Master Thesis

Developing a pipeline for deep analysis of cancer causing genes in the gut microbiome

Henrik Høybakk Olsvik

Acknowledgements

I would like to thank my advisors Trine Ballestad Rounge and Torbjørn Rognes for their thorough reviews and helpful guidance, without which I would not have found my way in the world of bioinformatics and metagenomics. I would also like to thank Einar Birkeland for his helpfulness in providing me with resources, advice and assistance.

Contents

1 Introduction and Background

The Norwegian Cancer Registry is currently engaging in a study named CRCbiome [1, 2]. The purpose of the CRCbiome study is to identify microbial biomarker genes for colorectal cancer (CRC) and investigate how lifestyle factors modify the microbiome [1]. The study is a sub-study of the Bowel Cancer Screening in Norway (BCSN), a pilot for a nationwide bowel cancer screening program [3]. Participants are recruited from the BCSN study to the CRCbiome study. The CRCbiome study receives stool samples from the recruited participants from the BCSN study. DNA from these collected samples are sequenced and sent through a data pipeline which analyses and assembles the sequences into a metagenomic dataset.

In this introduction we will introduce the key concepts in the field of colorectal cancer screening, the gut microbiome, data pipelining as well as machine learning and classifier analysis as they relate to metagenomic datasets. We will provide an entry into the related scientific literature surrounding these topics. The introduction will also present the plan to apply machine learning methods with data preprocessing on pipelined metagenomic datasets with functional gene annotations from shotgun sequenced samples of the gut microbiome in certain individuals. The purpose of this is gaining information about possible biomarker genes that might assist the detection of colorectal cancer in vivo. We will also examine the performance of classifiers trained with the aforementioned dataset. In addition the results from this the shall be compiled into a series of visualisations for a multi-disciplinary audience.

2 Colorectal Cancer

CRC is a term covering malignant neoplasms of the colon and rectum and is one of the most common forms of cancer in Norway. The age standardized incidence rate of CRC was 89.1 and 70.4 per 100000 person years for men and women respectively in 2018 [4]. This is higher than the global age standardized incidence rate per 100000 which was 23.1 for men and 15.7 for women in 2018 [5]. In Norway CRC caused 14.5% of total cancer mortality in 2017 and in 2018 it constituted 12.7% of all cancers in males and 12.8% of all cancers in females [4].

At which stage CRC is discovered has a large impact on chances for survival. The 5 year survival rate for CRC are around 98% for localised cancers, around 80% for cancers with a regional spread and around 20% for cancers that have spread to distant parts of the body. The numbers are similar for both women and men [4]. Screening leads to earlier detection of cancer, and the effect of screening in reducing CRC mortality has been documented [6]. The European Union recommends colorectal cancer screening in order to discover cancer cases early while still asymptomatic and surgically remove possible precursor lesions of CRC [7].

2.1 Colorectal Cancer Screening

Colorectal cancer screening enables the detection of CRC. In the BSCN study two ways of screening for CRC are applied. Faecal occult blood tests (FOBT), that involve checking for the presence of blood in faeces, and sigmoidoscopy, a minimally invasive medical examination of the rectum and sigmoid colon. Both these approaches have been shown to reduce colorectal cancer mortality [6]. FOBT tests have recently been replaced by faecal immunochemical tests (FIT) which perform better than FOBT tests [8]. In the case of a positive FIT then a colonoscopy is performed. FIT lack sensitivity for the early stages of CRC as well as specificity generally. Sigmoidoscopies may be too invasive and also miss a large portion of cancers [9]. The discovery of biomarker genes for CRC can lead to new approaches for CRC screening. The use of stool sampling for the detection of these genes is a possibility for future non-invasive CRC screening.

3 The Human Gut Microbiome

The human microbiome is the sum of organisms that live in or on humans [10]. The human gut harbors a large subset of the total human microbiome with abundance and genetic characteristics varying between individuals. Each individual has a relatively stable microbial community [11]. These bacteria exist in a commensal and symbiotic manner in our gut and play an important role in human digestion. Interindividual variation in microbial populations is due to both host genetics and environmental influences such as diet, stress, antibiotics, pets and likely other agents [12]. It has been observed that taxonomic distribution of the microbiome in the gut is different between subjects suffering from irritable bowel diseases, such as Crohn's disease and ulcerative colitis, and healthy individuals [13]. The microbial imbalances associated with such diseases is referred to as dysbiosis, defined as "a breakdown in the balance between putative species of "protective" versus "harmful" intestinal bacteria [..]" by Tamboli (2004)[14] [12].

3.1 Role in Cancer

It has been demonstrated that the gut microbiome of people with several forms of cancer, including CRC, differs from that of other individuals. We also know that certain microbes, such as *Streptococcus bovis* and *Clostridium septicum*, are strongly associated with CRC. We know that some single microbes can drive human cancer. Such as *Helicobacter pylori* for gastric adenocarcinoma, gastric lymphoma and esophagael adenocarcinoma, and the human papillomavirus for anogenital carinomas and oropharyngeal carinoma [12].

With the use of machine learning methods, classifiers have been trained that have been able to predict the occurrence of colorectal cancer with good accuracy [15]. Thus further emphasising the association between gut microbiome profile and colorectal cancer.

3.2 Data collection

As a part of the BCSN trial stool were donated by participants. The stool samples were checked for blood. If the samples were found to contain blood then the participants were referred on to colonoscopies. These collected stool samples contain an environmental footprint of the gut microbiome. The samples that were collected as a part of the BCSN trial were found to harbor diverse microbial profiles be usable for metagenomic sequencing and analysis [16].

A total of 2700 candidates that were found to have blood in their sample will invited to participate. The samples from the willing candidates constitute the data collected to be sequenced and analysed in the CRCbiome project and that this master thesis will utilize. Out of this group some of the participants will be found to have advanced neoplasias after a colonoscopy while others will not. Participants without advanced neoplasias will be selected as controls as they have a very low risk of developing CRC following a negative colonoscopy [17, 18].

3.3 Metagenomics and Metagenomic Data

Metagenomics as defined by Riesenfeld et al. 2004[19] is "[..] the functional and sequence-based analysis of the collective microbial genomes contained in an environmental sample". Metagenomics and metagenomic methods have enabled the study of a large amount of previously hard to study organisms due to them being difficult to cultivate in vitro [20, 21].

Metagenomic data is the data gathered from collective microbial genomes in an environmental sample. They contain the full breadth and diversity of the microbiome present in the host environment. Our metagenomic data is the data sequenced from the samples from the BCSN project [22].

A key term in relation to metagenomics is gene function. Gene function is the function a gene causes when it is expressed. Gene function analysis is the analysis of the functions of an organism based to the genes present in it. With metagenomic datasets we have a large set of microorganisms. By way of analyzing and examining the genes present in the sample, and their known functions, it is possible to gain insight into the total behaviour and function of entities contained in it.

3.4 Taxonomy

Taxonomy is a common term in biology and is the practice of naming, describing and classifying different organisms. Individual organisms can be grouped together into a taxon, a group of organisms recognized to form a unit, and these can be classified hierarchically by taxonomic rank. These ranks are, in hierarchical order, domain, kingdom, phylum, class, order, family, genus and species. Taxonomic profiling is the process of quantifying the taxa observed in order to create an overview of the distribution of taxa, a taxonomic profile, of the sample [23]. Much of the former work on machine learning on the gut microbiome has made use of taxonomic profiles [24]. Exact classification of each individual sequenced item might not be possible, but due to the homologous nature of the organisms in the gut microbiome they can generally be assigned to a higher taxonomic group or be clustered together with similar sequences. Clustering of similar sequences creates taxonomic groups referred to as operational taxonomic units (OTUs) [25].

3.5 Sequencing

Sequencing is the process of reading and storing the genetic code of organisms. DNA from the collected environmental samples need to be sequenced in order to be processed.

Two common approaches for sequencing microbiome are 16S rRNA gene amplification and whole genome shotgun sequencing. Targeted 16S rRNA gene amplification and sequencing entails the selective amplification of the 16S rRNA gene. 16S enables the clustering of the obtained sequences into operational taxonomic units (OTUs), by way of the variations in the 16S rRNA gene, from which the taxonomy of each OTU can be inferred [26]. Shotgun sequencing entails the sequencing of small sections of the DNA from the genomes of all organisms in the sample. During subsequent processing information can be gained about taxonomy as with 16S rRNA amplification. Unlike 16S rRNA amplification shotgun sequencing includes more than just the 16S gene, but rather representative fragments of the genes present in the organisms in the sample. That offers the possibility of reassembly and the use of database lookups on the genes which yields functional insight [27]. Shotgun sequencing thus enables functional annotation of the dataset for information about community-wide gene composition and function, which is relevant for this thesis [26].

The CRCbiome project conducts whole genome shotgun sequencing through a third party provider on the collected samples which yields a multitude of small, essentially random, reads of the genomes. This is done by first shearing the sample into appropriately small chunks and annotating them with specific barcodes in order to keep track of the sample they originate from. Adapters for the facilitation of sequencing are also added. The samples are subsequently sequenced via a high-throughput sequencing machine. For each individual DNA fragment there are two individual reads going in opposite direction. Each individual read generated is 100 base pairs long. The reads are annotated with a Phred[28] quality score which gives a probabilistic correctness rating to each sequenced base pair. These reads are stored in the FASTQ[29] format.

The reads are returned from the third party provider demultiplexed whereby the reads have been grouped with the other reads from their respective original samples. The barcodes that signify which sample each individual read originated from are also removed as a part of that process.

4 The Data Pipeline

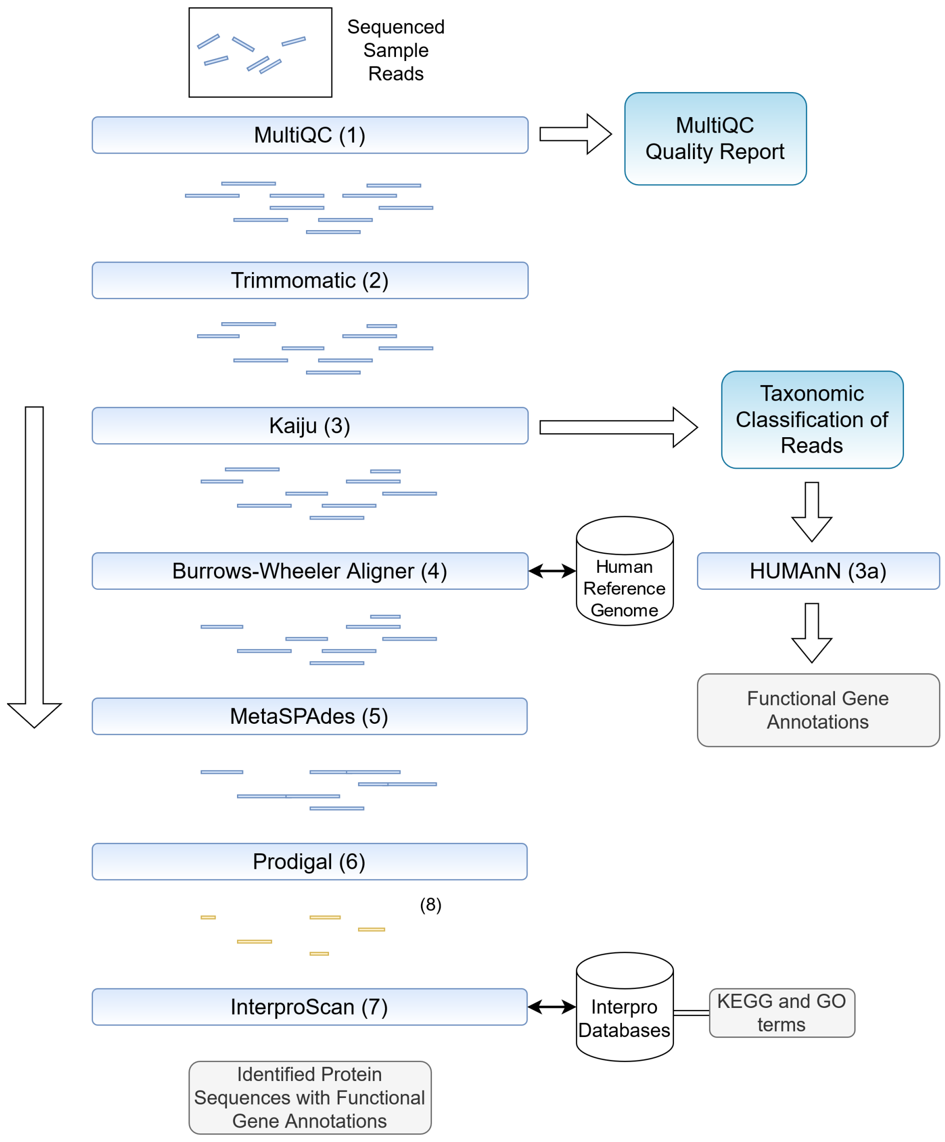

Figure 1: Data Pipeline Overview

(1) - MultiQC, a quality control tool. (2) - Trimmomatic, removes low quality sequences. (3) - Kaiju, performs taxonomic classification. (3a) - HUMAnN, does gene function abundance profiling. (4) - BWA, removes human reads. (5) - MetaSPAdes, tool for sequence assembly. (6) - Prodigal, protein encoding prediction. (7) - InterProScan, database protein lookup tool, (8) - Yellow colour indicates protein sequences

As a part of the CRCbiome project several data processing pipelines have been constructed and combined to handle incoming sequenced sample data. This section will give a board overview of the structure of the combined pipeline, a short summary of the steps involved and describe the structure of the resulting dataset. A more in-depth look at the pipeline will be available in forthcoming publications on the work of the CRCbiome team. The pipeline is implemented in Snakemake v5.3.0[30], a data pipelining tool for Python[31].

For each stage of the pipeline a set of input, output and temporary files are defined. This means that there is intermediate storage between every executed step in the data pipeline.

Incoming sequence data is first checked with MultiQC[32], a quality control tool that aggregates the quality metrics from the FASTQ files. The report generated by MultiQC will make it possible to verify whether or not there as been an issue during the sequencing process that might lessen the quality of the results. Subsequently Trimmomatic[33] is used to remove the adapters that were needed for the high-throughput sequencing and low quality bases.

The Snakemake pipeline then proceeds to run the samples through Kaiju. Kaiju[34] is a tool for providing fast and sensitive taxonomic classification for metagenomics. Kaiju processes the sequences from the sample in the pipeline; tagging their taxonomy and thus providing taxonomic classification of the sequences in the sample. That data is later used for various forms of analysis pertaining to the relative diversity and abundance of different taxa in the sample.

In a side step we make use of HUMAnN3[35] in order to functionally profile our reads. HUMAnN3 takes the sequences out from trimmomatic and conducts taxonomic classification on them via MetaPhlAn3[36]. After the taxonomic classification, the reads of the samples are aligned to their the entire gene set according to taxonomic classification. Subsequently, a translated search is run on the unclassified reads before quantifying gene families and pathways. The final results of HUMAnN is a profile with information about the abundance of the gene functions in each sample. This is the basis for one of our two datasets. One substantial motivation for including this dataset was the significantly faster execution speed as compared to the main pipeline. It also allows us to compare the results on this dataset against the main pipeline dataset results.

Then the pipeline removes genes from the which are found to belong to humans by way of using a Burrows-Wheeler Aligner on the human genome. This is done with regard to ethical considerations and restrictions pertaining to the analysis and processing of human DNA. It is done because the focus of the study is on identifying the specific genes in the gut microbiome related to or associated with CRC.

In the next step the MetaSPAdes[37] algorithm is used in order to assemble the small "shotgun" reads that were obtained during the high-through sequencing into longer fragments. After this step most of the reads have been reconstituted back into their original genes.

At this stage the data pipeline sends the assembled and processed sequences through Prodigal[38], a protein-coding gene prediction software tool for bacterial and archaeal genomes. Prodigal translates the gene sequences into the amino acid sequences that they are predicted to produce.

The translated sequences are subsequently run through InterProScan[39] which is an amino acid search and classification tool search tool that enables search through the InterPro databases [40]. Due to the nature of the InterPro databases and InterProScan not all databases must be included. Which of the constituent databases to include has not yet been established. InterProScan yields hits on sequences in the searched databases returning accession numbers, a unique identifier for each protein in the database, if any are found. Via this accession number a wide range of functional information can be extracted about the sequence.

4.1 Output Data Structure

The final InterProScan output is in tabular form and includes IPR (InterPro) accession numbers from hits to included databases, short descriptions of action extracted from the databases, and associated GO[41, 42] (Gene Ontology) and KEGG[43] (Kyoto Encyclopedia of Genes and Genomes) Terms. The GO and KEGG databases contain information on the functions of genes and each term is tied to a specific gene function recording in the respective databases. The terms are retrieved via the protein accession entries in the InterPro database. The resulting output is thus each individual sequence put in a biological context and from which a range of other additional attributes can be retrieved for analysis. From this point in this master thesis we will refer to InterPro accession numbers as IPR terms and GO id's as GO terms. This is for simplicity since we will use them as features. GO terms and InterPro accession numbers are not the same as InterPro accession numbers are database lookup id, while GO terms can be hierarchical in nature.

We also retrieve output data from HUMAnN which is also in tabular form. In the case of the HUMAnN dataset we have a matrix of samples and GO term abundance. While the results for InterProScan are organized such that every sample is represented by a file containing an overview of the reads and their mapping.

4.2 Data Preparation

An important step in the training of a machine learning classifier is the treatment of data before the training phase. There are a series of different steps that can be taken in order to treat data in order to improve classifier accuracy and performance across different datasets while not impairing the integrity of the dataset. In this section we will cover those as they relate to this project.

Normalization is the process by which the number of reads is normalised in case of differing numbers of sequences per sample, which is the case for our dataset. This ensures the intercomparability of samples in the study. A selection of methods for data normalization across samples have been tried [44, 45]. We have decided to apply l2-regularization from the scikit-learn preprocessing library[46].

In the realm of machine learning a feature is an attribute or a collection of attributes of the data [47]. In our case the attributes are the terms and accession numbers tied to each sequence in the samples and the information retrievable via the database entries for those terms and accession numbers. There are two ways to prepare features in the data - feature selection and feature extraction. Feature selection is the process of selecting which attributes to include as features, while feature extraction is the dimensionality reduction of the attributes in the dataset to extract features. Preparing appropriately independent, discriminating and relevant features is essential to train a capable classifier. In our resulting dataset the features will be discrete, but be quantified as non-negative real numbers. Any actions related to the treatment of features must be applied equally on the entire dataset in order to avoid distortions.

The selection and extraction of features for training of a classifier can significantly impact the results. Earlier work on predicting host trait characteristics has made use of the taxonomic distribution of the samples. Given the abundance of different taxa in each sample and the similarity of the individual sequences in the sample, they can be grouped into operational taxonomic units (OTUs) [48]. This is done to avoid excessive sparsity and feature counts in the dataset which could arise from not combining any sequences bar those that are identical.

A series of different approaches to the grouping and aggregation of OTUs exist. Approaches include Fizzy[49], MetAL[15] and hierarchical feature extraction (HFE) [50]. Zhou and Gallins 2019[24] found that the application of HFE generally improves prediction accuracy for several machine learning methods on three different metagenomic datasets. Since this project will use features based on gene data and not exclusively taxonomy, the feature selection approaches based on OTUs discussed here is not directly applicable. They do, however, give insight to the feature engineering work done for classifier training on environmental samples of microbiota for host trait prediction.

Another issue relating to the features of the dataset is data sparsity. In a collection of samples there might identified genes that are exceedingly rare or not present in other samples. Machine learning methods handle sparcity in input data with varying degrees of success with some, such as Lasso Regression[51], performing feature selection in and of themselves. For methods that do not do this sparcity should be handled in the data preprocessing stage. Several approaches exist. One approach to sparse features is to impute their abundance. A number of methods exist for this purpose they include. Imputation using mean/median values of the missing feature where present, imputation via kNN[52], imputation of the most frequent value where present and the MICE[53] algorithm for multivariate imputation by chained equations. Another approach to reducing the total number of features by way of feature selection is to remove the features that do not meet a required minimum abundance. This is a common approach with metagenomic datasets. Too aggressive an application of this approach might reduce the capacity of the dataset to be used for accurate classification as it might prune away important associations.

It's also possible to apply univariate feature selection (UFS) to the set. Whereby we select the features that have the highest p-value with the output label. This makes the dataset less feature rich, and is one approach to removing features that might not have much of an impact on the result. Such an approach might improve performance of the classifiers by removing noise and will also reduce training time.

From the sample collection discussed earlier we obtain data labels consisting of a discrete value per sample which indicates the CRC status in the sample host. The purpose of the classifier is to identify a specific host trait, in our case gender and CRC development stage, so each of the samples run through the pipeline are annotated with the associated data label before the training of the classifier. The appropriate approach to feature extraction from the GO, KEGG and IPR terms tied to the samples in the dataset will be explored. One approach could be quantifying the occurrence of each terms in each sample, but it might be advantageous to obtain other related data such as taxonomy tied to the attributes. In summary, the exact manner in which data processing will be done before the classifier training has been a concern for this project.

5 Machine Learning on Metagenomic Data for Host Trait Prediction

Emerging technologies give rise to new methods of processing metagenomic data with potentially novel and useful insights. Training classifiers with metagenomic datasets from the gut microbiome has been demonstrated to yield classifiers capable of predicting host traits such as type 2 diabetes and CRC with reasonable accuracy [15].

Much of this work has been on taxonomic profiles often derived from S16 rRNA sequencing, with the whole genome shotgun sequencing approach, we can look at gene functions. It is the genetic and functional profiles in microbial communities that determine their overall function [12]. Using the added functional profiles gained via shotgun sequencing, we might be able to uncover new associations.

5.1 Algorithms of Interest

The selection of classifiers has been made on the perceived performance of their classification ability as well as the interpretability of their predictions and models. Our dataset and datasets referenced from earlier work have been labeled datasets. In labeled datasets we know the trait or the thing to predict with the classifier. That entails the use of supervised machine learning algorithms. Another point is that for our dataset the classifier will be binary. Each sample will be predicted by the model to either belong to a cancer victim or not, that is also the case for most, but not all of the earlier work discussed here.

Based on previous work summarised in section 5.4, we think that lasso regression[51], support-vector machines (SVM)[54], random forest[55], XGBoost[56], multi-layer perception neural networks[57] would be good choices for this setting. Long short term memory recurrent neural networks (LSTM)[58] have not been used on the gut microbiome. Though it could be interesting to see how the neural network performs relative to the RNNs previously tried and they have been successful in other applications.

Lasso regression is a form of linear regression which minimizes the residual sum of squares subject to the sum of the absolute value of coefficients being less than a constant [51]. This results in certain coefficients being set to zero and thus performs a form of feature selection on the dataset. Lasso regression has been used for host trait predictions on metagenomic datasets with good accuracy and the classifiers generated are interpretable.

The use of SVMs is a common approach to supervised learning. SVMs construct maximum margin separators, separators that aim to minimize generalization loss over empirical loss, on the trained dataset. Since data that is not linearly separable in the original input space might be separable in higher-dimensional space, the kernel trick can be used to perform non-linear classification by way of inputs into high-dimensional feature spaces being implicitly mapped [59]. Thus the high dimensional linear separator is non linear in the original space [47]. SVMs have been used on metagenomic datasets relatively yielding good classification accuracy.

Random forest is a variant of the decision tree learning (DTL) algorithm. The random forest algorithm trains its classifier by generating several DTLs and aggregating the resulting classifiers. DTLs make binary splits on the feature in the dataset that has the highest level of impact on the classifier as measured by some metric (typically information gain, entropy or gini) until a specified number of nodes is reached [47]. The random forest algorithm generates a "forest" by creating an ensamble of decision trees that have been subjected to the random subspace method [60]. These trees are aggregated in a "bagging" process into a single classifier [61]. Another forest type classifier that has yielded promising results is XGBoost[56]. XGBoost uses gradient boosting to improve the classifier model.

The random forest algorithm has been used extensively with promising results from earlier studies. Due to the ensemble nature of the resulting classifier interpreting the predictions and the model is arduous. On their own they are therefore often seen as a black box. However, efforts have been made to make the predictions made by random forest classifiers interpretable. The package treeinterpreter for scikit-learn enables decomposition of predictions into bias (training set mean) and the individual feature contributions thereby enabling analysis and interpretation of the generated predictions [62].

Neural Networks are a broad category of algorithms with a multitude of different versions and variants. Common to them all is their ability to perform nonlinear regression to generate classifiers and doing so by utilizing multiple networked layers of nodes with different properties depending on the type of neural network [47]. A range of different neural networks have been used for host trait prediction on metagenomic data with good, but somewhat varied results with some variants performing better than others [63]. A significant downside to the use of neural networks is the relative lack of interpretability of the generated predictions which limits their use for analysis.

MLP is an early neural network design where by the network has three distinct layers an input layer, a hidden layer and an output layer. The MLP neural network is comprised entirely of n layers of regular perceptrons with training of the internal weights taking place via backpropagation, as with all neural networks. The number of nodes in the hidden layer is instrumental in the capacity of these neural networks performing nonlinearly separable classifications [47].

The LSTM neural network is a particular kind of RNN that is less prone to the vanishing gradients problem [58]. Recurrent neural networks are neural networks that retain memory in the nodes and incorporate recursive structures. By feeding the output of each node into its own inputs, they are able to support short term memory [47]. In LSTM differ from regular RNNs by using a set of gates to control the entry of new information, it's output and when it is forgotten. This enables greater control over the gradient flow and preservation of earlier training. LSTM RNNs have been seen to provide accurate classifiers in a series of scenarios, particularly relating to time series, but have not been studied on metagenomic data. On earlier work in regard to predicting of host traits with metagenomic data RNNs have been outperformed by other forms of neural networks [64].

5.2 Machine Learning Toolkits

There are a range of different machine learning toolkits and libraries available for use and these provide implementations of a wide variety of different classifiers. For our purposes scikit-learn[46] and keras[65] running on TensorFlow[66] seem well suited given their common use and coverage.

Scikit-learn is a free software Python module integrating a wide range of state-of-the-art machine learning algorithms for medium scale supervised and unsupervised problems. TensorFlow is an end-to-end open source platform for machine learning running on Python. Keras is a high-level neural networks API, written in Python that can be run on TensorFlow. Scikit-learn v0.23.2 will be used for the implementation of most of the classifiers, and keras on TensorFlow v2.3.0 for the neural networks that are not supported in scikit-learn. This project will rely on these programs' implementations of the classifiers that we intend to run on our dataset.

For interpretation we intend to make use of LIME[67]. LIME enables us to look at which features are deemed most important for the classification of any given sample. This gives us a breakdown which can allow us to look at which gene functions are deemed the most significant. While we have chosen to use LIME, there are other alternatives available. Some have a wide range of supported algorithms like ELI5[68]. Others are specifically designed for certain algorithms sometimes in specific packages. An example could be treeinterpreter[62] for scikit-learn. LIME was ultimately chosen because of the large range of supported algorithms, high adoption rate, useful applicable output and Python support.

5.3 Issues

A regular issue when using training classifiers is the overfitting of the classifier to the training data. Overfitting entails that the classifier is so closely mapped to the specific training set so as to not be a good classifier when presented with other similar data. In order to manage this the project will make use of k-fold cross validation, whereby the dataset is split into k chunks of equal size, one of which is designated as test set with the rest being set as a training set. *k*-fold splitting is executed before every round of classifier training to avoid a constant training and test chunk. This is a common approach to manage and control issues related to overfitting [47].

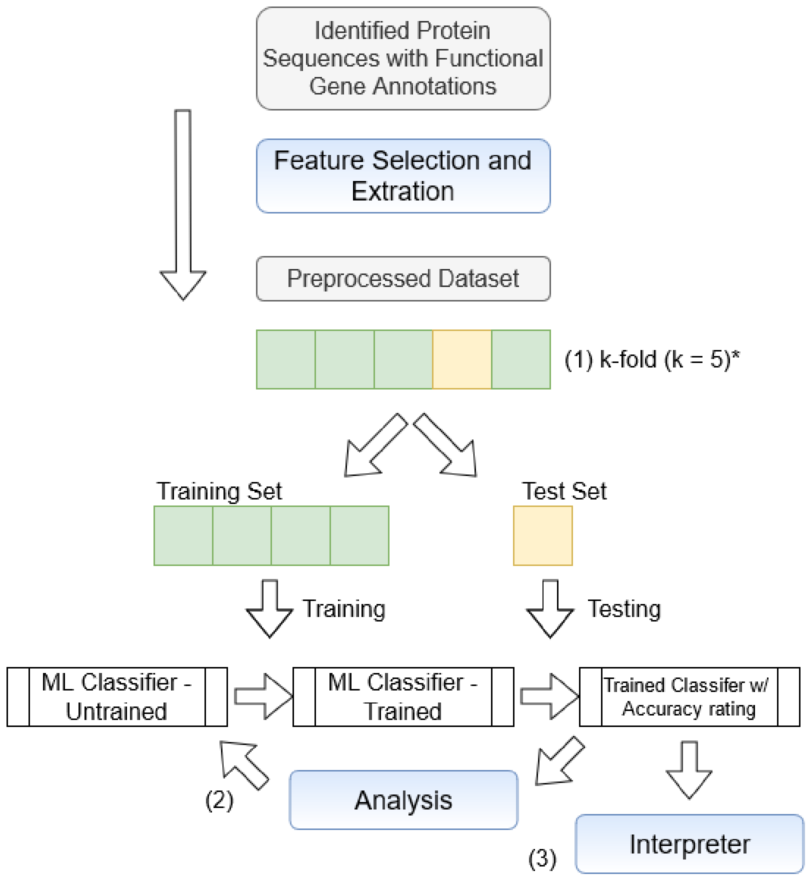
Identified Protein Sequences with Functional Gene Annotations
Feature Selection and Extration
Preprocessed Dataset
(1) k-fold (k = 5)*
Training Set
Test Set
Training
Testing
ML Classifier - Untrained
ML Classifier - Trained
Trained Classifer w/ Accuracy rating
(2)
Analysis
(3)
Interpreter

Figure 2: Machine Learning Component Overview

(1) - The dataset is split randomly according to k-fold splitting, (2) - Not always applicable. Influences adjustments. Analysis might also impact feature selection and extraction. (3) Interprets the classifiers models. * - For illustrative purposes

5.4 Accuracy Metrics

When specifying the accuracy of the predictions made by the classifiers there are a few metrics in use. The three we will reference or use are error rate, accuracy and Area Under Curve (AUC). Error rate is the rate of false classifications made. Accuracy is the rate of correct predictions made. AUC, also referred to as Area Under Receiver Operator Curve (AUROC), is a measure of the accuracy of a binary classifier. The AUC value is an expression of the rate of accurately predicted true positives and false positives by the classifier. A higher value implies greater accuracy. Given that AUC is a model for binary classifiers a AUC value of 0.5 implies a random guess on the part of the classifier.

5.5 Earlier work

A number of studies have applied machine learning methods to metagenomic datasets, comparing the relative performance of the machine learning algorithms. As referenced earlier the previous studies have been conducted on taxonomic profiles in contrast to our dataset.

For CRC prediction Pasolli et al. 2016[15] found that using random forest, support vector machines (SVMs), elastic net and lasso regression on the dataset from Zeller et al. 2014[69] yielded AUC values of 0.87, 0.81, 0.79 and 0.73 respectively. The sample size was 121, with 48 cases and 73 controls. The taxonomic profile covered the species level.

Also for CRC prediction, Ai et al. 2017[70] applied Bayes net, random forest and logistic regression separately on two different datasets; one from a Chinese cohort and one from a French cohort. The predictions yielded AUC values of 0.93, 0.94 and 0.98 for the Chinese cohort and AUC values of 0.86, 0.86 and 0.71 for the french cohort respectively.

Neural networks have been tried on a relatively large metagenomic dataset by Ditzler, Polikar et al. 2015[63] and Reiman et al. 2017[64]. They used the gut microbiome dataset from Caporaso et al. 2011[71]. Unlike the other sets discussed here, the Caporaso et al. dataset is composed mostly of temporal variations of the human microbiome. It sampled two individuals at four sites over 396 time points, one of the sites being a fecal sample.

Ditzler, Polikar et al. 2015[63] tried to predict host gender and sample body site origin from the dataset using a selection of neural networks with different parameters and one random forest application. The random forest implementation outperformed the neural networks, but the multi-layer perceptron (MLP) neural network also performed well with error rates of 0.01 and 0.01 for body site and 0.03 and 0.08 for host gender respectively. Reiman et al. 2017[64] had good results with an accuracy of at least 0.97 for at least one set of parameters for all implemented neural networks except for recursive neural networks (RNN) with an accuracy of 0.84.

Zhou and Gallins 2019[24] has assembled a comprehensive review of the application of machine learning classifiers on metagenomic data and taxonomic profiles for host trait prediction.

6 Aims

This introduction has presented the basis for this master thesis on the use of machine learning methods on processed metagenomic data for host trait prediction as a part of a data pipeline for deep analysis of metagenomic data. Given the basis of earlier work in applying machine learning methods on taxonomic distributions of gut microbiomes to predict host traits, it is clear that such an approach can yield insights. This project has special promise due to the functional gene data on which the classifiers will be trained as well as the large sample size.

As stated earlier, one of the aims of the CRCbiome is find gut microbiota biomarkers for CRC. The identification of such biomarkers could lead to new and better tests for CRC. This master thesis has been conducted as a part of the CRCbiome project. The concrete aims for this master thesis are.

6.1 Primary Aims

- Train machine learning classifiers for CRC prediction.

  Accurate machine learning classifiers for CRC prediction will be trained as a part of the thesis.

- Analyse classifier predictions to identify possible biomarker candidates for colorectal cancer.

  The identification of biomarker genes or biomarker gene functions could lead to new and better tests for CRC. While we know which bacterial taxa are associated with CRC [69], we want to obtain more information about which genes and gene functions are.

- Integrate the created machine learning module into the data pipeline.

  The machine learning module will be integrated into the data pipeline for the CRCbiome project.

This master thesis also has a set of secondary aims. These fulfillment of these aims are necessary for the achievement of the primary aims, but the secondary aims also carry value in their own right and may help inform best practice for similar projects.

6.2 Secondary Aims

- Identify appropriate classifiers for the metagenomic dataset.

  We have discussed and will trial and confirm likely appropriate classifiers. The judgement of which algorithms are appropriate have been made on the basis of the experiences of other projects using taxonomic profiles for host trait prediction in addition to general assessments. While not equivalent, the experiences from those projects have informed our initial choices.

- Identify appropriate parameters for the chosen classifiers where applicable.

  Here it is also expected that the experiences from other projects using taxonomic profiles based on OTUs for host trait prediction will be largely transferable, though to what extent is not yet clear.

- Ensure appropriate fitting of the model on the training set.

  Due to the novel and limited nature of this dataset, this will be closely examined. Mitigation methods such as k-fold splitting will be applied.

- Make assessments of the predictive ability of trained classifiers and the interpretability of their results.

  When training and testing the classifiers we will make note of and reflect on their comparative performance and models. Predictions and classifier models of any of the methods applied could perform poorly, differ significantly from the consensus or behave unexpectedly. Such an event, while possibly interesting and informative, could stem from errors and issues in the process. Additionally, it's important to recognise that higher accuracy might not always indicate a more correct model. Especially with regard to large, heterogeneous and diverse datasets such as metagenomic data from the gut microbiome. In this context it is also important to reflect on cohort overfitting.

- Perform appropriate data preprocessing.

  Common approaches for feature selection and extraction on datasets will be applied and evaluated. Given the opportunity to extract supplementary information from the identified protein sequences with functional annotations, we will consider making use of this. Feature imputation methods might also be applied if judged necessary or desirable.

- Visualise results.

  A Python framework or an R[72] framework will be used visualize the results. This is in order to provide clear and insightful visualizations to illustrate the findings and for the purpose of aiding visual analysis of the results.

7 Methodology

The software developed through this master thesis has been named "Multiple Algorithm Pipeline for machineLEarning on gene functions" (MAPLEgf). The Python IDE PyCharm was used to develop MAPLEgf in Python v3.6.10. Together with Python we used NumPy[73] v1.18.5. Git and GitHub were used for version control of the software.

For preprocessing two separate scripts were made, one for the HUManN3 dataset and one from the dataset from InterProScan. A short script was made for each of the applied algorithms while making use of extra functionality from an interface common to all the scripts. Each of the algorithms reads from a config test file which contains a series of parameters and some comments on how to modify them. These were used to apply different settings to the algorithms and by updating them Snakemake would rerun the scripts since one of the inputs would have been updated. The full source code can be accessed at the GitHub repository associated with the project[74]. Se figure 3 for a graphical representation of dataflow in the pipeline.

The algorithms were executed inside of the University of Oslo managed "Tjeneste for Sensitive Data" (TSD) in a Slurm managed system. This gave access to large amounts of computing power for potentially very heavy execution tasks. The script that used to execute the Snakemake pipeline is in the project GitHub repository.

Due to the COVID-19 pandemic of 2020/2021 there was some delay in obtaining sequenced samples due to the sequencing capacity being occupied by other sources. Additionally the pipeline requires some amount of time to process the sequenced sample data. At the time of the final executions we had a total of 468 annotated InterProScan samples and 953 HUMAnN3 annotated samples. The annotated sample data format for each set was tab-separated values (.tsv) and comma-separated values (.csv) respectively.

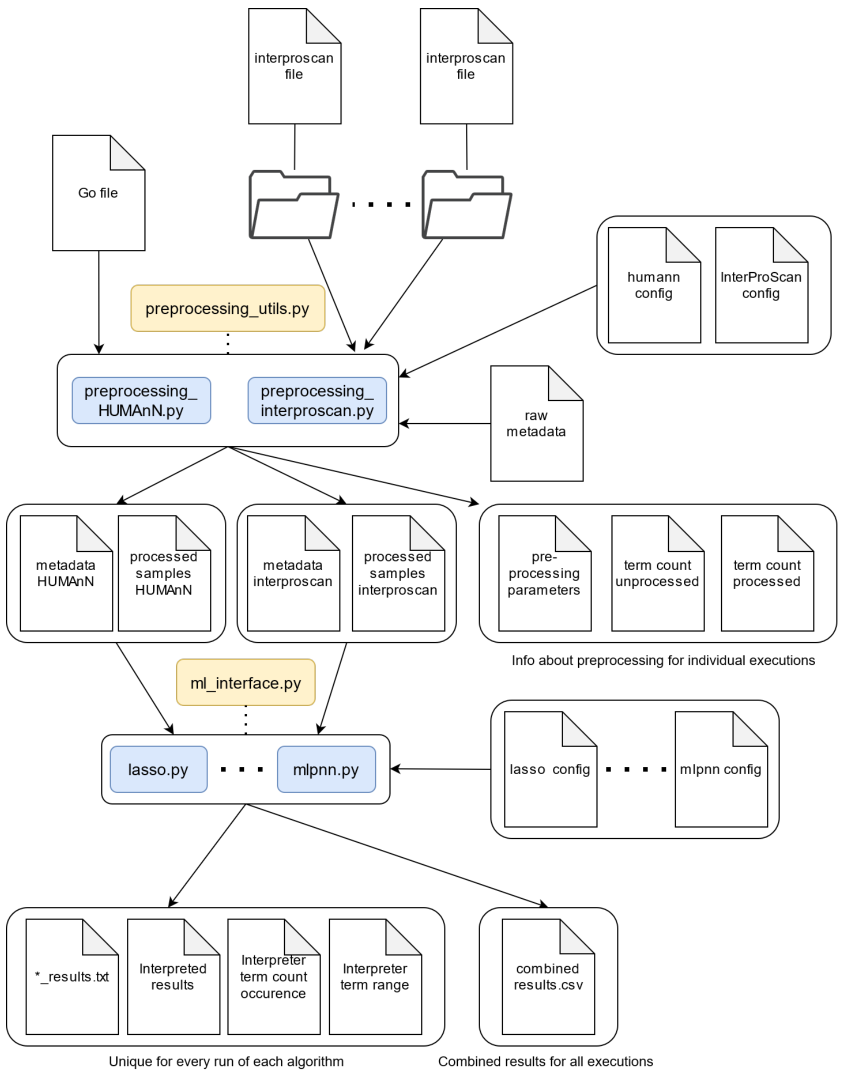

Figure 3: Overview of data flow in MAPLEgf

Blue rectangles are Python files directly executed. Yellow rectangles are Python files used as interfaces or utility function files. ... denote ranges of files. Arrows indicate dataflow.

When executing the preprocessing script takes in the sample data. It is then referenced to the supplied metadata file containing the labels for the data. The referenced data is subsequently formatted into a matrix with samples in and their respective feature abundance. For InterProScan it selects which term to include as per the config file provided. For every read in every sample contained in the InterProScan dataset it fetches the tagged terms and multiplies them with the coverage of the each specific term and the read length. The dataset from HUMAnN required no such preprocessing.

The preprocessing script also takes parameters for feature selection from a config file. As a result of the nature of the dataset sourcing, it's assumed to contain some wrongly annotated gene functions as noise. In order to handle this noise we would like to remove all features that are not found in a certain quantity in a certain share of the samples.

The labels for classification were sourced directly from study information where samples are matched to anonymized study participants and several information elements regarding them. The label metadata elements were grouped for gender into a male and female group and for cancer the metadata elements were grouped by cancer and advanced adenomas together and all earlier states of cancer progression separately.

The data can be normalized before execution via the config files for each algorithm. Each sample is normalized using the scikit-learn v0.23.2 preprocessing library.

Univariate Feature Selection (UFS) was attempted both before the running the machine learning algorithm and for each training set in each individual k-fold split. The UFS was implemented in the common interface for the machine learning scripts. Is is executed on a per classifier basis in the machine learning section of the pipeline. The settings for UFS are also defined from the config file.

Running the script for the analysis in TSD starts a Snakemake v5.3.0 pipeline which lists a series of jobs to be executed piping the inputs and outputs of scripts together.

When training the algorithms we first focused on getting results for predicting gender. It's established that there are differences in the gut microbiomes of men and women. Therefore we thought it as likely that it would be possible to train classifiers able to predict the host gender status based on the gene annotations.

For each execution on the dataset we made changes and optimizations to the parameters of the classifiers in order to improve their performance. Several values for the feature selection for the preprocessing script were also tried. The final results for each run are outputted to a individual .txt file for each result and a common .csv file for the combined results. As a final part of the executing a selection of results are run through the classifier interpretation tool LIME[67]. LIME v0.20.1 was used. The LIME library is used to write a series of .csv files with an overview over which terms were deemed most important for the classification of samples. The final results contain information about program execution times, AUC value and associated information about each run. There is also the associated slurm-log file listing the printed output for each complete run.

The explain .csv file generated by using LIME Python library gives an indication of which features are the most significant in the given model classifying a certain sample as either. Every distinct feature has a percentage impact listed to each term and value. Since our feature values are continuous, we made use of the discretize option in LIME with the value "quartile". This makes the different sample predictions easier to compare and to aggregate as we get only four intervals for each continuous value instead. For every execution we set a threshold of 100 for the number of features to include in the output. For every sample that was executed the results of these most important 100 features were then aggregated and average by number of samples to give an average indication for each feature.

In addition to this there are another two .csv files that are output from the program when interpretation is enabled. One lists the number of times each term was counted as one of the 100 most important features for classification. While the other lists how many times the specific interval occurred in the 100 most important feature values for each sample. These give us some indication as to which features are commonly found to be important and should be subject to examination.

Source and Term Type	**Terms**	**Samples**
InterProScan IPR	14923	468
InterProScan GO	3300	468
HUMAnN GO	9408	944

Table 1: Overview of sequenced samples provided for this thesis.

Host Gender Labels	**Males**	**Females**	**Total**
InterProScan IPR	293	171	464
InterProScan GO	293	171	464
HUMAnN GO	532	407	939

Table 2: Overview of host gender label information applied with the samples for this thesis.

Host CRC Development Stage Labels	**Cancer or Advanced Adenoma**	**Control**	**Total**
InterProScan IPR	221	247	468
InterProScan GO	221	247	468
HUMAnN GO	412	532	944

Table 3: Overview of host CRC development stage information applied with the samples for this thesis.

8 Results

The results presented are the best overall results obtained. The parameters for the classifiers were modified in such a way as to find the best possible classification performance. Several different feature selection levels for the dataset were also tried. On the training speed of the classifiers we have an overview over the run times for the algorithms. Here the interest was more in comparisons between the different algorithms as the specific conditions of the execution.

In evaluating the performance of results we will be comparing the performance of our classifiers to the performance of a baseline classifier. The baseline classifier we refer to in this thesis text is a mode classifier. Such a classifier classifies all samples as the most common class and on average manages to score an AUC equal to the size of the largest class. We tried different datasets that have different distributions of control and sample cases. This is reflected in the different baseline classifier scores. In order to properly get an impression of the performance of any given classifier on a dataset, the baseline AUC should be taken into consideration.

As laid out in the methodology chapter, each of the scripts takes in a small configuration file which controls some of the parameters in the application. These external configuration files were used in order to easily update parameters for the algorithms. No parameters except the ones in the configuration files were explicitly set for the machine learning algorithms and were therefore the default values for the packages. Table 3 summarizes the specific parameters for each machine learning algorithm.

Algorithm	**Parameters**
LSTM*	embedding_enabled=false, embedding_level=32, LSTM_depth=512, num_layers=2, denselayer_size=1, dropout=0.1, epochs=20, batch_size=32, l2_regularization=false, regularization_weight=0.0001
MLPNN	max_iter=10000, hidden_layer_sizes=1000, alpha=0.0001
Random Forest	n_estimators=2500
XGBoost	max_depth=32
SVM	C=0.1
Lasso	positive=true, tol=0.00001, alpha=0

Table 4: Parameters set for the different algorithms in preprocessing.
*LSTMs from TensorFlow have a more complicated setup than the algorithms implemented via scikit-learn. These are the adjustable parameters in the configuration file, but these are not directly passed to the method. Please refer to the source code in the GitHub repository for exact information.

8.1 Results of Preprocessing

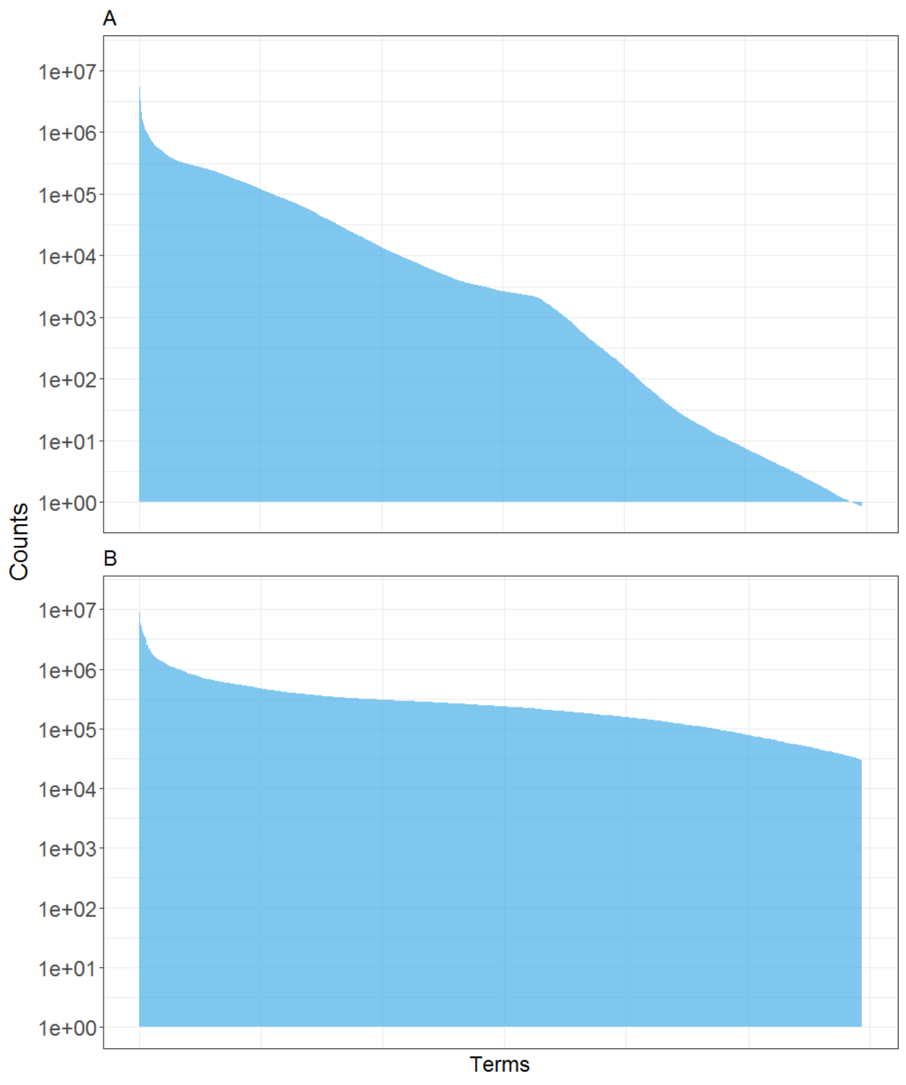

Figure 4: InterProScan IPR term abundance before and after preprocessing

A. InterProScan IPR terms before preprocessing, x-axis shows terms sorted in descending order by aggregated abundance for all 14923 terms. B. InterProScan IPR terms after preprocessing, x-axis shows terms sorted in descending order by aggregated abundance for all 2348 terms. y-axis shows the aggregated abundance of each term in the dataset for all the samples. y-axis is scaled by log10.

Figure 4. shows how many terms are present before and after the execution of preprocessing on the InterProScan IPR gene annotation dataset. The feature selection was executed with a minimum coverage of 250 per sample for a given term in at least 20% of the samples included.

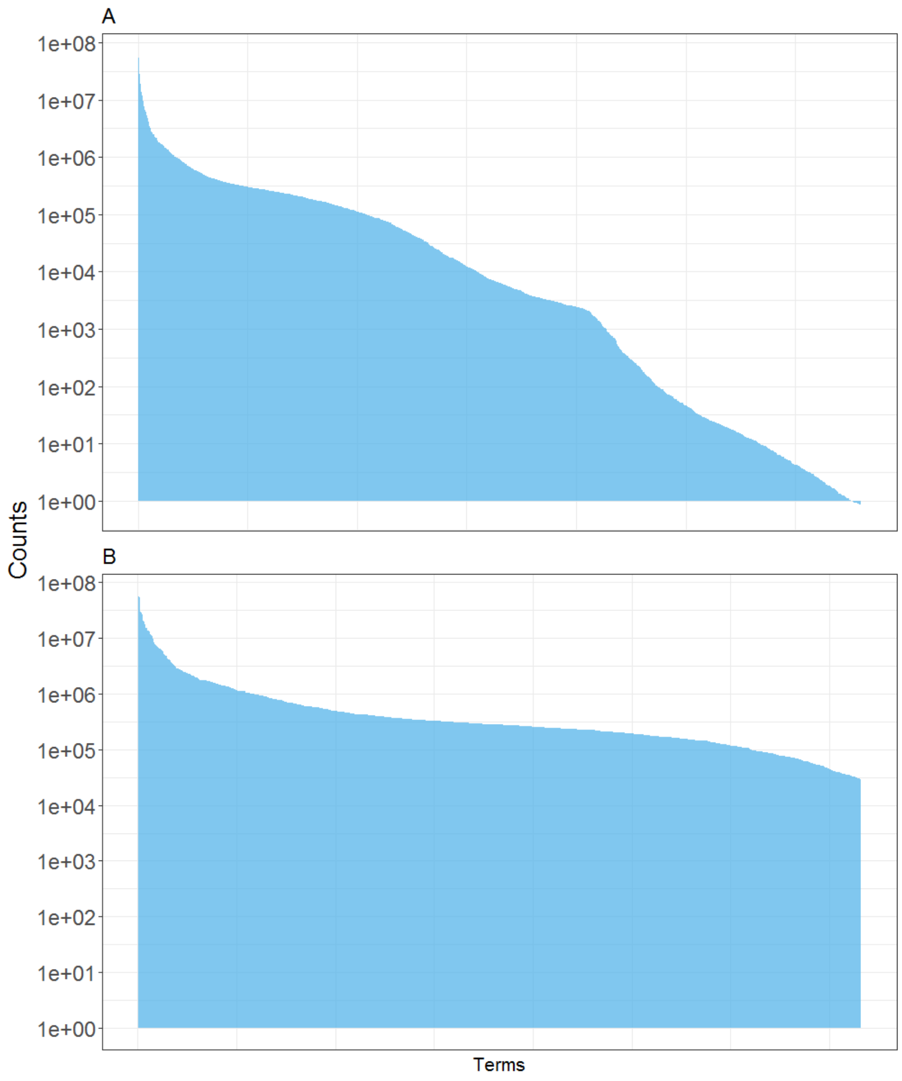

Figure 5: InterProScan GO term abundance before and after preprocessing

A. InterProScan GO terms before preprocessing, x-axis shows terms sorted in descending order by aggregated abundance for all 3300 terms. B. InterProScan GO terms after preprocessing, x-axis shows terms sorted in descending order by aggregated abundance for all 1098 terms. y-axis shows the total abundance of each term in the dataset for all the samples. y-axis is scaled by log10. x-axis follows each term sorted by the abundance in descending order.

Figure 5. shows how many terms are present before and after the execution of preprocessing on the InterProScan GO gene annotation dataset. The feature selection was executed with a minimum coverage of 250 per sample for a given term in at least 20% of the samples included.

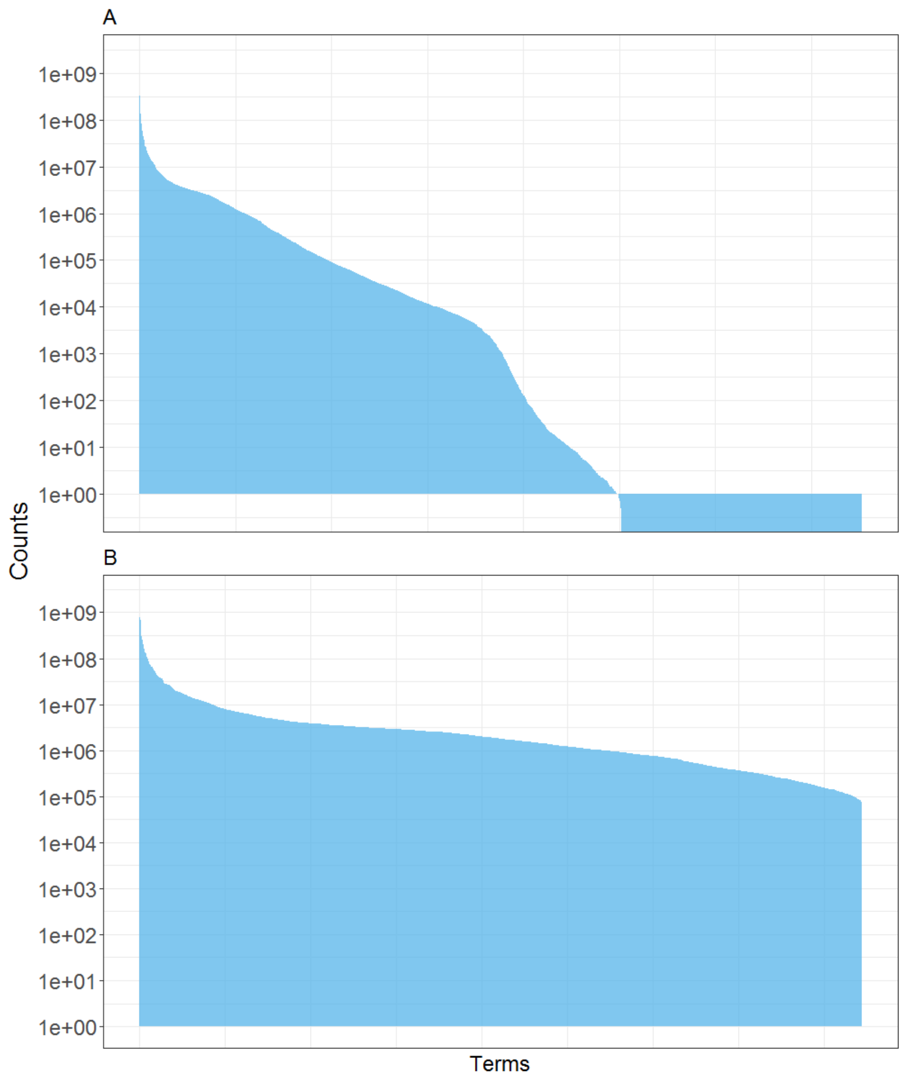

Figure 6: HUMAnN GO term abundance before and after preprocessing

A. HUMAnN GO terms before preprocessing, x-axis shows terms sorted in descending order by aggregated abundance for all 9408 terms. B. HUMAnN GO terms after preprocessing, x-axis shows terms sorted in descending order by aggregated abundance for all 2109 terms. y-axis shows the total abundance of each term in the dataset for all the samples. y-axis is scaled by log10. x-axis follows each term sorted by the abundance in descending order.

Figure 6. shows how many terms are present before and after the execution of preprocessing on the HUMAnN GO gene annotation dataset. The feature selection was executed with a minimum coverage of 250 per sample for a given term in at least 20% of the samples included.

Source	**Features Before**	**Features after**	**Samples**
InterProScan IPR	14923	2348	468
InterProScan GO	3300	1098	468
HUMAnN GO	9408	2109	953

Table 5: Overview of samples before and after preprocessing

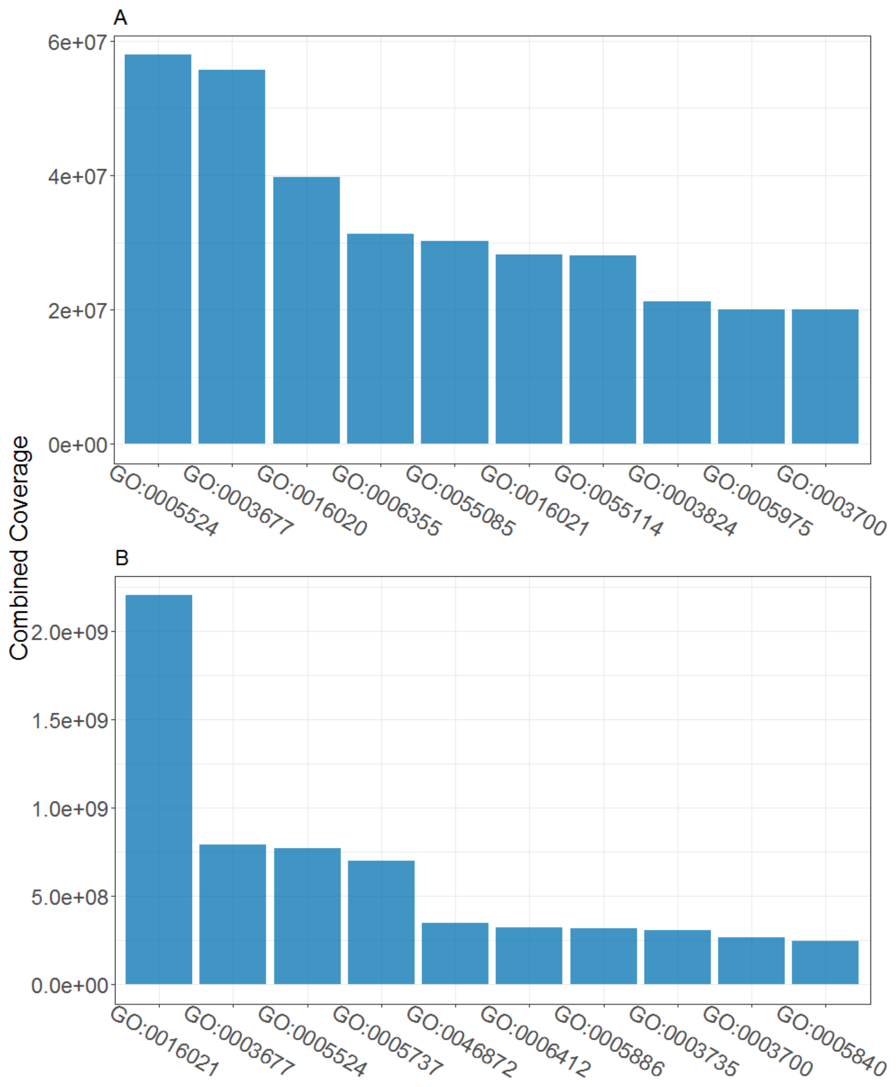

Figure 7: Comparison of the 10 most prevalent GO terms from InterProScan and HUMAnN gene annotations

A. InterProScan GO terms. B. HUMAnN GO terms. y-axis shows the total abundance of each term in the dataset for all the samples. x-axis follows each term sorted by the abundance in descending order.

Figure 7 gives an overview and comparison of the results from the

HUMAnN and InterProScan by comparing the 10 most prevalent features in each dataset for all samples. It's notable that despite the same source data the relative abundance of the data is fairly different. This goes for the ordering of the terms. For instance we find that both sets contain GO:0016021, GO:0003677, GO:0005524 and GO:0003700. These are all terms for common gene functions that would be expected to show up frequently.

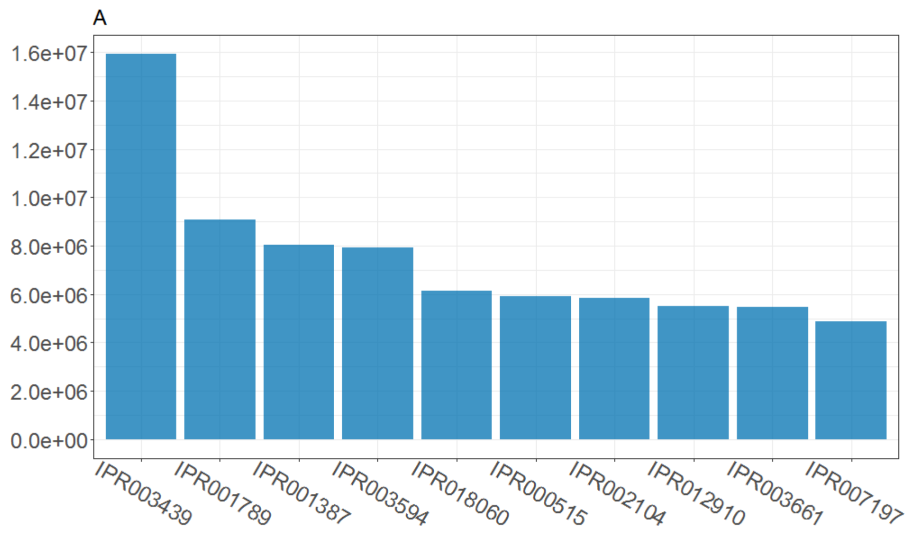

Figure 8: Overview of the 10 most prevalent IPR terms InterProScan gene annotation.

A. InterProScan IPR terms. y-axis shows the total abundance of each term in the dataset for all the samples. x-axis follows each term sorted by the abundance in descending order.

Figure 8 shows an overview of the 10 most prevalent terms. We do not have a comparison for this as only InterProScan yields IPR terms.

8.2 Results for Host Gender Prediction

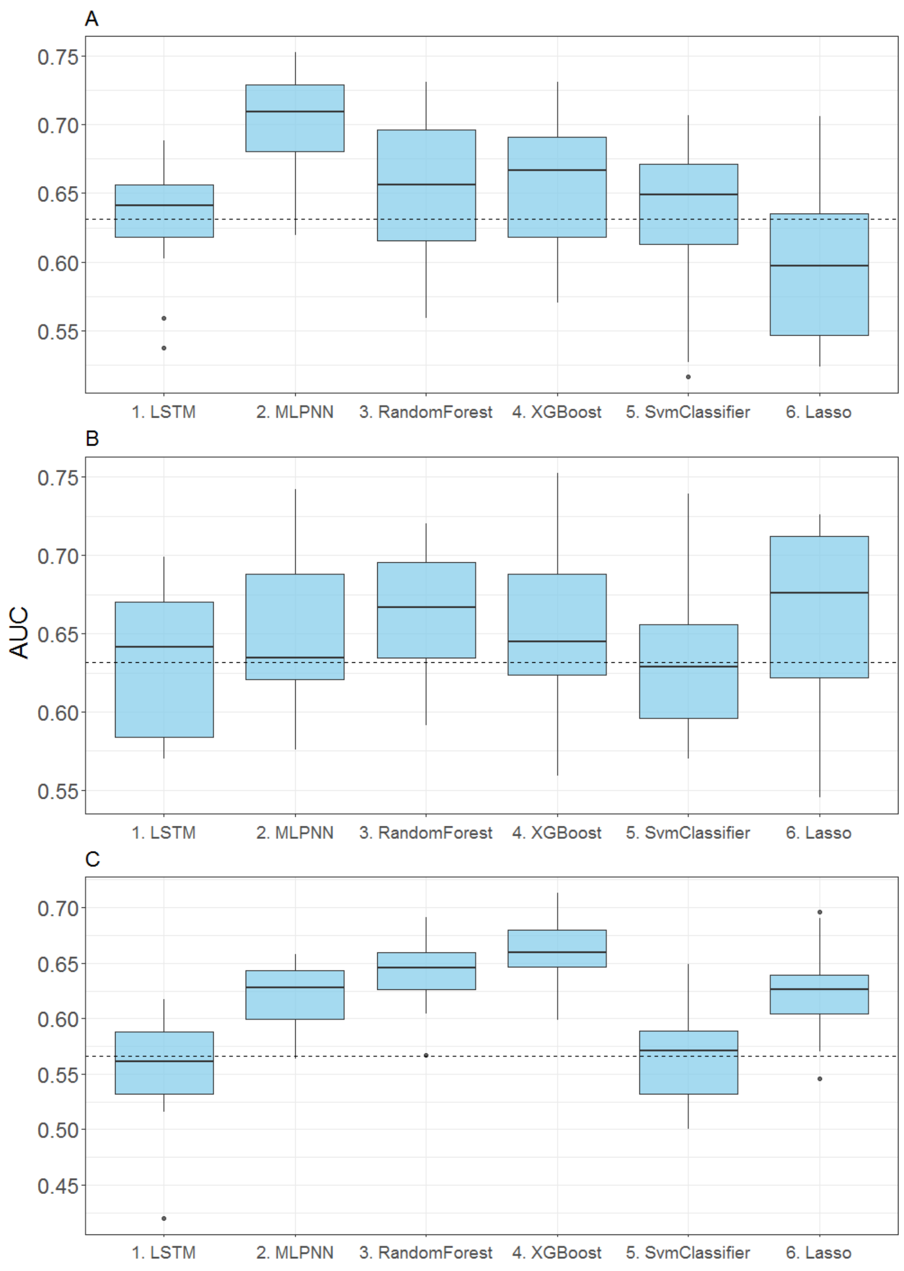

Figure 9: Performance for prediction of host gender by classifier.

A. Classifiers trained on InterProScan IPR terms. B. Classifiers trained on InterProScan GO terms. C. Classifiers trained on HUMAnN GO terms. y-axis shows AUC score. A total of 20 train/test cycles are included in the results for each algorithm. Dotted line marks the size of the largest class, an effective baseline performance rating.

For each run a classifier was trained with the respective dataset.

A total of 20 train/test cycles are included in the results for each algorithm in Figure 9. Please refer to the config files in GitHub for individual parameters for each method and version of each classifier.

Common for all algorithms were a k-fold split of 5. A greater splitting would give a larger set to train on, but would likely increase variance between each set due smaller test sets. The results for each k-fold train/test split makes up Figure 9. Also common for all samples is no UFS as performance was consistently worse than without (data not shown). All samples were l2 normalized. Dotted line marks the size of the largest class, an effective baseline performance rating.

The best results appear to come from MLPNN, random forest and XGBoost, in particular on the dataset HUMAnN GO terms and the InterProScan IPR terms. On the InterProScan IPR dataset the MLPNN also did very well. The larger set size of the HUMAnN GO dataset helps to explain the lower standard deviation of the results on it.

Classifier	**Mean AUC**	**St. Dev**	**95% CI**	**Baseline AUC**
LSTM	0.6314	0.0398	0.5519 - 0.7111	0.6315
MLPNN	0.6999	0.0358	0.6283 - 0.7714	0.6315
Random Forest	0.6509	0.0525	0.5459 - 0.7559	0.6315
XGBoost	0.6567	0.0479	0.5610 - 0.7525	0.6315
SVM	0.6374	0.0518	0.5336 - 0.7413	0.6315
Lasso	0.6024	0.0561	0.4901 - 0.7147	0.6315

Table 6: Results for classifiers trained with InterProScan IPR terms.

Classifier	**Mean AUC**	**St. Dev**	**95% CI**	**Baseline AUC**
LSTM	0.6315	0.0475	0.5634 - 0.7265	0.6315
MLPNN	0.6475	0.0511	0.5453 - 0.7498	0.6315
Random Forest	0.6622	0.0388	0.5846 - 0.7399	0.6315
XGBoost	0.6546	0.0443	0.5660 - 0.7431	0.6315
SVM	0.6315	0.04353	0.5444 - 0.7186	0.6315
Lasso	0.6634	0.06249	0.5384 - 0.7884	0.6315

Table 7: Results for classifiers trained with InterProScan GO terms.

Classifier	**Mean AUC**	**St. Dev**	**95% CI**	**Baseline AUC**
LSTM	0.5566	0.0481	0.4605 - 0.6527	0.5666
MLPNN	0.6192	0.0301	0.5590 - 0.6795	0.5666
Random Forest	0.6419	0.0275	0.5868 - 0.6969	0.5666
XGBoost	0.6595	0.0286	0.6021 - 0.7168	0.5666
SVM	0.5631	0.0393	0.4844 - 0.6418	0.5666
Lasso	0.6248	0.0360	0.5527 - 0.6968	0.5666

Table 8: Results for classifiers trained with HUMAnN GO terms.

Tables 6 to 8 show the results for the classifiers trained with their respective datasets.

8.3 Results for Colorectal Cancer Stage Prediction

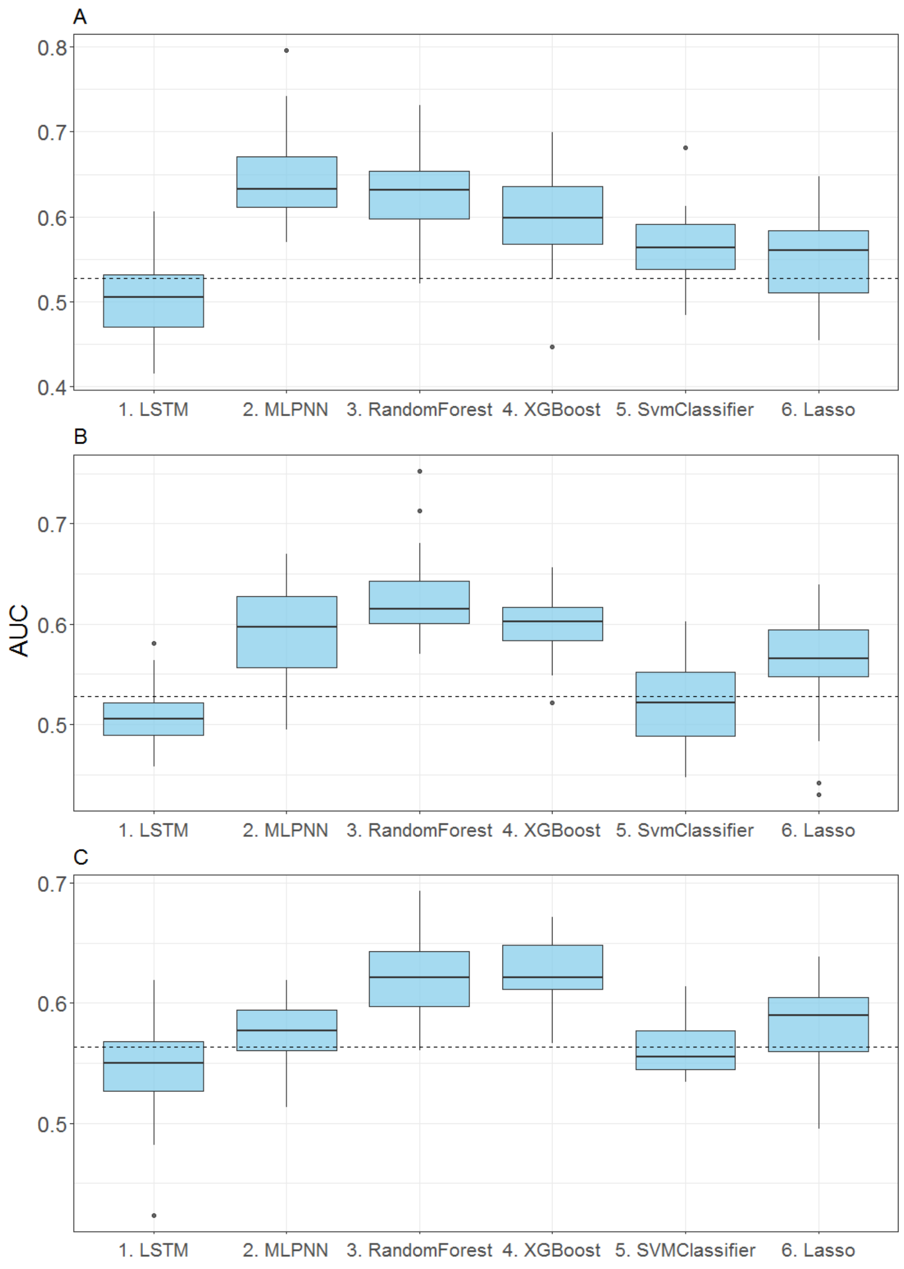

Figure 10: Prediction of host CRC progression state performance by classifier.

A. Classifiers trained on InterProScan IPR terms. B. Classifiers trained on InterProScan GO terms. C. Classifiers trained on HUMAnN GO terms. Dotted line shows performance expected of baseline classifier. y-axis shows AUC performance. A total of 20 train/test cycles are included in the results for each algorithm. Dotted line marks the size of the largest class, an effective baseline performance rating.

For CRC stage prediction the same settings were used as for host gender prediction. The results of CRC stage prediction appear to have higher AUCs the results than for host gender prediction. The classification task is a binary classification with groupings of advanced adenomas and cancer versus the rest of the samples as controls.

We find that the results are best for the MLPNN, random forest and XGBoost classifiers for both host gender prediction and CRC host prediction. The dataset with InterProScan IPR terms and the dataset with IPR GO terms seem to yield the best results based on AUC relative to baseline. The difference in variance in the results between the smaller dataset from InterProScan and the HUMAnN dataset is also noticeably smaller for CRC stage prediction than host gender prediction.

Classifier	**Mean AUC**	**St. Dev**	**95% CI**	**Baseline AUC**
LSTM	0.5021	0.0516	0.3989 - 0.6052	0.5277
MLPNN	0.6438	0.0521	0.5411 - 0.7496	0.5277
Random Forest	0.6288	0.0580	0.5127 - 0.7448	0.5277
XGBoost	0.5978	0.0587	0.4804 - 0.7153	0.5277
SVM	0.5641	0.0418	0.4804 - 0.6478	0.5277
Lasso	0.5533	0.0504	0.4525 - 0.6542	0.5277

Table 9: Results for classifiers trained with InterProScan IPR terms.

Classifier	**Mean AUC**	**St. Dev**	**95% CI**	**Baseline AUC**
LSTM	0.5085	0.0305	0.4474 - 0.5695	0.5277
MLPNN	0.5891	0.0534	0.4822 - 0.6959	0.5277
Random Forest	0.6271	0.0460	0.5352 - 0.7190	0.5277
XGBoost	0.5983	0.0331	0.5320 - 0.6645	0.5277
SVM	0.5204	0.0504	0.4196 - 0.6212	0.5277
Lasso	0.5658	0.0578	0.4501 - 0.6814	0.5277

Table 10: Results for classifiers trained with InterProScan GO terms.

Classifier	Mean AUC	St. Dev	95% CI	Baseline AUC
LSTM	0.5366	0.0486	0.4392 - 0.6338	0.5636
MLPNN	0.5742	0.0265	0.5522 - 0.6271	0.5636
Random Forest	0.6213	0.0321	0.5571 - 0.6855	0.5636
XGBoost	0.6252	0.0317	0.5617 - 0.6888	0.5636
SVM	0.5630	0.0220	0.5190 - 0.6071	0.5636
Lasso	0.5810	0.0332	0.5145 - 0.6475	0.5636

Table 11: Results for classifiers trained with HUMAnN GO terms.

Tables 9 to 11 show the results for the classifiers trained with their respective datasets.

8.4 Interpretation of Results

Using the LIME library we were able to deduct which features made the model classify any given sample as one class or another class. Running LIME on such a large dataset as we have even after preprocessing was relatively resource intensive. Therefore we selected only to perform the analysis on the best candidates. In this instance XGBoost and MLPNN. We found that XGBoost and Random forest have similar results for classification, however the interpretations for XGBoost had higher individual values than for random forest. Thus they give more value when trying to pinpoint key features associated with the later stages of CRC development.

Given a possibility that our models might select different features and feature ranges and important for each model training we decided to apply the interpretation for each test set in every cross-validation. This gives us a wider range of features. As referenced earlier we forced some level of discretization of values for the continuous features. Operating with quartiles we have 4 fold the amount of distinct possible feature ranges. For every interpretation we selected the 100 most important feature ranges.

All results presented are from 5-fold cross validations. Where each sample in the test set for each fold is run through the LIME interpreter.

8.4.1 Interpretation of Host Gender Prediction

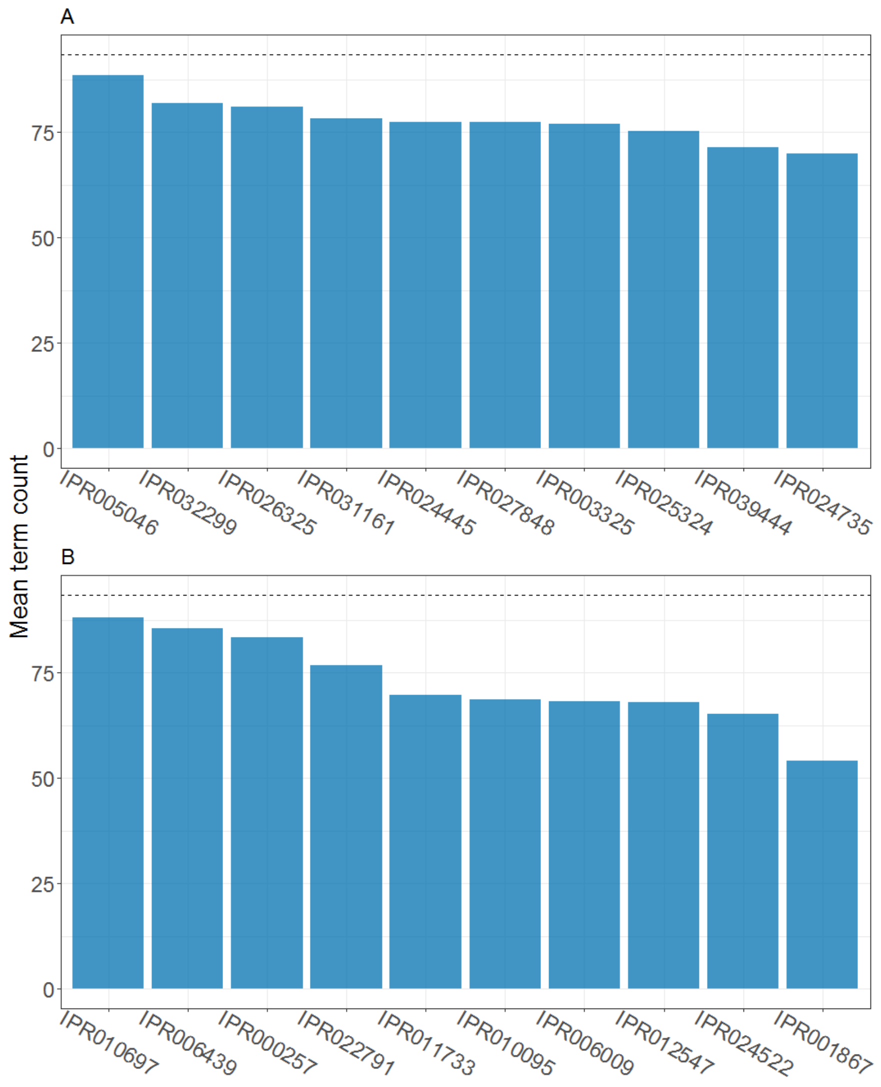

Figure 11: 10 most frequently occurring InterProScan IPR terms from MLPNN and XGBoost classifiers trained for host gender classification on the InterProScan IPR dataset and interpreted by LIME

A. Most common terms for MLPNN classifier trained for host gender prediction on InterProScan IPR terms. B. Most common terms for XGBoost classifier prediction trained for host gender prediction on InterProScan IPR terms. Values averaged from test cycles for classifiers with 5-fold cross validation. Dotted line shows highest possible value if present in all samples.

Figure 11 shows the most frequently occurring terms from the MLPNN and XGBoost classifiers on the InterProScan IPR dataset. This gives us an indication of which terms are deemed important. Note that they do not have an overlap. This implies that the features being deemed important for each classifier are different.

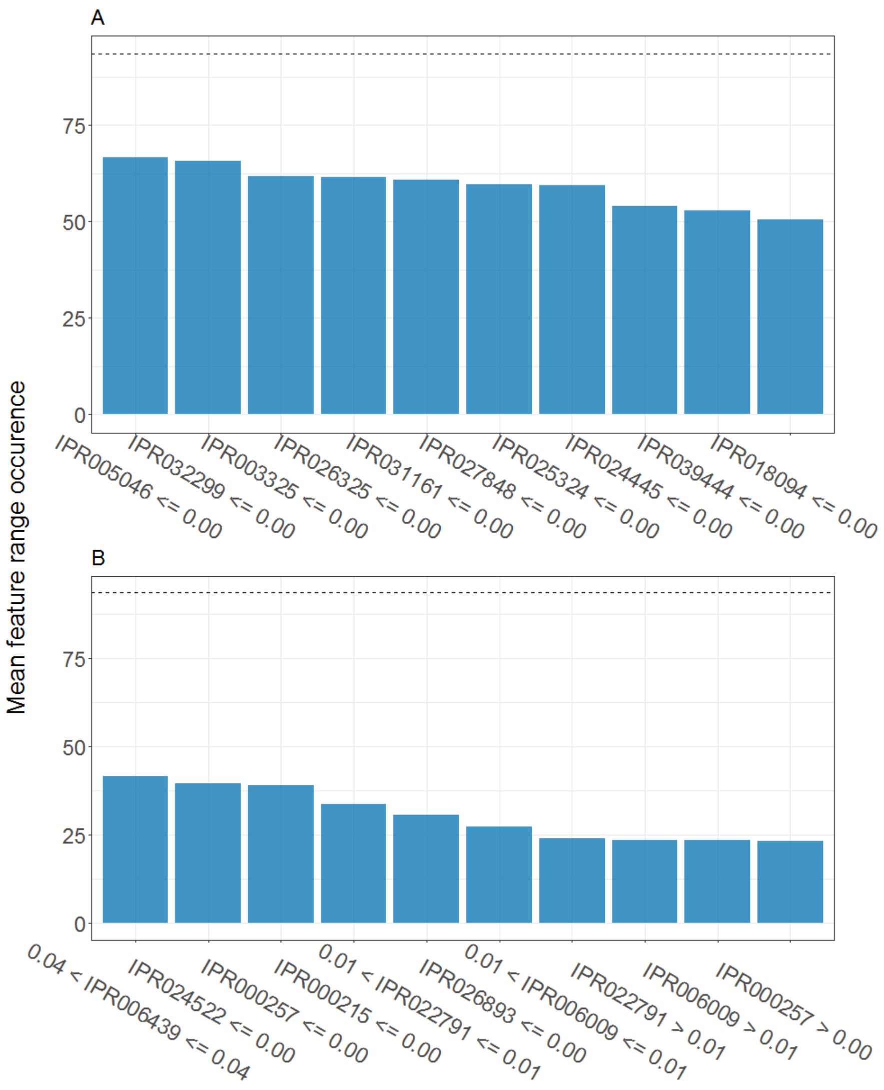

Figure 12: 10 most frequently occurring InterProScan IPR feature ranges from MLPNN and XGBoost classifiers trained for host gender classification on the InterProScan IPR dataset and interpreted by LIME

A. Most common terms value ranges for MLPNN classifier trained for host gender prediction on InterProScan IPR terms. B. Most common terms value ranges for XGBoost classifier trained for host gender prediction on InterProScan IPR terms. Values averaged from test cycles for classifiers with 5-fold cross validation. Dotted line shows highest possible value if present in all samples.

Figure 12 shows the specific term value ranges that were the most frequently occurring from the MLPNN and XGBoost classifiers on the InterProScan IPR dataset when predicting host gender. Note that they do not have an overlap. This implies that the features being deemed important for each classifier are different.

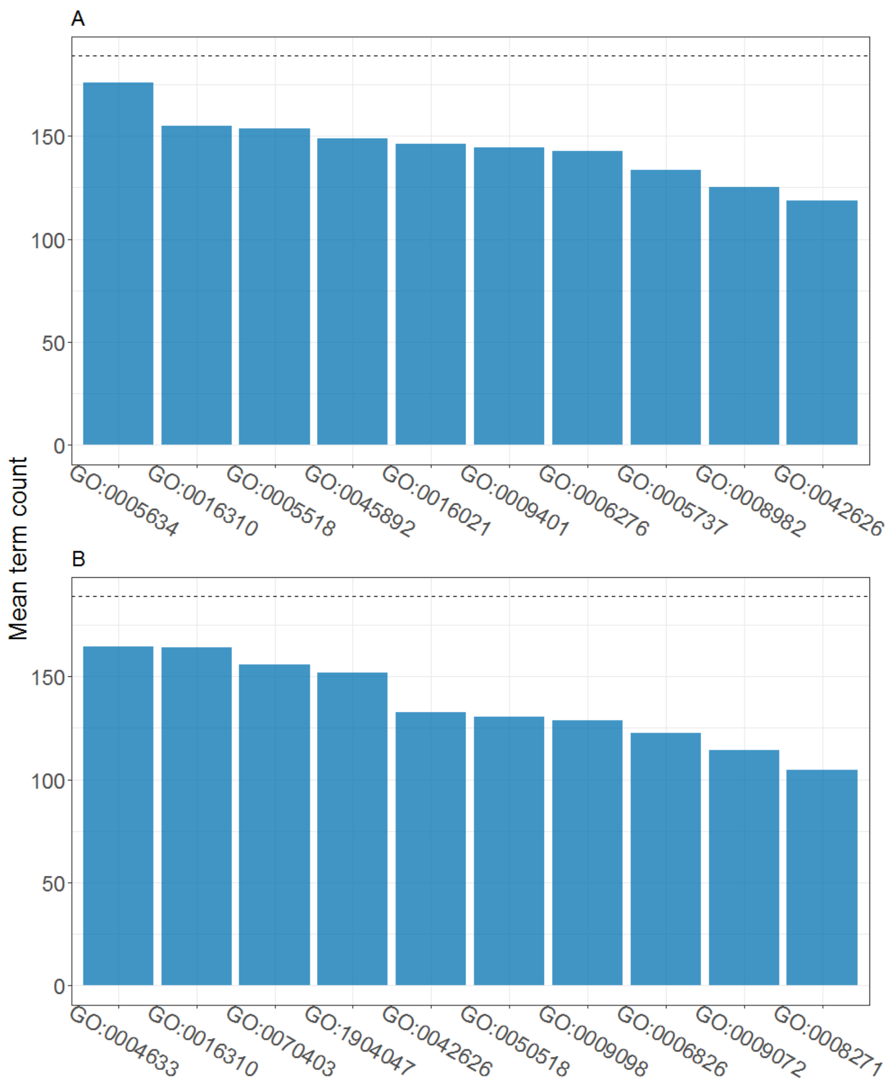

Figure 13: 10 most frequently occurring HUMAnN GO terms from MLPNN and XGBoost classifiers trained for host gender classification on the InterProScan IPR dataset and interpreted by LIME

A. Most common terms for MLPNN classifier for host gender prediction trained on HUMAnN Go term. B. Most common terms for XGBoost classifier trained for host gender prediction on HUMAnN GO terms. Values averaged from test cycles for classifiers with 5-fold cross validation. Dotted line shows highest possible value if present in all samples.

Figure 13 shows the most frequently occurring terms from the MLPNN and XGBoost classifiers on the HUMAnN GO dataset when predicting host gender. The numbers are higher here due to the larger sample size of the HUMAnN GO dataset.

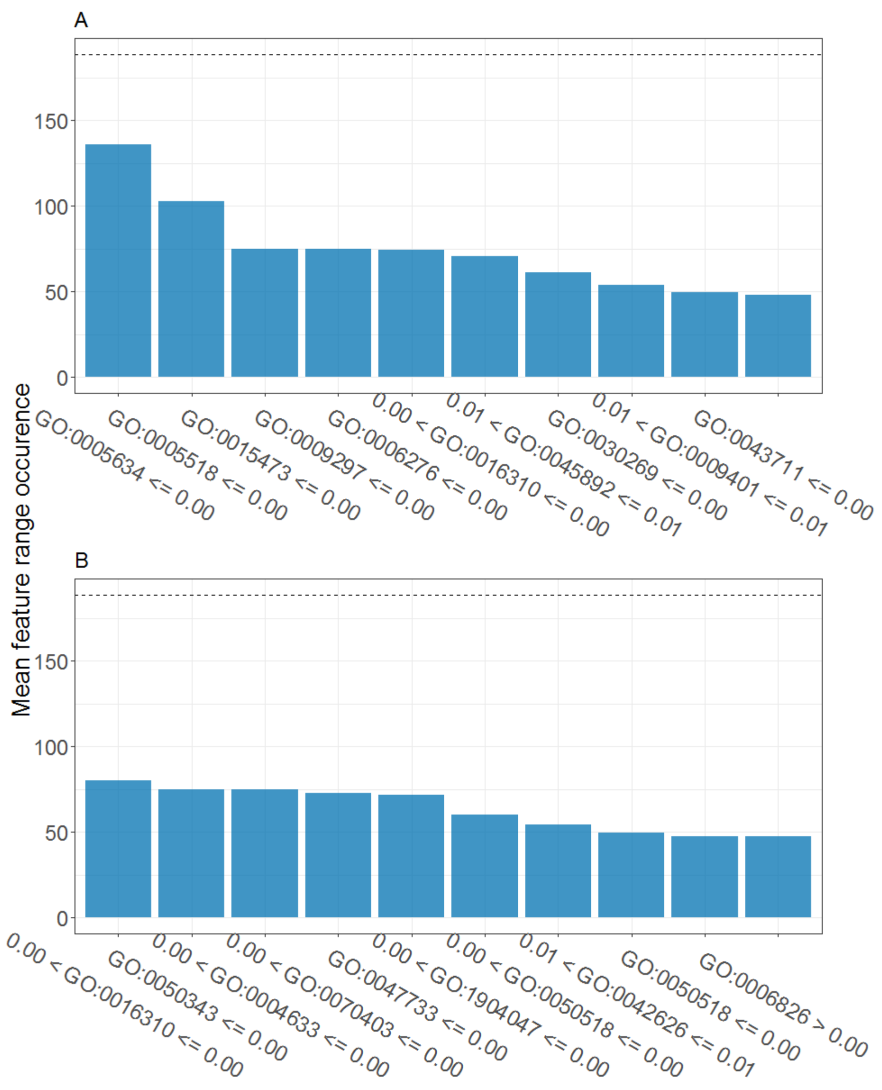

Figure 14: 10 most frequently occurring HUMAnN GO feature ranges from MLPNN and XGBoost classifiers trained for host gender classification on the InterProScan IPR dataset and interpreted by LIME

A. Most common terms for MLPNN Classifier trained on HUMAnN Go term. B. Most common terms for XGBoost classifier trained on HUMAnN GO terms. Values averaged from test cycles for classifiers with 5-fold cross validation. Dotted line shows highest possible value if present in all samples.

Figure 14 shows the most frequently occurring terms from the MLPNN and XGBoost classifiers on the HUMAnN GO dataset when predicting host gender. The numbers are higher here due to the larger sample size of the HUMAnN GO dataset.

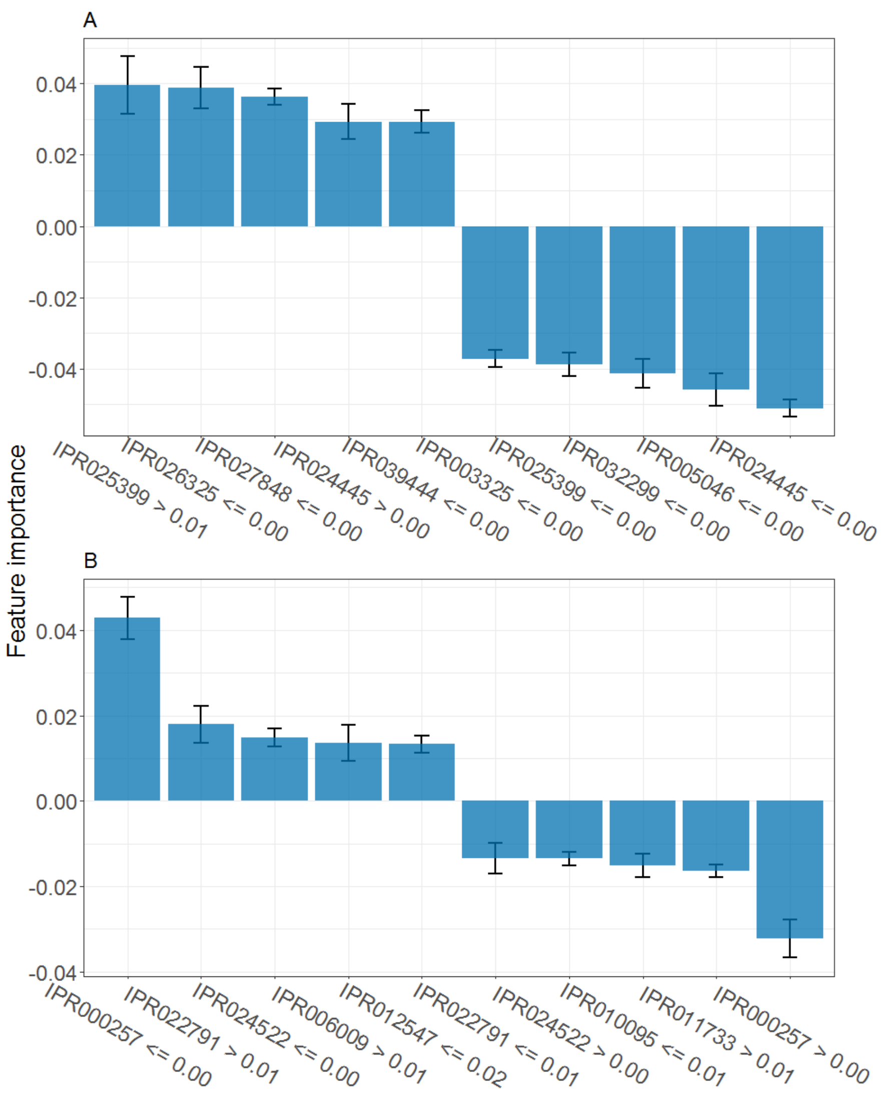

Figure 15: Figure of the 10 most important feature ranges for classification as male or female for InterProScan IPR terms from trained classifiers interpreted by LIME.

A. Interpretation of MLPNN trained on InterProScan IPR terms. B. Interpretation of XGBoost trained on InterProScan IPR terms. Positive values contribute towards classification as male, negative values contribute towards classification as female. y-axis is impact size as a share of one. x-axis show features and their respective value range.

Figure 15 shows the important feature values for the classification of samples. A value of 0.03 here indicates a 3% increased probability of a certain sample being classified as coming from a male. Note that the values are lower for XGBoost than MLPNN. XGBoost applies lower impact values for individual feature values in our model for the InterProScan IPR dataset.

Term Range	**Mean Importance**	**St. Dev**	**Short Description**
IPR025399 > 0.01	0.03954	0.00816	Domain of unknown function DUF4372
IPR026325 <= 0.00	0.03889	0.00581	Protein of unknown function DUF932
IPR027848 <= 0.00	0.03633	0.00231	Protein of unknown function DUF4494
IPR024445 > 0.00	0.02929	0.00497	SXO2-like transposase domain
IPR039444 <= 0.00	0.02929	0.00322	SIR2-like domain
IPR003325 <= 0.00	-0.03712	0.00251	TerD domain
IPR025399 <= 0.00	-0.03878	0.00322	Domain of unknown function DUF4372
IPR032299 <= 0.00	-0.04130	0.00414	Protein of unknown function DUF4843
IPR005046 <= 0.00	-0.04585	0.00450	Protein of unknown function DUF285
IPR024445 <= 0.00	-0.05107	0.00246	SXO2-like transposase domain

Table 12: Table of the 10 most important feature ranges for classification as male or female for InterProScan IPR terms from MLPNN classifier interpreted by LIME. Positive values male. Negative values female.

Term Range	**Mean Importance**	**St. Dev**	**Short Description**
IPR000257 <= 0.00	0.04282	0.00497	Uroporphyrinogen decarboxylase (URO-D)
IPR022791 > 0.01	0.01797	0.00435	Lysylphosphatidylglycerol synthetase/ glycosyltransferase AglD
IPR024522 <= 0.00	0.01486	0.00213	Protein of unknown function DUF3789
IPR006009 > 0.01	0.1359	0.00415	N-acetylglucosaminyltransferase, MurG
IPR012547 <= 0.02	0.01333	0.00201	PD-(D/E)XK nuclease superfamily 9
IPR022791 <= 0.01	-0.01348	0.00355	Lysylphosphatidylglycerol synthetase/ glycosyltransferase AglD
IPR024522 > 0.00	-0.01354	0.00163	Protein of unknown function DUF3789
IPR010095 <= 0.01	-0.01510	0.00273	Transposase IS605, OrfB, C-terminal
IPR011733 > 0.01	-0.01638	0.00151	Conserved hypothetical CHP02185, integral membrane
IPR000257 > 0.00	-0.03225	0.00449	Uroporphyrinogen decarboxylase (URO-D)

Table 13: Table of the 10 most important feature ranges for classification as male or female for InterProScan IPR terms from XGBoost classifier interpreted by LIME. Positive values male. Negative values female.

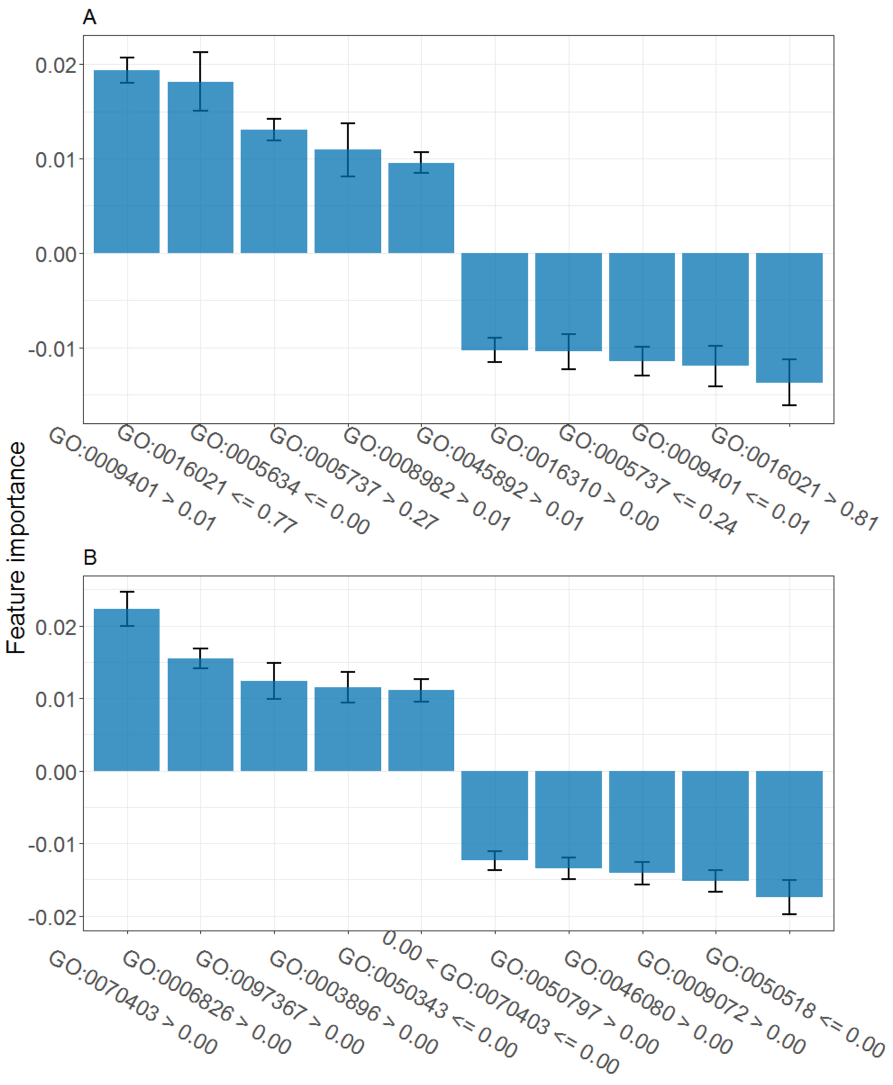

Figure 16: Figure of the 10 most important feature ranges for classification as male or female for HUMAnN GO terms from trained classifiers interpreted by LIME.

A. Interpretation of MLPNN trained on HUMAnN GO terms. B. Interpretation of XGBoost trained on HUMAnN GO terms. Positive values contribute towards classification as male, negative values contribute towards classification as female. x-axis show features and their respective value range. y-axis is impact size as a share of one.

Figure 16 shows the 10 most important feature ranges for the classifiers for on the HUMAnN GO dataset. Here the difference between the MLPNN classifier and the XGBoost classifier in terms of the magnitude of impact of the individual features is smaller than for the InterProScan IPR dataset.

Term Range	**Mean Importance**	**St. Dev**	**Short Description**
GO:0009401 > 0.01	0.01936	0.00135	phosphoenolpyruvate-dependent sugar phosphotransferase system
GO:0016021 <= 0.077	0.01814	0.00307	integral component of membrane
GO:0005634 <= 0.00	0.01305	0.00110	nucleus
GO:0005737 > 0.27	0.01090	0.00281	cytoplasm
GO:0008982 > 0.01	0.00955	0.00109	protein-N(PI)-phosphohistidine-sugar phosphotrans-ferase activity
GO:0045892 > 0.01	-0.01029	0.00125	negative regulation of transcription, DNA-templated
GO:0016310 > 0.00	-0.01043	0.00186	phosphorylation
GO:0005737 <= 0.24	-0.01148	0.00154	cytoplasm
GO:0009401 <= 0.01	-0.01199	0.00215	phosphoenolpyruvate-dependent sugar phosphotransferase system
GO:0016021 > 0.81	-0.01375	0.00244	integral component of membrane

Table 14: Table of the 10 most important feature ranges for classification as male or female for HUMAnN GO terms from MPLNN classifier interpreted by LIME. Positive values male. Negative values female.

Term Range	**Mean Importance**	**St. Dev**	**Short Description**
GO:0070403 > 0.00	0.02233	0.00237	NAD+ binding
GO:0006826 > 0.00	0.01548	0.00135	iron ion transport
GO:0097367 > 0.00	0.01239	0.00252	carbohydrate derivative binding
GO:0003896 > 0.00	0.01154	0.00209	DNA primase activity
GO:0050343 <= 0.00	0.01111	0.00155	trans-2-enoyl-CoA reductase (NAD+) activity
0.00 < GO:0070403 <= 0.00	-0.01238	0.00125	NAD+ binding
GO:0050797 > 0.00	-0.01344	0.00152	thymidylate synthase (FAD) activity
GO:0046080 > 0.00	-0.01412	0.00151	dUTP metabolic process
GO:0009072 > 0.00	-0.01514	0.00150	aromatic amino acid family metabolic process
GO:0050518 <= 0.00	-0.01743	0.00236	2-C-methyl-D-erythritol 4-phosphate cytidylyltransferase activity

Table 15: Table of the 10 most important feature ranges for classification as male or female for HUMAnN GO terms from XGBoost classifiers interpreted by LIME. Positive values male. Negative values female.

8.4.2 Interpretation of Colorectal Cancer Developement Stage Prediction

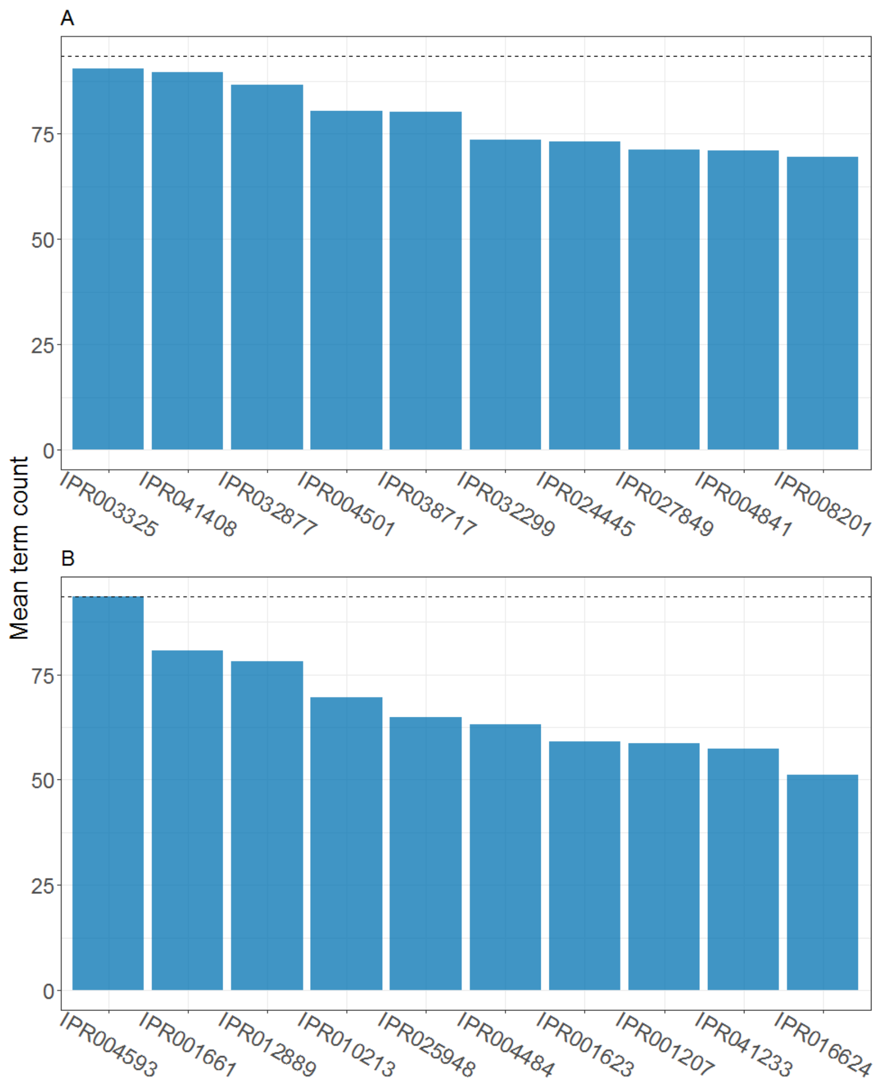

Figure 17: 10 most frequently occurring InterProScan IPR terms from MLPNN and XGBoost LIME interpretation

A. Most common terms for MLPNN classifier trained for CRC development stage prediction on InterProScan IPR terms. B. Most common terms for XGBoost classifier trained for CRC development stage on InterProScan IPR terms. Values averaged from test cycles for classifiers with 5-fold cross validation. Dotted line shows highest possible value if present in all samples.

Figure 17 shows the most frequently occurring terms from the MLPNN and XGBoost classifiers on the InterProScan IPR dataset. This gives us an indication of which terms are deemed important. Note that they do not have an overlap. This implies that the features being deemed important for each classifier are different.

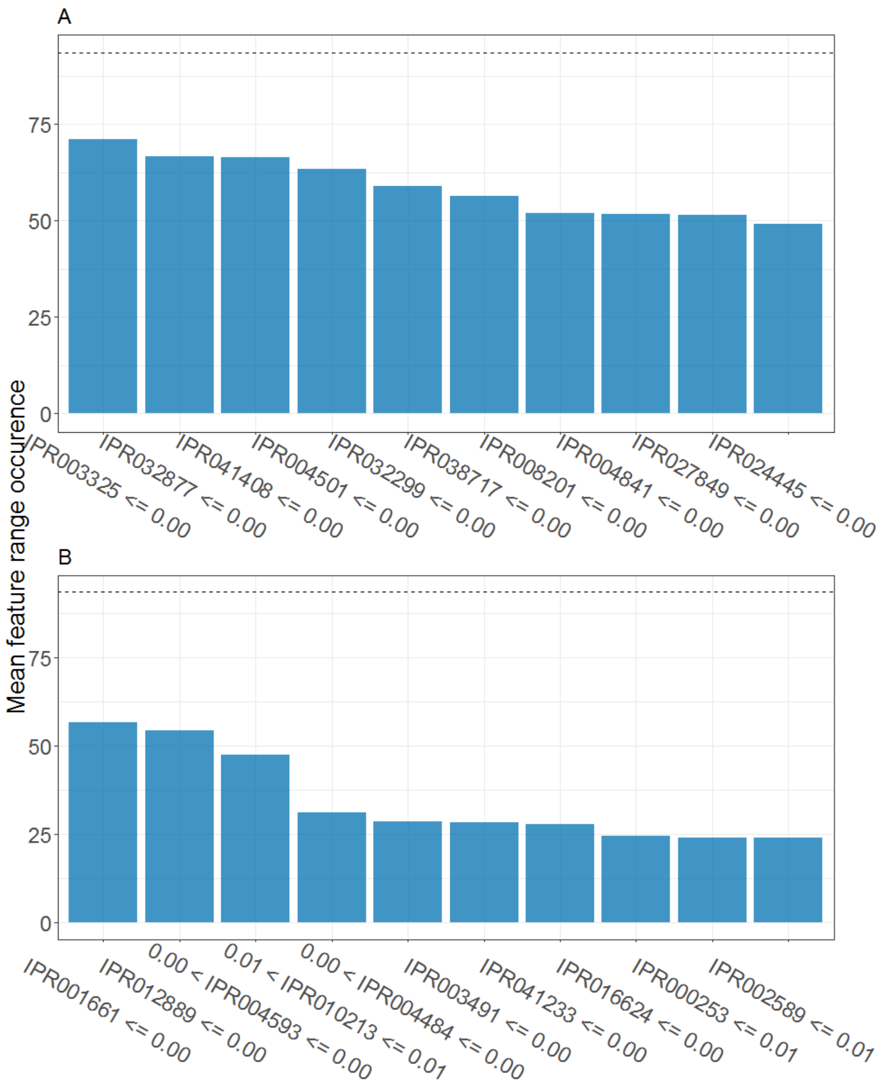

Figure 18: 10 most frequently occurring InterProScan IPR feature ranges from MLPNN and XGBoost LIME interpretation

A. Most common feature values for MLPNN classifier trained for CRC development stage prediction on InterProScan IPR terms. B. Most common feature values for XGBoost classifier trained for CRC development stage on InterProScan IPR terms. Values averaged from test cycles for classifiers with 5-fold cross validation. Dotted line shows highest possible value if present in all samples.

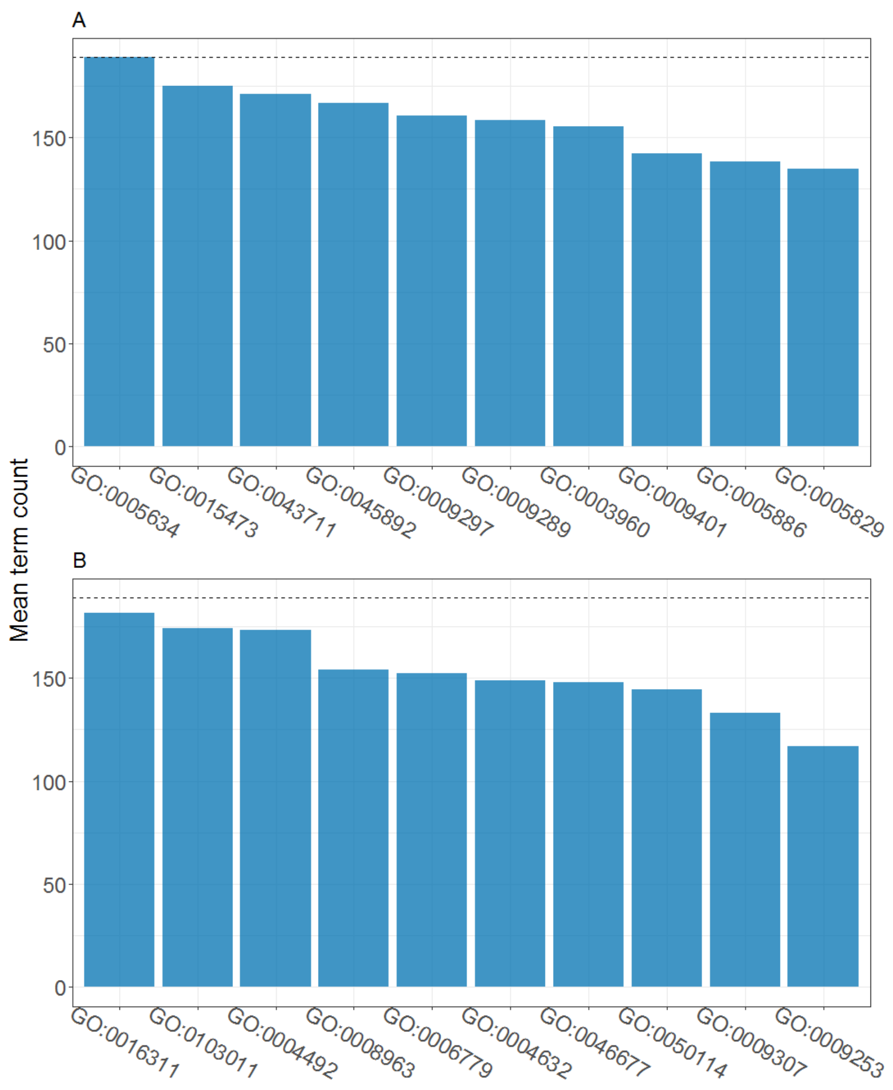

Figure 19: 10 most frequently occurring HUMAnN GO terms from MLPNN and XGBoost LIME interpretation

A. Most common terms for MLPNN classifier trained for CRC development stage prediction on HUMAnN GO terms. B. Most common terms for XGBoost classifier trained for CRC development stage on HUMAnN GO terms. Values averaged from test cycles for classifiers with 5-fold cross validation. Dotted line shows highest possible value if present in all samples.

Figure 19 shows the most frequently occurring terms from the MLPNN and XGBoost classifiers on the HUMAnN GO dataset. The numbers are higher here due to the larger sample size of the HUMAnN GO dataset.

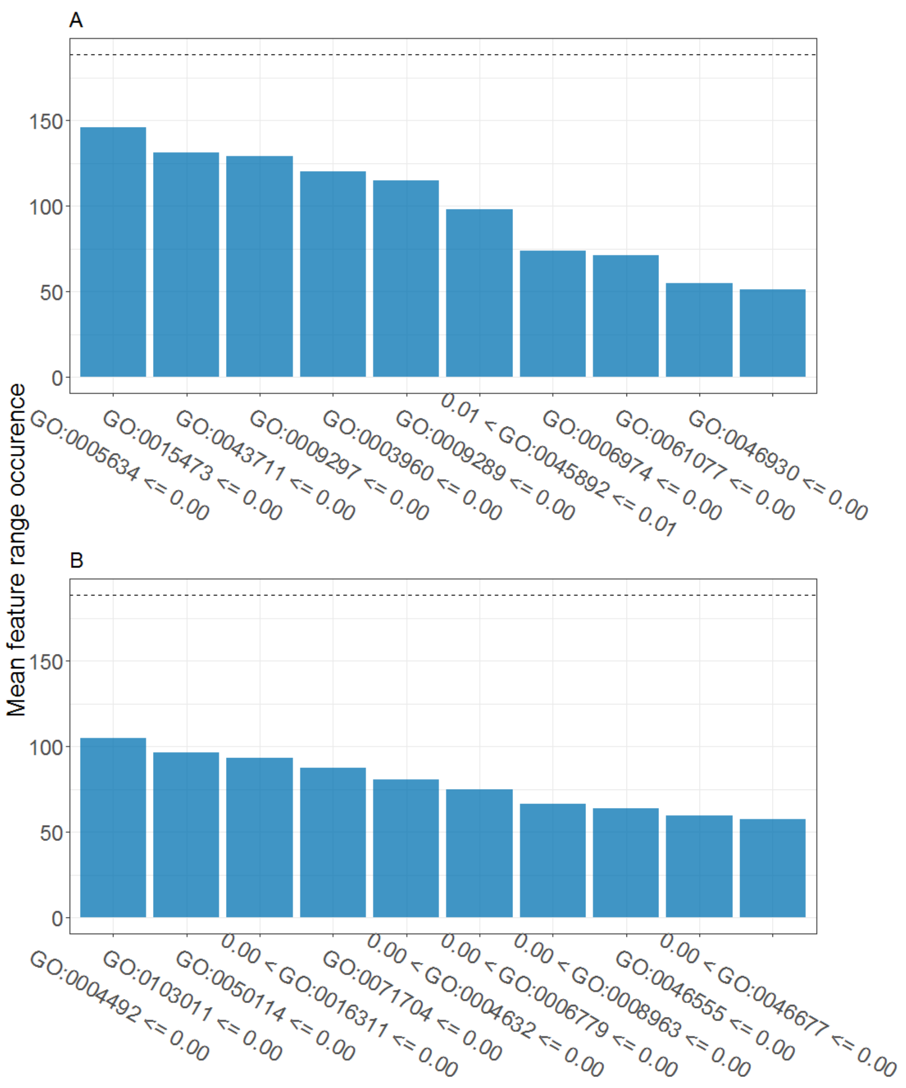

Figure 20: 10 most frequently occurring HUMAnN GO feature ranges from MLPNN and XGBoost LIME interpretation

A. Most common feature values for MLPNN classifier trained for CRC development stage prediction on HUMAnN GO terms. B. Most common feature values for XGBoost classifier trained for CRC development stage on HUMAnN GO terms. Values averaged from test cycles for classifiers with 5-fold cross validation. Dotted line shows highest possible value if present in all samples.

Figure 20 shows the most frequently occurring terms from the MLPNN and XGBoost classifiers on the HUMAnN GO dataset. The numbers are higher here due to the larger sample size of the HUMAnN GO dataset.

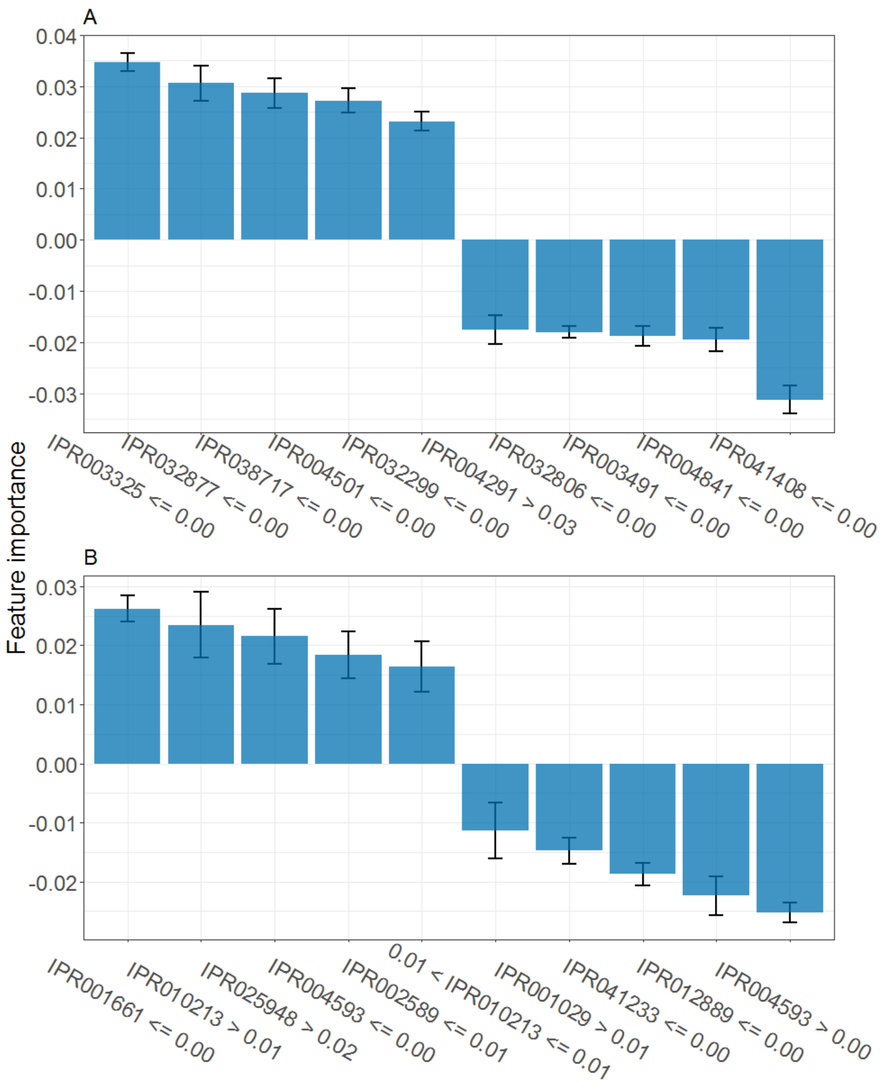

Figure 21: 10 most important feature ranges for cancer or advanced adenoma classification and control classification for InterProScan IPR terms from trained classifiers interpreted by LIME.

A. Interpretation of MLPNN trained on InterProScan IPR terms. B. Interpretation of XGBoost trained on InterProScan IPR terms. Positive values contribute towards classification as cancer or advanced adenoma, negative values contribute towards classification as control. y-axis is impact size as a share of one. x-axis show features and their respective value range.

Term Range	**Mean Importance**	**St. Dev**	**Short Description**
IPR003325 <= 0.00	0.03471	0.00181	TerD domain
IPR032877 <= 0.00	0.03061	0.00340	Transposase IS204/IS1001/IS1096 /IS1165, helix-turn-helix domain
IPR038717 <= 0.00	0.02867	0.00294	Tc1-like transposase, DDE domain
IPR004501 <= 0.00	0.02723	0.00240	Phosphotransferase system, EIIC component, type 3
IPR032299 <= 0.00	0.02318	0.00184	Protein of unknown function DUF4843
IPR041408 <= 0.00	-0.03123	0.00281	Hemolysin coregulated protein (Hcp) TssD
IPR004841 <= 0.00	-0.01948	0.00116	Amino acid permease/ SLC12A domain
IPR003491 <= 0.00	-0.01875	0.00195	Replication initiation factor
IPR032806 <= 0.00	-0.01806	0.00233	H repeat-associated protein, N-terminal
IPR004291 > 0.03	-0.01752	0.00281	Transposase, IS66

Table 16: Overview of the 5 most important feature ranges for cancer or advanced adenoma and the 5 important feature ranges for control classification for InterProScan IPR terms from MLPNN classifier interpreted by LIME.

Term Range	Mean Importance	St. Dev	Short Description
IPR001661 <= 0.00	0.02623	0.00220	Glycoside hydrolase, family 37
IPR010213 > 0.01	0.02351	0.00555	Transcription termination factor NusA
IPR025948 > 0.02	0.02153	0.00468	HTH-like domain
IPR004593 <= 0.00	0.01840	0.00401	Nuclease SbcCD subunit D
IPR002589 <= 0.01	0.01637	0.00430	A1pp domain
IPR004593 > 0.00	-0.02526	0.00480	Nuclease SbcCD subunit D
IPR012889 <= 0.00	-0.02237	0.00227	L-fucose isomerase, N-terminal-2
IPR041233 <= 0.00	-0.01872	0.00192	Alpha galactosidase, C-terminal beta sandwich domain
IPR001029 > 0.01	-0.01480	0.00329	Flagellin, D0/D1 domain
0.01 < IPR010213 <= 0.01	-0.01137	0.00169	Transcription termination factor NusA

Table 17: Overview of the 5 most important feature ranges for cancer or advanced adenoma and the 5 important feature ranges for control classification for InterProScan IPR terms from XGBoost classifier interpreted by LIME.

Figure 21 shows the important feature values for the classification of samples. A value of 0.03 here indicates a 3% increased probability of a certain sample being classified as cancer or advanced adenoma. Note that the values are lower for XGBoost than MLPNN. XGBoost applies lower impact values for individual feature values in our model for the InterProScan IPR dataset.

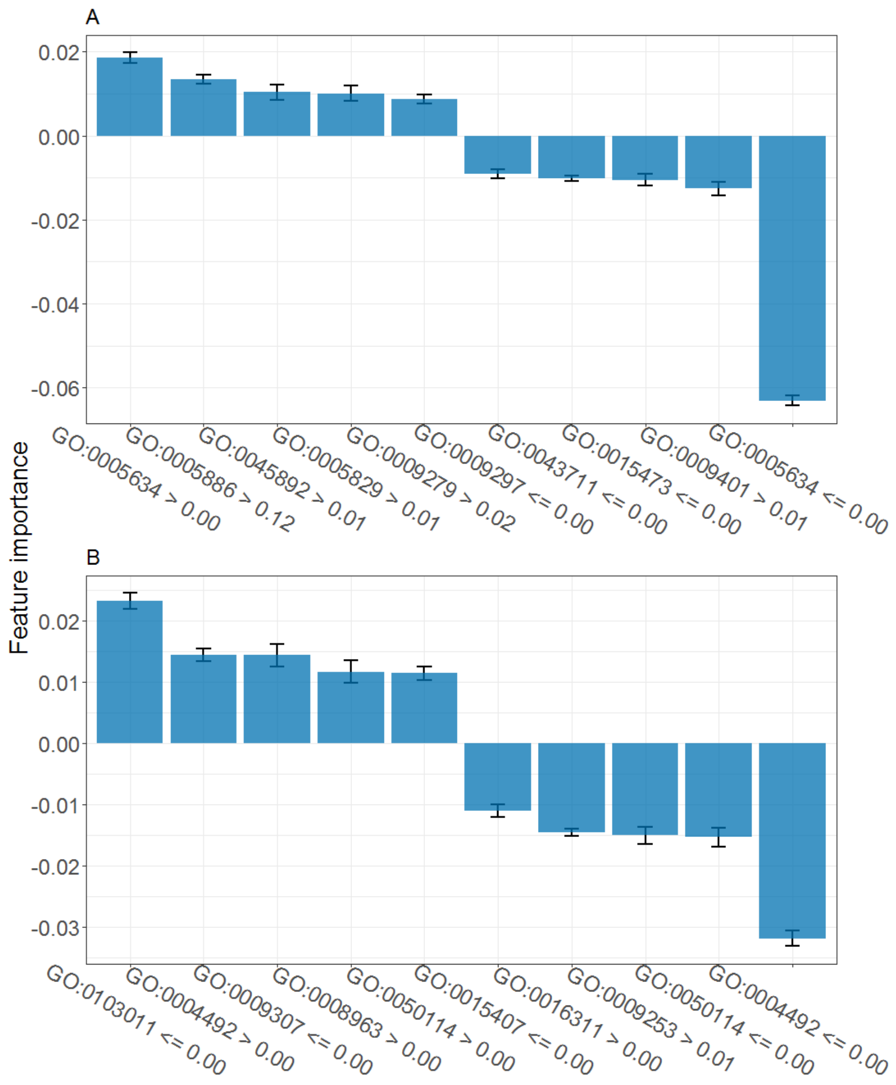

Figure 22: Figure of the 5 most important feature ranges for cancer or advanced adenoma and the 5 important feature ranges for control classification for HUMAnN GO terms from trained classifiers interpreted by LIME.

A. Interpretation of MLPNN trained on HUMAnN GO terms. B. Interpretation of XGBoost trained on HUMAnN GO terms. Positive values contribute towards classification as cancer or advanced adenoma, negative values contribute towards classification as control. x-axis show features and their respective value range. y-axis is impact size as a share of one.

Term Range	**Mean Importance**	**St. Dev**	**Short Description**
GO:0005634 > 0.00	0.01847	0.00215	nucleus
GO:0005886 > 0.12	0.01332	0.00216	plasma membrane
GO:0045892 > 0.01	0.01032	0.00056	negative regulation of transcription, DNA-templated
GO:0005829 > 0.01	0.01006	0.00104	cytosol
GO:0009279 > 0.02	0.00878	0.00086	cell outer membrane
GO:0005634 <= 0.00	-0.06302	0.00215	nucleus
GO:0009401 > 0.01	-0.01258	0.00182	phosphoenolpyruvate-dependent sugar phosphotransferase system
GO:0015473 <= 0.00	-0.01053	0.00042	fimbrial usher porin activity
GO:0043711 <= 0.00	-0.01017	0.00041	pilus organization
GO:0009297 <= 0.00	-0.00906	0.00052	pilus assembly

Table 18: Overview of the 5 most important feature ranges for cancer or advanced adenoma and the 5 important feature ranges for control classification for HUMAnN GO terms from MLPNN classifier interpreted by LIME.

Term Range	**Mean Importance**	**St. Dev**	**Short Description**
GO:0103011 <= 0.00	0.02324	0.00128	mannosylfructose-phosphate synthase activity
GO:0004492 > 0.00	0.01439	0.00105	methylmalonyl-CoA decarboxylase activity
GO:0009307 <= 0.00	0.01439	0.00184	DNA restriction-modification system
GO:0008963 > 0.00	0.01167	0.00185	phospho-N-acetylmuramoyl-pentapeptide-transferase activity
GO:0050114 > 0.00	0.01143	0.00107	myo-inosose-2 dehydratase activity
GO:0004492 <= 0.00	-0.03191	0.00102	methylmalonyl-CoA decarboxylase activity
GO:0050114 <= 0.00	-0.01536	0.00061	myo-inosose-2 dehydratase activity
GO:0009253 > 0.01	-0.01506	0.00140	peptidoglycan catabolic process
GO:0016311 > 0.00	-0.01455	0.00160	dephosphorylation
GO:0015407 <= 0.00	-0.01107	0.00126	ABC-type monosaccharide transporter activity

Table 19: Overview of the 5 most important feature ranges for cancer or advanced adenoma and the 5 important feature ranges for control classification for HUMAnN GO terms from XGBoost classifiers interpreted by LIME.

8.5 Execution Times for Classifiers

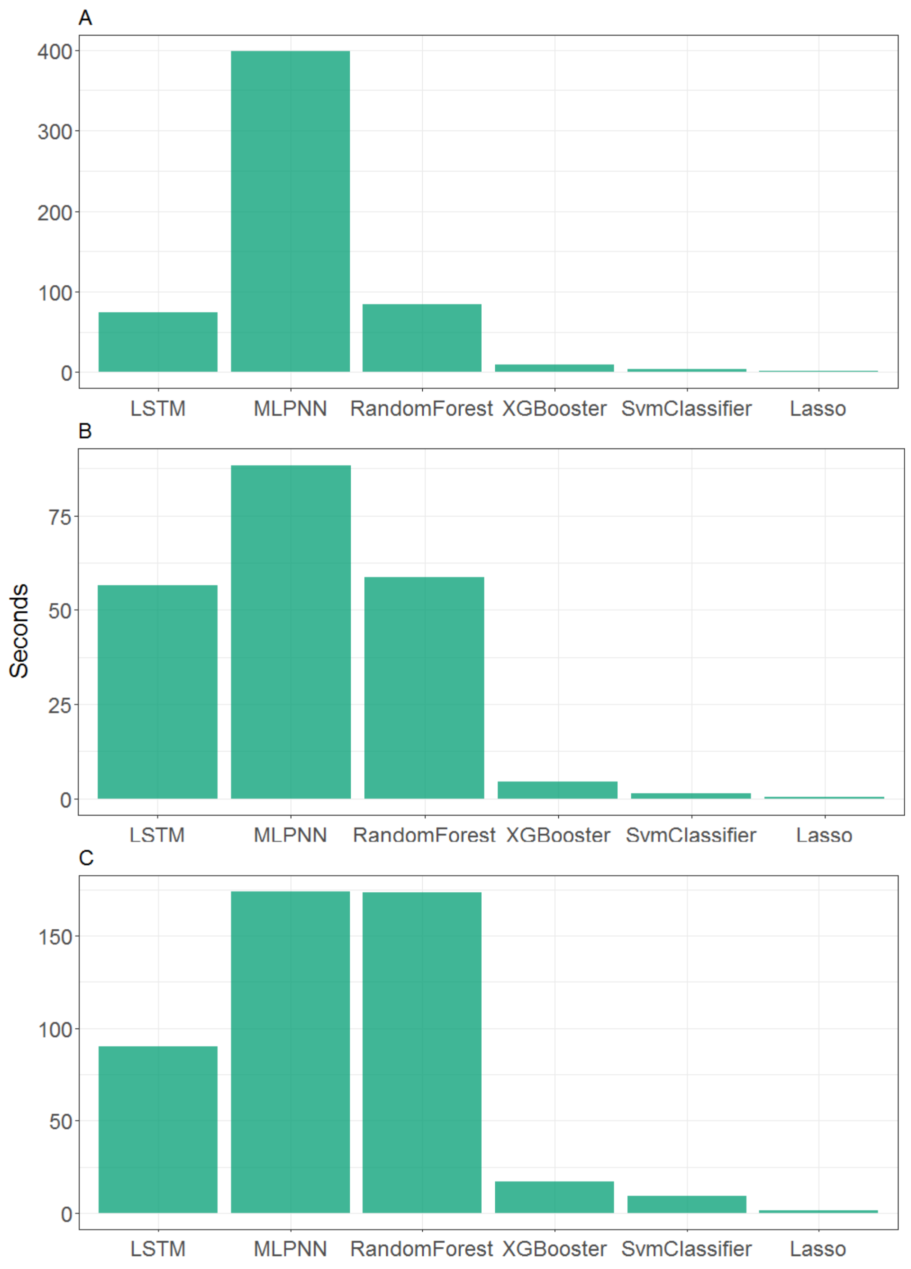

Figure 23: An overview of the execution times for the classifiers for gender on the datasets.

A. Classifiers trained on InterProScan IPR terms. B. Classifiers trained on InterProScan GO terms. C. Classifiers trained on HUMAnN GO terms. Execution time for script with 5 k-fold splitting and no interpretation. Y-axis in Log10 scale.

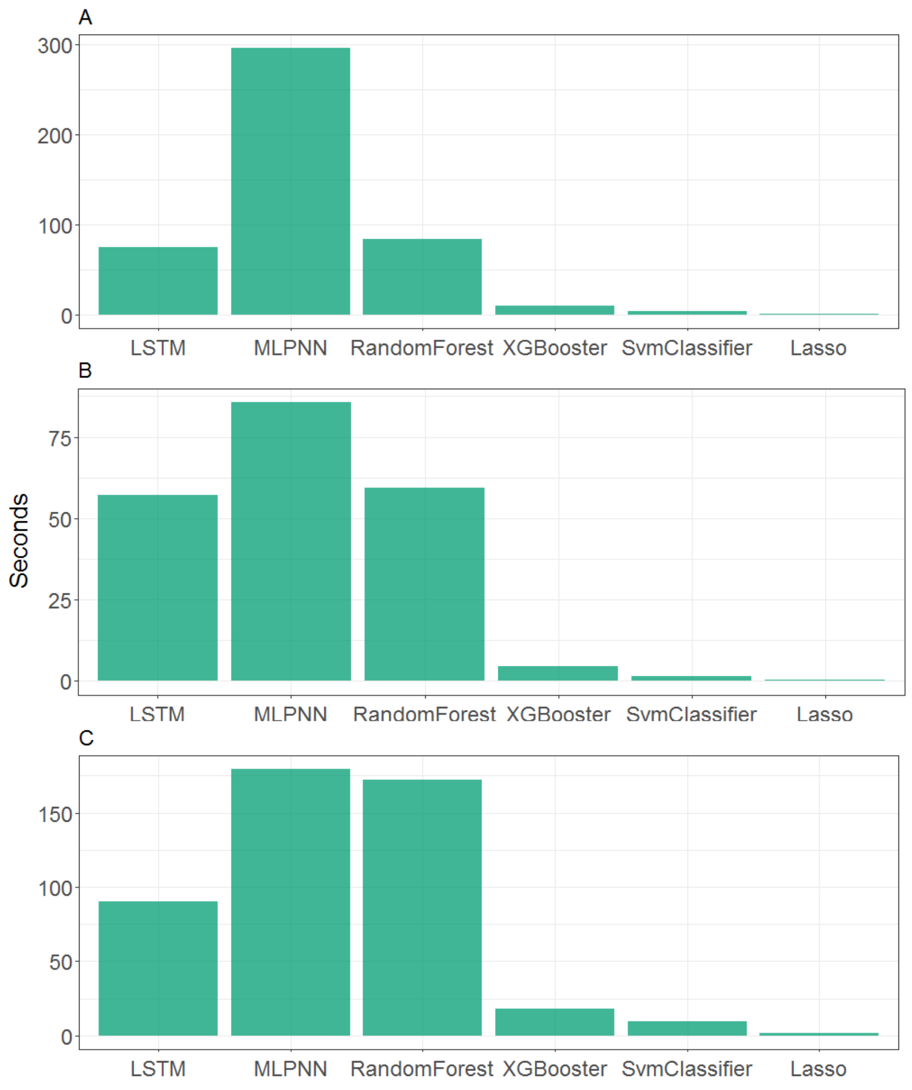

Figure 24: An overview of the execution times for the classifiers for cancer on the datasets.

A. Classifiers trained on InterProScan IPR terms. B. Classifiers trained on InterProScan GO terms. C. Classifiers trained on HUMAnN GO terms. Execution time for script with 5 k-fold splitting and no interpretation.

Figure 23 and 24 show a comparison of the execution times for the classifiers. The larger dataset from HUMAnN has slightly longer execution times. As both are binary classification tasks the difference between there is little difference between execution times for CRC stage prediction and host gender prediction. The Neural Network based algorithms and the random forest algorithm are the most demanding while the regression based classifiers take the least amount of time to execute. The execution times are for a set of 5-fold cross-validation and no interpretation.

Due to sharing of resources we did not make full use of the resources available to avoid waiting for resource allocation. We ran sets with 4 tasks in parallel using Snakemake with 24 cores. Effectively 6 cores per algorithm. The exact specifications for the cores available on TSD are not openly specified by TSD.

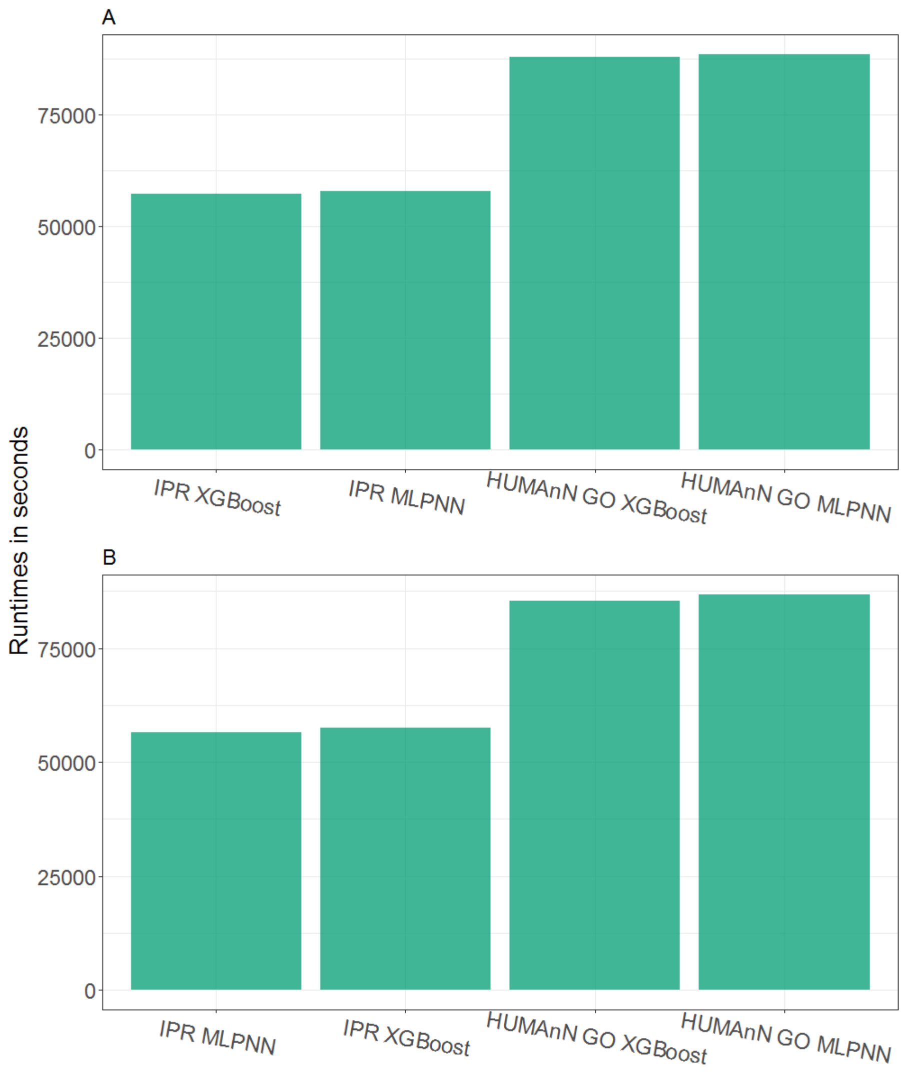

Figure 25: An overview of the execution times for the classifiers on the datasets.

A. Classifiers trained on InterProScan IPR terms with interpretation. B. Classifiers trained on HUMAnN GO terms with interpretation. Execution time for script with 5 k-fold splitting with interpretation.

Figure 25 shows the disparity between execution for classifier with LIME interpretation for a full k-fold cross validation with k = 5. There seems to be not much of a difference in terms of the classification task being performed. It's also clear that the training for the individual algorithms has little bearing on the total execution time for the interpreter.

9 Discussion

The purpose of this master thesis was to build a pipeline for deep analysis of cancer causing genes in the gut microbiome. In this section we will discuss the results for the machine learning pipeline we've developed. We will detail the strengths and weaknesses of our implementation and chart a course for the future expansion on the work.

9.1 Summary

We were able to successfully make a pipeline tool we named MAPLEgf (Multi-Algorithm Pipeline for machine LEarning on gene functions). We preprocessed, trained and interpreted results from a series of machine learning algorithms. MAPLEgf takes a defined set of inputs from two different supported sources and formats. It applies six different machine learning algorithms to the dataset with settings applied from a user configuration file. The results are then interpreted by LIME and we receive an overview of the most influential features.

With the constructed product we have a ready pipeline for final results when all the samples are finished. The results we have obtained so far as laid out in chapter 8 has shown that we have been able to get positive results. As we get more samples preprocessed and ready for MAPLEgf we are likely to get better results. Furthermore a focus of this project was to create interpretable results. We have succeeded in this regard, however, given the variable results from the interpreter some comparison and evaluation of which algorithms provide the most plausible interpretations will be required.

9.2 MAPLEgf

A substantial portion of this master thesis, and hence our results was the development process of a pipeline to conduct analysis of gene functions in the gut microbiome with machine learning algorithms. MAPLEgf is the result of that effort. The latest version of MAPLEgf as of the submission of this master thesis is v0.2. MAPLEgf is freely available on GitHub at `https://github.com/henrikolsvik/maplegf` and is licensed under the MIT Licence. The complete instructions for how to set up MAPLEgf are available in the readme.md file in the MAPLEgf repository. MAPLEgf has been confirmed to work with Python v3.6.2[31], NumPy v1.18.5[73], Snakemake v5.3.0[30], scikit-learn, v0.23.2[46], XGBoost v1.4.2[56] and TensorFlow v2.3.0[66]. Please refer to the requirements.txt file in the MAPLEgf GitHub repository for a complete list of all dependencies and their respective versions.

Figure 2 in the Methodology chapter shows the overall dataflow in MAPLEgf. MAPLEgf has proved a reliable tool for performing machine learning on gene function data, and it can be readily integrated into larger pipelines.

Execution times for MAPLEgf vary depending on the system configurations, but on TSD we were able to consistently make use of 6 AMD CPU cores per Snakemake task. We found that increasing the number of cores allocated to MALPEgf increased the performance, but not linearly. The subtasks making use of scikit-learn and TensorFlow had better multi-core scaling than the data preprocessing subtask (data not shown). MAPLEgf was only tested using CPU performance. TensorFlow has support for GPU accelerated machine learning and enabling that should reduce classifier training time. Memory use was at most 5GiB RAM in the 24 core 4 task configuration.

9.3 Discussion of Classifier Results

The best performing algorithms were MLPNN, random forest, XGBoost. Occasionally lasso also yielded results as good as the best classifiers. We have AUC results at least one standard deviation above baseline for all datasets for at least one algorithm. In some instances more than two standard deviations above baseline. By individual train/test cycles for the dataset we find a greater spread. When looking at the mean AUC of an entire execution cycle we have smaller variances between then than when looking at individual train/test cycles.

Our interpretation results come from an averaged model. Since we average the results for the importance of every feature range, we should have more certain results since the individual variations have been averaged out. We obtained values for feature importance in line with Yachida et al.2019[75] on KEGG Orthology[76] (KO) terms. Though it should be noted that they only operated with a set of 16 KO terms and in combination with metabolomic and taxonomic information. Their dataset was also of similar size with what we have processed and trained on so far. In the models where taxonomic information, metabolomic information and KO terms were included the overall AUC was higher. Transferring this to our study it seems to imply that we could make use of our taxonomic classification information to improve overall AUC for the classifiers.

The AUC numbers for our classifiers are consistently lower than from Pasolli et al. 2016[15] and Ai et al. 2017[70]. Comparing AUC values between different studies is not straightforward as uneven distributions of cases and controls might lead to high baseline values. For the french dataset from Ai et al. 2017[70] and the dataset from Pasolli et al. 2016[15] we have a baseline AUCs of 0.624 and 0.603 respectively. Given the different feature sets ours are not directly comparable, however, it is clear that it is possible to get higher AUC numbers for classifiers trained on metagenomic data.

Looking at the results from Yachida et al. 2019[75], which make use of gene functions, can give us a better indication as their results are more comparable to ours. For classification of host stage 3 or 4 CRC they had an AUC of 0.69 with a baseline AUC of 0.61 using gene functions with random forest. We managed results with an AUC of 0.6288 with a baseline AUC of 0.5277 using random forest on the InterProScan IPR dataset, which was the best performing dataset for random forest on host CRC stage prediction. Our colorectal cancer cases are also from an earlier stage of colorectal cancer development, which is promising in regard to the potential future clinical value of our findings. In summary these are encouraging results and give promise to the eventual final results with the entire study dataset.

Two other studies that make for interesting comparisons are Wirbel et al. 2019[77] and Thomas et al. 2019[78]. Wirbel et al. 2019 performed CRC stage classification on a combined dataset from several different studies using EGGnog[79] and taxonomic information. EGGnog provided functional annotations and so make for an interesting comparison. They obtained AUC values from 0.78 to 0.89 on data with slightly variable ratios of cases to controls. With that they were able to identify significant enriched gene clusters in the metagenomic profiles of CRC cases.

Thomas et al. 2019[78] also make use of datasets from several different studies. In common with our data they also have information from HUMAnN and seek to find metagenomic profiles associated with colorectal cancer. One interesting result from Thomas et al. 2019 is examining a minimum microbial signature for CRC detection. They looked at prediction accuracy relative to the number of features included and found that small subsets can give nearly as good predictive performance as the entire feature set. This was at a species level, but is interesting in relation to gene functions nonetheless, especially when looking at the out-sized importance placed by LIME in a few features. The overall AUC values from the study were higher than from our results.

9.4 Strengths

A general strength of MAPLEgf that has been constructed is that it is highly configurable. Parameters can easily be changed through a set of config files associated with each step in the process. This configurability makes it easy to produce batches of results and compare results with the parameters selected. The tools used in the making of MAPLEgf are widely supported and well documented. This will make it easier for the CRCbiome team to integrate it into the entire pipeline for metagenomic analysis. Another technical strength is that it's freely available to clone on GitHub with documentation for how to it set up and start an analysis.

Data about CRC development stage and the gut microbiome is sensitive personal information. In the course of this project we have conducted machine learning without direct identification by any other means than sample id. As a part of the preprocessing pipeline we also filter out human genes found in the gut microbiome. Thereby not doing any machine learning on human genes.

The pipeline has a wide range of algorithms implemented as it stands. This gives the user a good ability to compare the results between the different methods that are applied. In the event that more or different algorithms should be included in the future, the structure is such that it should be very simple to add them in the future. This is because all the algorithms implement an interface where they have a set of commonly accessible methods and where the program configuration is managed. The source code is entirely hosted on GitHub, and free to clone and by following the provided instructions anyone should be able to start using it with fairly little effort.

The pipeline is to the best of our knowledge the first of it's kind to look at gene functions with a dataset of this size and using both InterProScan and HUMAnN. As described in the earlier works section prediction CRC or CRC development stage using metagenomic data is not a completely novel idea, but it has not been done in this way and at this scale for gene functions before.

Because of the cross-validation applied to the analysis and the large sample size we have good protection from overfitting on our dataset. This is something that has been borne out by our results and helped some of the classifiers implemented using solvers, such as ADAM[80], which can optimize in a way that can reduces overfitting.

The project also has implemented gene-function interpretation that has not been done on this kind of data of this size and with these term sets. Therefore having it in the project pipeline is a unique advantage of our project. And is an important part of gaining practically applicable information about the role of the gut microbiome in colorectal cancer development. This pertaining to the goal of this part of the analysis which is to find which specific gene functions are biomarkers for CRC development.

Our samples come from participants taking part in a screening program. As such our results are interesting in the way that any biomarkers could be used in colorectal screening. For instance Yachida et al.2019 [75] attempt to predict stage 3 or 4 colorectal cancer. This is a late stage of cancer where the gut microbiome might be more affected by the CRC development, but where the benefit of screening for cancer is gone as the patient is already seriously ill.

9.5 Weaknesses

Currently we do not have the complete results for the project. Ideally we would have access to all complete data from the approximate total 2500 samples. Due to delays stemming from the COVID-19 pandemic the sequencing has taken longer than intended. A larger amount of samples would reduce the risk of overfitting, likely reduce AUC variance between each individual training session and possibly increase the average AUC for each run.

We have a very feature rich dataset. We found better results for a preprocessed dataset where we reduced the amount of features by a coverage filter (data not shown). By way of preprocessing we tried to remove the noise and improve classification accuracy and reduce the variance. This is an optimization issue, and it's not clear that we found the ideal parameters. Part of the reason why it's difficult to determine is due to the variance in classifier results. In order to properly get a very accurate reading for any one setting quite a few executions have to be made.

In regard to the individual classifier results they have been mostly in line with our expectations. We tried two regression based classifiers, two "forest"-based classifiers and two neural network based classifiers. On metagenomic data it was expected that we would get the highest AUC results from the neural network based classifiers, but get very good results from forest based classifiers. Our assumptions were on the basis of such a performance profile on metagenomic data for Pasolli et al. 2016[15] and This has broadly seem to been the case, however we found that LSTM underperformed relative to expectations, rarely performing better than the baseline value. That does seem improbable, but when working on the LSTM we were unable to improve performance. We have not applied time series datasets which is a special ability of LSTMs to handle, but without that they should perform around the same level as MLPNNs. More on this in the further works section.

From the time the samples are taken to the final gene annotations there are quite a few steps. In some of them there is no guarantee that the results are entirely accurate. The entire analysis pipeline has been constructed with state of the art resources and tools, so this is inevitable. Still, it is worth mentioning as a possible weakness as in the future there might be better resources and tools for metagenomic analysis which could improve results.

Our samples do not all originate from the same follow up stage of

CRC screening. When doing colonoscopies or sigmoidoscopies the gut microbiome is impacted. We know that and we considered the additional samples included to be more valuable than keeping the dataset baseline only. Still, in theory it would be ideal to have nothing but samples taken at the same stage of screening. With all the samples finally it would make sense to split this up and look at baselines and other screening stages separately.

9.6 Further Work

In regard to further work on this project the first point should be running the entire dataset through the pipeline as soon as it is ready. The increase in samples will likely reduce variance in results, in part by increasing test sample sizes since each k-fold will be bigger. A larger training set should give better results, less prone to overfitting and which are less prone to making significantly different models with every train cycle. In essence a better and more stable model. Though looking at the results from our interpreters and the relatively small standard deviation for the feature importance for CRC stage classification it seems that our model is at present fairly stable.

When preprocessing the data it should be possible to only include a subset of GO terms that are functionally relevant to CRC. That might exclude unexpected associations, but should reduce the number of features and by this reduce the amount of noise. Given that we do not remove any important links or features this should improve accuracy and give clearer results when interpreting by avoiding overfitting. This could be an avenue worth exploring at a later point in time.

We could explore letting the output about which terms are interpreted as relevant back into the analysis. We could for instance look at the classifier performance if we only include features that occur with a given frequency or higher in the 100 most important features in a LIME interpretation. This makes overfitting on the dataset much more likely, but it could be worth exploring, even if just to confirm that features outside of this set have little bearing on classifier AUC.

Since we have samples taken at different times in the screening process that enables us to try time series data for the LSTM algorithm. This was not tried in this instance as we do not have enough sequenced samples for all times in the sequencing cycle finished. Therefore this will have to wait until we have more of the results in from sequencing.

The interpretation phase of the pipeline yields information about which features have a large bearing on the predictions made by the model. Analysis and interpretation of the terms by someone with a background in molecular microbiology would give an indication of whether the terms brought out by the model are likely to be associated with CRC development. This would help with ascertaining which of the models yield the best interpretation among those tried.

We could also stratify the dataset and explicitly compare different groups in it. Such as only looking at men or women, specific stages of CRC development or certain lifestyle factors. As brought up in discussion of Yachida et al. 2019 [75], we could also improve accuracy by combining taxonomy and gene function information in classifier training.

10 Conclusion

Given the basis of earlier work in applying machine learning methods on taxonomic distributions of gut microbiomes to predict host traits, it is clear that such an approach can yield insights. We have succeeded in constructing a pipeline for deep analysis of cancer causing genes in the gut microbiome. For all datasets we have been able to get AUC results that were well above their respective baseline values for at least some of the classifiers applied.

These results have been interpreted using LIME and we have been able to identify some preliminary candidates as gene function biomarkers for CRC development stage. The results have been promising. In the future this pipeline will be applied to the entire dataset when all the samples included in the study have been sequenced and processed. That will provide some final results for this project. With the current conditions that will be around 2Q2022. If as a result of this we find certain gene functions acting as biomarkers for CRC it can help create better screening protocols for colorectal cancer in the future.

11 Abbreviations

AUC: Area Under Curve
BCSN: Bowel Cancer Screening in Norway
CRC: Colorectal Cancer
DLT: Decision Learning Tree
FOBT: Faecal Occult Blood Test
GO: Gene Onthology
HFE: Hierarchical Feature Extraction
LSTM: Long Short-Term Memory
MAPLEgf: Multiple Algorithm Pipeline for machineLEarning on gene functions
MLP: Multi-layer Perceptron
NN: Neural Network
KEGG: Kyoto Encyclopedia of Genes and Genomes
RNN: Recursive Neural Network
SVM: Support Vector Machines
TSD: Tjeneste for Sensitive Data
UFS: Univariate Feature Selection

References

[1] Cancer Registry of Norway. (n.d.). The microbiome as a colorectal cancer screening biomarker. Retrieved April 28, 2020, from https://www.kreftregisteret.no/en/Research/Projects/microbiota-and-lifestyle-in-colorectal-cancer-screeing/

[2] Cristin. (2017). The microbiome as a colorectal cancer screening biomarker. Retrieved April 28, 2020, from https://app.cristin.no/projects/show.jsf?id=2053238

[3] de Lange, T., Randel, K. R., Schult, A. L., Knudsen, M. D., Kirkøen, B., Botteri, E., Berstad, P., Jørgensen, A., Ursin, G., … Hoff, G. (2017). Sigmoidoscopy and faecal occult blood test - a comparative screening trial. *Journal of the Norwegian Medical Association [Tidsskrift for Den norske legeforening]*, *137*(10), 727–730. https://doi.org/10.4045/tidsskr.16.1031

[4] Cancer Registry of Norway. (2019). *Cancer in Norway 2018 - Cancer incidence, mortality, survival and prevalence in Norway*. Oslo: Cancer Registry of Norway. Retrieved April 28, 2020, from https://www.kreftregisteret.no/globalassets/cancer-in-norway/2018/cin2018.pdf

[5] Bray, F., Ferlay, J., Soerjomataram, I., Siegel, R. L., Torre, L. A. & Jemal, A. (2018). Global cancer statistics 2018: Globocan estimates of incidence and mortality worldwide for 36 cancers in 185 countries.

CA: A Cancer Journal for Clinicians, *68*(6), 394–424. https://doi.org/10.3322/caac.21492

[6] Holme, Ø., Bretthauer, M., Fretheim, A., Odgaard-Jensen, J. & Hoff, G. (2013). Flexible sigmoidoscopy versus faecal occult blood testing for colorectal cancer screening in asymptomatic individuals. *Cochrane Database of Systematic Reviews*, *2013*(9). https://doi.org/10.1002/14651858.CD009259.pub2

[7] Pox, C., Schmiegel, W. & Classen, M. (2007). Current status of screening colonoscopy in Europe and in the United States. *Endoscopy*, *39*(2), 168–173. https://doi.org/10.1055/s-2007-966182

[8] Brenner, H. & Tao, S. (2013). Superior diagnostic performance of faecal immunochemical tests for haemoglobin in a head-to-head comparison with guaiac based faecal occult blood test among 2235 participants of screening colonoscopy. *European Journal of Cancer*, *49*(14), 3049–3054. https://doi.org/10.1016/j.ejca.2013.04.023

[9] Kato, J., Morikawa, T., Kuriyama, M., Yamaji, Y., Wada, R., Mitsushima, T. & Yamamoto, K. (2009). Combination of Sigmoidoscopy and a Fecal Immunochemical Test to Detect Proximal Colon Neoplasia. *Clinical Gastroenterology and Hepatology*, *7*(12), 1341–1346. https://doi.org/10.1016/j.cgh.2009.04.025

[10] Turnbaugh, P. J., Ley, R. E., Hamady, M., Fraser-Liggett, C. M., Knight, R. & Gordon, J. I. (2007). The Human Microbiome Project. *Nature*, *449*(7164), 804–810. https://doi.org/10.1038/nature06244

[11] Huttenhower, C., Gevers, D., Knight, R., Abubucker, S., Badger, J. H., Chinwalla, A. T., Creasy, H. H., Earl, A. M., Fitzgerald, M. G., . . . White, O. (2012). Structure, function and diversity of the healthy human microbiome. *Nature*, *486*(7402), 207–214. https://doi.org/10.1038/nature11234

[12] Fessler, J., Matson, V. & Gajewski, T. F. (2019). Exploring the emerging role of the microbiome in cancer immunotherapy. *Journal for ImmunoTherapy of Cancer*, *7*(1), 108. https://doi.org/10.1186/s40425-019-0574-4

[13] Qin, J., Li, R., Raes, J., Arumugam, M., Burgdorf, K. S., Manichanh, C., Nielsen, T., Pons, N., Levenez, F., . . . Wang, J. (2010). A human gut microbial gene catalogue established by metagenomic sequencing. *Nature*, *464*(7285), 59–65. https://doi.org/10.1038/nature08821

[14] Tamboli, C. P. (2004). Dysbiosis in inflammatory bowel disease. *Gut*, *53*(1), 1–4. https://doi.org/10.1136/gut.53.1.1

[15] Pasolli, E., Truong, D. T., Malik, F., Waldron, L. & Segata, N. (2016). Machine Learning Meta-analysis of Large Metagenomic Datasets: Tools and Biological Insights (J. A. Eisen, Ed.). *PLOS Computational Biology*, *12*(7), e1004977. https://doi.org/10.1371/journal.pcbi.1004977

[16] Rounge, T. B., Meisal, R., Nordby, J. I., Ambur, O. H., de Lange, T. & Hoff, G. (2018). Evaluating gut microbiota profiles from archived fecal samples. *BMC Gastroenterology*, *18*(1), 171. https://doi.org/10.1186/s12876-018-0896-6

[17] Brenner, H., Altenhofen, L., Stock, C. & Hoffmeister, M. (2014). Incidence of Colorectal Adenomas: Birth Cohort Analysis among 4.3 Million Participants of Screening Colonoscopy. *Cancer Epidemiology Biomarkers & Prevention*, *23*(9), 1920–1927. https://doi.org/10.1158/1055-9965.EPI-14-0367

[18] Murthy, S., Dubé, C., Rostom, A., Benchimol, E., Ducharme, R., Sutradhar, R., Rabeneck, L., Paszat, L. & Tinmouth, J. (2017). Risk of colorectal cancer after a negative colonoscopy in low-to-moderate risk individuals: impact of a 10-year colonoscopy. *Endoscopy*, *49*(12), 1229–1236. https://doi.org/10.1055/s-0043-117402

[19] Riesenfeld, C. S., Schloss, P. D. & Handelsman, J. (2004). Metagenomics: Genomic Analysis of Microbial Communities. *Annual Review of Genetics*, *38*(1), 525–552. https://doi.org/10.1146/annurev.genet.38.072902.091216

[20] Rappé, M. S. & Giovannoni, S. J. (2003). The Uncultured Microbial Majority. *Annual Review of Microbiology*, *57*(1), 369–394. https://doi.org/10.1146/annurev.micro.57.030502.090759

[21] Chen, K. & Pachter, L. (2005). Bioinformatics for Whole-Genome Shotgun Sequencing of Microbial Communities. *PLoS Computational Biology*, *1*(2), e24. https://doi.org/10.1371/journal.pcbi.0010024

[22] Cancer Registry of Norway. (-). Bowel Cancer Screening in Norway – a pilot study. Retrieved April 28, 2020, from https://www.kreftregisteret.no/en/screening/Screening-for-colorectal-cancer/

[23] Fiannaca, A., La Paglia, L., La Rosa, M., Lo Bosco, G., Renda, G., Rizzo, R., Gaglio, S. & Urso, A. (2018). Deep learning models for bacteria taxonomic classification of metagenomic data. *BMC Bioinformatics*, *19*(S7), 198. https://doi.org/10.1186/s12859-018-2182-6

[24] Zhou, Y.-H. & Gallins, P. (2019). A Review and Tutorial of Machine Learning Methods for Microbiome Host Trait Prediction. *Frontiers in Genetics*, *10*(JUN), 579. https://doi.org/10.3389/fgene.2019.00579

[25] Porter, T. M. & Hajibabaei, M. (2018). Scaling up: A guide to high-throughput genomic approaches for biodiversity analysis. *Molecular Ecology*, *27*(2), 313–338. https://doi.org/10.1111/mec.14478

[26] Noecker, C., McNally, C. P., Eng, A. & Borenstein, E. (2017). High-resolution characterization of the human microbiome. *Translational Research*, *179*, 7–23. https://doi.org/10.1016/j.trsl.2016.07.012

[27] Roumpeka, D. D., Wallace, R. J., Escalettes, F., Fotheringham, I. & Watson, M. (2017). A Review of Bioinformatics Tools for Bio-Prospecting from Metagenomic Sequence Data. *Frontiers in Genetics*, *8*(MAR), 23. https://doi.org/10.3389/fgene.2017.00023

[28] Ewing, B. & Green, P. (1998). Base-Calling of Automated Sequencer Traces Using Phred. II. Error Probabilities. *Genome Research*, *8*(3), 186–194. https://doi.org/10.1101/gr.8.3.186

[29] Cock, P. J. A., Fields, C. J., Goto, N., Heuer, M. L. & Rice, P. M. (2010). The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. *Nucleic Acids Research*, *38*(6), 1767–1771. https://doi.org/10.1093/nar/gkp1137

[30] Koster, J. & Rahmann, S. (2012). Snakemake–a scalable bioinformatics workflow engine. *Bioinformatics*, *28*(19), 2520–2522. https://doi.org/10.1093/bioinformatics/bts480

[31] Van Rossum, G. & Drake, F. L. (2009). *Python 3 reference manual*. CreateSpace.

[32] Ewels, P., Magnusson, M., Lundin, S. & Käller, M. (2016). MultiQC: summarize analysis results for multiple tools and samples in a single report. *Bioinformatics*, *32*(19), 3047–3048. https://doi.org/10.1093/bioinformatics/btw354

[33] Bolger, A. M., Lohse, M. & Usadel, B. (2014). Trimmomatic: a flexible trimmer for Illumina sequence data. *Bioinformatics*, *30*(15), 2114–2120. https://doi.org/10.1093/bioinformatics/btu170

[34] Menzel, P., Ng, K. L. & Krogh, A. (2016). Fast and sensitive taxonomic classification for metagenomics with Kaiju. *Nature Communications*, *7*(1), 1–9. https://doi.org/10.1038/ncomms11257

[35] Franzosa, E. A., McIver, L. J., Rahnavard, G., Thompson, L. R., Schirmer, M., Weingart, G., Lipson, K. S., Knight, R., Caporaso, J. G., . . . Huttenhower, C. (2018). Species-level functional profiling of metagenomes and metatranscriptomes. *Nature Methods*, *15*(11), 962–968. https://doi.org/10.1038/s41592-018-0176-y

[36] Beghini, F., McIver, L. J., Blanco-Mıguez, A., Dubois, L., Asnicar, F., Maharjan, S., Mailyan, A., Thomas, A. M., Manghi, P., . . . Segata, N. (2020). Integrating taxonomic, functional, and strain-level profiling of diverse microbial communities with biobakery 3. *bioRxiv*. https://doi.org/10.1101/2020.11.19.388223

[37] Nurk, S., Meleshko, D., Korobeynikov, A. & Pevzner, P. A. (2017). metaSPAdes: a new versatile metagenomic assembler. *Genome Research*, *27*(5), 824–834. https://doi.org/10.1101/gr.213959.116

[38] Hyatt, D., Chen, G.-L., LoCascio, P. F., Land, M. L., Larimer, F. W. & Hauser, L. J. (2010). Prodigal: prokaryotic gene recognition and

translation initiation site identification. *BMC Bioinformatics*, *11*(1), 119. https://doi.org/10.1186/1471-2105-11-119

[39] Quevillon, E., Silventoinen, V., Pillai, S., Harte, N., Mulder, N., Apweiler, R. & Lopez, R. (2005). InterProScan: protein domains identifier. *Nucleic Acids Research*, *33*(suppl_2), W116–W120. https://doi.org/10.1093/nar/gki442

[40] Hunter, S., Apweiler, R., Attwood, T. K., Bairoch, A., Bateman, A., Binns, D., Bork, P., Das, U., Daugherty, L., ... Yeats, C. (2009). InterPro: the integrative protein signature database. *Nucleic Acids Research*, *37*(Database), D211–D215. https://doi.org/10.1093/nar/gkn785

[41] Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M., Davis, A. P., Dolinski, K., Dwight, S. S., ... Sherlock, G. (2000). Gene Ontology: tool for the unification of biology. *Nature Genetics*, *25*(1), 25–29. https://doi.org/10.1038/75556

[42] Gene Ontology Consortium. (2004). The Gene Ontology (GO) database and informatics resource. *Nucleic Acids Research*, *32*(suppl_1), D258–D261. https://doi.org/10.1093/nar/gkh036

[43] Kanehisa, M. (2000). KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Research*, *28*(1), 27–30. https://doi.org/10.1093/nar/28.1.27

[44] Weiss, S., Xu, Z. Z., Peddada, S., Amir, A., Bittinger, K., Gonzalez, A., Lozupone, C., Zaneveld, J. R., Vázquez-Baeza, Y., ... Knight, R. (2017). Normalization and microbial differential abundance strategies depend upon data characteristics. *Microbiome*, *5*(1), 27. https://doi.org/10.1186/s40168-017-0237-y

[45] Randolph, T. W., Zhao, S., Copeland, W., Hullar, M. & Shojaie, A. (2018). KERNEL-PENALIZED REGRESSION FOR ANALYSIS OF MICROBIOME DATA (2018/03/09). *The annals of applied statistics*, *12*(1), 540–566. https://doi.org/10.1214/17-AOAS1102

[46] Pedregosa, F., Michel, V., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Vanderplas, J., Cournapeau, D., Pedregosa, F., ... Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830. Retrieved April 28, 2020, from http://jmlr.org/papers/v12/pedregosa11a.html

[47] Russel, S. & Norvig, P. (2013). *Artificial intelligence: a modern approach*. Pearson Education Limited.

[48] Schmitt, S., Tsai, P., Bell, J., Fromont, J., Ilan, M., Lindquist, N., Perez, T., Rodrigo, A., Schupp, P. J., ... Taylor, M. W. (2012). Assessing the complex sponge microbiota: core, variable and species-specific bacterial communities in marine sponges. *The ISME Journal*, *6*(3), 564–576. https://doi.org/10.1038/ismej.2011.116

[49] Ditzler, G., Morrison, J. C., Lan, Y. & Rosen, G. L. (2015). Fizzy: feature subset selection for metagenomics. *BMC Bioinformatics*, *16*(1), 358. https://doi.org/10.1186/s12859-015-0793-8

[50] Oudah, M. & Henschel, A. (2018). Taxonomy-aware feature engineering for microbiome classification. *BMC Bioinformatics*, *19*(1), 227. https://doi.org/10.1186/s12859-018-2205-3

[51] Tibshirani, R. (1996). Regression Shrinkage and Selection Via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, *58*(1), 267–288. https://doi.org/10.1111/j.2517-6161.1996.tb02080.x

[52] Crookston, N. L. & Finley, A. O. (2008). yaImpute : An R Package for k NN Imputation. *Journal of Statistical Software*, *23*(10). https://doi.org/10.18637/jss.v023.i10

[53] van Buuren, S. & Groothuis-Oudshoorn, K. (2011). mice : Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, *45*(3), 1–68. https://doi.org/10.18637/jss.v045.i03

[54] Cortes, C. & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297. https://doi.org/10.1007/BF00994018

[55] Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

[56] Chen, T. & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. https://doi.org/10.1145/2939672.2939785

[57] Haykin, S. S. (2009). *Neural Networks and Learning Machines (3rd Edition)*. Pearson Prentice Hall.

[58] Hochreiter, S. & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[59] Boser, B. E., Guyon, I. M. & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. *Proceedings of the fifth annual workshop on Computational learning theory - COLT '92*, 144–152. https://doi.org/10.1145/130385.130401

[60] Tin Kam Ho. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*(8), 832–844. https://doi.org/10.1109/34.709601

[61] Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123–140. https://doi.org/10.1007/bf00058655

[62] Saabas, A. (2015). *Random forest interpretation with scikit-learn*. Retrieved April 9, 2020, from blog.datadive.net/random-forest-interpretation-with-scikit-learn/

[63] Ditzler, G., Polikar, R. & Rosen, G. (2015). Multi-Layer and Recursive Neural Networks for Metagenomic Classification. *IEEE Transactions on NanoBioscience*, *14*(6), 608–616. https://doi.org/10.1109/TNB.2015.2461219

[64] Reiman, D., Metwally, A. & Dai, Y. (2017). Using convolutional neural networks to explore the microbiome. *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 4269–4272. https://doi.org/10.1109/EMBC.2017.8037799

[65] Keras. (2020). *Keras: The python deep learning library*. Retrieved April 5, 2020, from https://keras.io/

[66] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M. et al. (2016). Tensorflow: A system for large-scale machine learning. *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 265–283. Retrieved April 29, 2020, from https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf

[67] Ribeiro, M. T., Singh, S. & Guestrin, C. (2016). Model-Agnostic Interpretability of Machine Learning. http://arxiv.org/abs/1606.05386

[68] TeamHG-Memex. (n.d.). Teamhg-memex/eli5. https://github.com/TeamHG-Memex/eli5

[69] Zeller, G., Tap, J., Voigt, A. Y., Sunagawa, S., Kultima, J. R., Costea, P. I., Amiot, A., Böhm, J., Brunetti, F., ... Bork, P. (2014). Potential of fecal microbiota for early-stage detection of colorectal cancer. *Molecular Systems Biology*, *10*(11), 766. https://doi.org/10.15252/msb.20145645

[70] Ai, L., Tian, H., Chen, Z., Chen, H., Xu, J. & Fang, J.-Y. (2017). Systematic evaluation of supervised classifiers for fecal microbiota-based prediction of colorectal cancer. *Oncotarget*, *8*(6), 9546–9556. https://doi.org/10.18632/oncotarget.14488

[71] Caporaso, J. G., Lauber, C. L., Costello, E. K., Berg-Lyons, D., Gonzalez, A., Stombaugh, J., Knights, D., Gajer, P., Ravel, J., ... Knight, R. (2011). Moving pictures of the human microbiome. *Genome Biology*, *12*(5), R50. https://doi.org/10.1186/gb-2011-12-5-r50

[72] R Core Team. (2019). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. Vienna, Austria. https://www.R-project.org/

[73] Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., ... Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

[74] Olsvik, H. H. (2021). Maplegf - multi-algorithm pipeline for machine learning on gene functions. https://github.com/henrikolsvik/maplegf

[75] Yachida, S., Mizutani, S., Shiroma, H., Shiba, S., Nakajima, T., Sakamoto, T., Watanabe, H., Masuda, K., Nishimoto, Y., ... Yamada, T. (2019). Metagenomic and metabolomic analyses reveal distinct stage-specific phenotypes of the gut microbiota in colorectal cancer. *Nature Medicine*, *25*(6), 968–976. https://doi.org/10.1038/s41591-019-0458-7

[76] Kanehisa, M., Sato, Y., Kawashima, M., Furumichi, M. & Tanabe, M. (2016). KEGG as a reference resource for gene and protein annotation. *Nucleic Acids Research*, *44*(D1), D457–D462. https://doi.org/10.1093/nar/gkv1070

[77] Wirbel, J., Pyl, P. T., Kartal, E., Zych, K., Kashani, A., Milanese, A., Fleck, J. S., Voigt, A. Y., Palleja, A., ... Zeller, G. (2019). Meta-analysis of fecal metagenomes reveals global microbial signatures that are specific for colorectal cancer. *Nature Medicine*, *25*(4), 679–689. https://doi.org/10.1038/s41591-019-0406-6

[78] Thomas, A. M., Manghi, P., Asnicar, F., Pasolli, E., Armanini, F., Zolfo, M., Beghini, F., Manara, S., Karcher, N., ... Segata, N. (2019). Metagenomic analysis of colorectal cancer datasets identifies cross-cohort microbial diagnostic signatures and a link with choline degradation. *Nature Medicine*, *25*(4), 667–678. https://doi.org/10.1038/s41591-019-0405-7

[79] Huerta-Cepas, J., Szklarczyk, D., Heller, D., Hernández-Plaza, A., Forslund, S. K., Cook, H., Mende, D. R., Letunic, I., Rattei, T., ... Bork, P. (2019). eggNOG 5.0: a hierarchical, functionally and phylogenetically annotated orthology resource based on 5090 organisms and 2502 viruses. *Nucleic Acids Research*, *47*(D1), D309–D314. https://doi.org/10.1093/nar/gky1085

[80] Kingma, D. P. & Ba, J. (2014). Adam: A Method for Stochastic Optimization. http://arxiv.org/abs/1412.6980