TÉCNICO LISBOA

AutoTCGA

An automatic machine learning tool for prediction of immune-evasion mechanisms in TCGA cancer samples

Andreia Filipa Ferreira Jorge Rogério

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor: Prof. Cláudia Martins Antunes

Examination Committee

Chairperson: Prof. Daniel Jorge Viegas Gonçalves

Supervisor: Prof. Claudia Martins Antunes

Member of the Committee: Prof. Rui Miguel Carrasqueiro Henriques

September 2019

ACKNOWLEDGMENTS

I would like to offer my special thanks to the Cancer Bioinformatics group, at King's College London, for their important training in bioinformatics and introduction to the field of Cancer Biology.

ABSTRACT

The incidence of cancer in society has become a growing concern and has encouraged the development of new treatments, like immunotherapy, which boosts the body's natural defences to fight cancer. Unfortunately, cancer cells have developed strategies to evade patients' immune system, called immune-evasion mechanisms. To tackle this problem, a machine learning pipeline was developed, AutoTCGA, which applies classification techniques to The Cancer Genome Atlas data and considers 2 immune-evasion mechanisms for prediction: depletion of neoantigens (the Cancer Antigenome, "CA") and enrichment of Tregs (the immunosuppressors, "IS").

The Decision Tree algorithm achieved cross-validation accuracy scores of 81% and AUC-ROC of 88%, for "IS" binary classification, and accuracy scores of 73% and AUC-ROC of 78% for "CA" binary classification, across the PanCancer population. A feature importance study is available using the tree-based model, revealing the characteristics of the patients with detected immune-evasion events.

These conclusions promote personalized medicine and the application of immunotherapy when suitable. This tool brings the possibility of testing well-defined hypothesis and answer open questions in the cancer biology field, for any user with or without machine learning expertise.

KEYWORDS

Automatic Machine Learning, Feature Selection, Immunotherapy, Immune-evasion, Cancer

INDEX

TABLES AND FIGURES INDEX

LIST OF ABBREVIATIONS

BRCA: BReast CAncer
CA: Cancer Antigenome
DNA: DeoxyriboNucleic Acid
ETL: Extraction, Transformation and Loading
GDC: Genomic Data Commons
IS: ImmunoSuppressors
LOH: Loss Of Heterozigozity
LUAD: LUng ADenocarcinoma
OCLR: One-Class Logistic Regression
TCGA: The Cancer Genome Atlas
TNBC: Triple Negative Breast Cancer

1 Introduction

In the past twenty years we have seen a revolution in genome sequencing with large increases in speed and efficiency coupled with massive reductions in cost, creating the "omics" field in biology [36]. More than 10,000 tumours have now been analysed and stored at The Cancer Genome Atlas (TCGA) [30] with the purpose of increasing the understanding of the genetic basis of this disease, through data analysis and visualization. This data base is widely used in many different cancer sub-fields as it includes thousands of patients, across 33 cancer types and a wide variety of features from clinical information to molecular characterization.

Even though TCGA provides the resources needed to advance cancer research at a multidisciplinary level, extracting the biological knowledge has become a challenge, not just due to the massive size of the data but also due to the diversity of data types. Many computational tools have emerged to face this problem, usually through data visualization techniques and by building correlations between variables, creating distributions across cancer types and Kaplan-Meyer plots [37]. However, there is a shortage of powerful automated data mining tools designed to analyse these data with the ability to find connections between all dimensions of attributes, going beyond a hypothesis testing tool and providing insights on novel sub-types of cancer.

Machine learning algorithms have proven to be strong allies in effective analytical workflows that guarantee reproducibility, statistical significance and result provenance, which are essential characteristics in software tools handling biological data. However, the most powerful algorithms are usually "black-boxes", hard to visualize and therefore interpret, and may even require more advanced programming skills which some life-science researchers may be lacking. Decision Trees, on the other hand, are algorithms proven to be very powerful handling diverse data types, including distinct numerical ranges, and leverage on a tree-based structure which can be visualized and interpreted by a non-expert in machine learning. Beyond classification, these algorithms can be used to model a population, testing a single feature in each node until the last nodes (leaf nodes) have the most homogeneous group of samples possible, considering the values of a chosen feature (class) as the criteria of homogeneity. Random Forest and XGBoost are ensemble algorithms that use Decision Trees as their basic unit (estimators), therefore gaining in performance but losing in interpretability [38].

The aim of this study is to build a machine learning pipeline able to assemble data from TCGA, train tree-based classifiers to predict cancer features in an automated fashion and export the results through a web app that allows the visualization of the models and respective interpretation by cancer biology experts who may not be familiar with machine learning techniques.

As motivation for the development of the tool, a biologically relevant feature was chosen to be predicted by the tree-based classifiers: the predominance of an immune-evasion event. These are non-incompatible mechanisms that have been observed in cancer cells able to evade the patients' immune system. The purpose of immunotherapy is to boost the body's natural defences to fight cancer, therefore disabling its evasion strategies [39]. It is of upmost importance to study these events and conclude which cancer features are related to each evasion mechanism, because it allows a tailored treatment according to the patient's characteristics – personalised medicine – and the application of

immunotherapy when suitable. However, the interactions between the tumour and the patient's immune system are not yet well understood and depend on many biological characteristics, which is a problem that a machine learning algorithm can help solving. In particular, we can use classification techniques considering the immune-evasion mechanisms detected in the patient's tumour as the classes to be predicted, which is ideally a multilabel classification problem, but each immune-event mechanism can be described as a binary classification problem.

At the moment, there are bioinformatic tools that analyse immune features of samples from cancer patients. None is available as a web application service nor as a combined tool able to analyse the predominance of each mechanism. The tools available with web-based interface are only data repertoires [2,3]. Most of the analyses published are comparisons between biological features and immune-evasion mechanisms, supported by p-value. Some more broad analyses use hierarchical clustering to group several samples [2], Cluster-Of-Cluster-Assignments (COCA) [3] to combine the results of several clustering approaches, or even model-based clustering [4] to understand the patterns between several features, and a smaller number uses classification techniques [2] to predict the cytolytic activity of tumours. This tool will allow the analysis of all available features and understanding of their relationship with the mechanisms, instead of studying each feature individually (as the statistic tests do), or independently of the events (as the clustering techniques do).

In conclusion, in this study the aim of the tool is to answer the question: **Can cancer TCGA genomic features predict immune evasion mechanisms?**

2 Background

To understand the problem underlying this thesis and successfully build a solution, it is necessary to have the basic knowledge of both the biological aspects of the object of study and the computational approaches needed, including those that were already used, effectively or not, and those that were never applied but show potential.

2.1 Biology background

Cancer is often thought of as a scary disease with no cure nor treatment. In reality, there are multiple types of cancer, many of which can today be effectively treated to eliminate, reduce or slow the impact of the disease, such as immunotherapy.

What is cancer? The human body is composed of many millions of cells, each a self-contained living unit. Normally, each cell coordinates with the others to compose tissues and organs. One way that this coordination occurs is reflected on cells reproduction. Normal cells grow and divide during youth and then stop growing and dividing. Thereafter, they only reproduce to replace defective or dying cells.

Cancer occurs when the cellular reproduction process goes out of control. In other words, cancer is a disease characterised by uncontrolled, uncoordinated and undesirable cell division. Unlike normal cells, cancer cells continue to grow and divide for their whole lives, replicating into more and more harmful cells.

The abnormal growth and division observed in cancer cells is caused by damage in these cells' DNA (genetic material inside cells that determines cellular characteristics and functioning). There are a variety of ways that cellular DNA can become damaged and defective. For example, environmental factors (such as exposure to tobacco smoke) can initiate a chain of events that results in cellular DNA defects that lead to cancer. Alternatively, defective DNA can be inherited from the parents.

As cancer cells divide and replicate, they often form into a clump of cancer cells known as a *tumour*. Tumours cause many of the symptoms of cancer by pressuring, crushing and destroying surrounding non-cancerous cells and tissues.

It is important to note that cancer is not a uniform illness, but rather has many forms and different symptoms, mostly dependent on the organ where the tumour first appeared, the *primary tumour*. The organ where the primary tumour grows is called *target organ* or *primary organ*. The target organ is what distinguishes cancer types and according to the National Institute of Health (NIH) there can be more than 200 cancer types, which can be grouped in 33 types according to The Cancer Genome Atlas (TCGA). Each cancer type has different characteristics, again dependent on the organ itself (for instance, the aggressiveness of the cancer: how fast it grows, whether it spreads to other organs creating metastasis, how it responds to each treatment). [5]

However, some studies have shown that cancers that start in different tissues actually share features at the molecular level, whereas cancers that originate from the same tissue can have very different genomic profiles (genomic features that characterize it, explained below).[6]

We can further group the 33 cancer types into 7 types according to target tissue: adenocarcinoma, squamous cell carcinoma, other carcinoma, sarcoma, leukaemia, lymphoma and other. Carcinomas account for 80 to 90 percent of

all cancer cases. Examples of carcinomas include cancers of the breast, prostate, lung, intestine, skin, pancreas, liver, kidneys, and bladder.

Who has cancer? A physician who suspects a patient may have cancer will perform a series of tests to diagnose it (or rule-out). Commonly, doctors will collect a sample of tissue or fluid to be analysed in the laboratory, a *biopsy*, which is usually enough to make a diagnosis. The technician also extracts the genetic material of all the cells present in the sample collected. The genetic material includes DNA and RNA.

Everybody knows about DNA, that it holds all the information that makes up the entire living organism in the form of genes, and it sits comfortably in the nucleus of the cell. DNA does not float around randomly through the nucleus, it is packed into chromosomes. Almost all animals are diploid, which means that they have 2 copies of each chromosome. So, of the 46 human chromosomes, 23 originate from the mother and 23 originate from the father. RNA is the lesser-known of the two, and the hard worker as well. RNA is a copy, or a transcription, of DNA, and is the one that goes out to do work throughout the cell, in order to achieve the goal of producing proteins.

The way that DNA encodes the instructions for proteins is through a set of four molecules called bases, each of which represents a letter of the genetic code (A = adenine, C = cytosine, G = guanine, and T = thymine). The RNA molecules are very much alike, the only difference being that in RNA synthesis no thymine (T) is used but uracil (U) is used instead. If there is a missing base, a switch base or even an added base, there is a mutation in the DNA code, that will be passed on to the RNA code and the protein may not be successfully produced (Fig. 1).

While there is only 1 copy of DNA in each cell, there can be many copies of RNA, which is useful when a lot of the same protein is needed. This means that the study of the DNA code provides a record of which proteins can be produced successfully or not, while the study of the RNA code provides a quantitative measure of how much each protein is actually produced, usually called the *genetic expression*.

Using sequencing machines, a geneticist can decipher the sequence of bases that build up both the DNA molecules and the RNA molecules. The DNA code is called the genetic code, or *genome*, of an individual, and is stored in the form of WXS files, while the RNA code is called the *transcriptome* and stored in RNA-seq files.

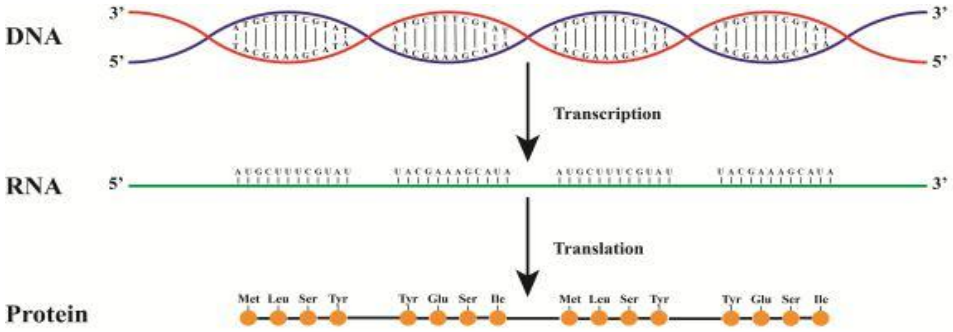


Figure 1: Central dogma of Genetics, from genetic code to proteins (Source: Science Explained)

How to treat cancer? The types of treatment will depend on the type of cancer and how advanced it is. Most people have a combination of treatments, such as surgery with chemotherapy and/or radiation therapy. There is also immunotherapy, targeted therapy, or hormone therapy.

Chemotherapy drugs kill dividing cells, which explains why it causes side effects. It affects healthy body tissues where the cells are constantly growing and dividing, such as the hair. **Radiation** is most commonly used to treat localised cancers as opposed to cancers that have spread throughout the body. **Hormone therapy** is used to treat prostate and breast cancers that use hormones to grow. **Targeted therapy** is the foundation of precision medicine. It is a type of cancer treatment that uses toxic molecules that target the changes in cancer cells that help them grow, divide and spread, by detecting small markers (explained below in chapter 2.1.1). These markers can be identified during the biopsy. **Immunotherapy** is a type of cancer treatment that helps the immune system fight cancer. The immune system includes several cells originated from the bone marrow, from where they migrate to circulate in the blood and in the lymphatic system. Among other approaches, the therapy works by giving the patient drugs called "Checkpoint inhibitors", which help the immune system respond more strongly to a tumour. These drugs work by releasing "brakes" that would keep T cells (a type of white blood cell and part of the immune system) from killing cancer cells. These drugs do not target the tumour directly. Instead, they interfere with the ability of cancer cells to avoid immune system attack – an event called *immune-evasion*.[7]

Immunotherapy is not yet as widely used as the remaining therapies, since different variants of its usage are still being studied in clinical trials. Nonetheless, its recent application has shown very good results and shows to be promising against many highly-aggressive cancer types, for which conventional therapies have low success rates [5]. For the immunotherapy to work, it's not enough to boost the activity of the immune-cells, they also need to be able to detect the tumour-cells, which doesn't happen when immune-evasion events occur. The origin of these events can be classified in two categories: **tumour-intrinsic** and **tumour-extrinsic**.[2] While the cancer antigenome load (CA), HLA mutation load (HM) and HLA absence (HA) events are intrinsic factors, the presence of immunosuppressive cells (IS) is extrinsic.

In the last decade, several studies have been done concerning the prevalence of each individual immune-evasion mechanism for each cancer type, and the relation of each mechanism with specific genes' mutations and genes' expression in cancer samples (features belonging to the genomic profile of the tumours). In a study including 20 solid cancers with more than 8000 tumour samples, it was observed that there are several mutated genes in the tumour that are related to the tumour escape mechanisms [2]. This is an indication that tumour genomic features can be used to explain the immune-evasion events. The increase in studies was motivated by the development of bioinformatic software that has been built to help extract specific genetic features from tumour samples.

2.1.1 Immune-evasion events

In the human immune system, there are white-blood cells (leucocytes) specialised in the detection of invader cells (virus) or cancer cells and the activation of an immune response, which are called *lymphocytes T-cells*. They work by detecting antigens at the surface of the cells, called HLA. By definition*, antigens* are substances to which the antibodies or the T lymphocyte antigen receptors specifically bind. The group of antigens of an individual is called *antigenome*.

HLA is the Human Leucocyte Antigen, the human version of the major histocompatibility complex (MHC), which consists in more than 200 genes located close together on chromosome 6. Genes in this complex are categorised into three basic groups: class I, class II, and class III. Humans have three main MHC class I genes, known as HLA-A, HLA-B, and HLA-C. The proteins produced from these genes are present on the surface of the cells. Once there, these proteins are bound to protein fragments called *epitopes* (peptides simplistically called "markers"), that have been exported from within the cell. MHC class I proteins display these peptides to the immune system. If the immune system recognises the peptides as foreign (such as viral or bacterial peptides), it responds by triggering the infected cell to self-destruct.[7]

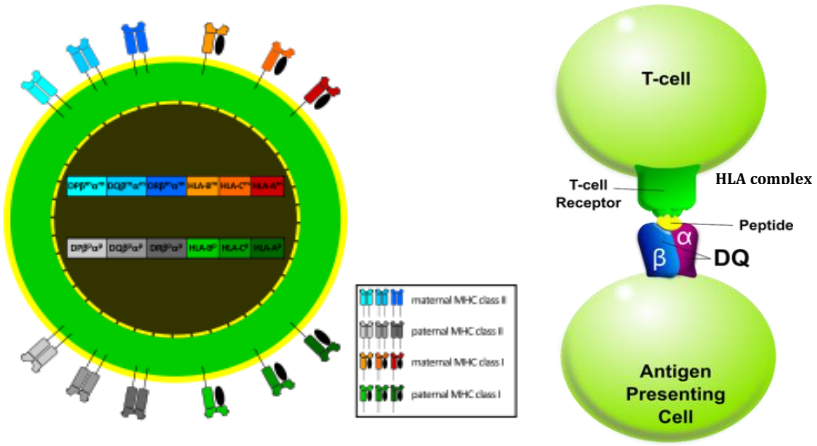


Figure 2: HLA alleles (left) and interaction between HLA, epitope (peptide) and T-cell (Source: Wikipedia)

Particularly, it is the CD4+ helper T cells (Th1) that binds HLA molecules of class II, and CD8+ cytotoxic T cells that binds HLA molecules class I. Both these cells inhibit the cancer development by producing of interferon (IFN)-gamma and cytotoxins (when no immune-evasion events happen). When leucocytes invade the cancer tissue, they are called *Tumour-Infiltrating Lymphocytes* (TILs). They can be T cells, B cells, natural killer cells, macrophages, neutrophils, dendritic cells, mast cells, eosinophils, basophils, etc., and exist in variable proportions according to the tumour type and cancer stage. These variables are part of the **immune features**.

The **cancer antigenome** is the group of antigens (surface molecules) of the cancer cells. It includes two types of antigens: the cancer-germline antigens (CGAs) and the neoantigens. The CGA's are molecules that exist in healthy tissue but have aberrant expression in tumour cells, whereas neoantigens only arise from somatic mutations or gene rearrangements that occur in tumour cells. The CGA expression levels (or burden) are derived from RNA-seq and can be quantified using metagenes (non-overlapping sets of genes that represent specific immune cell subpopulations and are not expressed in normal tissue) and Gene Set Enrichment Analysis (GSEA), that measures the overexpression level of a particular gene by comparing its rank with all the other genes' ranks. If they are being expressed then the immune system can detect the cancer cells, otherwise there is an immune-evasion event. [2]

During the development of a tumour certain mutations accumulate, which do not exist in the remaining tissues, and they are not transmitted from generation to generation. These are known as *somatic mutations*. The tumour genome

includes mutations that facilitate or are essential to the development of the tumour itself (known as tumour *drivers*) and others that have accumulated during the growth of the tumour (known as *passengers*). Driver mutations have a tendency to occur in protein-coding regions of genes and within important functional domains of the protein.[8]

Because driver mutations are by definition those resulting in cancer initiation and/or progression, they are seen as the weakness of tumours, sought after as targets for drugs, and used in making therapeutic decisions. There are generally two methods to classify a mutation as a driver or passenger: 1) by frequency (driver mutations should be mutated in a greater proportion of cancer samples than would be expected from the background mutation rate); 2) by prediction of functional impact (either via in-silico algorithms or cell/model-based assays). [9]

A neoantigen arises from cancer-specific mutations (drivers or passengers) that produce cancer-specific proteins (*neo-epitopes*) that may become able to bind the antigens (HLA) at the cell surface and become a target to immune cells. To detect a neoantigen, it is necessary to first detect all mutated peptides (that are not mutated in healthy cells) and test whether or not they bind to the antigens at the surface of the cancer cells. This is possible through a pipeline of several bioinformatic tools, whose results are available online for several cancer samples.

It may still happen that the antigens are not working properly due to mutations and, therefore, the epitopes are not exposed to the surface of the cell. Because the genes that code for the HLA molecules are highly polymorphic (there are multiple *alleles* possible for each gene), it is necessary to use a specifically designed bioinformatic tool to determine which HLA genes are in fact mutated.

There are several tools available to describe which HLA molecules each patient produces, i.e. *genotyping* the HLA. Polysolver is one that also reports mutations in those genes. In the article where Polysolver is first described[10], the authors observe differences in frequency, localisation and types of HLA mutations across cancer types. Furthermore, by analysing the RNA-seq of 4,512 samples across 11 cancer types, a set of genes were found to be significantly enriched in all the samples with mutated HLA genes. These results reflect the high variety of genomic features in tumours.

There are also other proteins responsible for the exposure of the antigens, such as membrane proteins LMP(2) and LMP7 and transporters associated with antigen processing (TAP). Thus, mutations in these proteins can also lead to immune-evasion[1]. Together, the mutations load in HLA and membrane and transporters, make the **HLA mutations load**, which in high values leads to immune-evasion.

Another possibility of downgrading the HLA presence would be the deletion of one or both copies of the HLA genes (there is one from each parent, see Fig.2), which would diminish the number of proteins synthesised by the cell. LOHHLA[11] is a software developed specifically to detect the loss of heterozygosity (LOH) event in HLA molecules, that in practice leads to **HLA absence**, and therefore immune-evasion. It was applied to lung cancer samples and the results were crossed with other features, mainly, gene expression and immune-cells burden. [11]

Simultaneously, there are also cells that induce tumour growth. **Immunosuppression** is mediated by CD4+CD25+FoxP3+ regulatory T cells (Tregs), that naturally have other functions in the body. These cells are drawn to the tumour microenvironment by the cell-mediated chemokine production, but also other molecules like cytokines

[1] [12]. The presence of all these mediators, quantified by their protein expression, is also part of the tumour **genomic features**.

Other immunosuppressive cells are Myeloid cells, particularly myeloid-derived suppressor cells (MDSCs), modulated dendritic cells (DC) and alternatively-activated M1 and M2 macrophages, that create an inflammatory microenvironment.[1]

These two cells (Tregs and MDSCs) are the extrinsic immunosuppressors or suppressor cells. Their high expression values may lead to immune-evasion by inhibition of the immune-system.

Other molecules in the human body also contribute to tumour progression through apoptosis of tumour-infiltrating lymphocytes (TILs)[1]. They represent intrinsic immunosuppressors, or immunoinhibitors, that should be considered **immune features** of the patient.

The presence of immunosuppressors can be quantified using the GSEA approach explained above[2]. A high immunosuppressive score is related with bad prognosis (low Overall Survival) in patients.[2]

Intuitively, the immune-evasion events can now start being referred to as the classes of a classification problem.

2.1.2 Tumour and immune features

In the TCGA database there are 10,000 tumour samples from 33 types of cancer. This database has been analysed several times, the most interesting approaches being:

A. Clustering of cluster assignments technique, which combines the results of 5 clustering approaches, each using a different set of features: RNA-seq, aneuploidy/genes copy-number, DNA hypermethylation, microRNA and Proteins. This technique resulted in 28 different clusters. [3]

B. Selection of 4691 genes from RNA-seq analysis, grouped into 160 signatures that were clustered into 5 representative traits. These were the features used to create 6 clusters of the data, using model-based clustering. [4]

C. Application of Random Forest to several tumour-intrinsic and extrinsic features, using cytolytic production as class. The discriminant features were used to build an immunophenoscore. This study used 8,000 samples of 20 cancer types.[2]

In the approach A the authors observed that the **cancer type** was a predominant feature but not determinant in all clusters. This approach also revealed the relative contributions of each feature type: 47% aneuploidy + genes copy-number, 42% RNA-seq, 11% DNA methylation. [3] Other distinguishing features found in this study were: the 7 **tissue types**, the **stromal fraction**, **leucocyte content, stemness index** [13], **mutations burden** and mutations signatures (1 to 20).

Stromal fraction is given by 1 minus purity, which reflects the proportion of tumour cells in the sample. Stemness, defined as the potential for self-renewal and differentiation from the cell of origin, was originally attributed to normal stem cells that possess the ability to give rise to all cell types in the adult organism. Cancer progression involves gradual loss of a differentiated phenotype and acquisition of progenitor-like, stem-cell-like features. Undifferentiated primary tumours are more likely to result in cancer cell spread to distant organs, causing disease progression and poor prognosis, particularly because metastases are usually resistant to available therapies. [13]

The mutations burden in each sample was obtained using MC3 software in study A. In other study (C) this feature has been categorized into three levels: tumours with high mutational load (upper quartile), intermediate (two intermediate quartiles) and low (low quartile). It was possible to identify a correlation between high mutational load tumours and depletion of immunosuppressors.[2]

Concerning mutations, the neoantigens burden can also be described in terms of mutation origin (driver gene, passenger gene, clonal and subclonal). This can be done using a library of driver genes[14] and estimators for clonality[15]. The origin of the neoantigens of the samples in TCGA was determined by a study where the Pan Software was developed and applied [32]. The distribution was analysed and is 7.6% of driver genes origin, and the remaining in passenger genes, amongst solid tumours. On average 56% of the neoantigens had its origin in clonal genes but ranged between 31% and 71%. The proportions clonal/subclonal also varied with cancer type [2].

In studies B and C, the genomic features used were more specific and included the class IS (Tregs and MDSCs) obtained using the expression of several genes (20 for each in ref [2] and 1 for each in ref [4]).

Some genes have been detected as being **differentially expressed genes** in tumours: chemokines, cytokines, and the "Cytolytic Activity" metagene given by the geometric mean of GZMA and PRF1 expression[10, 11, 12]. The genetic expression can also provide information about the tumour content in several **immune cells burden** (28 different types). This can be done using the GSEA approach, using 782 genes [2].

The intrinsic immunostimulators and intrinsic immunoinhibitors (described previously regarding the class IS) can also be identified in the same manner, represented by 47 genes and 24 genes respectively [2 fig 4H].

Another important expression feature is the HLA molecules expression and other related proteins (membrane and transporters) which are represented by 20 genes [2].

In study C the authors created a web page, The Cancer Immunome Atlas (https://tcia.at/home), that allows the user to access the data integrated and better understand the interactions between the features. However, the user is limited to the variables used, the number of samples, and the analysis available.

Regarding the patients' inherent features, the demographic characteristics as well as the survival are available. All the features publicly available can be found in a diagram (Figure 3) built with the same structure as the original tables. The TCGA database includes the patients' features, the immune system features and most of the tumour genomic features. The stemness index, the neoantigens' origin and the survival can be found in other sources (described in more detail in chapter 3).

2.2 Computational background

Bioinformatics is often used as an umbrella term to describe biological studies that use computer programming as part of their methodology, as well as specific analyses "pipelines" that are repeatedly used, particularly in the field of *genomics* (the study of the structure, function, evolution, mapping, and editing of genomes).

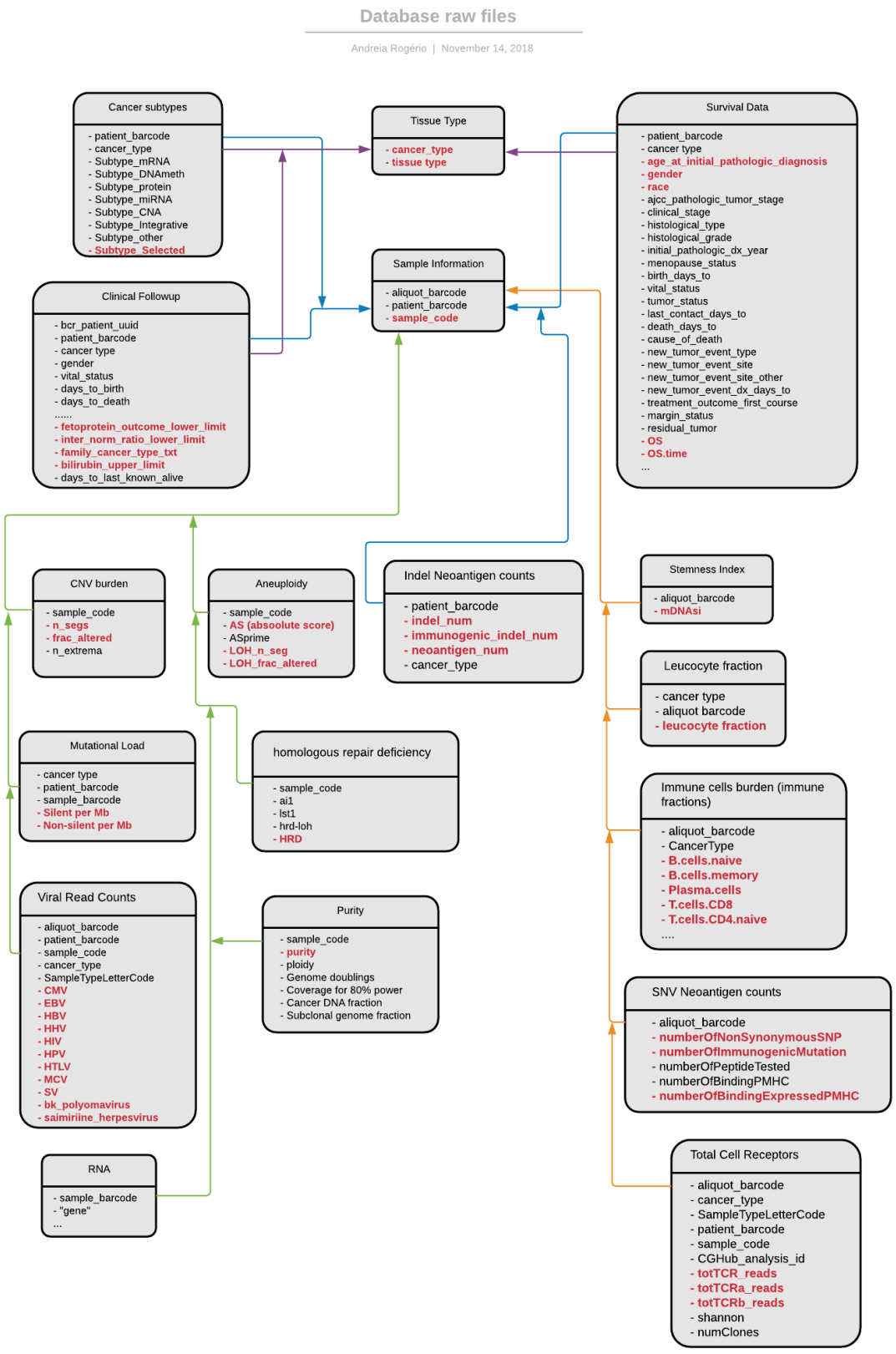


Figure 3: Data base diagram of data available in TCGA project webpage. In red are the features kept. The color of the arrows represents the identifier connecting the tables. Green: sample_code, Orange: aliquot_barcode, Blue: patient_barcode, Purple: cancer_type.

As mentioned, there are many bioinformatic tools available for free that take as input the RNA-seq files and output the expression of the desired genes. There are also tools to analyse the WXS files looking for mutations in genes or missing genes.

2.2.1 Immune-evasion mechanisms detection and genomic features quantification

Below there is an explanation of the necessary tools to generate features that are currently not available. These features belong to the genomic features of the tumour.

HLA Genotype and Mutations: Polysolver takes as input the WXS files from tumour sample and normal/germline sample of a patient. With the germline it computes the alleles of each MHC class I molecule, and with the tumour it computes the mutations observed. The algorithm uses a Bayesian calculation that considers the base qualities of aligned reads, sizes, as well as the ethnicity-dependent prior probabilities of each HLA gene. It recorded 97% of accuracy using HapMap samples.[10]

HLA LOH: LOHHLA software takes as input WXS files from tumour sample and normal/germline sample of a patient and its HLA genotype and calculates the copy number of the class I HLA alleles. It performs better than most genes copy number calculator softwares, as it leverages on the reads that map specifically to an individual germline (normal tissue) alleles rather than the human reference genome. The germline alleles are obtained with Polysolver or other HLA genotyping software. It showed 90% concordance with ASCAT software applied to the genomic segments adjacent to the HLA locus. [11]

RNA-seq/DEGs expression values: R package DESeq2 takes as input the raw RNA-seq read counts for analysis of differential expression. A FDR cutoff of 0.05 was used to determine genes significantly differentially expressed in lung cancer samples [10].

2.2.2 Model generation

In order to test the features that correlate to each class and the features that are correlated with each other, there are several statistical tests that can be performed to assess it, as well as pattern mining algorithms to find specific events (combinations of features' values in specific classes) that often occur.

At the same time the application of machine learning algorithms is highly useful to discover the patterns of features' values that characterise each immune-evasion mechanism. This is possible by training a classification algorithm, with a data set in which the class assigned to each instance is known. The goal of the classifier is to accurately predict the target class for each future instance, applying several if-then conditions to the features' values. Even though in this case the aim is not to apply the predictive power of the classifier, its learning process is of interest.

It is of upmost importance that the model, meaning the conditions applied to each feature, is easily interpretable by the user, which can be accomplished in decision tree-based algorithms (explained below).

It's also important to note that machine learning algorithms rely extremely on the quality of the data, reason why the pre-processing step is very important and includes data cleaning (missing values, impossible values, typos, etc) and techniques to assure the classes are balanced (same proportions).

One-to-one associations

Regarding single features' analysis, a heatmap visualisation is often used as well as hierarchical clustering to study mutational signatures across cancer types[3] or specific genes expression.

Concerning the association of two features' growth (for example CGAs and immune cells [2 fig 3A]), it can be done using Spearman rank correlation (s>0.3) and p-values can be adjusted according to the Benjamin-Hochberg method (p<0.1). Volcano plots are other type of representation of two variables based on GSEA scores of genes with NES>0 [2, fig 2B] .

Pattern Mining

Association rules learning is a method for discovering interesting relations (patterns or rules) between variables in large databases. In this case the genetic features of each sample and its class. It is intended to identify significant rules in databases using measures of interestingness.

The Apriori algorithm was proposed by Agrawal and Srikant in 1994. In this algorithm a set of items (an itemset) is a group of characteristics that may belong to an instance. Apriori uses a "bottom up" approach, where frequent subsets of items are extended one item at a time (a step known as candidate generation) and groups of candidate itemsets are tested against the data, to see if they are significant or not (according to the condition given by the user through the measures of interestingness). The algorithm terminates when no further successful extensions are found to characterise a relevant group of instances.

Eclat stands for Equivalence Class Transformation, and is a depth-first search algorithm based on set intersection. It is defined recursively: the initial call uses all the single items and in each recursive call the algorithm crosses each itemset with all the others to generate new candidates. If the new candidate is frequent (condition given by the user through the measures of interestingness), it is added to the set. Then, recursively, it finds all the frequent itemsets possible.

Automatic Machine Learning

Machine learning remains a hard problem when implementing existing algorithms and models to work well for a new application. It requires creativity, experimentation and persistence. Automatic machine learning (*AutoML*) refers to automation of optimisation of machine learning tasks: data pre-processing, algorithm selection, hyperparameter tuning, iterative modelling and model assessment.

A variant called grid search consists in testing the nodes of a discretisation grid (range given by the user) of all the machine learning tasks parameters. These Monte-Carlo-like algorithms can be a good starting point for low dimensional problems but with high number of parameters they will converge very slowly. Moreover, in the grid search case, if optimal strategies are far from the nodes defined, there is no chance to discover them. Alternatives are metaheuristics and Bayesian approaches.

It is important to notice that AutoML solutions do not perform better than human data scientists, but can help get convenient levels of accuracy with a lower effort and amount of time.

Automatic feature processing

In the context of machine learning, a feature can be described as a characteristic, or a set of characteristics, that explains the occurrence of a phenomenon. When these characteristics are converted into some measurable form, they are called features. Choosing informative, discriminating and independent features is a crucial step to build an effective algorithm. Informative features are those that have a high weight in the model and hold discriminant abilities that no other feature holds.

Dependent on the data type there are different approaches of data cleaning, feature engineering, and model generation (some algorithms work better with certain data types). *Feature analysis* comprises the study of the data type given the possible types: Useless, Nominal, Binary, Ordinal, Ratio, Time, Interval, Image, Video, Audio, Text.

In the useless section are the identifiers and other variables that have no possible relation with the class, either because they are generated randomly, having a high cardinality, or because they have a single unique value. Nominal data is usually one-hot-encoded (aka dummy encoded), while ordinal data is ordinal-encoded, being transformed in values between 0 and the number of unique values minus 1. Binary data is the particular case where only the values 0 and 1 are taken. Numerical data can be classified into ratio or interval, weather the zero value means null amount and it makes sense to subtract and add the measures (ratio) or not (interval). Interval data has equal spaces between the numbers and does not represent a temporal pattern.

In certain cases, it might be interesting categorising the numerical attributes in ranges, either with equal size of values' range in each category, or with equal number of samples in each category. It then becomes an ordinal feature. Finally, the normalisation of the features is performed, usually using the z-score (equation 1).

$$z = \frac{x - \mu}{\sigma} \tag{1}$$

Feature Engineering can simply be defined as the process of creating new features from the existing features in a dataset. Creating new features helps to provide more information to the model about the target variable, and its performance will go up. The manual engineered features are limited both by human creativity and patience: there are only so many features we can think to build and only so much time we have to make them. The promise of automated feature engineering is to surpass these limitations by taking a set of related tables and automatically build hundreds of useful features using code that can be applied across all problems.

We can group the operations of feature creation into two categories: transformations and aggregations. The first includes creating new features out of one or more of the existing attributes, like splitting a date attribute into day, month and year attributes, or transforming an absolute value into percentage. The aggregations are performed across tables and use a one-to-many relationship to group observations and then calculate statistics, such as the average, maximum, and minimum of attributes that belong to an entity, for instance several genes that are expressed by one tumour.

Featuretools[16] is an open source python framework for performing automated feature engineering. There are three major components of the package: Entities, Deep Feature Synthesis (DFS), feature primitives. An Entity can be considered as a representation of a Pandas DataFrame. An EntitySet is a structure that contains multiple dataframes

and relationships between them. Deep Feature Synthesis (DFS) has got nothing to do with deep learning, it is actually a Feature Engineering method and is the backbone of Featuretools. It enables the creation of new features from single or multiple dataframes by applying feature primitives to the Entity-relationships in an EntitySet. These primitives are the often-used methods to generate features manually (e.g. the primitive "mean"). Each of these features is built using simple aggregations and hence is human-interpretable. They can be complex and based on subject expertise because Featuretools allows the user to define custom primitives. In 2015, researchers at MIT presented the Deep Feature Synthesis algorithm and demonstrated its effectiveness in online data science competitions.

Automated feature engineering has solved one problem but created another: too many features. Having too many features can lead to poor model performance, because the less useful features drown out those that are more important – the *curse of dimensionality.*

Feature Selection methods identify and remove unneeded, irrelevant and redundant attributes from data that do not contribute to the accuracy of a predictive model or may in fact decrease its accuracy. It also reduces training time, since less data means the algorithm trains faster. Feature selection is different from dimensionality reduction. Both methods seek to reduce the number of features in the dataset, but a dimensionality reduction method does so by creating new combinations of attributes, whereas feature selection methods include and exclude attributes present in the data without changing them. There are three general classes of feature selection algorithms: filter methods, wrapper methods and embedded methods.

Filter feature selection methods apply a statistical measure or algorithmic criteria to assign a scoring to each feature. The features are ranked by the score and either selected to be kept or removed from the dataset. The methods consider the feature independently or with regard to the dependent variable (target). Some examples include the chi squared test, information gain and correlation coefficient scores. The PCA method, usually used in dimensionality reduction, can also be used.

It is also possible to use the feature's ranks generated by machine learning models, and then prune the features based on that ranking. This approach is good because it is non-parametric.

In the specific case of the decision tree's algorithms (explained in detail below), there are two ranking approaches: **mean decrease impurity** and **mean decrease accuracy**.[25] The first approach measures the Gini impurity, Gini index or Information gain. The second approach works by randomly permuting a certain variable. The intuition behind permutation importance is that if a feature is not useful for predicting an outcome, then altering or permuting its values will not result in a significant reduction in a model's performance. Thus, a reasonable measure for variable importance is the difference in prediction accuracy before and after permuting a variable. [22]

In python library scikit-learn there is a class variable called *feature_importances_* (in all tree-based classifiers) that implements the first approach, measuring all the features' importance by looking at how much the tree nodes which use that feature reduce impurity across all trees in the forest. It computes this score automatically for each feature

after training and scales the results, so that the sum of all is equal to 1. For a forest, the impurity decrease from each feature can be averaged across all classifiers.

There are a few things to keep in mind when using the impurity based ranking in decision trees. Firstly, feature selection based on impurity reduction is biased towards preferring variables with more categories [22]. This won't be a problem if the variables are all from the same type (for instance continuous).

Secondly, when the dataset has two (or more) correlated features, then any of these correlated features can be used as the predictor, with no concrete preference of one over the others. But once one of them is used, the importance of others is significantly reduced since effectively the impurity they can remove is already removed by the first feature. As a consequence, they will have a lower reported importance. This is not an issue when using feature selection to reduce overfitting, since it makes sense to remove features that are mostly duplicated by other features. But when interpreting the data, it can lead to the incorrect conclusion that one of the variables is a strong predictor while the others in the same group are unimportant, while actually they are very close in terms of their relationship with the response variable. The effect of this phenomenon is somewhat reduced thanks to random selection of features at each node creation (the variable selection for each split in the classification tree is conducted only from a small random subset of predictor variables).

Wrapper methods consider the selection of a set of features as a search problem, where different combinations are prepared, evaluated and compared to other combinations. A machine learning model is used to evaluate a combination of features and assign a score based on model accuracy. There are two different approaches: stability selection and recursive feature elimination.

The **stability selection** is a relatively novel method for feature selection, based on subsampling and applying a machine learning algorithm. The general idea is to apply a feature selection algorithm on different subsets of data and with different subsets of features. After repeating the process a number of times, the selection results can be aggregated, for example by checking how many times a feature ended up being selected as important when it was in an inspected feature subset. It is expected that strong features have scores close to 100%, since they are always selected when possible. Weaker, but still relevant features will also have non-zero scores, since they would be selected when stronger features are not present in the currently selected subset, while irrelevant features would have scores (close to) zero, since they would never be among selected features.[23]

Recursive feature elimination algorithm works by recursively removing attributes and building a machine learning model on those attributes that remain. It uses the model accuracy to identify which attributes (and combination of attributes) contribute the most to predicting the target attribute, so it depends highly on the chosen algorithm. It has shown better results using a ridge regression instead of a linear regression, as the chosen machine learning model. For datasets with many variables relatively strongly correlated with one another and relatively weakly correlated with the target variable, this approach may result in slightly different feature choices from those made by naive model-based selection.[35] The disadvantage is that, since it's necessary to train the model many times, this approach is multiplicatively slower.

Embedded methods learn which features best contribute to the accuracy of the model while the model is being created. The most common type of embedded feature selection methods are regularisation methods. Regularisation methods are also called penalisation methods that introduce additional constraints into the optimisation of a predictive algorithm (such as a regression algorithm) that bias the model toward lower complexity (fewer coefficients). Examples of regularization algorithms are the LASSO, Elastic Net, Ridge Regression and Logistic Regression. [23]

Additionally, there is a python tool available on GitHub [21] which includes some of the most common feature selection methods: features with a high percentage of missing values, collinear (highly correlated) features, features with zero importance in a tree-based model, features with low importance, features with a single unique value.

Given the individual tools available, it is possible to build a pipeline that applies them automatically to any dataset. It is then possible to calculate the average and variance of the importance given to the features by all the different approaches and apply feature selection based on that score.

Classification algorithms

Decision trees are algorithms that test a series of questions and conditions organised in a tree structure. In the decision tree, the root and internal nodes contain attribute test conditions to separate records that have different characteristics. All the terminal nodes are assigned the majority class of its records. The decision tree's algorithms must provide a method for choosing the test condition for different attribute types as well as an objective measure for evaluating the goodness of each test condition.

ID3 (Iterative Dichotomiser 3) was developed in 1986 by Ross Quinlan [25]. The algorithm creates a multiway tree, finding for each node (i.e. in a greedy manner) the categorical feature that will yield the largest information gain for categorical targets. Trees are grown to their maximum size and then a pruning step is usually applied to improve the ability of the tree to generalise to unseen data.

C4.5 [24] is the successor to ID3 and removed the restriction that features must be categorical by dynamically defining a discrete attribute (based on numerical variables) that partitions the continuous attribute value into a discrete set of intervals. Pruning is done by removing a rule's precondition if the accuracy of the rule improves without it.

CART (Classification and Regression Trees) is very similar to C4.5, but it differs in that it supports numerical target variables. CART constructs binary trees using the feature and threshold that yield the largest information gain at each node.

Scikit-learn uses a version of the CART algorithm that does not support categorical variables. In this implementation the features are always randomly permuted at each split. Therefore, the best found split may vary, even with the same training data. To obtain a deterministic behaviour during fitting, *random_state* has to be fixed.

The decision tree's algorithms require some previous data processing. It is important to balance the dataset before training to prevent the tree from being biased toward the classes that are dominant. **Class balancing** can be done by sampling an equal number of samples from each class, or by normalising the sum of the sample weights (*sample_weight*) for each class to the same value. The user can also use *min_samples_split* or *min_samples_leaf* to

ensure that multiple samples inform every decision in the tree. A very small number will usually mean the tree will overfit, whereas a large number will prevent the tree from learning the data.

- **Available tools:** R packages (e1071, rpart, caret, randomForest, nnet, arules, ROCR) and Python libraries (numpy, scipy, pandas, matplotlib, Scikit-learn, TensorFlow, PyTorch, nltk, scrapy)

Ensemble methods

The goal of ensemble methods is to combine the predictions of several base estimators built with a given learning algorithm in order to improve applicability and robustness over a single estimator. Two families of ensemble methods are usually distinguished: **averaging methods** and **boosting methods**.

In the first family, the driving principle is to build several classifiers independently and then to average their predictions. On average, the combined estimator is usually better than any of the single base estimators because its variance is reduced. In this family there is Bagging methods and Forests of randomised trees (Random Forests and Extremely Randomized Trees).

Bagging methods form a class of algorithms that fit several instances of a classifier on random subsets of the dataset and then aggregate their individual predictions to form a final prediction. These methods are used to reduce the variance of a base estimator (e.g., a decision tree), by introducing randomisation into its construction procedure. There are different approaches that mostly differ from each other by the way they draw random subsets of the training set: when random subsets of the dataset are drawn as random subsets of the samples, then this algorithm is known as Pasting[17]; when samples are drawn with replacement, then the method is known as Bagging[18]; when random subsets of the dataset are drawn as random subsets of the features, then the method is known as Random Subspaces[19]; when base estimators are built on subsets of both samples and features, then the method is known as Random Patches[20]. Bagging methods provide a way to reduce overfitting, in contrast with boosting methods.

A **forest of randomised trees** is an ensemble estimator that fits several decision tree classifiers on various subsets of the dataset and then aggregates using averaging to improve the predictive accuracy and control over-fitting. It distinguishes from bagging by considering only a subset of the features in each node (introduces more randomness), while bagging considers all. In the scikit-learn library there are two available algorithms, the RandomForestCalssifier and the ExtraTreesClassifier.

In the **Random Forest** algorithm's implementation, the size of the subset of samples is always the same as the original input sample size, but the samples are drawn with replacement if *bootstrap=True* (default). [35]

The **Extremely Randomised tree classifier** [26] (ExtraTree) differs from the previous in each individual decision tree. When looking for the best split to separate the samples of a node into two groups, random splits are drawn for each of the randomly selected features (maximum number considered is set by user) and the best split among those is chosen [35].

In the scikit-learn library the main parameters to adjust these methods are *n_estimators* and *max_features*. The former is the number of trees in the forest. The larger the better, but also the longer it will take to compute. In addition, the results will stop getting significantly better beyond a critical number of trees. The latter is the size of

the random subsets of features to consider when splitting a node. The lower the greater the reduction of variance, but also the greater the increase in bias. Empirical good default values *are max_features=sqrt(n_features)* where *n_features* is the number of features in the data. In addition, in random forests bootstrap samples are used by default *(bootstrap=True*) while the default strategy for extra-trees is to use the whole dataset (*bootstrap=False*).

Once trained, the tree can be exported in Graphviz format using the *export_graphviz* exporter, which allows an easily interpretable visualization of the decision tree. The *export_graphviz* exporter also supports a variety of aesthetic options, including coloring nodes by their class and using explicit variable and class names if desired.

By contrast, in **boosting methods**, base estimators are built sequentially, and each one tries to reduce the bias of the previous combined estimator. The motivation is to combine several weak models (i.e., models that are only slightly better than random guessing) to produce a powerful ensemble. This family includes AdaBoost and Gradient Tree Boosting.

The algorithm **AdaBoost** was introduced in 1995 by Freund and Schapire [27]. The core principle of AdaBoost is to fit a sequence of weak learners on repeatedly modified versions of the data. The predictions from all of them are then combined through a weighted majority vote (or sum) to produce the final prediction. The data modifications at each so-called boosting iteration consist of applying weights {w1,w2,…wN} to each of the N training samples. Initially, those weights are all set to wi = 1/N, and for each successive iteration the sample weights are individually modified, and the learning algorithm is reapplied to the reweighted data. At a given step, those training examples that were incorrectly predicted by the boosted model induced at the previous step have their weights increased, whereas the weights are decreased for those that were predicted correctly.

Gradient Tree Boosting is a generalisation of boosting to arbitrary differentiable loss functions (functions that measure the performance of the algorithm, usually by calculating the accuracy). Similar to other boosting algorithms, this one builds the additive model in a greedy fashion where the newly added tree tries to minimise the loss, given the previous ensemble. This approach shows higher robustness to outliers in output space (via robust loss functions). A stochastic gradient boosting was also proposed[29], which combines gradient boosting with bootstrap averaging (bagging). At each iteration the base classifier is trained on a fraction subsample of the available training data. The subsample is drawn without replacement. A typical value of subsample is 0.5.

In scikit-learn the number of weak learners (i.e. regression trees) is controlled by the parameter *n_estimators*. The *learning_rate* is a hyper-parameter in the range [0,1] that controls overfitting via shrinkage as it scales the step length the gradient descent procedure. Another strategy to reduce the variance is by considering subsets of the features, like the random splits in a Random Forest classifier. The number of features in each subset can be controlled via the *max_features* parameter. Empirically, it is possible to see that doing subsets of features with shrinkage can further increase the accuracy of the model while doing it without shrinkage does poorly. [35]

The parameter *learning_rate* strongly interacts with the parameter *n_estimators*, the number of weak learners to fit. Smaller values of *learning_rate* require larger numbers of weak learners to maintain a constant training error. Empirical evidence suggests that small values of *learning_rate* favor better test error.[35]

Gradient boosting models, however, comprise hundreds of regression trees thus they cannot be easily interpreted by visual inspection of the individual trees.

XGBoost (eXtreme Gradient Boosting) is an advanced implementation of gradient boosting algorithm. Standard gradient boosting implementation has no regularisation like XGBoost, therefore it also helps to reduce overfitting. Another difference regards tree pruning: a gradient boosting algorithm would stop splitting a node when it encounters a negative loss in the split, while XGBoost makes splits up to the *max_depth* specified and then starts pruning the tree backwards and remove splits beyond which there is no positive gain. Sometimes a split of negative loss may be followed by a split of positive loss (of higher absolute value). XGBoost will go deeper and it will see a combined effect of the split and keep both. Another advantage is that XGBoost allows running a cross-validation at each iteration of the boosting process, so it is easier to get the optimum number of boosting iterations in a single run.

2.3 Open issues

Common approaches in bioinformatics include the identification of genes and mutations with the aim of better understanding the genetic basis of a disease, unique adaptations (such as the tumour genome), desirable properties (in agricultural species), or differences between populations. Bioinformatic research groups often focus on one approach to study one biological problem, which means that in broad and heterogeneous fields like cancer biology, many projects become highly specific in their application and hardly transversal to other biological issues.

Immunotherapy is only one of the many cancer therapies under research, but the number of open questions related to it is still very high. Inducing immune responses to cancer requires a balancing phenomenon where the immune response against the cancer cells should be enhanced, while the autoimmune responses against normal cells should be minimized. Nonetheless, the sources of immune evasion are still under discussion and are constantly being discovered and updated. At the same time, the research in immunotherapy progresses in parallel, with a new therapy focused in each new immune-evasion mechanism, and a different subset of patients responding successfully to it.

The study of immunotherapy application requires the analysis of several biological data types that influence the phenomena, such as genomic features of the cancer, molecular features of the patients, and the immune system composition. Currently, researchers use many different tools and platforms to store and analyse biological data, often coming from TCGA database. However, it is often difficult to integrate and analyse data from these various platforms, and often researchers don't have access to the raw or primary data created by other studies or lack the computational tools and infrastructure necessary to integrate and analyse it.

To sum up, the main challenges are the different data types, the massive size of the data and the techniques available to analyse the data.

There is a need for visual platforms that allow the analysis of several data sources integrated, allowing the application of several approaches, mainly machine learning, and provide a user-friendly tool to speed up the progress in the field of cancer biology.

3 Machine Learning tool: AutoTCGA

Motivation

The existent problem regarding the study of the influence of cancer features in immune-evasion mechanisms occurrence requires a general solution that allows the users to make several particular requests to the system.

Usually, a laboratory team will focus on studying a single protein or group of proteins related with a single cellular process, and the research is focused on the function of a single disease or group of related diseases. For instance, at Guy's Cancer Centre in London, the Cancer Bioinformatics group is studying the influence of the expression of gene X (name omitted due to confidentiality concerns) in the occurrence of the immune-evasion mechanism "HLA absence" in people with triple-negative breast cancer (TN BRCA). These studies will often be approached with *ad-hoc* methods, such as statistical analysis, that work only for that specific question. A general approach will solve many of these typical hypothesis testing problems. In this case the general biological problem is the study of the influence of biological features (genomic instability, immune cells burden, etc) in the occurrence of immune-evasion mechanisms, across all cancer types. Once this is implemented, a research group studying a particular phenomenon in cancer is able to use it to test a hypothesis.

A generalisation of a biological problem is something that a computational approach can accomplish, implementing machine learning techniques to handle the large volumes of data. However, many of the bioinformatics tools currently available are also motivated by specific questions, hence the need to build large pipelines of different softwares to analyse data. A useful tool would assemble all steps from the raw data to the results' visualisation.

Problem formulation

The research project of the Cancer bioinformatics group can be translated into a classification problem, where the class is the "HLA absence" evasion mechanism, the patients considered are the breast cancer patients (BRCA), and the TCGA biological features are used to describe each patient. Then it would be relevant to assess the importance of the feature "gene X expression". Many research groups are interested in analogous problems, regarding other immune-evasion problems and different sub-sets of patients.

The biological requirements for this system include the choice of immune-evasion mechanism under study, the possibility of filtering the dataset using biological features values (such as demographic features, gene expression features or cancer type), and the visualization of the biological features distribution across the prevalence and absence of the immune-evasion event, as well as their importance to build the models. Computationally, this problem would be a multilabel classification problem, as the immune-evasion mechanisms are not incompatible, but each immune-event mechanism can be described as a binary classification problem as well.

This web tool can be described as a platform for multiple hypotheses testing, where a user with technical expertise in the biological field wishes to study the immune-evasion events (one of the classes) occurring for a particular cancer type (a specific population), training a tree-based binary, multiclass or multilabel classifier with a set of features (dataset). It can be found at https://github.com/andreiarog/AutoTCGA .

3.1 System's Architecture

The overall architecture of the system is described in Figure 4, including the main steps: Data Integration, Data Exploration, Pre-processing, Training, Evaluation and Visualization. Apart from the Data Integration module, the remaining modules of the system were built in Python, using the following libraries: NumPy, pandas, scikit-learn [35].
The user is the one who queries the system and filters the data that is cleaned and processed by the data exploration tools. The first module allows a preview of the dataset that will be used for training and evaluation of the classifier. Once approved by the user, the dataset will be given as input to the Data preparation module that further prepares it for modelling according to the user's instructions (dataset, population, class and other system requirements).
The dataset outputted by the Data preparation module will be given to the learning step (training and evaluation modules). The learning and evaluation modules include several types of algorithms for classification: Random Forest, Decision Trees and XGBoost classifier. The classification algorithms are limited to tree-based algorithms, since it's important to present the models to the users, more precisely the role of each feature in the modelling process, and not only its quantitative importance.
These three modules (data preparation, training and evaluation) are the core of the system and are distributed over 10 sub-modules, represented in Figure 5. The first 6 sub-modules are what composes Data preparation, while the remaining are the training and evaluation modules.
The core of the system is automatically connected and programmed (AutoML) because in this project the goal is to build several personalized models by request of the user. This is the major advantage of the system built, as it can analyse the dataset for any chosen combination of features, desired class, and chosen population of interest, and does so without human supervision, in an accessible web-platform. The automation of the model's hyperparameters is not a priority in this project but it is made available through a grid approach.
Finally, the learning module will output the information learnt regarding the weights of each feature, their distribution across the chosen class, the tree-based model built and the algorithm's performance.
The steps taken to build the system's architecture are the following:

1. Data collection from GDC portal (studies' results)
2. Data integration with Pentaho software
3. Data exploration (iteratively by the user)
    a. Data cleaning
    b. Filter datawarehouse with user's requests (preview of Data preparation)
4. Data preparation
5. Learning and evaluation
6. Assemble results and report to interface

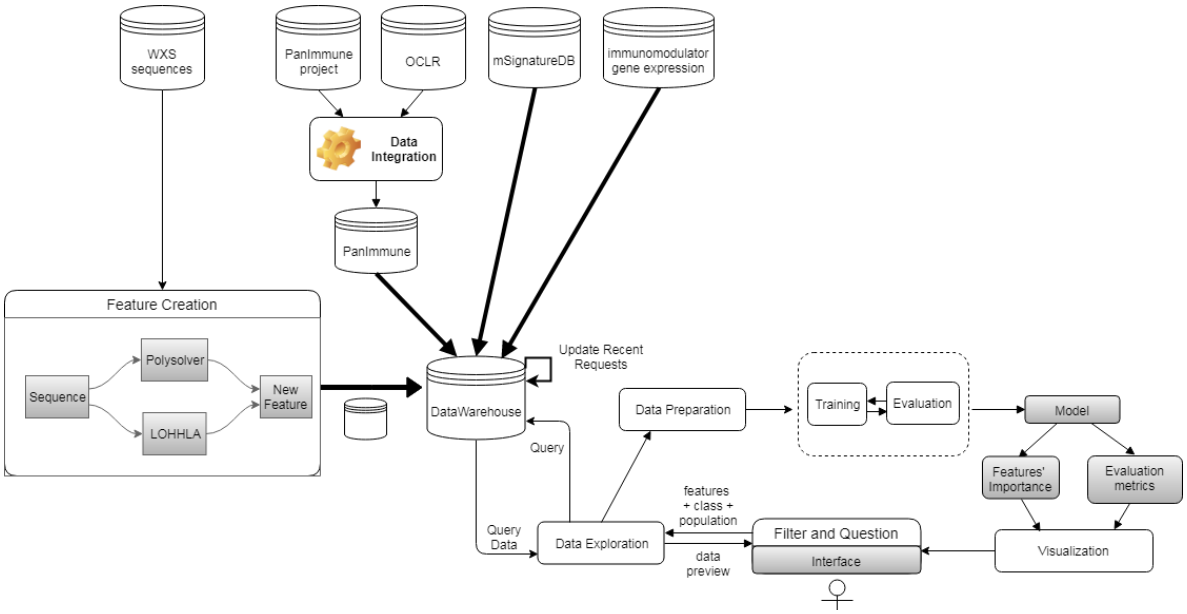


Figure 4: System's architecture.

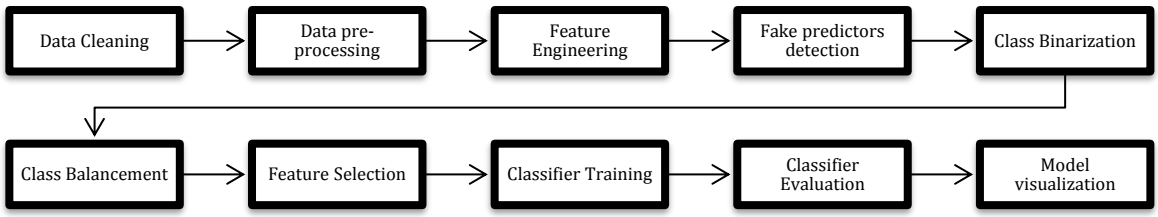


Figure 5: The 10 modules of the automatic system to build models of prediction of a chosen immune-evasion event.

3.2 System's Usage

The users can interact with the system through an interface where they can input their requirements to personalize the models they want to create.

The user has 3 datasets available in the system (detailed below in chapter *Data Source model*) and the attribute to be predicted can be any of the features of the chosen dataset. The datasets can be merged by user request. When choosing the PanImmune dataset, or when choosing to merge this dataset with others, it is also possible to choose "multiclass" or "multilabel" as a class, as well as the dataset pre-processing technique "Feature Engineering". Furthermore, the user can personalize the techniques used to pre-process the chosen dataset, from a selection of 6 possibilities if the PanImmune dataset is chosen (detailed in chapter *AutoTCGA parameters tuning; Pre-processing techniques study*), or 2 possibilities if only one of the other two datasets is chosen (as these only contain numerical variables).

The dataset's features can be filtered by the user, choosing only a subset to give to the system. This can be done either by using the parameters "Sub-set of features to include" or the parameter "Sub-set of features to exclude".

On the other hand, by choosing the population of interest (cancer type), instead of using the whole sample (PanCancer population), the user is filtering the patients given to the system. This can be done with any dataset.

Additionally, the user can input new features, as long as they have a column named "sample_code" with the TCGA identification code to enable the merging with the chosen dataset. When merging, the dataset's patients are filtered by the interception between the chosen dataset(s) and the given input (the join is done in a 'inner' fashion). The feature chosen to be predicted can be one from the chosen dataset(s) or one given as input by the user.

After filtering the dataset and before training the classifier, the dataset is pre-processed according to the user's instructions. Afterwards, several feature engineering and feature selection techniques are available, which can be activated or deactivated by the user (although "Feature Engineering" option is only available if "PanImmune" is selected as the dataset or as one of the datasets). At this stage the user may request a preview of the dataset and then move to model training or re-submit with different parameters.

The algorithm to model the problem can also be personalized by the user, from the available selection of classifiers: Random Forest, Decision Tree and XGBoost. However, if the user chooses to use XGBoost, the multilabel option is not available, as it is not possible to implement in the system, even when choosing the "PanImmune" dataset. The model visualization for this algorithm is also limited and doesn't include tree model visualization.

After training the classifier, the system outputs the tree model, and its features' importance and evaluation metrics. It is also possible to analyse the 5 most important features used in the trees, by visualizing their distributions across the population.

All parameters have default values in the system, which were defined based on the study of chapter *AutoTCGA parameters tuning* below. All models are saved based on the unique combination of class to be predicted, classifier, features used to predict and population, and are loaded when requested by the user, if it already exists in the system.

3.3 Data Source model

To train the machine learning algorithms, the instances can be obtained from three different sources: PanImmune [40], mSignatureDB [41], and immunomodulator gene expression [40].

The main one is the PanImmune dataset, created from a literature article [4], that contains data associated with cancer patients from the TCGA repository (The cancer Genome Atlas)[30], which includes more than 10.000 samples.

To build the PanImmune dataset, the PanImmune study csv files were extracted from the National Cancer Institute's Genomic Data Commons (GDC) [31], which is a data repository that enables sharing information from several cancer genomic studies, providing a unified source to the cancer research community. These files were complemented with the HLA genotype, which was obtained from the same source, using the gdc-client software, as it is considered controlled access information. Finally, another file was added from another source: OCLR application [13]. This source includes the stemness index of each sample, which was obtained from the application of one-class logistic regression (OCLR) machine-learning algorithm to the WXS sequences of TCGA patients.

The final dataset was built by merging the several files into a unique table, using the sample's TCGA id as the merging criteria, including 9977 samples belonging to 33 different cancer types, described by 83 features. The final features of the PanImmune dataset have the original name given in the tables extracted.

The features included in the PanImmune dataset were previously described in Figure 3 and are summarised in the entity-relationship model (Figure 6) where all the attributes are numerical continuous, except for: aliquot_barcode, sample_barcode, patient_barcode, which are identifiers; gender, race, cancer_type, tissue_type, subtype_selected and HLA A1-C2, which are nominal; and age_at_initial_pathological_diagnosis which can be considered ordinal. The final variables are described in Annex.

The entity-relationship model (Figure 6) describes the problem through the features: there is a **Patient** which has its individual genetic identity, represented by the **Genetic Expression**, and an individual **Immune system**, and was diagnosed with a **Cancer.** The cancer is described by its own genetic identity, represented by the **DNA damage,** and by a **Viral Content**. At the same time, the Cancer diagnosed to the patient has a specific type, associated with the primary organ, which can be described by cancer-specific features.

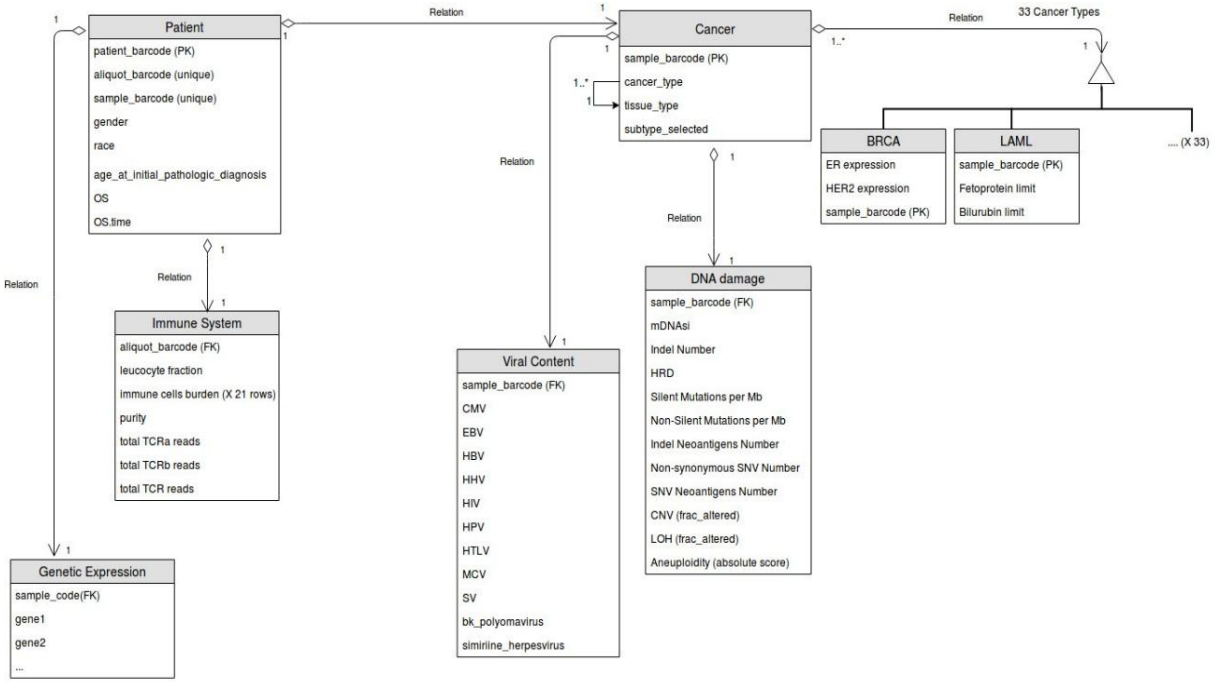


Figure 6: Database entity-relationship diagram of data available in several data sources: PanImmune public and controlled, and OCLR studies. On the right side there are 33 tables (31 omitted due to lack of space) with 745 cancer-type specific variables.

In Table 1 there is a summary of many features currently available at the PanImmune final dataset, or not available, in which case a "NA" is presented, and information regarding their sources. These features don't have a

correspondence with the tables in Figure 6, but instead are the features that require bioinformatics tools to be obtained, or external sources (for instance NIH).

Table 1: Sources of features that require bioinformatics tools to be obtained.

FEATURE	RAW DATA DEPENDENCY	AVAILABILITY
cancer type	-	NIH
aneuploidy/CNV	WXS	GDC – PanImmune
leucocyte content	DNA Methylation	GDC – PanImmune
steamness index	WXS + DNA Methylation	GDC – OCLR
mutations burden	WXS	GDC – PanImmune
viral read counts	WXS	GDC – PanImmune
immune cells burden	RNA-seq	GDC – PanImmune
HLA expression	RNA-seq	GDC – PanImmune
HLA LOH	WXS + HLA genotype + aneuploidy	NA
HLA mutations	WXS	NA
Survival	-	GDC - OCLR
HLA genotype	WXS	GDC – PanImmune

The PanImmune dataset includes the immune-evasion events "Immunesuppression" (IS) and "Cancer Antigenome" (CA) represented by numerical continuous variables (detailed below in chapter *Class generation*). To obtain the missing class "HLA absence" (HA) and the occurrence of "HLA mutations" (HM) it's necessary to use specific software to analyse the WXS sequences. The WXS sequences are the DNA sequences (whole exome sequences) of each tumour sample of each cancer patient from TCGA database. These sequences are available in the GDC portal using the same identifiers as the ones used in the TCGA database, under restricted access which requires a process of request by the university PI. This process was conducted and successful, but due to lack of physical storage space, the files were not downloaded and the module "Feature Creation" in Figure 4 was dismissed, being a possible future extension of the project. For this reason, only the classes "IS" and "CA" are available with the final PanImmune dataset.

The distributions of the variables "Neoantigens_total" and "T.cells.regulatory.Tregs." were plotted as these are two features biologically relevant as classes (Figure 7), the first being the class "CA" and the second a partial measure of the class "IS". Like many other immune system related variables, these two show a skewed distribution to the left side, which suggests the need for a logarithm transformation (used later in Feature Engineering module).

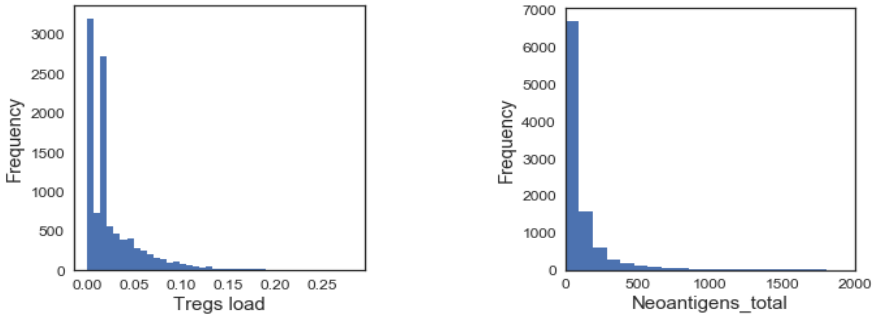


Figure 7: Histograms for Tregs load and Neantigens_total features.

Additionally, another feature was included in the PanImmune dataset: "Treg_cluster", which is a binary variable, obtained from clustering the dataset using the Kmeans algorithm (k=2) of scikit-learn library, and a set of 32 highly variable genes differentially expressed by Tregs, as implemented by Givenchian et. al [42]. This attribute splits the patients into two groups: enriched (1) and depleted (0) in regulatory T-cells, defining the binary class "IS".

The two other datasets available were provided by the Cancer Bioinformatics group at Guy's cancer centre in June of 2019. The mutations signatures database (*mSignatureDB*) contains the signatures from 1 to 30, and was extracted from its web database [41]. This dataset includes 30 continuous features and 4778 samples identified with TCGA codes.

The immunomodulator gene expression dataset contains 75 continuous features and 5869 patients. It was obtained from the PanImmune study [40] but includes a restricted number of patients and therefore was not merged.

Both the mSignatureDB dataset and the immunomodulator gene expression dataset contain the TCGA sample_code, which can be used to merge the datasets with the main PanImmune dataset (by user request), and also contain the cancer type of each patient which can be used to filter the dataset by cancer type.

The dataset choice is the first step in the system's implementation, followed by the merging of datasets, if several sources are requested. The merging of the datasets is done in a "inner" fashion, keeping only the TCGA identification codes that are in common.

The dataset (any of them) will only be pre-processed for classification purposes after going through cleaning, dataset merging, filtering of patients (through cancer type choice or user input of attributes), which are now in practice the **dataset's rows**, and filtering of features (through user defined inclusion or exclusion of a sub-set of features) which are now in practice the **dataset's columns**. At the end of these steps, and before applying any pre-processing techniques, the identifiers are removed (TCGA sample_code) as they couldn't be removed before due to possible merging and filtering.

3.3.1 Cleaning

In the case of PanImmune dataset, once all files are gathered from GDC portal, they will be integrated in a single file, using Pentaho software and applying **ETL** methodology (extraction, transformation and loading). The data transformation includes data cleaning, data generation and extraction of relevant features from the original files (genomic, demographic and immune system related features of the patient, and 2 immune-evasion events).

The remaining 2 datasets available were provided in ready-to-use conditions, in a single csv file each.

For all the datasets, the Data Cleaning module includes handling missing values (replacing by median in continuous attributes or "Unknown" in categorical attributes) and word standardization (detecting synonyms and spelling errors).

All variables in the datasets msignaturedb and immunomodulators gene expression are numerical variables.

This step is of particular importance in Python, as the scikit-learn library doesn't handle missing values, or non-numerical values.

3.3.2 Cancer-type specific dataset

There is the possibility of filtering any dataset to obtain the patients with a specific cancer type. This can be accomplished with the user defined input parameter *cancerType* which takes the acronym of the cancer type, from the possible values: 'GBM ', 'OV ', 'LUAD', 'LUSC', 'PRAD', 'UCEC', 'BLCA', 'TGCT', 'ESCA', 'PAAD', 'KIRP', 'LIHC', 'SARC', 'BRCA', 'THYM', 'MESO', 'COAD', 'STAD', 'CHOL', 'KIRC', 'THCA', 'HNSC', 'READ', 'SKCM', 'CESC', 'LGG ', 'DLBC', 'KICH', 'UCS ', 'ACC ', 'PCPG', 'UVM '.

This parameter will filter the dataset's rows, leaving only the patients with the chosen cancer type, but also adds new columns, which are cancer type specific features. These features were obtained from the patient follow-up records of the PanImmune data repertoire, through the GDC portal, in the same way as the other files, and joined to the final PanImmune dataset using the TCGA id.

To add these features to the dataset they must go through an extraction and cleaning process. They will first be cleaned with text cleaning functions, as most features are of nominal type, unlike the overall features: they will be transformed in lower key values, the spaces in the beginning and end of strings are removed, the empty strings are replaced by "nan" (which is identified as a null value in python), and other unknown-value strings are removed as well (such as "Not available" or "Not Applicable").

The features will then be selected based on the variance of the feature for the patients with the chosen cancer type: if the feature presents more than 2 unique values for that subgroup of patients than it is kept.

For the BRCA and LUAD cancer types, a more personalized cleaning step was implemented, looking at each potential feature and applying more transformations such as: detecting columns of numerical types when appropriate, extracting "year" attributes of date-type attributes, dropping repeated columns (with distinct names, impossible to detect automatically).

This step is done before the features are chosen (either filtered or added as input by the user).

3.3.3 User defined sub-set of features

In the case that the user wishes to use only a sub-set of the features included in the chosen dataset, the system will drop all other columns from the dataset. The class columns will be kept as well. This is accomplished by the parameter "Sub-set of features to include". On the contrary, if the user wishes to remove certain features from the dataset, it

is possible to do so by using the parameter "Sub-set of features to exclude", which will drop those features from the dataset, unless they are the chosen class.

3.3.4 User input of columns

The system allows the user the possibility of inputting columns, as long as there is a column named "sample_code" which has the TCGA identification code of the tumour sample in the format "TCGA_**_**" where * stands for a capital letter or number. The columns are merged with the dataset(s) chosen, in an analogous way as the datasets are merged: joining in a "inner" fashion, keeping only the patients with TCGA identification codes that are in common. This step is done after cleaning but before any pre-processing step.

3.4 Data Preparation: PanImmune

The data preparation step includes all approaches necessary to transform the dataset before using it to train the classification algorithms, which are the first 6 modules of the system (Figure 5), plus the optional module "Class generation" if multiclass, multilabel or class "CA" are chosen.

3.4.1 Pre-processing

The Data Pre-processing step handles the different types of data existent in the dataset and was built with stronger focus on the PanImmune [40] dataset, which is the most diverse dataset, both in data types and in feature's distributions. This step includes the normalization of continuous variables, the creation of dummy variables from categorical attributes, and the categorization of the variable Age in 4 bins: "Child", "Young Adult", "Adult", "Senior", according to the cut points [0,18,35,65,100]. This step can be defined by the user, using the *pre-processing* input parameter, which takes values 1 to 6, each being a combination of pre-processing steps to be applied (in the case of choosing one of the other two datasets, only values 1 and 3 are possible). If the dummy creation step is removed by the user, then all categorical attributes are dismissed, and therefore can't be chosen as classes. If the dummy step is kept, the user will be able to choose each dummy variable as a class. If a categorical attribute is chosen as the class, the patients with value "Unknown" for that categorical attribute are removed from the dataset before training and evaluation.

The six different combinations of pre-processing techniques that are available create six different datasets, starting from the original one, and always applying the step Data Cleaning previously:

1. Cleaned and no Missing Values – No categorical variables added: after cleaning the numerical variables remained unchanged and the categorical variables were removed.
2. Cleaned and no Missing Values – Categorical Variables dummy: after cleaning the numerical variables remained unchanged and the categorical variables were transformed in dummy variables.
3. Normalization of numeric variables – No categorical variables added: after cleaning the numerical variables were normalized between 0 and 1, and the categorical variables were removed.

4. Normalization of numeric variables – Categorical Variables dummy: after cleaning the numerical variables were normalized between 0 and 1, and the categorical variables were transformed in dummy variables.
5. Age categorization: after cleaning the numerical variables remained unchanged, the "Age" variable was categorized into 4 bins, and the categorical variables were transformed in dummy variables, including "Age".
6. Normalization of numeric variables and age categorization: after cleaning the numerical variables were normalized between 0 and 1, the "Age" variable was categorized into 4 bins, and the categorical variables were transformed in dummy variables, including "Age".

3.4.2 Class generation

Although the main purpose of the system is to use the variables "Treg_cluster" and "Neoantigens_total" as classes, represented by "IS" and "CA" respectively, the user can choose any column as the class to be predicted, as long as it's present in the dataset chosen (or merged datasets). Additionally, the user can input personal columns and choose those as classes to be predicted, or only as additional features. The remaining columns will be used as features of the classification problem.

In the case that the user chooses the class "CA" as the class to be predicted, the system will build the "Neoantigens_total" column, which is given by the sum of two other columns, two neoantigen types: 'Neoantigens_total' = 'numberOfBindingExpressedPMHC' + 'Neoantigen_num'.

Using the PanImmune dataset (alone or merging with other) it is possible to choose as class the option "multiclass" or "multilabel". By doing this the classifier won't be solving a binary classification problem but instead a multiclass or multilabel classification problem.

In the multiclass situation a column named "Immune_evasion" is created with 4 possible string values: "Neoantigens depletion & IS abundance", "Neoantigens depletion" , "IS abundance", "None". The values are given based on the binary classes "Neoantigens_total" and "Treg_cluster", assuming the value "Neoantigens depletion & IS abundance" when "Neoantigens_total" is 0 and "Treg_cluster" is 1, assuming the value "None" in the opposite situation, assuming "Neoantigens depletion" when both are 0, and assuming "IS abundance" when both are 1.

In the multilabel situation both attributes "Neoantigens_total" and "Treg_cluster" are given to the classifier, and the target labels are built with the column's values: [0,0], [0,1], [1,0], [1,1].

The definition of the multiclass and multilabel problem is the same, as they both represent a 4-class problem, but are implemented in different ways in the algorithms used from scikit-learn library.

3.4.3 Feature Engineering

In the Feature Engineering module, 3 approaches are used to create relevant features from the existing ones: Log-transformation, Sum of features, Unknown category addition. Opposed to the techniques used in the pre-processing module, in this module the original features are not removed or changed, only used to create new features. This module is only available when the PanImmune dataset is used, since the techniques rely on features present in that dataset.

The numerical features are log-transformed using the formula y=log(x+1) where y is the new feature and x the old feature. Not all are trasnformed, only those who show significant difference in distribution: 'LOH_n_seg', 'LOH_frac_altered', 'MutationsSilent', 'MutationsNonSilent', 'CNV_segs', 'CNV_frac', 'HRD', 'Virus_HCV', 'Virus_HPV', 'numberOfNonSynonymousSNP', 'numberOfImmunogenicMutation', 'numberOfBindingExpressedPMHC', 'Indel_num', 'Immunogenic_indel_num', 'Neoantigen_num', 'T.cells.CD4.memory.resting','totTCRa_reads', 'totTCRb_reads', 'mDNAsi'.

The sum features created were "Viral_Total", "Immune_cells_total" and "Mutations_total" using several variables: **Immune_cells_total** = B-cells + T-cells + Dendritic cells + NK cells + Neutrophils + Plasma cells + Eosinophils + Macrophages; **Mutations_total** = numberOfNonSynonymousSNP + MutationsSilent + MutationsNonSilent + Indel_num

The "Virus_total" variable uses all variables with the name "Virus_X" where X stands for a virus name or acronym. The features used to build the sum features continue to exist and if one of them is chosen as class, the respective sum feature is removed by the Fake predictors detection module (detailed below), and vice-versa.

Finally, some dummy variables created with Unknown were kept: "vital_status_Unknown" and "tumor_status_Unknown" as these may represent the social status of the patient.

3.4.4 Fake predictors detection

Upon analysing the continuous features' correlations (Figure 8), a significant number of highly correlated variables (either directly or inversely) was observed, and therefore the module Detection of Fake Predictors was created. Fake predictors are variables related to the class à priori (ex: total burden of immune cells related with b-cells burden), detected by a correlation factor with module higher than 0.75.

The list of fake predictors is detailed below. For each column X, if X is present in one of the lists, then all other columns on that list are considered fake_predictors of the column X. These lists were agreed after consulting the Cancer Bioinformatics staff at King's College.

mutations_columns = ['Immunogenic_indel_num','numberOfImmunogenicMutation', 'numberOfNonSynonymousSNP', 'Indel_num', 'MutationsNonSilent', 'MutationsSilent', 'Neoantigens_total', 'numberOfBindingExpressedPMHC', 'Neoantigen_num', 'Immunogenic_indel_num_log','numberOfImmunogenicMutation_log', 'numberOfNonSynonymousSNP_log', 'Indel_num_log', 'MutationsNonSilent_log', 'MutationsSilent_log', 'Neoantigens_total_log', 'numberOfBindingExpressedPMHC_log', 'Neoantigen_num_log', 'Mutations_total']

immune_cells_columns = ['Immune_cells_total', 'B.cells.naive', 'B.cells.memory', 'Plasma.cells', 'T.cells.CD8', 'T.cells.CD4.naive', 'T.cells.CD4.memory.resting', 'T.cells.CD4.memory.activated', 'T.cells.follicular.helper', 'T.cells.regulatory..Tregs.', 'T.cells.gamma.delta', 'NK.cells.resting', 'NK.cells.activated', 'Monocytes', 'Macrophages.M0','Macrophages.M1', 'Macrophages.M2', 'Dendritic.cells.resting', 'Dendritic.cells.activated', 'Mast.cells.resting', 'Mast.cells.activated', 'Eosinophils', 'Neutrophils']

virus_columns = ['Virus_HCV','Virus_HPV', 'Virus_BK_Polyomavirus','Virus_CMV','Virus_EBV', 'Virus_HBV','Virus_HHV','Virus_HIV', 'Virus_HTLV', 'Virus_MCV', 'Virus_SV' ,'Virus_Saimiirine_Herpesvirus','Virus_HCV_log','Virus_HPV_log', 'Virus_total']

The most relevant case of fake predictors are the mutations features, which were detected as fake predictors when using the variable "CA" as class. This variable is created from the sum of 'Neoantigen_num' with 'numberOfBindingExpressedPMHC', as previously explained, which are particular types of mutations related with immune-evasion events. These two features are naturally correlated with other mutations features such as: 'Immunogenic_indel_num', 'numberOfImmunogenicMutation', 'numberOfNonSynonymousSNP', 'Indel_num', etc, which represent wider groups of mutations. It is therefore possible to predict that a specific type of mutation will be predominant in individuals with high levels of overall mutations. For this reason, these features were being used by the algorithms to predict the "CA" class with high accuracy and AUC-ROC scores (Figure 9), but were not being informative, as they were showing a relationship already known à priori by the user.

Nonetheless, the user is able to control this module by choosing a personalized set of features to include or exclude using the parameters "Sub-set of features to include" and "Sub-set of features to exclude".

In the case of multiclass and multilabel, each attribute "Treg_cluster" and "Neoantigen_total" is analysed regarding their individual fake predictors, and they are removed accordingly.

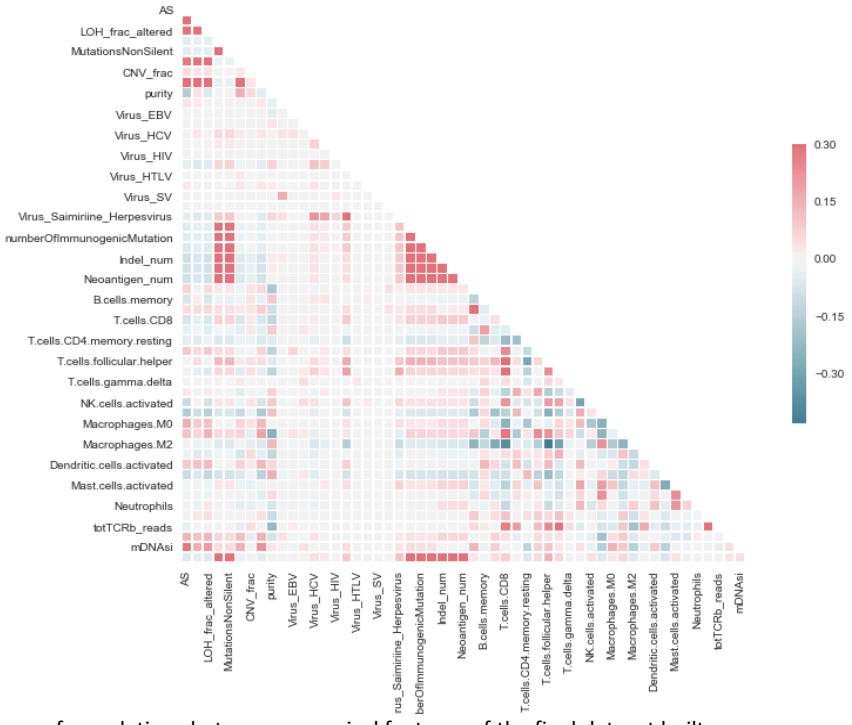


Figure 8: Heatmap of correlations between numerical features of the final dataset built.

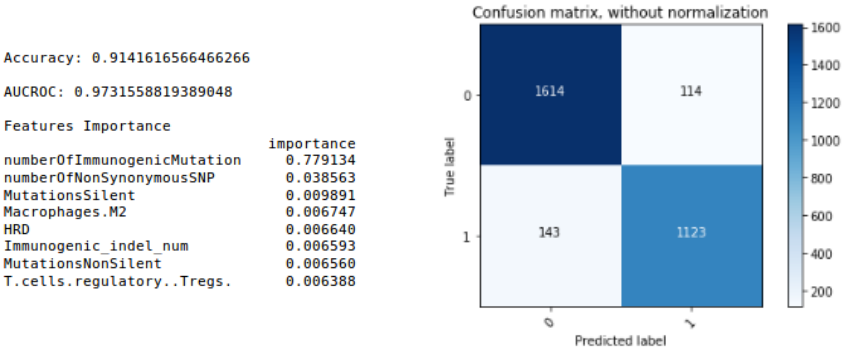


Figure 9: Evaluation metrics, important features and confusion matrix of model obtained when predicting class "CA" with a Decision Tree algorithm, with the fake predictors module inactive.

3.4.5 Class binarization

In the module Class Binarization, each class is binarized with 0 and 1 according to a threshold automatically calculated, if they are not already binarized in the original source. In the case of continuous classes, the binarization is done automatically, by splitting the samples in half after being ordered by the given class value. In the case of categorical classes, they will already be binarized after the dummy variables' creation at the Data pre-processing step. Exceptions to this are the numerical attribute "mDNAsi" and all attributes regarding viral content. For the "mDNAsi" the value 0.2 was used as the threshold below which is 0 and above is 1, because this variable showed a stepwise behaviour and not continuous. The viral content attributes were considered 1 when the value wasn't 0.

In the case of selecting "multiclass" as the class to be predicted, the binarization of each column is made prior to the class generation. Both columns "Treg_cluster" and "Neoantigens_total" are binarized and only then used to create "Immune_evasion" column.

In the case of "multilabel" classification, both columns given to the classifier are binarized: "Neoantigens_total" and "Treg_cluster".

3.4.6 Class balancement

The Class balancement module is called by the Class binarization module, according to the type of class chosen: if they are automatically binarized or not. In this module the classes binarized in a non-automatic way (dummy features and original binary features like "Treg_cluster") are balanced, using an up-sampling procedure which increases the number of samples in the minority class until reaching the number of samples of the majority class (by repeating existing samples).

In the case of selecting "multiclass" or "multilabel" for the prediction, both columns "Neoantigens_total" and "Treg_cluster" will be balanced individually during the class binarization module of each one.

3.4.7 Feature Selection

In the Feature Selection module there are three feature selection techniques applied: Logistic Regression Coefficients, Recursive Feature Elimination and PCA. The first is applied to the entire dataset and the features are considered important if they have coefficients higher than 0.2. The second approach works by recursively eliminating each feature from the dataset and measuring the performance of a chosen classifier, which should decrease for important features. The algorithm used was Logistic Regression with cross-validation (cv=10). The PCA approach works by considering a feature to be more important as it contributes most to the principal components' equations. The final set of selected features is given by the features that belong to at least one of the three sets previously obtained.

If the user chooses the "multiclass" or "multilabel" for prediction, the features selected will be the interception of features selected when applying feature selection to "Treg_cluster" and "Neoantigens_total" columns.

The Feature Selection module is of high importance if the dataset includes the categorical features transformed in dummy variables, because the final number of columns increases from 83 to 937 (see Annex).

3.5 Data profiling: PanImmune immune-evasion features

As explained, the main purpose of the tool developed is to build prediction models for the two immune-evasion features created: IS (Immunosupressors) and CA (Neoantigens), in order to understand the learning mechanism of the tree-based models. To better understand the immune-evasion features created in "Class generation" module, a data profiling analysis was conducted across cancer types, which is represented in a percentage stacked bar-plot in Figure 10. The classes used are the same as the ones generated in the multiclass scenario.

From the distribution it is possible to conclude that there is a high heterogeneity across cancer types, but the vast majority is characterized by having at least one or two dominant classes, and two other classes underrepresented.

Nonetheless, this is not an issue for the training procedure of binary classification, as each individual class is always balanced previously, but might be an issue for the multiclass and multilabel scenarios.

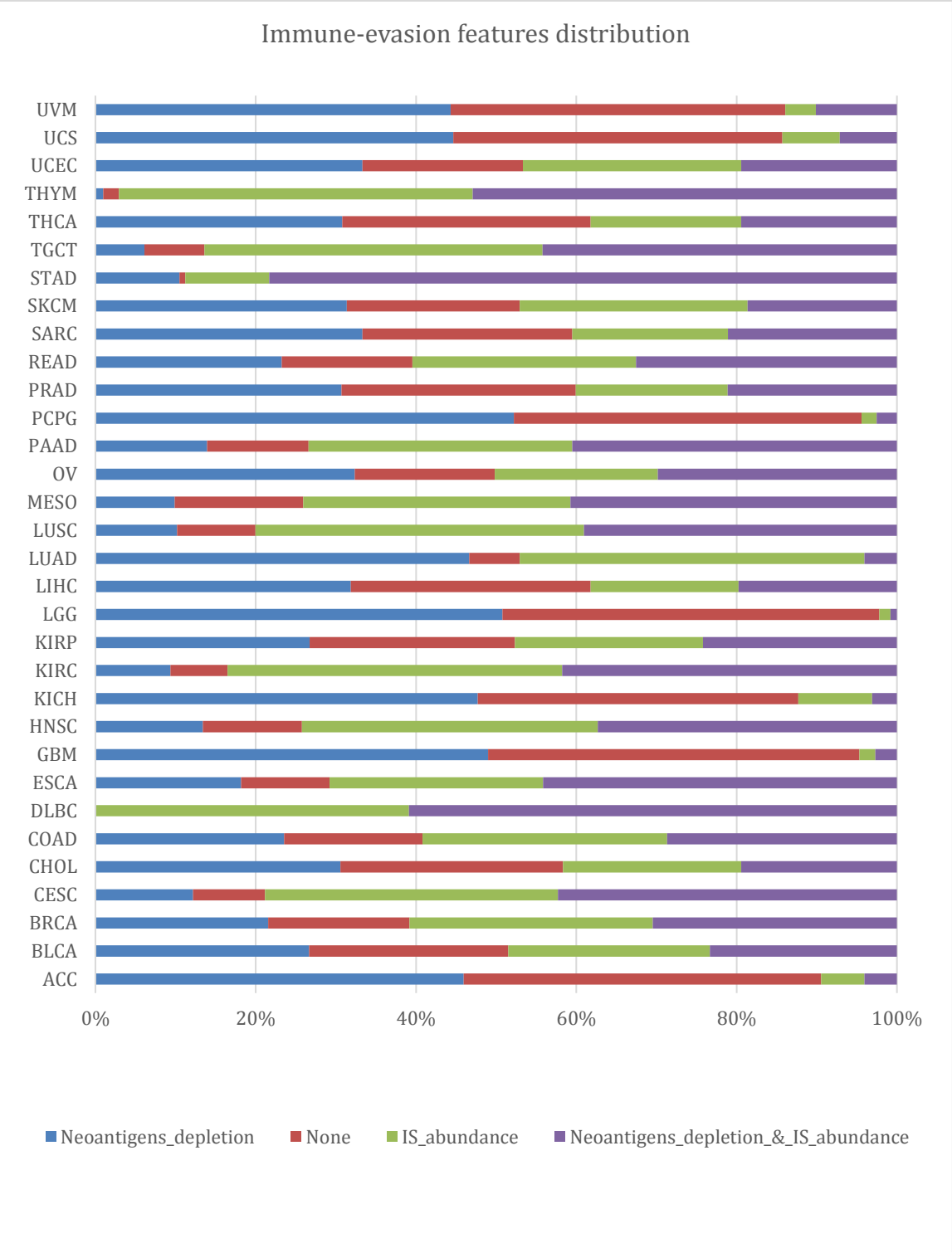


Figure 10: Immune-evasion features distribution across cancer types.

3.6 Data Preparation: Mutations' Signatures and Immunomodulators' Expression

As the variables are all numerical, the only pre-processing step available in the system is the normalization of the features. Additionally, as happens for PanImmune dataset, identifiers are removed in this module. A detailed study was not conducted, as was done for the PanImmune dataset.

The module class generation only applies if these datasets are merged with the PanImmune dataset which contains the immune-evasion mechanisms classes ("CA" and "IS").

The module Feature Engineering is not available when using these datasets individually, since the techniques are specific of the PanImmune dataset's features. The pre-processing module and the class chosen are also limited to some values, as explained previously. All other modules work for these datasets, analogous to the previously described for PanImmune dataset.

3.7 Training and Validation of models

Given a dataset, a class to be predicted and a population to be studied, as well as other user requirements (such as dataset filters and inputs), the system will build a tree-based classifier, trained by and tested by cross-validation, output that tree-based model together with its list of important features, their distributions across the class values, and the correspondent evaluation metrics, sustained by a statistical significance given by the p-value.

The algorithm can be a binary classifier for the prediction of any attribute in any dataset, but also a multiclass or multi-label classifier when predicting immune-evasion mechanisms using the PanImmune dataset. The algorithm is defined by the user, from a selection of 3 possibilities: Decision Tree, Random Forest or XGBoost from the scikit-learn python library. The set of available algorithms was determined during the study of the default parameters (chapter *AutoTCGA: parameter tuning*)

The Random Forest implementation uses the Gini index as the feature split criteria. The sub-sample size is always the same as the original input sample size, but the samples are drawn with replacement if bootstrap=True. The features are always randomly permuted at each split, so the best-found split may vary, even with the same training data, max_features=n_features and bootstrap=False. To obtain a deterministic behaviour during fitting, the random_state was fixed as "123". The Decision Tree implementation uses the same criteria and the same random_state value, to guarantee repeatability. The XGBoost algorithm was obtained from the xgboost python library [43]. It is important to notice that the XGBoost implementation used does not allow the prediction with the multilabel option (giving two columns as input), but only with the binary classification or multiclass scenario. It would be possible to apply an encoder to convert the multilabel prediction in a binary prediction, but that would be equal to the multiclass problem described.

The chosen algorithm is optimized with RandomSearchCV and GridSearchCV functions of scikit-learn library which implement cross-validation (cv=10) to train several instances of the classifier with different hyperparameters and return the one with best evaluation metrics (accuracy being the choice criteria). RandomSearchCV tests a set of hyperparameters' values significantly away from each other, and then GridSearchCV will test a grid of values close to

the value that obtained the best scores in the previous step. In contrast to GridSearchCV, not all parameter values are tried out by RandomSearchCV, but rather a fixed number of parameter settings is sampled from a given range. The initial unrestricted parameters used to train the Random Forest models are the following:

RandomSearch

```
param_grid = {
        'bootstrap': [True, False],
        'max_depth': [10, 100,500, None],
        'max_features': ['auto', 'sqrt'],
        'min_samples_leaf': [1, 10, 20],
        'min_samples_split': [2, 20, 40],
        'n_estimators': [10,100,500]
    }
```

GridSearch

```
param_grid = {
            'bootstrap': [bootstrap],
            'max_depth': [max_depth],
            'max_features': [max_features],
            'min_samples_leaf': [min_samples_leaf-1, min_samples_leaf, min_samples_leaf+1,min_samples_leaf+2],
            'min_samples_split': [min_samples_split-2, min_samples_split, min_samples_split+2],
            'n_estimators': [n_estimators-1, n_estimators, n_estimators+1, n_estimators+2]
        }
```

The Decision Tree parameters are the same but excluding the 'n_estimators' and 'bootstrapping' which are not applicable. Some parameters were limited to guarantee a good visibility of the tree model, and therefore easy interpretability of it, namely the tree-depth in all algorithms is bound to 6 levels and the number of trees in the Random Forest algorithm and XGBoost is bound to 10 trees:

RandomSearch

```
param_grid = {
        'bootstrap': [True, False],
        'max_depth': [3,5],
        'max_features': ['auto', 'sqrt', None],
        'min_samples_leaf': [2, 6, 10],
        'min_samples_split': [2, 10, 20],
        'n_estimators': [3,5]
    }
```

GridSearch

```
param_grid = {
        'bootstrap': [bootstrap],
        'max_depth': [max_depth-2, max_depth-1,max_depth, max_depth+1],
        'max_features': [max_features],
        'min_samples_leaf': [min_samples_leaf-1, min_samples_leaf, min_samples_leaf+1],
        'min_samples_split': [min_samples_split-2, min_samples_split, min_samples_split+2],
```

```
        'n_estimators': [n_estimators-1, n_estimators, n_estimators+1, n_estimators+2]
    }
```

A similar approach was used for XGBoost optimization but applying only the RandomSearchCV function, due to exceeding training time. The unrestricted parameters used were:

```
params = {
        "colsample_bytree": uniform(0.7, 0.3),
        "gamma": uniform(0, 0.5),
        "learning_rate": uniform(0.03, 0.3),
        "max_depth": randint(2, 6),
        "n_estimators": randint(100, 150),
        "subsample": uniform(0.6, 0.4)
  }
```

And the restricted parameters used were:

```
params = {
        "colsample_bytree": uniform(0.7, 0.3),
        "gamma": uniform(0, 0.5),
        "learning_rate": uniform(0.03, 0.3),
        "max_depth": randint(2, 6),
        "n_estimators": randint(1, 10),
        "subsample": uniform(0.6, 0.4)
  }
```

The quality of the models is given by the evaluation metrics: precision, recall, accuracy and AUC-ROC. A class can be chosen, using the common terminology of "positive" when an instance belongs to that class, and "negative" when it doesn't. **Precision** measures the proportion of actual positives samples from those that are identified as such (e.g., the percentage of patients whose tumour evade the immune system from those who are identified as doing so). It's computed by the ratio $\frac{TP}{TP+FP}$. **Recall** measures the proportion of samples identified as positives from those that are actual positives (e.g., the percentage of patients whose tumours evade the immune system and are correctly identified as doing so). It's computed by the ratio $\frac{TP}{TP+FN}$. **Accuracy** measures how accurate is the overall performance of a model, considering both positive and negative classes without worrying about the imbalance of a data set. It's computed by the ratio $\frac{TP+TN}{TP+TN+FP+FN}$. A forth evaluation metric considered is area under de ROC curve (**AUCROC**), which is an indication of the ability of the model to rank two observations, one from each class (positive and negative), if we randomly select them. To obtain this metric the algorithm calculates the Area Under the Receiver Operating Characteristic Curve (ROC AUC) from prediction scores. The ROC curve is created by plotting at various threshold settings the true positive rate (TPR), which is also known as the recall or sensitivity, against the false positive rate (FPR), which can be calculated as (1-TNR), where TNR is also known as specificity or selectivity.

For the multiclass and multilabel evaluation, the general formulas do not apply directly, as the approach used was the "micro" averaging for the calculation of precision, recall and accuracy. Therefore, the scores are obtained calculating the metrics globally, across all the N classes/labels, by counting the total true positives, false negatives and false positives: $TP = \sum_i^N TP_i$, $TN = \sum_i^N TN_i$ $FP = \sum_i^N FP_i$, $FN = \sum_i^N FN_i$. This averaged metric does not represent a uniform value among classes but, in comparison with a macro averaging (not-weighted), the micro averaging is better suited for problems where there may be imbalanced classes, which can be the case in multiclass and multilabel.

When evaluating a multiclass and multilabel problem, the AUC-ROC would be averaged over all classes in a one-versus-rest approach. This results in imbalanced datasets where each class is the minority compared to the other 3 classes together. For this reason, the AUC-ROC is not an appropriate measure for multiclass and multilabel scenarios, and should be replaced by PR-AUC (**Precision-Recall AUC**), as this metric reflects when the positive prediction is correct (precision), and when the model assures that as many of the positives are predicted as positives as possible (recall), which is relevant in an imbalanced dataset. Additionally, the computation of PR-AUC for multiclass classification was only possible by transforming the multiclass problem in a multilabel problem (in practice creating dummy features of the multiclass column, without intercepted classes) and computing the micro average of the PR-AUC for each label.

The evaluation metrics refer to the best cross-validation model chosen during the optimization and are not obtained using a test set. Therefore, the metrics are merely representative of the relative quality of the models when compared with other models (using different system parameters, for instance).

Using a permutation test score with 100 permutations, the cross-validation results can be sustained by a p-value. Given a dataset, there are many hypotheses which can be claimed, and tested, using a hypothesis test. In this case the permutations performed result in a population of metric scores of the model built previously. The hypothesis tested, the null hypothesis, is that the scores were obtained by chance. When performing a hypothesis test in statistics, a p-value helps determining the significance of the results. The p-value is a number between 0 and 1 and interpreted in the following way: a small p-value (typically ≤ 0.05) indicates strong evidence against the null hypothesis, so the null hypothesis is rejected. In practice the p-value is calculated as: (C + 1) / (n_permutations + 1) where C is the number of permutations whose score >= the true score, and therefore the best possible value is given by 1/(101) which is approximately 0,01 [45].

Overall, given an algorithm, the system will apply a cross-validation technique to optimize the training and output the statistically significant scoring metrics (defined by default as accuracy and AUC-ROC or PR-ROC by recommendation of staff at Cancer Bioinformatics). As mentioned, the list of important features, the respective feature importance, their distribution across the class values, the tree model trained, and the associated evaluation metrics will be outputted. The tree model has nodes coloured according to the predominant class (for instance two different colours in binary classification, one for each class) and each node opacity is given by its purity in the

predominant class. The feature weight is calculated based on how important each feature is to obtain leaf nodes with a high gini index, and are not necessarily proportional to the tree level that their nodes are in.

3.8 Interface Prototype

To facilitate the usage of the system, a web app was created using the Flask framework. The interface is presented in html pages with several forms for user input of parameters. The framework controls the behaviour of the html pages when POST and GET methods are called in them, implementing those behaviours in a python file (app.py) which loads the home page on the browser when it's run on the terminal. The web app includes a home page (home.html) with links to the other html pages: "tool", "about" and "home" (itself). The most important page is "tool" where the user inputs the list of available parameters: "Algorithm" (RF, DT and XGB), "Class" (Neoantigens, Immunosupressors, Other), "Dataset(s)" (PanImmune, Immunomodulators' gene expression, Mutation's Signatures), "Population" (cancer type from 33 available described above in chapter *Cancer-type specific dataset*), "Pre-processing" (1 to 6, according to the definitions above in chapter *Data Preparation: PanImmune; Pre-processing*), "Feature selection" and "Feature engineering" (Boolean variables) and further advanced settings: "Sub-set of features to exclude", "Sub-set of features to include", "Input columns". The parameters "pre-processing" and "Feature engineering" are only available when the dataset chosen is "PanImmune". By choosing the option "Other" in the parameter class, a text area appears where the user can input the name of the column to be used as feature, which can be any column present in the dataset(s) chosen or in the input columns given in the advanced settings. In each of the advanced settings' parameters there is a brief explanation on how the input should be provided in the placeholders of the text areas.

The page "about" also includes a detailed explanation on the type of inputs available and how they influence the system. In this page there is also a description on the models already saved in the system (the combinations tested in chapter *AutoTCGA parameters tuning*). The models not listed in this page are not saved and therefore will have to be built by the system, therefore taking more time to run.

The "tool" page includes links to two other pages: upon clicking the "Preview" button the user is redirected to the "preview" page, while when clicking the button "Submit" the user is redirected to the "results" page, both connections being implemented in the app.py file. The behaviour of the "preview" and "results" pages are also implemented in the app.py file. The behaviour of the "preview" page consists in calling a function of the system with the user defined parameters, given in the "tool" page, which simulates the system's Data Preparation, which includes modules from 1 to 7 (Figure 5) retrieving the dataset before forwarding it to the training stage. The "results" page calls the main function of the system which implements all 10 modules described previously.

As previously mentioned, the modules work automatically, regardless of the user input parameters. As mentioned, the output of the classification will be in the form of weights given to each feature by the algorithm (feature importance), the distribution of each of the 5 most important features in the chosen population, and a tree structure that represents the model (Model visualization module). To sustain the models, the evaluation metrics (accuracy and AUC-ROC or PR-AUC) will be displayed as well as the p-value. By looking at the tree model the user is able to traceback

the patterns in the feature's values that describe each class and complement those observations with their rank in the feature importance list.

3.9 AutoTCGA performance

The main goal of this project is to create a tool that ultimately builds a tree-based model to predict immune-evasion mechanisms in cancer patients. The implementation of the tool included two immune-evasion, which are not mutually exclusive. For this reason, predicting a multiclass or multilabel result would be the classification problem that best defined the biological problem.

Nonetheless, the results obtained in the chapter *AutoTCGA parameters tuning* showed that even with the best algorithms the evaluation metrics were not very high. It is possible that the low quality or diversity of the data available was responsible for the poor performance of the models when predicting both immune-evasion events at the same time. By uploading new features, it is possible to create completely different models, with better accuracy scores.

At the same time, predicting each immune-evasion individually resulted in models with very high accuracy and AUC-ROC scores. Besides being a good binary classification pipeline for immune-evasion mechanisms, the tool has shown to be highly robust and able to predict any other attribute equally well, which is a consequence of two important characteristics of the tool: code abstraction and modular structure. Therefore, this tool has accomplished better than expected in software engineering performance, as it can be applied to many biological objects of study, and not only to the primary object of study of this project, which were the immune-evasion mechanisms.

To access the performance of the tool amongst other tools, a comparative study was conducted, including tools that analyse TCGA cancer patients, focused on several biological features, besides the immunological features (which were the main initial focus of AutoTCGA tool).

Many of the tools are data visualization tools with very good user experience, that allow building variables distributions, correlations and Kaplan-Meyer curves. These tools have the main data types available at TCGA: gene expression, genomic instability and heterogeneity, and demographic information (clinical follow-up).

The only tool found to apply a classification algorithm was The Cancer Immunome Atlas (https://tcia.at/home). TCIA applies a Random Forest algorithm to predict the cytolytic activity and obtains a very intuitive visual model, the immunophenogram. However, TCIA is focused on characterizing cancer samples according to the immunophenogram results, the immunopheno-score, and is not capable of expanding to other classification targets. Therefore, this model's purpose is to obtain a classification score, and not the features' importance, and the conclusions taken from each tool are very distinct.

Another tool that also builds a visual model is TumorMap (https://tumormap.ucsc.edu/). This tool creates clusters from the patients records and provides their visualization in a two-dimensional grid (applying the OpenOrd algorithm which uses feature vector similarities so that nearby samples are similar to one another). It allows cluster visualization and allows the user to find connections between the attributes and their distributions on the map or to each other using a spatial correlation analysis, but it is hard to interpret quantitatively. This approach goes beyond direct sample

overlaps or correlations, in a way similar to the classification approach, as both techniques search for a pattern in the attributes that explain the similarity of the samples. The disadvantage of this technique is the unsupervised approach used, which does not target any feature or biological mechanism.

A third tool implementing machine learning approaches is iCluster [46], a tool that integrates the clustering of several sources with different data types that characterize the cancer samples (genomic, epigenomic, transcriptomic and proteomic levels). The algorithm allows finding novel cancer sub-types not previously known through clusters with alteration pattern across data types. Like TumorMap, this tool goes beyond correlations between 2 variables, but lacks the informative model of a classification algorithm, with a defined class.

AutoTCGA is to my knowledge the first tool to combine the interpretability power of Decision Trees algorithms, trained in an automatic machine learning pipeline, with the TCGA genomic data, generating statistically and biologically significant conclusions. Even though some tools have applied clustering techniques with good visualization platforms and others have applied powerful classification algorithms, none is able to provide what AutoTCGA does. With the tree-based models the user can traceback and understand the patterns of patients with an attribute of interest (the chosen class). Those results provide useful insight into associations between the variables, which may be a cause or a consequence of the feature under study.

AutoTCGA leverages the easy interpretation and visualization that tree-base models provide. However, the biggest disadvantage is the poor web interface development. The user experience is of high importance in a field where most professionals are not from IT and may not be experienced with programming. Secondly, AutoTCGA doesn't allow downloading as this is not a data visualization tool, so it does not fit the purpose of data repertoire, as many of the remaining tools do.

It has been demonstrated that AutoTCGA is an efficient proof-of-concept of the great application of machine learning pipeline to TCGA data and integrating the computational power of this tool with the high quality of the data visualization tools already developed has great future potential for success.

Table 2: Comparison of AutoTCGA with other tools.

	Features Available						Dataset personalization					Data Analysis				Data Visualization		
Tool/Characteristic	**Gemic Instability (CNV, mutations)**	**Heterogeneity (ploidy, purity)**	**Gemic Expression**	**Viral Content**	**Immune-cells burden**	**Immune events**	**Download**	**Feature selection**	**Patients selection by Cancer type**	**Patients selection by demographic features**	**Feature upload**	**Distributions**	**Correlations**	**Kaplan-Meyer**	**Machine-Learning**	**Web available**	**User-friendly App**	**Visual ML model**
AUTOTCGA	X	X	X	X	X	X		X	X		X	X			X		X	X
TCIA (THE CANCER IMMUME ATLAS) [47]	X	X	X		X		X	X	X	X	X	X	X	X	X	X	X	X
TCPA [48]	X	X	X				X	X	X	X		X	X	X		X	X	
XENA [49]	X	X	X					X	X	X	X	X	X	X		X	X	
TUMORMAP [50]	X		X				X	X	X	X	X	X			X	X	X	X
CRC AGGRESSIVENESS EXPLORER [51]	X	X	X													X	X	
ICLUSTER [46]	X		X					X	X	X								
CBIOPORTAL FOR CANCER GEMICS [52]	X	X	X				X	X	X	X		X	X	X		X	X	

4 AutoTCGA parameters tuning

4.1 Pre-processing techniques study

Using the 6 possible datasets created from the original PanImmune dataset with PanCancer population, a study was conducted to conclude which combination of pre-processing techniques provided best results in the system, and therefore should be used as default parameter. This study was not replicated for the remaining two datasets, as many of the combinations do not apply (only options 1 and 3 apply), since those datasets have only numerical features.

Five algorithms were trained with cross-validation (cv=5): Naïve Bayes (using the Gaussian Distribution for numerical variables), Logistic Regression, Decision Tree Classifier (CART implementation) and Random Forest ensemble. The criteria used to choose the algorithms was the ability for interpretability of the models created, which can be done looking at the weights given to each feature by the model (possible in all of them), or looking directly at the model structure (which can be done using a tree-based algorithm). The algorithms were evaluated by their accuracy, precision, recall and AUC-ROC.

The first class studied was "Neoantigens_total", and at this stage the hyperparameters were left as default by the scikit-learn library, and no tuning was applied. The evaluation results (accuracy and AUC-ROC) for all algorithms applied to the prediction of this class can be seen in Figure 11).

After analysing the results, the algorithm with best performance in all metrics was Random Forest, using pre-processing 5. The scores for precision and recall for Random Forest algorithm were 73,1±2,4% and 71,6±2,5% correspondently, across the 6 pre-processing techniques (results not shown). For this reason, the default pre-processing in the system was set as mode 5, which includes all categorical features and discretization of "Age" into labels but doesn't include normalization of numerical variables.

It is possible that due to the skewed distribution of many variables, the normalization squeezes the range and overlaps the values of many samples, increasing the difficulty in using the variables for prediction.

This analysis was conducted for the class mentioned and ten other classes (results not shown): Neoantigens_total, T.cells.regulatory..Tregs, B-cells memory, Absolute Score (AS), mDNasi, purity, Virus HPV, Copy number Variation (CNV n_segs), Loss of Heterozigozity (LOH _n_segs), Homologous Recombination Deficiency (HRD), Overall Survival (OS). All of them had analogous results, reinforcing the previous conclusions, and supporting the decision of choosing pre-processing 5 as the default parameter of the system.

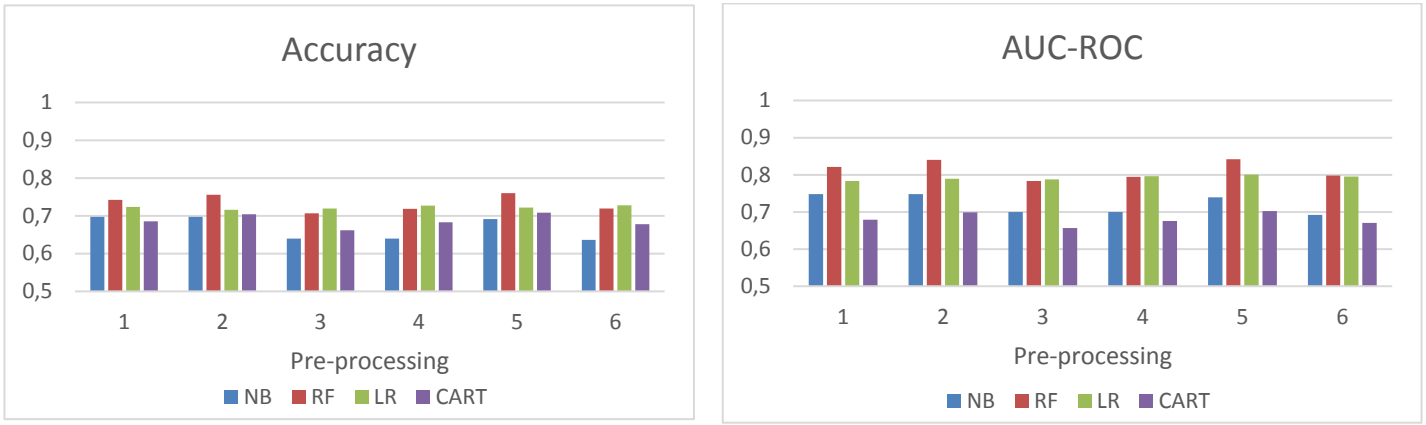


Figure 11: AUC-ROC and accuracy scores obtained when training 4 machine learning algorithms with 6 pre-processing steps, using the full dataset built and the default parameters of sklearn library.

4.2 Feature techniques study

Three combinations of feature techniques were studied: Feature Selection, Feature Engineering and Feature Engineering followed by Feature Selection. These 3 approaches produced 3 different datasets from the original dataset.

Using the best pre-processing approach (number 5) all the three algorithms available in the system were tested: Random Forest, Decision Tree and XGBoost. The criteria used for the choice of algorithms was the results obtained during the pre-processing study of PanImmune dataset which concluded that Random Forest obtained the best evaluation results. The Decision Tree algorithm was kept due to its easy interpretability. XGBoost was introduced as a possible choice as it is a boosting algorithm whose estimators are Decision Trees as well, like Random Forest, and it's an algorithm known for its good performance [44].

Opposed to the pre-processing study, the algorithms were not used with the default parameters, but instead they were optimized for each of the 4 datasets (original plus three datasets produced), using a wide range of hyperparameters and using the restricted hyperparameters (maximum of 6 levels in each tree and maximum of 10 estimators in ensembles), as explained previously in *Training and Validation of models* chapter. For comparison reasons, the results for unrestricted model were also kept, in order to understand the consequences of such restrictions in the models' quality.

The algorithms were evaluated using the accuracy and AUC-ROC metrics (or PR-AUC in the case of multilabel and multiclass), as these were considered to be the most familiar metrics and most easy to interpret by the staff at Cancer Bioinformatics group. The threshold used for visualization purposes in the graphs was 50% for binary classification, and 25% for the multiclass and multilabel problems (graphs adjusted accordingly).

Moreover, another metric was also calculated: the relative standard deviation across the 4 datasets (original plus 3 produced datasets). This score is calculated as the ratio between the standard deviation [53] and the average of the results.

For this study only the biologically relevant classes were tested: "CA" (cancer antigenome) and "IS" (immunosuppressor cells); and the "multiclass" and "multilabel" scenarios were also studied for each population and algorithm combination. Only one dataset was given to the algorithms: the PanImmune dataset; and three populations were chosen: the PanCancer population with 9977 patients, the Breast Cancer (BRCA) population with 1026 patients and the Lung Adenocarcinoma (LUAD) population with 498 cancer patients.

It is of high biological importance to study the best models obtained with the entire PanImmune dataset of 9977 patients (PanCancer) and compare them with the models obtained using the subset of population with BRCA cancer type and LUAD cancer type, as this allows the understanding of particular mechanisms in these cancer types that make the patients more or less potential candidates for immunotherapy.

The BRCA and LUAD cancer types were chosen due to their importance in society. Lung cancer is the leading cause of cancer-related death worldwide, accounting for 1.59 million deaths out of the 8.2 million total cancer deaths in 2012, and Lung adenocarcinoma (LUAD) accounts for 40% of all total lung cancer cases [54]. It usually forms on the periphery of the lungs and may be associated with smoking but it is the most common lung cancer type among non-smokers [55]. For this reason, it is of high interest to study its genetic and molecular characteristics. Breast cancer is the most commonly occurring cancer in women and the second most common cancer overall [56]. Breast cancer is always caused by a genetic abnormality, which makes it a very interesting case study.

The best results were obtained using Random Forest to predict the class "IS" using the LUAD population, but as expected the XGBoost matched and outperformed many of the Random Forest scores across the other populations and binary classes "IS" and "CA". All results shown here were evaluated with a permutation test to ensure statistical significance, and all scores have shown a p-value<0,0099, which is considered significant.

With the results obtained, it was possible to define the default parameters of the system (in addition to the pre-processing techniques already studied): feature selection and engineering combination and default classifier. These parameters are used for any dataset and any population the user desires to study and chosen class to be predicted. The classifier's hyperparameters are distinct for each combination of class, classifier, population and features used. The models represented in these analyses were saved and are immediately loaded when the user requests them to the system. Other models will be made upon request and saved for future users to load if requested.

4.2.1 Classification Baseline

To evaluate the performance of the algorithms in an absolute manner, and not just relative to each algorithm and pre-processing approach, we used the ensemble algorithm AutoSklearn [57] to classify two of the most biologically relevant classes with the PanCancer population. The accuracy results were 0.813 for the prediction of "Neoantigens_total" ("CA") and 0.901 for the prediction of "Treg_cluster" ("IS") using pre-processing approach number 5 and no feature selection or feature engineering techniques applied. Given that the best results obtained with the system weren't too distant (Random Forest algorithm with default parameters applied to dataset PanImmune with population PanCancer and pre-processing 5 obtained accuracy of 0.800 and 0.848 respectively, see Figure 12 and Figure 13) this was considered a satisfactory baseline to continue the analysis.

4.2.2 PanCancer population (all cancer types in PanImmune dataset)

4.2.2.1 Binary classification

For the full PanCancer population, comprising 9977 patients, the best results were obtained when training a Random Forest and XGBoost (very close results), both for prediction of "IS" and "CA". For the former class, models achieved evaluation scores of 80% in accuracy and 88% in AUC-ROC, and for the latter class reached 83% in accuracy and 92% in AUC-ROC, using Random Forest and XGBoost algorithms with unrestricted hyperparameters and using the original dataset. After applying the above-mentioned restrictions, the models' scores remained the same for the IS prediction but decreased to 75% in accuracy and 83% in AUC-ROC for the CA prediction using the Random Forest algorithm, and 78% and 86% in accuracy and AUC-ROC respectively, using the XGBoost algorithm for the CA prediction. Therefore, in the restricted scenario (the actual conditions implemented in the system) the XGBoost algorithm outperforms the Random Forest algorithm when predicting "CA" class in the PanCancer population.

Across the feature engineering and selection techniques the algorithm which obtained higher values of relative standard deviation across the 4 combinations of techniques was the Decision Tree with 2% and 2,5% for accuracy and AUC-ROC results when predicting the "IS" class in unrestricted parameters condition. Indeed, the Decision Tree achieved its highest results when feature selection was applied, reaching 81% for accuracy and 88% for AUC-ROC. Without this technique, the algorithm reached values of 81% for both accuracy and AUC-ROC. For the restricted condition an increase in AUC-ROC was also observed, from 81% with the original dataset to 88% using both feature selection and feature engineering techniques.

The remaining algorithms, and the Decision Tree applied to the prediction of class "CA", showed relative standard deviations lower than 1% which shows that the techniques applied did not increase significantly the scores already obtained with the original PanImmune dataset and PanCancer population.

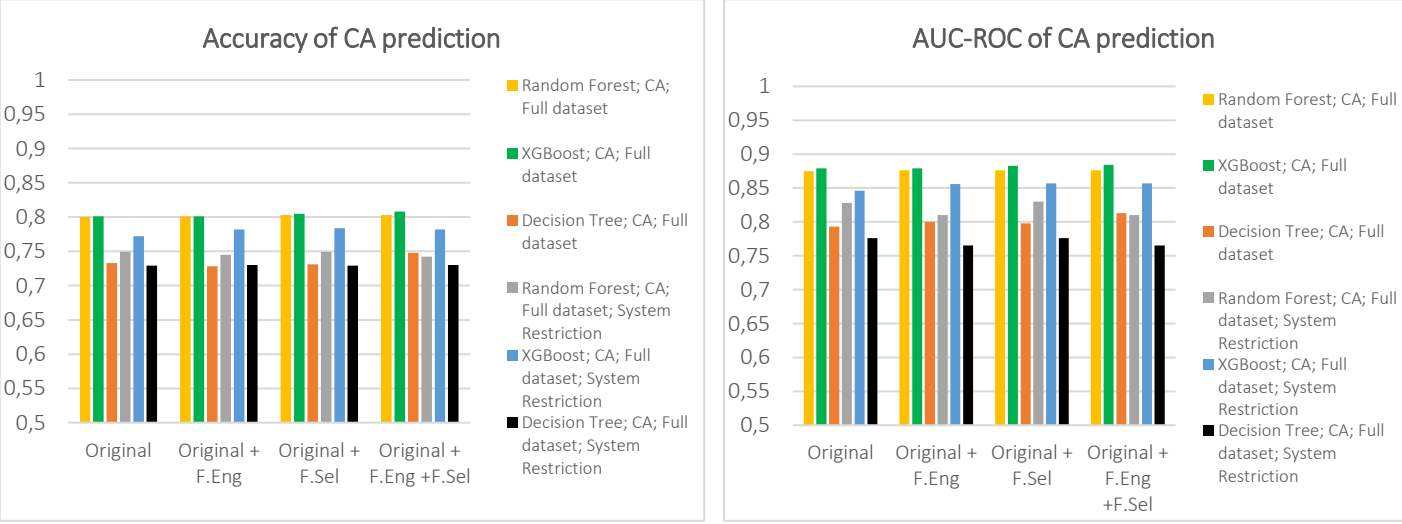


Figure 12: AUC-ROC and accuracy scores obtained when training Random Forest algorithm, XGBoost and Decision Tree with PanCancer dataset, applying Feature Selection and/or Feature Engineering techniques to predict CA.

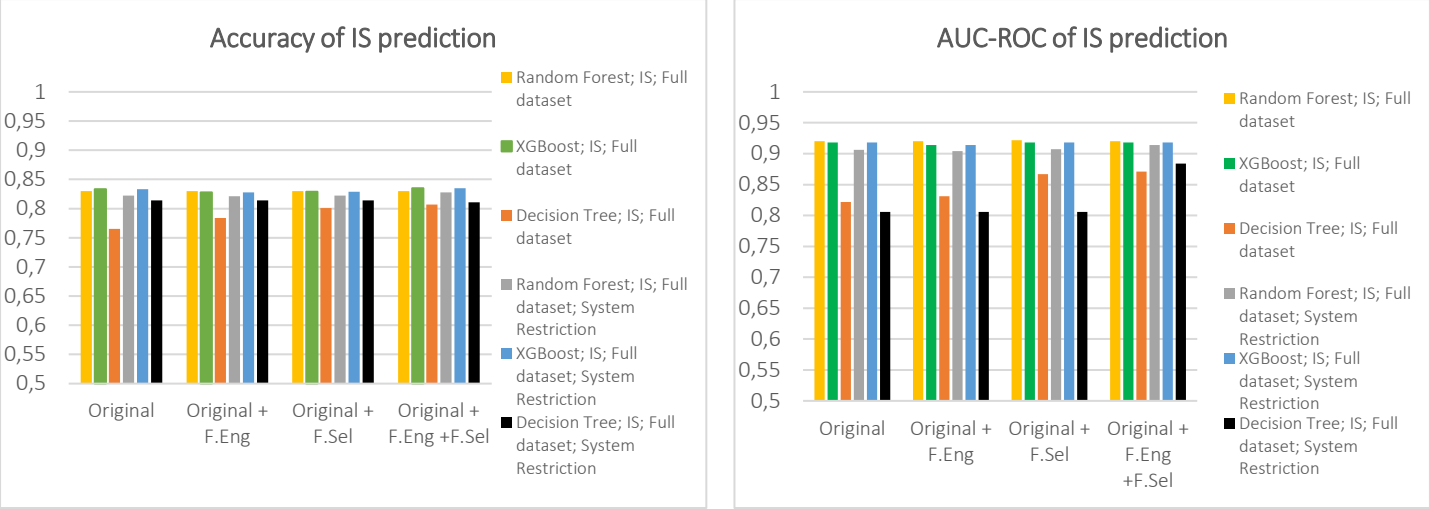


Figure 13: AUC-ROC and accuracy scores obtained when training Random Forest algorithm, XGBoost and Decision Tree with PanCancer dataset, applying Feature Selection and/or Feature Engineering techniques to predict IS.

4.2.2.2 Multi-class and multilabel classification

The evaluation scores obtained for the multiclass and multilabel problems were significantly lower than the scores obtained in the binary classification problems. In the multilabel problem Random Forest achieved values of 68% and 75% for accuracy and PR-AUC respectively, using the original dataset (Figure 15**Figure *15*:** PR-AUC and accuracy scores obtained when training Random Forest algorithm, XGBoost and Decision Tree with PanCancer dataset, applying Feature Selection and/or Feature Engineering techniques to predict multilabel.). The algorithm with better scores in the multiclass scenario was XGBoost (Figure 14), with scoring values of 65% for accuracy and 51% for PR-AUC, using the original dataset.

As expected, by restricting the range of hyperparameters tested by the classifiers, most of the scores obtained decreased. The best score with restricted conditions was obtained with the Random Forest algorithm applied to the multilabel problem in the original dataset, reaching values of 63% for accuracy and 71% for PR-AUC, but nonetheless the XGBoost outperformed the Random Forest in the multiclass scenario, reaching values of 62% and 48% in accuracy and PR-ROC.

In the multilabel scenario, using the unrestricted models, both Random Forest algorithm and Decision Tree had high relative standard deviation results across the 4 datasets, with approximately 4% in both accuracy and PR-AUC.
On the one hand, Random Forest algorithm decreased in both accuracy and PR-AUC when applying Feature Selection techniques (alone or together with Feature Engineering), decreasing from 68% in accuracy and 75% in PR-AUC to 61% and 68% respectively. This was also observed in Random Forest with the restricted system decreasing from 63% in accuracy and 71% in PR-AUC to 58% and 65%, respectively.

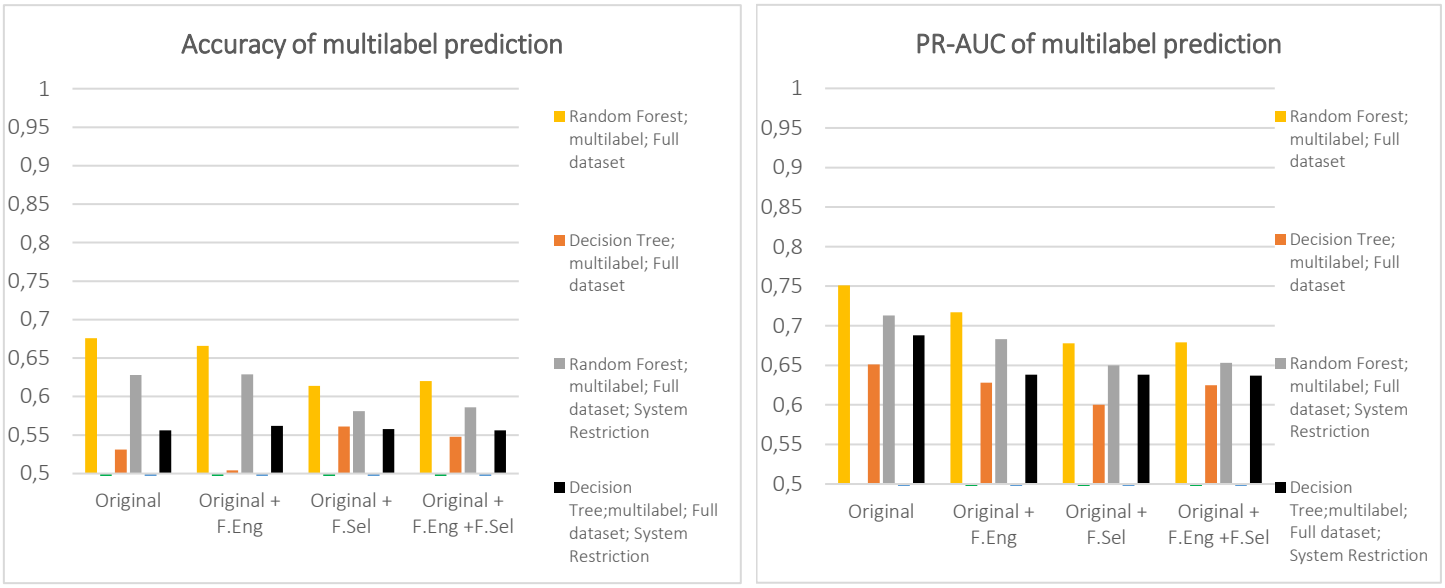


Figure 15: PR-AUC and accuracy scores obtained when training Random Forest algorithm, XGBoost and Decision Tree with PanCancer dataset, applying Feature Selection and/or Feature Engineering techniques to predict multilabel.

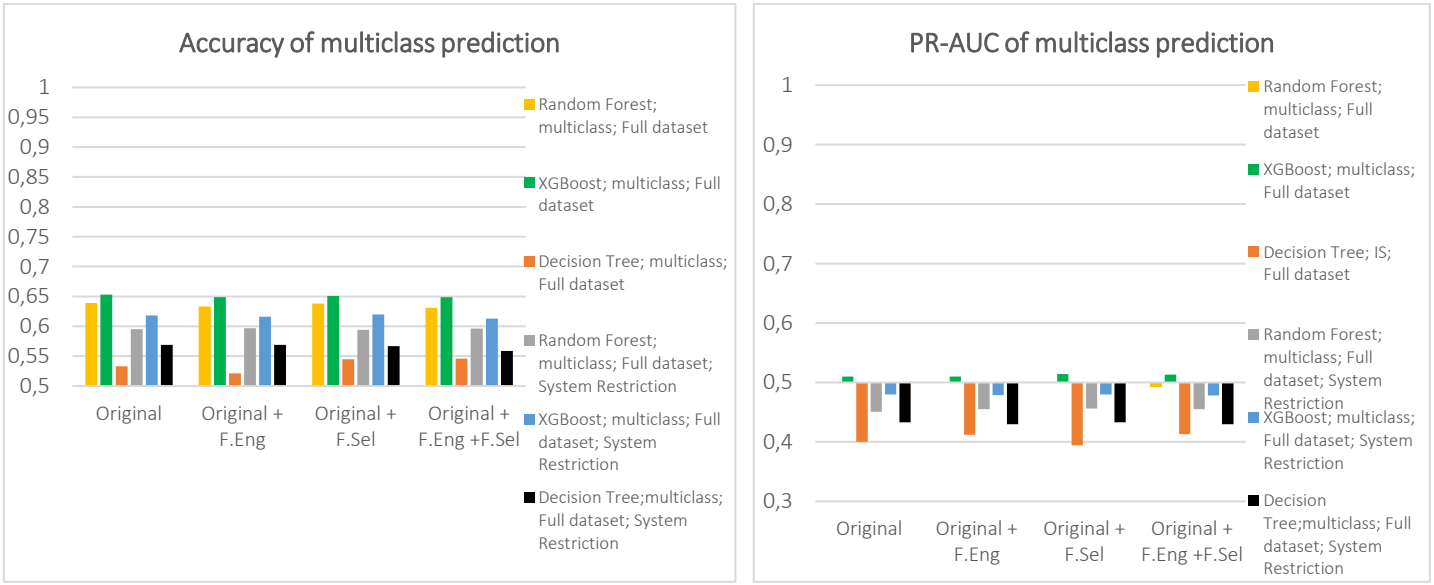


Figure 14: PR-AUC and accuracy scores obtained when training Random Forest algorithm, XGBoost and Decision Tree with PanCancer dataset, applying Feature Selection and/or Feature Engineering techniques to predict multiclass.

On the other hand, Decision Tree was benefited from the feature selection technique, increasing from 53% to 56% in accuracy although decreasing from 65% to 63% in PR-AUC. A similar observation was made for the Decision Tree in the multiclass scenario, but less significantly, increasing from 53% to 55% in accuracy. In the unrestricted model using Decision Tree algorithm, there was no changes observed across the datasets.

4.2.3 BRCA population

4.2.3.1 Binary classification

For the full BRCA population, comprising 1026 patients, the best results were obtained when training a Random Forest to predict the "CA" class and XGBoost to predict the "IS" class. For the former the model achieved values of 72% in accuracy and 78% in AUC-ROC, and the latter reached 90% in accuracy and 95% in AUC-ROC, with unrestricted hyperparameters. With the restricted models the best algorithms remained the same, but the scores decreased to 69% in accuracy and 74% in AUC-ROC using the Random Forest to predict the "CA" class. The scores obtained for the "IS" remained the same.

Across the feature engineering and selection techniques the results were identical to the already observed for the PanCancer population. The algorithm which obtained higher values of relative standard deviation across the 4 combinations of techniques was again the Decision Tree with 3% and 5% for accuracy and AUC-ROC, respectively, when predicting the "IS" class in unrestricted parameters condition, while no significant deviations were observed under restricted conditions.

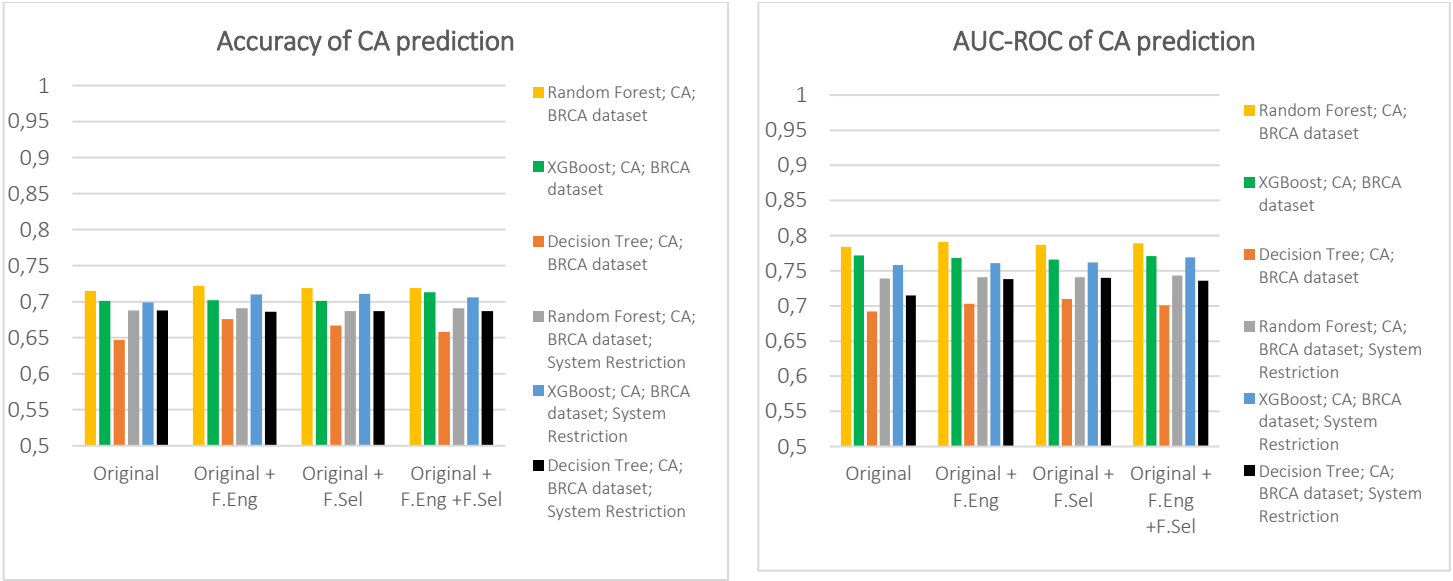


Figure 16: AUC-ROC and accuracy scores obtained when training Random Forest algorithm, XGBoost and Decision Tree with BRCA dataset, applying Feature Selection and/or Feature Engineering techniques to predict CA.

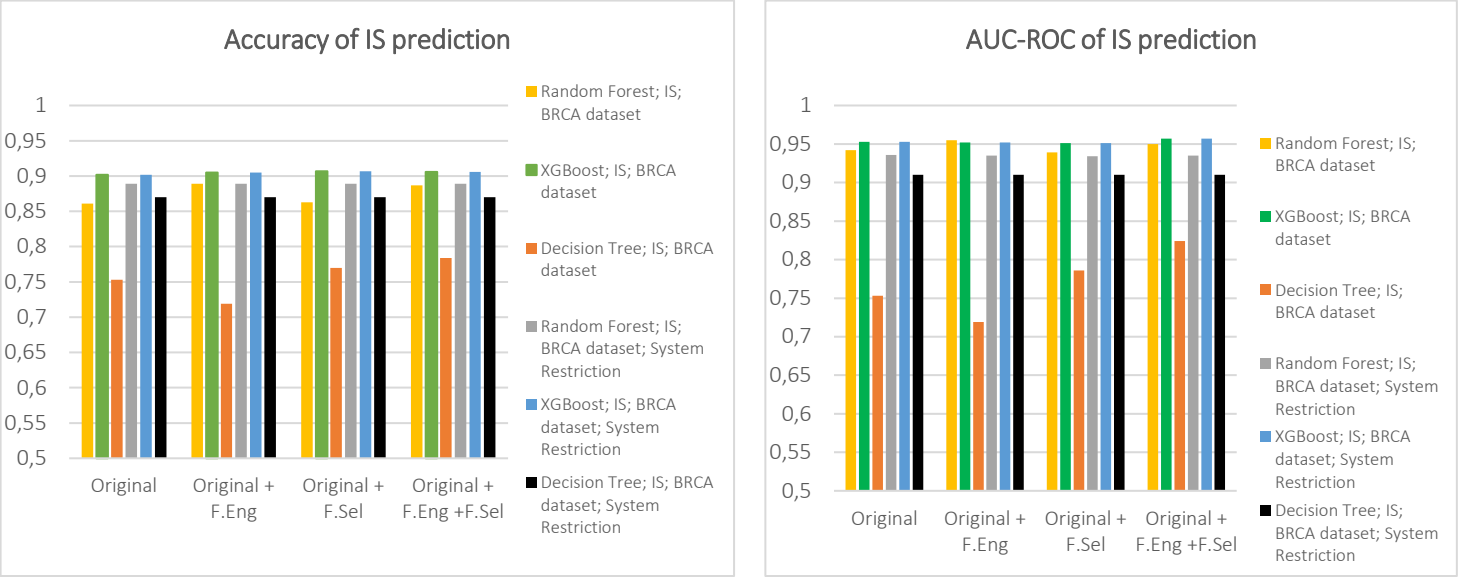


Figure 17: AUC-ROC and accuracy scores obtained when training Random Forest algorithm, XGBoost and Decision Tree with BRCA dataset, applying Feature Selection and/or Feature Engineering techniques to predict IS.

4.2.3.2 Multi-class and multilabel classification

For BRCA population, analogous to the observed in PanCancer population, the evaluation scores obtained for the multiclass and multilabel problems were significantly lower than the scores obtained in the binary classification problems. In this population the Decision Tree accuracy scores obtained in the multiclass scenario were lower than 50%, which is why they are not present in Figure 19.

Under unrestricted conditions, in the multilabel problem, Random Forest achieved values of 62% and 68% in accuracy and PR-AUC respectively, using the dataset with feature engineering and feature selection applied (Figure 18). The algorithm with better scores, in the same unrestricted conditions, in the multiclass scenario was also Random Forest with 62% for accuracy and 48% for PR-AUC, using the dataset with feature engineering applied (Figure 19).

Indeed, high relative standard deviation values were observed for both Random Forest and Decision Tree in multiclass and multilabel scenarios, under unrestricted conditions, which explains why the original dataset didn't achieve the best results. Nonetheless, by restricting the range of hyperparameters, those values lowered to less than 1%, meaning that neither feature selection nor feature engineering changed the models' scores in the restricted conditions.

Again, as expected, in the restricted conditions most of the models' scores decreased, compared to the scores with unrestricted conditions. The best scores with restricted conditions was obtained with Random Forest using the original dataset in the multilabel scenario and achieved 58% and 74% in accuracy and PR-AUC. In the multiclass scenario, XGBoost was the best with the original dataset, reaching values of 60% and 45% in accuracy and PR-AUC.

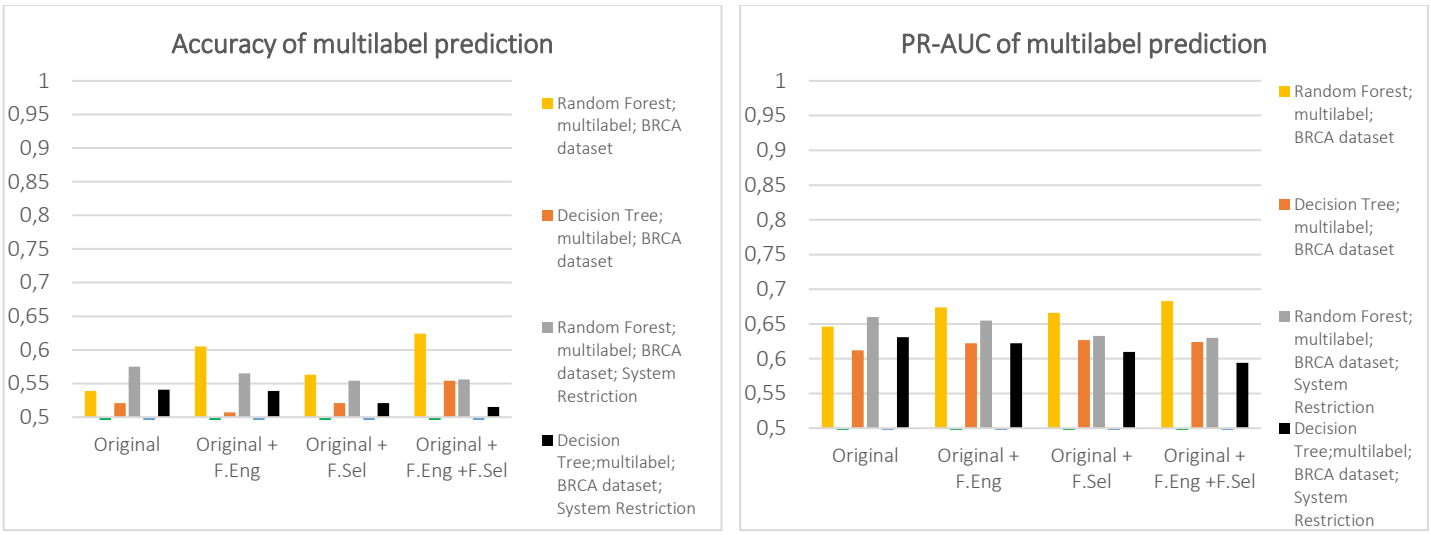


Figure 18: PR-AUC and accuracy scores obtained when training Random Forest algorithm, XGBoost and Decision Tree with BRCA dataset, applying Feature Selection and/or Feature Engineering techniques to predict multilabel.

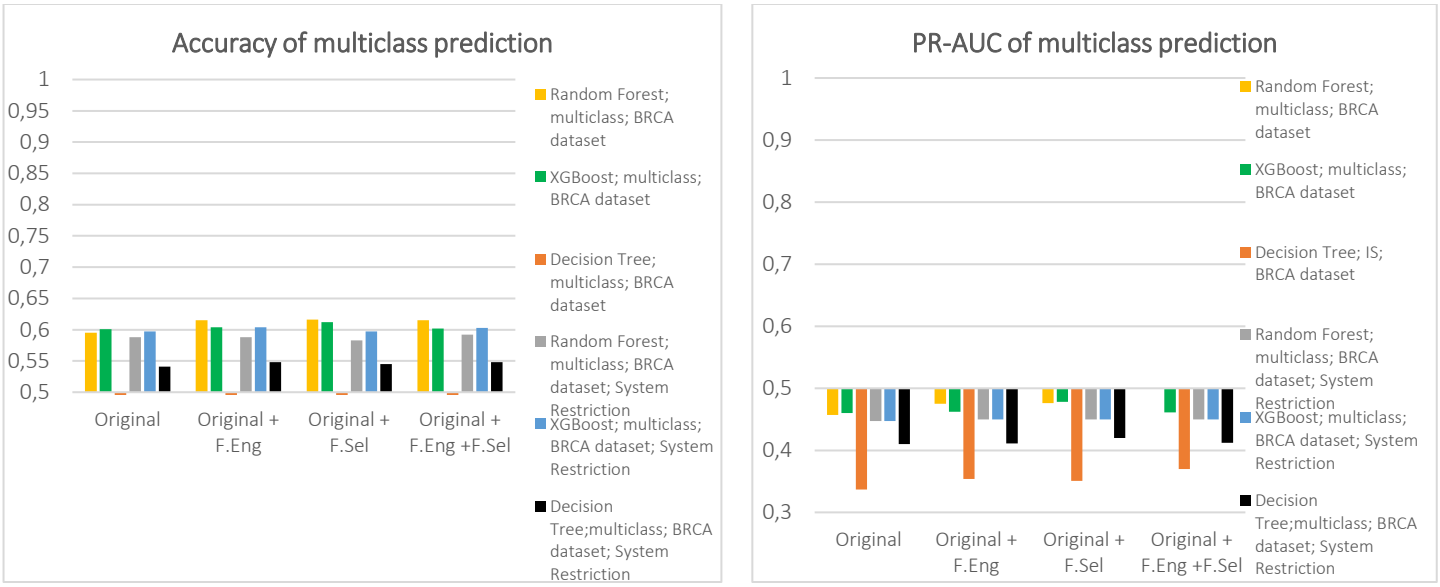


Figure 19: PR-AUC and accuracy scores obtained when training Random Forest algorithm, XGBoost and Decision Tree with BRCA dataset, applying Feature Selection and/or Feature Engineering techniques to predict multiclass.

4.2.4 LUAD population

4.2.4.1 Binary classification

For the LUAD population, comprising 498 patients, the best results were obtained when training a Random Forest for prediction of "IS" class, achieving results of 100% in both accuracy and AUC-ROC with the original dataset. The results obtained for prediction of the class "CA" were significantly lower: 70% for accuracy and 73% for AUC-ROC using Random Forest applying feature engineering techniques and with unrestricted conditions. With the system under restricted hyperparameters, the best model was XGBoost in both "IS" and "CA" prediction. It achieved 96% and 99% in accuracy and AUC-ROC respectively, for the prediction of "IS" using the original dataset, and 69% and

75% for the prediction of "CA", applying feature selection and feature engineering techniques to the dataset. Therefore, in the restricted scenario (the actual conditions implemented in the system) the XGBoost algorithm outperforms the Random Forest algorithm when predicting both classes in the LUAD population.

Across the feature engineering and selection techniques the algorithm which obtained higher values of relative standard deviation across the 4 combinations of techniques was Decision Tree with 2% and 3% when predicting the "CA" class in unrestricted conditions, but also Random Forest with 2% and 3% when predicting "CA" class in restricted conditions. Indeed, in restricted conditions, the evaluation scores of Random Forest increased when predicting the "CA" class after applying feature engineering techniques, comparing with the original dataset, but decreased after applying feature selection techniques. Nonetheless, XGBoost outperformed Random Forest with the original dataset.

Figure 20: AUC-ROC and accuracy scores obtained when training Random Forest algorithm, XGBoost and Decision Tree with LUAD dataset, applying Feature Selection and/or Feature Engineering techniques to predict CA.

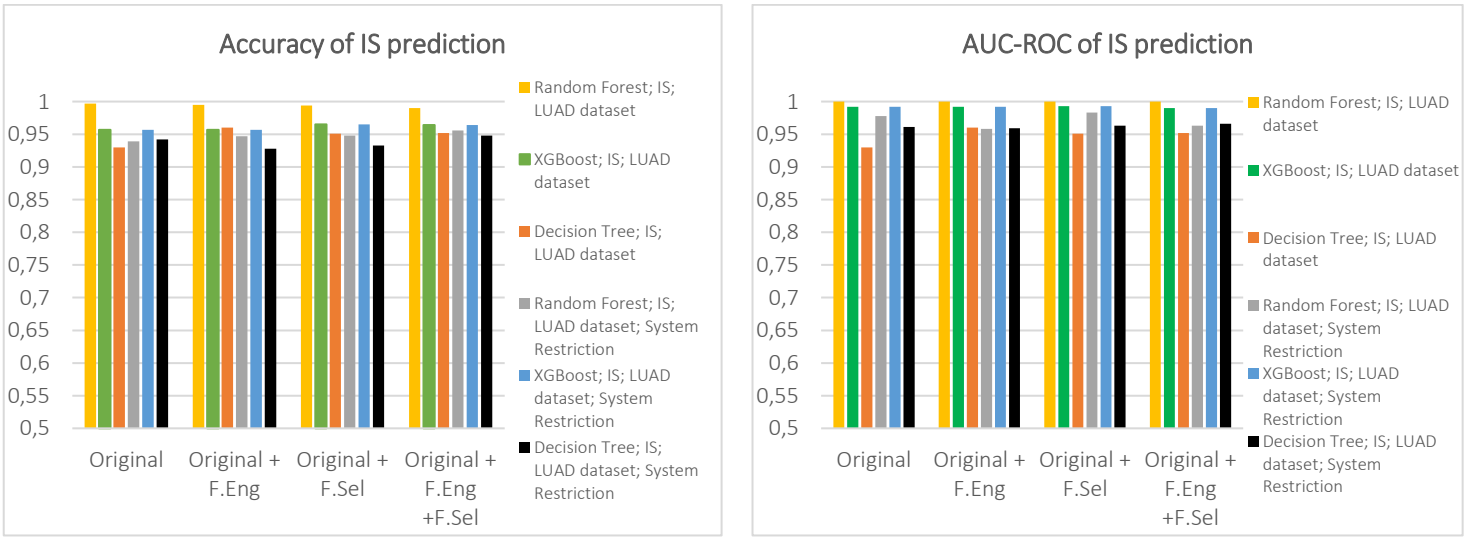


Figure 21: AUC-ROC and accuracy scores obtained when training Random Forest algorithm, XGBoost and Decision Tree with LUAD dataset, applying Feature Selection and/or Feature Engineering techniques to predict IS.

4.2.4.2 Multi-class and multilabel classification

Unlike the PanCancer and BRCA populations, using LUAD population in the multilabel scenario the scores obtained were higher than the observed in prediction of "CA" class. The best result was obtained by Random Forest after applying feature engineering techniques to the dataset, scoring 83% and 92% in accuracy and PR-AUC, in unrestricted conditions. In fact, the relative standard deviation results were not very high for any algorithm and prediction combination, but Random Forest had better evaluation scores when applying feature engineering techniques in multilabel prediction (as well as in multiclass prediction).

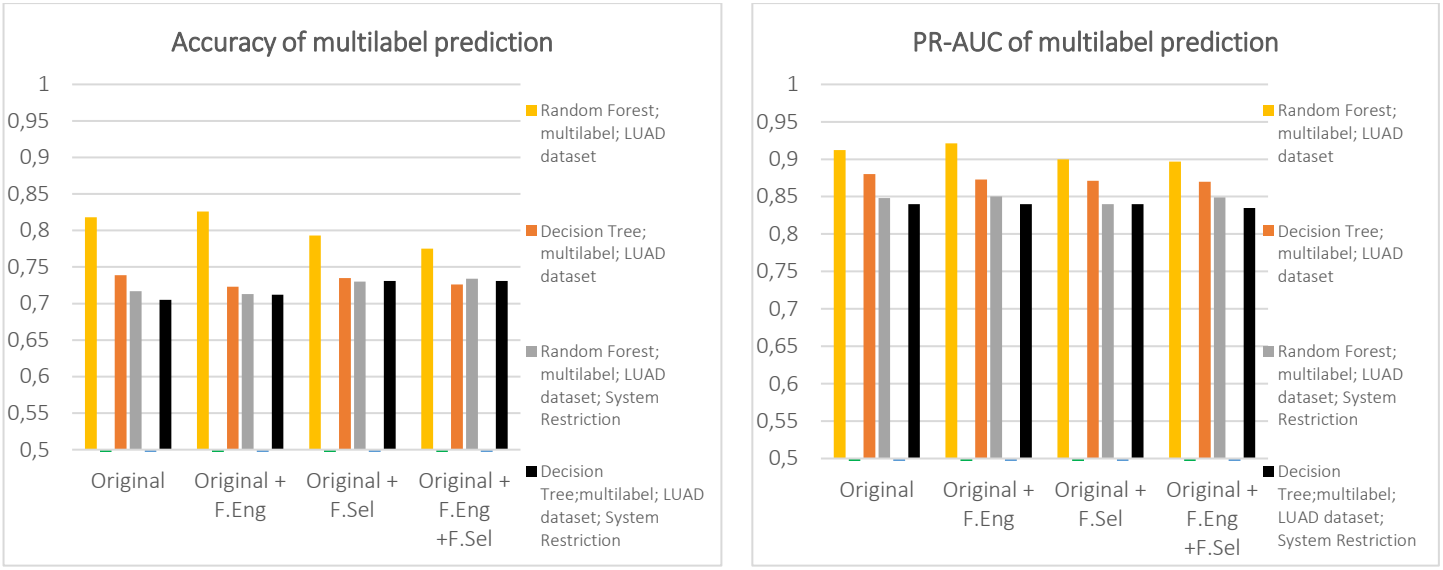


Figure 23: PR-AUC and accuracy scores obtained when training Random Forest algorithm, XGBoost and Decision Tree with LUAD dataset, applying Feature Selection and/or Feature Engineering techniques to predict multilabel.

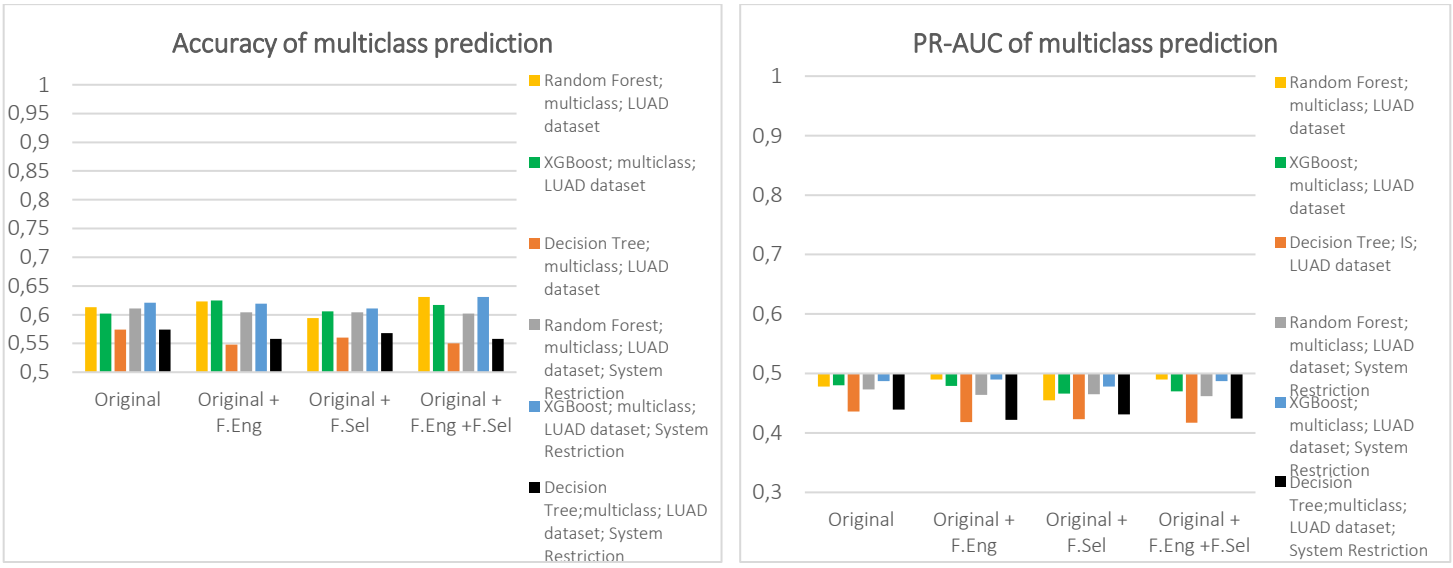


Figure 22: PR-AUC and accuracy scores obtained when training Random Forest algorithm, XGBoost and Decision Tree with LUAD dataset, applying Feature Selection and/or Feature Engineering techniques to predict multiclass.

Still in the multilabel scenario, by restricting the range of hyperparameters, feature selection improved the models' scores: Random Forest and Decision Tree achieved identical scores of 73% and 84% for accuracy and PR-AUC respectively. It's important to notice, however, that Decision Tree was only able to match the Random Forest's performance in restricted conditions when applying feature selection techniques to the dataset.

In the multiclass scenario XGBoost performed the best, achieving 63% in accuracy and 49% in PR-AUC, using a dataset with feature engineering and feature selection. Surprisingly, these results were obtained under restricted conditions. However, both Random Forest and Decision Tree achieved mostly worst results under restricted conditions, compared to the unrestricted conditions. There was no other observation concerning the contribution of features' techniques in the multiclass prediction.

4.3 Discussion

The study conducted in this chapter concerned the analysis and comparison of the scoring metrics obtained when training and cross-validating the three algorithms available in the system, under several combinations of user defined parameters, such as population, class and dataset processing techniques. The final goal is to determine the best feature selection and feature engineering parameters to set as default in the system (active or inactive), for future usage. Therefore, it is relevant to outline the advantages of such techniques in the algorithms with restricted hyperparameters (the implementation in the system).

With unrestricted parameters there were several observations of evaluation scores' improvements due to the application of feature selection and/or feature engineering. For instance, in the PanCancer population Decision Tree benefited from feature selection alone when predicting "IS" and in multilabel and multiclass scenarios. This was not observed in restricted parameters which means that for smaller trees (up until 6 levels) the application of feature selection alone does not benefit the performance of Decision Trees. One way that feature selection can improve the models' performance is by avoiding overfitting when the model chooses highly specific features in lower levels of the tree, which in smaller trees is not an issue.

However in the restricted conditions the improvements were scarce. Using both feature selection and feature engineering was beneficial in three situations: when training a Decision Tree to predict the class "IS" with the PanCancer population, when training XGBoost to predict the class "CA" in the LUAD population and when training XGBoost in the multiclass prediction using the LUAD population. The application of feature selection alone was beneficial when training a Decision Tree to predict the multilabel scenario with the LUAD population. The application of feature engineering alone was beneficial when training a Random Forest to predict the "CA" class with the LUAD population. Additionally, for both PanCancer and BRCA populations, Random Forest was observed having worst results in the multilabel scenario when feature selection was applied, both in the restricted and unrestricted models. Since activating the feature selection and feature engineering dataset processing is both computational and time expensive, and since most of the algorithms had no benefit with their activation under restricted conditions (and some even had worst results), both feature selection and feature engineering are deactivated by default in the system.

Ideally the parameters would be set depending on the algorithm used and population under study, but after discussing the system's usability with staff from the Cancer Bioinformatics group, at King's College London, it was agreed that having constant default values for the parameters was beneficial for the user experience and understanding of the system.

4.3.1 Original pre-processed dataset achieves highest performance

The pre-processing approaches used before the Feature Selection and Feature Engineering modules, proved to be good enough to achieve most of the highest values of accuracy and AUC-ROC (Original dataset in Figure 11). The application of the Feature Selection and Feature Engineering didn't add any relevant improvement to the performance metric's values. This is an indication that these modules may require a revision in a future work, or to be tested with other classes and/or other cancer types.

To further investigate the role of each individual feature selection and feature engineering technique, a set of 6 datasets was built, each applying only one technique at a time. A train-test split approach was used to evaluate the Random Forest algorithm when predicting the class "CA" in the PanCancer population with restricted hyperparameters. By using train-test split approach instead of cross-validation it is possible to extract the number of features and access the impact of each technique in that number. The results are available in Figure 24.

Figure 24: Accuracy and AUC-ROC of Random Forest predicting the "CA" class with the PanCancer population across several individual feature selection and feature engineering techniques. #feats represent the number of features used by the classifier, not the dataset's.

None of the individual techniques improved significantly the accuracy or AUC-ROC of the Random Forest. Regarding the number of features used, it was significantly lower after applying recursive feature extraction, but that did not reflect on the evaluation metrics. The application of the feature engineering techniques "Sum" and "Log" also produced models that used less features, indicating that the new features replaced the usage of many of the original features, but again that did not impact the evaluation metrics of the models.

In Random Forest the features used are only a sub-sample of the features available, so the model applies a feature selection step on its own. For this reason, the feature selection effect may be "diluted" when using this model.

Another possible explanation is the existence of highly correlated features, as observed before (Figure 8) in Fake predictors detection module. When a class is chosen, the features that highly correlate with it are removed, but the remaining features are kept. This results in groups of features correlated with each other but not with the chosen class and may result in collinearity issues. As explained previously in chapter 2, collinearity issues are not relevant for the training performance of tree-based algorithms, but the model itself is influenced. What happens is that highly correlated features supply the same information to the model, which reflects in the gini index (criteria for feature choice), and if feature A and B are correlated, and feature A is chosen to be in the tree, then feature B won't be chosen afterwards, even though they are in reality equally important. For this reason, feature selection may be removing feature B, and therefore no direct improvement is observed.

4.3.2 Restricted models achieve accuracy scores of 70%

As expected, in all populations (PanCancer, BRCA and LUAD) most of the evaluation scores obtained with the restricted models were lower than the results obtained with unrestricted models, for each algorithm. However, three exceptions were observed.

In binary classification of "IS" class, both in class PanCancer and BRCA, the results obtained were equal for Random Forest and XGBoost. In LUAD population this was only observed for Random Forest algorithm. These observations may be related to the presence of the feature "T.cells.regulatory..Tregs.", which is not highly correlated with the class "IS" (otherwise would be removed by the Fake predictors detection module), but has shown to be highly important in all models, and is present with or without restriction in tree 's maximum level or number of trees used. It is also possible that for the LUAD population the number of samples was too small to obtain high evaluation metrics, even with the presence of the feature "T.cells.regulatory..Tregs.", and therefore the XGBoost algorithm required more freedom in the maximum number of levels of each tree and number of trees to obtain higher scores.

The other two exceptions were observed for the BRCA population, when using the Decision Tree to predict the "IS" class and for the LUAD population when training the XGBoost in the multiclass scenario: the scores were higher with the restricted model than with the unrestricted model (Figure 17 and Figure 22).

In Decision Trees there is a higher chance of overfitting (comparing to ensemble models) since the model uses all records and considers all features to use, and therefore an optimum value is required in the minimum number of samples in each leaf (or minimum number of samples required to split a node) to replicate the pruning process which is not available. The Decision Tree observation can be explained by the way the optimization is done, as the

RandomizedSearchCV algorithm covers a range of values widely separated, and only afterwards GridSearchCV will run a range of values closer to each other. The difference between the restricted algorithms and the unrestricted is that during the RandomSearch the values given to the unrestricted system are min_samples_split = [2, 20, 40], min_samples_leaf = [2, 10, 20] and max_depth = [10, 100,500] among others, while the values given to the restricted system were min_samples_split = [2, 10, 20], min_samples_leaf = [2, 6, 10] and max_depth = [3,5] among others. In both cases the GridSearch then tests the best value +-2. For this reason, if the optimum value is, for instance, obtained with min_samples_split or min_samples_leaf equal to 6, the unrestricted model would never test this possibility. This logic would not be applied to max_depth as the unrestricted optimization will test smaller values for this parameter. Another possibility is the case when the unrestricted model finds a local minimum and scores higher results with min_samples_leaf = 20 than min_samples_leaf = 2 and if the optimum value was 4 then during the GridSearchCV that value would not be tested, even though in theory it could. With the restricted model the value min_samples_leaf=20 is not possible and therefore the model won't fall on the local minimum. Indeed, the models obtained using feature selection and engineering were:

Unrestricted:

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=98,
        max_features='auto', max_leaf_nodes=None,
        min_impurity_decrease=0.0, min_impurity_split=None,
        min_samples_leaf=2, min_samples_split=2,
        min_weight_fraction_leaf=0.0, presort=False, random_state=123,
        splitter='best')
```

Restricted:

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=3,
        max_features=None, max_leaf_nodes=None,
        min_impurity_decrease=0.0, min_impurity_split=None,
        min_samples_leaf=9, min_samples_split=2,
        min_weight_fraction_leaf=0.0, presort=False, random_state=123,
        splitter='best')
```

This shows that the unrestricted model is overfitting when choosing min_samples_leaf=2, while the restricted uses min_samples_leaf=9. It was probably caused by a local minima during RandomSearchCV which outputted 2 as being better than 10, while for the restricted system 2 was not chosen.

For the XGBoost observation, the only explanation relies on the n_estimators, since it was the only parameter restricted in the system: n_estimators: randint(100, 150) restricted to n_estimators: randint(1, 10). The number of estimators is in theory directly related with the increase in performance and not a source of overfitting in Random Forests [25], but this increases flexibility and complexity, and generalization performance of an XGBoost may suffer if the number of estimators is increased too much (i.e. test set performance may decrease). It has been observed that

an increase in error rate occurs for an increase in n_estimators after a certain value [58], especially in small samples, where outliers may have more impact and where the distribution is not as smooth as it is in a bigger sample. This would explain the higher values in the restricted models for the LUAD population, which can achieve n_estimators between 1 and 10, while in the unrestricted model the n_estimators is between 100 and 150. Indeed, the models were:

Unrestricted:

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
    colsample_bytree=0.7042431416791128, gamma=0.42382349761173876,
    learning_rate=0.3065759584886718, max_delta_step=0, max_depth=5,
    min_child_weight=1, missing=None, n_estimators=113, n_jobs=1,
    nthread=None, objective='multi:softprob', random_state=0,
    reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
    silent=True, subsample=0.7072084467138464)
```

Restricted:

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
    colsample_bytree=0.9813005070371787, gamma=0.45400554183152003,
    learning_rate=0.13463919482609732, max_delta_step=0, max_depth=5,
    min_child_weight=1, missing=None, n_estimators=8, n_jobs=1,
    nthread=None, objective='multi:softprob', random_state=0,
    reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
    silent=True, subsample=0.7095368846612303)
```

The number of estimators was indeed much lower in the restricted system (8 compared to 113), but also the learning rate, which indicates that the training will progress very slowly as the model is making very small updates to the weights of each record. However, if the learning rate is set too high, it can cause undesirable divergent behaviour in the loss function and that may be the cause of low performance of the unrestricted model. The parameter colsample_bytree is the subsample ratio of columns when constructing each tree, which occurs once for every tree constructed. It appears that the restricted system uses almost all features when constructing each tree, and that improves performance.

Overall, by restricting the algorithms parameters, the accuracy scores lowered by 0,3 on average, by algorithm/population/prediction, and by 0,1 for AUC scores. These values were higher for Decision Trees but were lower for XGBoost. In all populations, the binary classification achieved around 70% for both "CA" and "IS" prediction, and some algorithms achieved higher values.

4.3.3 Binary classification achieves higher scores than multilabel and multiclass prediction

The results using the multilabel and multiclass scenarios were overall worse than in the binary classification. Even though a 4-class problem with non-incompatible immune-evasion events would be the closest to the biological problem, it is in fact a problem harder to model than each individual immune-evasion binary classification.

It is interesting to notice that in the multilabel scenario both Random Forest and Decision Tree achieved better scoring results than in the multiclass (it is not possible to compare XGBoost). It is possibly related to how the algorithms handle the numerical input given in the multiclass, instead of the tuple of target variables given in the multilabel problem, even though they represent the same 4-classes problem. In the multilabel implementation the accuracy and PR-AUC are calculated based on the outputted vector of size 2 (each of the classes), so for a patient having both immune-evasion mechanisms the correct class would be [0,1], and a wrong prediction of [1,1] would be closer than a wrong prediction of [1,0] and would contribute differently to the metrics. In the multiclass scenario, the labels are all equally distant, so predicting correctly only one of the events is as wrong as predicting none correctly.

The exception of the bad results achieved with multilabel and multiclass scenarios were the results obtained with the LUAD population. This was unexpected because being a smaller population (498 patients), it would be expected that the training sample wouldn't be large enough to be representative, and therefore the metric results obtained with the test set would be worse than those obtained with larger populations. It may be an indication that the LUAD population is very homogeneous or has a small number of outliers.

The number of samples of each class is in Table 3 (nomenclature used in multiclass, but equivalent in multilabel vectors). It is possible to conclude that in the LUAD population there is a highly imbalanced class which is "Neoantigens depletion" and therefore the dataset is dominated by the classes "Neoantigens depletion & IS abundance" and "IS abundance" (these classes have no instances in common).

To further analyse the multiclass and multilabel predictions, the precision and recall scores were analysed for the three populations (see figures below), including error bars with the standard deviation of the values across the 4 classes. To obtain the error bars, the recall and precision values were calculated individually for each class, but the precision and recall represented in the plot bars were calculated with the micro average approach (as explained previously in chapter *Training and Validation of models)*, and consequently the precision values is equal to the recall values in the multiclass scenario. If there is a false positive, there will always also be a false negative and vice versa, because the algorithm always predicts one of the classes. If class A is predicted and the true label is B, then there is

Table 3: Number of instances in each class, using multiclass nomenclature, across PanCancer, BRCA and LUAD populations.

	PanCancer	**BRCA**	**LUAD**
Neoantigens depletion & IS abundance	2471	311	230
IS abundance	2455	310	212
Neoantigens depletion	2451	220	20
None	1381	180	31

a FP for A and a FN for B. If the prediction is correct, i.e. class A is predicted and A is also the true label, then there is neither a false positive nor a false negative but only a true positive [59]. Therefore, there is no possible scenario that would increase only FP or FN but not both, which is the only parameter that distinguishes precision from recall: $Precision = \frac{TP}{TP+FP}, Recall = \frac{TP}{TP+FN}$.

It is possible to observe that in all populations, the precision values have smaller error bars than the recall values, which indicates more heterogeneity in recall measures. Recall can be interpreted as the amount of positive records that were correctly classified as positive. This means that the classifier can predict some classes with a low number of false negatives (for instance, predicts almost all instances that are truly labelled A as being A, but also predicts the label A for many instances that are truly B), but to do so the remaining classes will have a high number of false negatives (for instance B in the previous example). Therefore, the recall can vary strongly across classes.

The results reinforce previous observations, as Decision Tree scores the worst in both precision and recall, both in multiclass and multilabel predictions. The restricted conditions also decreased the models scores, compared with the unrestricted conditions. It is also possible to observe that feature techniques had no influence in the error bars in each graph, but had in the precision and recall values, decreasing both precision and recall in all populations. Additionally, the algorithms didn't show any association with the error bars size either.

Comparing multiclass and multilabel scenarios, indeed the multilabel scenario presents higher values of precision and recall, as expected by the higher values of accuracy and AUC previously observed.

Analysing the precision and recall values across populations it's possible to observe that the PanCancer population achieves the best values in multiclass scenario, followed by BRCA and LUAD together, with similar values, while the LUAD population achieves the best results in multilabel scenario with values around 90%, followed by PanCancer population with scores around 80% and finally BRCA with scores around 70%. That was also the order of the accuracy scores.

Across populations the error bars were significantly bigger in the LUAD population for both multiclass and multilabel scenarios, and in the BRCA population in the multilabel scenario. This reflects the high variability in precision and recall across the 4 classes, when using these small populations, which is a product of the high imbalance across classes.

Overall, the multilabel prediction in LUAD achieved the highest values for both precision and recall, but with a very broad error bar, which reflects an algorithm biased to one class or a sub-set of the classes in both multiclass and multilabel scenario. However, in the multiclass scenario the error bar is also very broad but the precision and recall values are not very high. The difference observed of precision and recall values between multiclass and multilabel may be explained by the difference between the multiclass and multilabel implementation explained previously, which reflects on the algorithm's performance metrics calculation. If the algorithm is biased to classify all instances as belonging to the class "Neoantigens depletion & IS abundance", it will be predicting correctly 230 instances, while being "half-correct" for the 212 instances belonging to class "IS abundance" and the 30 instances belonging to

Figure 25: Precision and recall scores obtained when training Random Forest algorithm, XGBoost and Decision Tree with PanCancer population, applying Feature Selection and/or Feature Engineering techniques to predict multiclass.

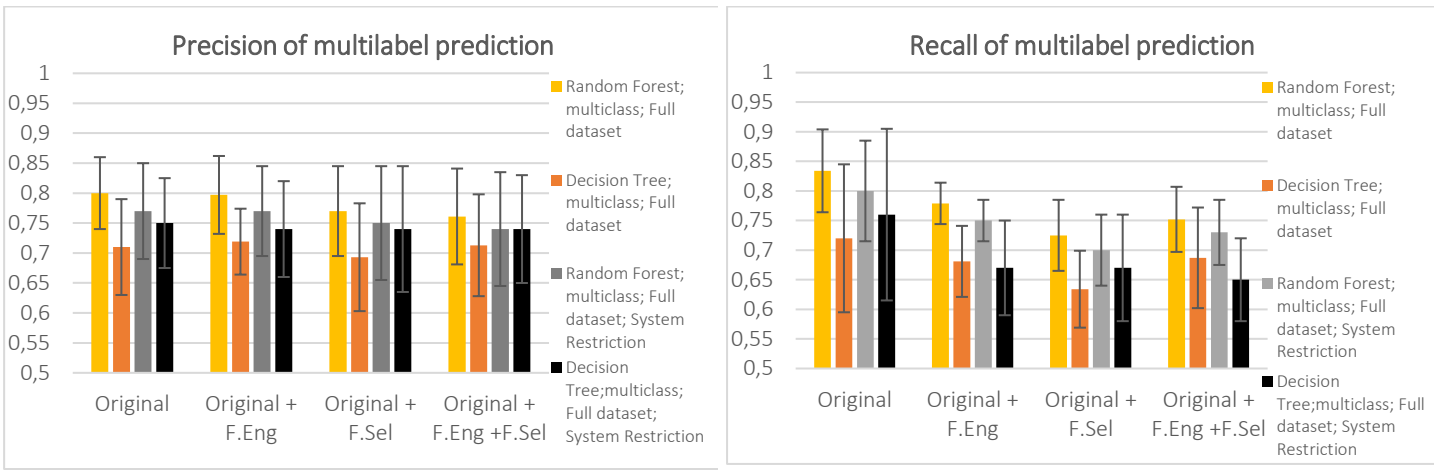


Figure 26: Precision and recall scores obtained when training Random Forest algorithm, XGBoost and Decision Tree with PanCancer population, applying Feature Selection and/or Feature Engineering techniques to predict multilabel.

"Neoantigens depletion". In the multiclass scenario all the instances that are not truly from the "Neoantigens depletion & IS abundance" class would be considered wrongly labelled.

Given the observed results, it is possible to conclude that the multiclass scenario is the best approach, compared to the multilabel scenario, since it is less susceptible to suffer from the consequences of imbalancements in one or more of the 4 classes. Although in the multilabel scenario, the algorithms achieved better results with all populations, it reflects a model that was built using information biased towards a dominant class and will therefore not be very informative.

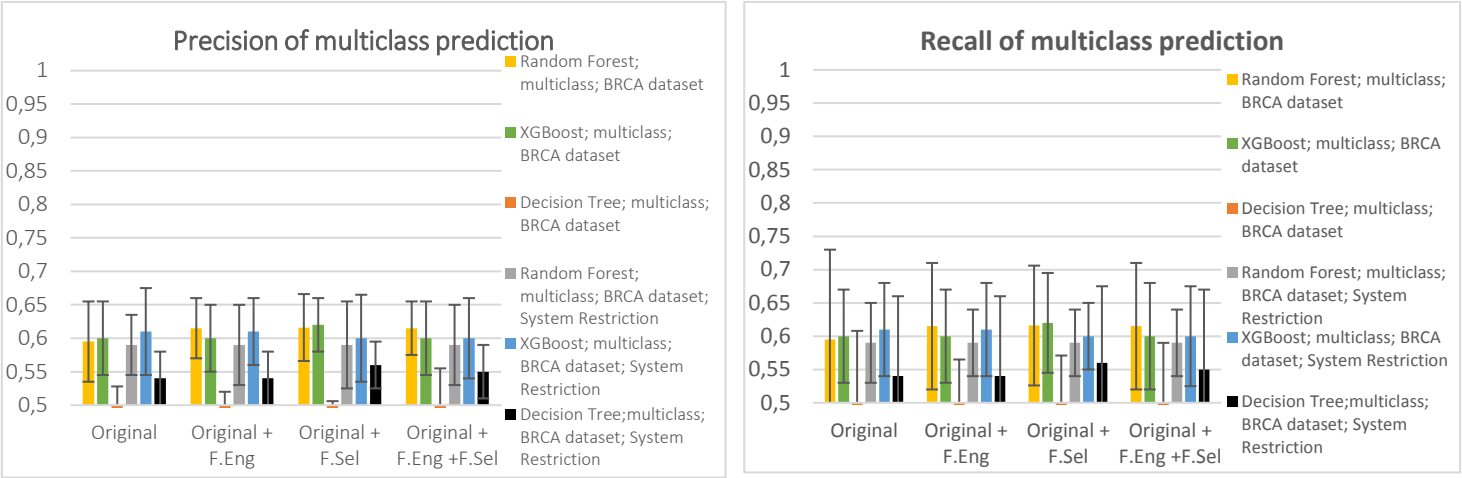


Figure 27: Precision and recall scores obtained when training Random Forest algorithm, XGBoost and Decision Tree with BRCA dataset, applying Feature Selection and/or Feature Engineering techniques to predict multiclass.

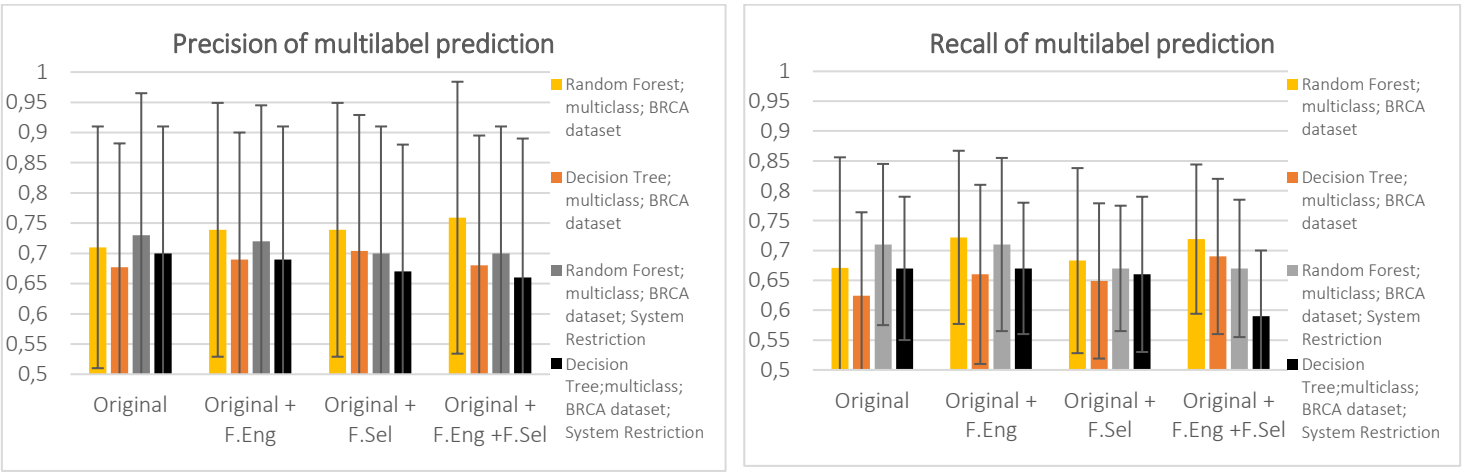


Figure 28: Precision and recall scores obtained when training Random Forest algorithm, XGBoost and Decision Tree with BRCA dataset, applying Feature Selection and/or Feature Engineering techniques to predict multilabel.

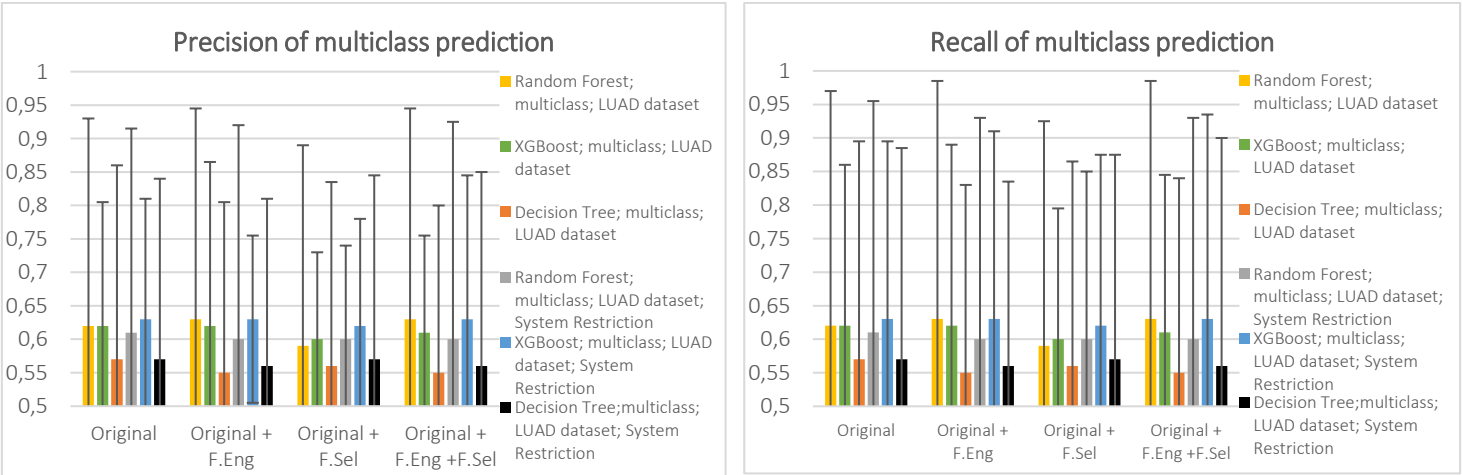


Figure 29: Precision and recall scores obtained when training Random Forest algorithm, XGBoost and Decision Tree with LUAD dataset, applying Feature Selection and/or Feature Engineering techniques to predict multiclass. The bottom scores of the error bars are not visible as the graph scale doesn't start at value zero.

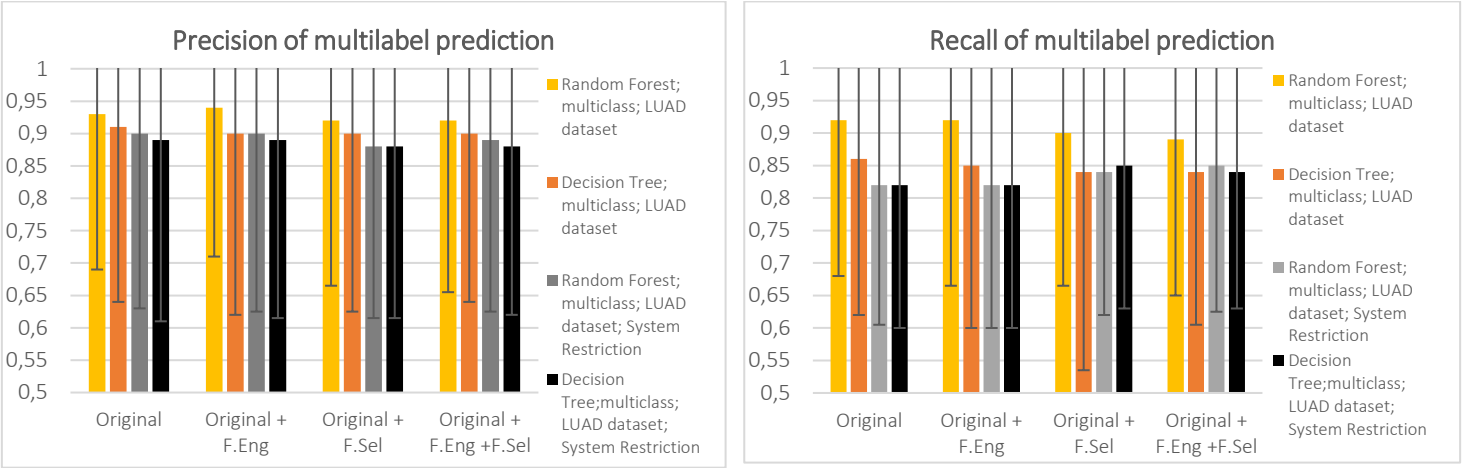


Figure 30: Precision and recall scores obtained when training Random Forest algorithm, XGBoost and Decision Tree with LUAD dataset, applying Feature Selection and/or Feature Engineering techniques to predict multilabel. The error bars show a top value higher than the scale maximum, which is a side product of using the mean of a micro average scheme and the standard deviation of individual scores.

4.3.4 IS prediction achieves better results than CA prediction

Given that the "IS" class is obtained with a previously tested and approved method [42], it was expected that this was a well-defined classification problem, as opposed to the prediction of the "CA" class, which is automatically binarized in the population (dividing the population in half according to the values in the "Neoantigens_total" column). It is possible that the automatic division of the population is not the best approach and should be revised.

Nonetheless, it is possible that the prediction of "IS" was only higher due to the presence of the highly important feature "T.cells.regulatory..Tregs.", as previously mentioned.

4.3.5 PanCancer population achieves better results than BRCA and LUAD populations

There was a decrease in both accuracy and AUC-ROC, for the prediction of both classes "CA" and "IS", when using the dataset BRCA or LUAD, compared to the PanCancer population results. The dataset filtered for BRCA patients has 1023 rows, which is approximately 10% of the full dataset, while the LUAD dataset is roughly 5% of the PanCancer population. The classifiers chosen are ensemble approaches that use as estimators Decision Tree classifiers, which are algorithms that rely on the size of the dataset to achieve good performance metrics. When faced with a smaller dataset these algorithms may overfit and lose the ability to generalize their prediction power, therefore having worst scoring results. To avoid overfitting problems the models created with the BRCA and LUAD datasets have to be simpler (lower number of nodes in each tree), which explains the overall lower performance obtained, compared to the classifiers trained with the PanCancer population.

Additionally, when filtering the PanImmune dataset by cancer type, the system adds a set of new features specific of that cancer type, which may interfere with the performance of the models. By having too many features it is possible

that the algorithms don't choose the most important ones in each node of the tree, since the feature's choice is done from a random sub-set of features from the dataset.

Nonetheless the values obtained are all close to 70% or higher (both accuracy and AUC), and the feature's importance together with the tree model obtained with the Decision Tree algorithm are of high relevance to understand the immune-evasion mechanisms in the specific case of BRCA and LUAD patients.

4.3.6 XGBoost and Random Forest outperform Decision Tree

As expected, the ensemble classifiers outperformed the Decision Tree classifier in all combinations of class/feature techniques/population of interest. Nonetheless, the results achieved by Decision tree classifier were very satisfactory (most of the results are higher than 70% for accuracy and AUC).

Given that the Decision Tree algorithm outperforms the remaining algorithms in interpretability by the user, and speed of training and validation, this algorithm was chosen as the default classifier for the system. By doing this the scores are lowered in 0.5 for accuracy and 0.3 for AUC on average, compared with the algorithm that achieved best scores in each population/class combination, under restricted conditions.

5 Case studies

To demonstrate the application of the AutoTCGA, two case studies were performed, using different datasets, populations and chosen classes to be predicted.

5.1 IS prediction in PanCancer patients with sub-set of original PanImmune dataset

The prediction of class "IS" (Immunosuppressors) is an important case study as this class represents a possible immune-evasion event: the enrichment of "IS" in the tumour is an indication of tumour immune-evasion events, while their depletion is an indication of the high immune system activity near the tumour. By training a tree-based classifier to predict the enrichment/depletion of "IS" in the PanCancer population, it is possible to gain insight into what distinguishes one group from another and support a tailored immunotherapy treatment.

For a sub-set of features in the PanImmune dataset, under the application of a Decision Tree to predict the IS class, and without any feature selection or feature engineering techniques applied, the accuracy score obtained was 81% and AUC-ROC of 88%. The tree structure obtained has 61 nodes with 5 levels. In this tree there are 15 features selected as important, which are represented below (Figure 31) as well as the model (Figure 32) and the 5 most important features distribution (Figure 33). A screenshot of the tool is also represented below (Figure 34 and Figure 35).

From the model we can extract 3 rules to predict the abundance of Immunosuppressor cells in cancer patients:

1. totTCRb_reads > 10.5 & purity <= 0.725 & Study Abbreviation NOT 'GBM' , gini = 0.084, samples = 2362, value=[104, 2258], class=1
2. 4.5< totTCRb_reads <= 10.5 & purity <= 0.725 & Macropahges.M2 <=0.392 & MutationsNonSilent > 1.067, gini=0.214, samples = 1122, value = [137, 985], class=1
3. totTCRb_reads > 10.5 & purity > 0.725 & T.cells.CD8 > 0.058 & Macrophages.M2 <= 0.37, gini = 0.223, samples= 359, value=846, 313], class=1

These rules were extracted from nodes with a gini index lower than 0.3 (high homogeneity) and number of samples higher than 100. The inverse rules describe the population with value 0 for class IS, which means they are depleted of Immunosupressors.

Some of the rules were expected, and serve as control, as is the case of rule 2, which states that a population with low purity and low levels of Macropahges.M2 is abundant in immunosuppressors. This result makes sense as the purity is a measure of the amount of tumour cells in the sample, and a low value possibly indicates that other cells are infiltrated in the tumour, like immunosuppressors. The high levels of Macrophages.M2 are an indication that beneficial immune cells of this type are in low amount, which favours the presence of other types of cells, like immunosuppressors. These results can be observed in the distributions of Macrophages.M2 and purity grouped by "IS" values (Figure 33). The remaining rules should be analysed by experts in the cancer biology field as they do not have a straightforward interpretation.

The sub-set of features was chosen based on the most relevant features in cancer immunology, and includes: 'AS', 'LOH_n_seg', 'LOH_frac_altered', 'MutationsSilent', 'MutationsNonSilent', 'Study Abbreviation', 'CNV_segs', 'CNV_frac', 'HRD', 'purity', 'numberOfNonSynonymousSNP', 'numberOfImmunogenicMutation', 'numberOfBindingExpressedPMHC', 'Indel_num', 'Immunogenic_indel_num', 'Neoantigen_num', 'B.cells.naive', 'B.cells.memory', 'Plasma.cells', 'T.cells.CD8', 'T.cells.CD4.naive', 'T.cells.CD4.memory.resting', 'T.cells.CD4.memory.activated', 'T.cells.follicular.helper', 'T.cells.regulatory..Tregs.', 'T.cells.gamma.delta', 'NK.cells.resting', 'NK.cells.activated', 'Monocytes', 'Macrophages.M0', 'Macrophages.M1', 'Macrophages.M2', 'Dendritic.cells.resting', 'Dendritic.cells.activated', 'Mast.cells.resting', 'Mast.cells.activated', 'Eosinophils', 'Neutrophils', 'totTCRa_reads', 'totTCRb_reads'.

```
Features Importance
Non-zero importance features listed:  15
                                importance
totTCRb_reads                     0.752567
purity                            0.096303
totTCRa_reads                     0.049958
Macrophages.M2                    0.020541
T.cells.CD8                       0.019106
MutationsNonSilent                0.016785
T.cells.regulatory..Tregs.        0.015615
MutationsSilent                   0.006071
numberOfImmunogenicMutation       0.005029
Virus_HHV                         0.004592
Study Abbreviation_GBM            0.004420
Study Abbreviation_LUAD           0.004311
Monocytes                         0.002932
ajcc_pathologic_tumor_stage_NA    0.001728
Virus_HCV                         0.000043
```

Figure 31: Feature Importance list of model trained with original PanImmune dataset, to predict IS class.

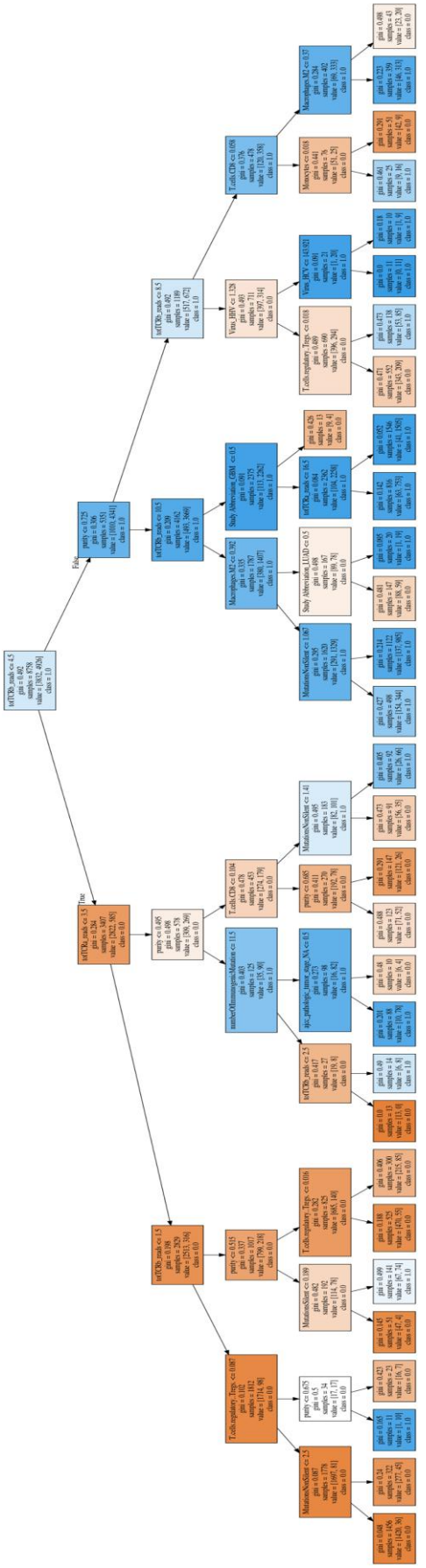


Figure 32: Decision Tree model trained with original PanImmune dataset, to predict IS class.

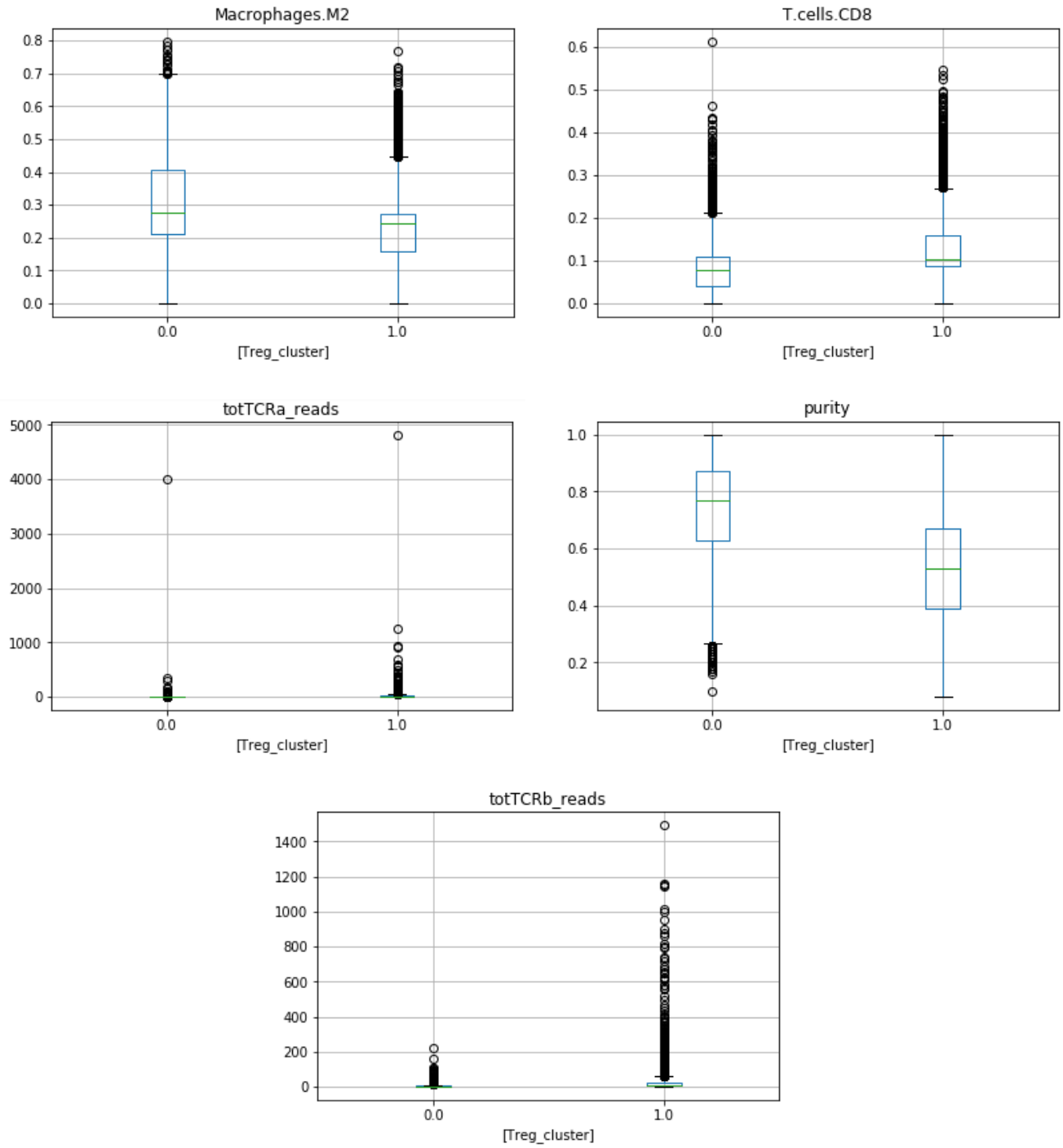


Figure 33: Distribution of 5 most important features of model trained with original PanImmune dataset, to predict IS class.

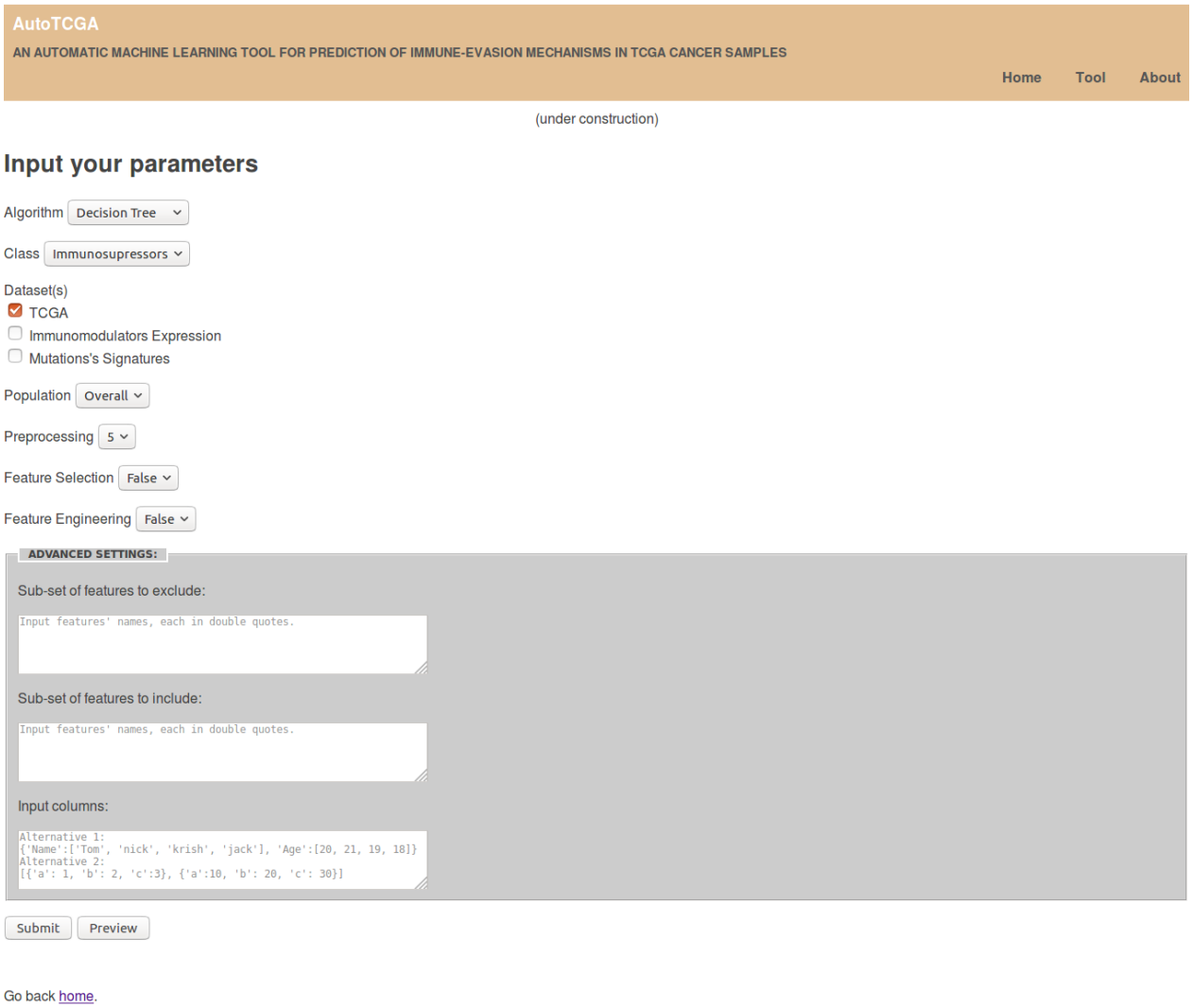


Figure 34: AutoTCGA web-app parameters input window for case study 1.

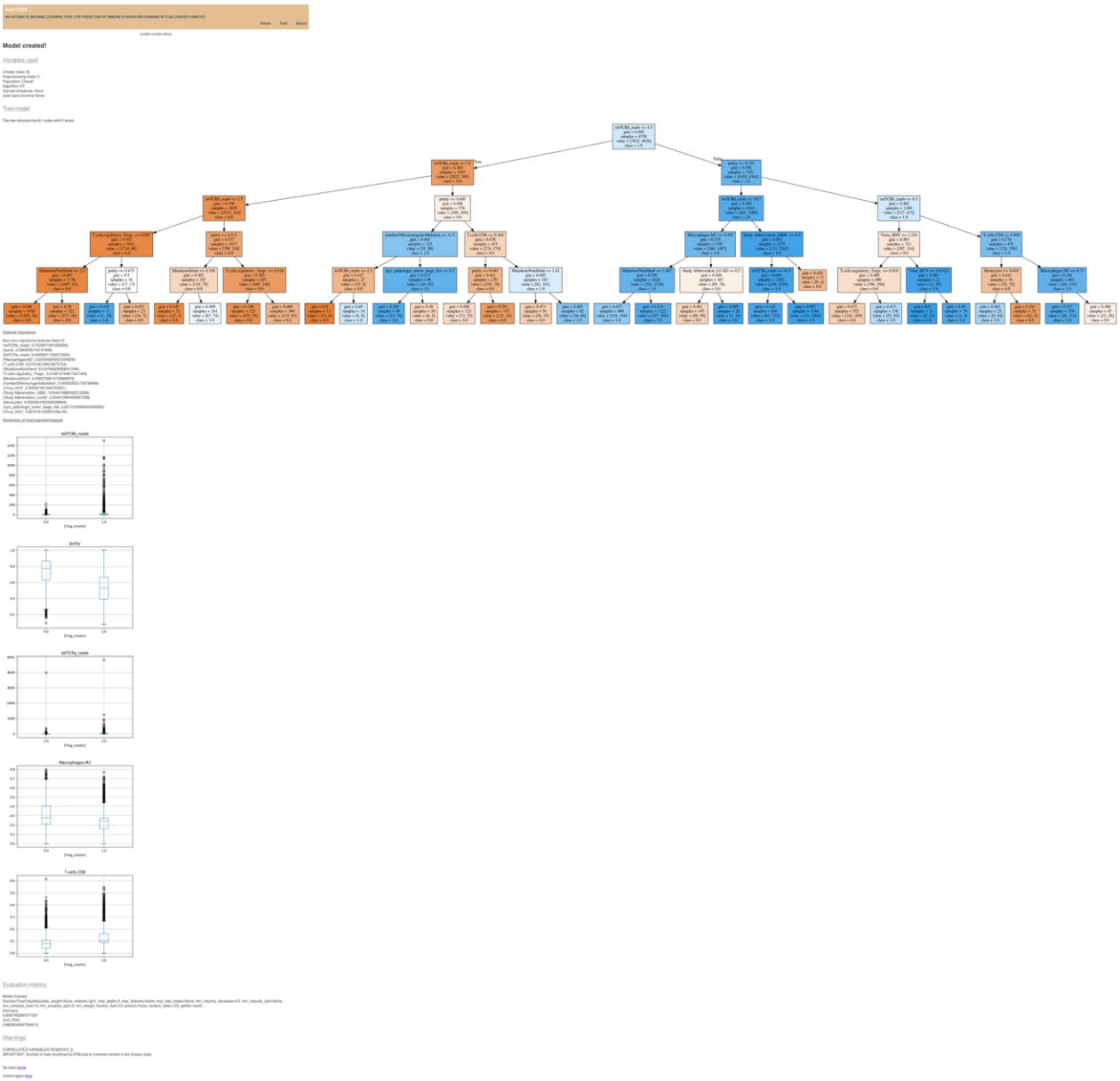


Figure 35: AutoTCGA web-app results preview for case study 1.

5.2 HPV virus prediction in OV cancer patients with full PanImmune dataset

The prediction of class "Virus_HPV" is an important case study as this class represents a family of virus with some cancer potential, in particular human papillomaviruses 16 and 18 carry a risk of becoming cancerous [61].

For the full set of features in the PanImmune dataset, choosing the population of cancer patients with OV cancer type, a Decision Tree was trained to predict the "Virus_HPV" class, and without any feature selection or feature engineering techniques applied, the accuracy score obtained was 67% and AUC-ROC of 76%. The tree structure

obtained has 33 nodes with 5 levels. In this tree there are 14 features selected as important, which are represented below (Figure 36) as well as the model (Figure 37) and the 5 most important features distribution (Figure 38). A screenshot of the tool is also represented below (Figure 39 and Figure 40).

From the model we can extract 3 rules to predict the abundance of virus HPV in cancer patients:

1. T.cells.CD4.memory.resting <= 0.124 & T.cells.CD8 > 0.078 & Virus_saimiriine_Herpesvirus > 0.008 & MutationsNonSilent <= 1.627, gini = 0.165, samples = 11, value=[1, 10], class=1
2. T.cells.CD4.memory.resting > 0.124 & HRD > 32.5 & Macrophages.M2 > 0.431 , gini=0.0 , samples = 9, value = [0,9], class=1
3. T.cells.CD4.memory.resting > 0.124 & HRD > 32.5 & Macrophages.M2 <= 0.431 & T.cells.follicular.helper > 0.053 & NK.cells.resting <= 0.287, gini = 0.287, samples = 23, value = [4, 19], class =1

These rules were extracted from nodes with a gini index lower than 0.3 (high homogeneity) but no restriction in samples number was applied as the OV population is small. The inverse rules describe the population with value 0 for class "Virus_HPV", which means they are depleted in HPV virus.

The rules obtained are more complex than the previous case study rules. The majority of the rules suggest that when T-cells are present (from various types: CD4 memory resting, CD8, follicular helper) the virus is also abundant, which makes sense, as the immune cells are naturally induced to respond and increase their numbers in the presence of an outsider such as a virus. The remaining rules should be analysed by experts in the field as they do not have a straightforward interpretation, as previously observed in Case Study 1.

Interestingly observing the boxplots (Figure 38) there is one feature with high importance not mentioned in the rules, which is "Treg_cluster", the "IS" attribute (immunosuppressors attribute, used as chosen class in Case study 1). The immunosuppressors seem to be in abundance in the presence of Virus, which goes against the above conclusions obtained from the rules: the immune system is being supressed by the immunosuppressors in high numbers.

```
Features Importance
Non-zero importance features listed:  14
                              importance
Treg_cluster                    0.299511
T.cells.CD4.memory.resting      0.166016
T.cells.CD8                     0.134580
Virus_Saimiriine_Herpesvirus    0.064781
NK.cells.activated              0.058050
Macrophages.M1                  0.042208
MutationsSilent                 0.036932
T.cells.follicular.helper       0.036273
Macrophages.M2                  0.033199
Plasma.cells                    0.030144
HRD                             0.027796
MutationsNonSilent              0.025674
NK.cells.resting                0.023070
LOH_n_seg                       0.021764
```

Figure 36: Feature Importance list of model trained with original PanImmune dataset of OV population, to predict Virus_HPV class.

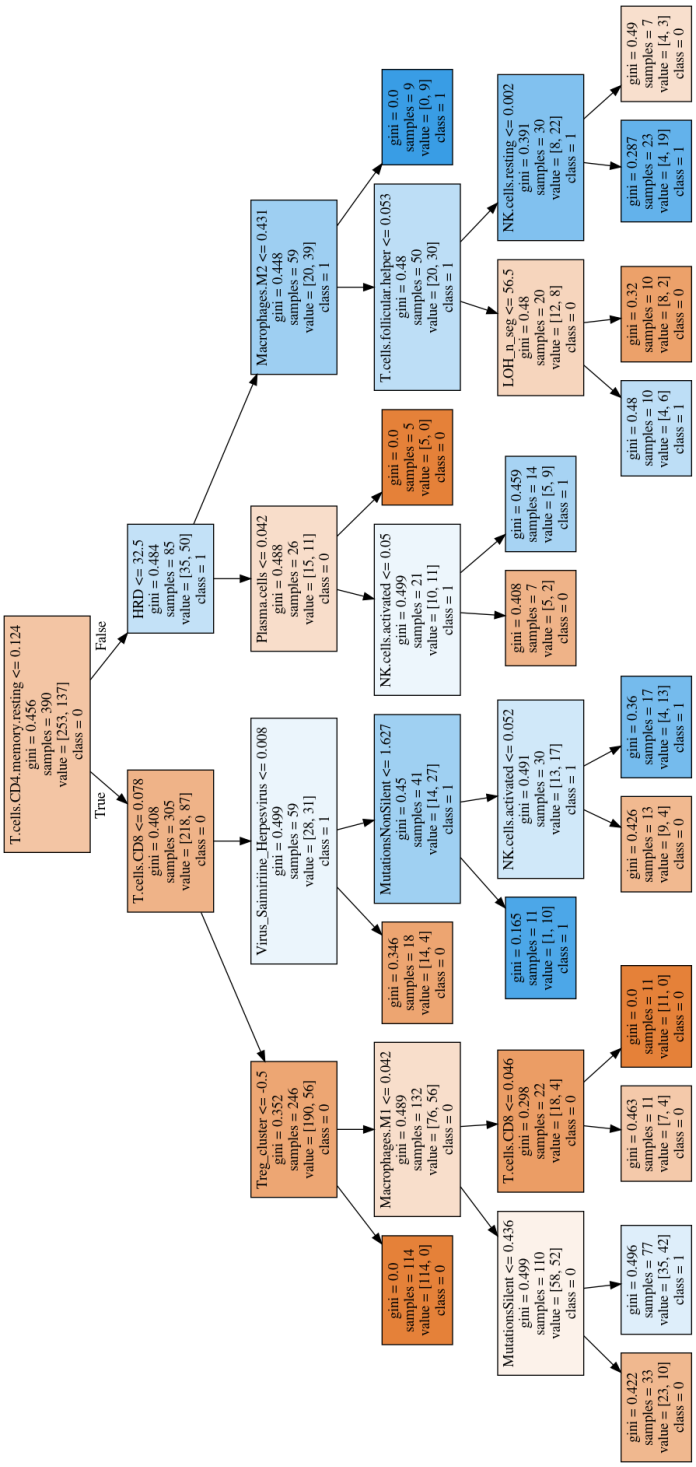


Figure 37: Decision Tree model trained with original PanImmune dataset of OV population, to predict Virus_HPV class.

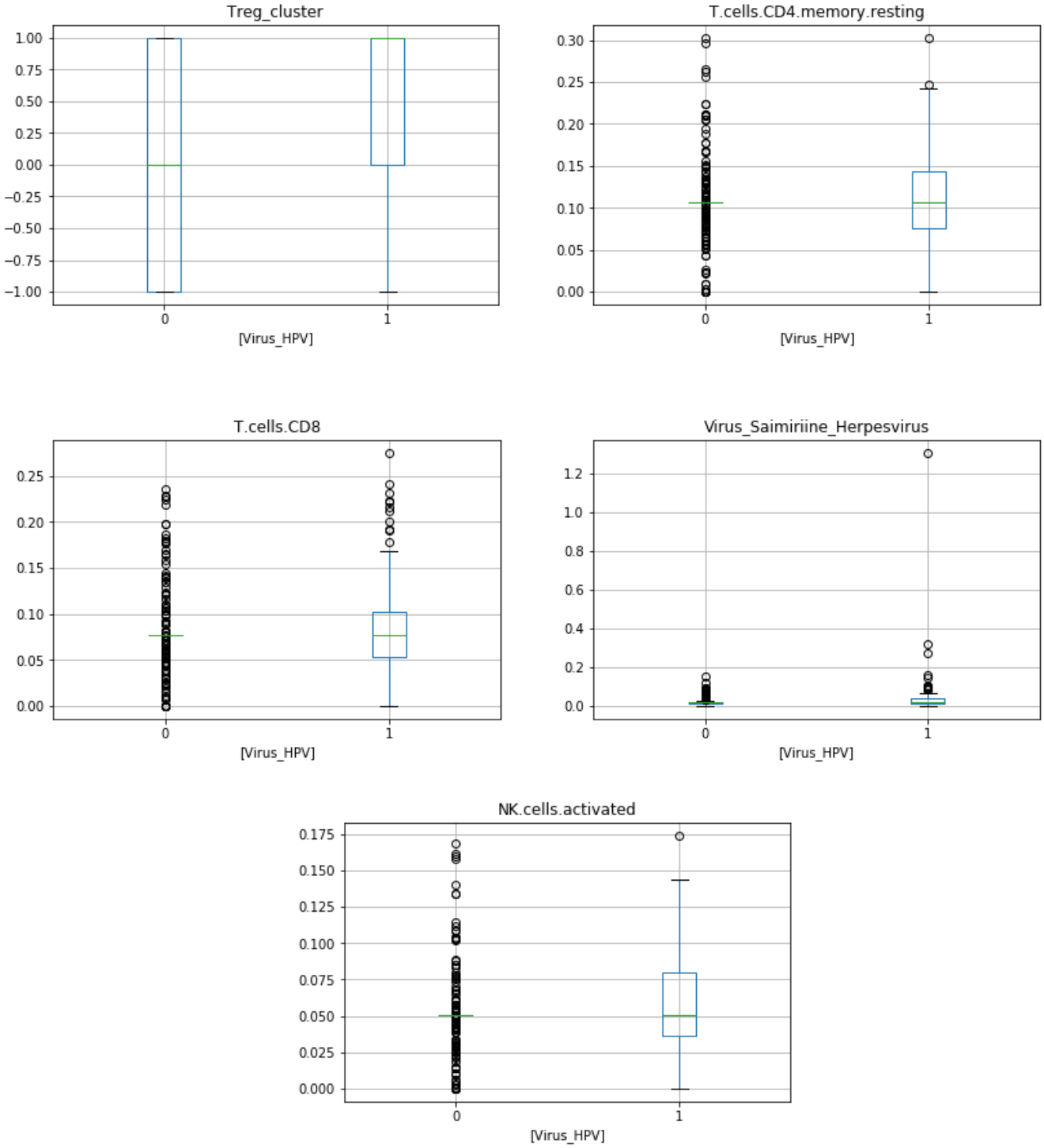


Figure 38: Distribution of 5 most important features of model trained with original PanImmune dataset of OV population, to predict Virus_HPV class.

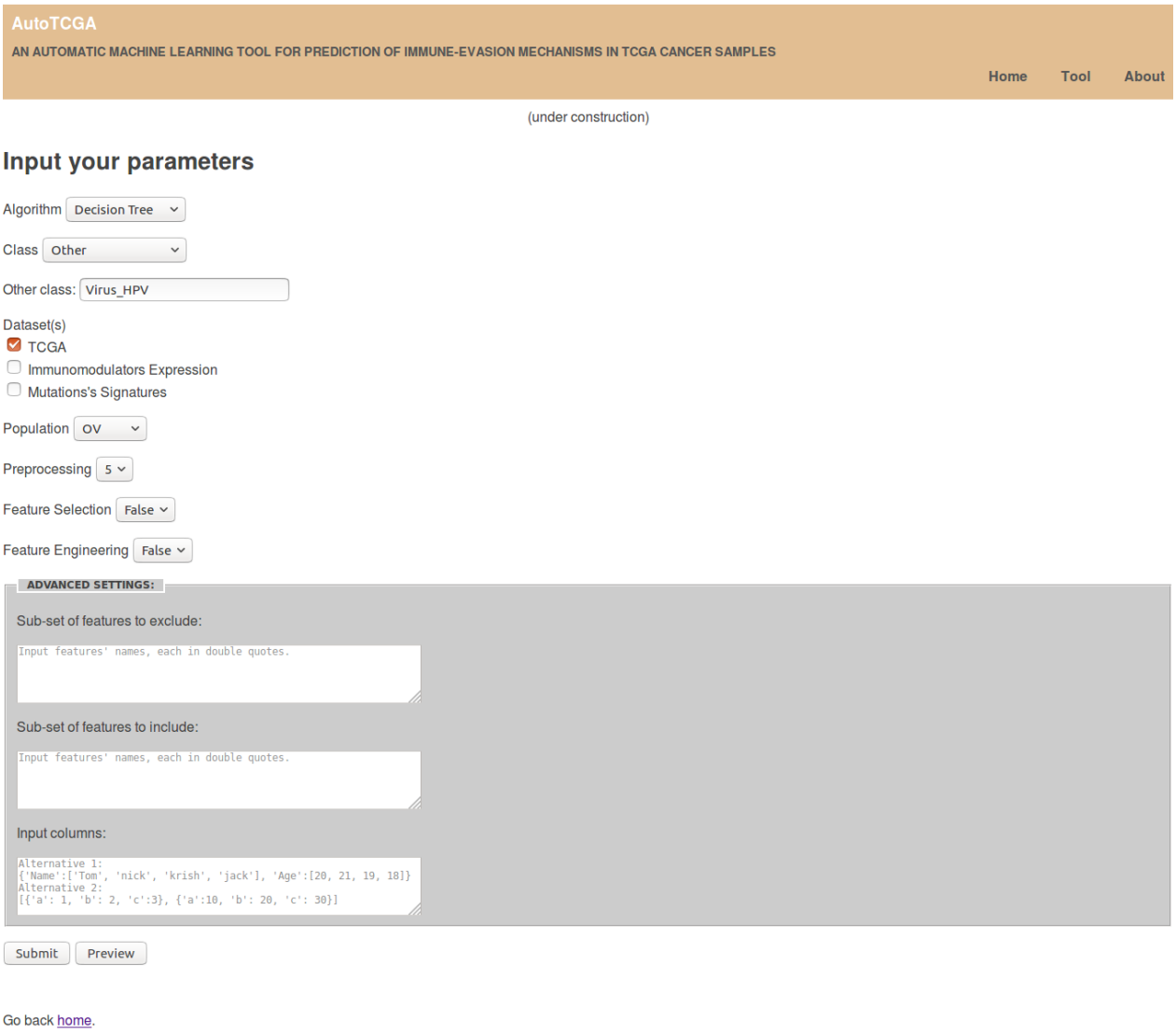


Figure 39: AutoTCGA web-app parameters input window for case study 2.

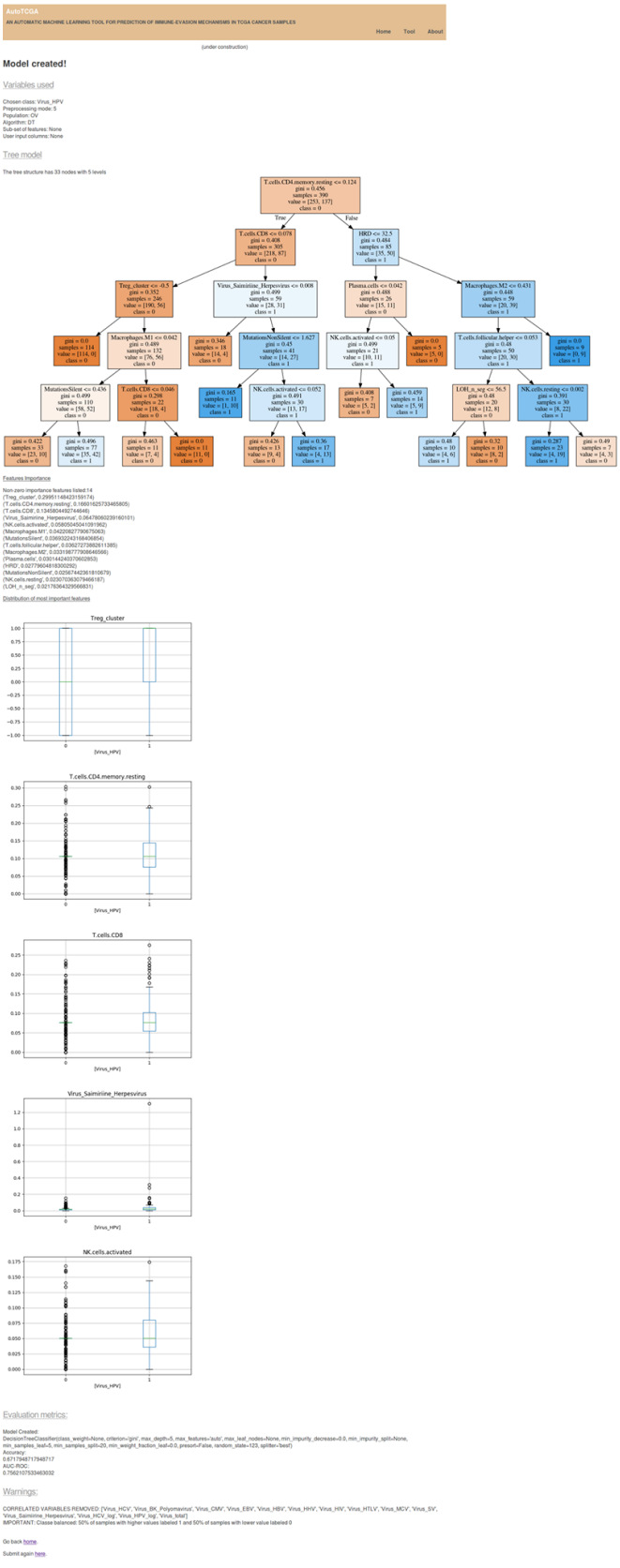


Figure 40: AutoTCGA web-app results preview for case study 2.

6 Conclusion

This project consists in the development of a classification system to predict the cancer immune-evasion mechanisms (considered classes) using biological features of cancer patients (training features) across different types of cancers (population of interest). This system is available to the user through a Web-based tool where the user can select the features, the desired class chosen population, and obtain the final models. This automatic process leverages on domain information to remove fake predictors and create potential engineered features.

The system built has the completeness of a general system, but it is also characterized by its high level of personalization, as the user is able to choose all the classification parameters and the dataset itself. In AutoTCGA, any feature from the dataset can be chosen as a class to be predicted, not just an immune-evasion event, while the remaining features will be the training attributes. There is a wide selection of features available for prediction, which can be filtered or removed, and it is possible to extend the dataset by uploading user features. The strategy used to deal with different data types included a class binarization approach, when continuous variables are chosen as classes, together with the creation of dummy features of all categorical attributes. Given the good results of the algorithm's performance metrics, obtained in a binary classification scenario, this strategy was considered successful.

Conducting both the pre-processing techniques study and the system's parameters tuning analyses showed that the system is highly flexible and allows a high number of possible combinations of parameters (a personalized usage of the system), which results in very different models. At the same time, the models produced by this tool fit in a wide range of values of evaluation metrics, depending on the chosen algorithm, dataset, pre-processing techniques, population, and class, as expected in a classification problem. Many models were found to be of low relevance given that the evaluation metrics were low, mainly Decision Tree models and models predicting a multiclass or multilabel scenario. But many models achieved high evaluation scores, namely XGBoost and Random Forest models predicting the "IS" class.

To accomplish such a flexible system, the architecture was built in a modular fashion and each module counts on a high level of code abstraction. Consequently, the system is highly robust and reproducible, as it is possible to recreate every result using the same combination of input parameters. As an example, the datasets Immunomodulator's expression and Mutation's signatures were added after the system had been completely built with the PanImmune dataset and this addition was done with minimal effort: simply adding the dataset parameter to the system, instead of having that fixed as "PanImmune". This shows how the system is built in an abstract way and not hard coded for the dataset, algorithm or population of interest.

The system's modularity also allows the system to be highly extensible as the modules can be activated or deactivated, and new modules can be added to the pipeline without changing the remaining modules in the system. After showing the system to the cancer biologists at King's College, they requested the addition of two other modules: user input of columns (so they could use the system with confidential data), user defined features, both

including or excluding columns from the dataset, and the possibility of merging datasets. All these extensions were implemented and tested in two days of work.

These system's parameters tuning analyses also led to two other conclusions. Firstly, the multilabel and multiclass definitions of the problem were not successfully modelled by the algorithms, as intended since they are closer to the biological problem, but the evaluation metrics showed very good results for the binary classification problems. Secondly, choosing Decision Tree as the default algorithm and restricting the algorithm's hyperparameters (maximum of 6 levels in each tree), decreased the models' evaluation scores (AUC-ROC and accuracy) in only 5% on average, while increasing significantly their interpretability and visualization.

AutoTCGA outperforms the existing tools in its interpretability power applied to the TCGA data, and it is the first to apply machine learning techniques to further understand immune-evasion mechanisms. Even though some tools have applied clustering techniques with good visualization platforms and others have applied powerful classification algorithms, none is able to provide what AutoTCGA does. With the tree-based models the user is able to traceback and understand the patterns of patients with a feature of interest. Those results provide useful insight into associations between the variables, which may be a cause or a consequence of the feature under study.

It is highly important to notice, however, that the information available to the user is the information learnt by the models, which is completely determined by the raw data available in the data sources considered. The user should take into account the evaluation metrics associated with each result, as a way to better acknowledge the significance of such result.

Several models are already available with the tool, namely the models created upon analysing the impact of the application of feature selection and feature engineering in accuracy and AUC-ROC of the algorithms available, trained with the PanImmune dataset, across the PanCancer, BRCA and LUAD populations. Since the other datasets available (Mutation's signatures and Immunomodulator's expression) do not allow the application of feature engineering (at least in the way that the module was built) and also don't allow many of the pre-processing combinations (PanImmune has 6 possible values while the other datasets have only 2), it was decided that it wouldn't be relevant to test them. However, it is a limitation on the analysis made, because the default parameters of the system were decided based on the results of PanImmune dataset but will be used regardless of the dataset.

Overall, the system does not guarantee that all models have high accuracy and AUC-ROC for all parameters combinations, as those scores depend on the data itself. However, the system is always able to provide a model and support it with the evaluation metrics, and statistical significance, informing the user of its quality. For good quality models, the system leverages on the web interface and tree-based algorithms to explain the models to all users, even non-experts in machine learning. Moreover, the system can be easily improved in the future, due to its abstract code and modular structure.

Finally, the importance of AutoTCGA in health care should be reinforced. AutoTCGA was designed to handle any biological feature as the chosen class to be predicted but was tailored to predict immune-evasion mechanisms and has shown great potential for application in targeted immunotherapy. It allows not only testing hypothesis on the importance of certain features, but also studying open questions on what features are important to predict a chosen class. In this study, it has been demonstrated that AutoTCGA is an efficient proof-of-concept of the great application of machine learning pipeline to TCGA data and integrating the computational power of this tool with the high quality of the data visualization tools already developed has great future potential for success.

Future work

One of the most important conclusions taken with the system's parameters analyses concerns the low relevance of both feature selection and feature engineering techniques. In the future, these data analysis techniques should be revised and improved, for instance by testing other approaches such as statistical methods of feature selection (ex: f_classification [60]). Another procedure which was already discussed concerns the detection of collinearity on the dataset, and the creation of a module for collinearity detection could also benefit the results obtained with the feature techniques.

Another important observation concerned the prediction of class "CA" which is created based on an automatically binarization of the values, therefore assuring a balanced population. All other numerical variables will be processed in the same way if the user chooses them as classes. This approach was not proved to be biologically correct, and in fact the model's scores were lower for the prediction of "CA" comparing to the prediction of "IS". In the future, the biologically correct values for each class should be collected (if known) and added to the system's module "Class binarization". Furthermore, the class generation model comes after the data preparation modules (which includes cancer type filtering), and consequently one patient can be considered to have an immune-evasion mechanism active in its cancer type population, and not in the PanCancer population, or vice-versa.

Alternatively, to avoid the class binarization problem, the prediction could be made with continuous variables as it is done in regression problems, replacing the algorithms used (classifiers) by regressors. This approach was implemented in the system and it is functional, but was not included in the analysis, and therefore is not available for the user. To make this functionality available it would be necessary to find a comparison method between the regression performance and the classification performance, which is does not have a straightforward solution, as they differ in the evaluation metrics used.

Regarding the prediction of both multiclass and multilabel targets, it was possible to conclude by the system's parameter tuning analyses that for some populations the 4 classes are significantly imbalanced which influences the models' metrics. This should be fixed, implementing a class balancing module, appropriate for multiclass and multilabel problems.

From the performance analysis of AutoTCGA, when compared with other tools, it was possible to conclude that there were important characteristics present in other tools but absent in AutoTCGA, namely the possibility of filtering the dataset using demographic features and building Kaplan-Meyer curves with the data.

Regarding the visualization of the results, there are still a lot to improve as well, regarding the user experience of the web-app, as well as more particular aspects such as the visualization of the XGBoost models which is not available, and the visualization of the features' importance in both Random Forest and XGBoost models, which should be calculated based on all the estimators features' importance.

Some modules were left for future extension on this project such as "Feature creation" and "Model suggestion". The former focuses on creating two important biological classes: "HLA absence" and "HLA mutations", which require that softwares LOHHLA [11] and Polysolver [10] process the WXS of the patients. This was not possible for the time being, due to storage limitations to handle the WXS files, as previously mentioned. The latter module would make suggestions of models to the user, based on saved models created previously in other sessions (by that user or others). The final purpose would be to suggest models with better evaluation scores than the scores obtained by the model requested by the user. The models suggested would have similar requirements, but would be possibly missing one or more requirements, or would be using a different value in one of the parameters. An extension of this module would be possible when the user requests for a model in population of the dataset (indicating the cancer type) and the system would then provide the requested model as well as a model obtained with the full population (PanCancer) that would serve as control. To implement both these modules more resources would be necessary in order to handle the high levels of data storage needed.

References

1. Dass S.Vinay, et. al (2015) Immune evasion in cancer: Mechanistic basis and therapeutic strategies
2. Charoentong P, et. al (2017) Pan-cancer Immunogenomic Analyses Reveal Genotype-Immunophenotype Relationships and Predictors of Response to Checkpoint Blockade
3. Hoadley KA, et. al (2018) Cell-of-Origin Patterns Dominate the Molecular Classification of 10,000 Tumors from 33 Types of Cancer
4. Thorsson V, et. al (2018) The Immune Landscape of Cancer.
5. Cancer UK https://www.cancerresearchuk.org/ (accessed November 2018)
6. TCGA project https://cancergenome.nih.gov/ (accessed October 2018)
7. National Cancer Institute (of National Institute of Health) https://www.cancer.gov/ (accessed July 2018)
8. Vazquez, M., et. al (2012) Cancer Genome Analysis, Chapter 14, PLoS Comput Biol. 2012
9. Turski, Michelle, PhD, The Lund Berginstitute http://mini-media.net/thelundberginstitute/ (accessed December 2018)
10. Shukla, SA, (2016) Comprehensive analysis of cancer-associated somatic mutations in class I HLA genes
11. McGranahan NA, (2017) Allele-Specific HLA Loss and Immune Escape in Lung Cancer Evolution
12. Benbaruch A. (2006). Inflammation-associated immune suppression in cancer: The roles played by cytokines, chemokines and additional mediators. Seminars in Cancer Biology, 16(1), 38–52. doi:10.1016/j.semcancer.2005.07.00
13. Malta t., (2018) Machine Learning Identifies Stemness Features Associated with Oncogenic Dedifferentiation
14. Rubio-Perez, C, et al, (2015) In silico prescription of anticancer drugs to cohorts of 28 tumor types reveals targeting opportunities.
15. Landau, DA, et al (2013) Evolution and impact of subclonal mutations in chronic lymphocytic leukemia.
16. Feature Tools documentation https://www.featuretools.com/
17. L. Breiman, (1999) "Pasting small votes for classification in large databases and on-line", Machine Learning, 36(1), 85-103.
18. L. Breiman, (1996) "Bagging predictors", Machine Learning, 24(2), 123-140.
19. T. Ho, (1998) "The random subspace method for constructing decision forests", Pattern Analysis and Machine Intelligence, 20(8), 832-844.
20. G. Louppe and P. Geurts, (2012) "Ensembles on Random Patches", Machine Learning and Knowledge Discovery in Databases, 346-361
21. GitHub page of FeatureSelector tool https://github.com/WillKoehrsen/feature-selector
22. Strobl c., et. al, (2007) Bias in random forest variable importance measures: Illustrations, sources and a solution
23. Liu H. (1998) Feature Selection for Knowledge Discovery and Data Mining
24. J.R. Quinlan. (1993) C4. 5: programs for machine learning. Morgan Kaufmann
25. T. Hastie, R. Tibshirani and J. Friedman. 2009 Elements of Statistical Learning, Springer.
26. P. Geurts, D. Ernst., and L. Wehenkel, (2006) "Extremely randomized trees", 63(1), 3-42
27. Y. Freund, and R. Schapire, (1997) "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting"
28. Zhu, H. Zou, S. Rosset, T. Hastie, (2009) "Multi-class AdaBoost"
29. J. Friedman, (1999) "Stochastic Gradient Boosting"
30. TCGA repository
https://www.cell.com/pb-assets/consortium/pancanceratlas/pancani3/index.html
31. GDC portal https://portal.gdc.cancer.gov/

32. Bailey H., et. al (2018) Comprehensive Characterization of Cancer Driver Genes and Mutations
33. Fisher, R. A. (1922). "On the interpretation of χ2 from contingency tables, and the calculation of P". Journal of the Royal Statistical Society. 85 (1): 87–94. doi:10.2307/2340521. JSTOR 2340521.
34. Sidhom JW (2018) ImmunoMap: A Bioinformatics Tool for T-cell Repertoire Analysis.
35. Python library "sklearn" documentation https://scikit-learn.org/stable/documentation.html
36. Gaye Lightbody, et. al (2018) *Review of applications of high-throughput sequencing in personalized medicine: barriers and facilitators of future progress in research and clinical application*, Briefings in Bioinformatics
37. TCGA Computational Tools, https://www.cancer.gov/about-nci/organization/ccg/research/structural-genomics/tcga/using-tcga/tools (accessed July 2019)
38. Pivotal Engineering Journal, *Interpreting Decision Trees and Random Forests*, http://engineering.pivotal.io/post/interpreting-decision-trees-and-random-forests/ (accessed July 2019)
39. American Cancer Society, https://www.cancer.org/treatment/treatments-and-side-effects/treatment-types/immunotherapy.html
40. PanImmune data repertoire, https://gdc.cancer.gov/about-data/publications/panimmune
41. Mutations signatures platform, *mSignaturedb*, http://tardis.cgu.edu.tw/msignaturedb/
42. Givechian, Kevin B. , et. al (2018) *Identification of an immune gene expression signature associated with favorable clinical features in Treg-enriched patient tumor samples*
43. XGBooost library for python https://xgboost.readthedocs.io/en/latest/python/python_intro.html
44. Tianqi Chen et. al (2016) *XGBoost: A Scalable Tree Boosting System*
45. Permutation score documentation: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.permutation_test_score.html
46. Ronglai Shen et. al (2009) *Integrative clustering of multiple genomic data types using a joint latent variable model with application to breast and lung cancer subtype analysis*
47. Charoentong, P. et. al (2016) *Pan-cancer Immunogenomic Analyses Reveal Genotype-Immunophenotype Relationships and Predictors of Response to Checkpoint Blockade*
48. TCPA software (The Cancer Proteome Atlas Portal) https://www.tcpaportal.org/index.html
49. Mary Goldman et. al (2019) *The UCSC Xena platform for public and private cancer genomics data visualization and interpretation*
50. Yulia Newton et. al (2017) *TumorMap: Exploring the Molecular Similarities of Cancer Samples in an Interactive Portal*
51. CRC Aggressiveness Explorer *http://explorer.cancerregulome.org/crc_agg/*
52. Ethan Cerami et. al (2012) *The cBio Cancer Genomics Portal: An Open Platform for Exploring Multidimensional Cancer Genomics Data*
53. Standard Deviation documentation https://support.office.com/pt-pt/article/desvpad-p-fun%C3%A7%C3%A3o-desvpad-p-6e917c05-31a0-496f-ade7-4f4e7462f285
54. http://www.who.int/mediacentre/factsheets/fs297/en/ (accessed July 2019)
55. http://www.cancer.org/cancer/lungcancer-non-smallcell/detailedguide/non-small-cell-lung-cancer-what-is-non-small-cell-lung-cancer. (accessed July 2019)
56. https://www.wcrf.org/dietandcancer/cancer-trends/breast-cancer-statistics (accessed July 2019)
57. AutoSklearn documentation https://github.com/automl/auto-sklearn
58. Philip Probst et. al (2018) *To Tune or Not to Tune theNumber of Trees in Random Forest*
59. https://simonhessner.de/why-are-precision-recall-and-f1-score-equal-when-using-micro-averaging-in-a-multi-class-problem/ (accessed July 2019)
60. F_classification approach for feature selection implemented in scikit-learn library https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.f_classif.html#sklearn.feature_selection.f_classif

61. Muñoz N, et. al (2006). *"Chapter 1: HPV in the etiology of human cancer".* Vaccine. 24 Suppl 3

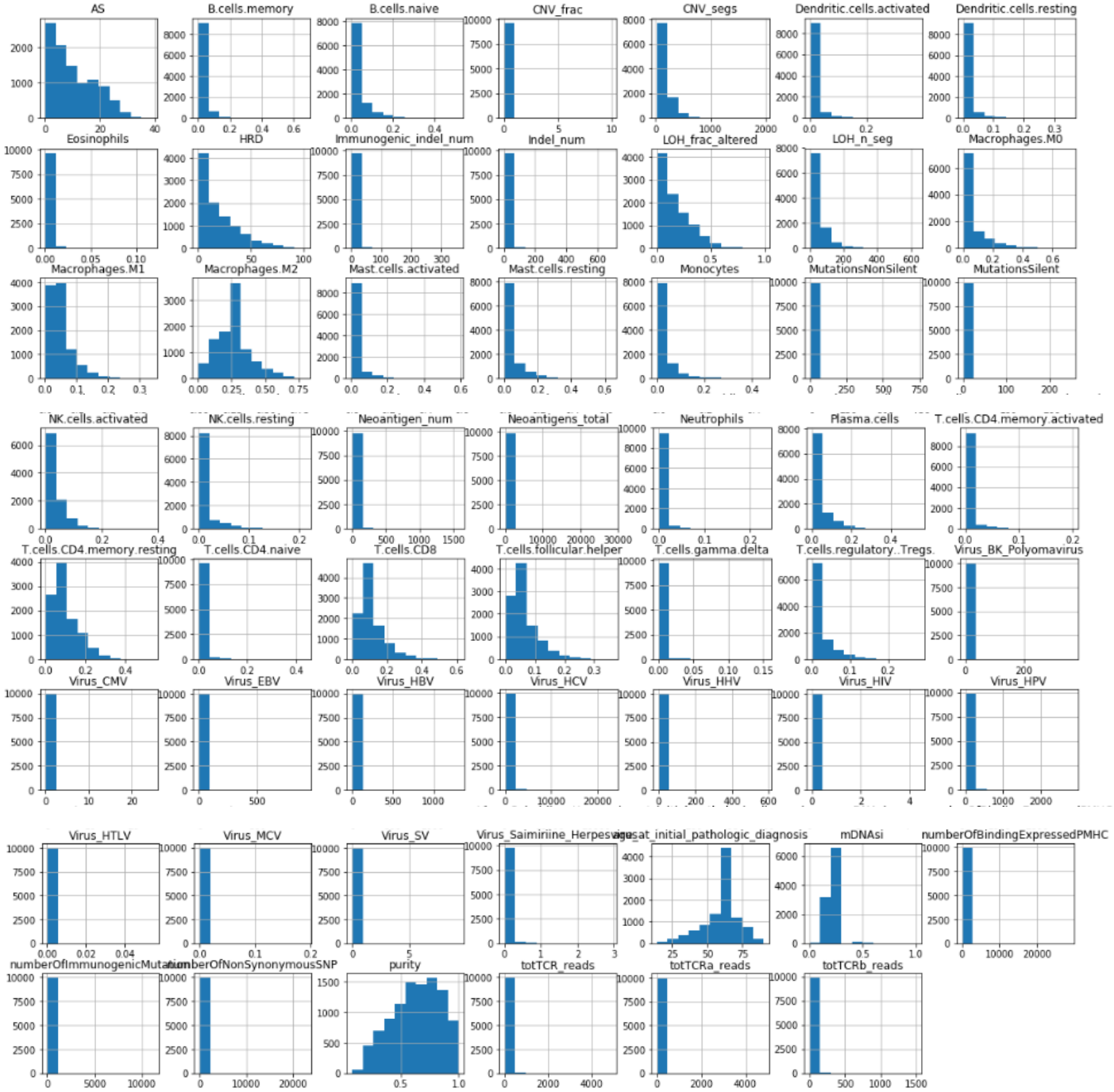
Annex

1. Metadata
    a. Dataset variables description

Column Name	Type	Range of Values (AC)	Count Unique (AC)	NonMissing Values (BC)	Description
AS	Continuous	0-39	-	9977	Absolute Score
LOH_n_seg	Continuous	0-634	-	9977	Number of segments with LOH
LOH_frac_altered	Continuous	0-0.994437	-	9977	
MutationsSilent	Continuous	0-247.236293	-	9162	
MutationsNonSilent	Continuous	0-733.234354		9162	
Study Abbreviation	Categorical	-	33	9977	
Study Name	Categorical	-	33	9977	
CNV_segs	Continuous	23-1966	-	9872	Copy Number variations
CNV_frac	Continuous	0-9.899119	-	9977	Copy Number variations
HRD	Continuous	0-101	-	9977	Homologous Recombinant Defficiency
purity	Continuous	0.08 - 1	-	9977	
Virus_CMV	Continuous	0-24.789	-	9233	Load of Virus X (12)
Virus_EBV	Continuous	0- 922.065	-	9233	
Virus_HBV	Continuous	0 - 1317.718	-	9233	
Virus_HCV	Continuous	0-23336.108	-	9233	
Virus_HHV	Continuous	0-585.007	-	9233	
Virus_HIV	Continuous	0-4.391	-	9233	
Virus_HPV	Continuous	0-2831.601	-	9233	
Virus_HTLV	Continuous	0-0.055	-	9233	
Virus_MCV	Continuous	0-0.192	-	9233	
Virus_SV	Continuous	0-9.369	-	9233	
Virus_BK_Polyomavirus	Continuous	0-364.331	-	9233	
Virus_Saimiriine_Herpesvirus	Continuous	0-2.918	-	9233	
numberOfNonSynonymousSNP	Continuous	1-22831	-	8261	
numberOfImmunogenicMutation	Continuous	0-11244	-	8261	
numberOfBindingExpressedPMHC	Continuous	0-28358	-	8261	SNV Neoantigens
Indel_num	Continuous	1-685	-	6455	Indel mutations
Immunogenic_indel_num	Continuous	0-355	-	6455	Indel Immunogenic Mutations
Neoantigen_num	Continuous	0-1586	-	6455	Indel Neoantigens
Subtype_mRNA	Categorical	-	23	9977	Clustering result in feature X
Subtype_DNAmeth	Categorical	-	15	9977	
Subtype_protein	Categorical	-	6	8177	
Subtype_miRNA	Categorical	-	9	9977	
Subtype_CNA	Categorical	-	13	9977	
Subtype_Integrative	Categorical	-	6	9977	
Subtype_other	Categorical	-	20	9977	
Subtype_Selected	Categorical	-	45	9977	
B.cells.naive	Continuous	0- 0.512	-	7930	
B.cells.memory	Continuous	0- 0.672	-	7930	

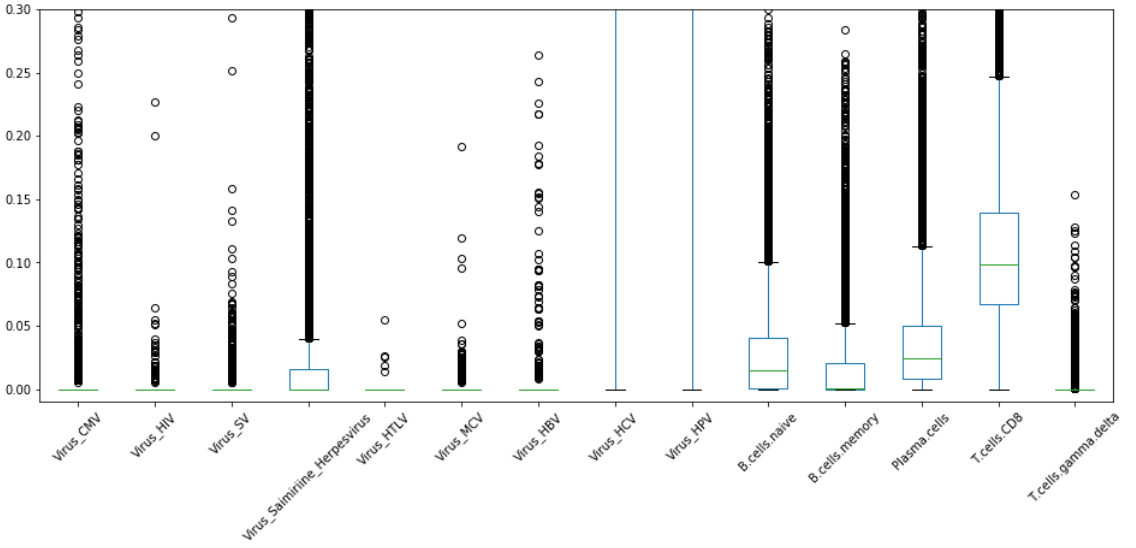
Plasma.cells	Continuous	0- 0.543	-	7930	
T.cells.CD8	Continuous	0- 0.614	-	7930	
T.cells.CD4.naive	Continuous	0- 0.441	-	7930	
T.cells.CD4.memory.resting	Continuous	0- 0.539	-	7930	
T.cells.CD4.memory.activated	Continuous	0- 0.199	-	7930	
T.cells.follicular.helper	Continuous	0- 0.362	-	7930	
T.cells.regulatory..Tregs.	Continuous	0- 0.281	-	7930	
T.cells.gamma.delta	Continuous	0- 0.154	-	7930	
NK.cells.resting	Continuous	0- 0.213	-	7930	
NK.cells.activated	Continuous	0- 0.381	-	7930	
Monocytes	Continuous	0- 0.452	-	7930	
Macrophages.M0	Continuous	0- 0.717	-	7930	
Macrophages.M1	Continuous	0- 0.339	-	7930	
Macrophages.M2	Continuous	0- 0.797	-	7930	
Dendritic.cells.resting	Continuous	0- 0.351	-	7930	
Dendritic.cells.activated	Continuous	0- 0.378	-	7930	
Mast.cells.resting	Continuous	0-0.636	-	7930	
Mast.cells.activated	Continuous	0-0.585	-	7930	
Eosinophils	Continuous	0-0.117	-	7930	
Neutrophils	Continuous	0-0.234	-	7930	
totTCR_reads	Continuous	0-4925	-	8123	Total Cell Receptors
totTCRa_reads	Continuous	0-4818	-	8123	
totTCRb_reads	Continuous	0-1496	-	8123	
age_at_initial_pathologic_diagnosis	Continuous	14-90	-	9977	Age
gender	Binary	-	2	9977	
race	Categorical	-	6	9977	
ajcc_pathologic_tumor_stage	Categorical	-	20	9977	Tumor Stage
histological_type	Categorical	-	124	9977	
histological_grade	Categorical	-	8	9977	
vital_status	Categorical	-	3	9977	
tumor_status	Categorical	-	4	9977	
cause_of_death	Categorical	-	12	9977	
treatment_outcome_first_course	Categorical	-	9	9977	Outcome of treatment
OS	Binary	-	-	9977	Overall survival
mDNAsi	Continuous	0-1	-	7287	Stemness index
A1	Categorical	-	79	8326	HLA allele A1
A2	Categorical	-	79	8326	HLA allele A2
B1	Categorical	-	144	8326	HLA allele B1
B2	Categorical	-	146	8326	HLA allele B2
C1	Categorical	-	55	8326	HLA allele C1
C2	Categorical	-	63	8326	HLA allele C2

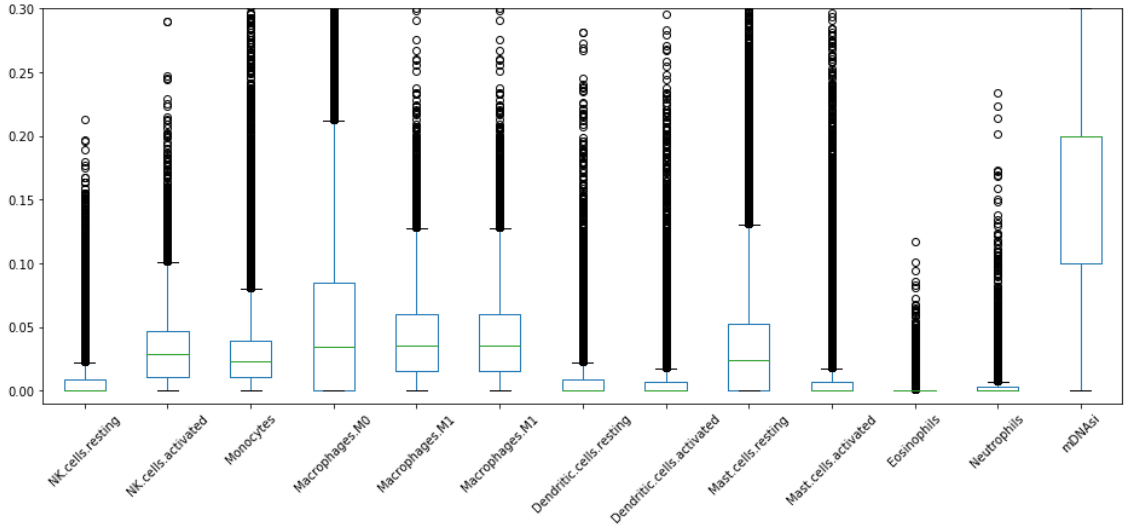
b. Continuous features histograms



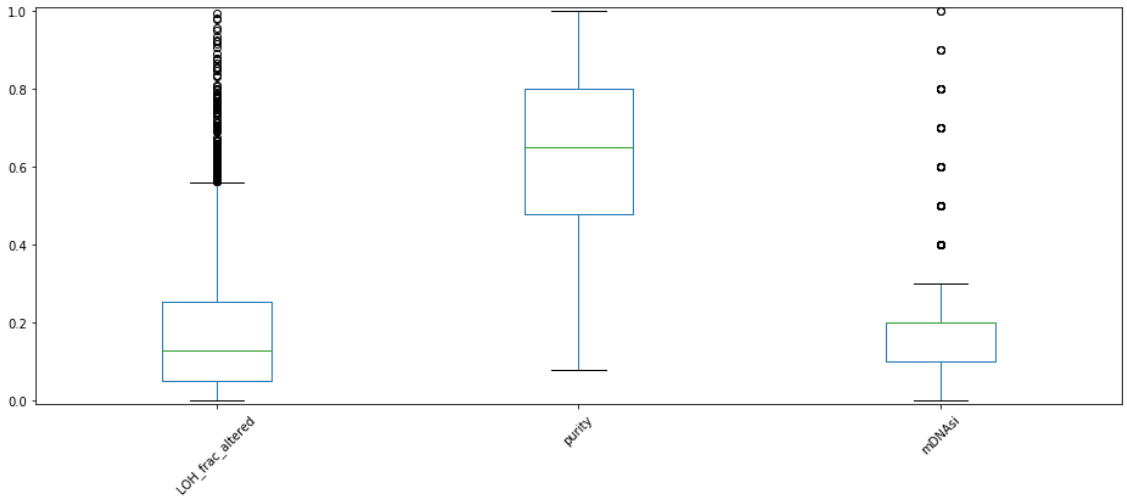
c. Continuous Variables Boxplots

i. Variables ranging 0-0.3

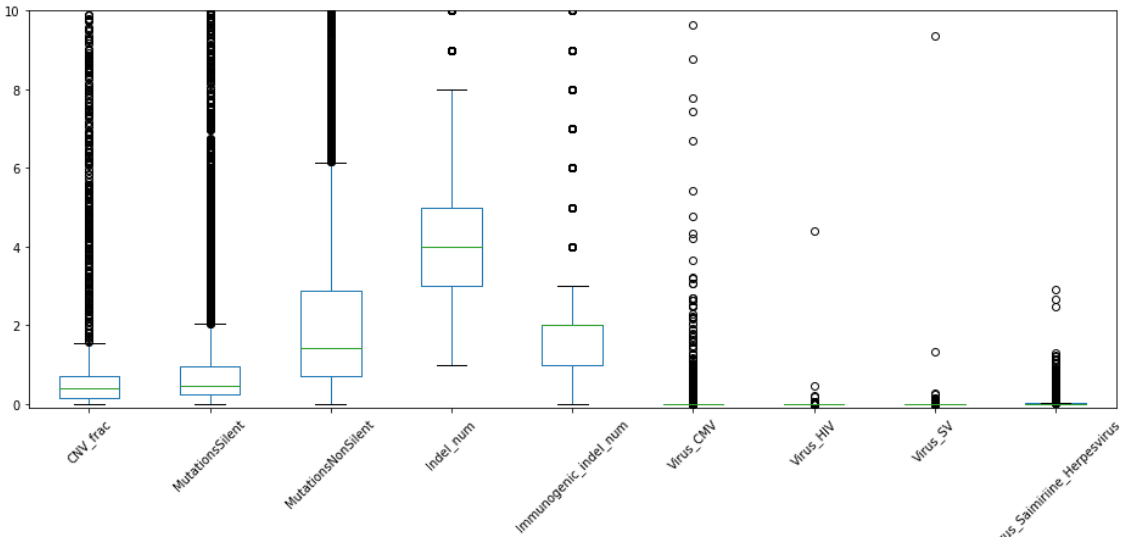




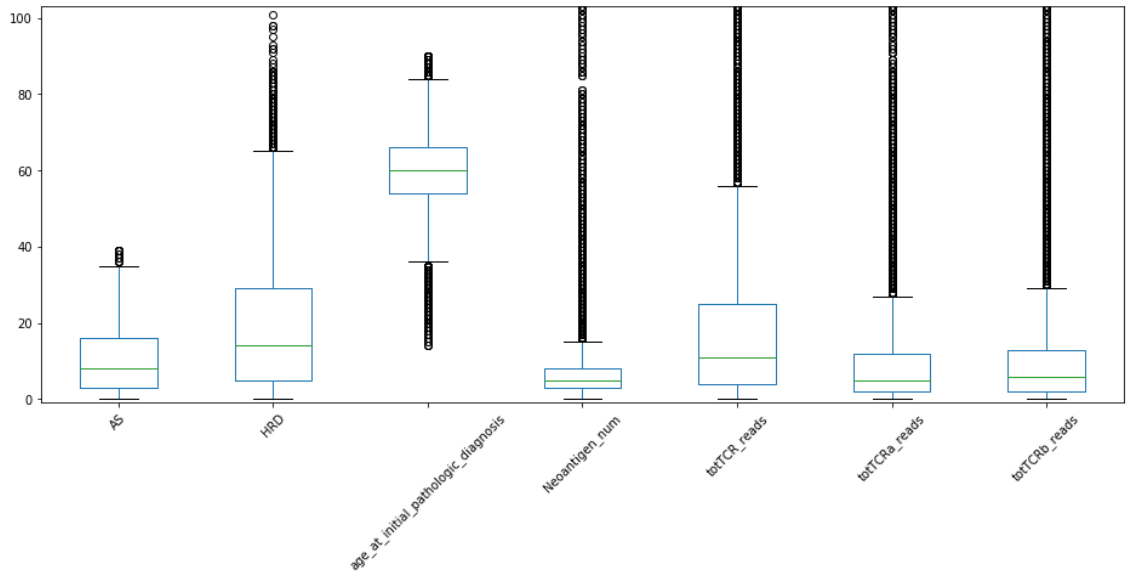
ii. Variables ranging 0-1



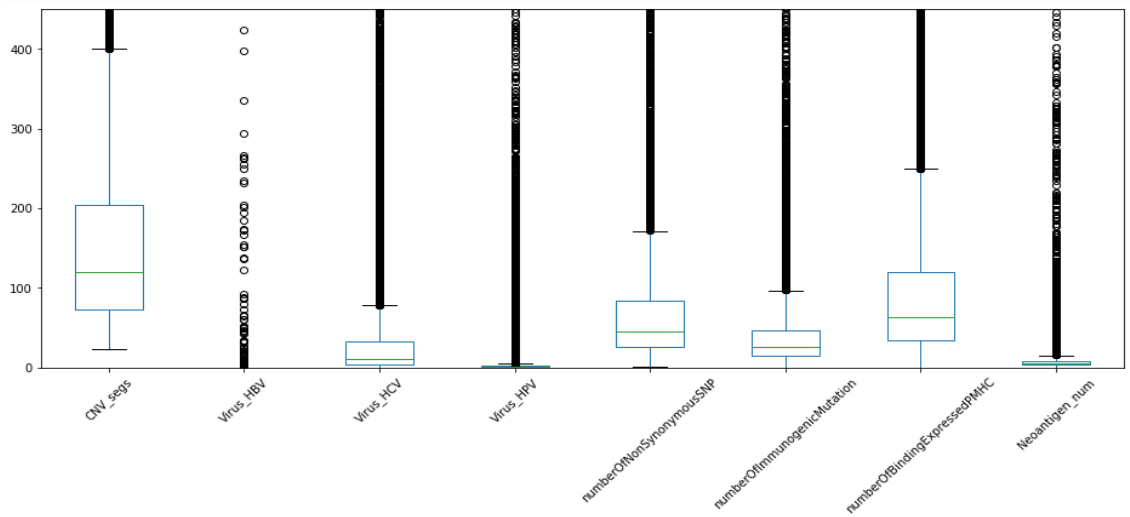
iii. Variables ranging 0-10



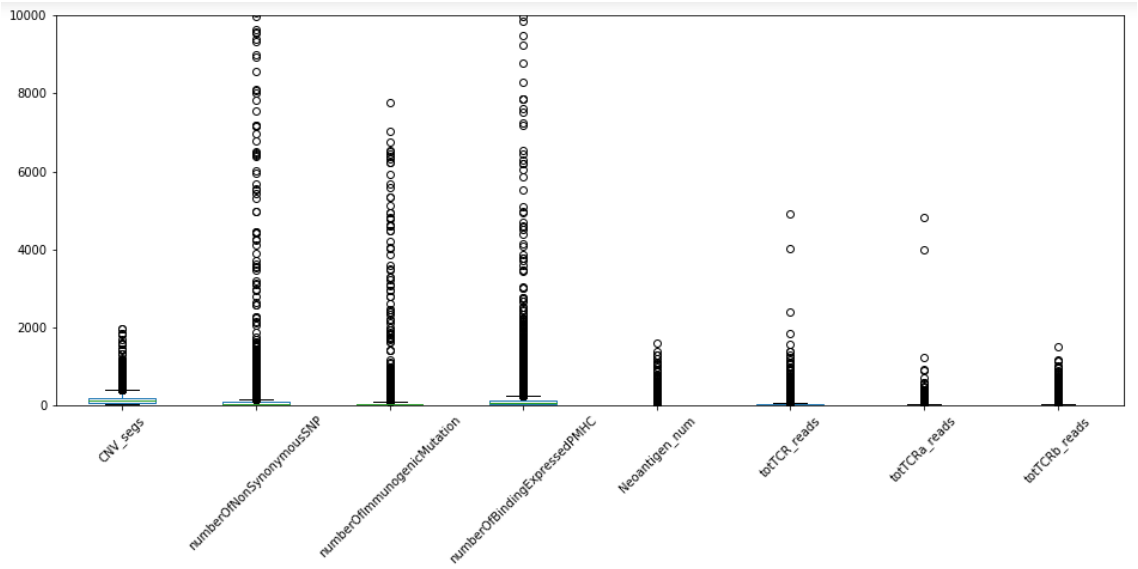
iv. Variables ranging 0-100



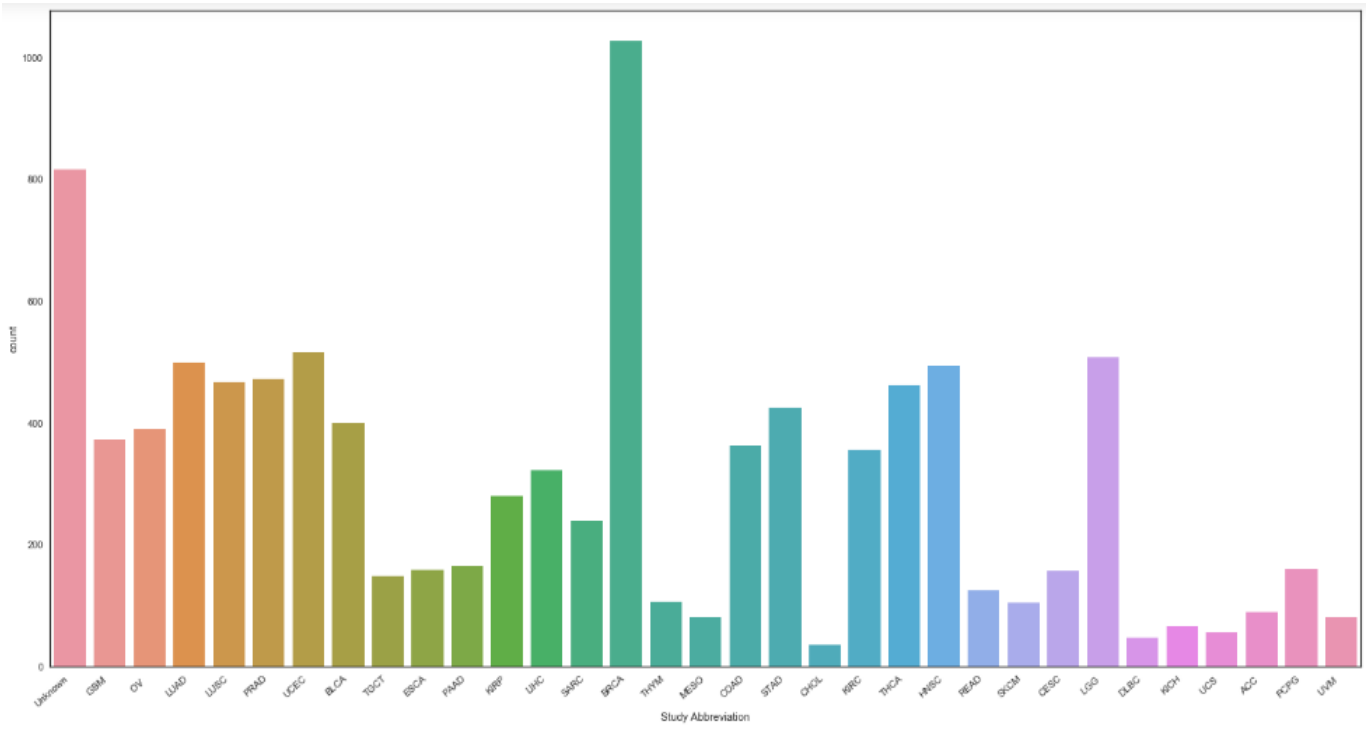
v. Variables ranging 0-500

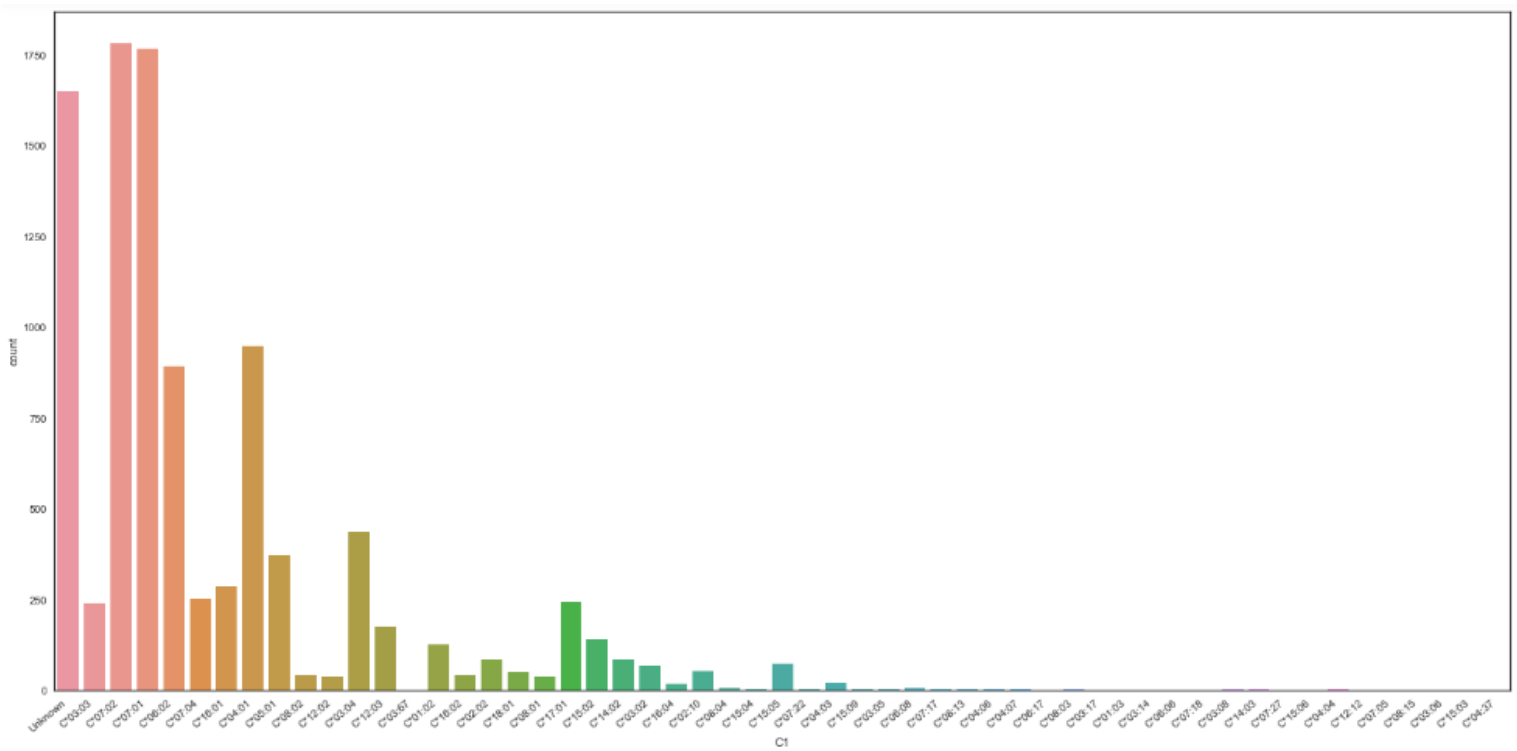


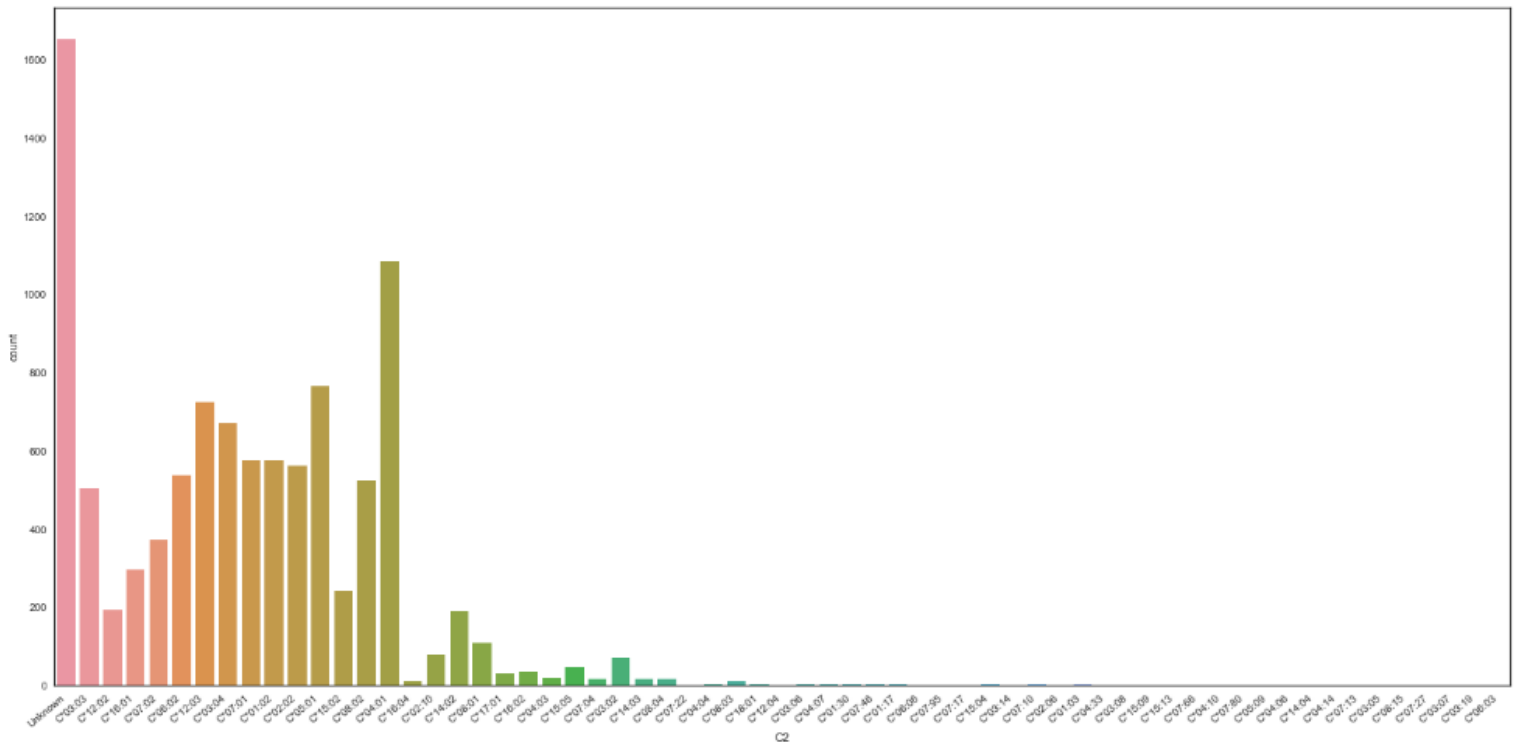
vi. Variables ranging 0-10.000

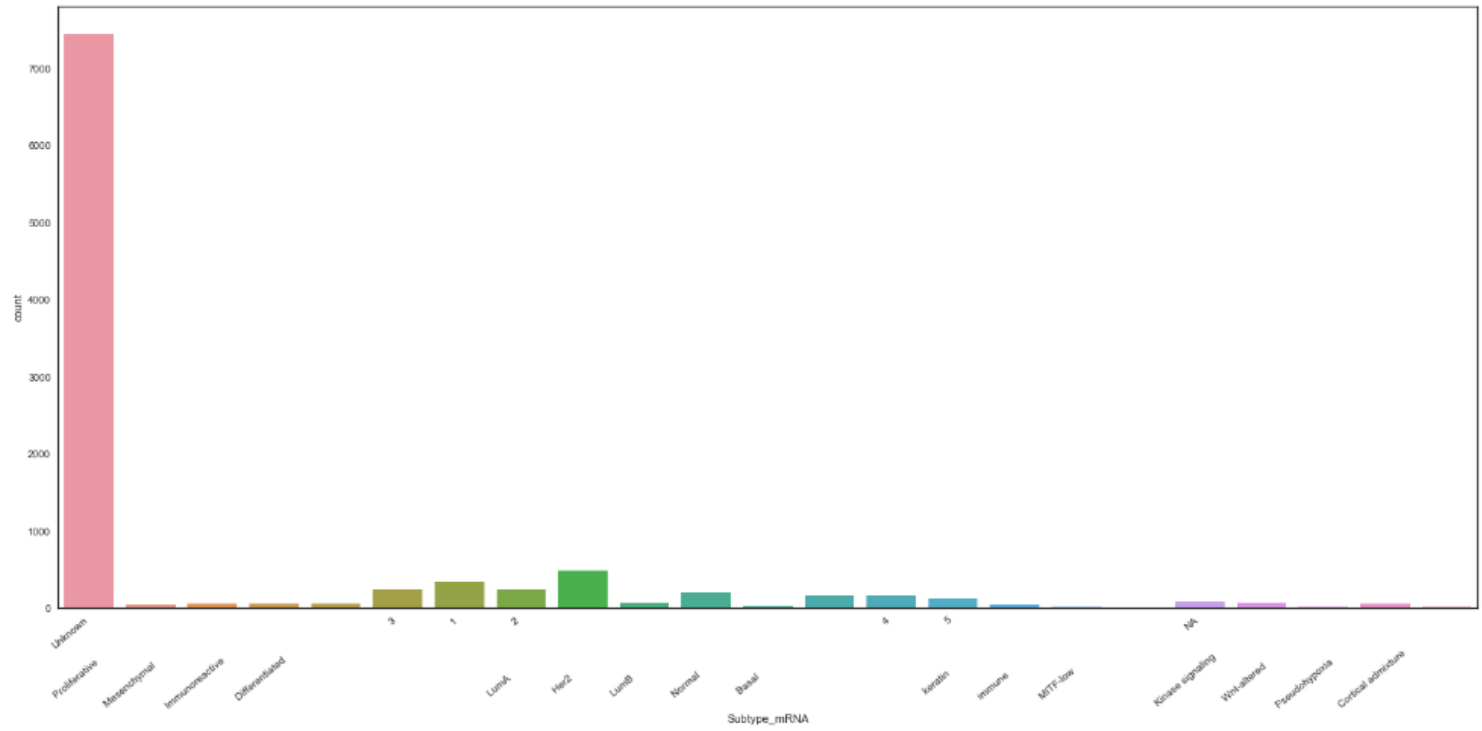


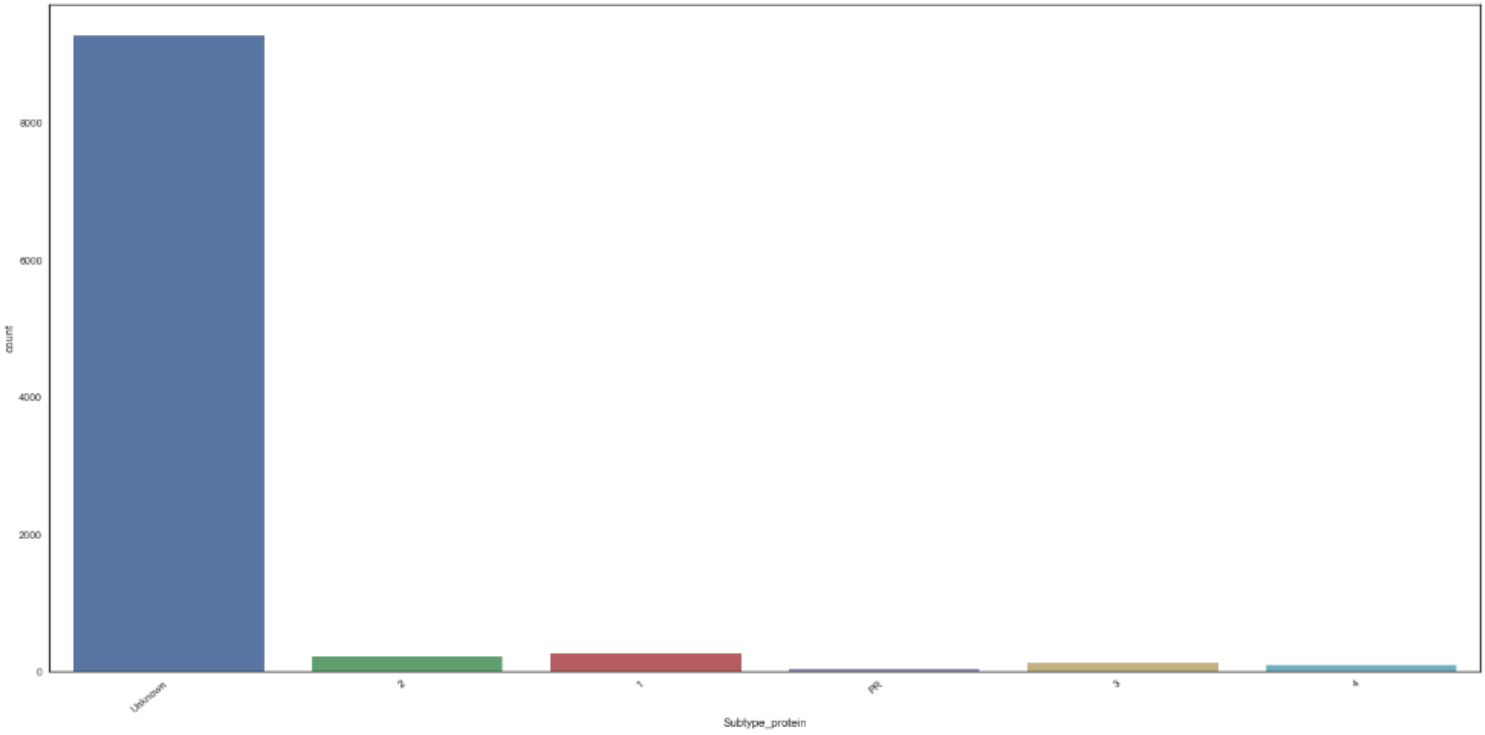
d. Categorical Variables histograms

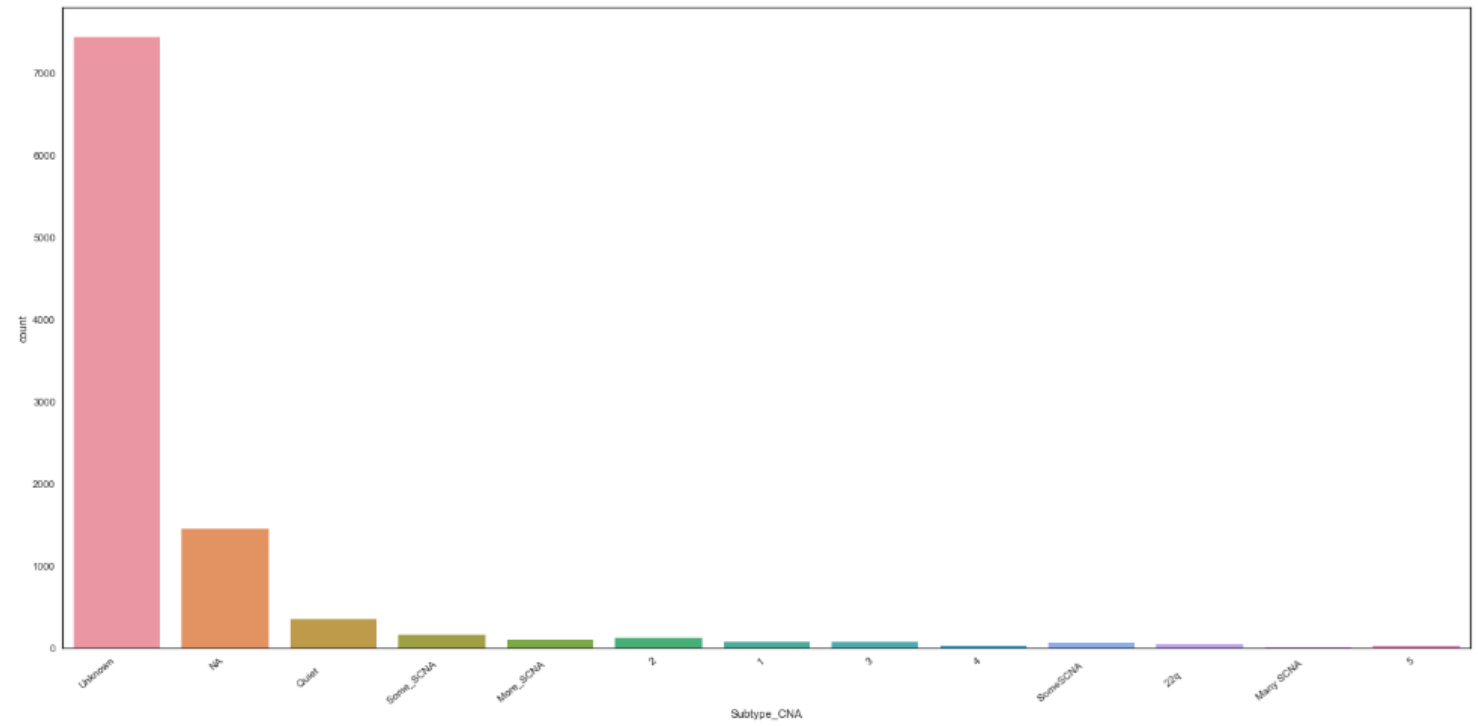


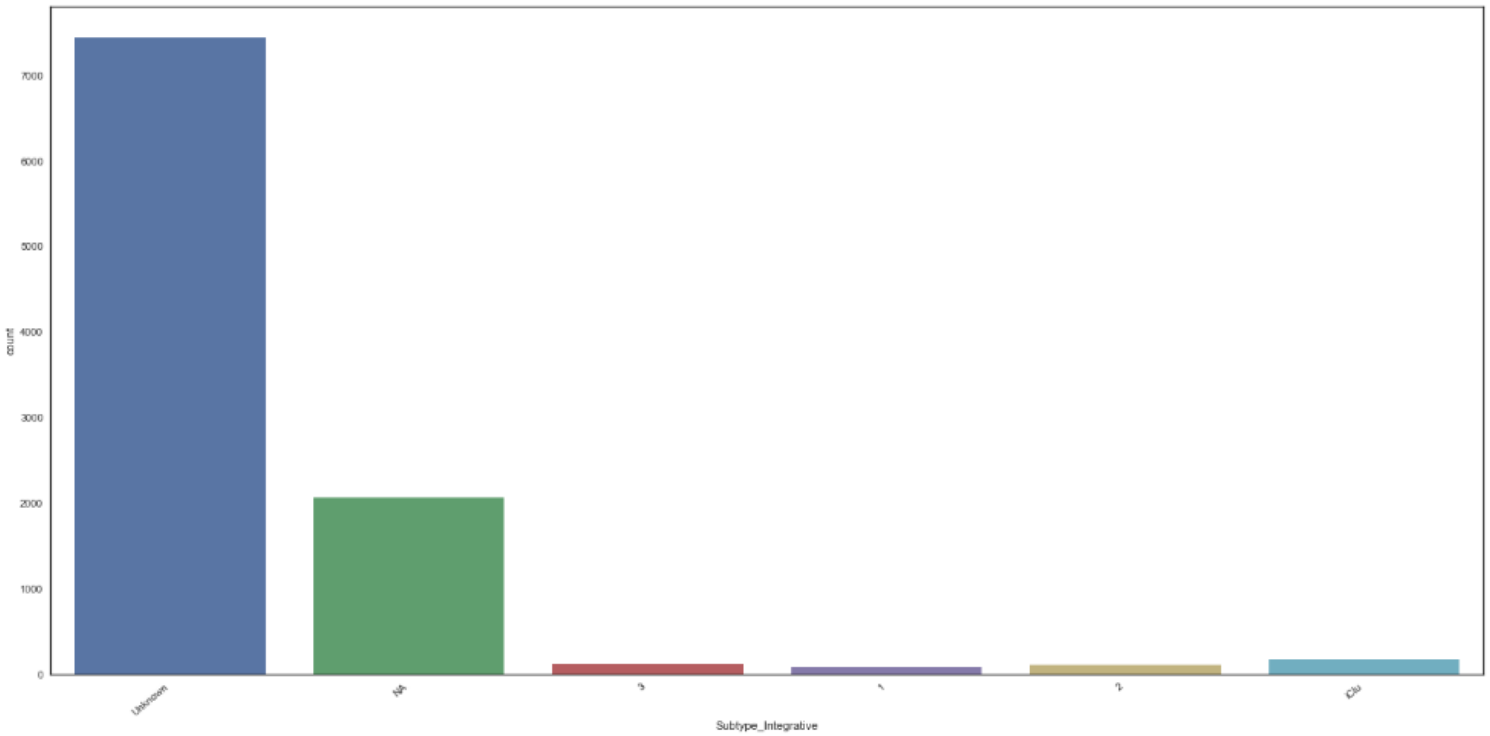

count
C1
Unknown
C*03:03
C*07:02
C*07:01
C*06:02
C*07:04
C*16:01
C*04:01
C*05:01
C*08:02
C*12:02
C*03:04
C*12:03
C*03:67
C*01:02
C*16:02
C*02:02
C*18:01
C*08:01
C*17:01
C*15:02
C*14:02
C*03:02
C*16:04
C*02:10
C*08:04
C*15:04
C*15:05
C*07:22
C*04:03
C*15:09
C*03:05
C*06:06
C*07:17
C*06:13
C*04:06
C*04:07
C*06:17
C*08:03
C*03:17
C*01:03
C*03:14
C*06:06
C*07:16
C*03:08
C*14:03
C*07:27
C*15:06
C*04:04
C*12:12
C*07:05
C*08:15
C*03:06
C*15:03
C*04:27

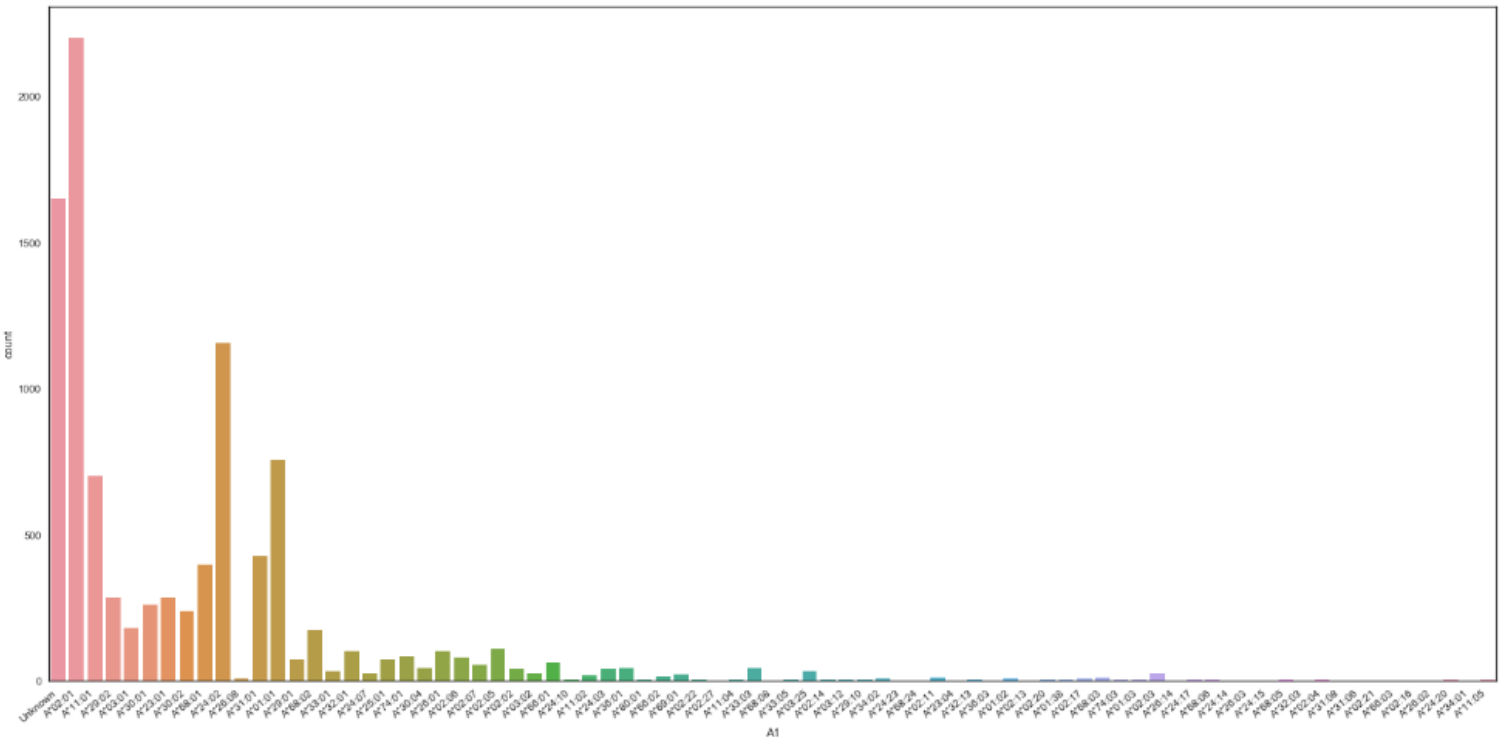

count
C2
Unknown
C*03:03
C*12:02
C*16:01
C*07:02
C*06:02
C*12:03
C*03:04
C*07:01
C*01:02
C*02:02
C*05:01
C*15:02
C*08:02
C*04:01
C*16:04
C*02:10
C*14:02
C*08:01
C*17:01
C*16:02
C*04:03
C*15:05
C*07:04
C*03:02
C*14:03
C*08:04
C*07:22
C*04:04
C*08:03
C*18:01
C*12:04
C*03:06
C*04:07
C*01:30
C*07:46
C*01:17
C*06:06
C*07:05
C*07:17
C*15:04
C*03:14
C*07:10
C*02:06
C*01:03
C*04:33
C*03:08
C*15:09
C*15:13
C*07:66
C*04:10
C*07:60
C*05:09
C*04:06
C*14:04
C*04:14
C*07:13
C*03:05
C*06:15
C*07:27
C*03:07
C*03:19
C*06:03

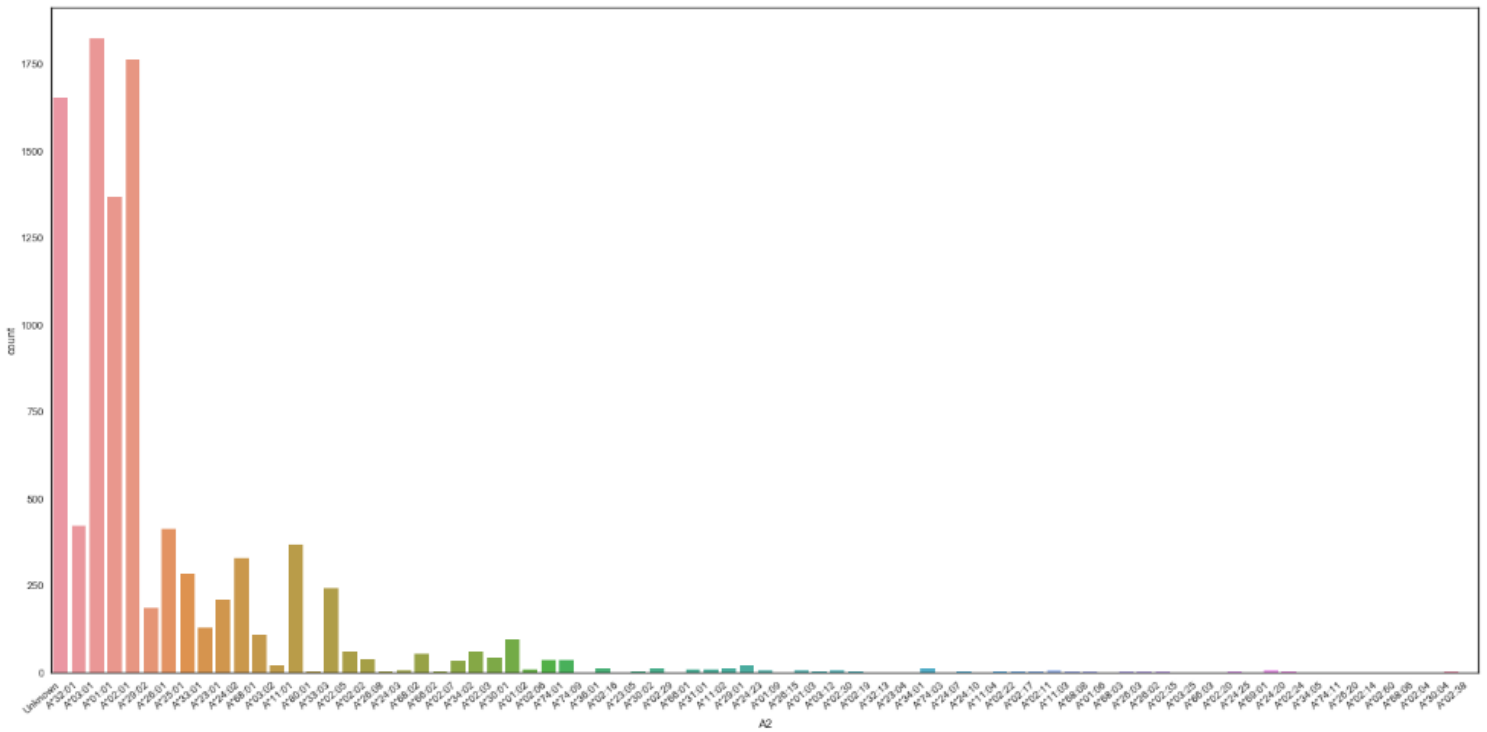

count
7000
6000
5000
4000
3000
2000
1000
0
Unknown
Proliferative
Mesenchymal
Immunoreactive
Differentiated
3
1
LumA
2
Her2
LumB
Normal
Basal
4
Keratin
5
immune
MTF-low
NA
Kinase signaling
Wnt-altered
Pseudohypoxia
Cortical admixture
Subtype_mRNA

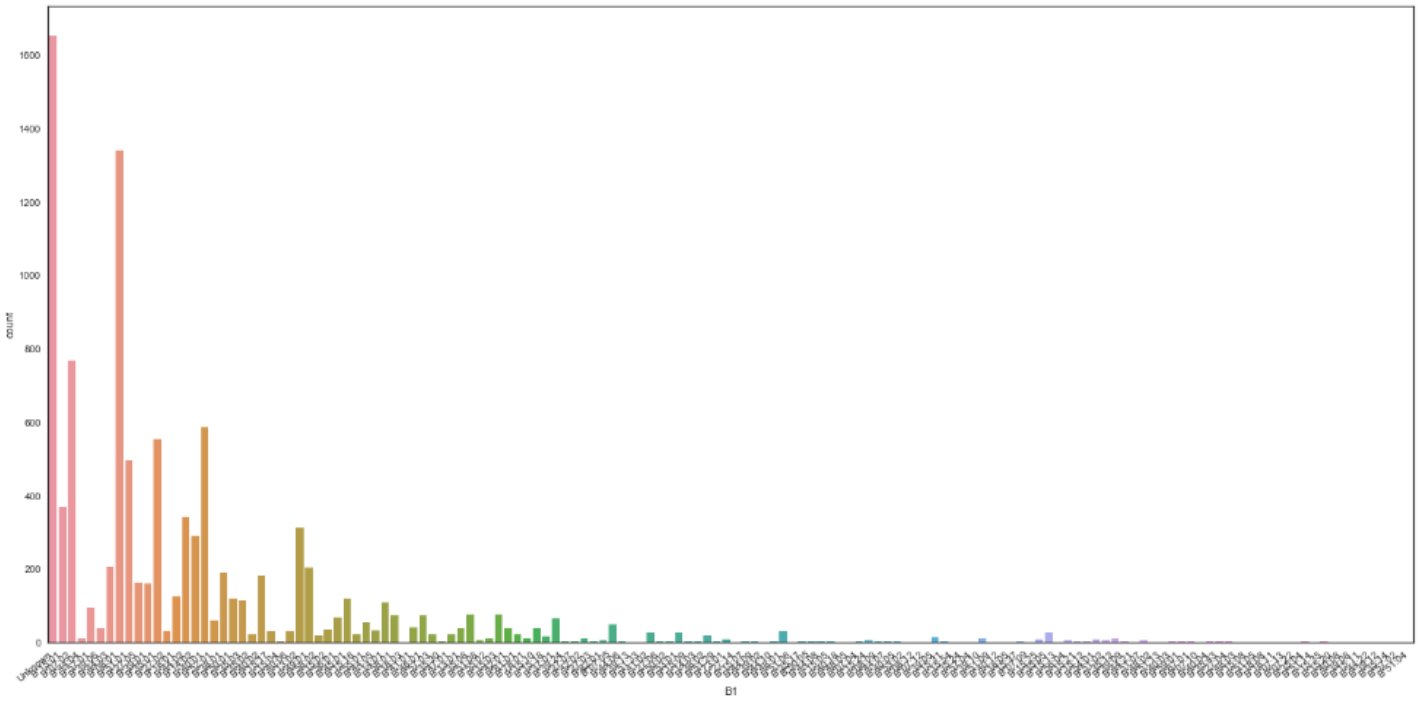

count
8000
6000
4000
2000
0
Unknown
2
1
PR
3
4
Subtype_protein

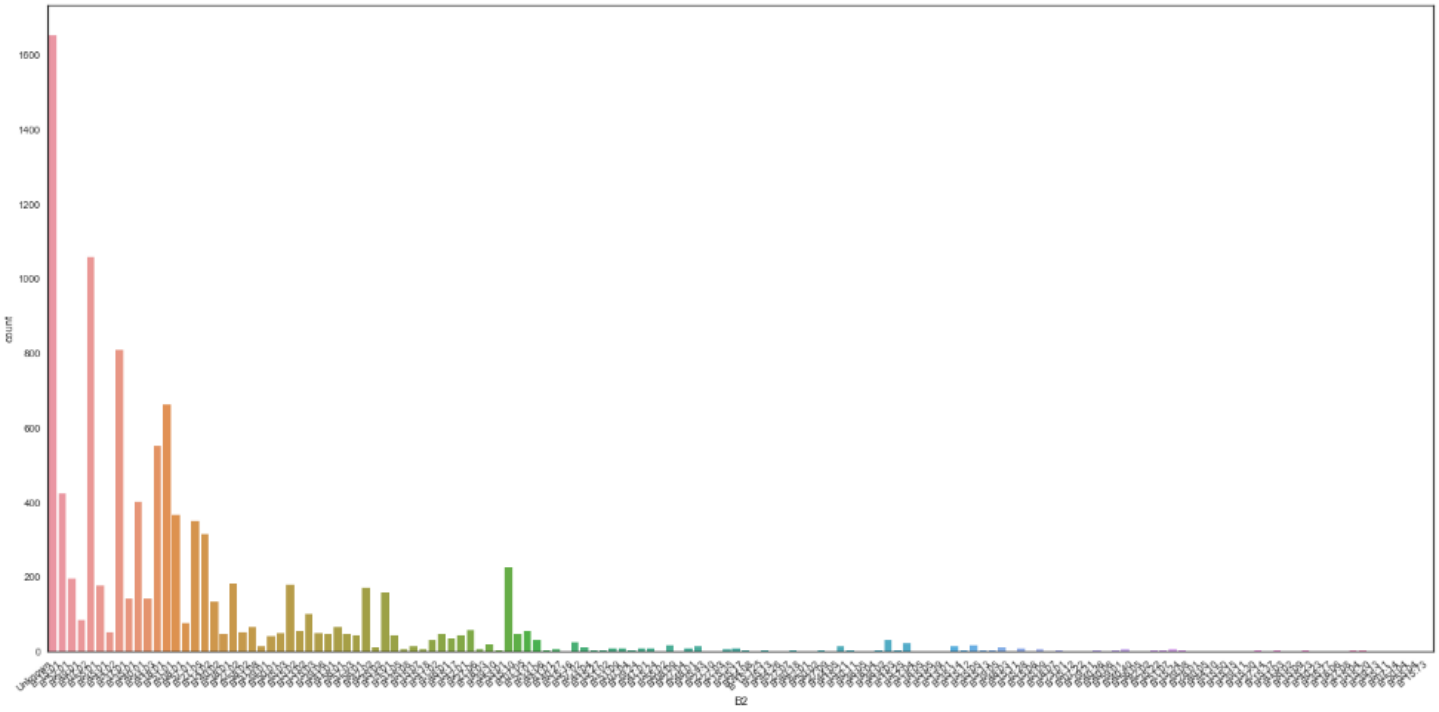

count
7000
6000
5000
4000
3000
2000
1000
0
Unknown
NA
Quiet
Some_SCNA
More_SCNA
2
1
3
4
SomeSCNA
22q
Many SCNA
5
Subtype_CNA

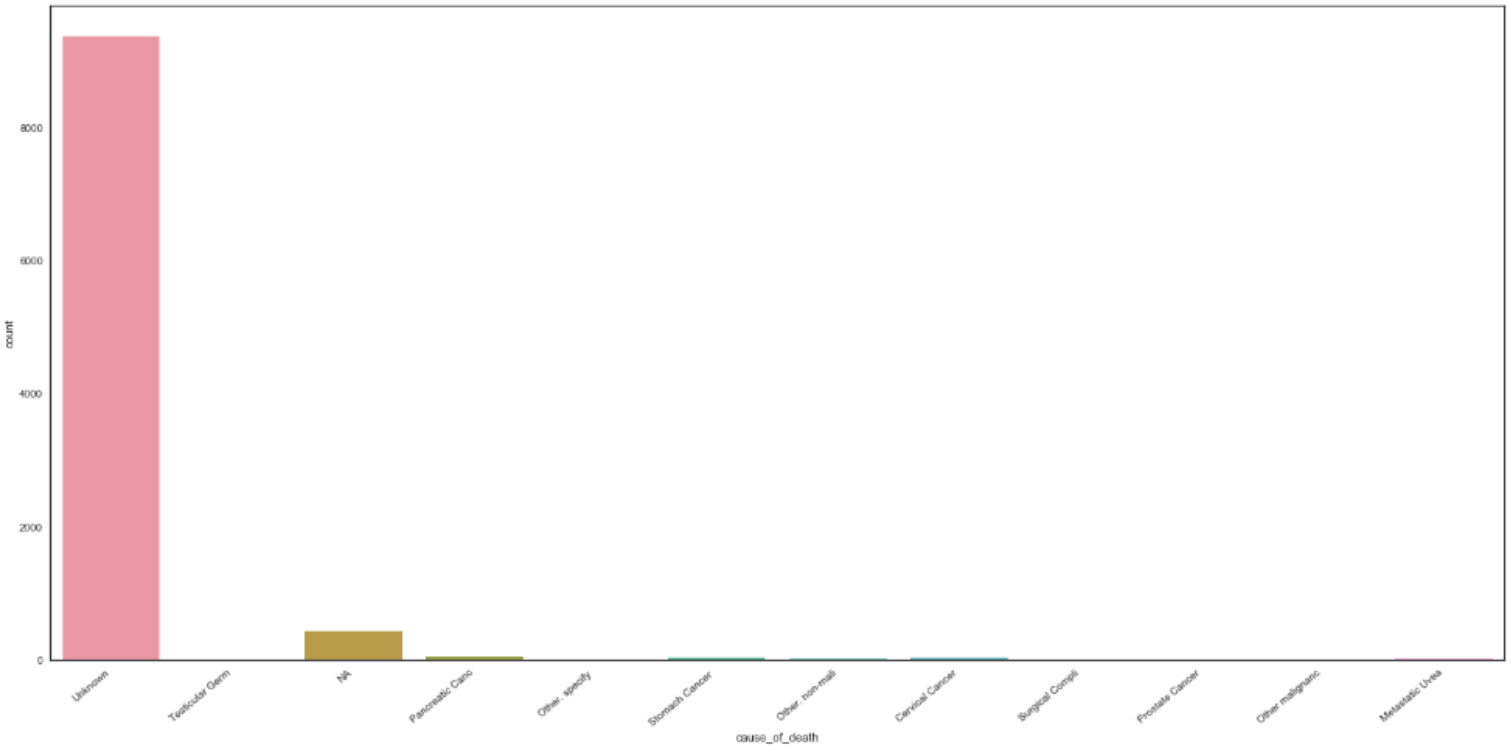

count
7000
6000
5000
4000
3000
2000
1000
0
Unknown
NA
3
1
2
iClu
Subtype_Integrative

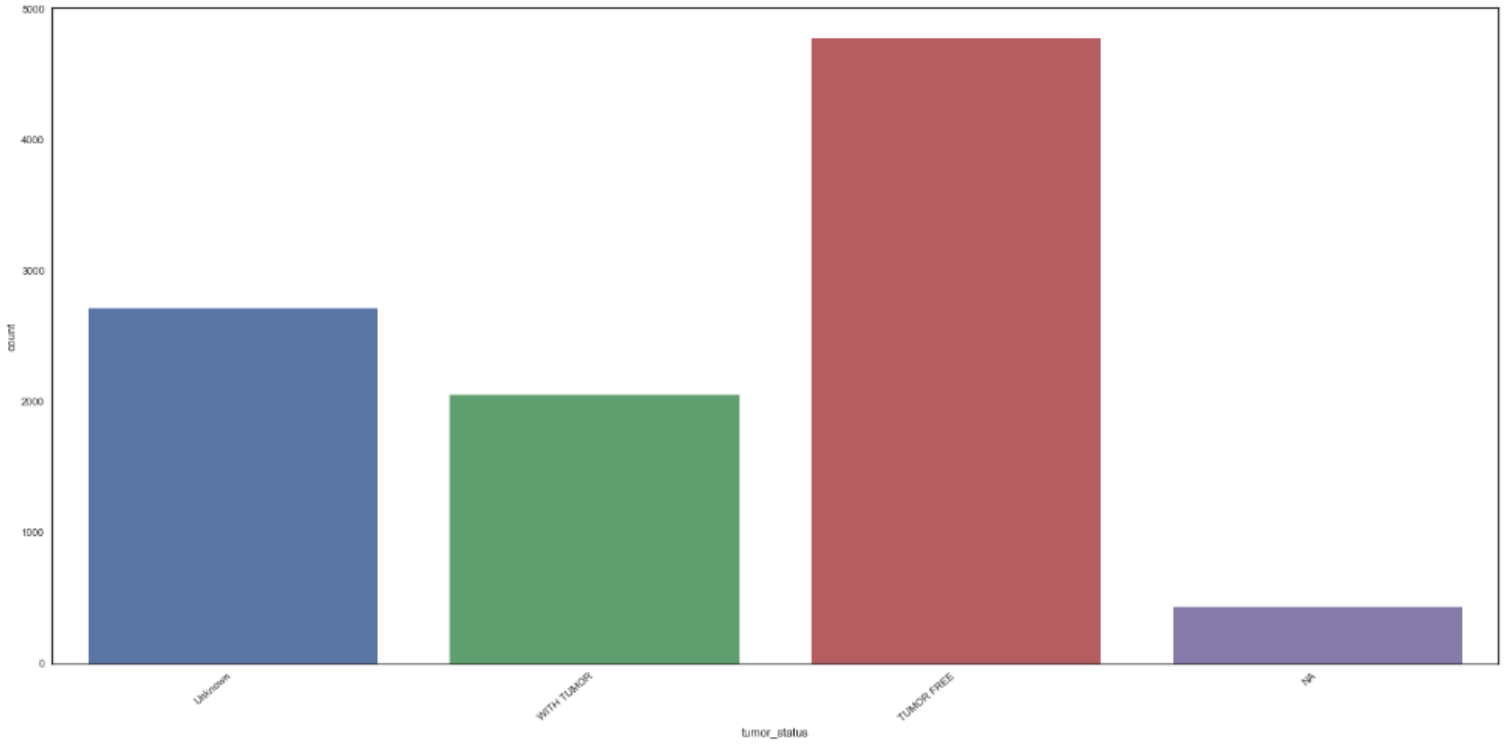

count
2000
1500
1000
500
0
Unknown
A*02:01
A*11:01
A*29:02
A*03:01
A*30:01
A*23:01
A*30:02
A*68:01
A*23:02
A*26:08
A*31:01
A*01:01
A*29:01
A*68:02
A*33:01
A*32:01
A*24:07
A*25:01
A*74:01
A*30:04
A*26:01
A*02:06
A*02:07
A*02:05
A*03:02
A*03:02
A*66:01
A*29:10
A*11:02
A*24:03
A*36:01
A*80:01
A*66:02
A*69:01
A*02:22
A*02:27
A*11:04
A*33:09
A*68:04
A*33:05
A*02:29
A*02:14
A*03:12
A*29:10
A*34:02
A*24:23
A*68:24
A*02:11
A*23:04
A*32:19
A*36:03
A*01:02
A*02:13
A*02:20
A*01:08
A*02:17
A*68:03
A*74:03
A*01:03
A*02:03
A*26:14
A*24:17
A*68:06
A*24:14
A*26:03
A*24:15
A*66:05
A*32:03
A*02:04
A*31:06
A*02:21
A*66:03
A*02:16
A*29:02
A*24:20
A*34:01
A*11:05
A1

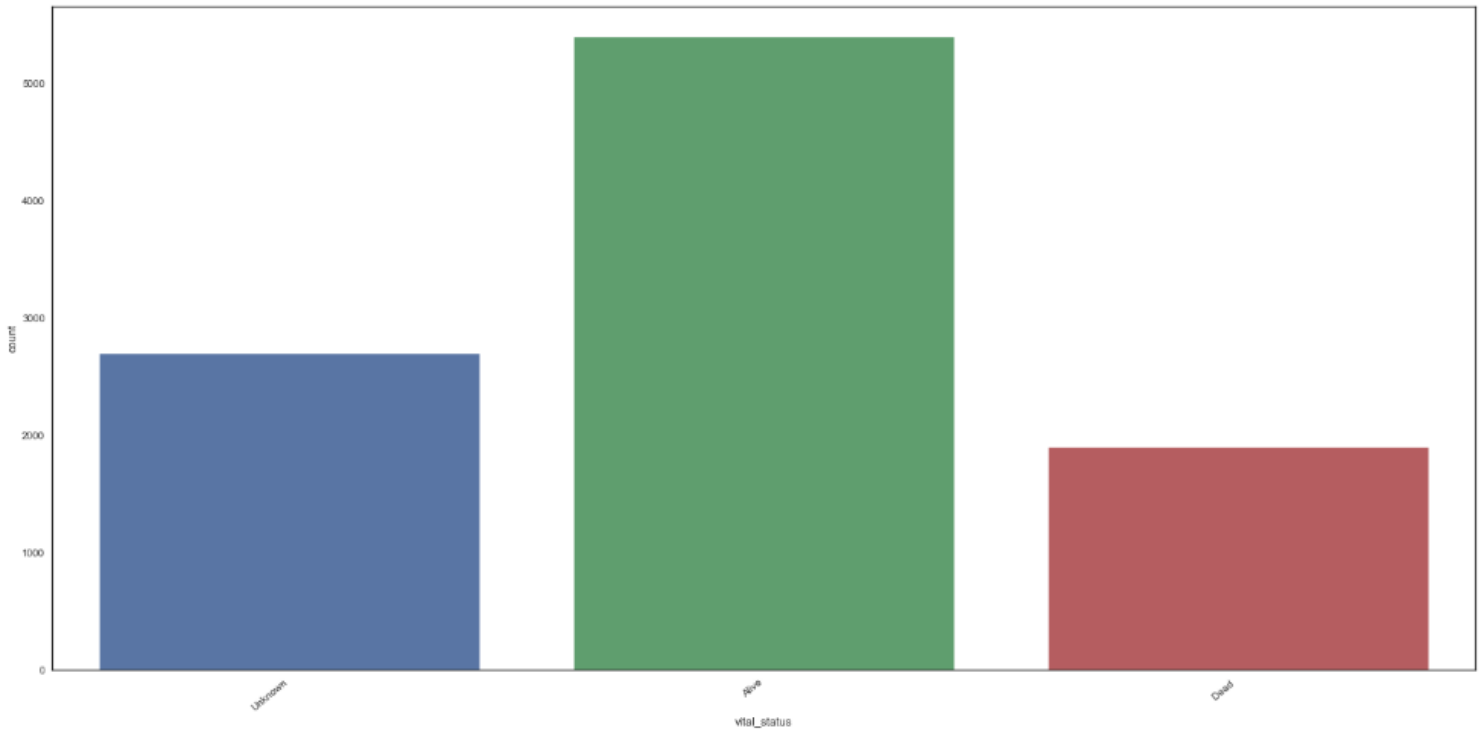

count
1750
1500
1250
1000
750
500
250
0
Unknown
A*02:01
A*03:01
A*01:01
A*02:01
A*29:02
A*26:01
A*23:01
A*03:01
A*24:02
A*68:01
A*03:02
A*11:01
A*80:01
A*33:03
A*02:05
A*02:03
A*26:08
A*24:03
A*68:02
A*02:07
A*34:02
A*02:09
A*30:01
A*01:02
A*02:06
A*74:01
A*74:03
A*36:01
A*02:16
A*23:05
A*30:02
A*02:29
A*68:01
A*31:01
A*11:02
A*29:01
A*24:23
A*01:09
A*26:15
A*01:03
A*03:13
A*02:30
A*02:19
A*32:13
A*23:04
A*34:01
A*74:03
A*24:07
A*24:10
A*11:04
A*02:22
A*02:17
A*11:03
A*68:06
A*01:06
A*68:03
A*25:03
A*26:02
A*02:35
A*03:25
A*66:03
A*02:20
A*24:25
A*69:01
A*24:29
A*02:34
A*24:05
A*74:11
A*25:29
A*02:14
A*68:08
A*02:04
A*30:04
A*02:39
A2

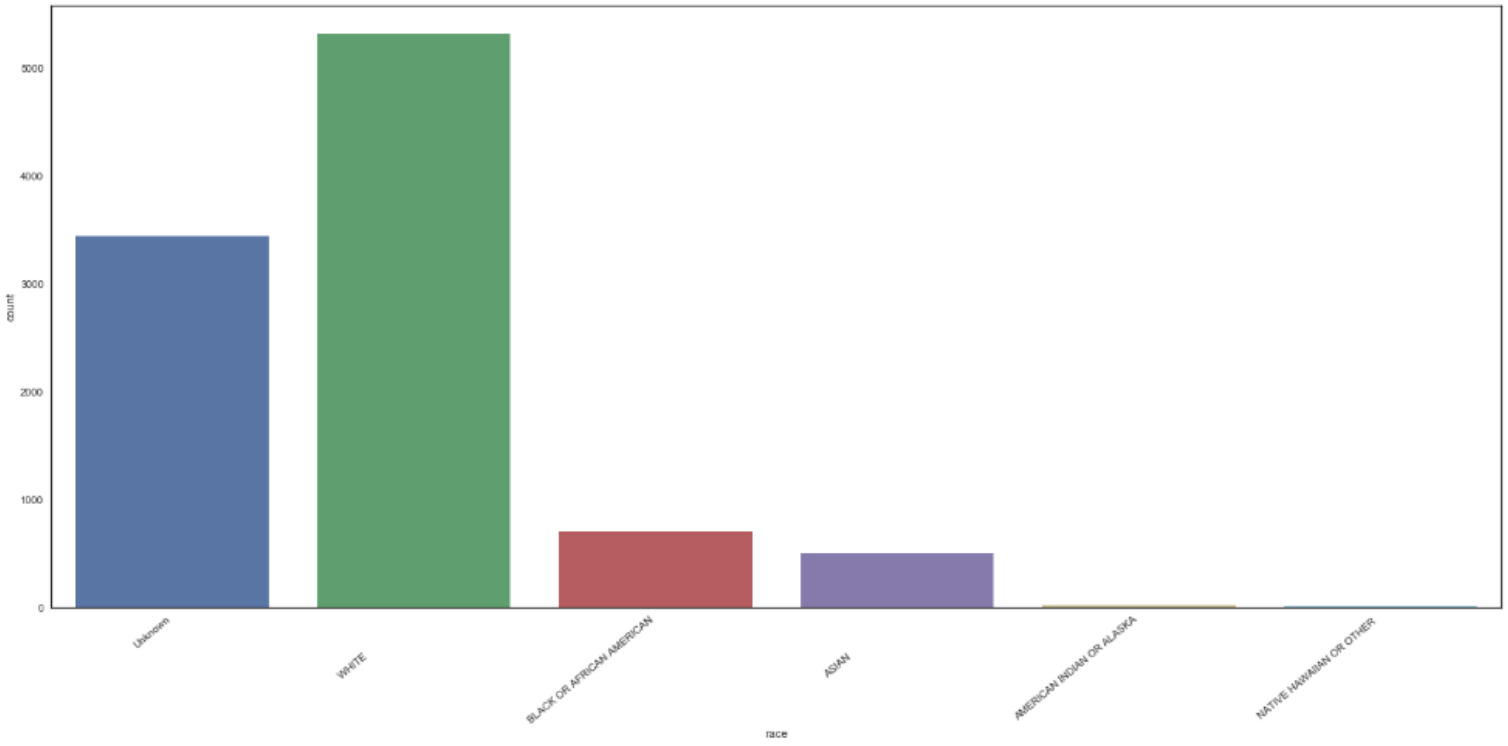

count
B1

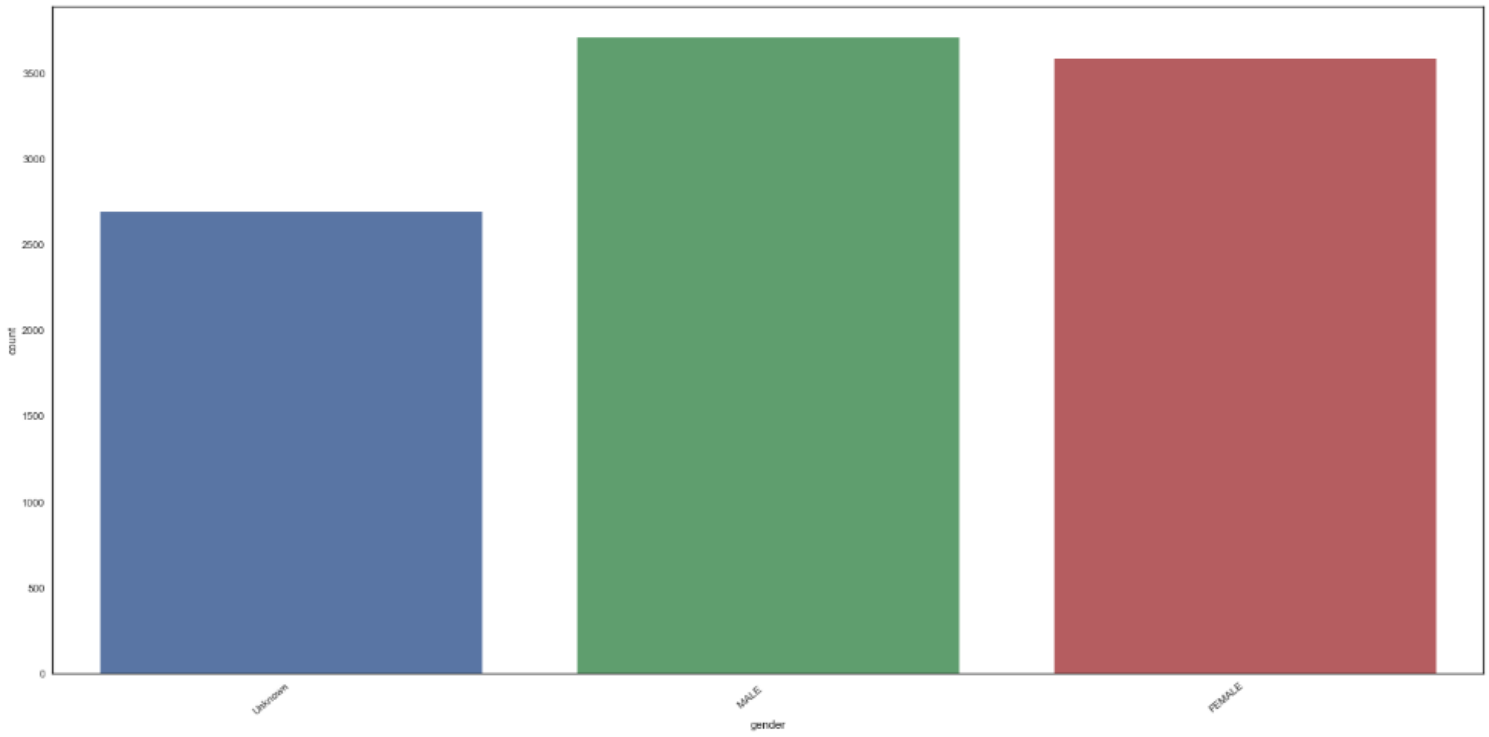

count
B2

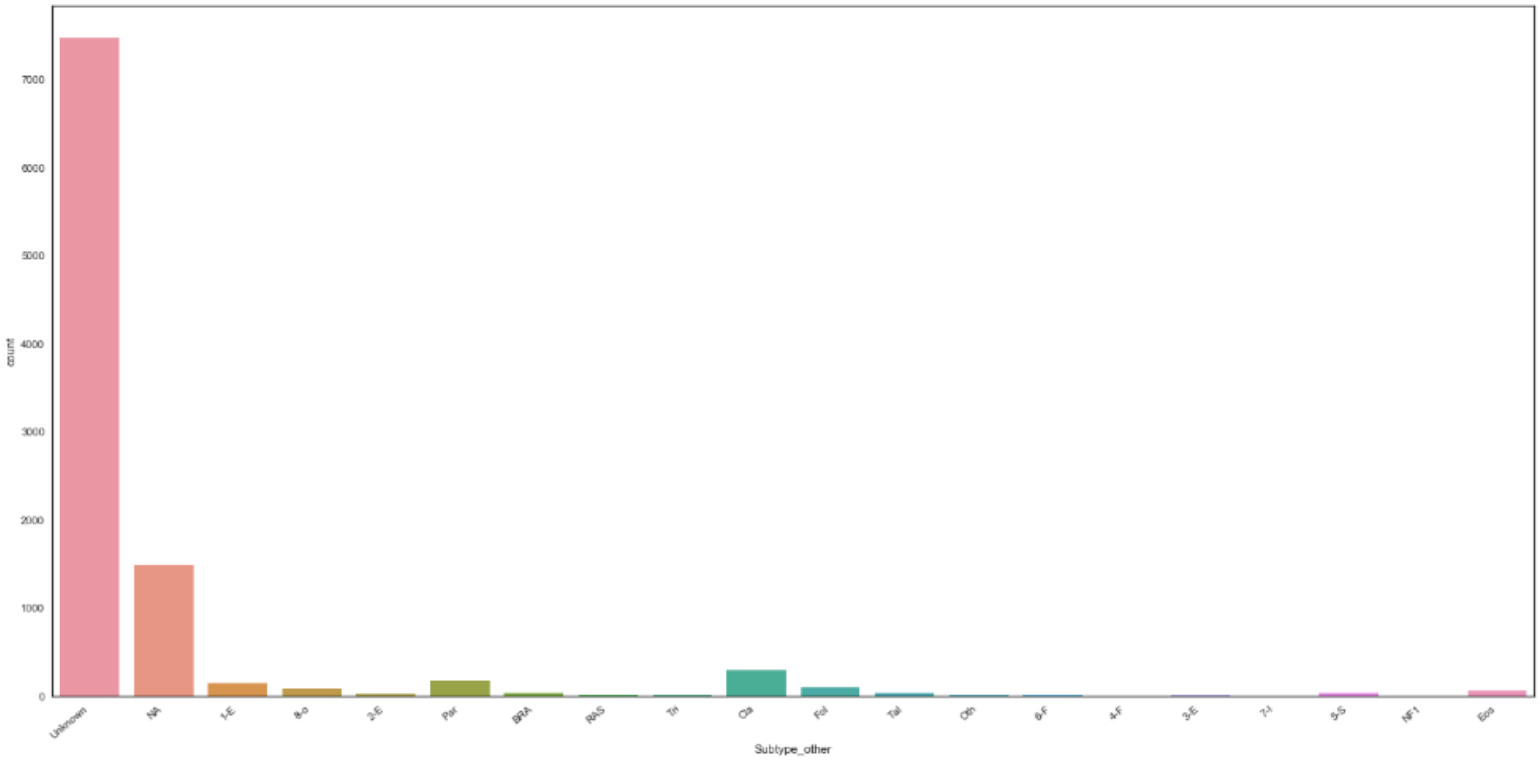

8000
6000
4000
2000
0
count
Unknown
Testicular Germ
NA
Pancreatic Canc
Other, specify
Stomach Cancer
Other, non-mali
Cervical Cancer
Surgical Compli
Prostate Cancer
Other malignanc
Metastatic Uvea
cause_of_death

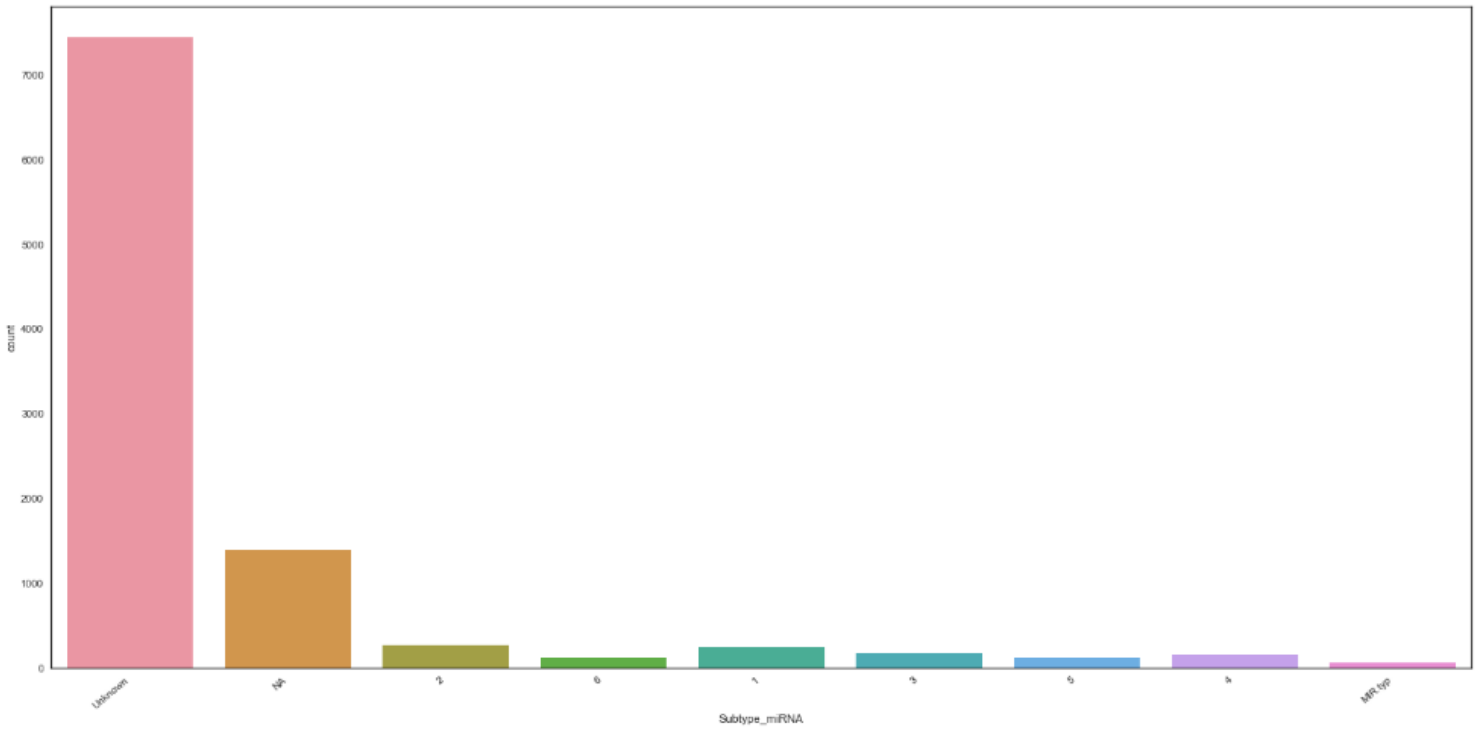

5000
4000
3000
2000
1000
0
count
Unknown
WITH TUMOR
TUMOR FREE
NA
tumor_status

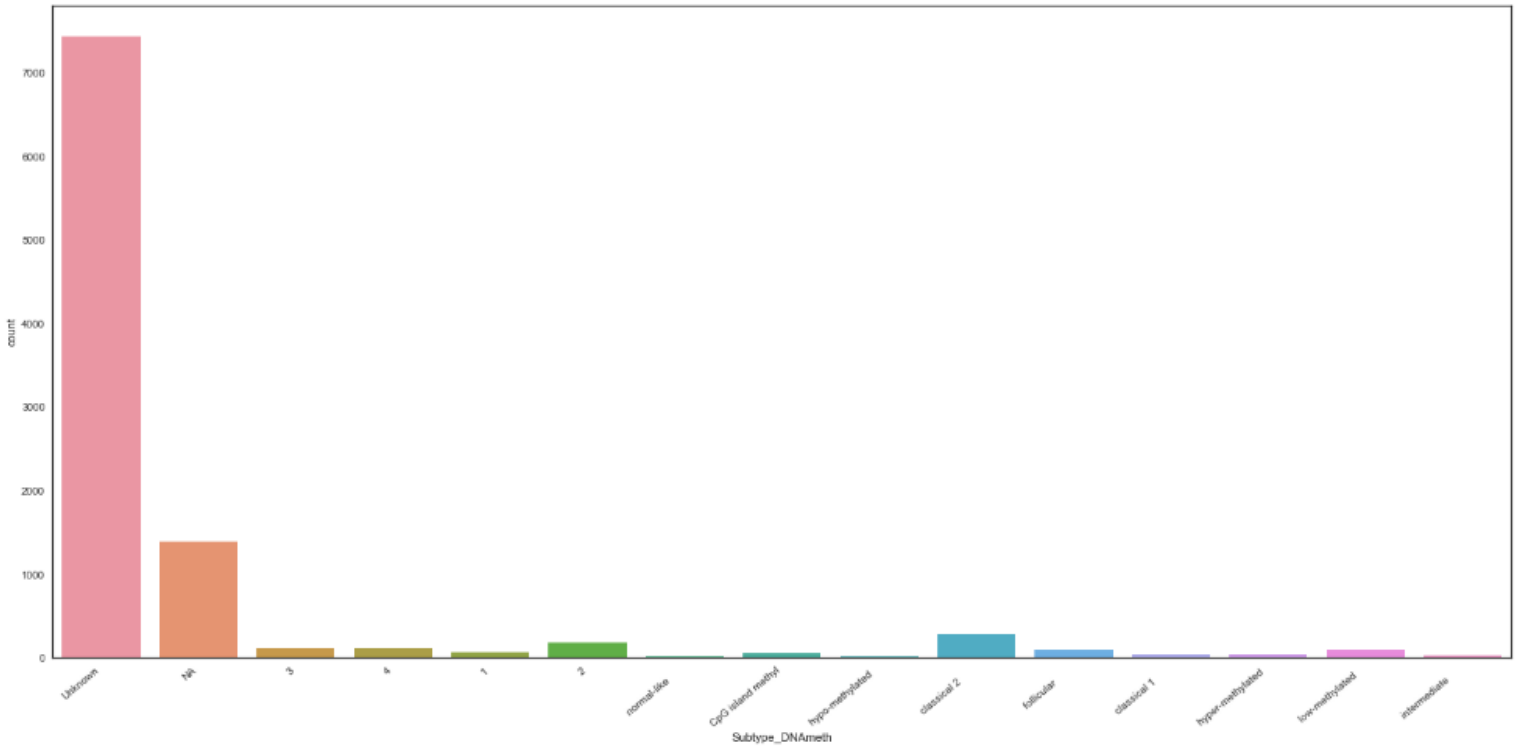

5000
4000
3000
2000
1000
0
count
Unknown
Alive
Dead
vital_status

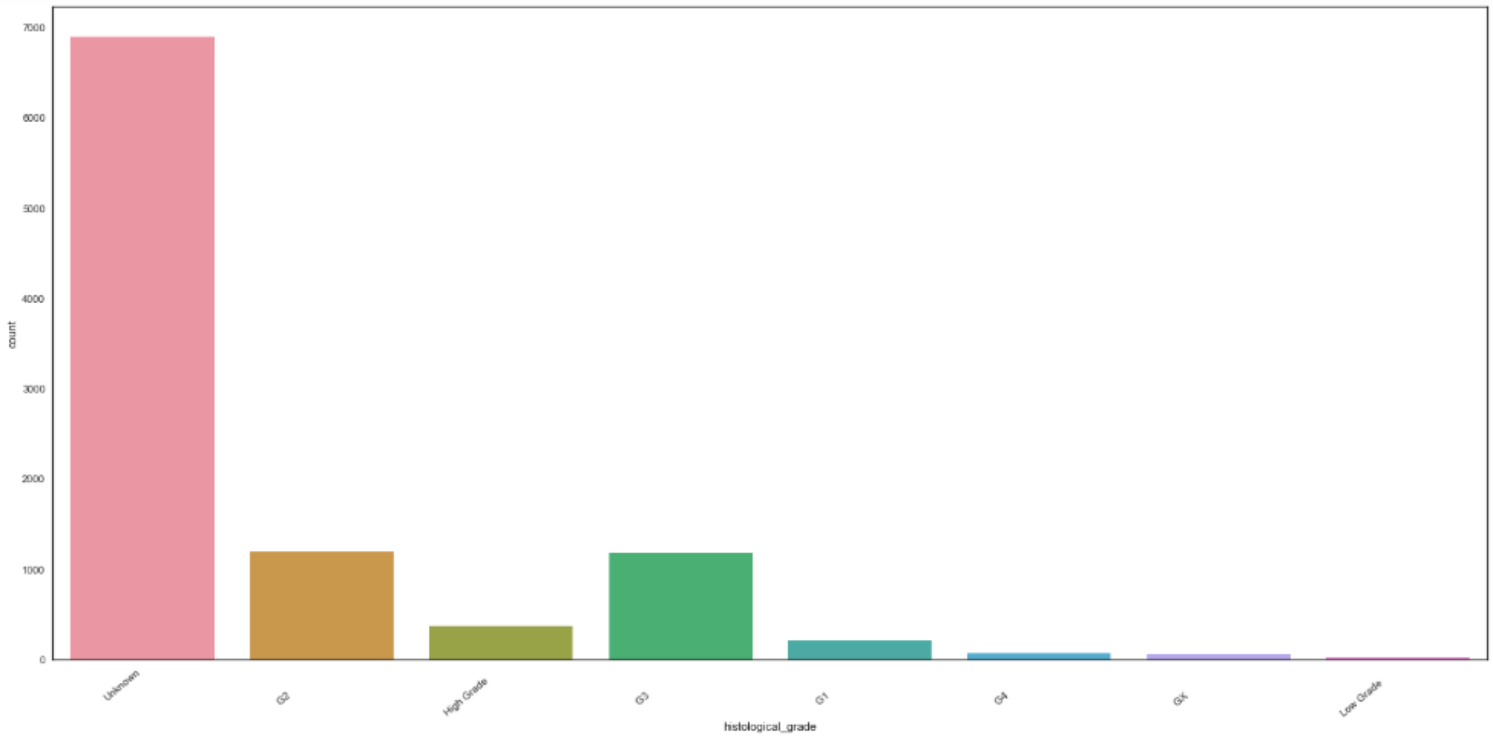

5000
4000
3000
2000
1000
0
count
Unknown
WHITE
BLACK OR AFRICAN AMERICAN
ASIAN
AMERICAN INDIAN OR ALASKA
NATIVE HAWAIIAN OR OTHER
race

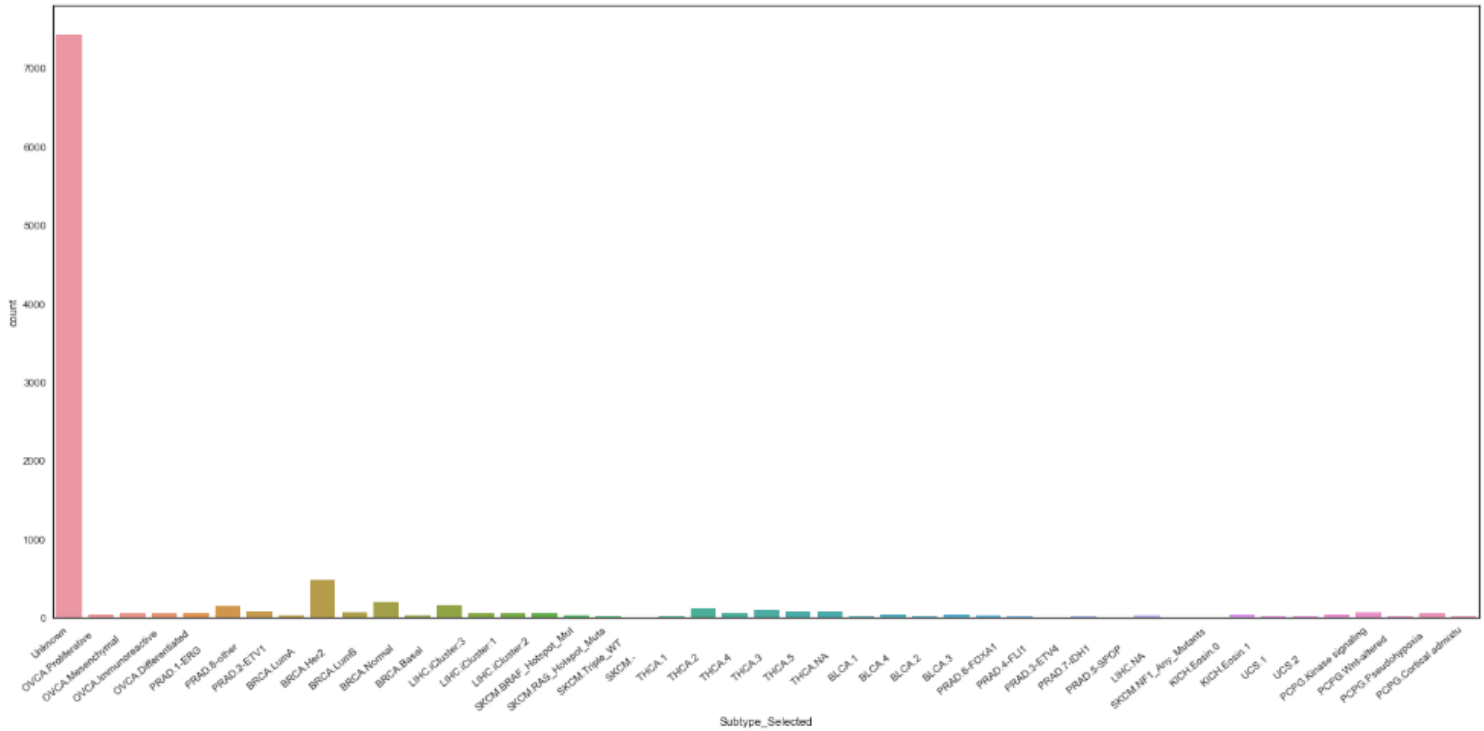

3500
3000
2500
2000
1500
1000
500
0
count
Unknown
MALE
FEMALE
gender

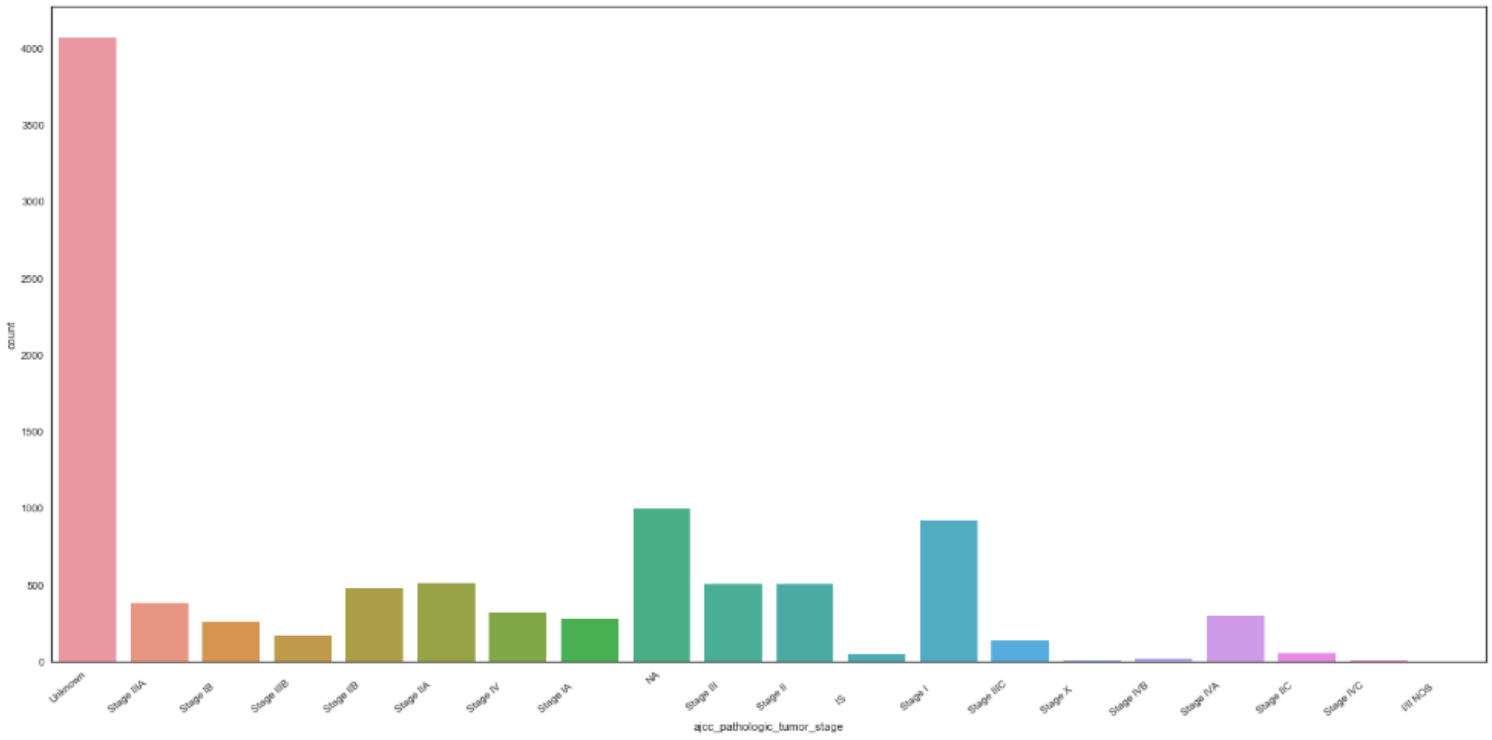

7000
6000
5000
4000
3000
2000
1000
0
count
Unknown
NA
I-E
II-o
2-E
Par
BRA
RAS
Tri
Cla
Fol
Tal
Oth
6-F
4-F
3-E
7-I
5-S
NF1
Eos
Subtype_other

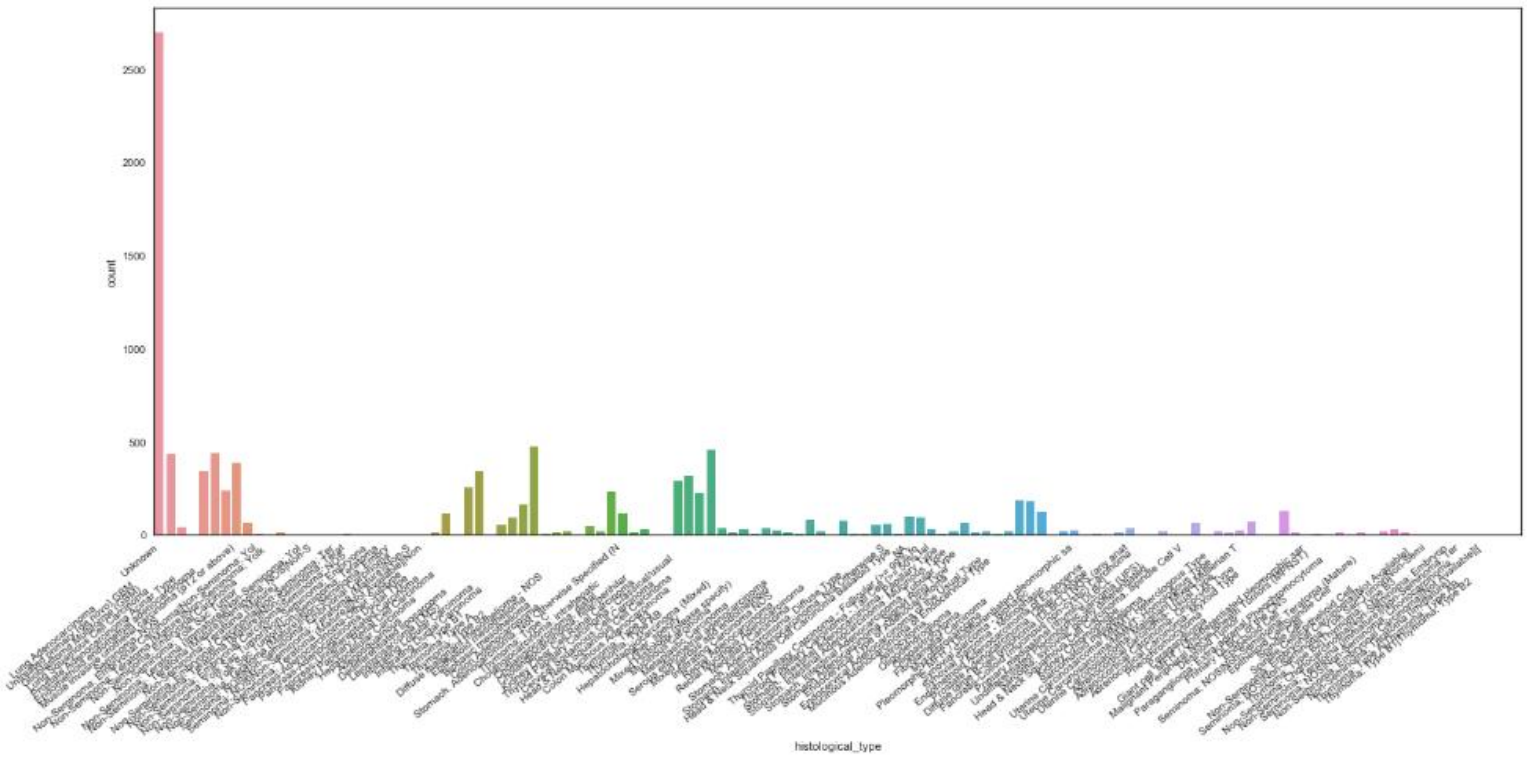

count
7000
6000
5000
4000
3000
2000
1000
0
Unknown
NA
2
6
1
3
5
4
MR typ
Subtype_miRNA


count
7000
6000
5000
4000
3000
2000
1000
0
Unknown
NA
3
4
1
2
normal like
CpG island methyl
hypo-methylated
classical 2
follicular
classical 1
hyper-methylated
low-methylated
intermediate
Subtype_DNAmeth


count
7000
6000
5000
4000
3000
2000
1000
0
Unknown
G2
High Grade
G3
G1
G4
GX
Low Grade
histological_grade


count
7000
6000
5000
4000
3000
2000
1000
0
Unknown
OVCA.Proliferative
OVCA.Mesenchymal
OVCA.Immunoreactive
OVCA.Differentiated
PRAD.1-ERG
PRAD.6-other
PRAD.2-ETV1
BRCA.LumA
BRCA.Her2
BRCA.LumB
BRCA.Normal
BRCA.Basal
LIHC.iCluster.3
LIHC.iCluster.1
LIHC.iCluster.2
SKCM.BRAF_Hotspot_Mut
SKCM.RAS_Hotspot_Muts
SKCM.Triple_WT
SKCM.-
THCA.1
THCA.2
THCA.4
THCA.3
THCA.5
THCA.NA
BLCA.1
BLCA.4
BLCA.2
BLCA.3
PRAD.6-FOXA1
PRAD.4-FLI1
PRAD.3-ETV4
PRAD.7-IDH1
PRAD.5-SPOP
LIHC.NA
SKCM.NF1_Any_Mutants
KICH.Exon.0
KICH.Exon.1
UCS.1
UCS.2
PCPG.Kinase signaling
PCPG.Wnt-altered
PCPG.Pseudohypoxia
PCPG.Cortical admixtu
Subtype_Selected


count
4000
3500
3000
2500
2000
1500
1000
500
0
Unknown
Stage IIIA
Stage IB
Stage IIIB
Stage IIB
Stage IIA
Stage IV
Stage IA
NA
Stage III
Stage II
IS
Stage I
Stage IIIC
Stage X
Stage IVB
Stage IVA
Stage IIC
Stage IVC
III NOS
ajcc_pathologic_tumor_stage


count
2500
2000
1500
1000
500
0
Unknown
histological_type