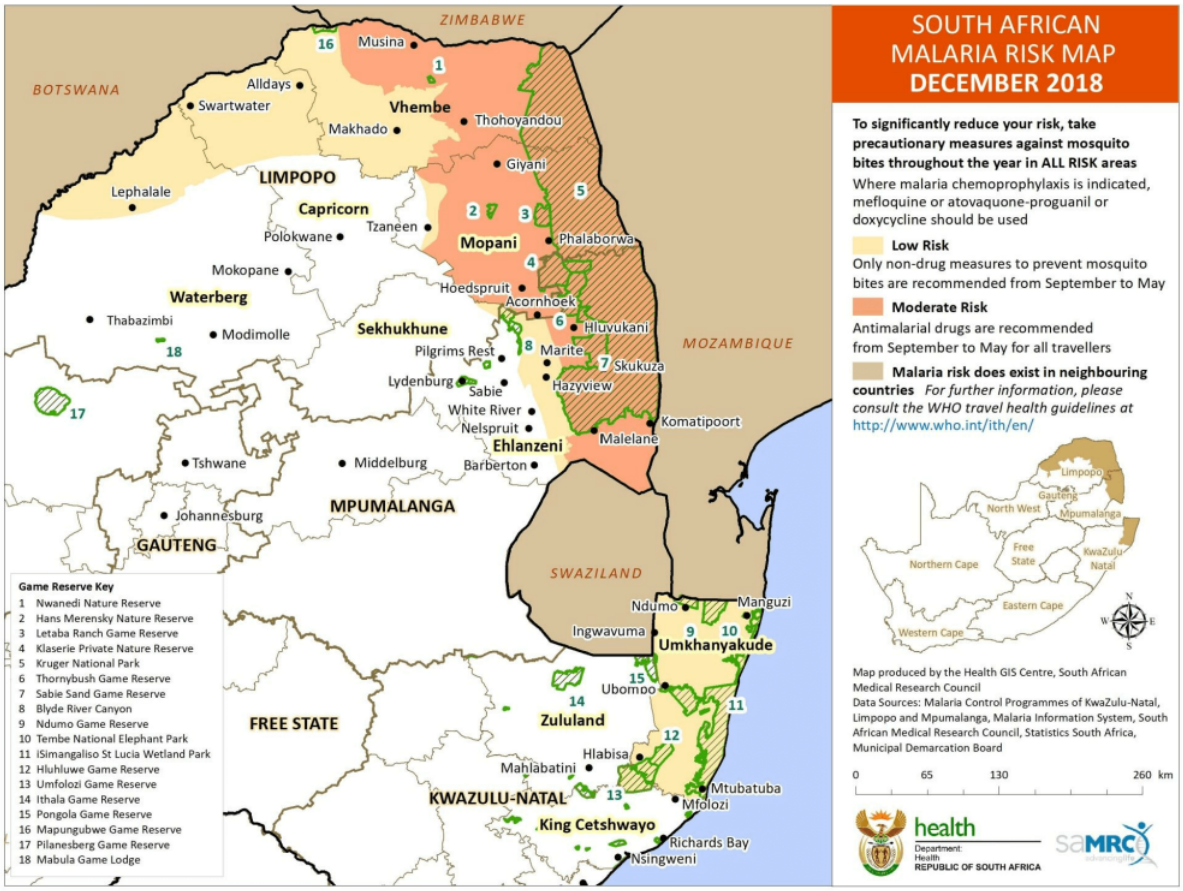


Figure 2.2: SA Malaria Risk Map December 2018. Image credit: DoH SA

Diagnosis of COVID-19, a Category-1 NMC is based on biomarkers that are related to the organisms that cause disease. The United States Center for Disease Control and Prevention recommends two types of tests; a viral test that detects current infection and an antibody test that detects the previous infection. The approved assays used for testing detect either COVID-19 nucleic acid or antigen in the upper or lower respiratory specimen which are either the oral or nasal swabs to determine whether an individual has COVID-19 or not [33].

2.2.2 Case Identification

Almost all Malaria deaths are caused by *Plasmodium falciparum* [34] with pregnant women, older persons, children under 5 years, and those with co-morbidities at greater risk. Symptomatically, *uncomplicated* Malaria is known to cause fevers and chills, headache, and general body weakness in those infected by the parasite. Left unattended, the disease may rapidly progress into *Severe* with patients exhibiting one or more conditions including very low blood glucose levels, low haemoglobin[1], pulmonary oedema, renal failure, breathing

[1] Less than 50 g/L (5 g/dL)

distress, relaxed blood pressure[2], convulsions, and sometimes multisystem failure among others.

The most common symptoms at onset COVID-19 are fever, cough, and myalgia or fatigue while the less common symptoms are sputum production, headache, hemoptysis, and diarrhoea [35]. However, studies continue to indicate many patients with COVID-19 either do not manifest any symptoms or registering mild symptoms of the disease and these cases spread the virus to other non-infected persons [36]. These COVID-19 Asymptotic cases increase complexities in active surveillance especially and screening or classification, a factor impeding efficient prevention and control of the disease. Studies have indicated the relationship between the risk of infection and comorbidities. There is an increased risk of COVID-19 infection especially in persons with pre-existing conditions [37, 38]. It is reported diseases such as hypertension, diabetes, and respiratory disease are more prevalent among fatal cases [39].

2.2.3 Eradication Efforts

Eradication of Malaria requires a multi-disciplinary effort from the active treatment of asymptomatic cases [40] to socio-economic improvement [41]. With the Malaria vaccine[3] in the trial phase [42], artesunate-based medications are still the WHO recommended standard treatment for both uncomplicated and severe Malaria in humans. Without proper management, *falciparum* Malaria is known to persist in some individuals several years after leaving Malaria-endemic areas [43]. Therefore, any individual who has a fever and has been to a Malaria-endemic area is at risk. Galatas, Bassat, and Mayor [40] argue symptomless cases that persist fuel transmission. Therefore, to minimize missed diagnosis of sub-clinical Malaria, a high index of suspicion is required [44].

Incorporating patient demographic information with reported symptoms and laboratory markers are essential for more accurate results. For example, Luo et al. [45] showed that incorporating patient demographics and laboratory results provided a powerful discriminant for ferritin. In a 2017 study, with 8-features from the Kenyan Malaria Indicator Survey data, Rajpurkar, Polamreddi, and Balakrishnan [46] proposed a deep learning agent to predict the likelihood of one testing positive for Malaria using individual demographic characteristics. In the same way, in a 2020 study by Lee, Choi, and Shin [47], six ML models were compared using patient clinical information to predict Malaria. In this study, by incorporating ones' nationality as a demographic characteristic alongside recorded symptoms, the Random Forest yielded the best scores with an accuracy of 90.3% (AUC = 73.2%)

[2] Less than 70 mmHg in adults and 50 mmHg in children
[3] RTS,S/AS01 (RTS,S)

In South Africa, COVID-19 eradication efforts so far are geared towards preventive approaches to stop the further spread of the virus. As of April 2021, these measures have been boosted with vaccination roll-out starting with the most at-risk populations. As Huang et al. [35] asserts global efforts to control and eventually eliminate COVID-19 still lack early detection methods. These approaches should include improved methods for prediction and classification of the disease to reduce the transmission and improve patient survival rate.

Different ML algorithms have been applied in the prediction and classification of COVID-19. For example in a 2020 study, Hamed, Sobhy, and Nassar [48] employed a KNN variant algorithm to determine COVID-19 disease classification using incomplete heterogeneous data. The experiments showed KNN-variant algorithm outperformed both the modified KNN and standard KNN on the accuracy, precision, recall, and F1 scores performance metrics. Moreover, in a similar study, Iwendi et al. [49] reported the Boosted RF algorithm as an optimal predictor for COVID-19 where data was imbalanced[4].

2.3 Classification

In this study, we theoretically define classification as relating to the possible outcome of events occurring in a finite space i.e belonging to a specific category. However, this concept may not be interpreted the same way as the widely adopted WHO International Classification of Disease (ICD) which is essentially a list of Causes-of-Death to inform mortality and morbidity statistics [50].

James et al. [51] define classification as predicting a qualitative[5] response for an observation. Given an instance, classification algorithms induce predictive rules based on features and patterns in the data to predict classes with predicted labels assuming a minimum of two levels [52]. These classifiers employ statistical and computational models to segregate datasets into categories. As an example, an algorithm that distinguishes kidney functionality based on their estimated glomerular filtration rate *"Severe/Abnormal/Normal"* in patients can be regarded as a Quaternary-classifier; the outcome result must belong to only one level i.e severe, moderate, abnormal or normal.

There exist far more complex classifiers for example document categorization algorithms that scavenge thousands of topics and group them into themes. Although more than one class may be considered, for simplicity, this research focuses on **Binary Classification** i.e.

[4] A dataset where the class distribution is unequal
[5] Often referred to as categorical variable

Presence or *Absence* of disease. In this section, we explore both Rule-based and ML classifiers.

2.4 Inferential Classifiers: Rule-sets

Rule-set classifiers rely on a set of predetermined inferential rules to determined classes. These inference rules are set into the system to churn data into discreet prediction outcomes [53] i.e. absence or presence of disease. Theoretically, in rule-based methodologies, there is no limit on rules applied. However, as with any other strategy, these classifiers do not come without challenges. The approaches are characterized by inconsistencies, difficulty in maintaining business rules, and long load times, among other drawbacks [54]. Because of this subjective nature, there is always a trade-off between complexity in decision logic and accuracy in prediction outcomes.

Rule-set classifiers adopt inductive logic programming where each rule consists of a *prior condition* sometimes called an antecedent and a consequent/resultant. These classifiers take the form

$$if LEFT then RIGHT \tag{2.1}$$

The rules dictate if the *"LEFT"* hand-side of the rule is satisfied, it should imply the *"RIGHT"* hand-side which in this case is the class label we are predicting. In practice, Rule-based classifiers take into account all the rules to determine their performance. To estimate Rule Quality, we use Rule Interestingness Measures (RIM) to distinguish between rules. This is an area still under research with no standard notations available yet [55]. Moreover, Piatetsky-Shapiro [56] proposes three criteria every RIM should satisfy.

1. The measure should be Zero if $N_{Both} = (N_{Left} \text{ x } N_{Right})/N_{Total}$
2. The measure should increase monotonically with N_{Both}
3. The measure should decrease monotonically with each of N_{Left} and N_{Right}

where

N_{Left} : Count of instances matching LEFT
N_{Right} : Count of instances matching RIGHT
N_{Both} : Count of instances matching both LEFT and RIGHT
N_{Total} : Total number of instances

2.5 Machine Learning Classifiers

Because of their ability to adapt, learn and continuously improve, ML algorithms are increasingly being used to make predictions in critical contexts [57] where the main goal is to maximize generalization i.e the ability to classify new data [53] previously unexposed to the classifier. These algorithms can pass data, learn from it and apply the newfound knowledge to make intelligent decisions. This is achieved by creating mathematical functions that relate input to desired output with differing complexities. On a broad scale, the algorithms are organized into a taxonomy based on the desired outcome [58]. In this section, we briefly describe the three general categories.

2.5.1 Unsupervised Learning

Sometimes, we are not fully aware of what features (X) should inform modeling solutions to classify a target/outcome (y). Our goal is then to explore the data and discover interesting patterns and properties [59] in the data as opposed to prediction. These learning methods are termed *Unsupervised Learning*. Techniques such as Principal Components Analysis [6] and Clustering[7] are typically used to provide labels (clusters) or values (rankings) [60] before supervised techniques are applied. Unsupervised learning is by far a subjective process and for this reason, can be hard assessing performance from these approaches. As James et al. [51] argue, there is no universally adopted mechanism to validate results from an independent dataset.

2.5.2 Semi-supervised Learning

A variation of Unsupervised Learning, *Semi-supervised Learning* is sometimes the appropriate choice especially when the datasets contain only a small portion of labeled data. Using ensemble methods[8], the algorithms then generate annotations for the unlabelled data to quantities large enough to appropriately train the models. In principle, the bootstrapping process employs a supervised learning approach to classify these unseen data. To evaluate these classifiers, it is worthwhile having genuine annotated data for evaluation [60].

2.5.3 Supervised Learning

These learning algorithms are ideal for discreet outcomes i.e. the underlying output variable can only assume one of two states such as diseased or not-diseased (binary classification). Algorithms such as decision tree induction, SVM, KNN, RF [61] among others

[6] A tool used for data visualization or data pre-processing
[7] A broad class of methods for discovering unknown subgroups in data
[8] A set of classifiers whose individual decision are combined in some way to classify new examples

provide mechanisms to learn from annotated data and make predictions on new data. All these algorithms exhibit unique strengths and largely depend on the data quality and task at hand.

In supervised learning, we assume a functional relationship exists between input and output. Let $\{x, y\}$ be a set of attributes where y is the class label of instance x, then the attributes for disease (D) classification will be a set of dependent variables together with the clinical diagnosis consisting of Positive cases (D^+) and disease Negative cases (D^-). In other words, the algorithm assumes the form $D^+ \bigcap D^- = \emptyset$ where the output has been labeled a priori [62] i.e. there is some knowledge of the data.

2.6 Estimating Classifier Performance

To measure how well a model performs, it has to be evaluated on specified metrics. This is done by subjecting the algorithm to data previously unused in the training process or employing other proven schemes. A common approach is to split the data into chunks with 80% for training, 20% for testing. This is done to determine how accurately our predicted classes match the known labels in the evaluation set [60].

2.6.1 Optimization

However, as with most ML tasks, splitting data is hampered by the number of observations to apportion for training, testing and evaluation. As a result, some of the data are likely to be used both during model training as well as testing at the same time. This situation is sometimes called contamination and is likely to result in invalid estimates. On the other hand, not all features are important to train models. Sometimes less-important attributes may be used to fit classifiers [51]. This allows models to learn non-linear patterns in the data leading to high variance. This behavior is also called model over-fitting and usually happens where a model performs well on training data and suboptimal on out-of-sample data.

On the other hand, poor performance during training may yield better results during out-of-sample data testing. In this case, the model is said to **underfit** the data. Model-misfit (under-fitting and over-fitting) is often a problem in predictive analytics and require attention. One way to address model-misfit is to account for the unequal distribution of classes. In this approach, a classifier is fit with distribution weights of the target specified as hyper-parameter. Another alternative is to resample the data to obtain equal representation in the target [63].

Another approach involves using robust approaches like Cross-Validation (CV) with *K-Fold* and *Grid Search* strategies. CV provides a method for evaluating how good a model fitted generalizes new data. With K-fold CV, the data (X, y) is randomly split into disjoined K [9] subsets. The classifier is then iteratively trained using every single bin as testing data with the rest (K - 1 subsets) as training data after which the average performance is determined. In the Grid Search technique, a set of all possible combinations of settings specified in the parameter grid are iteratively passed to a model using the CV strategy. After this iterative process, settings that yielded the highest scores from the validation are returned for model specification and generalizability. Given that the technique can be computationally intensive, it is highly dependent on performance metrics from the K-fold CV optimizer.

Notation: Assume a labeled dataset (X, y) with an input matrix X of n x m dimension and output vector y of $nx1$ dimension, fitting a statistical model p which given the i-th sample from X can predict the i-th element in y. Now the goal is to fit p such that for new input X_i we are still able to predict y_i.

2.6.2 Model Selection

For this research, we used two common techniques in predictive analytics i.e the *Confusion Matrix*[10] also called the Error Table and AUC. In addition to the confusion matrices and the AUC, we also visually represented the classifier performance using the Receiver Operator Characteristic and the Precision-Recall (PR) curves. These are further explained in Section 3.9.

[9] K denotes a positive integer greater than 2. usually 10 is appropriate

[10] A special kind of Contingency Table with two rows and two columns that reports the number of false positives, false negatives, true positives, and true negatives. *https://en.wikipedia.org/wiki/Confusion_matrix*

3 Materials and Methods

In this chapter, we describe the study population from which data were drawn along ethical considerations guiding the research. We also systematically address the procedures taken adopting specific supervised ML algorithms to address the objectives stated in Chapter 1, from model specification to evaluation criteria.

3.1 Approach

This is a non-population-based retrospective study that analyses secondary Malaria data collected over a 5-year period (January 2015 to December 2019), and COVID-19 data collected over one year (March 2019 to March 2020) by the National Health Laboratory Services (NHLS), South Africa.
The study utilizes pre-processed data generated or accumulated through suspected-case notification systems informing NMC surveillance from point-of-care/health facilities as well as laboratory results originating from sample testing.

3.2 Study Site

This research was conducted at the NICD, a division of the National Health Laboratory Services[1] located at Sandringham, Johannesburg South Africa.
Located on the southern tip of the African continent between 29°00'S and 24°00'E, South Africa experiences a varied climate throughout the year with colder months between June to August with the warmer months from December to February[2].
Covering a total area of 1,219,602km^2, the country's landscape ranges from the lowvelds and bushvelds of Limpopo and Mpumalanga, the highvelds of Gauteng and Free State, the Eastern Highlands of Kwazulu-Natal and parts of Eastern Cape, the great Karoo of Western Cape to the Bushland, Namaqua Lands and Griqualand of Western and North Western and Northern Cape[64].

[1] The largest diagnostic pathology service in South Africa
[2] Geography and climate | South African Government. (n.d.). Retrieved October 19, 2021, from https://www.gov.za/about-sa/geography-and-climate

3.3 Study Population and Data Sources

These data composed of clinical diagnosis, recorded vital signs and symptoms, reported risk factors, laboratory results, and demographic information from suspected cases in both migrant and static populations as they presented at point-of-care facilities. Owing to topographical and socio-economic differences, we selected data from all districts and subdistricts accumulated from all the nine provinces stored in the Surveillance Data Warehouse at the NICD .

Malaria:
Malaria data used for this research was made up of three Comma Separated V files. Described below, these files were linked together using *episode_no* as key-field.

- MalariaDemographics - consisting of clinical notification data (cases identified through the NMC app) with 222,805 unique observations from 10 variables
- MalariaResults - A repeated measures file with 766,074 observations from Laboratory tests
- MalariaExtra1 with 40094 observation and 20 attributes (excluding key-field) containing observed symptom along with treatment information, records of travel (including dates) and contact history.

COVID-19
This was a unit dataset (COVID-19.csv) with 35202 observations from 25 variables, excluding *episode number* (key-field). This datafile contained patient demographic information, recorded signs and symptoms, and reported comorbidities.

3.4 Computational Environment

While there exist alternative Integrated Development Environments that yield the same results, the pros and cons associated with them is a subjective topic. However, we chose and setup our computational environment using both licensed commercial and open-sourced BSD licensed tools. These tools and utilities were hosted using Microsoft Windows Operating System. In the Table 3.1 below is a full listing of resources used.

3.5 Conceptual Framework

We conformed with the agile software development methodology and adapted the generally acceptable framework for supervised ML illustrated Figure 3.1 below. In focus, we

Table 3.1: Computational Environment

Utility	Version	Notes
PANDAS	0.24.2	For high-performance data structures and analysis tools
SKLEARN	0.24.1	Tools for predictive data analysis
SEABORN	0.9.0	Python data visualization library based on matplotlib[3]
NUMPY	1.16.2	Numerical library to facilitate the data management process
PYTHON	3.7.4	Programming language. Based on the Anaconda Integrated Development Environment (IDE)
STATA	15.1	Statistical Packaged for analysis (IC Edition)
COMPUTER	10 Pro	Intel® Core™ i5 2.3GHz Processor, 16Gb memory 64-bit Microsoft Windows Operating System

SKLEARN includes class libraries for ML models. Source: https://scikit-learn.org/stable
SEABORN based on MATPLOTLIB graphic library
PYTHON, interpreted object-oriented programming language. Source: www.python.org/
STATA statistical software with annual updates. Source: www.stata.com/

tackled aspects of the iterative process that drives the development of ML models. Key considerations included volume and nature of data used, distributions in attributes, industry standards and approaches, assumptions to decisions taken among others.

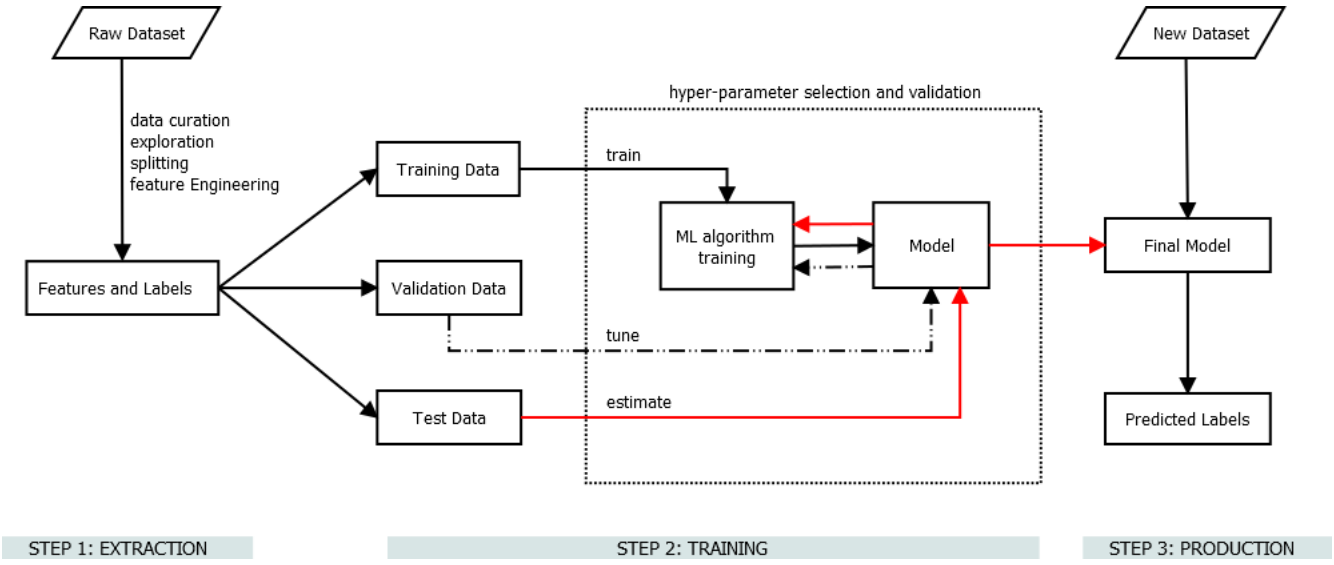


Figure 3.1: A conceptual framework for Supervised Machine Learning; adapted from various internet sources

3.6 Preprocessing

During the data-extraction phase, key vectors that will define the dataset and tune the algorithm are deduced, cleaned and standardized. Because Malaria data were received as

three separate files uniquely identified by *episode* numbers, we concatenated them using the Inner Join[4] strategy to obtain a single entity. An insight into the pre-processing is summarised below:-

3.6.1 Curation

In an ideal world, data is clean and ready for analysis. However, this is not always the case. Real-world data are messy. Adopting normalization approaches as employed in relational databases, Wickham [65] proposes the *tidy-data* model where every variable forms a column, each observation forms a row and each type of observational unit forms an entity. There are interesting proposals in the literature regarding data tidying. However, proposed methodologies may not necessarily be applicable in all situations as datasets usually differ.
In this research, data were received as flat tables, with features clearly defined by columns and rows denoting observations. Checks for missing values and transcription errors were done and where possible corrected from referenced features. String values were encoded by mapping feature schemas[5] and where possible data re-coded with appropriate data types enforced. Correlation matrices were used to identify probable patterns and relationships between attributes.

To detect peculiarities in the data, exploratory data analysis was done using distribution plots to detect peculiarities[6] values in the data using the notation below.
Let T_k denote the threshold value for a certain feature K that follows a skewed distribution, $T_k = 1.5 * IQR$[7]; a value is said to be peculiar if it falls either below or above the threshold T_k. Consequently, these instances were dropped from the dataset. An example from the Malaria data is shown in Figure 3.2 below where we noted 1544 probable erroneously recorded ages i.e above 140 years. These implausible data did not fit within *normal* limits and were consequently not considered for analysis.

3.6.2 Data Definition

We filtered and subjected only records with information across the two datasets to the data cleansing and feature engineering process as described in subsection 3.6.3. An observation was considered candidate for use if and only if a laboratory test record was successfully linked in the demographic-information dataset via an *episode* number.

[4] A join operation in relational algebra - combining from entities in a relational environment (wikipedia)
[5] Categorical classification of a variable
[6] Out-of-range
[7] Interquartile Range also called Midspread is a statistical measure of dispersion

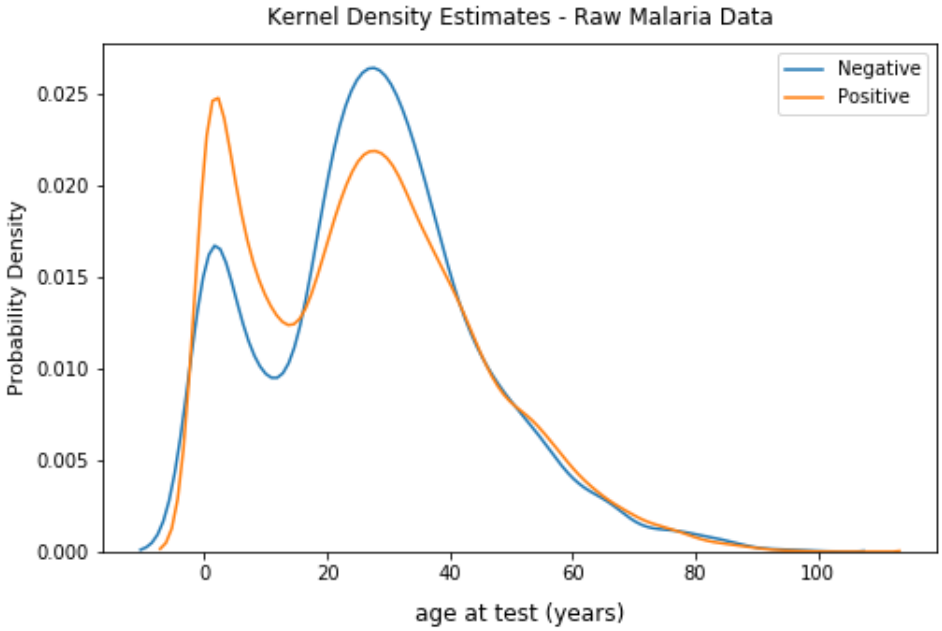


Figure 3.2: Kernel Density Estimate plot for age at testing (years)

From this iterative process, we deduced feature vectors to inform approaches at particular steps in ML. As Jutte, Roos, and Brownell [66] assert, the process requires extensive resources to assemble key indicators. In our research, two data files were used; *Malaria* and *COVID-19*.

Malaria:
This dataset consisted of laboratory markers from laboratory measures as well as demographic attributes informed by case-notification data. These data included test methods, clinical symptoms, specimen measures parasite and cell counts, test dates, triage information like *episode number*, *admission status* among others. The aggregated dataset contained 37 features. Of the 216,408 observations, more than half the predictor variables had over 80 percent missing information. These predictors could neither be imputed nor used for model specification. Consequently, these missing data were dropped from the final analytical dataset as shown in Figure 3.3 below. A missing values report is shown in Table 7.1 annexed in Appendix 7.

In the table below is a high-level description of the Malaria analytical data used with a descriptive summary presented in Section 4.1.

COVID-19:
On the contrary, in deducing the COVID-19 analytical dataset, we did not drop missing

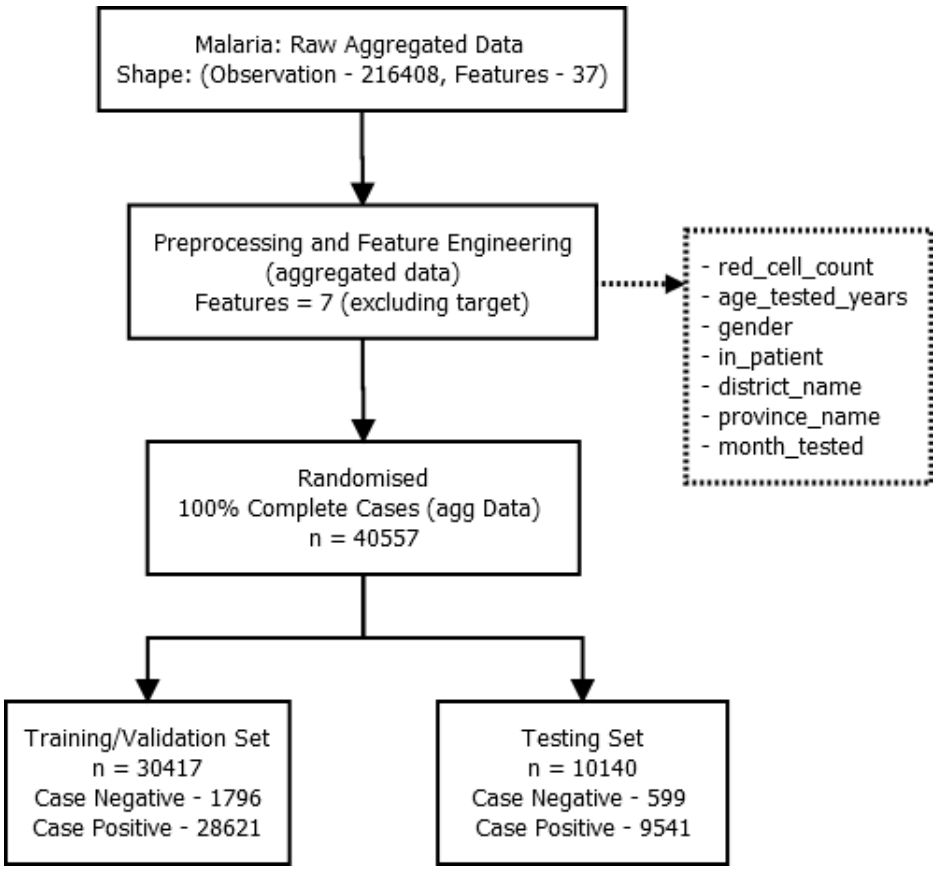


Figure 3.3: Preprocessing flow - Malaria dataset.

Table 3.2: Malaria Dataset definition

sn	Variable	DataType	Description
1	Target	String	A laboratory confirmed malaria test result
2	Gender	String	Participant recorded gender
3	in_patient	string	Participant hospitalization status
4	age_tested_years	Integer	Participant recorded age (in years) at time of malaria test
5	red_cell_count	float	Red Blood Cell (RBC) count from laboratory
6	weather	string	Calendar season deduced from the South Africa meteorological calendar
7	district_name	string	District where test was done
8	province	string	Province reporting malaria test-result

data. Using regular-expression text processing techniques, we inferred 14 features; 13 binary and 1 continuous from 24 candidate attributes in the raw data. A missing value report is annexed in Table 7.2 of Appendix 7. To inform probable symptoms, these features were categorized to fall in either of the following groups:- *fever/chills*, *cough*, *sore throat*, *shortness of breath*, *diarrhoea*, *muscle/joint pains*, *malaise*, *fatigue/lethargy*, *influenza*, and *vomiting/nausea*.

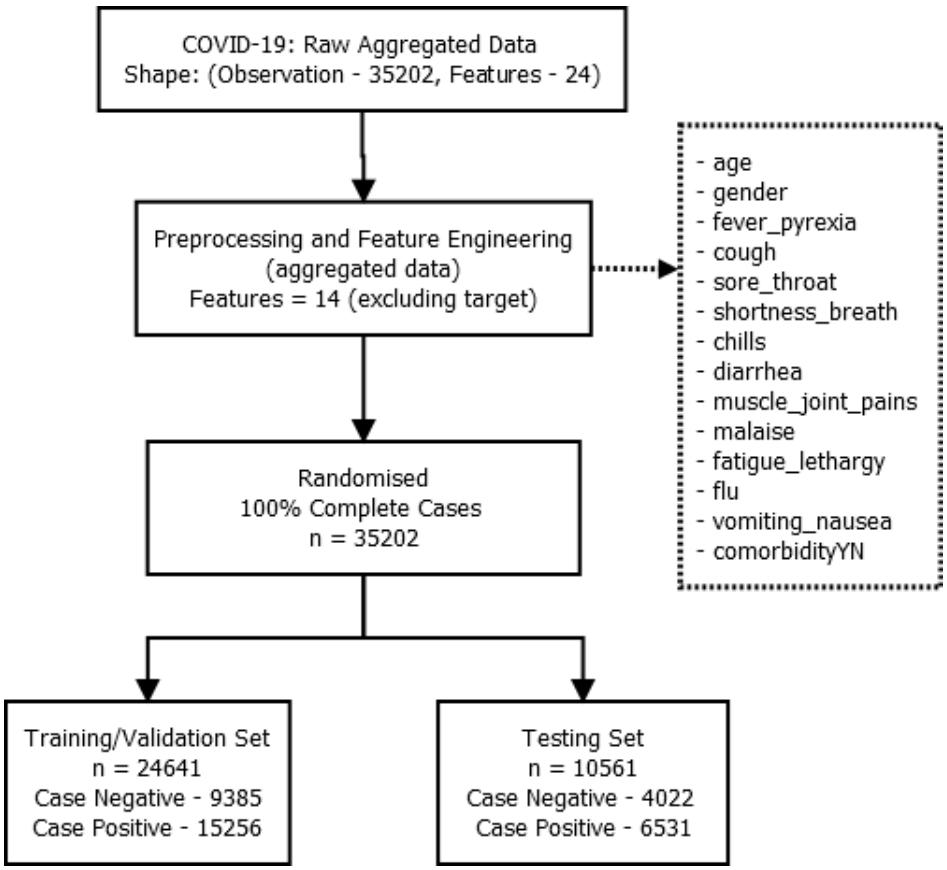


Figure 3.4: Preprocessing flow - COVID-19 dataset.

Because of inconsistencies in which this information was captured, a pooled indicator *ComorbidityYN* was created to indicate a 'Yes' - 1 if any comorbidity were registered or 'No' - 0 if neither. Below is a description of the COVID-19 dataset with descriptive summary statistics provided in section 4.1.

3.6.3 Feature Selection and Engineering

In predictive analysis, not all features in a dataset are important for classification and prediction. Yet, there is no one-size-fits-all method to this task. One approach is to use unsupervised statistical techniques like Principal Component Analysis (PCA) to attain this. In this research, we employed domain knowledge, a manual dimension-reduction technique to carefully select principle features for the task. This search problem aimed at minimizing collinearity and model misfit by removing correlated features and noise.

Ensemble Selection was performed on Categorical attributes *Sub-District*, *Province*. Proposed by Niculescu-mizil et al. [67], we limited OneHotEncoding to the top-10 levels for

Table 3.3: COVID-19 Dataset definition.

sn	Variable	DataType	Description
1	Target	String	A confirmed PCR COVID-19 test result
2	Age	Integer	Participant recorded age (in years) at time of COVID-19 PCR test
3	Gender	String	Participant recorded gender
4	Fever/Chills	boolean	Deduced symptom from statements inferring absence/presence of fever
5	Cough	boolean	Deduced symptom from statements inferring absence/presence of cough
6	Sore Throat	boolean	Deduced symptom from statements inferring absence/presence of sore throat
7	Shortness of Breath	boolean	Deduced symptom from statements inferring absence/presence of shortness of breath or difficulty in breathing
8	Diarrhoea	boolean	Deduced symptom from statements inferring absence/presence of diarrhoea
9	Joins/Muscle Pains	boolean	Deduced symptom from statements inferring absence/presence of joint and muscle pains
10	Malaise	boolean	Deduced symptom from statements inferring absence/presence of malaise
11	Fatigue/Lethargy	boolean	Deduced symptom from statements inferring absence/presence of fatigue or lethargy
12	Influenza	boolean	Deduced symptom from statements inferring absence/presence of influenza, common cold and sneezes
13	Vomiting/Nausea	boolean	Deduced symptom from statements inferring absence/presence of vomiting or nausea
14	ComorbidityYN	boolean	Deduced from statements indicating absence/presence of any underlying comorbidity

these categorical attributes. By mapping categorical data onto a binary scale, the OneHotEncode process involves converting each categorical value into distinct attributes consisting of '1' or ' 0' denoting a presence or absence of a level. Demographic variables *Hospitalization Status*: 'Y', 'N' and *Gender*: 'M', 'F' were label encoded to **1, 0** denoting a Positive and Negative response respectively. For both datasets, the target (dependent/identifier) variable was classified as binary with '1' and '0' denoting an observed *positive* and *negative* clinical outcome respectively.

Computational and time complexities were minimized by standardizing all features on a continuum to fit between Zero to One using the **MinMaxScaler** implementation through

the SKLEARN library. This is denoted by:

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{3.1}$$

where

X_{scaled} : the new transformed vector
X : the vector instance to transform
X_{min} : the minimum value of X in the vector domain
X_{max} : the maximum value of X in the vector domain

To minimize data leakage, the training and out-of-sample datasets were engineered independently through SKLEARN fit-transform methods. Because of the skewness in distribution, missing values in categorical features were imputed with the *most-frequent* observations. Because of the adequate trade-off between precision of imputation without distorting the structure of the data [68], features on a continuum were imputed using the *Nearest Neighbor* strategy.
In the COVID-19 analytical dataset, we probed for possible patterns with missing data though we did not find any positive findings. We summarily concluded that missing data was completely at random. We therefore imputed missing values for *age* using the Nearest Neighbor imputation strategy and *gender* using the *most frequent* strategy [69].

3.7 Model Specification

In the model specification phase, the training dataset is used to learn the underlying algorithm [70] before iteratively parsing it until a best-fit model that predicts the outcome is derived. This stage involved modeling and optimizing to achieve scores that can be generalized. After preparing the data (data preparation phase), the classification models and their unary derivatives were then subjected to an iterative split-train-predict process to obtain generalized scores. The process is briefly explained below.

3.7.1 Splitting

For this research, generally acceptable ML data partitioning schemes are applied. Firstly, the datasets are divided into two portions, stratified on the **Target** variable (clinical outcome) with $(1/4)$ as **out-of-sample** and the rest $(3/4)$ for model specification.

3.7.2 Classification Strategies

In the classification and prediction of COVID-19 and Malaria clinical outcomes, we used three strategies. Firstly, we under-sampled the majority class to obtain equal representation of clinical outcomes for both positive and negative groups. In the second approach, we accounted for the distribution imbalances in the data in the classification and prediction of clinical outcomes. The third approach, a novelty technique involved classifying the minority category as outliers and then predicting labels that presumably belong to this class using out-of-sample data.

3.7.3 Hyper-parameter selection

To define a generalized model that can be deployed on out-of-sample data, we defined initial possible parameters for the predictions. Using, a stratified Parameter Grid search approach, the models were continuously deployed and refactored on all possible combinations using a 5-fold Cross-Validation scheme. Parameter combinations yielding the best F1-Score were identified and subsequently re-fit to define a generalized model. This approach was repeated for model specification on under-sampled (balanced) data and models accounting for distribution weights.

3.7.4 Support Vector Machines

SVM models are trained to distinguish and segregate all instances of a unit class versus the rest. Using the concept of margins[8], SVM aims to find an optimal hyperplane that best separates the data groups [53]. In an ideal scenario, data is separable i.e. positive cases distinguished from negative cases. We can then draw a hyperplane to achieve this by identifying support vectors[9] to distinguish the separation. SVM approaches seek to predict labels by finding parameter functions that maximize the margin. This classifier naturally avoids over-fitting and bias problems by choosing a fitting less complex function yielding minimum training errors; a technique called regularization. In Figure 3.5 is a high-level illustration of the SVM classification technique.

Notation: Assume a set of N training examples that can be identified as belonging to either class say T and F. Let these examples take on values $+1$ and -1 respectively with each data point x_i having several of attributes K. Then, the training data takes the form (x_i, y_i) where $i = 1, ..., N$ and $y_i \subseteq \{+1, -1\}$ and $x \subseteq R^K$ where R is an instance of K as illustrated in Figure 3.5.
This implies $y_i = +1$ if $x_i \subseteq T$ and $y_i = -1$ if $x_i \subseteq F$. For linearly separable data, the

[8]The distance between a hyperplane and the closest points; also called support vectors
[9]Data points on the lines that define the margins

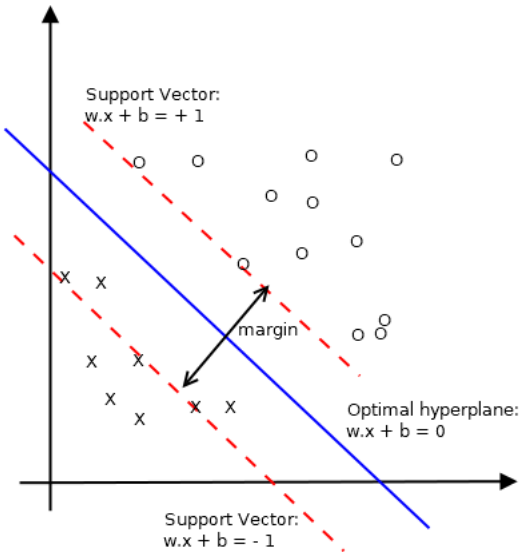


Figure 3.5: SVM classifier a case of linearly separable data

hyperplane assumes the function

$$w^T x + b = 0 \tag{3.2}$$

where

w = Normal vector that is perpendicular to the plane

b = Bias determining point location relative to the origin.

Then, SVM searches for a separating hyperplane by maximizing $\frac{1}{|w|}$. Therefore, new data points x_i can then be classified using the decision rule expressed as

$$f(x) = sign(w^T x + b) \tag{3.3}$$

For linearly separable data, SVM introduces a slack variable ε_i for the constraints and assigns a penalty to data points on the wrong side of the hyperplane. A general expression of the function for linearly separable data can then be expressed as

$$\forall_i y_i (w^T x + b) - 1 + \varepsilon_i \geq 0 \tag{3.4}$$

However, there are instances when the data can only be separated by a curved decision boundary i.e not separable linearly. In such cases, SVM assumes a soft margin to re-classify data points that were wrongly classified and introduces a penalty C for this misclassification by using the linear separation technique. The SVM then assumes a function

$$min \frac{1}{2} |w| + C \sum_{i=1}^{N} \forall_i y_i (w^T x + b) - 1 + \varepsilon_i \geq 0 \tag{3.5}$$

Being a classification task we adopt the Support Vector Classifier (SVC), a variation of

the SVM to classify and predict Malaria and COVID-19 clinical outcomes. Furthermore, because of time complexities involved with the algorithm, the kernel hyper-parameter was set 'linear'.

3.7.5 The k-Nearest Neighbor Method

This is a non-parametric classification[10] technique that relies on readily available data to predict classes on new data. Novel labels are predicted based on properties shared with other data points i.e an object is classified by the popular vote of its neighbors (K). KNN is dependent on the distance function used to measure similarity [57] between data instances i.e. proximity of K data points.
With K=1, all new data points will be classified according to properties shared by their immediate neighbor. However, misclassification errors (false positives and false negatives) arise with larger values of K. For example, a Malaria-positive case may be labeled negative if the dataset contains the majority of such (negative) cases. Therefore, The optimal size of K is the one that minimizes the classification error. Illustrated in Figure 3.6, instance 'N' will be classified by defined characteristics of either 3 or 7 nearest Neighbors. With KNN,

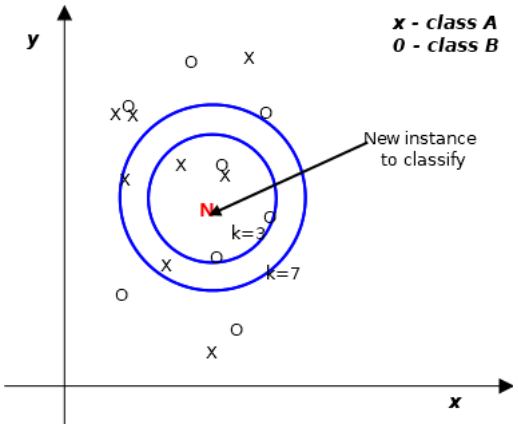


Figure 3.6: KNN classifier

proximity is estimated by the Euclidean distance function

$$D(x,y) = \sqrt{\sum_{i=1}^{m}(x_i - y_i)^2} \tag{3.6}$$

Where $x = x_1, ..., x_m$, $y = y_1, ...y_n$ and and m = the attribute values of two points $x \quad and \quad y$

[10]Methods that make statistical inferences without regard to any underlying distribution.

3.7.6 Decision Tree Learning: Random Forests

These use the concept of decision trees to classify data into discreet classes by using a set of rules. These set of rules iteratively split the data on key attributes i.e characteristics that most separate the data until further splits are not more informative. A decision tree cascades a set of rules, where each rule may infer either another rule or lead to a decision. Recursive algorithms like Iterative Dichotomizer 3 are used to construct decision trees. In the analogy of a forest, numerous decision trees collectively form Random Forests.

In the learning process, at each node, the algorithm *gains information* that best discriminates labels. A form of dimensionality reduction, information-gain can also be used for feature selection where each candidate feature is evaluated in the context of the target. This *information* is then used by the child node in a cascade manner until a decision is reached. Figure 3.7 illustrates the RF classifier.

Notation: Assume a dataset S, with attribute A, let S_v denote an instance/subset of the data; $S_v \subset S$. Also let $A = v$ and $Values_A$ is the set of all possible values of A, then Information gain[11] can be expressed using the function below

$$Gain(S, A) = Enthropy(S) - \sum_{v \epsilon Values(A)} \frac{|S_v|}{|S|} Enthropy(S_v) \tag{3.7}$$

where Entropy is the measure of uncertainty of a random variable A

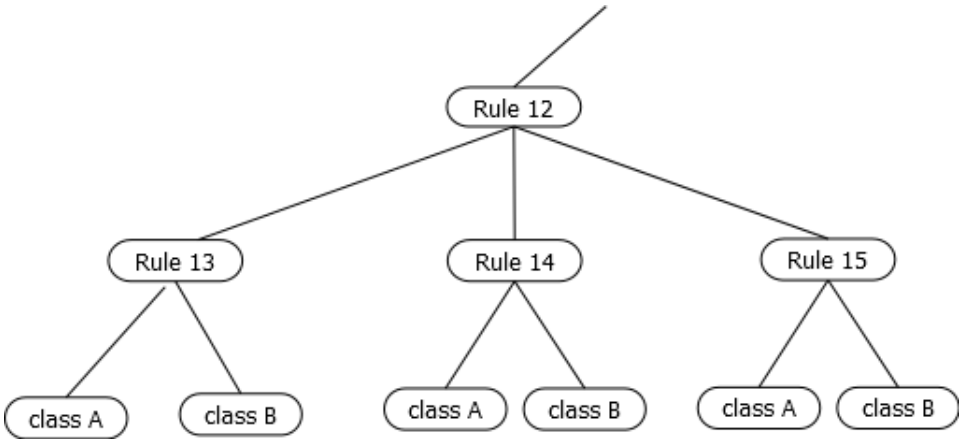


Figure 3.7: Decision Tree classifier branch

[11] A measure of this change in entropy

3.8 Novelty Detection Approaches

The ML classification models discussed so far have one common assumption; the data to some degree has equal representation in the outcome hence balanced. However, this may not always be the case. Epidemiological studies have shown certain morbidities occur less frequently than others. In such cases, the surveillance and/or clinical datasets tracking these conditions are by far more likely to be biassed towards less frequent occurrences. In ML, scarcity of a particular class label in a prediction dataset is what defines anomalies [71].

Anomalies have two distinct quantitative properties; consist of fewer instances in the dataset and have peculiar characteristics (data values) compared to the majority (normal) instances. In principle, novelty detection focuses on the identification of abnormal patterns from large amounts of normal data [72]. As Sun, Wong, and Kamel [73] argue, classification rules that predict the small classes tend to be rare or undiscovered. As a result, out-of-sample data belonging to small classes are likely to be misclassified compared to those belonging to the majority class. Under those circumstances, deviations from the 'normal' provide an alternative to model the minority class as anomalies, a technique sometimes called Novelty detection of Unary/One-class classification.

Whereas kernel optimization methods, statistical approaches, and Neural Networks [74] amongst other strategies are available for anomaly detection, in this research, we focus on two strategies i.e iForest and OCSVM.

3.8.1 Isolation Forest

Proposed by Liu, Ting, and Zhou [75], the iForest algorithm recursively partitions instances randomly until all possible instances are completely isolated. The approach uses a Binary Search Tree algorithm to construct isolation Trees (iTree) from randomly selected attributes, collectively forming an iForest.

Assume $X = \{x_1, ..., x_n\}$ is a sample of the data with N training examples, an iTree is constructed by recursively splitting X by randomly selecting an attribute q and split value p until (a) the tree has reached a height limit or (b) $|X| = 1$ or (c) all data in X have the same values. This approach creates shorter paths from the node and is independent of distance or density measures.

3.8.2 OneClass SVM

In OCSVM, data presumed to originate from the normal class is used to train the support vector model, after which the model is tested on **contaminated**[12] data to ascertain performance metrics for the segregation. Originally developed for two-class classification tasks, extensions and enhancements to the SVM like Support Vector Data Description [76] and Local-Density OCSVM [77] have been proposed with empirical results suggesting a better performance of the classifiers compared to the original OCSVM.

3.9 Learning Criteria

3.9.1 Contingency Table Metrics

To fully understand model evaluation using supervised binary classifiers we contextualize all possible results in a 2X2 table illustrated in Figure 3.8 below. Sometimes referred to as an *Error matrix* or *Confusion Matrix*, we deduce statistics that inform classifier performance from two possible cases i.e true classifications and miss-classifications. We enumerated these two possible scenarios as TP: True Positives - correctly predicted labels, TN: True Negatives - correctly predicted negative labels, FP: False Positives - incorrectly predicted negative labels (Type-I error) and FN: False Negatives - incorrectly predicted positive labels (Type-II error).

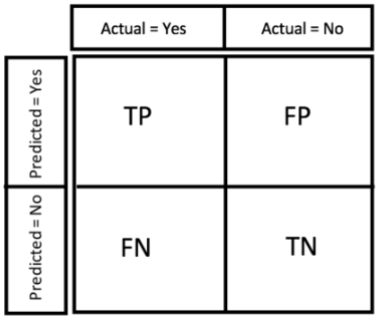


Figure 3.8: Error Matrix

Firstly we examine **accuracy**, an estimation of how well an algorithm discriminates unseen instances. Often expressed as a percentage, it is computed by dividing the total count of correctly classified instances by the overall predictions. Whereas *Accuracy* as a metric was used to estimate model performances on out-of-sample data, the metric is prone to distribution bias since it is reliant on the relative class balance of the outcome. In this scenario, a more robust variation of the Accuracy score is to use the Mathews Correlation

[12] In the sense refers to a dataset where the wholesome containment of the majority class is altered by presence of a minority class.

Coefficient (MCC) that takes into account all four quantities of the confusion matrix [78, 79].

To quantify the predictive capacity of a classifier, we estimate the **Precision** i.e the fraction of predicted true cases out of total true cases. Expressed by dividing the number of all correctly predicted outcomes by the total number of outcomes positive predictions. This is sometimes referred to as Positive Predictive Value (PPV). Moreover, in clinical diagnoses, it is much tolerable to commit Type-I errors as opposed to Type-II errors. False Negative results are far more fatal compared to False Positives. We then adopt a measure that helps us prove right on all positive instances [60]; **Sensitivity/Recall**. Sometimes referred to as True Positive Rate [13] is the total number of relevant/positive results correctly classified by the algorithm.

High recall values are indicative of low Type-II error; low FN counts. Therefore, better classifiers should have both a high degree of precision and recall. Thus to ease interpretation, the **F1-score**[14] was used to estimate classification performance of the algorithms. These performance metrics are summarised in Table 3.4 below. An alternative performance measures used is Specificity/True Negative Rate (TNR), essentially the number of observations predicted as Negatives out of total Negative classifications. However, as with any predictive model, wrong predictions (misclassification) are a reality and we need a measure to quantify wrong predictions by classifiers. The Misclassification Rate; a measure of falsely classified data by versus total classifications is then used to quantify this error. Let $\hat{y}_i$ be the prediction of data point i for label y_i, then the error rate can be defined as

$$misc_n = \frac{1}{n} * \sum_i (y_i \neq \hat{y}_i) \tag{3.8}$$

In Table 3.4 below is a summary of performance metrics used in this research

Table 3.4: Evaluation measures for the Confusion Matrix

Metric	Expression
Accuracy	TP + TN / ((TP + FP + TN + FN))
MCC	(TP * TN) - (FP * FN) / sqrt(((TP + FP)(TP + FN)(TN + FP)(TN + FN)))
Precision/PPV	TP / (TP + FP)
Sensitivity/Recall	TP / (TP + FN)
F1-Score	2 * (Precision * Recall) / (Precision + Recall)
Specificity	TN / (FP + TN)

[13] A measure os a classifiers' completeness
[14] The weighted harmonic mean the algorithms' Precision and Recall

3.9.2 Area Under curve (AUC)

In addition to the confusion Matrix, we need a way to visualize, organize and select classifiers based on their performance as compared to classifiers with **No Skill**.
We define *No skill* classifiers as models that predict by chance i.e predictions are not informed by any prior patterns in the data.

With a balanced binary outcome, we obtain post estimation statistics using ROC curves. This is a graphical representation of the trade-off between FN and FP rates for every possible cut-off and is useful for visualising Recall/sensitivity and specificity. Better classifiers present curves more proximal to the top-left corner in the ROC space. This interpretation is however very subjective and quantifying these would prove a more meaningful rationale. Ranging from 0 to 1, the AUC can be used to quantify the discrimination capacity of the model with an AUC of 0.5 suggesting otherwise.

With Novelty detection, class-imbalances[15], ROC curves no longer give reliable estimates of model performance. Similar to ROC, PR curves are used to estimate models' performance. As illustrated in Figure 4.10, the goal is to be in the upper-right-hand corner [80, 81]. PR curves provide a way to summarize the trade-off between TPR and PPV value for a predictive model using different probability thresholds [82].

3.10 Ethics

The ethical and methodological aspects of this research were approved by the University of Witwatersrand Human Research Ethics Committee (M200509) and NHLS Academic Affairs and Research Office (28 September, 2020). No human subjects were involved. Surveillance data from the NMC data warehouse was extracted, de-identified, and made available in a compressed and encrypted WinRAR[16] format. All computational experiments were conducted on a bit-lock-encrypted personal laptop only accessible by the researcher.

[15] A difference in the numbers of positive and negative instances
[16] A shareware file archiver and data compression program

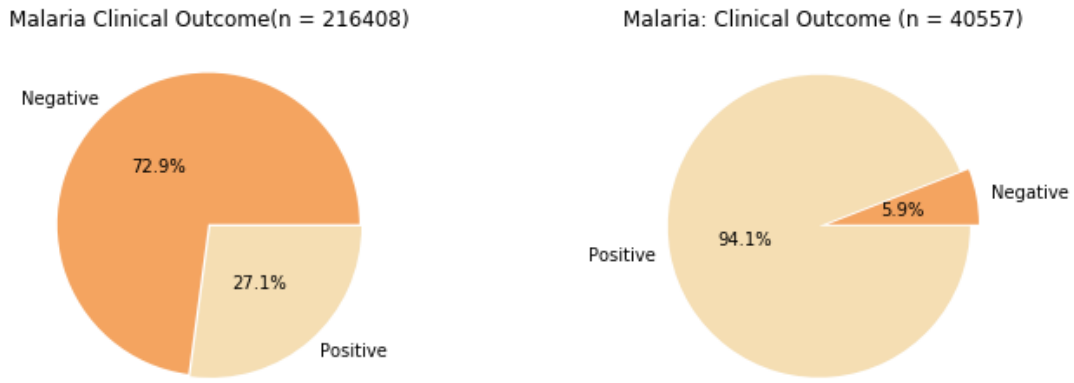
4 Results

In this Chapter, we present the results of the COVID-19 and Malaria classification models from the NMC surveillance data. We start by describing the population, then findings from preprocessing and lastly prediction results according to the classification strategies defined in Section 3.7. We also briefly report on patterns we found interesting from procedures taken.

4.1 Descriptive Statistics

4.1.1 Malaria Analytical Data

This section presents descriptive statistics for the Malaria out-of-sample analytical dataset. Firstly, we examine the distribution of clinical outcomes before and after data pre-processing.



(a) Before preprocessing **(b)** After preprocessing

Figure 4.1: Distribution of Malaria clinical outcome

In the raw (unprocessed) Malaria dataset, the positive-negative clinical diagnosis ratio was 100:268 from 216408 observations. As illustrated in Figure 4.1 above, after preprocessing, this distribution was distorted. Of the total 40557 observations considered for analysis, clinically diagnosed Malaria-positive cases accounted for 94.1% (n=38162) with the negatives contributing 5.9% (n=2395); a positive-negative ratio of 159:10. In the Malaria analytical dataset, we only considered complete cases from 7 features of which 58.35% were male

with the rest female. It was also observed that population *age at test* followed a bimodal distribution. Up to about 20 years, there were more Malaria positive than negative cases. However, as illustrated in Figure 4.2, between the ages of 20 to 45 years, there was more Malaria negative than positive cases.

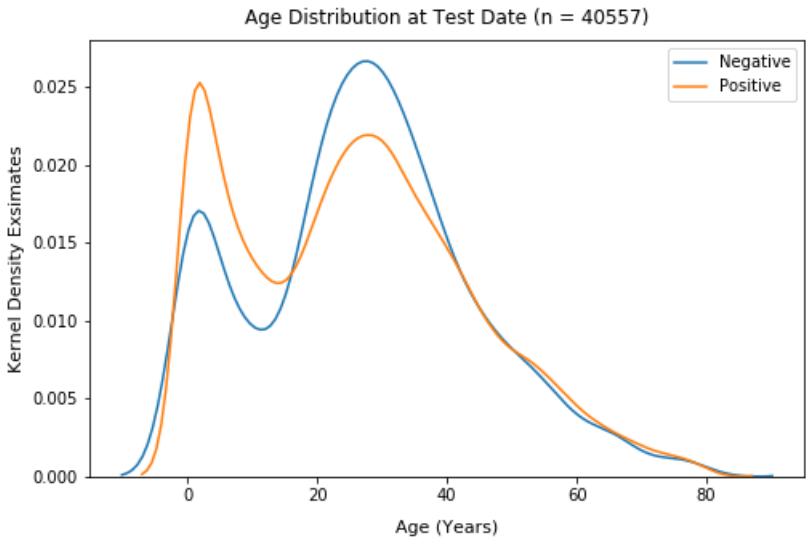


Figure 4.2: Age distribution at Test Date

We therefore, adopted an approach similar to the WHO reporting standards by creating age categories according to the WHO [2] Malaria risk population age groups. The 15-49 year-old population contributed the largest number of observations (approximately 58%, n=23512) with the older population i.e those over seventy years of age accounting for the least (1%, n=591).

Looking at the testing periods, 17.6% (n=7145) tests were done in January with the least done in July. We then investigated the relationship between *age* at sample test and *calendar month* when the test was done. From Figure 4.3, we observe the following; Those up to the age of 60 were likely to have a Malaria test between April and August. There was no clear period in which the older population (above 60 years of age) was likely to have a Malaria test.

Months were extracted from the sample collection dates, after which we categorized them into four bands according to the South-African weather season.

From Figure 4.4, we observed a fairly even number of samples tested done during autumn[1] months (n=14,443) and in the summer[2] (n=14,440). Descriptive summaries are presented

[1]March, April, May
[2]December, January and February

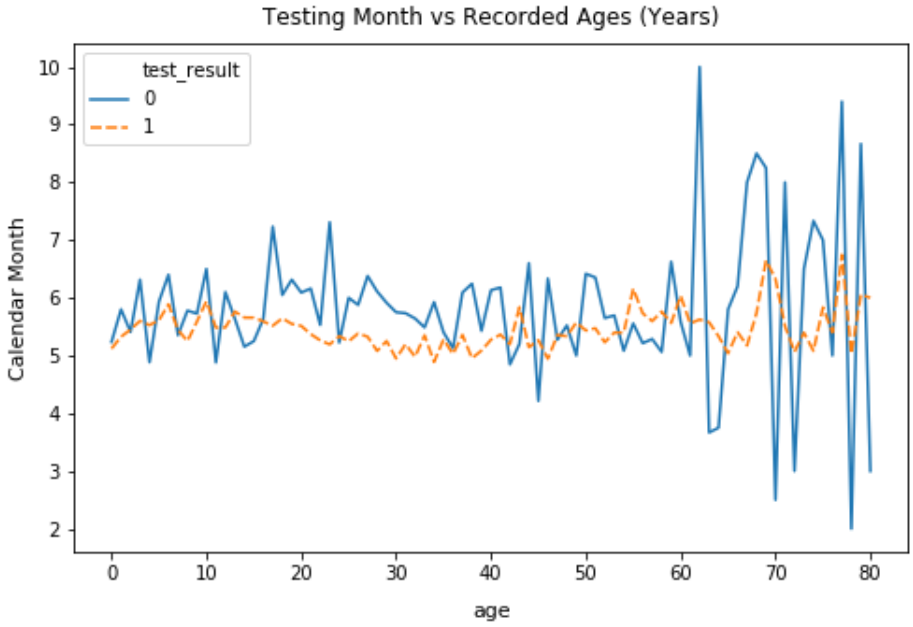


Figure 4.3: Average monthly tests by age (years)

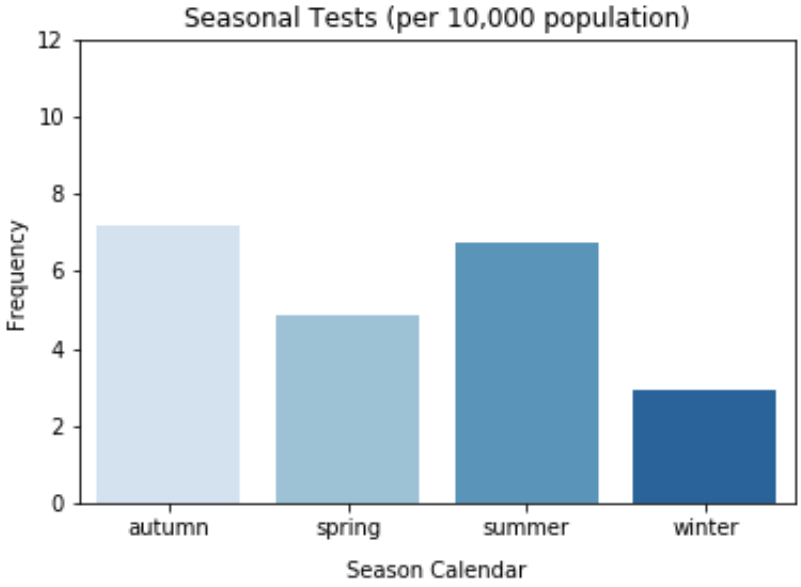


Figure 4.4: Malaria tests done per Season Calendar

in Table 4.1 below.

The Malaria analytical dataset comprised of four locality attributes namely *province, district, sub-district* and *health-facility*. We observed less variability in the *sub-district* and *health-facility* and therefore not likely to add predictive power to the classifiers, hence considering only *province* and *district* as features.
Illustrated in Figure 4.5, we observed Limpopo and Mpumalanga provinces accounted for

Table 4.1: Descriptive Summary of Malaria Dataset

Malaria dataset file description (n = 40557)		
Characteristic	**Sub group**	**Distribution: n(%)**
Clinical Test Result (Target)	Positive	38162 (94.09%)
	Negative	2395 (5.91%)
Gender	Male	23666 (58.35%)
	Female	16891 (41.65%)
Age group	under5	6703 (16.53%)
	5-14	5736 (14.14%)
	15-49	23512 (57.97%)
	50-69	4015 (9.9%)
	70+	591 (1.46%)
Hospitalization Status	In-patient	21378 (52.71%)
	Out-patient	19179 (47.29%)
Red Blood Cell count	Median (IQR)	4.37 (3.7, 4.91)
Calendar Season at test date	Autumn	14443 (35.61%)
	Summer	14440 (35.6%)
	Spring	8036 (19.81%)
	Winter	3638 (8.97%)
Province reporting result	Limpopo	19683 (48.53%)
	Mpumalanga	8425 (20.77%)
	Gauteng	6649 (16.39%)
	Kwazulu-Natal	2339 (5.77%)
	North West	1176 (2.9%)
	Western Cape	1096 (2.7%)
	Eastern Cape	519 (1.28%)
	Free State	515 (1.27%)
	Northern Cape	155 (0.38%)
District where test was done	Mopani	9624 (23.73%)
	Ehlanzeni	6847 (16.88%)
	Vhembe	6529 (16.1%)
	Ekurhuleni Metro	3554 (8.76%)
	Capricorn	1426 (3.52%)
	West Rand	1357 (3.35%)
	Waterberg	1161 (2.86%)
	Nkangala	1035 (2.55%)
	Ethekwini Metro	1006 (2.48%)
	Sekhukhune	943 (2.33%)

*IQR: Interquartile Range

There were 52 levels in the feature reporting *District where the test was done*. The last 10 districts with a reported test done were *Joe Gqabi* and *John Taolo Gaetsewe* had 13 observations each with *Harry Gwala* reporting 12 tests done. *Uthukela, Amajuba, Zf Mgcawu, Umzinyathi*, and *Namakwa* had 11, 9, 8, 7, and 3 observations respectively. Both *Xhariep* and *Central Karoo* had 2 observations each.

approximately 70% of the analytical dataset (n=28108) with less than 1000 tests done in Northern Cape province. Because of the extended levels in the attribute *district*, only the 10-most-frequent districts reported were considered and used for model specification.

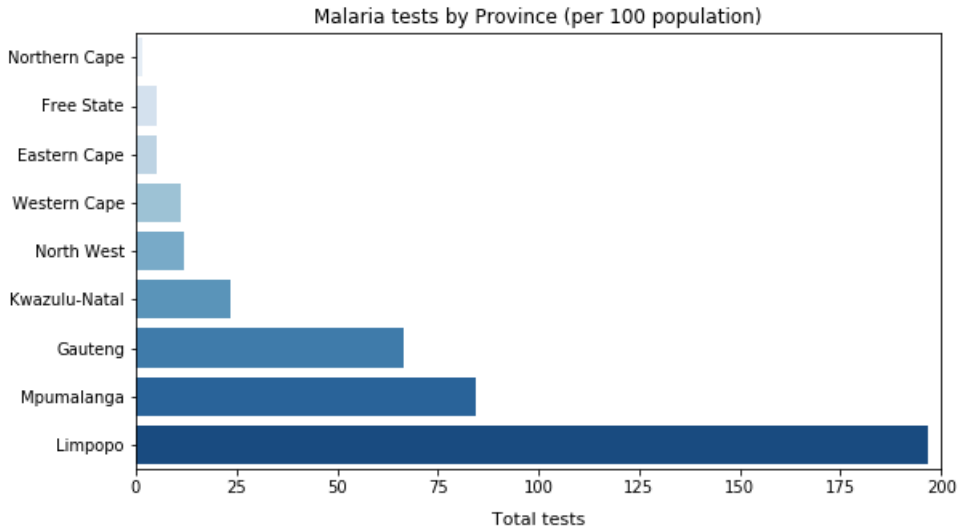


Figure 4.5: Malaria tests done per province

4.1.2 COVID-19 Analytical Data

The distribution of clinically diagnosed COVID-19 positive-negative ratio was 162:100. Because no observations were dropped from the COVID-19 raw dataset, the before/after preprocessing distribution of the clinical outcome are identical.

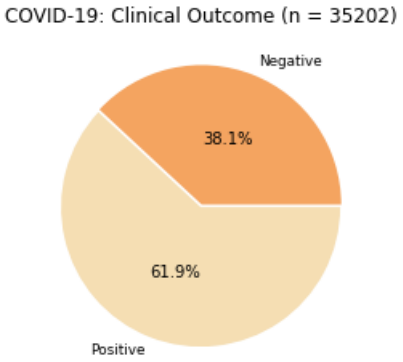


Figure 4.6: COVID-19 clinical outcome (raw dataset)

Of the 14 features considered for the analytical dataset, two contained missing values i.e 2.1% (n=739) in *age* and 0.1% (n=35) in *gender*.
As illustrated in Figure 4.7, between ages of 10 and 90 years, it was observed that the COVID-19 negative population was slightly older compared to those with a positive; 58

years (IQR 32.0, 53.0) vs 43 years (IQR 31.0, 54.0). However, suspected COVID-19 cases at the time of sample collection were on average 31.5 years old (IQR: 31.0 - 53.0).

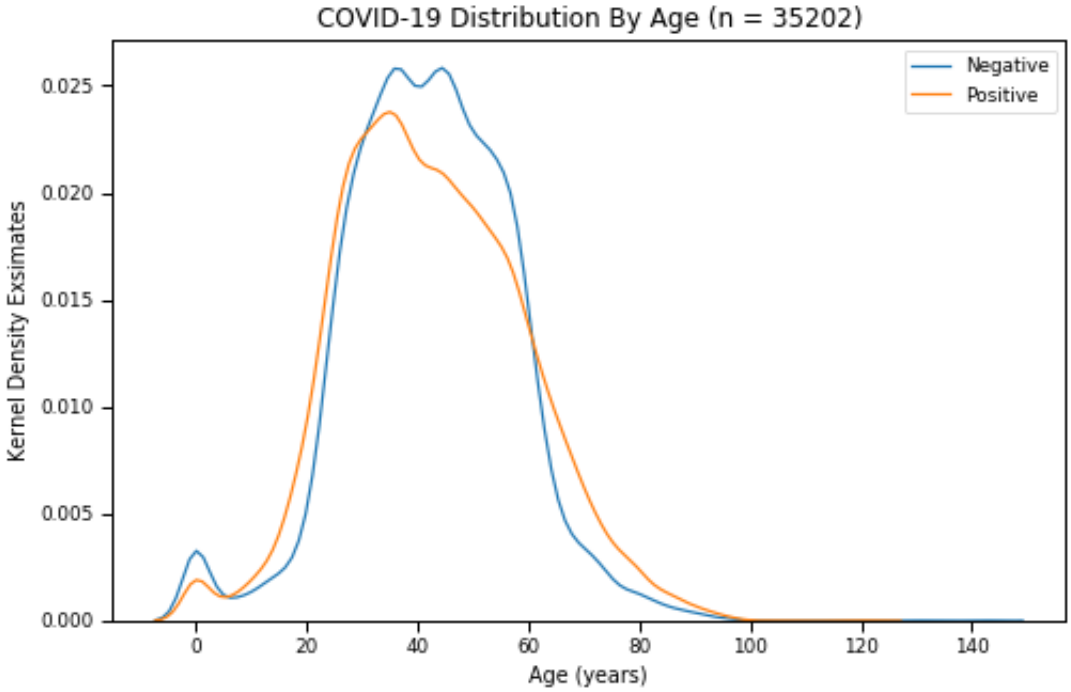


Figure 4.7: COVID-19 age distribution of the population

It was also noted that not all suspected cases registered symptoms as would be expected from such a highly infectious condition. Out of approximately 35,000 tests done, *Sore Throat* was reported in about 25% (n=8807) of the population.
Although *cough*, *fever*, and *malaise* have been reported in COVID-19 cases, these symptoms were less frequently reported in the COVID-19 surveillance as reported in Table 4.2. Figure 4.8 illustrates the logarithmic distribution of recorded symptoms in the COVID-19 analytical dataset with descriptive summary statistics of the unstratified COVID-19 analytical dataset in Table 4.2 below.

4.2 Predicting Probable Cases

To determine probable cases for Malaria and COVID-19, first, we investigated relationships between all numerical features to the target using both correlation analysis an chi-square.From the correlation analysis using the Malaria analytical dataset, we observed to a considerably large extent weaker relationships between features and the target. However, the inter-feature correlation between *gender* and *red Cell counts* was positive ($r = 0.18$).

Table 4.2: Descriptive Summary of COVID-19 Dataset

COVID-19 dataset description (n = 35202)

Characteristic	**Sub group**	**Distribution: n(%)**
Clinical Test Result (Target)	Positive	21795 (61.91%)
	Negative	13407 (38.09%)
Gender	Male	11559 (32.87%)
	Female	23608 (67.13%)
Age groups	Below 60 years	27676 (86.1%)
	60 years and above	4788 (13.9%)
Fever/Chills/Pyrexia	Absent	35175 (99.92%)
	Present	27 (0.08%)
Cough	Absent	35165 (99.89%)
	Present	37 (0.11%)
Sore Throat	Absent	26395 (74.98%)
	Present	8807 (25.02%)
Shortness of Breath	Absent	35192 (99.97%)
	Present	10 (0.03%)
Diarrhoea	Absent	35199 (99.99%)
	Present	3 (0.01%)
Muscle or Joint aches	Absent	35195 (99.98%)
	Present	7 (0.02%)
Malaise	Absent	35197 (99.99%)
	Present	2 (0.01%)
Fatigue or Lethargy	Absent	35201 (99.99%)
	Present	1 (0.01%)
Flu	Absent	35200 (99.99%)
	Present	2 (0.01%)
Vomiting or Nausea	Absent	35201 (99.99%)
	Present	2 (0.01%)
Any Comorbidity	Absent	25687 (72.97%)
	Present	9515 (27.03%)

*IQR: Interquartile Range
Recorded comorbidities included HIV/AIDS, Tuberculosis, Hypertension, Diabetes, Asthma, Obesity and Cancer, and Chronic Obstructive Pulmonary Disease (COPD)

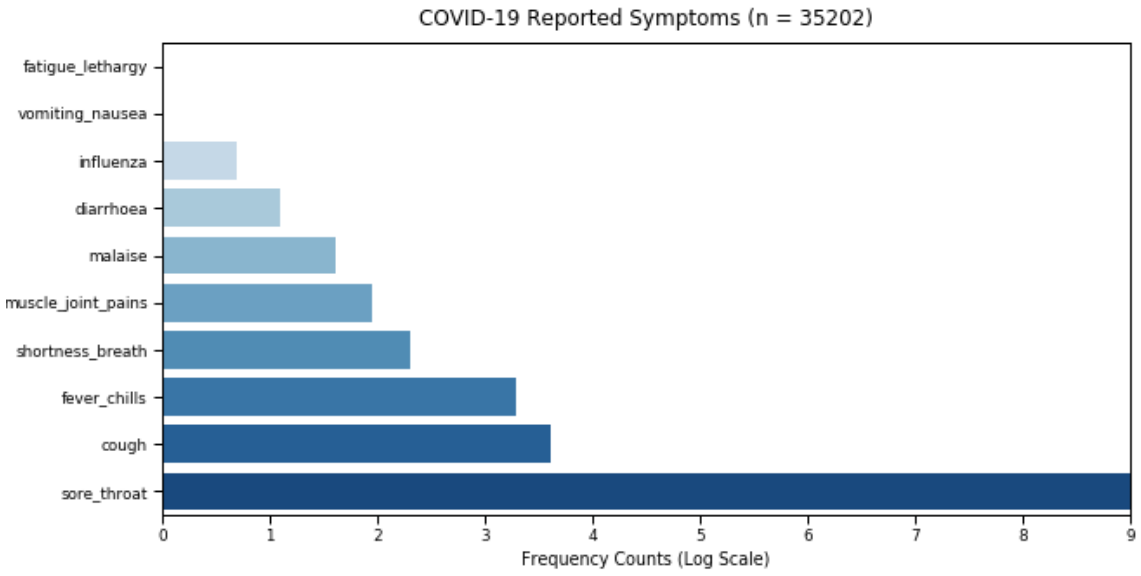


Figure 4.8: Frequency distribution of recorded symptoms on a log scale

4.2.1 Predictions using Balanced Datasets

From results presented in Table 4.3, all three classifiers (SVC, RF, and KNN) scored equally on accuracy (94%). However, accuracies were lower when predicting clinical outcomes from COVID-19 out-of-sample data. The KNN scored lowest on accuracy (59%) whereas the SVC attained the highest predictive accuracy though these differences were marginal. In both Malaria and COVID-19 out-of-sample data, SVC yielded the highest on sensitivity (100%) compared to KNN and RF.

Table 4.3: Performance Metrics on Balanced data (percentage scores on out-of-sample data).

	Malaria			Covid-19		
Performance Metric	SVC	RF	KNN	SVC	RF	KNN
Accuracy	0.941	0.94	0.942	0.62	0.611	0.59
MCC	0	0.242	0.231	0.026	0.025	0.039
Sensitivity/Recall	1	0.99	0.994	1	0.944	0.826
Precision (PPV)	0.941	0.949	0.947	0.62	0.622	0.628
F1-Measure	0.97	0.969	0.97	0.765	0.75	0.714

Classifier predicted values are presented in Table 4.4 below. At 97%, the SVC, RF, and KNN classifiers scored higher F1-measure when using Malaria out-of-sample data compared to COVID-19; 76%, 75% and 71% respectively.

The classifiers generally scored high in predicting positive outcomes averaging at 94%

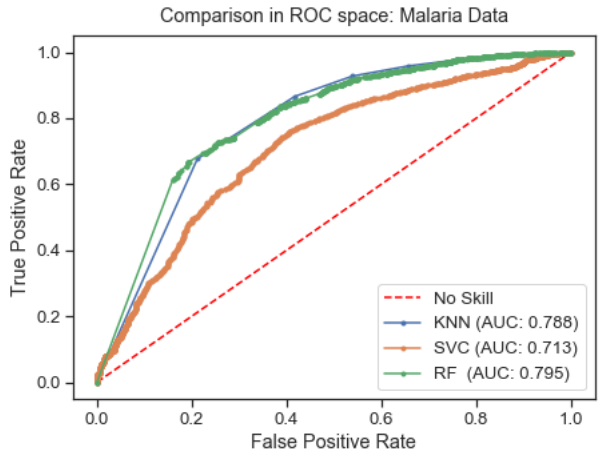
with the Malaria out-of-sample data. Moreover, compared to COVID-19 out-of-sample data, PPV scores were generally lower compared to Malaria data; 62.0% for SVC, RF at 62.4%, and KNN 62.8%.

Table 4.4: Confusion Matrices for Malaria and COVID-19: Balanced Data

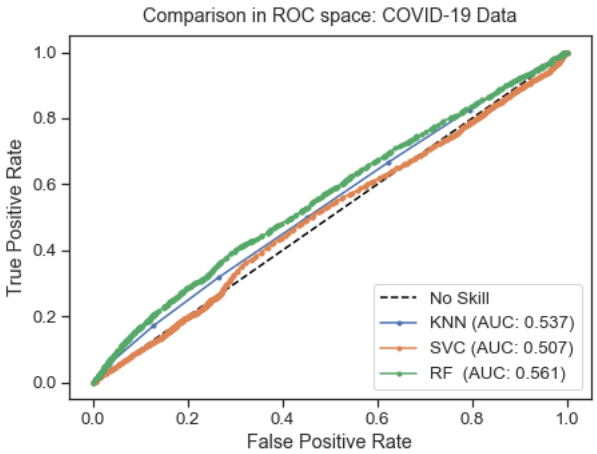
	Malaria					COVID-19				
Model/Classifier	**TN**	**FN**	**TP**	**FP**	**ER**	**TN**	**FN**	**TP**	**FP**	**ER**
SVC	0	0	9541	599	0.059	9	3	6536	4013	0.38
RF	89	98	9443	510	0.06	275	366	6173	3747	0.389
KNN	70	61	9480	529	0.058	827	1139	5400	3195	0.41

ER: Error Rate
The number of correct and incorrect predictions by SVC, RF KNN stratified by category.

Using AUC are performance metric, we observed a generally better performance in predicting clinical outcomes using Malaria out-of-sample data compared to COVID-19 where the SVC, KNN, and RF classifier predicted about 20% more accurately with Malaria data. These results are presented Figure 4.9 below



(a) Malaria (n=1389 per class)



(b) COVID-19 (n=8938 per class)

Figure 4.9: Classifier performance in ROC space

4.2.2 Predictions using Weighted Datasets: Imbalanced Learning

In this approach, the models were refitted on the same sample data this time accounting for distribution weights of clinical outcomes in model specification.

Observing results presented in Table 4.5, we observe that the SVM yielded a recall score of 100% on both datasets i.e. the model did not predict a single TN outcome from the 599 observations. The same poor prediction was observed in the COVID-19 data where the SVC classifier was only able to accurately predict 9 out of 4021 clinically negative observations. We also observe the SVC classifier predicted the highest number of clinically positive observations (n=9541).

Table 4.5: Performance Metrics on Weighted data (percentage scores on out-of-sample data)

	Malaria			Covid-19		
Performance Metric	SVC	RF	KNN	SVC	RF	KNN
MCC	0	0.227	0.239	0.026	0.084	0.032
Sensitivity/Recall	1	0.986	0.992	1	0.486	0.786
Precision (PPV)	0.941	0.949	0.948	0.62	0.663	0.628
F1-Measure	0.97	0.967	0.969	0.765	0.568	0.698

The RF model for COVID-19 had the lowest sensitivity with 48.6%. Accounting for target distribution weights, all three classifiers (SVC, RF, and KNN) scored higher PPV with Malaria compared to COVID-19 at 97% across. Results from error matrices are presented in Table 4.6 below.

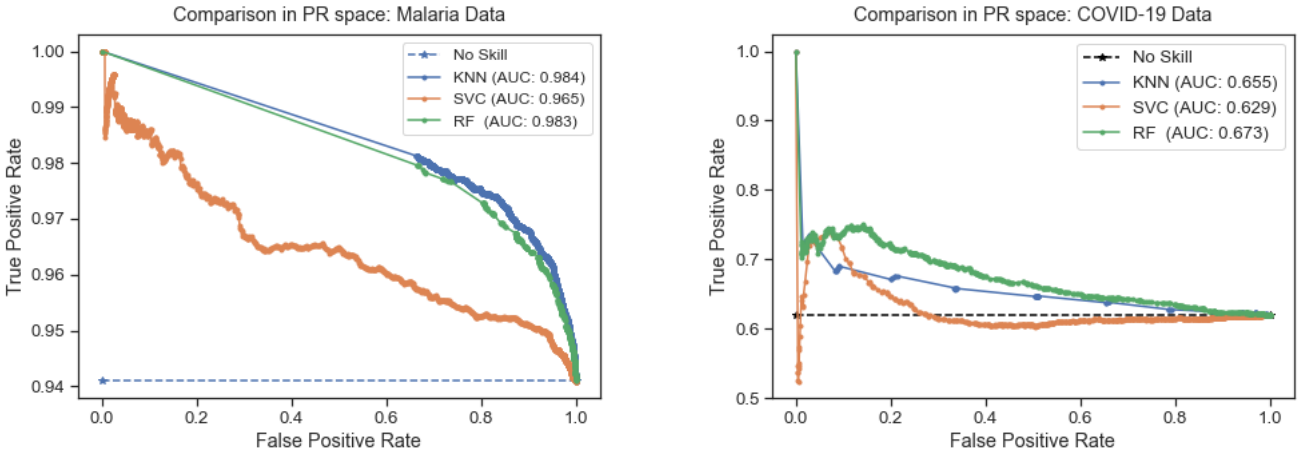
Table 4.6: Confusion Matrices for Malaria and COVID-19: Weighted Data

	Malaria					**COVID-19**				
Model/Classifier	**TN**	**FN**	**TP**	**FP**	**ER**	**TN**	**FN**	**TP**	**FP**	**ER**
SVC	0	0	9541	599	0.059	9	3	6536	4013	0.38
RF	94	134	9407	505	0.063	2371	3291	3248	1651	0.468
KNN	81	80	9461	518	0.059	927	1399	5140	3050	0.42

ER: Error Rate
The number of correct and incorrect predictions by SVC, RF KNN stratified by category.

With F1-measure as primary metric, we observe that SVM outperforms RF and KNN classifiers when modeled using COVID-19 data, 76%, 57% and 70% respectively. However, the scores remained similar (97%) when the models were subjected to the Malaria data.

Taking into account imbalances in the class distribution, we present the AUC results over the Precision-Recall space in Figure 4.10.



(a) Malaria Positive:Negative (1590:100)

(b) COVID-19 Positive:Negative (162:100)

Figure 4.10: Classifier performance in Precision-Recall space

We observe higher predictions in Malaria out-of-sample data compared to COVID-19. The RF AUC was higher with Malaria out-of-sample data testing (98%) versus to 67% with COVID-19. Though marginally lower compared to RF, AUC for SVC was still higher with the Malaria data. The same is observed with KNN classifier where the yield was 98% using Malaria data. The same algorithm correctly predicted 65.5% on COVID-19 outcomes using out-of-sample data.

4.3 Novelty Detection results

For novelty detection, we first examined the distribution of the target in both the training and out-of-sample data. We observed severe skewness in Malaria data. The positive/negative ratio was 16:1 in both the training and out-of-sample Malaria data. Negative observations (minority) were categorized as outliers to predict this category in the out-of-sample data. In Table 4.7 below, we present results from unary classification approaches.

Table 4.7: Performance metrics using Unary classification on Malaria data (percentage scores on out-of-sample data)

Performance Metric	OCSVM	iForest
MCC	-0.05	0.037
Specificity/TNR	0.232	0.09
F1-Measure	0.071	0.092

We evaluate classifier performance in predicting 599 out-of-sample malaria-negative observations. Results are presented in the Confusion Matrix in Table 4.8 below. Worth noting, OCSVM predicted more negative observation compared to the iForest; 139 versus 54 respectively hence a higher specificity score.

Table 4.8: Confusion Matrix from Unary classification: Malaria data

Model/Classifier	**TN**	**FN**	**TP**	**FP**
OneClassSVM	139	3162	6376	460
iForest	54	517	9024	545

5 Discussion

In this Chapter, we discuss empirical findings to the research objectives proposed in Chapter 1. We present both a quantitative as well as a qualitative discussion of the empirical data reported in Chapter 4. In the last section is a brief discussion of limitations encountered during the research.

5.1 Malaria and COVID-19 Surveillance Data Profiles

Between January 2015 and December 2019, Malaria prevalence stood at 27% (n=58,692). Among those clinically diagnosed with Malaria, 57.2% (n=33,198) were from Limpopo province followed by Mpumalanga (17%, n=9,623). Northern Cape had the least Malaria cases. Two possible explanations support this result. First, the climatic conditions favor Malaria prevalence in this area. Secondly, being border provinces, Mpumalanga and Limpopo have a higher immigrant population from neighboring countries of Mozambique and Zimbabwe which are Malaria-endemic. These findings are in agreement with statistics reported in Guidelines for the Treatment of Malaria in South Africa [24].

From the COVID-19 summary statistics reported in Table 4.2, fever was prevalent in less than 1% (n=27) of the population; 8 observed in COVID-19 positive population with 19 in the negative population. The same was observed with cough (n=37) and fatigue/lethargy. The most prevalent symptom in the population with clinical suspicion of COVID-19 was *sore throat* (25%, n=8,807).
Among the COVID-19 positive cases, *sore throat* was prevalent in 25% (n=5,450) of the population. On investigating the relationship between COVID-19 clinical outcomes and presence/absence of *sore throat*, we did not seem to find statistical evidence in the data to suggest existence of an association between COVID-19 clinical outcomes and the presence/absence of *sore throat* (p-value = 0.94).

In disease surveillance, attention is generally accorded to patients with a clinically positive as opposed to a negative outcome. This variation in capturing data creates gaps in data over time. Because of this, we considered only complete-cases for the Malaria analytical

dataset. Several observations were dropped, of which, the majority had a negative clinical outcome, reducing the dataset to 40,557 observations. Therefore, reporting stratified statistics from this biased sample would lead to over-estimating Malaria prevalence.

We investigated the association of risk-factors with clinical outcomes. Although people of all ages are at risk, the older population i.e those above sixty years of age are more susceptible to COVID-19 infection [83]. However, in the analytical dataset, the older population accounted for 16% (n=3422) of the 21,795 COVID-19 positive cases with the majority under sixty years of age. A 2020 systematic review by Yang et al. [84] suggests an association between age and comorbidities among COVID-19 patients; suggesting age and comorbidities as risk factors. The authors reported comorbidities to be more prevalent in high-risk populations i.e older patients who report the presence of at least one comorbidity.
In this study, among those who tested positive for COVID-19, comorbidities were registered 27.2% (n=5,935) of the population. This prevalence was similar among those who tested negative for COVID-19 (26.7%, n=3,580). Using Mantel-Haenszel estimates we observed that those who registered at least one comorbidity had the same risk of testing positive for COVID-19 as those who tested negative (OR[1] = 1.02; 95% CI[2]: 0.98, 1.08). However, adjusting for the effects of age, the older population had a 20% (OR = 1.20; 95% CI: 1.06, 1.36) increased risk of testing positive for COVID-19 compared to the younger population. Noteworthy, out of the 35,202 clinically suspected COVID-19 cases in this research, we observed 6.4% (n=2,256) of the population were high-risk i.e sixty years or older, and had at least one comorbidity registered. Although we found overwhelming statistical evidence suggesting an association existed between *gender* and COVID-19 clinical outcomes (p-value < .01), we did not find any epidemiological evidence to support this finding.

5.2 Classification and Prediction of Clinical Outcomes Malaria and COVID-19

We applied three different approaches to predict probable cases for Malaria and COVID-19. The first two approaches to classification and prediction involved resampling the data and modeling solutions with the data as-is but accounting for distribution weights of the target. In these two approaches, our primary measure of performance was the AUC. In the third, a novelty approach, *specificity* scores were the primary evaluation metric.

Three models were subjected to data in which the target had equal representation. With 1389 observations in each class, the SVM, RF, and KNN models performed better with

[1]Odds Ratio
[2]Confidence Interval

Malaria compared to COVID-19 out-of-sample data. Though samples in each class were incomparable, 1389 for Malaria and 8939 for COVID-19, this difference did not seem to alter the predictive power of models. As illustrated in Figure 4.9, at 80%, the RF classifier recorded the highest predictive power compared to SVM and KNN (75.4% and 78.8% respectively). We also observed, with COVID-19 out-of-sample data, the models did not perform any better compared to guess i.e models with no-skill.

As illustrated using Precision-Recall and ROC curves, we observed an overall improvement in classification and prediction when classifiers account for distribution weights of the target in predictions. However, scores with COVID-19 data though remained lower compared to Malaria data. Using COVID-19 data, significant improvements were noticed with the RF, KNN, and SVC classifiers. The RF AUC improved from 56.1% to 67.3% and the KNN from 53.7% to 65.5% and the SVC from 50.7% to 62.9%. This twelve percent gain may be attributed the increased number of observations that the models were trained on, hence learning more from the data to improve predictions. We also observed the prediction error rate was generally higher with RF classifier at though compared to KNN and SVM, this difference was marginal.

In consideration of distributions in the target as presented in Section 4.1, novelty approaches were employed only on malaria data. Comparing classification and prediction of negative observations, the OCSVM performed better than the iForest. Out of 599 out-of-sample negative observations, the OCSVM predicted 139 correctly attaining a specificity score of 0.23 versus 0.09 (TN: 54). These results are unsatisfactory as highlighted prior by the correlation matrix in Section 36.

5.3 Qualitative Evaluation of Results

While in this study we employed disease predictors in singularity, a viable option as proposed in literature is to consider features in combination. For example, in the predictive diagnosis of Malaria from symptoms, individuals reporting a fever and had a previous Malaria episode in their household are more likely to yield a positive Malaria result. This approach will likely increase the predictive power of models compared to models that consider symptom information distinctively.

Whereas a clinical outcome was available in all observations, symptomatic information was less informative in the COVID-19 analytical dataset and completely absent in the Malaria dataset. On the contrary, while the Malaria dataset consisted of laboratory markers (red blood cell counts), this information was completely lacking in the COVID-19 data.

This dissimilarity in data structures was more pronounced in results where models run on out-of-sample data performed overwhelmingly better on Malaria data compared to COVID-19. Research has shown that red blood cell counts are an indication of infection in humans, explaining the variation in prediction results.

When implementing supervised ML predictive models, it is necessary to identify beforehand relationships between features and targets as well as inter-feature correlations. In both Malaria and COVID-19 analytical datasets, we identified weaker correlations between features and targets. Scant prevalences made it difficult to deduce informative correlations between clinical outcomes with selected features. For this reason, we could not determine distinct predictors in the COVID-19 dataset and to a fair amount in the Malaria dataset. One approach to mitigate this pitfall is to employ comprehensive data quality assessments at data-collection points. Embedding automated integrity and validation checks in data collection instruments enforces the tidy-data model proposed by Wickham [65] described in Chapter 3.

In this research, we described the current surveillance data structures and profiles for Malaria and COVID-19 at the NHLS, South Africa. While clinical outcomes were observed for all observations, other attributes in the surveillance datasets were inconsistent with gaps. For example, between 2015 and 2019, disease symptoms, treatment information, travel and contact history, and case notes were not recorded in up to 100% observations in Malaria surveillance data. This hugely limited how much information we were able to use for classification and prediction. In a similar way, among the demographic attributes observed in the COVID-19 surveillance data, location information was lacking. Therefore, stratified analysis to determine cases per province to inform the DoH on resource allocation for COVID-19 was not possible.

In summary, examining results from classification and prediction presented in Chapter 4, we can summarily state that the classifiers did not perform the same using surveillance data on Malaria and COVID-19. Therefore, supervised ML approaches may not be appropriate in situations where weaker or no correlations exist between attributes and targets. In this case, total dependence on laboratory results may not be an option.

5.4 Limitations

Though differing on the number of observations, we observed a higher prevalence of COVID-19 (38%, n=6,539) over a one-year interval (March 2020 to March 2021). However, because locality information was lacking, a stratified analysis by location was not possible.

Although most cases are mild, three key symptoms, fever, cough, and fatigue have been largely reported in COVID-19 patients [35], we noted an overwhelming number of observations where these symptoms were not registered hence affecting machine supervised learning

While we evaluated the performance of selected ML algorithms, a comparison with the current rule-set methods used to categorize disease status at the NICD was not possible. With the COVID-19 pandemic largely dictating remote operations for all engagements in this research, we were unable to physically benchmark these rule-set classifiers. However, resources permitting, future work should consider examining how these rule-set methodologies compare with ML alternatives when exposed to the same data structures.

6 Conclusion and Future Directions

In this study, we not only described the current Malaria and COVID-19 surveillance data structures but also used supervised ML approaches to predict clinical outcomes for these two conditions at the NICD, South Africa. This study showed the impact of data quality in disease surveillance with respect to predictive modeling for Malaria and COVID-19 medical conditions. Individual demographic characteristics, reported and recorded signs and symptoms among other attributes hold vital information for syndromic disease surveillance. However, these were lacking. The data were characterized by large proportions of incompleteness.

This study shows the impact of data quality in disease surveillance with respect to predictive modeling. We showed supervised ML can have high predictive power in determining clinical outcomes for Malaria. However, the same can not be said of COVID-19 where performance was suboptimal. We identified this performance inconsistency to emanate largely from data quality. Though the data structures in the two datasets were dissimilar, they were characterized by large proportions of incompleteness as reported in Table 7.2 of Appendix 7. In accordance, weaker correlations as those identified in the COVID-19 data were indicative of final suboptimal results. This was partially explained by missing symptoms for the COVID-19 asymptomatic cases.

Speaking generally, these models could be improved by firstly streamlining methods aimed at improving data quality and secondly, collecting more data defining disease characteristics. A future consideration could consider the use of unsupervised ML to explore patterns in these surveillance data. We can Summarily conclude that the current NMC surveillance data are not yet ready to benefit from ML supervised ML approaches. While we showed promising results using ML approaches on Malaria data, these findings can not be generalized in other settings. Therefore, more resources need to be invested in the current NMC surveillance system to support proper monitoring and syndromic surveillance of COVID-19 and Malaria at the NHLS.

Bibliography

[1] Julius Nyerere Odhiambo and Benn Sartorius. "Spatio - temporal modelling assessing the burden of malaria in affected low and middle-income countries : a scoping review". In: 2016 (2018), pp. 6–11. DOI: 10.1136/bmjopen-2018-023071.

[2] World Health Organization, ed. *World malaria report 2019*. World Health Organization, 2019. ISBN: 9789241565721.

[3] Richard E. Cibulskis et al. "Malaria: Global progress 2000 - 2015 and future challenges". In: *Infectious Diseases of Poverty* 5.1 (2016), pp. 1–8. ISSN: 20499957. DOI: 10.1186/s40249-016-0151-8.

[4] Janet Hemingway et al. "Tools and Strategies for Malaria Control and Elimination: What Do We Need to Achieve a Grand Convergence in Malaria?" In: *PLoS Biology* 14.3 (2016), pp. 1–14. ISSN: 15457885. DOI: 10.1371/journal.pbio.1002380.

[5] Penghui Yang and Xiliang Wang. *COVID-19: a new challenge for human beings*. 2020. DOI: 10.1038/s41423-020-0407-x.

[6] Srikanth Umakanthan et al. *Origin, transmission, diagnosis and management of coronavirus disease 2019 (COVID-19)*. 2020. DOI: 10.1136/postgradmedj-2020-138234.

[7] *Weekly epidemiological update - 2 March 2021*. URL: https://www.who.int/publications/m/item/weekly-epidemiological-update---2-march-2021.

[8] Addis Adera Gebru et al. "Global burden of COVID-19: Situationalanalyis and reviewGlobal burden of COVID-19: Situationalanalyis and review". In: *North America* 555.17 (), p. 403. ISSN: 1093-2607. DOI: 10.3233/HAB-200420.

[9] Shabir Ahmad Lone and Aijaz Ahmad. *COVID-19 pandemic–an African perspective*. 2020. DOI: 10.1080/22221751.2020.1775132.

[10] Srikanta Basu and Puneet Kaur Sahi. "Malaria: An Update". In: *Indian Journal of Pediatrics* 84.7 (2017), pp. 521–528. ISSN: 09737693. DOI: 10.1007/s12098-017-2332-2.

[11] *francis.pdf*.

[12] Pan Zhai et al. "Since January 2020 Elsevier has created a COVID-19 resource centre with free information in English and Mandarin on the novel coronavirus COVID- 19 . The COVID-19 resource centre is hosted on Elsevier Connect , the company ' s public news and information". In: *journal of Antimicrobial agents* 55.January (2020), pp. 105–955.

[13] Mahmoud M. Berekaa. "Insights into the COVID-19 pandemic: Origin, pathogenesis, diagnosis, and therapeutic interventions". In: *Frontiers in bioscience (Elite edition)* 13 (2021), pp. 117–139. ISSN: 19450508. DOI: 10.2741/874.

[14] Oumaima Terrada et al. "Classification and Prediction of atherosclerosis diseases using machine learning algorithms". In: *2019 International Conference on Optimization and Applications, ICOA 2019* (2019). DOI: 10.1109/ICOA.2019.8727688.

[15] Dimitris Spathis and Panayiotis Vlamos. "Diagnosing asthma and chronic obstructive pulmonary disease with machine learning". In: *Health Informatics Journal* 25.3 (2019), pp. 811–827. ISSN: 17412811. DOI: 10.1177/1460458217723169.

[16] Najeed Ahmed Khan et al. "Unsupervised identification of malaria parasites using computer vision". In: *Pakistan Journal of Pharmaceutical Sciences* 30.1 (2017), pp. 223–228. ISSN: 1011601X.

[17] Angel Molina et al. "Sequential classification system for recognition of malaria infection using peripheral blood cell images". In: *Journal of Clinical Pathology* 73.10 (2020), pp. 665–670. ISSN: 14724146. DOI: 10.1136/jclinpath-2019-206419.

[18] Mahdieh Poostchi et al. "Image analysis and machine learning for detecting malaria". In: *Translational Research* 194 (2018), pp. 36–55. ISSN: 18781810. DOI: 10.1016/j.trsl.2017.12.004. URL: https://doi.org/10.1016/j.trsl.2017.12.004.

[19] Zeno Bisoffi, Federico Gobbi, and Jef Van Den Ende. "Rapid diagnostic tests for malaria". In: *BMJ (Online)* 348.1 (2014), pp. 66–78. ISSN: 17561833. DOI: 10.1136/bmj.g3846.

[20] Jaspreet Toor et al. "The design of schistosomiasis monitoring and evaluation programmes: The importance of collecting adult data to inform treatment strategies for Schistosoma mansoni". In: *PLoS neglected tropical diseases* 12.10 (2018), e0006717. ISSN: 19352735. DOI: 10.1371/journal.pntd.0006717.

[21] World Health Organization. "Monitoring and evaluation of health systems strengthening: An operational framework". In: November (2009), pp. 1–26. ISSN: 16549880. DOI: 10.3402/gha.v6i0.20001. arXiv: WHOLibraryCataloguing-in-PublicationData. URL: http://www.who.int/healthinfo/HSS{_}MandE{_}framework{_}Nov{_}2009.pdf{\%}5Cnwww.cpc.inc.edu/measure{\%}5Cnhttp://www.who.int/healthmetrics/documents/hmn{_}framework200803.pdf.

[22] Jaspreet Toor et al. "Are we on our way to achieving the 2020 goals for schistosomiasis morbidity control using current world health organization guidelines?" In: *Clinical Infectious Diseases* 66 (2018), S245–S252. ISSN: 15376591. DOI: 10.1093/cid/ciy001.

[23] Editorial Office. "Standard Operating Procedures for paper based reporting of notifiable medical conditions (NMC)". In: *Professional Nursing Today* 21.4 (2017), pp. 60–72. ISSN: 1607-6672.

[24] Ministry O F Health. "National Guidelines for the Treatment of Malaria in Rwanda". In: (2013).

[25] Sun L Rei Yan, Felipe Wakasuqui, and Carsten Wrenger. "Point-of-care tests for malaria : speeding up the diagnostics at the bedside and challenges in malaria cases detection". In: *Diagnostic Microbiology & Infectious Disease* 98.3 (2020), p. 115122. ISSN: 0732-8893. DOI: 10.1016/j.diagmicrobio.2020.115122. URL: https://doi.org/10.1016/j.diagmicrobio.2020.115122.

[26] Afoma Mbanefo and Nirbhay Kumar. "Evaluation of malaria diagnostic methods as a key for successful control and elimination programs". In: *Tropical Medicine and Infectious Disease* 5.2 (2020). ISSN: 24146366. DOI: 10.3390/tropicalmed5020102.

[27] Lara Cotta Amaral et al. "Ribosomal and non-ribosomal PCR targets for the detection of low-density and mixed malaria infections". In: *Malaria Journal* 18.1 (2019), pp. 1–14. ISSN: 14752875. DOI: 10.1186/s12936-019-2781-3. URL: https://doi.org/10.1186/s12936-019-2781-3.

[28] Kenji O. Mfuh et al. "A comparison of thick-film microscopy, rapid diagnostic test, and polymerase chain reaction for accurate diagnosis of Plasmodium falciparum malaria". In: *Malaria Journal* 18.1 (2019), pp. 1–8. ISSN: 14752875. DOI: 10.1186/s12936-019-2711-4. URL: https://doi.org/10.1186/s12936-019-2711-4.

[29] Tham J.M. et al. "Detection and species determination of malaria parasites by PCR: Comparison with microscopy and with ParaSight-F and ICT Malaria Pf tests in a clinical environment". In: *Journal of Clinical Microbiology* 37.5 (1999), pp. 1269–1273. ISSN: 0095-1137. URL: http://www.embase.com/search/results?subaction=viewrecord{\&}from=export{\&}id=L29187782{\%}0Ahttp://sfx.library.uu.nl/utrecht?sid=EMBASE{\&}issn=00951137{\&}id=doi:{\&}atitle=Detection+and+species+determination+of+malaria+parasites+by+PCR{\%}3A+Comparison+with+microscopy+and.

[30] Ryoko Makuuchi et al. "The correlation between malaria RDT (Paracheck pf.®) faint test bands and microscopy in the diagnosis of malaria in Malawi". In: *BMC Infectious Diseases* 17.1 (2017), pp. 1–9. ISSN: 14712334. DOI: 10.1186/s12879-017-2413-x.

[31] Pedro Berzosa et al. "Comparison of three diagnostic methods (microscopy, RDT, and PCR) for the detection of malaria parasites in representative samples from Equatorial Guinea 11 Medical and Health Sciences 1108 Medical Microbiology". In: *Malaria Journal* 17.1 (2018), pp. 1–12. ISSN: 14752875. DOI: 10.1186/s12936-018-2481-4. URL: https://doi.org/10.1186/s12936-018-2481-4.

[32] *Malaria – National Department of Health*. URL: https://www.health.gov.za/malaria/ (visited on 10/26/2021).

[33] Dr R B McFee. "COVID-19 Laboratory Testing/CDC Guidelines". In: *Disease-a-Month* 66.9 (2020), p. 101067. ISSN: 15578194. DOI: 10.1016/j.disamonth.2020.101067. URL: /pmc/articles/PMC7409931/https://www-ncbi-nlm-nih-gov.ezproxy.nihlibrary.nih.gov/pmc/articles/PMC7409931/.

[34] Stephen L. Hoffman and Judith E. Epstein. "Malaria vaccines". In: *Essential Malariology, Fourth Edition* 4 (2017), pp. 313–325. DOI: 10.1201/9780203756621.

[35] Chaolin Huang et al. "Clinical features of patients infected with 2019 novel coronavirus in Wuhan, China". In: *The Lancet* 395.10223 (2020), pp. 497–506. ISSN: 1474547X. DOI: 10.1016/S0140-6736(20)30183-5. URL: https://isaric.tghn.org/protocols/.

[36] Zhiru Gao et al. *A systematic review of asymptomatic infections with COVID-19*. 2020. DOI: 10.1016/j.jmii.2020.05.001. URL: https://doi.org/10.1016/j.jmii.2020.05.001.

[37] Fei Zhou et al. "Clinical course and risk factors for mortality of adult inpatients with COVID-19 in Wuhan, China: a retrospective cohort study". In: *The Lancet* 395.10229 (2020), pp. 1054–

1062. ISSN: 1474547X. DOI: 10.1016/S0140-6736(20)30566-3. URL: https://doi.org/10.1016/.

[38] Hasan Ejaz et al. "COVID-19 and comorbidities: Deleterious impact on infected patients". In: *Journal of Infection and Public Health* 13.12 (2020), pp. 1833–1839. ISSN: 1876035X. DOI: 10.1016/j.jiph.2020.07.014.

[39] Morgan Spencer Gold et al. "COVID-19 and comorbidities: a systematic review and meta-analysis". In: *Postgraduate Medicine* (2020), pp. 1–7. ISSN: 19419260. DOI: 10.1080/00325481.2020.1786964.

[40] Beatriz Galatas, Quique Bassat, and Alfredo Mayor. "Malaria Parasites in the Asymptomatic: Looking for the Hay in the Haystack". In: *Trends in Parasitology* 32.4 (2016), pp. 296–308. ISSN: 14715007. DOI: 10.1016/j.pt.2015.11.015. URL: http://dx.doi.org/10.1016/j.pt.2015.11.015.

[41] *WHO EMRO | Control of malaria outbreak due to Plasmodium vivax in Aswan Governorate, Egypt | Volume 22, issue 4 | EMHJ volume 22, 2016*. URL: http://www.emro.who.int/emhj-volume-22-2016/volume-22-issue-4/control-of-malaria-outbreak-due-to-plasmodium-vivax-in-aswan-governorate-egypt.html (visited on 12/30/2020).

[42] Matthew B. Laurens. "RTS,S/AS01 vaccine (Mosquirix™): an overview". In: *Human Vaccines and Immunotherapeutics* (2019). ISSN: 2164554X. DOI: 10.1080/21645515.2019.1669415.

[43] Caroline Theunissen et al. *Falciparum malaria in patient 9 years after leaving malaria-endemic area*. 2009. DOI: 10.3201/eid1501.080909.

[44] James M. Crutcher and Stephen L. Hoffman. *Medical Microbiology for Malaria*. Ed. by Baron S. 4th. University of Texas Medical Branch at Galveston, 1996. ISBN: 0963117211. URL: https://www.ncbi.nlm.nih.gov/books/NBK8584/.

[45] Yuan Luo et al. "Using machine learning to predict laboratory test results". In: *American Journal of Clinical Pathology* 145.6 (2016), pp. 778–788. ISSN: 19437722. DOI: 10.1093/ajcp/aqw064.

[46] Pranav Rajpurkar, Vinaya Polamreddi, and Anusha Balakrishnan. "Malaria likelihood prediction by effectively surveying households using deep reinforcement learning". In: *arXiv* Nips (2017). arXiv: 1711.09223.

[47] You Won Lee, Jae Woo Choi, and Eun-Hee Shin. "Machine learning model for predicting malaria using clinical information". In: *Computers in Biology and Medicine* 129.November 2020 (2020), p. 104151. ISSN: 00104825. DOI: 10.1016/j.compbiomed.2020.104151. URL: https://doi.org/10.1016/j.compbiomed.2020.104151.

[48] Ahmed Hamed, Ahmed Sobhy, and Hamed Nassar. "Accurate Classification of COVID-19 Based on Incomplete Heterogeneous Data using a KNN Variant Algorithm". In: *Arabian Journal for Science and Engineering* (2021). ISSN: 21914281. DOI: 10.1007/s13369-020-05212-z.

[49] Celestine Iwendi et al. "COVID-19 patient health prediction using boosted random forest algorithm". In: *Frontiers in Public Health* 8 (2020). ISSN: 22962565. DOI: 10.3389/fpubh.2020.00357.

[50] Tenth (World Health Organization) Revision and Second Edition. "ICD-10". In: *WHO Library Cataloguing-in-Publication Data* 3.2nd ed. (2004), p. 131. URL: www.who.int/classifications.

[51] Gareth James et al. *An Introduction to Statistical Learning*. Ed. by First. First: Springer, Dordrecht, 2017, pp. 127–. ISBN: 9781461471370. DOI: 10.1007/978-1-4614-7138-7. URL: http://www.springer.com/series/417.

[52] Yan Qing Zhang and Jagath C. Rajapakse. "Machine Learning in Bioinformatics". In: *Machine Learning in Bioinformatics* 7.1 (2008), pp. 1–456. DOI: 10.1002/9780470397428.

[53] Amr E Mohamed. "Comparative Study of Four Supervised Machine Learning Techniques for Classification". In: *International Journal of Applied Science and Technology* 7.2 (2017), pp. 5–18. ISSN: 2221-0997.

[54] Vishal Gour et al. "Improve Performance of Extract, Transform and Load ({ETL)} in Data Warehouse". In: *International Journal on Computer Science & Engineering* 1.3 (2010), pp. 786–789. ISSN: 09753397.

[55] Max Bramer. *Principles of Data Mining*. Third. London: Springer International Publishing, 2016, p. 239. ISBN: 9781447173076. DOI: 10.1007/978-1-4471-7307-6. URL: http://www.springer.com/series/7592.

[56] Gregory Piatetsky-Shapiro. "Discovery, Analysis, and Presentation of Strong Rules". In: *Knowledge Discovery in Databases*. Ed. by Gregory Piatetsky-Shapiro and William J Frawley. AAAI/MIT Press, 1991, pp. 229–248. ISBN: 0-262-62080-4. URL: http://dblp.uni-trier.de/db/books/collections/PiatetskyF91.html{\#}Piatetsky91.

[57] Alejandro Barredo Arrieta et al. "Explainable Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI". In: *Information Fusion* 58 (2020), pp. 82–115. ISSN: 15662535. DOI: 10.1016/j.inffus.2019.12.012. arXiv: 1910.10045.

[58] Hadi Hormozi, Elham Hormozi, and Hamed Rahimi Nohooji. "The Classification of the Applicable Machine Learning Methods in Robot Manipulators". In: *International Journal of Machine Learning and Computing* 2.5 (2013), pp. 560–563. ISSN: 20103700. DOI: 10.7763/ijmlc.2012.v2.189.

[59] Miroslav Kubat. *An Introduction to Machine Learning*. Second. Springer International Publishing, 2017, pp. 273–297. ISBN: 9783319639123. DOI: 10.1007/978-3-319-63913-0.

[60] Steven S Skiena. *The Data Science Design Manual*. Ed. by Gries David, Orit Hazzan, and Fred B. Schneider. First. Switzerland: Springer, Cham, 2017, pp. 212 –225, 372 –375. ISBN: 9783319554433. DOI: 10.1007/978-3-319-55444-0. URL: http://www.springer.com/series/3191.

[61] G.(Purdue University) Elfeky et al. "Record Linkage : A Machine Learning Approach , A Toolbox , and a Digital Government Web Service". In: *Purdue e-Pubs* (2003), pp. 1–29.

[62] Aik Choon Tan and David Gilbert. "Ensemble machine learning on gene expression data for cancer classification." In: *Applied bioinformatics* 2.3 Suppl (2003), pp. 1–10. ISSN: 11755636.

[63] Miroslav Kubat. *An Introduction to Machine Learning*. Springer, Dordrecht. ISBN: 9783319639123.

[64] *South Africa - Soils | Britannica*. URL: https://www.britannica.com/place/South-Africa/Soils#ref44026 (visited on 10/19/2021).

[65] Hadley Wickham. "Tiday Data". In: *Journal of Statistical Software* 59.10 (2014), pp. 1–23. URL: http://www.jstatsoft.org/.

[66] Douglas P. Jutte, Leslie L. Roos, and Marni D. Brownell. "Administrative Record Linkage as a Tool for Public Health Research". In: *Annual Review of Public Health* 32.1 (2011), pp. 91–108. ISSN: 0163-7525. DOI: 10.1146/annurev-publhealth-031210-100700.

[67] Alexandru Niculescu-mizil et al. "Winning the KDD Cup Orange Challenge with Ensemble Selection . Winning the KDD Cup Orange Challenge with Ensemble Selection". In: January (2009).

[68] Lorenzo Beretta and Alessandro Santaniello. "Nearest neighbor imputation algorithms: A critical evaluation". In: *BMC Medical Informatics and Decision Making* 16.Suppl 3 (2016). ISSN: 14726947. DOI: 10.1186/s12911-016-0318-z. URL: http://dx.doi.org/10.1186/s12911-016-0318-z.

[69] *sklearn.impute.SimpleImputer*. URL: https://www.scikit-learn.org/stable/modules/generated/sklearn.impute.SimpleImputer.html (visited on 10/19/2021).

[70] Shahadat Uddin et al. "Comparing different supervised machine learning algorithms for disease prediction". In: *BMC Medical Informatics and Decision Making* 19.1 (2019), pp. 1–16. ISSN: 14726947. DOI: 10.1186/s12911-019-1004-8.

[71] *ML Studio (classic): One-Class Support Vector Machine - Azure | Microsoft Docs*. URL: https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/one-class-support-vector-machine (visited on 02/25/2021).

[72] Dubravko Miljković. "Review of Novelty Detection Methods". In: May 2010 (2016).

[73] Yanmin Sun, Andrew K.C. Wong, and Mohamed S. Kamel. "Classification of imbalanced data: A review". In: *International Journal of Pattern Recognition and Artificial Intelligence* 23.4 (2009), pp. 687–719. ISSN: 02180014. DOI: 10.1142/S0218001409007326.

[74] Victor Fragoso et al. "One-class slab support vector machine". In: *Proceedings - International Conference on Pattern Recognition* 0 (2016), pp. 420–425. ISSN: 10514651. DOI: 10.1109/ICPR.2016.7899670. arXiv: 1608.01026.

[75] Fei Tony Liu, Kai Ming Ting, and Zhi Hua Zhou. "Isolation forest". In: *Proceedings - IEEE International Conference on Data Mining, ICDM* (2008), pp. 413–422. ISSN: 15504786. DOI: 10.1109/ICDM.2008.17.

[76] E. Burnaev, P. Erofeev, and D. Smolyakov. "Model selection for anomaly detection". In: *arXiv* December (2017). ISSN: 23318422. DOI: 10.1117/12.2228794. arXiv: 1707.03909.

[77] Jiang Tian et al. "Local density one-class support vector machines for anomaly detection". In: *Nonlinear Dynamics* 64.1-2 (2011), pp. 127–130. ISSN: 0924090X. DOI: 10.1007/s11071-010-9851-y.

[78] Sabri Boughorbel, Fethi Jarray, and Mohammed El-Anbari. "Optimal classifier for imbalanced data using Matthews Correlation Coefficient metric". In: *PLoS ONE* 12.6 (2017), pp. 1–17. ISSN: 19326203. DOI: 10.1371/journal.pone.0177678.

[79] Davide Chicco and Giuseppe Jurman. "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation". In: *BMC Genomics* 21.1 (2020), pp. 1–13. ISSN: 14712164. DOI: 10.1186/s12864-019-6413-7.

[80] Mahsa Taheri, Néhémy Lim, and Johannes Lederer. "Balancing Statistical and Computational Precision and Applications to Penalized Linear Regression with Group Sparsity". In: (2016), pp. 233–240. arXiv: 1609.07195. URL: http://arxiv.org/abs/1609.07195.

[81] Takaya Saito and Marc Rehmsmeier. "The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets". In: *PLoS ONE* 10.3 (2015), pp. 1–21. ISSN: 19326203. DOI: 10.1371/journal.pone.0118432.

[82] Brownlee Jason. *How to Use ROC Curves and Precision-Recall Curves for Classification in Python*. URL: https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/ (visited on 02/23/2021).

[83] *COVID-19 advice - High risk groups | WHO Western Pacific*. URL: https://www.who.int/westernpacific/emergencies/covid-19/information/high-risk-groups (visited on 04/25/2021).

[84] Jing Yang et al. "Prevalence of comorbidities and its effects in patients infected with SARS-CoV-2: a systematic review and meta-analysis." In: *International Journal of Infectious Diseases* 94.March (2020), pp. 91–95. URL: https://pubmed.ncbi.nlm.nih.gov/32173574/.

7 Supplementary Tables and Graphs

7.1 Missing Value Report - Malaria Data

Table 7.1: Proportion of missing data - Malaria raw dataset

Attribute	Proportion	Attribute	Proportion
case_notes	1	travel_history_available_ind	0.996867
travel2_city_town_name	1	malaria_parasite_load	0.981798
travel1_city_town_name	1	white_cells_counted	0.976128
date_last_vaccination	0.999972	malaria_count	0.808824
travel2_province_name	0.999972	percentage_infestation	0.78261
date_travel2_start	0.999945	red_cells_counted	0.780734
date_travel2_end	0.999931	parasites_counted	0.776542
travel2_country_name	0.999926	red_cell_count	0.271325
treatment2	0.999399	white_cell_count	0.267194
travel1_province_name	0.999335	province_name	0.006317
pregnant_ind	0.998896	district_name	0.000892
date_travel1_end	0.998738	sub_district_name	0.000892
date_travel1_start	0.99872	facility_name	0.000892
travel1_country_name	0.998568	in_out_patient	0.000055
symptom2	0.997643	date_specimen	0.000014
symptom1	0.997435	test_result_text	0
treatment1	0.997214	age_tested_years	0
vaccination_status	0.997089	gender	0
contact_history	0.996932	episode_no	0

7.2 Missing Value Report - COVID-19 Data

Table 7.2: Proportion of missing data - COVID-19 raw dataset

Attribute	Proportion	Attribute	Proportion
comorbidity_10	0.999659	symptom_1	0.800722
symptom_b	0.990029	symptom_5	0.793534
symptom_a	0.985029	comorbidity_3	0.78916
symptom_6	0.946509	symptom_4	0.775751
comorbidity_4	0.939236	symptom_7	0.754361
comorbidity_9	0.928754	symptom_2	0.750469
comorbidity_8	0.928442	symptom_3	0.658741
comorbidity_2	0.920601	patient_age_year_fraction	0.019061
comorbidity_6	0.911795	symptom_10	0.01855
comorbidity_7	0.903244	symptom_8	0.01855
comorbidity_5	0.859042	symptom_9	0.01855
comorbidity_1	0.854753	patient_gender	0
		conformed_result	0

7.3 Correlation Matrix COVID-19 - Malaria

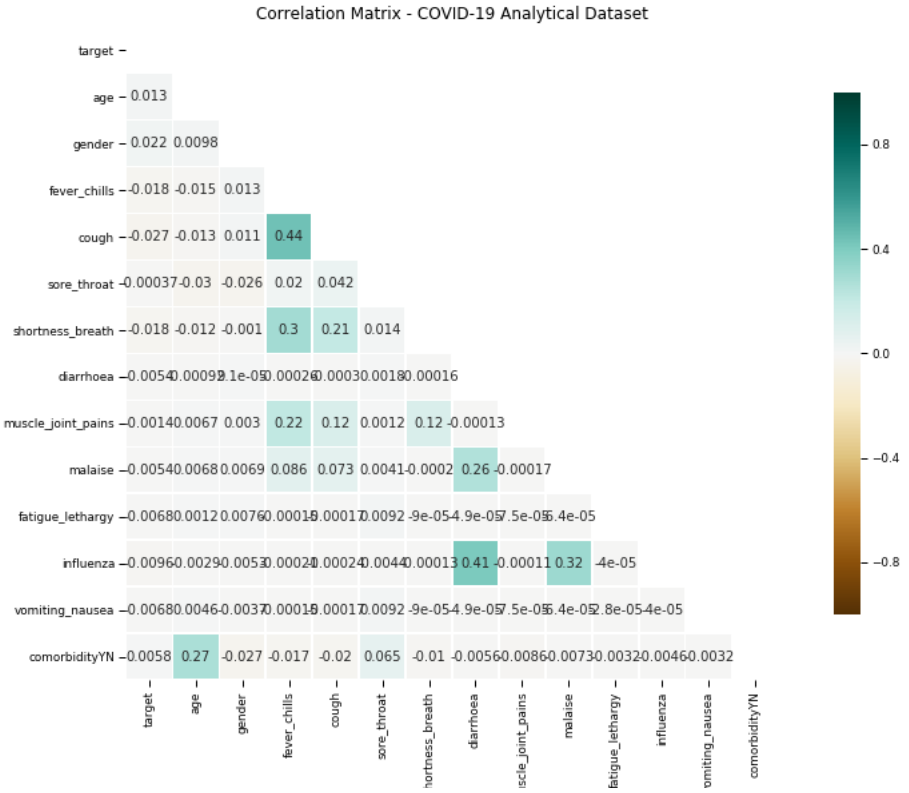


Figure 7.1: Correlation Matrix for COVID-19 analytical dataset

Note: Stronger correlation on both ends of the spectrum pop in darker color with weaker correlation in lighter shades.

8 Plagiarism Declaration

UNIVERSITY OF THE WITWATERSRAND, JOHANNESBURG

FACULTY OF HEALTH SCIENCES

PLAGIARISM DECLARATION TO BE SIGNED BY ALL HIGHER DEGREE STUDENTS

SENATE PLAGIARISM POLICY: APPENDIX ONE

I Innocent Lino ERONE (Student number: 1075688) am a student registered for the degree of MSc in Epidemiology - PHI in the academic year 2021.

I hereby declare the following:

- I am aware that plagiarism (the use of someone else's work without their permission and/or without acknowledging the original source) is wrong.
- I confirm that the work submitted for assessment for the above degree is my own unaided work except where I have explicitly indicated otherwise.
- I have followed the required conventions in referencing the thoughts and ideas of others.
- I understand that the University of the Witwatersrand may take disciplinary action against me if there is a belief that this is not my own unaided work or that I have failed to acknowledge the source of the ideas or words in my writing.
- I have included as an appendix a report from "Turnitin" (or other approved plagiarism detection) software indicating the level of plagiarism in my research document.

Signature: _____ Date: 30 April, 2021

9 TurnItIn Report

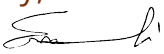
1075688:ThesisERONE30042021.docx

ORIGINALITY REPORT

5%	4%	3%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	d-nb.info Internet Source	1%
2	en.wikipedia.org Internet Source	<1%
3	dissertations.port.ac.uk Internet Source	<1%
4	Mark Goadrich, Louis Oliphant, Jude Shavlik. "Gleaner: Creating ensembles of first-order clauses to improve recall-precision curves", Machine Learning, 2006 Publication	<1%
5	powcoder.com Internet Source	<1%
6	Undergraduate Topics in Computer Science, 2013. Publication	<1%
7	Jacek Smereka, Mateusz Puslecki, Kurt Ruetzler, Krzysztof J. Filipiak, Milosz Jaguszewski, Jerzy R. Ladny, Lukasz Szarpak.	<1%



Innocent Lino ERONE
30 April, 2021

10 HREC Research Clearance Certificate

UNIVERSITY OF THE WITWATERSRAND, JOHANNESBURG

R14/49 Mr IL Erone

HUMAN RESEARCH ETHICS COMMITTEE (MEDICAL)
CLEARANCE CERTIFICATE NO. M200509

NAME: (Principal Investigator)	Mr IL Erone
DEPARTMENT:	School of Public Health Division of Epidemiology and Biostatistics Medical School University
PROJECT TITLE:	Applying machine learning to classify disease status for selected notifiable medical conditions in South Africa
DATE CONSIDERED:	2020/05/29
DECISION:	Approved unconditionally
CONDITIONS:	
SUPERVISOR:	Mr M Mapundu and Dr T Bell
APPROVED BY:	Dr CB Penny, Chairperson, HREC (Medical)
DATE OF APPROVAL:	2020/11/02

This clearance certificate is valid for 5 years from the date of approval. Extension may be applied for.

DECLARATION OF INVESTIGATORS

To be completed in duplicate and **ONE COPY** returned to the Research Office Secretary on the 3rd Floor, Phillip Tobias Building, Parktown, University of the Witwatersrand, Johannesburg.
I/we fully understand the conditions under which I am/we are authorized to carry out the above-mentioned research and I/we undertake to ensure compliance with these conditions. Should any departure be contemplated, from the research protocol as approved, I/we undertake to submit details to the Committee. **I agree to submit a yearly progress report**. When a funder requires annual re-certification, the application date will be one year after the date when the study was initially reviewed. In this case, the study was initially reviewed in **May** and will therefore reports and re-certification will be due early in the month of **May** each year. Unreported changes to the application may invalidate the clearance given by the HREC (Medical).

Principal Investigator Signature

05/11/2020
Date

11 NHLS Research Clearance

Academic Affairs and Research
Modderfontein Road, Sandringham, 2031
Tel: +27 (0)11 386 6142
Fax: +27 (0)11 386 6296
Email: babatyi.kgokong@nhls.ac.za
Web: www.nhls.ac.za

05 November 2020

Applicant: Innocent Lino Erone
Institution: University of the Witwatersrand
Department: Epidemiology
Email: 1075688@students.wits.ac.za / inno.erone@gmail.com
Cell: 075 260 2960

CC: Trevor Bell
Lead Data Analyst – NICD

Re: Approval to access National Health Laboratory Service (NHLS) Data

Your application to undertake a research project **"Applying Machine Learning To Classify Disease Status For Selected Notifiableable Medical Conditions In South Africa"** using data from the NHLS database has been reviewed. This letter serves to advise that the application has been approved and the required data will be made available to you ***without patient names*** to conduct the proposed study as outlined in the submitted application. Submissions should be made annually on the AARMS system – https://aarms.nhls.ac.za.

Please note that approval is granted on your compliance with the NHLS conditions of service and that the study can only be undertaken provided that the following conditions have been met.

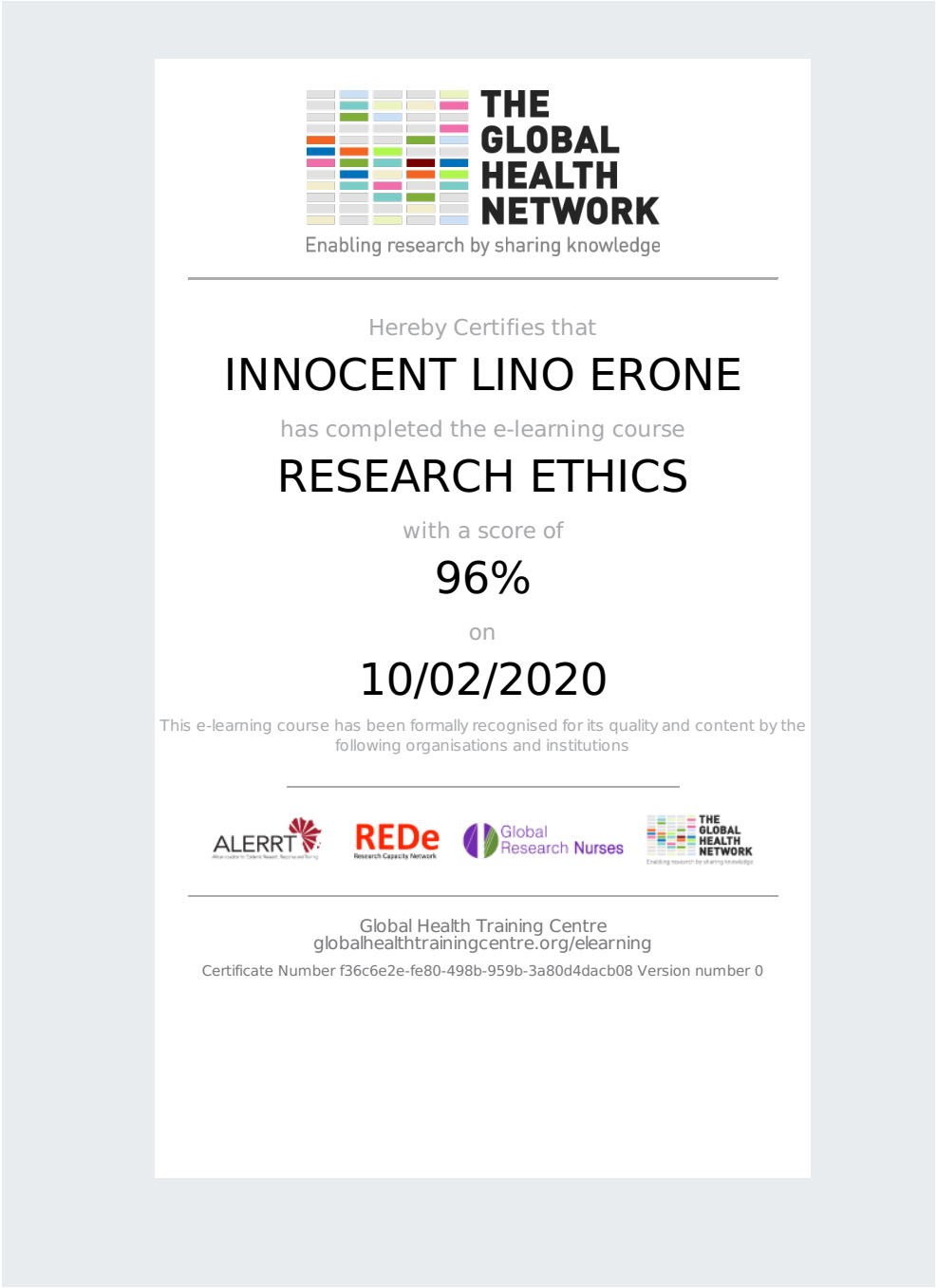
- Processes are discussed with the relevant NHLS departments (i.e. Information Management Unit and Operations Office) and are agreed upon.
- Confidentiality is maintained at participant and institutional level and there is no disclosure of personal information or confidential information as described by the NHLS policy.
- NHLS Data cannot be used to track patients as no pre-approval/consent is obtained from Patients.
- All data requested should be in accordance with the research protocol submitted and approved by the relevant Ethics Committee.
- Request for the inclusion of the NHLS as a source of data in the original protocol to be approved by Ethics as NHLS does not have a Human Research Ethics Committee.
- A final report of the research study and any published paper resulting from this study are submitted and addressed to the NHLS Academic Affairs and Research office and the NHLS has been acknowledged appropriately.
- Trevor Bell is noted as NHLS collaborator for this research.

Please note that this letter constitutes approval by the NHLS Academic Affairs and Research Office. Any data related queries may be directed to NHLS Corporate Data Warehouse, contact number: 011 386 6074 email: zarina.sabat@nhls.ac.za

P.P. Refilwe Machomoko

Dr Babatyi Malope-Kgokong
National Manager: Academic Affairs and Research

12 Research Ethics Training Certificate



THE GLOBAL HEALTH NETWORK

Enabling research by sharing knowledge

Hereby Certifies that

INNOCENT LINO ERONE

has completed the e-learning course

RESEARCH ETHICS

with a score of

96%

on

10/02/2020

This e-learning course has been formally recognised for its quality and content by the following organisations and institutions

ALERRT REDe Research Capacity Network Global Research Nurses THE GLOBAL HEALTH NETWORK

Global Health Training Centre
globalhealthtrainingcentre.org/elearning

Certificate Number f36c6e2e-fe80-498b-959b-3a80d4dacb08 Version number 0

13 Programming and Analysis Codes

Below is some of the python code used for data wrangling and analysis. The complete code follows herein

```
[ ]: # Gender
     s = SimpleImputer(missing_values=np.nan, strategy='most_frequent', verbose=1)
     s.fit(covid_train_valid[['gender']])

     covid_train_valid['gender'] = s.transform(covid_train_valid[['gender']]).reshape(-1,1)

     # Test Data
     covid_test['gender'] = s.transform(covid_test[['gender']]).reshape(-1,1)

[ ]: covid_train_valid.head(3)
```

Scale features in a contnuum

```
[ ]: scaler = MinMaxScaler(feature_range=(0, 1))
     scaler.fit(covid_train_valid[['age']])

     # transform the train
     covid_train_valid['age'] = scaler.transform(covid_train_valid[['age']])

     #transform test
     covid_test['age'] = scaler.transform(covid_test[['age']])

[ ]: covid_train_valid.head(3)

[ ]: covid_test.head(3)

[ ]: # Save a copy for training-testing
     covid_train_valid.to_csv(f"{PATH}covid_train_valid.csv", index=False)
     covid_test.to_csv(f"{PATH}covid_test.csv", index=False)
```

Isolation Forest

```
[7]: clf = IsolationForest(contamination = 0.06, random_state=SEED)
     outlierPredict(clf,
                    Xtrain = Xtraining.copy(deep=True),
                    ytrain = ytraining.copy(deep=True),
                    Xtest  = Xtesting.copy(deep=True),
                    ytest  = ytesting.copy(deep=True)
                   )

                    pred (-)  pred (+)
     True Neg (-)         54       545
     True Pos (+)        517      9024

     -----------AVERAGE SCORES----------
     mcc: 0.037
     specificity: 0.09
     F1-score: 0.092
```

Covid01_Preprocess

April 30, 2021

1 COVID-19

1.1 Preprocessing: (Dataset: 2021-02-25)

In [1]:
```
from matplotlib import pyplot
import seaborn as sns
%matplotlib inline

sns.set_context('paper')

import statistics as stats
import pandas as pd
import numpy as np
import random
import re
import ipystata # necessary to run stata commands
from sklearn.model_selection import train_test_split
from sklearn.impute import KNNImputer
from sklearn.impute import SimpleImputer

from sklearn.preprocessing import MinMaxScaler

# To ignore warnings in the notebook
import warnings
warnings.filterwarnings('ignore')

pd.set_option('display.max_rows', 100)

PATH    = 'D:/THE UNIVERSITY OF WITWATERSRAND/16.2.JupyterWorks/msc_report/data/'
FIGPATH = "D:/THE UNIVERSITY OF WITWATERSRAND/16.REPORT/Figures/"

SEED    = 888
TESTSIZE = .3
```

In [2]:
```
# Fetch data
%time covid = pd.read_csv(f'{PATH}COVID-19.csv', sep="æ")
print(covid.shape)
covid.sample()
```

```
Wall time: 633 ms
(35202, 25)
```

Out[2]:

	PATIENT_AGE_YEAR_FRACTION	PATIENT_GENDER	SYMPTOM_A	SYMPTOM_B	\
29938	31.0	F	NaN	NaN	

	COMORBIDITY_1	COMORBIDITY_2	COMORBIDITY_3	COMORBIDITY_4	COMORBIDITY_5	\
29938	NaN	NaN	NaN	NaN	NaN	

	COMORBIDITY_6	...	SYMPTOM_2	SYMPTOM_3	SYMPTOM_4	SYMPTOM_5	SYMPTOM_6	\
29938	NaN	...	0	0	0	0	0	

	SYMPTOM_7	SYMPTOM_8	SYMPTOM_9	SYMPTOM_10	CONFORMED_RESULT
29938	0	0.0	0.0	0.0	POSITIVE

[1 rows x 25 columns]

In [3]:
```
# Rename columns to lower case for ease
covid.rename(str.lower, axis='columns', inplace=True)
print(covid.columns)
```

```
Index(['patient_age_year_fraction', 'patient_gender', 'symptom_a', 'symptom_b',
       'comorbidity_1', 'comorbidity_2', 'comorbidity_3', 'comorbidity_4',
       'comorbidity_5', 'comorbidity_6', 'comorbidity_7', 'comorbidity_8',
       'comorbidity_9', 'comorbidity_10', 'symptom_1', 'symptom_2',
       'symptom_3', 'symptom_4', 'symptom_5', 'symptom_6', 'symptom_7',
       'symptom_8', 'symptom_9', 'symptom_10', 'conformed_result'],
      dtype='object')
```

In [4]:
```
covid.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 35202 entries, 0 to 35201
Data columns (total 25 columns):
patient_age_year_fraction    34531 non-null float64
patient_gender               35202 non-null object
symptom_a                    527 non-null object
symptom_b                    351 non-null object
comorbidity_1                5113 non-null object
comorbidity_2                2795 non-null object
comorbidity_3                7422 non-null object
comorbidity_4                2139 non-null object
comorbidity_5                4962 non-null object
comorbidity_6                3105 non-null object
comorbidity_7                3406 non-null object
comorbidity_8                2519 non-null object
comorbidity_9                2508 non-null object
```

```
comorbidity_10               12 non-null object
symptom_1                    34909 non-null object
symptom_2                    34753 non-null object
symptom_3                    34617 non-null object
symptom_4                    34565 non-null object
symptom_5                    34549 non-null object
symptom_6                    34549 non-null object
symptom_7                    34549 non-null object
symptom_8                    34549 non-null float64
symptom_9                    34549 non-null float64
symptom_10                   34549 non-null float64
conformed_result             35202 non-null object
dtypes: float64(4), object(21)
memory usage: 6.7+ MB
```

```
In [5]: covid.symptom_3.value_counts()
```

```
Out[5]: 0                      22604
        Cough                  11945
        Shortness of breath       34
        Sore throat               33
        Fever                      1
        Name: symptom_3, dtype: int64
```

```
In [6]: # Missing values recorded as ZERO
        for col in covid.columns:
            # reformat missing values
            if covid[col].dtype == 'object':
                covid[col].replace({'0':np.nan}, inplace=True)
```

```
In [7]: covid.symptom_1.value_counts()
```

```
Out[7]: Fever >= 38C           6655
        Cough                   240
        Shortness of breath      42
        Fever                    41
        Sore throat              37
        Name: symptom_1, dtype: int64
```

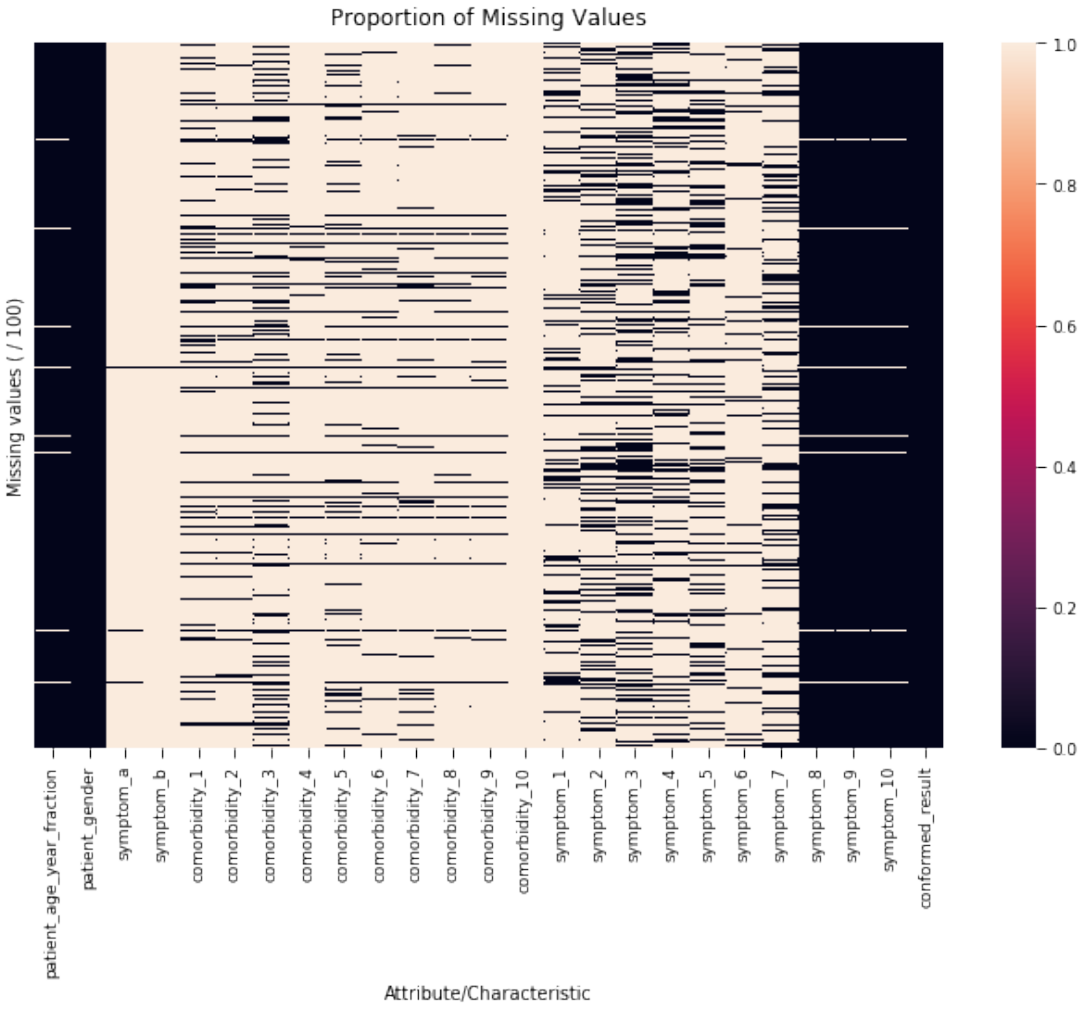
```
In [8]: covid.symptom_3.value_counts()
```

```
Out[8]: Cough                  11945
        Shortness of breath       34
        Sore throat               33
        Fever                      1
        Name: symptom_3, dtype: int64
```

```
In [9]: covid.comorbidity_10.value_counts()
```

```
Out[9]: Esophageas Cancer                                        3
        Eczema                                                   2
        Athritis                                                 2
        Hyperlipidaemia                                          1
        CHRONIC KIDNEY DISEASE, HYPOKALAEMIA, HYPO-OSMOLAL       1
        Just Gave Birth                                          1
        Intercanal hypertension                                  1
        HPT                                                      1
        Name: comorbidity_10, dtype: int64
```

```
In [10]: # Check missing information
         # covid.isnull().mean().sort_values().plot.bar(colors='#66b3ff')

         fig, ax = pyplot.subplots(figsize=(11,7))
         ax.set_xticklabels(labels = covid.columns, rotation=90)

         g = sns.heatmap(covid.isnull(), yticklabels=False, ax=ax)

         pyplot.title('Proportion of Missing Values', pad=10, fontsize=12)
         pyplot.xlabel('Attribute/Characteristic')
         pyplot.ylabel('Missing values ( / 100)')
         pyplot.savefig(f'{FIGPATH}Covid19MissingData.png')
```




```
In [11]: print(np.char.center('Percentage of Missing Data', 35, fillchar='-'))
         covid.isnull().mean().sort_values(ascending=False)
```

```
-----Percentage of Missing Data----
```

```
Out[11]: comorbidity_10          0.999659
         symptom_b               0.990029
         symptom_a               0.985029
         symptom_6               0.946509
         comorbidity_4           0.939236
         comorbidity_9           0.928754
         comorbidity_8           0.928442
         comorbidity_2           0.920601
         comorbidity_6           0.911795
```

```
comorbidity_7                0.903244
comorbidity_5                0.859042
comorbidity_1                0.854753
symptom_1                    0.800722
symptom_5                    0.793534
comorbidity_3                0.789160
symptom_4                    0.775751
symptom_7                    0.754361
symptom_2                    0.750469
symptom_3                    0.658741
patient_age_year_fraction    0.019061
symptom_10                   0.018550
symptom_8                    0.018550
symptom_9                    0.018550
patient_gender               0.000000
conformed_result             0.000000
dtype: float64
```

Rename and format for ease

```
In [12]: # rename columns
         covid.rename(columns={'conformed_result':'target',
                               'patient_age_year_fraction':'age',
                               'patient_gender':'gender'
                              }, inplace=True)

In [13]: covid.gender.value_counts()
         covid['gender'] = covid.gender.map({'F':0, 'M':1, 'S':np.nan, 'U':np.nan})

In [14]: covid.target.value_counts()
         covid['target'] = covid.target.map({'POSITIVE':1, 'NEGATIVE':0})
```

Recorded Symptoms

```
In [15]: # Scavenge through all symptom columns and draw possible symptoms from the free-text
         symptom_cols = [x for x in covid.columns if x[0:7] == 'symptom']
         symptom_cols = symptom_cols[0:9]
         print(symptom_cols)

['symptom_a', 'symptom_b', 'symptom_1', 'symptom_2', 'symptom_3', 'symptom_4', 'symptom_5', 'sy


In [16]: for col in symptom_cols[8:9]:
             covid[col] = covid[col].str.lower().str.strip()
         #     print()
         #     print(covid[col].value_counts())

In [17]: # regular expressions
         constructs = {'fever_chills' : 'sweats|fever|f[ae]brile|pyrex|temp[ea]rat|temp|hot|ho
```

Covid02_Models

April 30, 2021

1 COVID-19 DATA

This script contains two parts 1) Models optimised for Balanced datasets and 2) Models optimised for imbalanced datasets accounting for class weights in the target

In [1]:
```
import pandas as pd
import numpy as np
from collections import Counter

from sklearn.utils import resample

from sklearn.model_selection import GridSearchCV, StratifiedKFold
from sklearn.model_selection import cross_val_score

from sklearn.svm import LinearSVC, SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier

from sklearn.calibration import CalibratedClassifierCV

from sklearn import metrics

from matplotlib import pyplot
import seaborn as sns; sns.set(font_scale=1.2, style="ticks", color_codes=True)
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

Constants

In [2]:
```
PATH       = 'D:/THE UNIVERSITY OF WITWATERSRAND/16.2.JupyterWorks/msc_report/data/'
FIGPATH    = "D:/THE UNIVERSITY OF WITWATERSRAND/16.REPORT/Figures/"

TRAIN_DATA = 'covid_train_valid'
TEST_DATA  = 'covid_test'

cv         = StratifiedKFold(n_splits=5)
SEED = 888
```

```
        # class_weight = [{1:1.0,   0:16.0}]
        # class_weight = ['balanced']
        scoring      = metrics.make_scorer('f1')
```

Test Data

```
In [3]: testData = pd.read_csv(f"{PATH}{TEST_DATA}.csv")
        print(f'SHAPE: {testData.shape}')
        testData.sample(2)
        print(f'Class distributions : {Counter(testData.target)}')

        testDataX = testData[[j for j in testData.columns if j != 'target']]
        testDatay = testData.iloc[:, 0]
```

```
SHAPE: (10561, 14)
Class distributions : Counter({1: 6539, 0: 4022})
```

2 Balanced Data Models

2.1 Models optimised on Resampled Data : Observed : NotObserved [1:1]

Training Data

```
In [4]: data = pd.read_csv(f"{PATH}{TRAIN_DATA}.csv")
        print(f'SHAPE: {data.shape}')

        # Check the distribution in the outcome
        print(f'Class distributions: {Counter(data.target)}')

        # X = malaria.drop('target', axis='columns')
        X = data[[j for j in data.columns if j != 'target']]
        y = data.iloc[:, 0]
```

```
SHAPE: (24641, 14)
Class distributions: Counter({1: 15256, 0: 9385})
```

Re-sample to Balance the Training Data

```
In [5]: # REF: https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your
        # Down-sample the majority class using the length of minority class as a measure

        pos_samples = data.loc[data.target==1, :]
        neg_samples = data.loc[data.target==0, :]

        test_result_1_downsampled = resample(pos_samples,
                                             n_samples = len(neg_samples),
```

```
                                                random_state = SEED,
                                                replace=False
                                               )
        print(f'\nPositive/Majority class downsampled: {len(test_result_1_downsampled)}')

        # merge the downsample to back to the initially minority sample. We expect a 50:50 bal
        dataSampled = pd.concat([test_result_1_downsampled, neg_samples], axis=0)
        print()
        print(f'Malaria Obs:Nobs {dataSampled.target.value_counts(normalize=True).ravel()}')
        print(dataSampled.target.value_counts())


        # USE THE RESAMPLED DATA
        data = dataSampled.copy(deep=True)
```

```
Positive/Majority class downsampled: 9385

Malaria Obs:Nobs [0.5 0.5]
1    9385
0    9385
Name: target, dtype: int64
```

2.1.1 Default CV Estimates - Training Data

In [6]:
```
# LinearSVC
# Because of the time complexities that come with the defaut 'rbf' kernel, we set this
# using a modified LinearSVC()

score = cross_val_score(LinearSVC(), X=X, y=y, cv=cv, scoring='f1')
print(score)
print(f'SVC - F1-Score: {np.round(score.mean(), decimals=3)}')
```

```
[0.7646542  0.76508215 0.76427048 0.76565834 0.76494923]
SVC - F1-Score: 0.765
```

In [7]:
```
# RF

score = cross_val_score(RandomForestClassifier(), X=X, y=y, cv=cv, scoring='f1')
print(score)
print(f'Random Forest -  F1-Score: {np.round(score.mean(), decimals=3)}')
```

```
[0.745402   0.75499804 0.74538745 0.74096869 0.74349835]
Random Forest -  F1-Score: 0.746
```

In [8]:
```
# KNN
```

```
        score = cross_val_score(KNeighborsClassifier(), X=X, y=y, cv=cv, scoring='f1')
        print(score)
        print(f'KNN - F1-Score: {np.round(score.mean(), decimals=3)}')
```

```
[0.67931977 0.67935445 0.69660942 0.6797326  0.66575716]
KNN - F1-Score: 0.68
```

2.1.2 GridSearch CV estimates - Training data

In [9]:
```
        # LinearSVC

        penalty = ['l1', 'l2']
        C       = [1, 10, 25]
        loss    = ['hinge', 'hinge_squared']
        param_grid = dict(C=C, penalty=penalty, loss=loss)

        svcgs = GridSearchCV( LinearSVC(), param_grid=param_grid,  cv=cv, scoring='f1', n_jobs

        svcgs.fit(X, y)
        print(svcgs.best_score_)
        print(svcgs.best_estimator_)

        # Best parameters
        C      = svcgs.best_params_['C']
        penalty  = svcgs.best_params_['penalty']
        loss = svcgs.best_params_['loss']
        # ======


        # C = [1, 10, 25]
        # gamma = [0.01, 0.1, 1]
        # class_weight = ['balanced']
        # kernel = ['linear']
        # param_grid = dict(C=C, gamma=gamma, kernel=kernel, class_weight=class_weight)

        # svcgs = GridSearchCV( SVC(), param_grid=param_grid,  cv=cv, scoring='f1', n_jobs = -

        # svcgs.fit(X, y)
        # print(svcgs.best_score_)
        # print(svcgs.best_estimator_)

        # # Best parameters
        # C            = svcgs.best_params_['C']
        # gamma        = svcgs.best_params_['gamma']
        # kernel       = svcgs.best_params_['kernel']
        # weight_class = svcgs.best_params_['class_weight']
```

```
0.7649198194152894
```

```
LinearSVC(C=1, loss='hinge')
```

In [10]:
```
# RANDOM FOREST

n_estimators      = [10, 12, 14, 18]
min_samples_split = [2,3,4]
criterion         = ["gini", "entropy"]
bootstrap         = [True, False]
param_grid = dict(n_estimators=n_estimators, min_samples_split=min_samples_split, cri

rfgs = GridSearchCV( RandomForestClassifier(), param_grid=param_grid,  cv=cv, scoring

rfgs.fit(X, y)
print(rfgs.best_score_)
print(rfgs.best_estimator_)

# Best parameters
bootstrap         = rfgs.best_params_['bootstrap']
criterion         = rfgs.best_params_['criterion']
min_samples_split = rfgs.best_params_['min_samples_split']
n_estimators      = rfgs.best_params_['n_estimators']
```

```
0.7462801649538445
RandomForestClassifier(n_estimators=12)
```

In [11]:
```
# K-NEIGHBORS

n_neighbors = list(np.arange(5,16,1))
algorithm = ['ball_tree', 'kd_tree', 'brute']
weights = ['uniform']
param_grid = dict(n_neighbors=n_neighbors, weights=weights, algorithm=algorithm)

knngs = GridSearchCV( KNeighborsClassifier(), param_grid=param_grid,  cv=cv, scoring=

knngs.fit(X, y)
print(knngs.best_score_)
print(knngs.best_estimator_)

# Best parameters
algorithm   = knngs.best_params_['algorithm']
n_neighbors = knngs.best_params_['n_neighbors']
weights     = knngs.best_params_['weights']
```

```
0.7119355904575971
KNeighborsClassifier(algorithm='brute', n_neighbors=15)
```

2.2 Scores on out-of-sample data

2.2.1 Default Models

In [12]:
```
# Default - LinearSVC

# svc = LinearSVC()
# svc.fit(X, y)

SV = CalibratedClassifierCV( LinearSVC() )
SV.fit(X, y)

yhat = SV.predict(testDataX)

precision = metrics.precision_score(testDatay, yhat)
recall    = metrics.recall_score(testDatay, yhat)
f1score   = metrics.f1_score(testDatay, yhat)

# predict probabilities AND keep probabilities for the positive outcome only
AAA = SV.predict_proba(testDataX)
AAA = AAA[:, 1]

# get auc and error
sv_auc = metrics.roc_auc_score(testDatay, AAA)
error  = metrics.zero_one_loss(testDatay, yhat)

print('SVC: Precision=%.3f  Recall=%.3f  f1=%.3f  auc=%.3f  error=%.3f' % (precision,
```

```
SVC: Precision=0.620  Recall=1.000  f1=0.765  auc=0.510  error=0.380
```

In [13]:
```
# Default - RF

RF = RandomForestClassifier().fit(X, y)
yhat = RF.predict(testDataX)

precision = metrics.precision_score(testDatay, yhat)
recall    = metrics.recall_score(testDatay, yhat)
f1score   = metrics.f1_score(testDatay, yhat)

# predict probabilities AND keep probabilities for the positive outcome only
BBB = RF.predict_proba(testDataX)
BBB = BBB[:, 1]

# get auc and error
rfc_auc = metrics.roc_auc_score(testDatay, BBB)
error  = metrics.zero_one_loss(testDatay, yhat)

print('RFC: Precision=%.3f  Recall=%.3f  f1=%.3f  auc=%.3f  error=%.3f' % (precision,
```

```
RFC: Precision=0.623  Recall=0.945  f1=0.751  auc=0.560  error=0.389
```

```
In [14]: # Default - KNN

         KNN = KNeighborsClassifier().fit(X, y)
         yhat = KNN.predict(testDataX)

         precision = metrics.precision_score(testDatay, yhat)
         recall    = metrics.recall_score(testDatay, yhat)
         f1score   = metrics.f1_score(testDatay, yhat)

         # predict probabilities AND keep probabilities for the positive outcome only
         CCC = KNN.predict_proba(testDataX)
         CCC = CCC[:, 1]

         # get auc and error
         knn_auc = metrics.roc_auc_score(testDatay, CCC)
         error   = metrics.zero_one_loss(testDatay, yhat)

         print('KNN: Precision=%.3f  Recall=%.3f  f1=%.3f  auc=%.3f  error=%.3f' % (precision,
```

```
KNN: Precision=0.633  Recall=0.743  f1=0.684  auc=0.532  error=0.426
```

2.2.2 Optimised Models

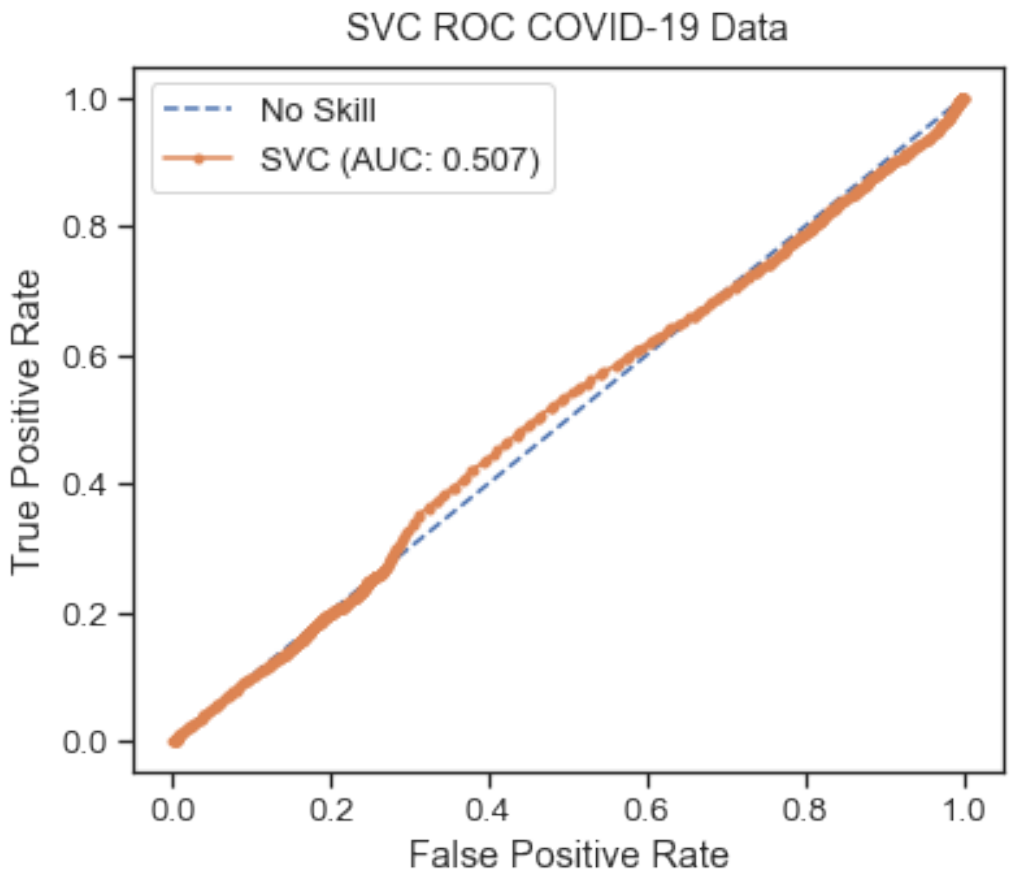
Support Vector Classifier

```
In [15]: # Get the parameters of the best parameters and fit the model THEN predict
         # yhat = svcgs.predict(testDataX)

         # SV = SVC(C=C, gamma=gamma, kernel=kernel, probability=True)
         # SV.fit(X, y)

         SV = CalibratedClassifierCV( LinearSVC(C=C, penalty=penalty, loss=loss) )
         SV.fit(X, y)

         yhat = SV.predict(testDataX)


         # predict probabilities AND keep probabilities for the positive outcome only
         DDD = SV.predict_proba(testDataX)
         DDD = DDD[:, 1]

         # generate a no skill prediction (majority class)
         noskill_probs = [0 for _ in range(len(testDatay))]

         # # calculate scores
```

```
        # noskill_auc = metrics.roc_auc_score(testDatay, noskill_probs)

        print(f'predicted   : {Counter(yhat)}')

        accuracy  = metrics.accuracy_score(testDatay, yhat)
        mcc       = metrics.matthews_corrcoef(testDatay, yhat)
        error     = metrics.zero_one_loss(testDatay, yhat)

        auc       = metrics.roc_auc_score(testDatay, DDD)

        recall    = metrics.recall_score(testDatay, yhat)
        precision = metrics.precision_score(testDatay, yhat)
        f1score   = metrics.f1_score(testDatay, yhat)

        print()
        print(pd.DataFrame(metrics.confusion_matrix(testDatay, yhat), index=['True Neg (-) ',
        print()
        print('SVC BALANCED: \n accuracy=%.3f \n mcc=%.3f \n error=%.3f \n auc=%.3f \n recall=



        # define a label to use for the graph
        LABEL_SVC = 'SVC (AUC: '  +  str(np.round(auc, decimals=3))  + ')'

        # calculate roc curves
        noskill_fpr, noskill_tpr, _ = metrics.roc_curve(testDatay, noskill_probs)
        sv_model_fpr, sv_model_tpr, _ = metrics.roc_curve(testDatay, DDD)

        # plot the roc curve for the model
        thisfigure = pyplot.figure(figsize=[6, 5])
        pyplot.plot(noskill_fpr, noskill_tpr, linestyle='--', label='No Skill')
        pyplot.plot(sv_model_fpr, sv_model_tpr, marker='.', label=LABEL_SVC)

        # axis labels
        pyplot.title('SVC ROC COVID-19 Data', pad=10)
        pyplot.xlabel('False Positive Rate')
        pyplot.ylabel('True Positive Rate')
        pyplot.legend()   # show the legend
        pyplot.show()     # show the plot
        thisfigure.savefig(f'{FIGPATH}SVC_Balanced_Covid.png')
```

```
predicted   : Counter({1: 10549, 0: 12})

               pred (-)  pred (+)
True Neg (-)          9      4013
True Pos (+)          3      6536

SVC BALANCED:
```

```
accuracy=0.620
mcc=0.026
error=0.380
auc=0.507
recall=1.000
precision=0.620
f1score=0.765
```

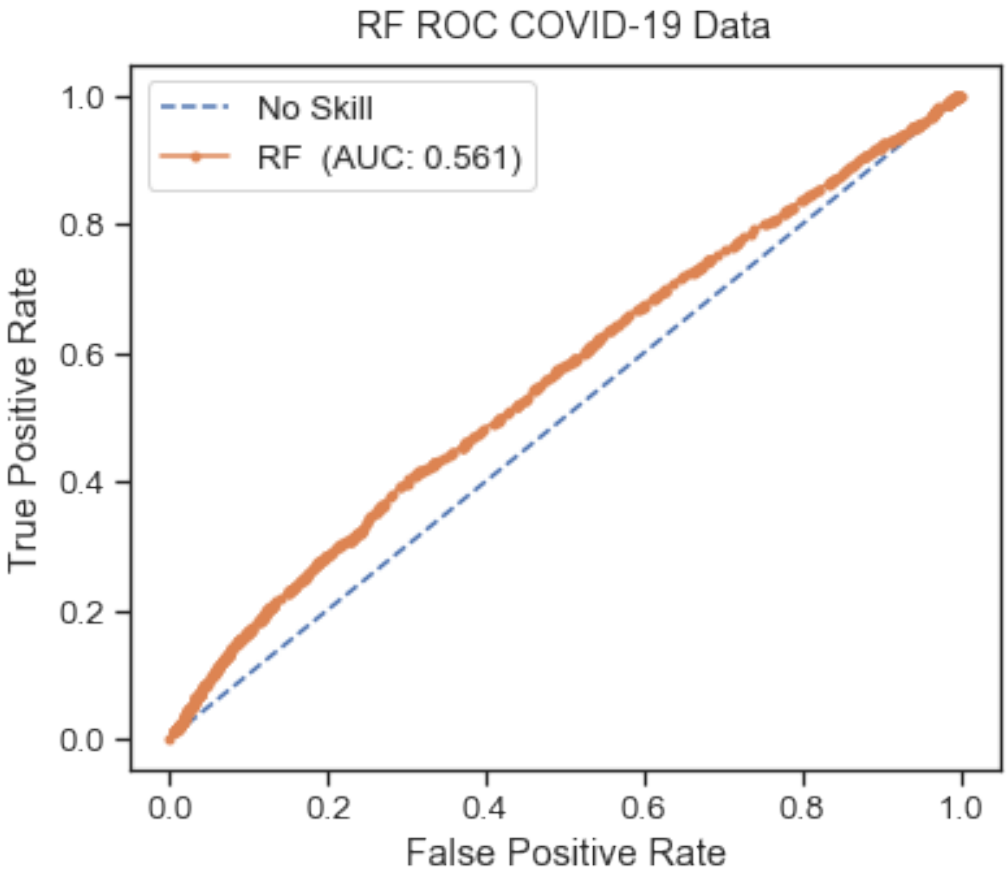


Random Forest Classifier

```
In [16]: # Get the parameters of the best parameters and fit the model THEN predict
         RF = RandomForestClassifier(n_estimators=n_estimators, min_samples_split=min_samples_s
         yhat = RF.predict(testDataX)

         accuracy  = metrics.accuracy_score(testDatay, yhat)
         precision = metrics.precision_score(testDatay, yhat)
         recall    = metrics.recall_score(testDatay, yhat)
         f1score   = metrics.f1_score(testDatay, yhat)
```

```
# predict probabilities AND keep probabilities for the positive outcome only
EEE = RF.predict_proba(testDataX)
EEE = EEE[:, 1]


# generate a no skill prediction (majority class)
noskill_probs = [0 for _ in range(len(testDatay))]


# calculate scores
# noskill_auc = metrics.roc_auc_score(testDatay, noskill_probs)

print(f'predicted   : {Counter(yhat)}')

accuracy  = metrics.accuracy_score(testDatay, yhat)
mcc       = metrics.matthews_corrcoef(testDatay, yhat)
error     = metrics.zero_one_loss(testDatay, yhat)

auc       = metrics.roc_auc_score(testDatay, EEE)

recall    = metrics.recall_score(testDatay, yhat)
precision = metrics.precision_score(testDatay, yhat)
f1score   = metrics.f1_score(testDatay, yhat)

print()
print(pd.DataFrame(metrics.confusion_matrix(testDatay, yhat), index=['True Neg (-) ',
print()
print('RF BALANCED: \n accuracy=%.3f \n mcc=%.3f \n error=%.3f \n auc=%.3f \n recall=



# define a label to use for the graph
LABEL_RF = 'RF  (AUC: '  +  str(np.round(auc, decimals=3))  + ')'

# calculate roc curves
noskill_fpr, noskill_tpr, _ = metrics.roc_curve(testDatay, noskill_probs)
rf_model_fpr, rf_model_tpr, _ = metrics.roc_curve(testDatay, EEE)

# plot the roc curve for the model
thisfigure = pyplot.figure(figsize=[6, 5])
pyplot.plot(noskill_fpr, noskill_tpr, linestyle='--', label='No Skill')
pyplot.plot(rf_model_fpr, rf_model_tpr, marker='.', label=LABEL_RF)

# axis labels
pyplot.title('RF ROC COVID-19 Data', pad=10)
pyplot.xlabel('False Positive Rate')
pyplot.ylabel('True Positive Rate')
```

```
        pyplot.legend()    # show the legend
        pyplot.show()      # show the plot
        thisfigure.savefig(f'{FIGPATH}RF_Balanced_Covid.png')

predicted   : Counter({1: 9920, 0: 641})

               pred (-)  pred (+)
True Neg (-)        275      3747
True Pos (+)        366      6173

RF BALANCED:
 accuracy=0.611
 mcc=0.025
 error=0.389
 auc=0.561
 recall=0.944
 precision=0.622
 f1score=0.750
```

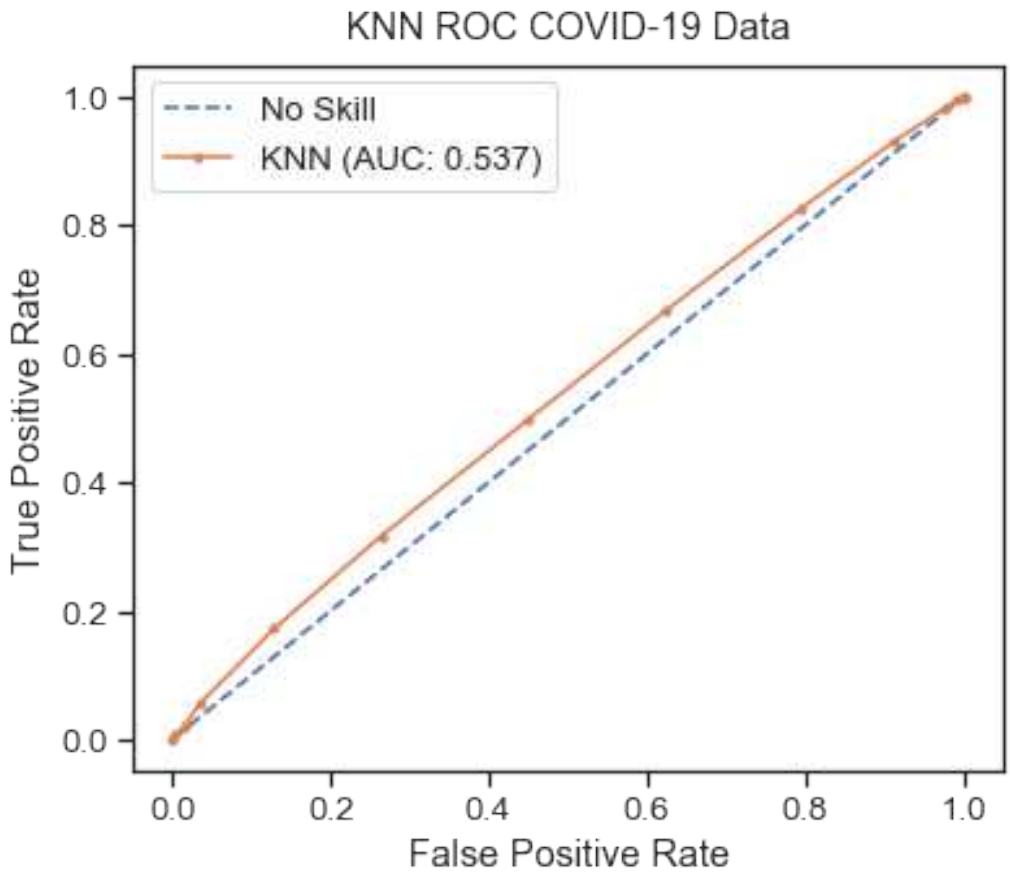


KNN

In [17]:
```
KNN = KNeighborsClassifier(algorithm=algorithm, n_neighbors=n_neighbors, weights=weigh
yhat = KNN.predict(testDataX)


### PRECISION-RECALL-CURVES

# predict probabilities AND keep probabilities for the positive outcome only
FFF = KNN.predict_proba(testDataX)
FFF = FFF[:, 1]

# generate a no skill prediction (majority class)
noskill_probs = [0 for _ in range(len(testDatay))]

# calculate scores
# noskill_auc = metrics.roc_auc_score(testDatay, noskill_probs)

print(f'predicted   : {Counter(yhat)}')


accuracy  = metrics.accuracy_score(testDatay, yhat)
mcc       = metrics.matthews_corrcoef(testDatay, yhat)
error     = metrics.zero_one_loss(testDatay, yhat)

auc       = metrics.roc_auc_score(testDatay, FFF)

recall    = metrics.recall_score(testDatay, yhat)
precision = metrics.precision_score(testDatay, yhat)
f1score   = metrics.f1_score(testDatay, yhat)

print()
print(pd.DataFrame(metrics.confusion_matrix(testDatay, yhat), index=['True Neg (-) ',
print()
print('KNN: \n accuracy=%.3f \n mcc=%.3f \n error=%.3f \n auc=%.3f \n recall=%.3f \n p



# define a label to use for the graph
LABEL_KNN = 'KNN (AUC: '  +  str(np.round(auc, decimals=3))  + ')'

# calculate roc curves
noskill_fpr, noskill_tpr, _ = metrics.roc_curve(testDatay, noskill_probs)
knn_model_fpr, knn_model_tpr, _ = metrics.roc_curve(testDatay, FFF)

# plot the roc curve for the model
thisfigure = pyplot.figure(figsize=[6, 5])
pyplot.plot(noskill_fpr, noskill_tpr, linestyle='--', label='No Skill')
```

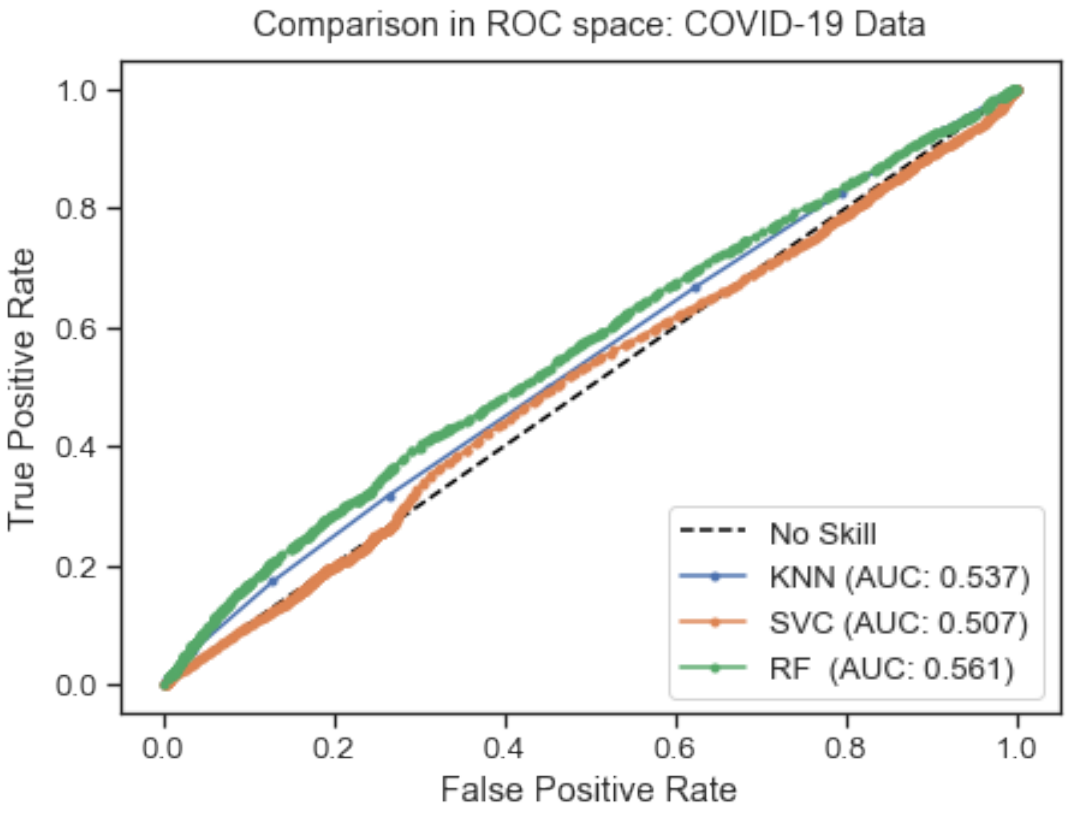
```
        pyplot.plot(knn_model_fpr, knn_model_tpr, marker='.', label=LABEL_KNN)

        # axis labels
        pyplot.title('KNN ROC COVID-19 Data', pad=10)
        pyplot.xlabel('False Positive Rate')
        pyplot.ylabel('True Positive Rate')
        pyplot.legend()   # show the legend
        pyplot.show()     # show the plot
        thisfigure.savefig(f'{FIGPATH}KNN_Balanced_covid.png')
```

```
predicted   : Counter({1: 8595, 0: 1966})

               pred (-)  pred (+)
True Neg (-)        827      3195
True Pos (+)       1139      5400

KNN:
 accuracy=0.590
 mcc=0.039
 error=0.410
 auc=0.537
 recall=0.826
 precision=0.628
 f1score=0.714
```



Combined balanced models - Balanced Datasets

```
In [18]: # plot the roc curve for the model
         thisfigure = pyplot.figure(figsize=[7, 5])
         pyplot.plot(noskill_fpr, noskill_tpr, linestyle='--', label='No Skill', color='black'
         pyplot.plot(knn_model_fpr, knn_model_tpr, marker='.', label=LABEL_KNN, color='C0')
         pyplot.plot(sv_model_fpr, sv_model_tpr, marker='.', label=LABEL_SVC, color='C1')
         pyplot.plot(rf_model_fpr, rf_model_tpr, marker='.', label=LABEL_RF, color='C2')

         # axis labels
         pyplot.title('Comparison in ROC space: COVID-19 Data', pad=10)
         pyplot.xlabel('False Positive Rate')
         pyplot.ylabel('True Positive Rate')
         pyplot.legend(loc='lower right')
         pyplot.show()
         thisfigure.savefig(f'{FIGPATH}Combined_Models_Balanced_Data_covid.png')
```



3 Weighted Models

3.1 Accounting for distribution weights in the target: Observed : NotObserved [2:1]

Training Data

```
In [19]: data = pd.read_csv(f"{PATH}{TRAIN_DATA}.csv")
         print(f'SHAPE: {data.shape}')

         # Check the distribution in the outcome
         print(f'Class distributions: {Counter(data.target)}')

         # X = malaria.drop('target', axis='columns')
         X = data[[j for j in data.columns if j != 'target']]
         y = data.iloc[:, 0]
```

```
SHAPE: (24641, 14)
Class distributions: Counter({1: 15256, 0: 9385})
```

```
In [20]: # Default - LinearSVC

         score = cross_val_score(LinearSVC(), X=X, y=y, cv=cv, scoring='f1')
         print(score)
         print(f'SVC F1-Score: {np.round(score.mean(), decimals=3)}')
```

```
[0.7646542  0.76508215 0.76427048 0.76565834 0.76494923]
SVC F1-Score: 0.765
```

```
In [21]: # Default - Random Forests

         score = cross_val_score(RandomForestClassifier(), X=X, y=y, cv=cv, scoring='f1')
         print(score)
         print(f'CrossValidated F1-Score: {np.round(score.mean(), decimals=3)}')
```

```
[0.74306839 0.75444561 0.74558862 0.74256766 0.7433254 ]
CrossValidated F1-Score: 0.746
```

```
In [22]: # Default - KNN

         score = cross_val_score(KNeighborsClassifier(), X=X, y=y, cv=cv, scoring='f1')
         print(score)
         print(f'KNN - F1-Score: {np.round(score.mean(), decimals=3)}')
```

```
[0.67931977 0.67935445 0.69660942 0.6797326  0.66575716]
KNN - F1-Score: 0.68
```

3.1.1 GridSearch CV estimates - Training Data

```
In [23]: # LinearSVC

         penalty = ['l1', 'l2']
         C       = [1, 10, 25]
         loss    = ['hinge', 'hinge_squared']
         param_grid = dict(C=C, penalty=penalty, loss=loss)

         svcgs = GridSearchCV( LinearSVC(), param_grid=param_grid,  cv=cv, scoring='f1', n_jobs

         svcgs.fit(X, y)
         print(svcgs.best_score_)
         print(svcgs.best_estimator_)

         # Best parameters
         C       = svcgs.best_params_['C']
         penalty = svcgs.best_params_['penalty']
         loss    = svcgs.best_params_['loss']
```

```
        # ===

        # C = [1, 10, 25]
        # gamma = [0.01, 0.1, 1]
        # class_weight = ['balanced']
        # kernel = ['linear']
        # param_grid = dict(C=C, gamma=gamma, kernel=kernel, class_weight=class_weight)

        # svcgs = GridSearchCV( SVC(), param_grid=param_grid,  cv=cv, scoring='f1', n_jobs =
        # svcgs.fit(X, y)


        # print(svcgs.best_score_)
        # print(svcgs.best_estimator_)

        # # Best parameters
        # C            = svcgs.best_params_['C']
        # gamma        = svcgs.best_params_['gamma']
        # kernel       = svcgs.best_params_['kernel']
        # weight_class = svcgs.best_params_['class_weight']
```

```
0.7649198194152894
LinearSVC(C=1, loss='hinge')
```

In [24]:
```
# RANDOM FOREST

n_estimators      = [10, 12, 14, 18]
min_samples_split = [2,3,4]
criterion         = ["gini", "entropy"]
bootstrap         = [True, False]
class_weight      = ['balanced']

param_grid = dict(n_estimators=n_estimators, min_samples_split=min_samples_split,
                  criterion=criterion, bootstrap=bootstrap, class_weight=class_weight

rfgs = GridSearchCV( RandomForestClassifier(), param_grid=param_grid,  cv=5, scoring=

rfgs.fit(X, y)
print(rfgs.best_score_)
print(rfgs.best_estimator_)

# Best parameters
bootstrap         = rfgs.best_params_['bootstrap']
criterion         = rfgs.best_params_['criterion']
min_samples_split = rfgs.best_params_['min_samples_split']
n_estimators      = rfgs.best_params_['n_estimators']
class_weight      = rfgs.best_params_['class_weight']
```

```
0.5607858587879447
RandomForestClassifier(class_weight='balanced', min_samples_split=3,
                       n_estimators=10)
```

```
In [25]: # KNN

         n_neighbors = list(np.arange(5,16,1))
         algorithm = ['ball_tree', 'kd_tree', 'brute']
         weights = ['distance']
         param_grid = dict(n_neighbors=n_neighbors, weights=weights, algorithm=algorithm)

         knngs = GridSearchCV( KNeighborsClassifier(), param_grid=param_grid,  cv=4, scoring='

         knngs.fit(X, y)
         print(knngs.best_score_)
         print(knngs.best_estimator_)

         # Best parameters
         algorithm   = knngs.best_params_['algorithm']
         n_neighbors = knngs.best_params_['n_neighbors']
         weights     = knngs.best_params_['weights']
```

```
0.7010271898109306
KNeighborsClassifier(algorithm='ball_tree', n_neighbors=15, weights='distance')
```

3.2 Scores on out-of-sample data

3.2.1 Default Models

```
In [26]: # LinearSVC

         # svc = SVC(kernel='linear', probability=True).fit(X, y)

         SV = CalibratedClassifierCV( LinearSVC() )
         SV.fit(X, y)
         yhat = SV.predict(testDataX)

         precision = metrics.precision_score(testDatay, yhat)
         recall    = metrics.recall_score(testDatay, yhat)
         f1score   = metrics.f1_score(testDatay, yhat)

         # predict probabilities AND keep probabilities for the positive outcome only
         GGG = SV.predict_proba(testDataX)
         GGG = GGG[:, 1]

         # get precision and recall probabilities
         def_precision , def_recall , _ = metrics.precision_recall_curve(testDatay, GGG)
```

```
auc       = metrics.auc(def_recall, def_precision)
error     = metrics.zero_one_loss(testDatay, yhat)

print('SVC: Precision=%.3f  Recall=%.3f  f1=%.3f  auc=%.3f  error=%.3f' % (precision,
```

```
SVC: Precision=0.620  Recall=1.000  f1=0.765  auc=0.636  error=0.380
```

In [27]:
```
# Random Forest

RF = RandomForestClassifier().fit(X, y)
yhat = RF.predict(testDataX)

yhat = RF.predict(testDataX)

precision = metrics.precision_score(testDatay, yhat)
recall    = metrics.recall_score(testDatay, yhat)
f1score   = metrics.f1_score(testDatay, yhat)

# predict probabilities AND keep probabilities for the positive outcome only
HHH = RF.predict_proba(testDataX)
HHH = HHH[:, 1]

# get precision and recall probabilities
def_precision , def_recall , _ = metrics.precision_recall_curve(testDatay, HHH)
auc       = metrics.auc(def_recall, def_precision)
error     = metrics.zero_one_loss(testDatay, yhat)

print('RFC: Precision=%.3f  Recall=%.3f  f1=%.3f  auc=%.3f  error=%.3f' % (precision,
```

```
RFC: Precision=0.623  Recall=0.937  f1=0.749  auc=0.674  error=0.390
```

In [28]:
```
# K-Neighbors

KNN = KNeighborsClassifier().fit(X, y)
yhat = KNN.predict(testDataX)

precision = metrics.precision_score(testDatay, yhat)
recall    = metrics.recall_score(testDatay, yhat)
f1score   = metrics.f1_score(testDatay, yhat)

# predict probabilities AND keep probabilities for the positive outcome only
III = KNN.predict_proba(testDataX)
III = III[:, 1]

# get precision and recall probabilities
def_precision , def_recall , _ = metrics.precision_recall_curve(testDatay, III)
auc       = metrics.auc(def_recall, def_precision)
```

```
error     = metrics.zero_one_loss(testDatay, yhat)

print('KNN: Precision=%.3f  Recall=%.3f  f1=%.3f  auc=%.3f  error=%.3f' % (precision,
```

```
KNN: Precision=0.633  Recall=0.743  f1=0.684  auc=0.663  error=0.426
```

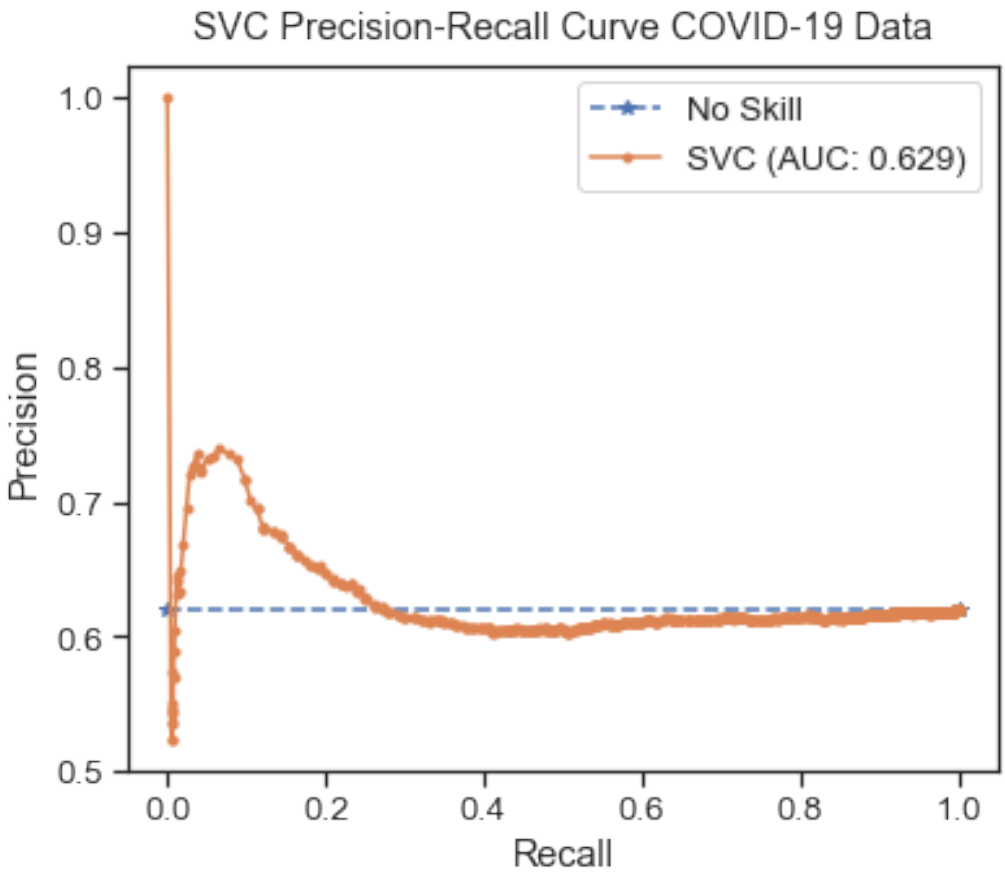
3.2.2 Optimised Models

Support Vector Classifier

In [29]:
```
# Get the parameters of the best parameters and fit the model THEN predict
# yhat = svcgs.predict(testDataX)

SV = CalibratedClassifierCV( LinearSVC(C=C, penalty=penalty, loss=loss) )
# SV = SVC(C=C, gamma=gamma, kernel=kernel, probability=True)
SV.fit(X, y)

yhat = SV.predict(testDataX)


## PRECISION - RECALL CURVE
# predict probabilities AND keep probabilities for the positive outcome only
JJJ = SV.predict_proba(testDataX)
JJJ = JJJ[:, 1]

# get precision and recall probabilities
sv_precision , sv_recall , _ = metrics.precision_recall_curve(testDatay, JJJ)

# summarize scores
accuracy  = metrics.accuracy_score(testDatay, yhat)
mcc       = metrics.matthews_corrcoef(testDatay, yhat)
error     = metrics.zero_one_loss(testDatay, yhat)

auc       = metrics.auc(sv_recall, sv_precision)

recall    = metrics.recall_score(testDatay, yhat)
precision = metrics.precision_score(testDatay, yhat)
f1score   = metrics.f1_score(testDatay, yhat)

print()
print(pd.DataFrame(metrics.confusion_matrix(testDatay, yhat), index=['True Neg (-) ',
print()
print('SVC: \n accuracy=%.3f \n mcc=%.3f \n error=%.3f \n auc=%.3f \n recall=%.3f \n p


# define a label to use for the graph
LABEL_SVC = 'SVC (AUC: '  +  str(np.round(auc, decimals=3))  + ')'
```

```
        # plot the precision-recall curves
        no_skill = len(testDatay[testDatay==1]) / len(testDatay)

        thisfigure = pyplot.figure(figsize=[6, 5])
        pyplot.plot([0, 1], [no_skill, no_skill], linestyle='--', label='No Skill', marker='*
        pyplot.plot(sv_recall, sv_precision, marker='.', label=LABEL_SVC)          # model

        # axis labels
        pyplot.title('SVC Precision-Recall Curve COVID-19 Data', pad=10)
        pyplot.xlabel('Recall')
        pyplot.ylabel('Precision')
        pyplot.legend()
        pyplot.show()
        thisfigure.savefig(f'{FIGPATH}SVC_COVID_PRCurve.png')
```

```
               pred (-)  pred (+)
True Neg (-)          9      4013
True Pos (+)          3      6536

SVC:
 accuracy=0.620
 mcc=0.026
 error=0.380
 auc=0.629
 recall=1.000
 precision=0.620
 f1score=0.765
```

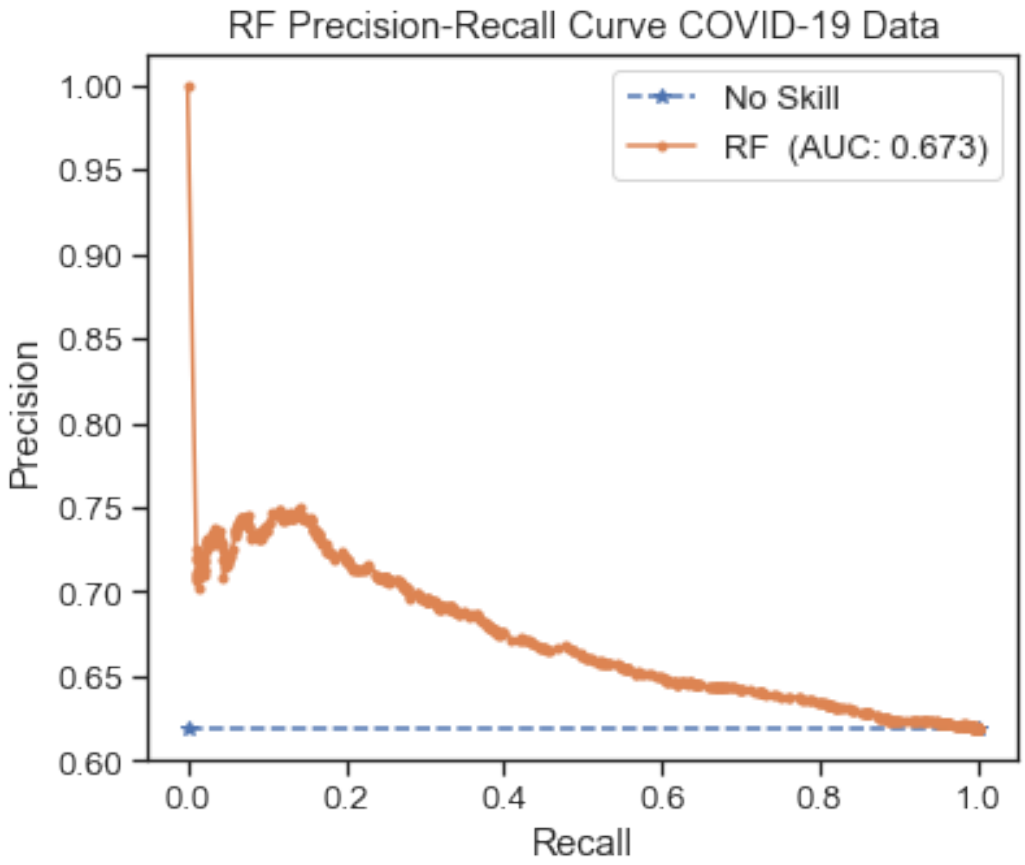


Random Forest Classifier

```
In [30]: # Get the parameters of the best parameters and fit the model THEN predict
         RF = RandomForestClassifier(n_estimators=n_estimators, min_samples_split=min_samples_s
                                     criterion=criterion, class_weight=class_weight, bootstr
                                    ).fit(X, y)
         yhat = RF.predict(testDataX)


         ## PRECISION - RECALL CURVE

         # predict probabilities AND keep probabilities for the positive outcome only
         KKK = RF.predict_proba(testDataX)
         KKK = KKK[:, 1]

         # get precision and recall probabilities
         rf_precision , rf_recall , _ = metrics.precision_recall_curve(testDatay, KKK)


         # summarize scores
```

```
accuracy  = metrics.accuracy_score(testDatay, yhat)
mcc       = metrics.matthews_corrcoef(testDatay, yhat)
error     = metrics.zero_one_loss(testDatay, yhat)

auc       = metrics.auc(rf_recall, rf_precision)

recall    = metrics.recall_score(testDatay, yhat)
precision = metrics.precision_score(testDatay, yhat)
f1score   = metrics.f1_score(testDatay, yhat)

print()
print(pd.DataFrame(metrics.confusion_matrix(testDatay, yhat), index=['True Neg (-) ',
print()
print('RF: \n accuracy=%.3f \n mcc=%.3f \n error=%.3f \n auc=%.3f \n recall=%.3f \n pr



# define a label to use for the graph
LABEL_RF = 'RF  (AUC: '  +  str(np.round(auc, decimals=3))  + ')'

# plot the precision-recall curves
no_skill = len(testDatay[testDatay==1]) / len(testDatay)

thisfigure = pyplot.figure(figsize=[6, 5])
pyplot.plot([0, 1], [no_skill, no_skill], linestyle='--', label='No Skill', marker='*
pyplot.plot(rf_recall, rf_precision, marker='.', label=LABEL_RF)

# axis labels
pyplot.title('RF Precision-Recall Curve COVID-19 Data')
pyplot.xlabel('Recall')
pyplot.ylabel('Precision')
pyplot.legend()
pyplot.show()
thisfigure.savefig(f'{FIGPATH}RF_COVID_PRCurve.png')
```

```
               pred (-)  pred (+)
True Neg (-)       2371      1651
True Pos (+)       3291      3248

RF:
 accuracy=0.532
 mcc=0.084
 error=0.468
 auc=0.673
 recall=0.497
 precision=0.663
 f1score=0.568
```

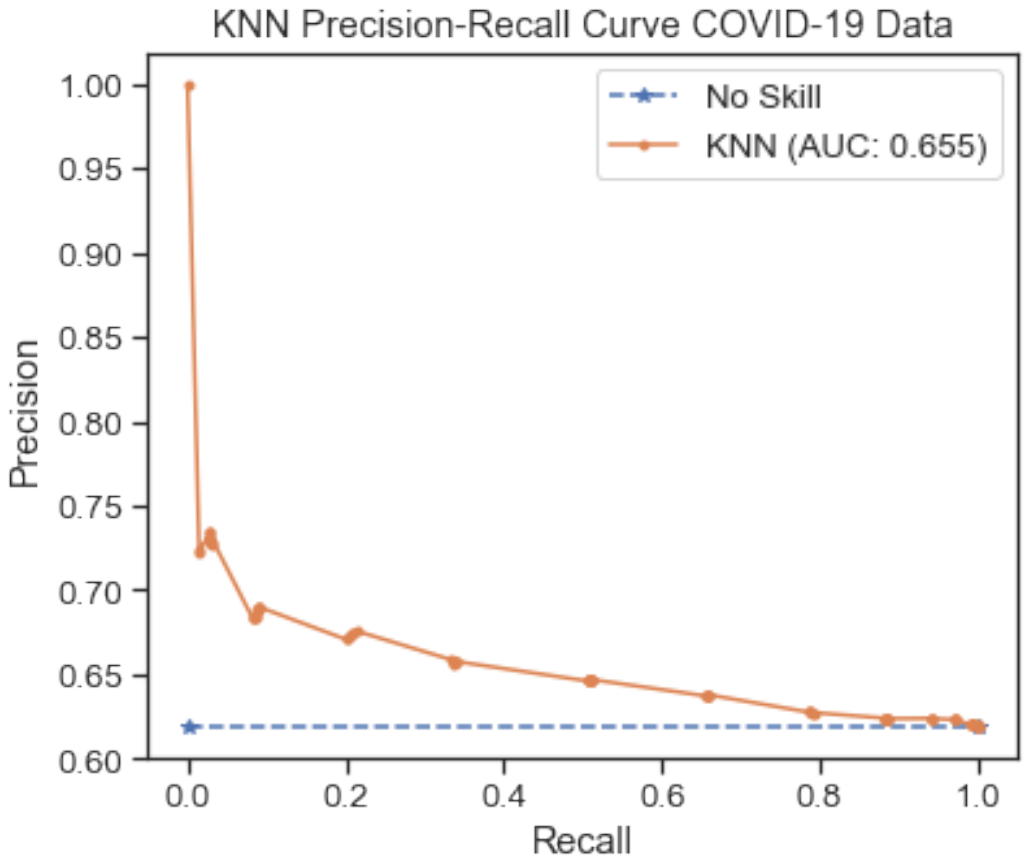


K-Neighbors

```
In [31]: KNN = KNeighborsClassifier(algorithm=algorithm, n_neighbors=n_neighbors, weights=weig
         yhat = KNN.predict(testDataX)


         ## PRECISION - RECALL CURVE

         # predict probabilities AND keep probabilities for the positive outcome only
         LLL = KNN.predict_proba(testDataX)
         LLL = LLL[:, 1]

         # get precision and recall probabilities
         knn_precision , knn_recall , _ = metrics.precision_recall_curve(testDatay, LLL)

         # summarize scores
         accuracy  = metrics.accuracy_score(testDatay, yhat)
         mcc       = metrics.matthews_corrcoef(testDatay, yhat)
```

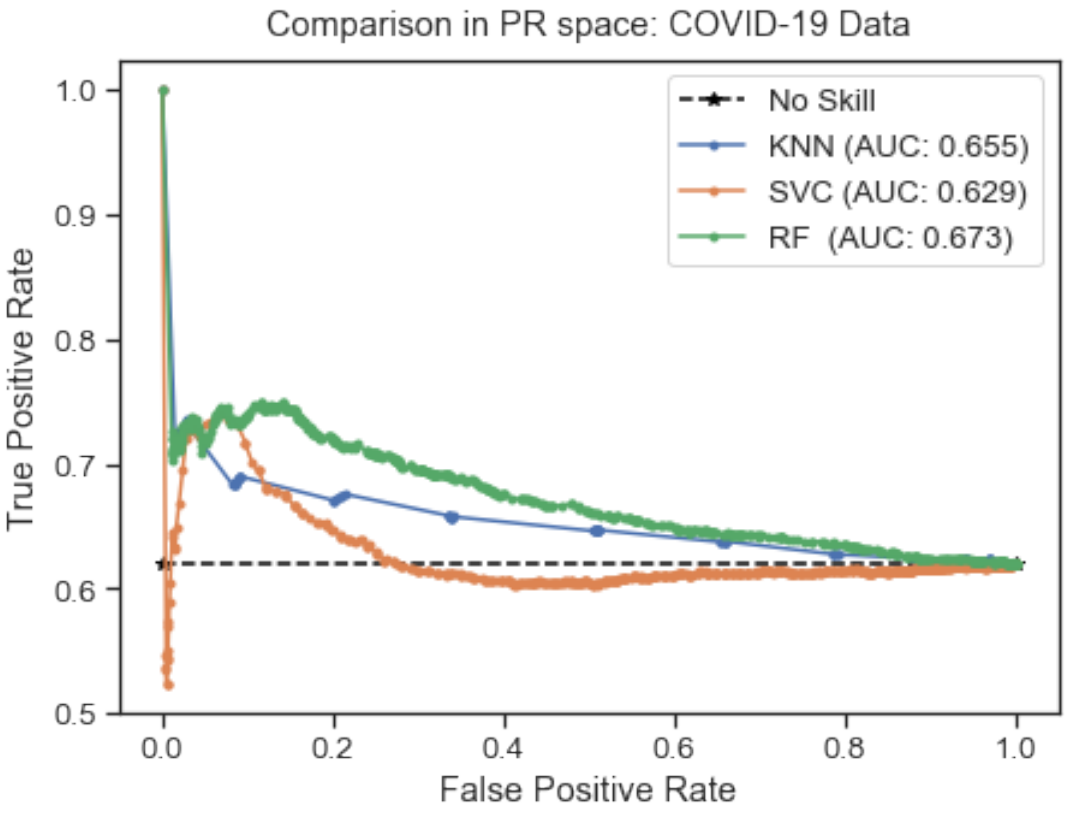
```
error     = metrics.zero_one_loss(testDatay, yhat)

auc       = metrics.auc(knn_recall, knn_precision)

recall    = metrics.recall_score(testDatay, yhat)
precision = metrics.precision_score(testDatay, yhat)
f1score   = metrics.f1_score(testDatay, yhat)

print()
print(pd.DataFrame(metrics.confusion_matrix(testDatay, yhat), index=['True Neg (-) ',
print()
print('KNN: \n accuracy=%.3f \n mcc=%.3f \n error=%.3f \n auc=%.3f \n recall=%.3f \n p


# define a label to use for the graph
LABEL_KNN = 'KNN (AUC: '  +  str(np.round(auc, decimals=3))  + ')'

# plot the precision-recall curves
no_skill = len(testDatay[testDatay==1]) / len(testDatay)

thisfigure = pyplot.figure(figsize=[6, 5])
pyplot.plot([0, 1], [no_skill, no_skill], linestyle='--', label='No Skill', marker='*
pyplot.plot(knn_recall, knn_precision, marker='.', label=LABEL_KNN)

# axis labels
pyplot.title('KNN Precision-Recall Curve COVID-19 Data')
pyplot.xlabel('Recall')
pyplot.ylabel('Precision')
pyplot.legend()
pyplot.show()
thisfigure.savefig(f'{FIGPATH}KNN_COVID_PRCurve.png')
```

```
               pred (-)  pred (+)
True Neg (-)        972      3050
True Pos (+)       1399      5140

KNN:
 accuracy=0.579
 mcc=0.032
 error=0.421
 auc=0.655
 recall=0.786
 precision=0.628
 f1score=0.698
```



Combined balanced models - Weighted Datasets

In [32]:
```
# plot the roc curve for the model
thisfigure = pyplot.figure(figsize=[7, 5])

pyplot.plot([0, 1], [no_skill, no_skill], linestyle='--', label='No Skill', marker='*

pyplot.plot(knn_recall, knn_precision, marker='.', label=LABEL_KNN, color='C0')  # mo
pyplot.plot(sv_recall, sv_precision, marker='.', label=LABEL_SVC, color='C1')    # mo
pyplot.plot(rf_recall, rf_precision, marker='.', label=LABEL_RF, color='C2')     # mo

# axis labels
pyplot.title('Comparison in PR space: COVID-19 Data', pad=10)
pyplot.xlabel('False Positive Rate')
pyplot.ylabel('True Positive Rate')
pyplot.legend(loc='best')
pyplot.show()
thisfigure.savefig(f'{FIGPATH}Combined_Models_weighted_Data_COVID.png')
```


Comparison in PR space: COVID-19 Data
No Skill
KNN (AUC: 0.655)
SVC (AUC: 0.629)
RF (AUC: 0.673)
True Positive Rate
False Positive Rate
1.0
0.9
0.8
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0

Malaria01_Preprocess

April 30, 2021

1 MALARIA

1.1 Preprocessing: (Datasets: 2020-11-12)

Explore the malaria dataset and filter out usable data for model building. Remember data is comming from three separate files `.csv` files: `MalariaDemographics`, `MalariaResults` and `MalariaExtra1`

In [1]:
```
from matplotlib import pyplot
import seaborn as sns
%matplotlib inline

import statistics as stats
import pandas as pd
import numpy as np
import random

import ipystata  # necessary to run stata commands
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.impute import KNNImputer

from sklearn.preprocessing import StandardScaler, MinMaxScaler

# To ignore warnings in the notebook
import warnings
warnings.filterwarnings('ignore')

pd.set_option('display.max_rows', 100)
```

Constants

In [2]:
```
PATH = "D:/THE UNIVERSITY OF WITWATERSRAND/16.2.JupyterWorks/msc_report/data/"
FIGPATH = "D:/THE UNIVERSITY OF WITWATERSRAND/16.REPORT/Figures/"

TESTSIZE = .25
SEED = 888
```

```
In [3]: # %cd D:/
        # %cd D:/THE UNIVERSITY OF WITWATERSRAND/16.2.JupyterWorks/msc_report
        # %ls "data/"
```

2 Extraction and aggregation

2.0.1 Dataset 1: MalariaDemographics

```
In [4]: # Fetch the demographics file
        %time demog_raw = pd.read_csv(f"{PATH}MalariaDemographics.csv", sep="æ")
```

```
Wall time: 3.4 s
```

```
In [5]: # Explore the size of the dataset
        demog_raw.sample(3, random_state = random.randint(1,30))
        print(demog_raw.shape)
```

```
(222805, 10)
```

```
In [6]: # How do variables look like?
        demog_raw.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 222805 entries, 0 to 222804
Data columns (total 10 columns):
EPISODE_NO                  222802 non-null object
AGE_TESTED_YEARS            222805 non-null object
GENDER                      222805 non-null object
NOTE                        92517 non-null object
IN_OUT_PATIENT              222792 non-null object
FACILITY_NAME               222612 non-null object
SUB_DISTRICT_NAME           222612 non-null object
DISTRICT_NAME               222609 non-null object
PROVINCE_NAME               222609 non-null object
DATE_SPECIMEN_REGISTERED    222799 non-null object
dtypes: object(10)
memory usage: 17.0+ MB
```

```
In [7]: # Reset column names for ease
        # demog_raw.rename(columns=lambda x: x.lower(), inplace=True)
        demog_raw.rename(str.lower, axis='columns', inplace=True)
```

```
In [8]: # AGE TESTED
        # ERROR! ValueError: Unable to parse string "N" at position 41208
        demog_raw['age_tested_years'] = pd.to_numeric(demog_raw['age_tested_years'], downcast=
        print('Mean Age at test: ', np.round(demog_raw.age_tested_years.mean(), decimals=2))
```

```
Mean Age at test:  26.94
```

```
In [9]: # GENDER: Replace gender values that do not make sense

        # demog_raw.gender.value_counts()
        demog_raw['gender'].replace({'HEIDEVELD EMERGENCY CENTRE': np.nan,
                                     'HEIDELBERG HOSPITAL':np.nan,
                                     'TEMBISA HOSPITAL': np.nan}, inplace=True)
        demog_raw.gender.value_counts(dropna=False)
```

```
Out[9]: M      113568
        F      104563
        U        4671
        NaN         3
        Name: gender, dtype: int64
```

```
In [10]: # NOTE:
         # demog_raw.note.value_counts().sample(5, random_state = random.randint(1,15))
         # The variable `note` is a free-text column. We are dropping it from the list of vari

         demog_raw.drop(columns=['note'], inplace=True)
```

```
In [11]: # IN/OUT PATIENT?
         print(demog_raw.in_out_patient.value_counts(dropna=False))
         demog_raw['in_out_patient'].replace({'SEDIBENG': np.nan,
                                              'EKURHULENI METRO': np.nan,
                                              'O': np.nan,
                                              'CITY OF CAPE TOWN METRO': np.nan}, inplace=True)
         demog_raw.in_out_patient.value_counts(dropna=False)
```

```
Y                          116109
N                          106679
NaN                            13
CITY OF CAPE TOWN METRO         1
EKURHULENI METRO                1
O                               1
SEDIBENG                        1
Name: in_out_patient, dtype: int64
```

```
Out[11]: Y      116109
         N      106679
         NaN        17
         Name: in_out_patient, dtype: int64
```

```
In [12]: # FACILITY: First 10
         demog_raw.facility_name.value_counts(dropna=False).sort_values(ascending=False).head(
```

```
Out[12]: DONALD FRASER HOSPITAL    16822
         NKHENSANI HOSPITAL        12542
         ROB FERREIRA HOSPITAL     11153
         TSHILIDZINI HOSPITAL      10716
         MALAMULELE HOSPITAL       10345
         SILOAM HOSPITAL            8019
         TEMBISA HOSPITAL           6529
         HELEN JOSEPH HOSPITAL      5900
         KALAFONG HOSPITAL          5760
         ELIM HOSPITAL              5656
         Name: facility_name, dtype: int64

In [13]: # SUB-DISTRICT; top 10
         demog_raw.sub_district_name.value_counts(dropna=False).sort_values(ascending=False).he

Out[13]: THULAMELA                    27813
         CITY OF MBOMBELA             16806
         MAKHADO                      14108
         GREATER GIYANI               12778
         COLLINS CHABANE              10367
         GREATER TZANEEN               9473
         BUSHBUCKRIDGE                 9175
         JOHANNESBURG B                7266
         EKURHULENI SUB-DISTRICT B     6538
         ETHEKWINI                     6099
         Name: sub_district_name, dtype: int64

In [14]: # DISTRICT
         demog_raw.district_name.value_counts(dropna=False).sort_values(ascending=False).head(

Out[14]: VHEMBE                        56420
         MOPANI                        33671
         EHLANZENI                     32130
         EKURHULENI METRO              17520
         CITY OF JOHANNESBURG METRO     9097
         CAPRICORN                      8314
         WEST RAND                      7434
         CITY OF TSHWANE METRO          7018
         CITY OF CAPE TOWN METRO        6339
         ETHEKWINI METRO                6099
         Name: district_name, dtype: int64

In [15]: # PROVINCE
         print(demog_raw.province_name.value_counts(dropna=False))
         demog_raw['province_name'].replace(to_replace='UNKNOWN', value=np.nan, inplace=True)

LIMPOPO          106235
GAUTENG           43341
MPUMALANGA        38509
```

Malaria02_Models

April 30, 2021

1 MALARIA DATASETS

This script contains two parts 1) Models optimised for Balanced datasets and 2) Models optimised for imbalanced datasets accounting for class weights in the target

In [1]:
```
import pandas as pd
import numpy as np
from collections import Counter

from sklearn.utils import resample

from sklearn.model_selection import GridSearchCV, StratifiedKFold
from sklearn.model_selection import cross_val_score

from sklearn.svm import SVC, LinearSVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier

from sklearn.calibration import CalibratedClassifierCV

from sklearn import metrics

from matplotlib import pyplot
import seaborn as sns; sns.set(font_scale=1.2, style="ticks", color_codes=True)
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

Constants

In [2]:
```
PATH       = 'D:/THE UNIVERSITY OF WITWATERSRAND/16.2.JupyterWorks/msc_report/data/'
FIGPATH    = "D:/THE UNIVERSITY OF WITWATERSRAND/16.REPORT/Figures/"

TRAIN_DATA = 'malaria_train_valid'
TEST_DATA  = 'malaria_test'

SEED       = 888
cv         = StratifiedKFold(n_splits=5)
scoring    = metrics.make_scorer('f1')
```

Test Data

```
In [3]: testData = pd.read_csv(f"{PATH}{TEST_DATA}.csv")
        print(f'SHAPE: {testData.shape}')
        testData.sample(2)
        print(f'Class distributions : {Counter(testData.target)}')

        testDataX = testData[[j for j in testData.columns if j != 'target']]
        testDatay = testData.iloc[:, 0]
```

```
SHAPE: (10140, 29)
Class distributions : Counter({1: 9541, 0: 599})
```

2 Balanced Data Models

2.1 Models optimised on Resampled Data : Observed : NotObserved [1:1]

Training Data

```
In [4]: data = pd.read_csv(f"{PATH}{TRAIN_DATA}.csv")
        print(f'SHAPE: {data.shape}')

        # Check the distribution in the outcome
        print(f'Class distributions: {Counter(data.target)}')

        # X = malaria.drop('target', axis='columns')
        X = data[[j for j in data.columns if j != 'target']]
        y = data.iloc[:, 0]
```

```
SHAPE: (30417, 29)
Class distributions: Counter({1: 28621, 0: 1796})
```

Re-sample to Balance the Training Data

```
In [5]: # Get sample sizes
        data.target.value_counts()
```

```
Out[5]: 1    28621
        0     1796
        Name: target, dtype: int64
```

```
In [6]: # REF: https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your
        # Down-sample the majority class using the length of minority class as a measure

        pos_samples = data.loc[data.target==1, :]     # 28621
        neg_samples = data.loc[data.target==0, :]     # 1796
```

```
        data_downsampled = resample(pos_samples,
                                    n_samples = len(neg_samples),
                                    random_state = SEED,
                                    replace=False
                                   )
        print(f'\nPositive/Majority class downsampled: {len(data_downsampled)}')

        # merge the downsample to back to the initially minority sample. We expect a 50:50 bal
        dataSampled = pd.concat([data_downsampled, pos_samples], axis=0)
        print()
        print(f'Malaria Obs:Nobs {dataSampled.target.value_counts(normalize=True).ravel()}')
        print(dataSampled.target.value_counts())


        # USE THE RESAMPLED DATA
        data = dataSampled.copy(deep=True)
```

```
Positive/Majority class downsampled: 1796

Malaria Obs:Nobs [1.]
1    30417
Name: target, dtype: int64
```

2.1.1 Default CV Estimates - Training Data

```
In [7]: # LinearSVC
        # Because of the time complexities that come with the defaut 'rbf' kernel, we set this

        score = cross_val_score(LinearSVC(), X=X, y=y, cv=cv, scoring='f1')
        print(score)
        print(f'SVC - F1-Score: {np.round(score.mean(), decimals=3)}')
```

```
[0.96959946 0.9695122  0.96959431 0.96959431 0.96959431]
SVC - F1-Score: 0.97
```

```
In [8]: # RF

        score = cross_val_score(RandomForestClassifier(), X=X, y=y, cv=cv, scoring='f1')
        print(score)
        print(f'Random Forest -  F1-Score: {np.round(score.mean(), decimals=3)}')
```

```
[0.96757081 0.966701   0.96774194 0.96844452 0.96793003]
Random Forest -  F1-Score: 0.968
```

```
In [9]: # KNN
```

```
        score = cross_val_score(KNeighborsClassifier(), X=X, y=y, cv=cv, scoring='f1')
        print(score)
        print(f'KNN - F1-Score: {np.round(score.mean(), decimals=3)}')
```

```
[0.96836525 0.96668956 0.96858684 0.9682227  0.96801303]
KNN - F1-Score: 0.968
```

2.1.2 GridSearch CV estimates - Training data

In [10]:
```
# LinearSVC

penalty = ['l1', 'l2']
C       = [1, 10, 25]
loss    = ['hinge', 'hinge_squared']
param_grid = dict(C=C, penalty=penalty, loss=loss)

svcgs = GridSearchCV( LinearSVC(), param_grid=param_grid,  cv=cv, scoring='f1', n_jobs

svcgs.fit(X, y)
print(svcgs.best_score_)
print(svcgs.best_estimator_)

# Best parameters
C      = svcgs.best_params_['C']
penalty  = svcgs.best_params_['penalty']
loss = svcgs.best_params_['loss']
```

```
0.9695789157091352
LinearSVC(C=1, loss='hinge')
```

In [11]:
```
# RANDOM FOREST

n_estimators      = [10, 12, 14, 18]
min_samples_split = [2,3,4]
criterion         = ["gini", "entropy"]
bootstrap         = [True, False]
param_grid = dict(n_estimators=n_estimators, min_samples_split=min_samples_split, cri

rfgs = GridSearchCV( RandomForestClassifier(), param_grid=param_grid,  cv=cv, scoring

rfgs.fit(X, y)
print(rfgs.best_score_)
print(rfgs.best_estimator_)

# Best parameters
bootstrap         = rfgs.best_params_['bootstrap']
criterion         = rfgs.best_params_['criterion']
```

```
min_samples_split = rfgs.best_params_['min_samples_split']
n_estimators      = rfgs.best_params_['n_estimators']
```

```
0.9683438929628971
RandomForestClassifier(criterion='entropy', min_samples_split=4,
                       n_estimators=14)
```

```
In [12]: # K-NEIGHBORS

         n_neighbors = list(np.arange(5,16,1))
         algorithm = ['ball_tree', 'kd_tree', 'brute']
         weights = ['uniform']
         param_grid = dict(n_neighbors=n_neighbors, weights=weights, algorithm=algorithm)

         knngs = GridSearchCV( KNeighborsClassifier(), param_grid=param_grid,  cv=cv, scoring=

         knngs.fit(X, y)
         print(knngs.best_score_)
         print(knngs.best_estimator_)

         # Best parameters
         algorithm   = knngs.best_params_['algorithm']
         n_neighbors = knngs.best_params_['n_neighbors']
         weights     = knngs.best_params_['weights']
```

```
0.9699606459705661
KNeighborsClassifier(algorithm='ball_tree', n_neighbors=13)
```

2.2 Scores on out-of-sample data

2.2.1 Default Models

```
In [13]: # Default - LinearSVC

         # svc = LinearSVC()
         # svc.fit(X, y)

         SV = CalibratedClassifierCV( LinearSVC() )
         SV.fit(X, y)

         yhat = SV.predict(testDataX)

         precision = metrics.precision_score(testDatay, yhat)
         recall    = metrics.recall_score(testDatay, yhat)
         f1score   = metrics.f1_score(testDatay, yhat)

         # predict probabilities AND keep probabilities for the positive outcome only
```

```
        AAA = SV.predict_proba(testDataX)
        AAA = AAA[:, 1]

        # get auc and error
        sv_auc = metrics.roc_auc_score(testDatay, AAA)
        error  = metrics.zero_one_loss(testDatay, yhat)

        print('SVC: Precision=%.3f  Recall=%.3f  f1=%.3f  auc=%.3f  error=%.3f' % (precision,
```

```
SVC: Precision=0.948  Recall=0.993  f1=0.970  auc=0.798  error=0.058
```

```
In [14]: # Default - RF

        RF = RandomForestClassifier().fit(X, y)
        yhat = RF.predict(testDataX)

        precision = metrics.precision_score(testDatay, yhat)
        recall    = metrics.recall_score(testDatay, yhat)
        f1score   = metrics.f1_score(testDatay, yhat)

        # predict probabilities AND keep probabilities for the positive outcome only
        BBB = RF.predict_proba(testDataX)
        BBB = BBB[:, 1]

        # get auc and error
        rfc_auc = metrics.roc_auc_score(testDatay, BBB)
        error  = metrics.zero_one_loss(testDatay, yhat)

        print('RFC: Precision=%.3f  Recall=%.3f  f1=%.3f  auc=%.3f  error=%.3f' % (precision,
```

```
RFC: Precision=0.949  Recall=0.990  f1=0.969  auc=0.809  error=0.060
```

```
In [15]: # Default - KNN

        KNN = KNeighborsClassifier().fit(X, y)
        yhat = KNN.predict(testDataX)

        precision = metrics.precision_score(testDatay, yhat)
        recall    = metrics.recall_score(testDatay, yhat)
        f1score   = metrics.f1_score(testDatay, yhat)

        # predict probabilities AND keep probabilities for the positive outcome only
        CCC = KNN.predict_proba(testDataX)
        CCC = CCC[:, 1]

        # get auc and error
        knn_auc = metrics.roc_auc_score(testDatay, CCC)
```

```
        error  = metrics.zero_one_loss(testDatay, yhat)

        print('KNN: Precision=%.3f  Recall=%.3f  f1=%.3f  auc=%.3f  error=%.3f' % (precision,
```

```
KNN: Precision=0.949  Recall=0.986  f1=0.967  auc=0.717  error=0.063
```

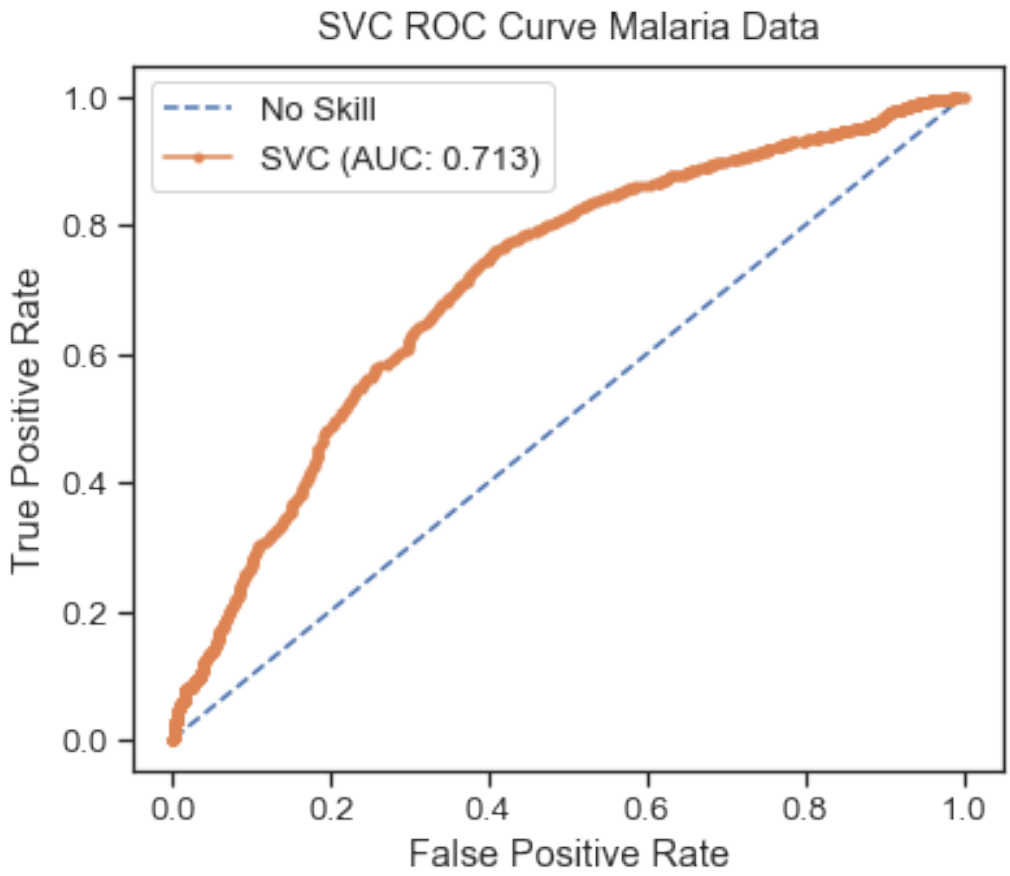
2.2.2 Optimised Models

Support Vector Classifier

In [16]:
```
        # Get the parameters of the best parameters and fit the model THEN predict
        # yhat = svcgs.predict(testDataX)

        # SV = SVC(C=C, gamma=gamma, kernel=kernel, probability=True)
        # SV.fit(X, y)

        SV = CalibratedClassifierCV( LinearSVC(C=C, penalty=penalty, loss=loss) )
        SV.fit(X, y)

        yhat = SV.predict(testDataX)

        accuracy  = metrics.accuracy_score(testDatay, yhat)
        precision = metrics.precision_score(testDatay, yhat)
        recall    = metrics.recall_score(testDatay, yhat)
        f1score   = metrics.f1_score(testDatay, yhat)

        # predict probabilities AND keep probabilities for the positive outcome only
        DDD = SV.predict_proba(testDataX)
        DDD = DDD[:, 1]


        # generate a no skill prediction (majority class)
        noskill_probs = [0 for _ in range(len(testDatay))]

        # # calculate scores
        # noskill_auc = metrics.roc_auc_score(testDatay, noskill_probs)

        print(f'predicted   : {Counter(yhat)}')

        accuracy  = metrics.accuracy_score(testDatay, yhat)
        mcc       = metrics.matthews_corrcoef(testDatay, yhat)
        error     = metrics.zero_one_loss(testDatay, yhat)

        auc       = metrics.roc_auc_score(testDatay, DDD)

        recall    = metrics.recall_score(testDatay, yhat)
        precision = metrics.precision_score(testDatay, yhat)
```

```
        f1score   = metrics.f1_score(testDatay, yhat)

        print()
        print(pd.DataFrame(metrics.confusion_matrix(testDatay, yhat), index=['True Neg (-) ',
        print()
        print('SVC BALANCED: \n accuracy=%.3f \n mcc=%.3f \n error=%.3f \n auc=%.3f \n recall=

        # define a label to use for the graph
        LABEL_SVC = 'SVC (AUC: '  +  str(np.round(auc, decimals=3))  + ')'

        # calculate roc curves
        noskill_fpr, noskill_tpr, _ = metrics.roc_curve(testDatay, noskill_probs)
        sv_model_fpr, sv_model_tpr, _ = metrics.roc_curve(testDatay, DDD)

        # plot the roc curve for the model
        thisfigure = pyplot.figure(figsize=[6, 5])
        pyplot.plot(noskill_fpr, noskill_tpr, linestyle='--', label='No Skill')
        pyplot.plot(sv_model_fpr, sv_model_tpr, marker='.', label=LABEL_SVC)

        # axis labels
        pyplot.title('SVC ROC Curve Malaria Data', pad=10)
        pyplot.xlabel('False Positive Rate')
        pyplot.ylabel('True Positive Rate')
        pyplot.legend()
        pyplot.show()
        thisfigure.savefig(f'{FIGPATH}SVC_Balanced_Malaria.png')
```

```
predicted   : Counter({1: 10140})

               pred (-)  pred (+)
True Neg (-)          0       599
True Pos (+)          0      9541

SVC BALANCED:
 accuracy=0.941
 mcc=0.000
 error=0.059
 auc=0.713
 recall=1.000
 precision=0.941
 f1score=0.970
```



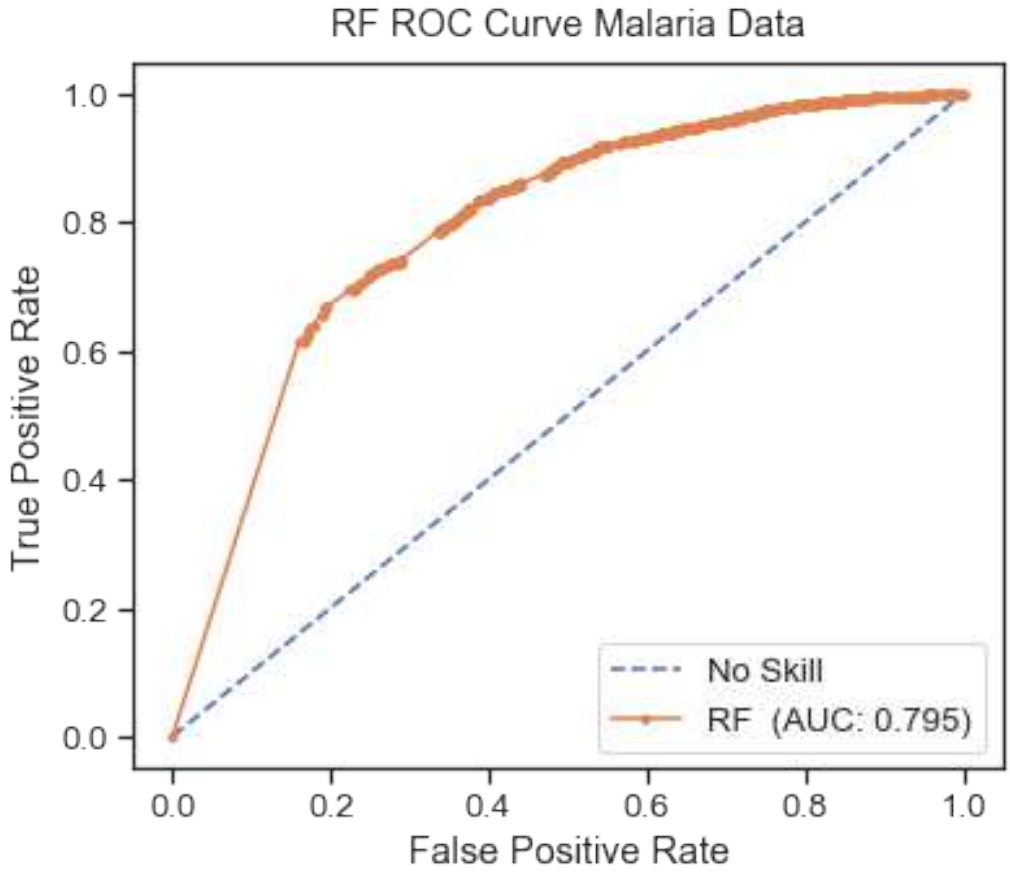
Random Forest Classifier

```
In [17]: # Get the parameters of the best parameters and fit the model THEN predict
         RF = RandomForestClassifier(n_estimators=n_estimators, min_samples_split=min_samples_s
         yhat = RF.predict(testDataX)

         accuracy  = metrics.accuracy_score(testDatay, yhat)
         precision = metrics.precision_score(testDatay, yhat)
         recall    = metrics.recall_score(testDatay, yhat)
         f1score   = metrics.f1_score(testDatay, yhat)

         # predict probabilities AND keep probabilities for the positive outcome only
         EEE = RF.predict_proba(testDataX)
         EEE = EEE[:, 1]


         # generate a no skill prediction (majority class)
         noskill_probs = [0 for _ in range(len(testDatay))]
```

```
        # calculate scores
        # noskill_auc = metrics.roc_auc_score(testDatay, noskill_probs)

        print(f'predicted   : {Counter(yhat)}')

        accuracy  = metrics.accuracy_score(testDatay, yhat)
        mcc       = metrics.matthews_corrcoef(testDatay, yhat)
        error     = metrics.zero_one_loss(testDatay, yhat)

        auc       = metrics.roc_auc_score(testDatay, EEE)

        recall    = metrics.recall_score(testDatay, yhat)
        precision = metrics.precision_score(testDatay, yhat)
        f1score   = metrics.f1_score(testDatay, yhat)

        print()
        print(pd.DataFrame(metrics.confusion_matrix(testDatay, yhat), index=['True Neg (-) ',
        print()
        print('RF BALANCED: \n accuracy=%.3f \n mcc=%.3f \n error=%.3f \n auc=%.3f \n recall=%



        # define a label to use for the graph
        LABEL_RF = 'RF  (AUC: '  +  str(np.round(auc, decimals=3))  + ')'

        # calculate roc curves
        noskill_fpr, noskill_tpr, _ = metrics.roc_curve(testDatay, noskill_probs)
        rf_model_fpr, rf_model_tpr, _ = metrics.roc_curve(testDatay, EEE)

        # plot the roc curve for the model
        thisfigure = pyplot.figure(figsize=[6, 5])
        pyplot.plot(noskill_fpr, noskill_tpr, linestyle='--', label='No Skill')
        pyplot.plot(rf_model_fpr, rf_model_tpr, marker='.', label=LABEL_RF)

        # axis labels
        pyplot.title('RF ROC Curve Malaria Data', pad=10)
        pyplot.xlabel('False Positive Rate')
        pyplot.ylabel('True Positive Rate')
        pyplot.legend()   # show the legend
        pyplot.show()     # show the plot
        thisfigure.savefig(f'{FIGPATH}RF_Balanced_Malaria.png')
```

```
predicted   : Counter({1: 9953, 0: 187})

               pred (-)  pred (+)
True Neg (-)         89       510
True Pos (+)         98      9443
```

```
RF BALANCED:
 accuracy=0.940
 mcc=0.242
 error=0.060
 auc=0.795
 recall=0.990
 precision=0.949
 f1score=0.969
```

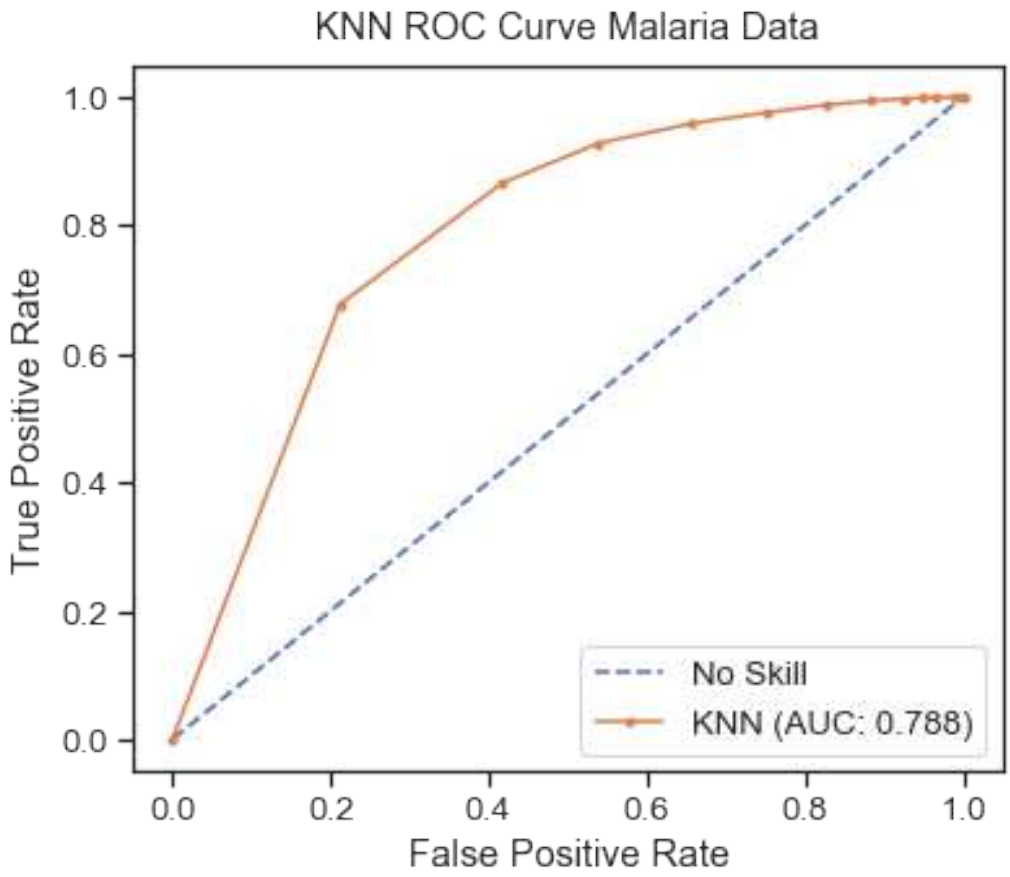


KNN

```
In [18]: KNN = KNeighborsClassifier(algorithm=algorithm, n_neighbors=n_neighbors, weights=weigh
         yhat = KNN.predict(testDataX)

         accuracy  = metrics.accuracy_score(testDatay, yhat)
         precision = metrics.precision_score(testDatay, yhat)
         recall    = metrics.recall_score(testDatay, yhat)
         f1score   = metrics.f1_score(testDatay, yhat)
```

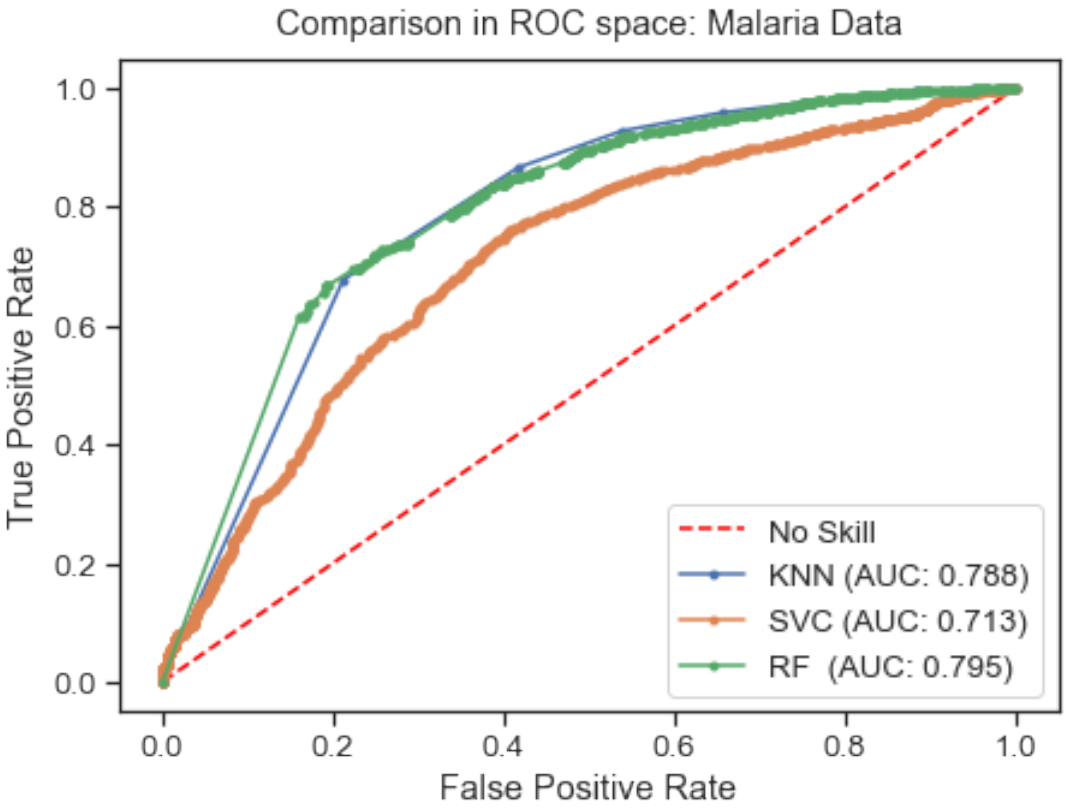
```
### PRECISION-RECALL-CURVES

# predict probabilities AND keep probabilities for the positive outcome only
FFF = KNN.predict_proba(testDataX)
FFF = FFF[:, 1]


# generate a no skill prediction (majority class)
noskill_probs = [0 for _ in range(len(testDatay))]

# calculate scores
# noskill_auc = metrics.roc_auc_score(testDatay, noskill_probs)

print(f'predicted   : {Counter(yhat)}')


accuracy  = metrics.accuracy_score(testDatay, yhat)
mcc       = metrics.matthews_corrcoef(testDatay, yhat)
error     = metrics.zero_one_loss(testDatay, yhat)

auc       = metrics.roc_auc_score(testDatay, FFF)

recall    = metrics.recall_score(testDatay, yhat)
precision = metrics.precision_score(testDatay, yhat)
f1score   = metrics.f1_score(testDatay, yhat)

print()
print(pd.DataFrame(metrics.confusion_matrix(testDatay, yhat), index=['True Neg (-) ',
print()
print('KNN: \n accuracy=%.3f \n mcc=%.3f \n error=%.3f \n auc=%.3f \n recall=%.3f \n p



# define a label to use for the graph
LABEL_KNN = 'KNN (AUC: '  +  str(np.round(auc, decimals=3))  + ')'

# calculate roc curves
noskill_fpr, noskill_tpr, _ = metrics.roc_curve(testDatay, noskill_probs)
knn_model_fpr, knn_model_tpr, _ = metrics.roc_curve(testDatay, FFF)

# plot the roc curve for the model
thisfigure = pyplot.figure(figsize=[6, 5])
pyplot.plot(noskill_fpr, noskill_tpr, linestyle='--', label='No Skill')
pyplot.plot(knn_model_fpr, knn_model_tpr, marker='.', label=LABEL_KNN)

# axis labels
pyplot.title('KNN ROC Curve Malaria Data', pad=10)
pyplot.xlabel('False Positive Rate')
```

```
        pyplot.ylabel('True Positive Rate')
        pyplot.legend()
        pyplot.show()
        thisfigure.savefig(f'{FIGPATH}KNN_Balanced_Malaria.png')
predicted   : Counter({1: 10009, 0: 131})

               pred (-)  pred (+)
True Neg (-)         70       529
True Pos (+)         61      9480

KNN:
 accuracy=0.942
 mcc=0.231
 error=0.058
 auc=0.788
 recall=0.994
 precision=0.947
 f1score=0.970
```



Combined balanced models - Balanced Datasets

In [19]:
```
# plot the roc curve for the model
thisfigure = pyplot.figure(figsize=[7, 5])
pyplot.plot(noskill_fpr, noskill_tpr, linestyle='--', label='No Skill', color='Red')
pyplot.plot(knn_model_fpr, knn_model_tpr, marker='.', label=LABEL_KNN, color='C0')
pyplot.plot(sv_model_fpr, sv_model_tpr, marker='.', label=LABEL_SVC, color='C1')
pyplot.plot(rf_model_fpr, rf_model_tpr, marker='.', label=LABEL_RF, color='C2')

# axis labels
pyplot.title('Comparison in ROC space: Malaria Data', pad=10)
pyplot.xlabel('False Positive Rate')
pyplot.ylabel('True Positive Rate')
pyplot.legend(loc='lower right')
pyplot.show()
thisfigure.savefig(f'{FIGPATH}Combined_Models_Balanced_Data_Malaria.png')
```



3 Weighted Models

3.1 Accounting for distribution weights in the target: Observed : NotObserved [16:1]

Training Data

```
In [20]: data = pd.read_csv(f"{PATH}{TRAIN_DATA}.csv")
         print(f'SHAPE: {data.shape}')

         # Check the distribution in the outcome
         print(f'Class distributions: {Counter(data.target)}')

         # X = malaria.drop('target', axis='columns')
         X = data[[j for j in data.columns if j != 'target']]
         y = data.iloc[:, 0]
```

```
SHAPE: (30417, 29)
Class distributions: Counter({1: 28621, 0: 1796})
```

```
In [21]: # Default - SVC

         score = cross_val_score(SVC(kernel='linear'), X=X, y=y, cv=cv, scoring='f1')
         print(score)
         print(f'SVC F1-Score: {np.round(score.mean(), decimals=3)}')
```

```
[0.96959946 0.9695122  0.96959431 0.96959431 0.96959431]
SVC F1-Score: 0.97
```

```
In [22]: # Default - Random Forests

         score = cross_val_score(RandomForestClassifier(), X=X, y=y, cv=cv, scoring='f1')
         print(score)
         print(f'CrossValidated F1-Score: {np.round(score.mean(), decimals=3)}')
```

```
[0.9675597  0.96668956 0.96897497 0.96792453 0.967753  ]
CrossValidated F1-Score: 0.968
```

```
In [23]: # Default - KNN

         score = cross_val_score(KNeighborsClassifier(), X=X, y=y, cv=cv, scoring='f1')
         print(score)
         print(f'KNN - F1-Score: {np.round(score.mean(), decimals=3)}')
```

```
[0.96836525 0.96668956 0.96858684 0.9682227  0.96801303]
KNN - F1-Score: 0.968
```

3.1.1 GridSearch CV estimates - Training Data

In [24]:
```
# LinearSVC

penalty = ['l1', 'l2']
C       = [1, 10, 25]
loss    = ['hinge', 'hinge_squared']
param_grid = dict(C=C, penalty=penalty, loss=loss)

svcgs = GridSearchCV( LinearSVC(), param_grid=param_grid,  cv=cv, scoring='f1', n_jobs

svcgs.fit(X, y)
print(svcgs.best_score_)
print(svcgs.best_estimator_)

# Best parameters
C      = svcgs.best_params_['C']
penalty  = svcgs.best_params_['penalty']
loss = svcgs.best_params_['loss']
```

```
0.9695789157091352
LinearSVC(C=1, loss='hinge')
```

In [25]:
```
# RANDOM FOREST

n_estimators      = [10, 12, 14, 18]
min_samples_split = [2,3,4]
criterion         = ["gini", "entropy"]
bootstrap         = [True, False]
class_weight      = ['balanced']

param_grid = dict(n_estimators=n_estimators, min_samples_split=min_samples_split,
                  criterion=criterion, bootstrap=bootstrap, class_weight=class_weight

rfgs = GridSearchCV( RandomForestClassifier(), param_grid=param_grid,  cv=5, scoring=

rfgs.fit(X, y)
print(rfgs.best_score_)
print(rfgs.best_estimator_)

# Best parameters
bootstrap         = rfgs.best_params_['bootstrap']
criterion         = rfgs.best_params_['criterion']
min_samples_split = rfgs.best_params_['min_samples_split']
n_estimators      = rfgs.best_params_['n_estimators']
class_weight      = rfgs.best_params_['class_weight']
```

```
0.9669664039084072
RandomForestClassifier(class_weight='balanced', min_samples_split=3,
```

```
                    n_estimators=14)
```

In [26]:
```
# KNN

n_neighbors = list(np.arange(5,16,1))
algorithm   = ['ball_tree', 'kd_tree', 'brute']
weights     = ['distance']
param_grid  = dict(n_neighbors=n_neighbors, weights=weights, algorithm=algorithm)

knngs = GridSearchCV( KNeighborsClassifier(), param_grid=param_grid,  cv=4, scoring='

knngs.fit(X, y)
print(knngs.best_score_)
print(knngs.best_estimator_)

# Best parameters
algorithm   = knngs.best_params_['algorithm']
n_neighbors = knngs.best_params_['n_neighbors']
weights     = knngs.best_params_['weights']
```

```
0.9691020766986422
KNeighborsClassifier(algorithm='ball_tree', n_neighbors=14, weights='distance')
```

3.2 Scores on out-of-sample data

3.2.1 Default Models

In [27]:
```
# LinearSVC

SV = CalibratedClassifierCV( LinearSVC() )
SV.fit(X, y)
yhat = SV.predict(testDataX)

precision = metrics.precision_score(testDatay, yhat)
recall    = metrics.recall_score(testDatay, yhat)
f1score   = metrics.f1_score(testDatay, yhat)

# predict probabilities AND keep probabilities for the positive outcome only
GGG = SV.predict_proba(testDataX)
GGG = GGG[:, 1]

# get precision and recall probabilities
def_precision , def_recall , _ = metrics.precision_recall_curve(testDatay, GGG)
auc       = metrics.auc(def_recall, def_precision)
error     = metrics.zero_one_loss(testDatay, yhat)

print('SVC: Precision=%.3f  Recall=%.3f  f1=%.3f  auc=%.3f  error=%.3f' % (precision,
```

```
SVC: Precision=0.948  Recall=0.993  f1=0.970  auc=0.978  error=0.058
```

In [28]:
```
# Random Forest

RF = RandomForestClassifier()
RF.fit(X, y)
yhat = RF.predict(testDataX)

# get scores
yhat = RF.predict(testDataX)

precision = metrics.precision_score(testDatay, yhat)
recall    = metrics.recall_score(testDatay, yhat)
f1score   = metrics.f1_score(testDatay, yhat)

# predict probabilities AND keep probabilities for the positive outcome only
HHH = RF.predict_proba(testDataX)
HHH = HHH[:, 1]

# get precision and recall probabilities
def_precision , def_recall , _ = metrics.precision_recall_curve(testDatay, HHH)
auc       = metrics.auc(def_recall, def_precision)
error     = metrics.zero_one_loss(testDatay, yhat)

print('RFC: Precision=%.3f  Recall=%.3f  f1=%.3f  auc=%.3f  error=%.3f' % (precision,
```

```
RFC: Precision=0.948  Recall=0.988  f1=0.968  auc=0.984  error=0.062
```

In [29]:
```
# K-Neighbors

knn = KNeighborsClassifier().fit(X, y)
yhat = knn.predict(testDataX)

precision = metrics.precision_score(testDatay, yhat)
recall    = metrics.recall_score(testDatay, yhat)
f1score   = metrics.f1_score(testDatay, yhat)

# predict probabilities AND keep probabilities for the positive outcome only
III = knn.predict_proba(testDataX)
III = III[:, 1]

# get precision and recall probabilities
def_precision , def_recall , _ = metrics.precision_recall_curve(testDatay, III)
auc       = metrics.auc(def_recall, def_precision)
error     = metrics.zero_one_loss(testDatay, yhat)

print('KNN: Precision=%.3f  Recall=%.3f  f1=%.3f  auc=%.3f  error=%.3f' % (precision,
```

```
KNN: Precision=0.949  Recall=0.986  f1=0.967  auc=0.980  error=0.063
```

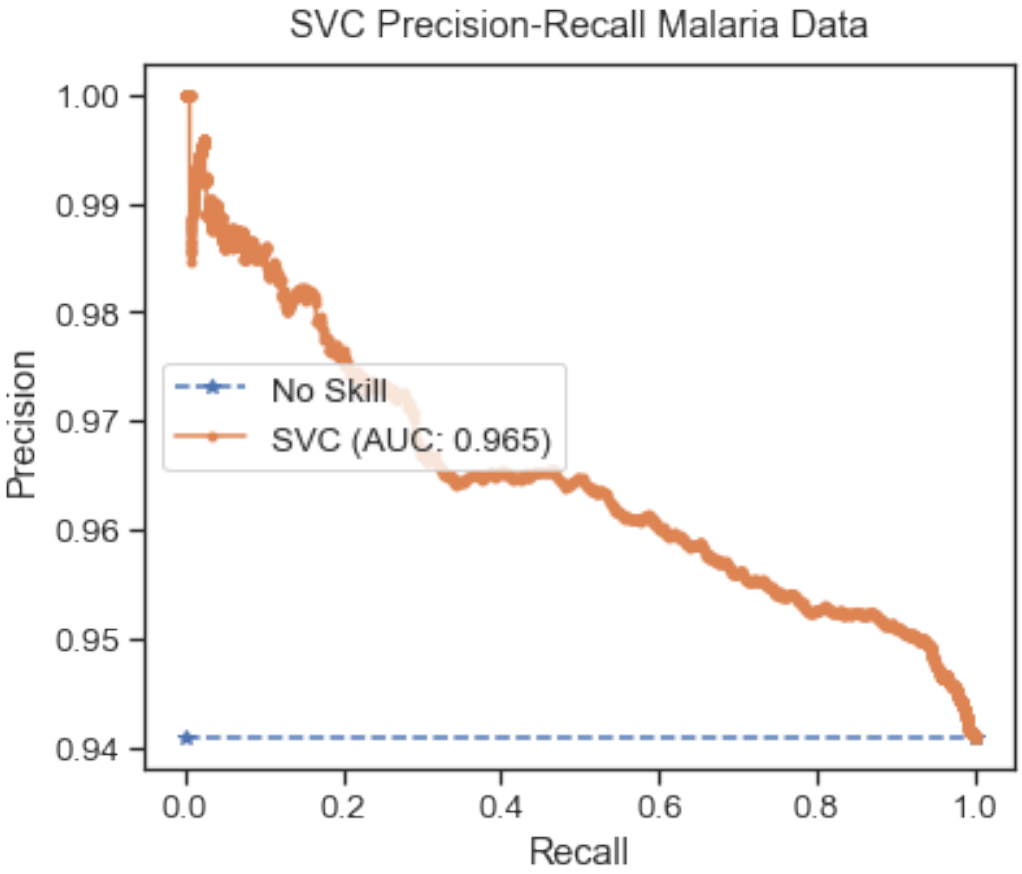
3.2.2 Optimised Models

Support Vector Classifier

```
In [30]: # Get the parameters of the best parameters and fit the model THEN predict
         # yhat = svcgs.predict(testDataX)

         SV = CalibratedClassifierCV( LinearSVC(C=C, penalty=penalty, loss=loss) )
         SV.fit(X, y)
         yhat = SV.predict(testDataX)

         precision = metrics.precision_score(testDatay, yhat)
         recall    = metrics.recall_score(testDatay, yhat)
         f1score   = metrics.f1_score(testDatay, yhat)

         ## PRECISION - RECALL CURVE

         # predict probabilities AND keep probabilities for the positive outcome only
         JJJ = SV.predict_proba(testDataX)
         JJJ = JJJ[:, 1]

         # get precision and recall probabilities
         sv_precision , sv_recall , _ = metrics.precision_recall_curve(testDatay, JJJ)


         # summarize scores
         accuracy  = metrics.accuracy_score(testDatay, yhat)
         mcc       = metrics.matthews_corrcoef(testDatay, yhat)
         error     = metrics.zero_one_loss(testDatay, yhat)

         auc       = metrics.auc(sv_recall, sv_precision)

         recall    = metrics.recall_score(testDatay, yhat)
         precision = metrics.precision_score(testDatay, yhat)
         f1score   = metrics.f1_score(testDatay, yhat)

         print()
         print(pd.DataFrame(metrics.confusion_matrix(testDatay, yhat), index=['True Neg (-) ',
         print()
         print('SVC: \n accuracy=%.3f \n mcc=%.3f \n error=%.3f \n auc=%.3f \n recall=%.3f \n p


         # define a label to use for the graph
         LABEL_SVC = 'SVC (AUC: '  +  str(np.round(auc, decimals=3))  + ')'
```

```
        # plot the precision-recall curves
        no_skill = len(testDatay[testDatay==1]) / len(testDatay)

        thisfigure = pyplot.figure(figsize=[6, 5])
        pyplot.plot([0, 1], [no_skill, no_skill], linestyle='--', label='No Skill', marker='*
        pyplot.plot(sv_recall, sv_precision, marker='.', label=LABEL_SVC)          # model

        # axis labels
        pyplot.title('SVC Precision-Recall Malaria Data', pad=10)
        pyplot.xlabel('Recall')
        pyplot.ylabel('Precision')
        pyplot.legend(loc='center left')
        pyplot.show()
        thisfigure.savefig(f'{FIGPATH}SVC_MAL_PRCurve.png')
```

```
               pred (-)  pred (+)
True Neg (-)          0       599
True Pos (+)          0      9541

SVC:
 accuracy=0.941
 mcc=0.000
 error=0.059
 auc=0.965
 recall=1.000
 precision=0.941
 f1score=0.970
```

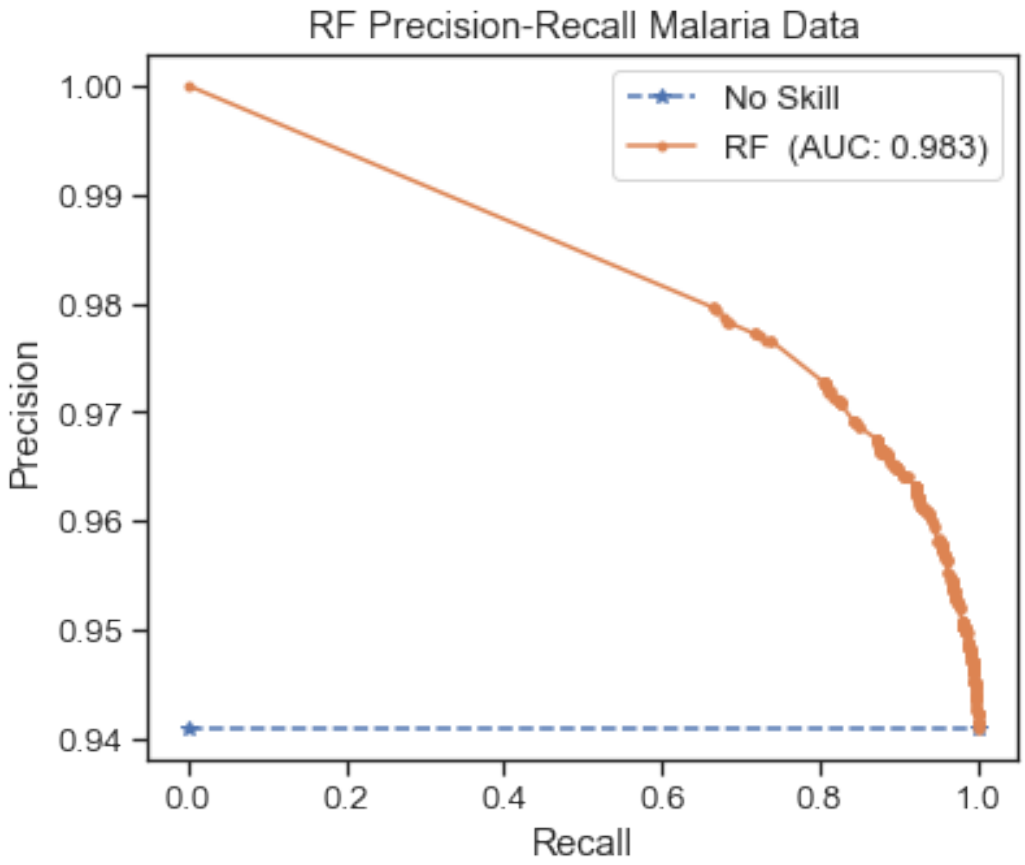


Random Forest Classifier

```
In [31]: # Get the parameters of the best parameters and fit the model THEN predict
         RF = RandomForestClassifier(n_estimators=n_estimators, min_samples_split=min_samples_s
                                     criterion=criterion, class_weight=class_weight, bootstr
         yhat = RF.predict(testDataX)
         precision = metrics.precision_score(testDatay, yhat)
         recall    = metrics.recall_score(testDatay, yhat)
         f1score   = metrics.f1_score(testDatay, yhat)

         ## PRECISION - RECALL CURVE

         # predict probabilities AND keep probabilities for the positive outcome only
         KKK = RF.predict_proba(testDataX)
         KKK = KKK[:, 1]

         # get precision and recall probabilities
         rf_precision , rf_recall , _ = metrics.precision_recall_curve(testDatay, KKK)
```

```
        # summarize scores
        accuracy  = metrics.accuracy_score(testDatay, yhat)
        mcc       = metrics.matthews_corrcoef(testDatay, yhat)
        error     = metrics.zero_one_loss(testDatay, yhat)

        auc       = metrics.auc(rf_recall, rf_precision)

        recall    = metrics.recall_score(testDatay, yhat)
        precision = metrics.precision_score(testDatay, yhat)
        f1score   = metrics.f1_score(testDatay, yhat)

        print()
        print(pd.DataFrame(metrics.confusion_matrix(testDatay, yhat), index=['True Neg (-) ',
        print()
        print('RF: \n accuracy=%.3f \n mcc=%.3f \n error=%.3f \n auc=%.3f \n recall=%.3f \n pr


        # define a label to use for the graph
        LABEL_RF = 'RF  (AUC: '  +  str(np.round(auc, decimals=3))  + ')'

        # plot the precision-recall curves
        no_skill = len(testDatay[testDatay==1]) / len(testDatay)

        thisfigure = pyplot.figure(figsize=[6, 5])
        pyplot.plot([0, 1], [no_skill, no_skill], linestyle='--', label='No Skill', marker='*
        pyplot.plot(rf_recall, rf_precision, marker='.', label=LABEL_RF)

        # axis labels
        pyplot.title('RF Precision-Recall Malaria Data')
        pyplot.xlabel('Recall')
        pyplot.ylabel('Precision')
        pyplot.legend()
        pyplot.show()
        thisfigure.savefig(f'{FIGPATH}RF_MAL_PRCurve.png')
```

```
               pred (-)  pred (+)
True Neg (-)         94       505
True Pos (+)        134      9407

RF:
 accuracy=0.937
 mcc=0.227
 error=0.063
 auc=0.983
 recall=0.986
 precision=0.949
 f1score=0.967
```



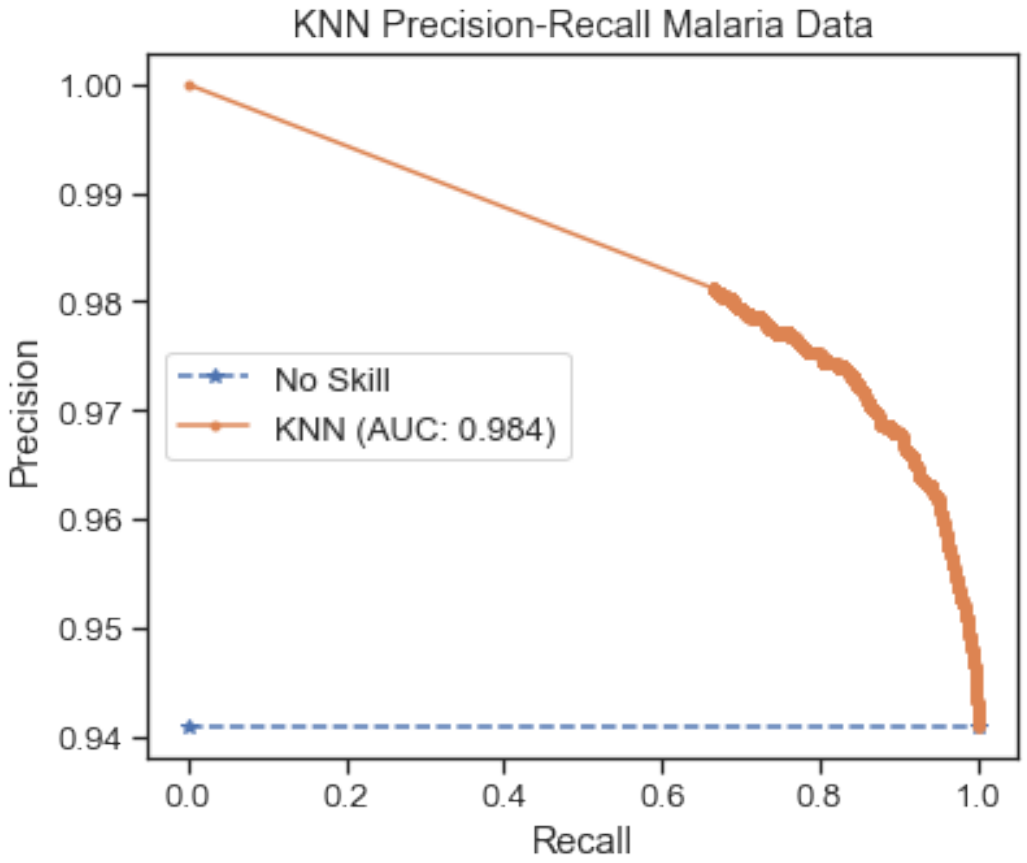
K-Neighbors

```
In [32]: KNN = KNeighborsClassifier(algorithm=algorithm, n_neighbors=n_neighbors, weights=weig
         yhat = KNN.predict(testDataX)

         accuracy  = metrics.accuracy_score(testDatay, yhat)
         precision = metrics.precision_score(testDatay, yhat)
         recall    = metrics.recall_score(testDatay, yhat)
         f1score   = metrics.f1_score(testDatay, yhat)

         ## PRECISION - RECALL CURVE

         # predict probabilities AND keep probabilities for the positive outcome only
         LLL = KNN.predict_proba(testDataX)
         LLL = LLL[:, 1]

         # get precision and recall probabilities
         knn_precision , knn_recall , _ = metrics.precision_recall_curve(testDatay, LLL)
```

```
        # summarize scores
        accuracy  = metrics.accuracy_score(testDatay, yhat)
        mcc       = metrics.matthews_corrcoef(testDatay, yhat)
        error     = metrics.zero_one_loss(testDatay, yhat)

        auc       = metrics.auc(knn_recall, knn_precision)

        recall    = metrics.recall_score(testDatay, yhat)
        precision = metrics.precision_score(testDatay, yhat)
        f1score   = metrics.f1_score(testDatay, yhat)

        print()
        print(pd.DataFrame(metrics.confusion_matrix(testDatay, yhat), index=['True Neg (-) ',
        print()
        print('KNN: \n accuracy=%.3f \n mcc=%.3f \n error=%.3f \n auc=%.3f \n recall=%.3f \n p


        # define a label to use for the graph
        LABEL_KNN = 'KNN (AUC: '  +  str(np.round(auc, decimals=3))  + ')'

        # plot the precision-recall curves
        no_skill = len(testDatay[testDatay==1]) / len(testDatay)

        thisfigure = pyplot.figure(figsize=[6, 5])
        pyplot.plot([0, 1], [no_skill, no_skill], linestyle='--', label='No Skill', marker='*
        pyplot.plot(knn_recall, knn_precision, marker='.', label=LABEL_KNN)

        # axis labels
        pyplot.title('KNN Precision-Recall Malaria Data')
        pyplot.xlabel('Recall')
        pyplot.ylabel('Precision')
        pyplot.legend(loc='center left')
        pyplot.show()
        thisfigure.savefig(f'{FIGPATH}KNN_MAL_PRCurve.png')
```

```
                pred (-)  pred (+)
True Neg (-)          81       518
True Pos (+)          80      9461

KNN:
 accuracy=0.941
 mcc=0.239
 error=0.059
 auc=0.984
 recall=0.992
```

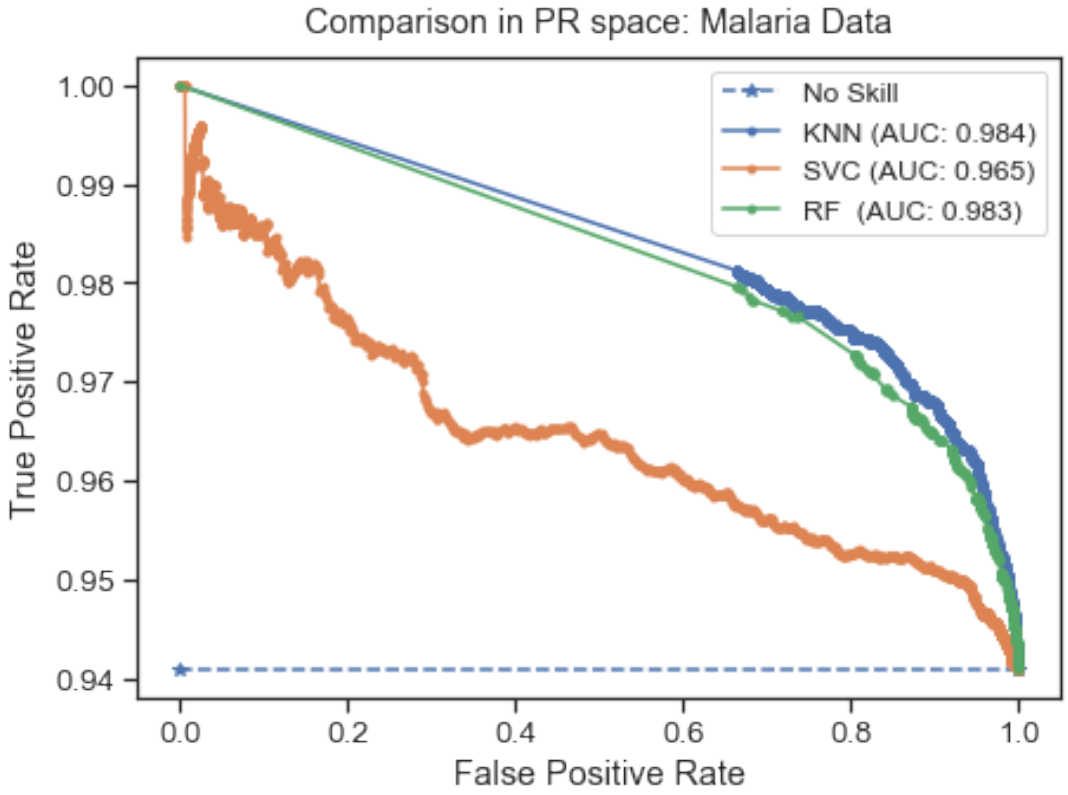
```
precision=0.948
f1score=0.969
```



Combined balanced models - Weighted Datasets

```
In [33]: # plot the roc curve for the model
         thisfigure = pyplot.figure(figsize=[7, 5])

         pyplot.plot([0, 1], [no_skill, no_skill], linestyle='--', label='No Skill', marker='*
         pyplot.plot(knn_recall, knn_precision, marker='.', label=LABEL_KNN, color='C0')  # mo
         pyplot.plot(sv_recall, sv_precision, marker='.', label=LABEL_SVC, color='C1')  # mode
         pyplot.plot(rf_recall, rf_precision, marker='.', label=LABEL_RF, color='C2')     # mo

         # axis labels
         pyplot.title('Comparison in PR space: Malaria Data', pad=10)
         pyplot.xlabel('False Positive Rate')
         pyplot.ylabel('True Positive Rate')
         pyplot.legend(loc='best', fontsize='large')
```

```
pyplot.show()
thisfigure.savefig(f'{FIGPATH}Combined_Models_weighted_Data_Malaria.png')
```



Novel_MalariaCovid19

April 30, 2021

1 Novel/Unary Classifiers

1.0.1 Malaria and COVID-19 data

In [1]:
```
import random
import pandas as pd
import numpy as np
from collections import Counter

from sklearn.svm import OneClassSVM
from sklearn.ensemble import IsolationForest
from sklearn.neighbors import LocalOutlierFactor
from sklearn import metrics

import matplotlib.pyplot as pyplot
import seaborn as sns; sns.set(font_scale=1.2, style="ticks", color_codes=True)
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

Constants

In [2]:
```
PATH       = 'D:/THE UNIVERSITY OF WITWATERSRAND/16.2.JupyterWorks/msc_report/data/'
FIGPATH    = "D:/THE UNIVERSITY OF WITWATERSRAND/16.REPORT/Figures/"

TRAIN_DATA1 = 'malaria_train_valid'
TEST_DATA1  = 'malaria_test'

TRAIN_DATA2 = 'covid_train_valid'
TEST_DATA2  = 'covid_test'

SEED       = 123
```

2 MALARIA

Training Data

```
In [3]: datain = pd.read_csv(f"{PATH}{TRAIN_DATA1}.csv")
        print(f'SHAPE: {datain.shape}')
        print(f'Class distributions: {Counter(datain.target)}')


        # temprarily alter labels
        datain['target'].replace({1:'p', 0:'n'}, inplace = True)
        print(datain['target'].value_counts())
        # make majority 0 and minority 1
        datain['target'].replace({'p':0, 'n':1}, inplace = True)
        print()
        print(datain['target'].value_counts())



        # split predictors and target
        Xtraining = datain[[j for j in datain.columns if j != 'target']]
        ytraining = datain.iloc[:, 0]
```

```
SHAPE: (30417, 29)
Class distributions: Counter({1: 28621, 0: 1796})
p    28621
n     1796
Name: target, dtype: int64

0    28621
1     1796
Name: target, dtype: int64
```

Test Data

```
In [4]: datain = pd.read_csv(f"{PATH}{TEST_DATA1}.csv")
        print(f'SHAPE: {datain.shape}')
        datain.sample(2)
        print(f'Class distributions : {Counter(datain.target)}')


        # temprarily alter labels
        datain['target'].replace({1:'p', 0:'n'}, inplace = True)
        print(datain['target'].value_counts())
        # make majority 0 and minority 1
        datain['target'].replace({'p':0, 'n':1}, inplace = True)
        print()
        print(datain['target'].value_counts())


        Xtesting = datain[[j for j in datain.columns if j != 'target']]
        ytesting = datain.iloc[:, 0]
```

```
SHAPE: (10140, 29)
Class distributions : Counter({1: 9541, 0: 599})
p    9541
n     599
Name: target, dtype: int64

0    9541
1     599
Name: target, dtype: int64
```

Find the best parameters using parameter grid

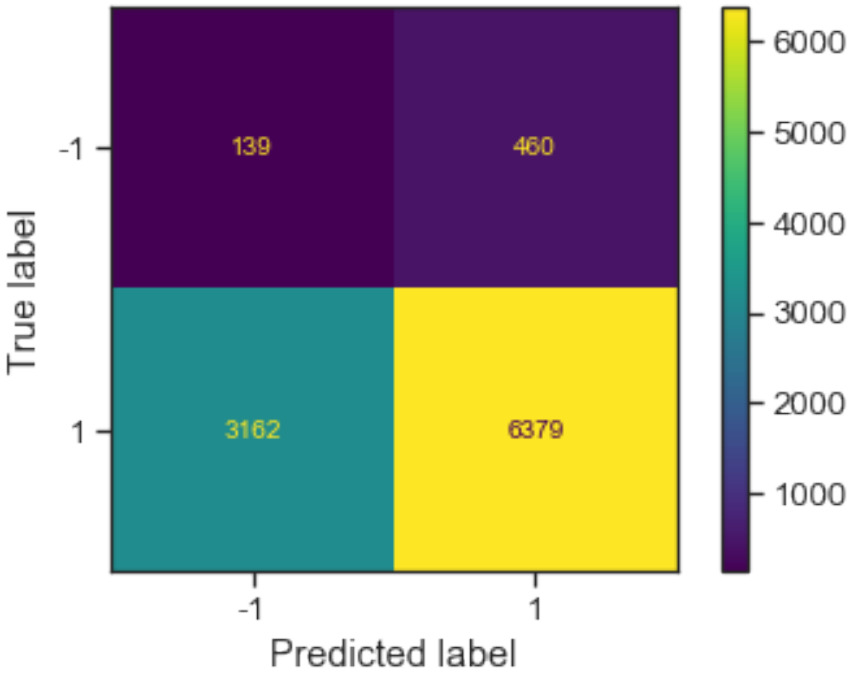
```
In [5]: def outlierPredict(clf, Xtrain, ytrain, Xtest, ytest, s = 3):

            # fit on majority class
            X = Xtrain[ytrain==0]
            clf.fit(X)

            # detect outliers in the test set
            yhat = clf.predict(Xtest)

            # mark inliers 1, outliers -1
            ytest[ytest == 1] = -1
            ytest[ytest == 0] = 1

            # calculate score & Matrix
            print(pd.DataFrame(metrics.confusion_matrix(ytest, yhat),
                               columns = ['pred (-)', 'pred (+)'],
                               index=['True Neg (-) ', 'True Pos (+)'])
                 )

            f1score = np.round(metrics.f1_score(ytest, yhat, pos_label = -1), s)
            precision = np.round(metrics.precision_score(ytest, yhat, pos_label = -1), s)
            specificity    = np.round(metrics.recall_score(ytest, yhat, pos_label = -1), s)

            accuracy  = np.round(metrics.accuracy_score(ytest, yhat), s)
            mcc       = np.round(metrics.matthews_corrcoef(ytest, yhat), s)

            print()
            print(np.char.center('AVERAGE SCORES', 35, '-'))
            print(f'mcc: {mcc}')
            print(f'specificity: {specificity}')
        #     print(f'Precision: {precision}')
            print(f'F1-score: {f1score}')

            cm = metrics.confusion_matrix(ytest, yhat,  labels=[-1, 1])
            disp = metrics.ConfusionMatrixDisplay(confusion_matrix = cm, display_labels = [-1,
```

```
            disp.plot()

            return  None
```

2.1 OC Support Vector Machine

```
In [6]: clf = OneClassSVM(gamma = 'scale', nu = 0.06)
        outlierPredict(clf,
                       Xtrain = Xtraining.copy(deep=True),
                       ytrain = ytraining.copy(deep=True),
                       Xtest  = Xtesting.copy(deep=True),
                       ytest  = ytesting.copy(deep=True)
                      )
```

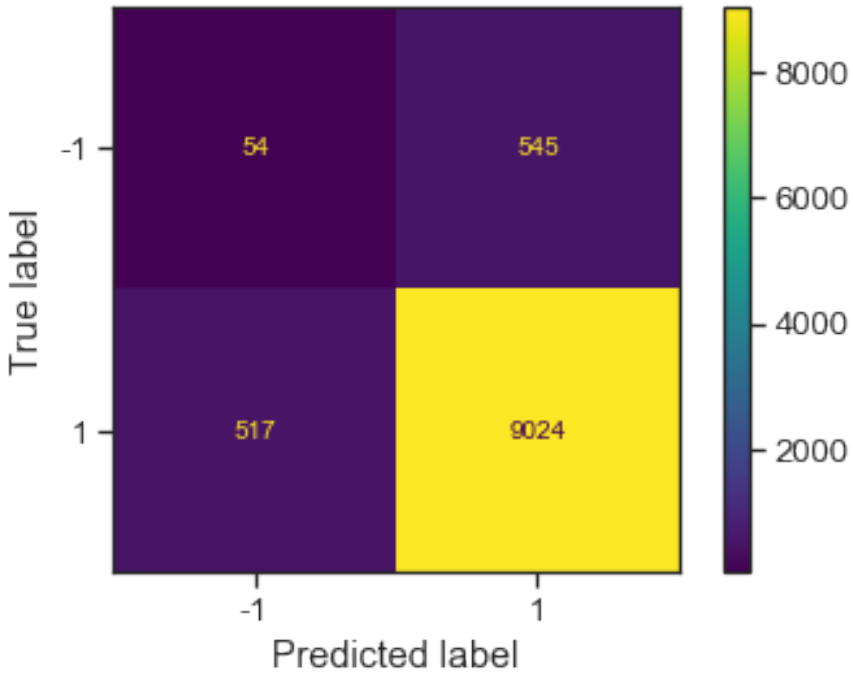
```
               pred (-)  pred (+)
True Neg (-)        139       460
True Pos (+)       3162      6379

----------AVERAGE SCORES----------
mcc: -0.05
specificity: 0.232
F1-score: 0.071
```



2.2 Isolation Forest

```
In [7]: clf = IsolationForest(contamination = 0.06, random_state=SEED)
        outlierPredict(clf,
                       Xtrain = Xtraining.copy(deep=True),
                       ytrain = ytraining.copy(deep=True),
                       Xtest  = Xtesting.copy(deep=True),
                       ytest  = ytesting.copy(deep=True)
                      )
```

```
               pred (-)  pred (+)
True Neg (-)         54       545
True Pos (+)        517      9024

----------AVERAGE SCORES----------
mcc: 0.037
specificity: 0.09
F1-score: 0.092
```




```
In [8]: == CODE BELOW WAS NOT RUN ==
```

```
          File "<ipython-input-8-795dc8a29269>", line 1
        ==
```

```
      ^
SyntaxError: invalid syntax
```

2.3 Local Outlier Factor

In [ ]:
```
# make a prediction with a lof model
```

```
def lof_predict(clf, Xtrain, Xtest ):
    # create one large dataset
    composite = np.vstack((Xtrain, Xtest))
    # make prediction on composite dataset
    yhat = clf.fit_predict(composite)
    # return just the predictions on the test set
    return yhat[len(Xtrain):]

# define outlier detection model
clf = LocalOutlierFactor(contamination=0.06)


# get examples for just the majority class
Xtrain = Xtrain[ytrain==0]


# detect outliers in the test set
yhat = lof_predict(clf, Xtrain, Xtest)


# mark inliers 1, outliers -1
ytest[ytest == 1] = -1
ytest[ytest == 0] = 1

print(pd.DataFrame(metrics.confusion_matrix(ytest, yhat),
                   columns = ['pred (-)', 'pred (+)'],
                   index=['True Neg (-) ', 'True Pos (+)'])
     )


f1score = np.round(metrics.f1_score(ytest, yhat, pos_label = -1), 3)
precision = np.round(metrics.precision_score(ytest, yhat, pos_label = -1), 3)
recall    = np.round(metrics.recall_score(ytest, yhat, pos_label = -1), 3)

print()
```

```
print(np.char.center('AVERAGE SCORES', 35, '-'))
print(f'F1-score:  {f1score}')
print(f'Precision: {precision}')
print(f'Recall:    {recall}')

cm = metrics.confusion_matrix(ytest, yhat,  labels=[-1, 1])
disp = metrics.ConfusionMatrixDisplay(confusion_matrix = cm, display_labels = [-1,1])
disp.plot()
```