RANDOM FORESTS FOR PLASMID DETECTION

An exercise in Model Building and Evaluation

Abstract

Plasmids are bacterial genetic elements that are replicated and transferred independently from the chromosome. Because of their independent mechanisms of replication and transfer, the study of plasmids is of special interest in epidemiology. The introduction of short read sequencers has brought an abundance of data to microbial genomics with great potential to increase knowledge of microbial biology and inform epidemiological decision making. With this increase in data availability comes the need for computational methods to extract meaningful information from that data. Machine learning tools have been developed to distinguish plasmids from chromosomes in short read draft genome assemblies. RFPlasmid is such a tool that uses random forests to classify bacterial contigs. To explore potential improvements of RFPlasmid, a machine learning pipeline was developed in Scikit-Learn. The machine learning pipeline addresses the issue of imbalanced datasets, which is a common problem as generally more chromosomes are sequenced than plasmids. It also probed several methods of feature selection to aid in separating signal from noise in a wide and sparse dataset and thereby improve classifications. Imbalance remains a difficult challenge which requires a multi-faceted approach to improve models of species for which few plasmid sequences are publicly available. Feature selection did not improve explainability or reduce model complexity. Critical issues came to light showing the combination of fully grown random forests using kmers is problematic when modeling plasmids. The insights from this project can be used as a starting point to develop better machine learning algorithms for plasmid detection. However, other computational methods, including graph, mapping and clustering based approaches may be more promising.

Gilliquet, Ethel

Bioinformatics Project Report

Under supervision of Aldert Zomer & Linda van der Graaf - van Bloois

Utrecht University

2021-12-21

Table of Contents

1. Introduction: Plasmid Detection with Models

1.1. Antimicrobial Resistance & Plasmid Mobility - The Problem

Bacterial genomes are made up of two types of DNA sequences: chromosomes and plasmids [Fig1]. Chromosome DNA gets passed on to daughter bacteria when a bacterium divides, while plasmid DNA can be shared with other bacteria, not just through inheritance. This means that Plasmid DNA can spread faster, comparable to a viral video spreading through YouTube, when a video is reposted by many channels. Chromosomes spread more like when a Youtuber starts a new channel with copies of all their old videos, which is much less frequent. When bacteria cause disease in humans, animals or plants, antibiotics are often used to fight them. Unfortunately, not all pathogenic bacteria are killed, and some can have genes or mutations that makes them immune to antibiotics – this is called antimicrobial resistance (AMR). This is a big problem, because if bacteria spread that are resistant to antibiotics, there is no way to fight them. AMR genes that make bacteria resistant are often found on plasmids and therefore can spread faster through bacterial populations. Thus, it is important to detect bacteria that harbor AMR genes on plasmids as soon as possible to prevent wide-spread infection.

To identify the genetic content of bacteria, the plasmids and chromosomes of bacteria are fragmented into small pieces of DNA and these are sequenced. The DNA sequences are read by machines that output the order of the DNA bases. The fragmented sequences are put back together like a puzzle with a computer because there are so many sequences and possible combinations, it is not possible to do by hand. The finished puzzles are called contigs. Ideally, there is one contig for the chromosome and one contig per plasmid which makes it possible to tell if the AMR gene is on a plasmid or chromosome. This is never the case because identical DNA sequences occur on both chromosomes and plasmids and repetitive DNA sequences. This causes the reassembly of the reads to be fragmented, making it unclear whether a contig originates from a chromosome or plasmid [Fig1]. Because of this, tools have been developed to classify contigs as chromosomal or plasmid. These programs take many different features of the sequences into account and species-specific signatures to detect plasmids in different types of bacteria. Besides being relevant for detecting AMR genes on plasmids in epidemiology, tools for bacterial DNA classification can also be applied in any context where bacterial DNA is available such as in fundamental microbial genomics research, healthcare, biotechnology, and agriculture.

RFPlasmid is such a tool, available on github[1], with the main purpose of classifying bacterial contig data into two classes: originating from plasmids or chromosomes. The implementation of RFPlasmid has three main parts: feature extraction, classification, and training. Feature extraction identifies a variety of features from the contigs based on the identity and prevalence of marker genes on contigs using Blast[2] and CheckM[3] as well as sequence specific features, namely kmer fractions and length of the contig sequence. The classification part classifies contigs with one of the 18 pre-trained random forest models currently available. The training part trains a new model with a user input contig dataset.

The training and classification of RFPlasmid are currently implemented in the R randomForest package[4], whereas the remainder of the pipeline is in Python. The aim of this project is for the entire pipeline to be written in Python. Replacing the R machine learning with Python (using Scikit-Learn[5]) has the potential of further improving classification and performance by allowing for different hyperparameters and options during model training. Besides potentially improving classifications, removing the dependency on R will simplify installation for users. The random forest implementation of Scikit-Learn is also much faster, allowing for more training in shorter time and faster classifications.

1.2. General Modeling Approach

In machine learning, statistics, and science in general, the goal is to obtain a model based on data that can inform us about the real world. A model is made up of two core components; an algorithm and the data used to train the algorithm: Data + Algorithm = Model [Fig3]. The goal of this project was to create a new computational model for detecting plasmid contigs in draft assemblies [Fig2]. The model should contain logic to classify contigs as originating from plasmids or chromosomes and make correct classifications when used on real-world contig data.

1.2.1. What is a good dataset?

A dataset is good when it is a unbiased, representative sample of the population and there is high confidence in its correctness. Statistically speaking, a dataset needs to meet certain criteria: not too noisy (inconsistent results for similar inputs), no missing values, balanced (meaning it has equal numbers of chromosomal and plasmid contigs), and meets the assumptions of the algorithm (normal distribution and within a certain range without many outliers). Most datasets are not ideal and need to be preprocessed: missing data points need to be imputed, columns need to be transformed to be normally distributed, samples need to be balanced by resampling, and bad features need to be removed. What preprocessing steps are required generally depend on the assumptions of the algorithm.

1.2.2. What is a good algorithm?

A good algorithm is one that fits the data, purpose and produces generalizable models. The implementation of an algorithm needs to be transparent, developer-friendly with good documentation and reliable.

1.1.1. What is a good model?

A good model is generalizable. In other words, a good model can make true classifications about reality. It is not limited to classifying training data but can also be applied to new data. A model that is not better than chance and does not make classifications based on relevant features of a dataset is useless.

A good model is explainable. It is not just a black box but can give insight into why it makes the classifications it makes. There must be some confidence that relevant features of the dataset determine a classification and not noise or a bug in the code. Other ways in which an explainable model can be useful is to help in decision-making or to give new insights about the underlying relationships in the data.

A good model is usable. It must be fit for purpose and easy to use. For example, a model that cannot be installed on a normal computer because it is too large is not a good model.

In summary, a good model is a meaningful simplification. It is not too simple or complex. For example, a complex model will turn out to not be generalizable because overfits to the training dataset by memorizing details of a dataset that are not in other datasets. When a model memorizes its training dataset, it will perform very well on its training dataset, and the classifications on this dataset will be correct, which is misleading because the performance will be much lower when used with any other dataset. Memorizing the training data also makes the model so complex that it becomes difficult, even for the expert, to explain how the classifications of the model are actually made. An overly complex model, that is not generalizable or explainable, will not classify real world data correctly. However, an overly simple model will perform poorly on both its training dataset and other datasets. It might be very explainable because of its simplicity, but because it does not generalize, it is nonetheless useless for classifying real world data. [Fig 4 Bottom]

To generate a good model, it is important to have a good algorithm and a good dataset that work well together. If an algorithm is amazing but the data is too bad, it cannot produce a good model, and the acronym GIGO applies: "Garbage in, garbage out". The reverse is true as well, a great dataset modeled with the wrong algorithm cannot result in a good model. It is important to realize that a model can look good on paper, but not be generalizable or useful [Box1]. Therefore, it is essential to evaluate a model carefully and consider several dimensions of "model goodness". Also, it is necessary to consider when a model would be too bad to use. How bad does a model need to be that it is not fit for purpose? Can a flaw be compensated, or is it a showstopper? For an overview of requirements for Modeling, see [Table1].

Box 1 ***3 ways in which a 99% accurate model can be useless***

1. A model's accuracy should be representative of most relevant combinations of user input data. For example, a model that detects bacterial plasmid contigs but was trained mostly on enterobacterial contigs, will not be that useful to a user who wants to classify *Vibrio* contigs.
2. The training data is very imbalanced, it consists of 99% chromosome contigs. Then an accuracy of 99% can be achieved even by a model that always classifies contigs to be chromosomal contigs.
3. The model is truly very good, but even then, it can be dangerous, as a model that is marketed as being "99% accurate" can make the user overconfident and overlook signs that the model might be wrong in their specific use case, which might be the 1% where it does not give correct classifications. A model that is "only 66%" accurate might then actually be better because it aids the user to think for themselves, instead of relying on the model.

1.3. Outline of report

Section 2 of the report outlines the three core elements of the machine learning pipeline: the dataset, algorithm and performance metrics to evaluate the models. Section 3 outlines the outcome of the model evaluation in the four key areas: generalizability, explainability, usability and complexity. Section 4 discusses the findings in Section 3 and, whether the machine learning pipeline and the models are 'good enough' to replace the existing machine learning pipeline in R.

2. Method

This section outlines the three core elements of the machine learning pipeline: the dataset, algorithm and performance metrics to evaluate the models.

2.1. The Datasets

The datasets of RFPlasmid consist of 17 taxon-specific datasets and one taxon-agnostic dataset, which is derived from the 17 taxon-specific datasets. The datasets contain contigs assembled from simulated reads. Plasmid and chromosome sequences were fragmented into random 400 bp fragments with 50x coverage and reassembled. The sequences were obtained from publically available complete bacterial genome reference sequences, except for Enterobacteriaceae and Campylobacter, which are inhouse manually labeled datasets, and Listeria, a separate publicly available dataset. For more detail on the generation of the simulated contigs and feature extraction, see the RFPlasmid prepub[6]. The taxons represented in the datasets are on a bacterial family or genus level, except for the taxon-agnostic dataset (*Bacteria*), which is on the domain level. The starting point for the machine learning pipeline is a table per dataset (1 csv file per taxon), where the contigs (or samples) form the rows of the dataset, one column representing the known label of the contig (chromosomal or plasmid) and 1031 feature columns [Table3].

Dataset size and imbalance

The taxon-specific datasets range in size from 634 contigs (Cyanothece) to 28,544 contigs (Enterobacteriaceae), and the taxon-agnostic dataset (Bacteria) contains 222,723 contigs (the combined contigs of the taxon-agnostic datasets) [Table2]. The imbalance of the datasets ranges from 57% (Rhizobium) to 98% (Clostridium and Pseudomonas) [Table2]. Over half of the datasets have an imbalance of 90% or higher. All datasets have a chromosomal contig majority, except for Rhizobium (57% plasmid contigs) and Borrelia (93% plasmid contigs) [Fig5]. In terms of contig length in basepairs, the imbalance is even more pronounced in favor of the chromosomes, ranging from 67% to over 99% imbalance [Fig5]. The datasets contain many duplicate contigs (meaning contigs are indistinguishable because their features are identical). The Enterococcus, Enterobacteria, Listeria and Campylobacter datasets have the highest fraction of plasmid contig duplicate groups [Table3 middle].

The contig features

There are a total of 1031 features for each dataset. The kmers make the dataset sparse, wide, and noisy. Kmers are noisy because each kmer fraction carries little information per contig, they have a low signal-to-noise ratio. Kmers are sparse because there are many zeroes and wide because they collectively make up 1024 columns of the dataset. Because the dataset is wide, it is to be expected that some features will be used because of chance, and not because they contain meaningful information to distinguish between the origin of contigs. The non-kmer features are meaninglessly precise because they have many decimals. The number of decimals does not matter for most algorithms, but for the random forest, it matters how precise the features are because it gives more opportunities for splitting the data and overfitting (see below in 2.2). Many features are highly correlated, as for example the k-mers 'TTTTT' and 'AAAAA' are identical, and features such as 'kmer number' and 'contig length' or 'SCM' and 'SCM genes' are correlated as one is computed based on the other [Table3 bottom].

The contig labels

The labels are based on the reference genome annotations. Contigs that were classified incorrectly by RFPlasmid were mapped back to the references and the label was manually reassigned if it improved classifications. Contig relabeling was length dependent. Long plasmid contigs that were incorrectly classified as chromosome contigs were often relabeled to be chromosome contigs by relabeling the entire replicon. Short contigs that were wrongly classified were not relabeled. It is suspected that relabeling artificially increased the imbalance and introduced other errors, which could have been prevented by allowing for a third class. The labeling was not part of this project and the assigned labels were taken as ground truth, meaning, they were accepted as is.

2.2. Choice of algorithm and implementation

The choice of algorithm and implementation was guided by the existing pipeline of RFPlasmid which makes use of the R package randomForest. SKLearn's RandomForestClassifier[7] was chosen because it is the only widely used Python module to implement a random forest. A benefit of the Python implementation compared to that of R, is that it is optimized for speed, which should make the new pipeline faster and more usable. Both the R and Python implementations are based on the random forest algorithm description of the method in the paper by Leo Breiman[8] and the FORTRAN reference implementation.

2.2.1. Random Forest – A Sketch

A random forest is made up of several decision trees. A decision tree is a directed acyclical graph that looks like a flowchart. At each internal node a decision is made based on a feature, each branch represents the outcome of the decision, and each terminal node (leaf) represents a class label or final decision (here chromosome or plasmid).[9] [Fig6]

A random forest is made up of decision trees that have been trained on random subsets of the samples (here contigs) of a training dataset of which the true label is known. At each split of each tree, a random subset of the features is sampled from which the best feature to split the sample is greedily selected (greedy means, it deterministically chooses the best among the options it has without foresight or considering that another option might in the long run yield better results). By only selecting smaller random sets, the issues normally associated with greedy selection (non-local optima, overfitting) are circumvented (but not when there are many duplicates in samples and features). If there are duplicates in the dataset however, the assumptions of the random forest are not met[8]. The trees are grown until the leaves are pure (they contain only samples of one class)[8]. A random forest quickly becomes very complex. See for an example of a random forest of 100 trees, each with a depth 20 in [Fig7].

After training, the random forest is used for classification. Each contig is run through every tree and at every node, depending on the value of that feature, the contig is moved left or right down the branches until it ends in a terminal node. The tree then votes according to the proportion of classes in the terminal node, so if 3 out of 10 contigs are plasmid contigs in that terminal node, the tree votes 30%. The votes from all the trees in the forest are averaged and if the average of the votes is 50% or above, the contig is determined to be a plasmid contig, else it is a chromosome contig. The distance of the terminal node from the root node does not matter for the votes, which means that a terminal leaf with just 1 chromosomal contig (representing an outlier) is weighted the same as a terminal node containing 1000 chromosomal contigs (representing meaningful biological information). Not only are all terminal nodes equally weighted, each tree in the forest is equally weighted regardless of quality as well. This means that big, complex trees, which make very detailed distinctions, have the same vote as simple trees, which represent more general aspects of the dataset. The simpler trees should have more weight in the classification as they represent more meaningful biological information that distinguishes chromosomal and plasmid contigs. Random forests are not robust to small changes in datasets because they overfit easily to outliers in the dataset, making it classify incorrectly on real-world data. This is a severe limitation for real-world use.

The Random Forest algorithm has a built-in performance evaluation, the Out-of-Bag classification (OOB). After a tree is built with a subset of the samples (in-bag), the tree then classifies the samples which were not part of that tree's training set (out-of-bag). At the end, all votes for all out-of-bag samples over all the trees are averaged per sample, yielding the OOB classification. Because the classifications are based on the samples that a tree had not seen, it is comparable to cross-validations with a training set or a test set. Duplicates in the data however allow the trees to "cheat" because the in-bag and out-of-bag contain identical contigs, this is a type of leakage from the training to the test data. This means that the OOB classification scores of models trained on duplicate data will be increased artificially and look better than they are. On the other hand, if there are duplicates that have different labels, that will decrease the OOB classification scores, as one of the duplicates will always be classified incorrectly.

2.2.2. How to set hyperparameters

In machine learning, there are many options for hyperparameter settings – unless one wants to use the default, which is probably not going to work well for all datasets. An example of a hyperparameter is the number of trees in the forest. Common approaches are: trial and error, literature study (online and in published literature) to learn about random forests hyperprameters, and automated hyperparameter optimization. There are many options for hyperparameter optimization, some are built into Scikit-Learn, others are separate packages that enable fully automated machine learning (AutoML). Hyperparameter optimization turned out to just shift the question to 'how to tune the hyperparameter optimization' instead of answering the problem. AutoML always had over 99% accuracies, which seemed to be too good to be true and overfit to the datasets.

Ultimately the approach was taken to first learn how to navigate Scikit-Learn by trial-and-error and then make informed decisions based on literature; reading online blog posts, searching QA sites [towardsdatascience, machinelearningmastery, stackexchange, stackoverflow] and published machine learning papers[8,10,11]. The settings were chosen to resemble the settings of the RFPlasmid models to make a fair comparison. RFPlasmid grows 5000 trees per forest which costly in terms of time and space. Some time was spent to look for a tree number that was "good enough" and 500 trees were deemed sufficient, except for the most imbalanced datasets, which required 1000 or 2000 trees. Because the trees are fully grown in RFPlasmid, this was done in the SKLearn implentattion as well.

2.3. Dataset Challenge 1: Approach to fix unbalanced datasets

The imbalance of the datasets ranges from 57% (Rhizobium) to 98% (Clostridium and Pseudomonas) [Table2, Fig5], with over half of the datasets having an imbalance of 90% or higher. This is problematic, because a model will always be fit to the majority class more than the minority class. In most of the datasets (except for Rhizobium and Borrelia), the plasmid contigs are underrepresented. The models trained on these datasets will therefore be better at detecting chromosomes. To address this issue, the package Imbalanced-Learn was used. The package Imbalanced-Learn is specifically designed as an add-on to SKLearn to address the problem of machine learning with imbalanced datasets[12] by applying samplers. In short, each tree is trained using the same number of chromosomal and plasmid contigs, in order to be less biased toward the chromosomal contigs. For a good overview of the effect of different types of models, settings and samplers on imbalanced datasets[13,14].

2.4. Dataset Challenge 2: How to select good features

Per contig, there are 1031 features meaning there is a high chance of overfitting to noise (as stated earlier when discussing the dataset in Section 2.1). Two types of feature selection were used to address this issue: feature selection based on feature importances and recursive feature elimination with cross-validation (rfecv)[18]. SKLearn has a function SelectFromModel[17], which selects features better than average. What is "better than average" depends on feature importances built into the random forest. Rfecv is time-consuming for big datasets, as potentially hundreds of random forests are built and evaluated. Feature selection was performed on just 75% of the training set to prevent overfitting[11].

2.5. Dataset Challenge 3: How to remove noise from the features

Random forests have an inbuilt metric to measure feature importance. While researching random forests, it became clear that feature importances in random forests can be misleading.[15] [16] If the built-in feature importances are not to be taken at face value, this would imply a fundamental problem, namely that there is something wrong with how a random forest values and selects features to make splits. Here is an example: Two contigs, one with a kmerX fraction of 0.000010 and another contig with a kmerX fraction of 0.000011. These kmer fractions are not significantly different and not representative of the underlying distinctions between contigs, but the random forest might use the small difference to make a classification decision, based only on that tiny difference. This is because at each node, all else being equal, a feature with more unique values (high cardinality) is more likely to be selected because there are more opportunities to make a cut-off. This is training on noise. Good features should be selected because of meaningful information and not only because a random forest is biased towards noisy features that have many meaningless decimals that make the contigs appear more different than they actually are. In other words, fix the forest, not the importances. To remove the noise, the features could be rounded. Unfortunately, rounding introduces many more duplicate contigs than are already in the dataset, so this issue remains unresolved.

2.6. Metrics for Model Evaluation

2.6.1. Performance metrics

Once a model is trained, it is important to assess how good it is at classifying. In other words, how correct are the model's classifications? In a binary classification problem with a Positive and Negative Class, there are 4 possible outcomes for each instance: True Positive, False Negative, False Positive and True Negative, collectively called a confusion matrix. Here the plasmid contigs are the positive class and chromosomal contigs are the negative class. For each contig there are two possibilities; correctly classified or incorrectly classified. This makes 4 possible outcomes: An actual plasmid contig can be correctly classified as a plasmid contig, or incorrectly classified as a chromosomal contig, an actual chromosomal contig can be correctly classified as a chromosomal contig or incorrectly classified as a plasmid contig [Fig9]. A good model will have few incorrect classifications and many correct classifications.

Positive Class	Plasmid contigs
Negative Class	Chromosome contigs

True Positive	Plasmid contig, correctly classified as a plasmid contig
False Negative	Plasmid contig, incorrectly classified as a chromosomal contig
False Positive	Chromosomal contig, incorrectly classified as a plasmid contig
True Negative	Chromosomal contig, correctly classified as a chromosomal contig

2.6.2. Summary Performance Metrics

Accuracy is probably the most widely used metric to assess machine learning models. It is the sum of all correctly classified samples divided by all the samples. It works very well for balanced datasets, where the proportion between the classes is close to equal. Unfortunately, Accuracy overstates the model's performance when the dataset is very imbalanced because the classifications of the majority are often much better than that of the minority class. For example, in a fictional situation of the majority class making up 95% of the samples, an overall Accuracy of 95% is possible, while the minority class accuracy may be as low as0%. Given the imbalanced nature of these datasets [Table2, Fig5], reporting Accuracy makes an overly optimistic impression, reflecting that chromosomal contigs are classified well but this might hide a poor performance for plasmid contigs. Accuracy alone is not sufficient. Ideally one would want a metric that is more balanced also with imbalanced datasets, where a higher number means that both classes are classified well and not just the majority class.

Various established metrics are reported for a holistic understanding of the model performance. Of special interest for plasmid classification are Recall (also known as True Positive Rate, Sensitivity or Power), Precision (also known as Positive Predictive Value) and the Threat Score as they pertain to the plasmids [Fig9] ('Confusion Matrix' wikipedia for an overview of all the possible metrics[19]). The Threat Score is the same as Accuracy, except it disregards the True Negatives (Correctly predicted chromosomal contigs) and is therefore a good summary metric for the minority class, here, the plasmid contigs. Another metric, which considers all information from the confusion matrix, the Matthew's Correlation Coefficient (MCC), is also reported[20]. It is a summary statistic that incorporates the ratios between all four corners of the confusion matrix and is less affected by imbalance.

Recall	True Plasmid Rate	Correctly predicted plasmid contigs / All true plasmid contigs
Precision	Plasmid Predictive Value	Correctly predicted plasmid contigs / All classified plasmid contigs

Besides reporting metrics in terms of contig count, RFPlasmid also reports metrics in terms of contig length in basepairs. This is not advisable, as it makes the imbalance even worse and overstates the chromosomal contig accuracy even more, leading to misleadingly high metrics, which can hide poor performance in what is the important unit here: the plasmid, which happens to on average be shorter and produce shorter contigs and is in the minority in most datasets [Fig5].

2.6.3. Complexity Metrics

To assess the models' complexity, several metrics were tracked before and after training. Reported here are the Model size in MB, the forest size in number of trees and the average tree size in terms of number of nodes and tree depth. Models that are large and complex indicate overfitting. Large models are not a useful simplification, rather they are an inefficient memorization of the dataset, especially if the models are larger than their training dataset, which they often were, see 3.3 below. Large trees in terms of number of nodes and depth indicate overfitting as well, as the trees make many fine-grained distinctions between the contigs that will likely not be present in other datasets. Other than being overfit, large models require computers with a lot of memory to use and will take longer to load before being available to use.

3. Results

The following section discusses model performance, generalizability, complexity, and explainability.

3.1. Model Performance

Summary Metrics

The Summary Metrics [Fig10Left] show that the Accuracy of all the models is high as it is always above 85%. However, 8 models have low MCC and Threat Scores. These are considered not useful for plasmid classifications because their Threat Score is lower than 50%. This exposes that Accuracy is obscuring where the plasmid contig classifications are incorrect and should not be used evaluate models of imbalanced datasets.

The Effect of Balancing

A closer look at Recall and Precision shows that the low MCC and Threat Score are due to low Precision [Fig10 Right]. Overall, balancing improved Recall considerably, especially for the most imbalanced datasets, meaning there are a lot less plasmid contigs incorrectly predicted to be chromosomal contigs. However, Precision for the most imbalanced datasets decreases more than the Recall increases, meaning unbalanced datasets now have more chromosomal contigs classified to be plasmid contigs [Fig10 Right]. Up to 85% of the time the model is classifying the contigs to be plasmid contigs, when they are actually chromosomal contigs. Confidence in plasmid classifications is therefore very low as the model's plasmid classifications cannot be trusted. Real world datasets would likely have an even less reliable plasmid classification rate.

For example, the Recall of the balanced Vibrio model increases from 25% (unbalanced model) to over 90% (balanced model) but the precision goes down from 90% (unbalanced model) to 15% (balanced model). Which means if the balanced Vibrio model classifies a contig to be a plasmid contig, there is an 85% chance that it is not a plasmid contig.

This was not the intended effect of balancing. A low Precision of say 60% might be acceptable if these models were only used as filters and further tests conducted. However, these Precision scores are alarmingly low, with eight models having correct plasmid contig classifications less than 50% of the time. Clearly, balancing the datasets during model training did not solve the imbalance for the most imbalanced datasets. Instead of solving the problem, balancing merely displaced the problem and made it worse. But it was precisely these imbalanced datasets that needed balancing the most. The Precision of these models on real-world datasets are expected to be much lower even, suggesting that the plasmid classifications or these models will be useless for most purposes.

3.2. Model Generalizability

To estimate how the model would perform on a dataset it has not seen during training, the performance on the test dataset and the performance on the training dataset were compared (in this case the OOB was used as a stand in for the test dataset). The OOB scores [Fig10] are subtracted from the training scores. This difference between the two scores is shown in [Fig11]. The test Accuracy is at the most 10 points lower for the test data. The Accuracy is very consistent between the test and train data which fails to represent that there are considerable differences between the two scores in the MCC and Threat Score. These differences can be seen clearly when looking at the MCC and Threat Score columns [Fig11]. For example, the Listeria model has a training Threat Score of 91% when classifying the training dataset, which is good. But the Threat Score is only 65% when the same model classifies the test dataset, which is not good. As the chart shows in the MCC and Threat Score columns, many models show a considerable difference between the training and test metrics. Many models that have an acceptable MCC on the test dataset, have a much higher MCC on the training dataset (Borrelia, Rhizobium, Cyanothece, Lactococcus and Listeria). Only four models have a difference in Threat Score of less than 10 points (Borrelia, Campylobacter, Burkholderia and Enterobacteriaceae). Overall, there are large differences which further demonstrate overfitting. Classifications on real-world data are expected to be worse than reported here as the models are indeed overfit.

3.3. Model Complexity

A look at how complex the models are, in terms of the size of the trees in the forest and the model size in megabytes [Fig 12], indicates that the models are very complex. There is a strong relationship between how imbalanced and how large the datasets are and how complex the models are. That should not be the case. Dataset size and degree of imbalance should not determine model size. By definition, a model must generalize the information in the training dataset and therefore reduce its size. The model must be a simplification, otherwise it defeats the purpose of having a model. However, most random forest models were larger than the input dataset, which means the random forest is expanding the data rather than simplifying it. This is also a sign of overfitting and shows that the models will not be generalizable.

Further, the trees are growing very deep and wide. Enterobacteriaceae has the largest taxon-specific dataset and has an average of 2,103 nodes per tree in the forest. This illustrates how the trees grow until all contigs are split off into a pure node, which requires many splits because the dataset has more contigs compared to the other datasets. When using the Enterobacteriaceae model for classification, it means going through if-then steps of the form, "if the contig has kmerX < 0.00687 and kmerY < 0.00485 and kmer < 0.00222 and length < 2658 and plasmid genes > 5 and kmerX > 0.00123 and plasmid genes <11 … (21 more if-then steps excluded)". On average, there are 14,000 of such individual if-then steps (28 logic steps per tree and 500 separate trees) required before a single contig is classified. With this level of fine-grained distinctions, it is doubtful whether the trees in the Enterobacteriaceae model are reflecting the underlying plasmid biology. What holds for other algorithms, that the larger the datasets, the easier it is to generalize to the most important aspects of a dataset clearly does not apply to random forests. Besides, if this is just a way to memorize the dataset, there are more efficient ways to do this. Only 4 datasets have a smaller model size compared to training dataset size. For big datasets, this also means that the models will be unwieldy to use, loading 2 GB for the Bacteria model or 900 MB for Enterobacteriaceae can be a problem for computers that do not have that kind of memory available.

3.4. Model Explainability

Given the complexity of the models, shown above and in [Fig12], explaining the classifications of the models is not possible. As the models grow in complexity, they become increasingly less transparent and reproducible as well. A slight change in the dataset could yield significant changes to the features used, the trees and the splits applied to make classifications.

The feature importances [Fig13] show the effect of three feature selection approaches: No Selection, Simple Selection and Recursive Selection. A higher number in the feature importance means the feature is used more often. The 'plasmid genes %' and 'plasmid genes #' features are consistently the highest scoring features in all models. The 'plasmid genes' features are predictive of plasmid contigs which does make biological sense. The next-best features are nearly indistinguishable in terms of importance. For example, in Cyanothece, with 'No Selection', the importance of 'plasmid genes %' and 'plasmid genes #' together contribute only 15% with the remaining 85% split between the remaining 1029 features. The 3rd most important feature, 'SCM genes #', contributes only about 1% to making classifications. Kmers appear to add mostly noise. Feature selection appears to help raise the importance of the features in the Top 20, by cutting out many very low scoring features, but the effect is small at best. In the Cyanothece models, recursive feature selection is selecting out features that previously were in the Top 20. Specifically features 8, 11, 12, 15, 16, 18, 19 in 'No Selection' are removed entirely from the 'Recursive Selection' model. This is surprising, that seven features out of the Top 20 would be entirely absent after recursive feature selection. It would be expected that Top 20 features from the 'No Selection' model would have a much higher score after the 'Recursive Selection,' but instead they are completely removed. Another surprising point is that, in general, the larger the dataset, the more features are left after feature selection (not shown here). This should not be the case, as it is the dataset size that should not affect how many features are selected as informative for the model. The feature importances look as though the models have indeed overfit as, for Campylobacter, not a single non-kmer feature made it into the Top 20 (except plasmid genes of course), which is quite unexpected. After analyzing the feature importances, the random forest does not appear to be explainable or transparent in making decisions.

4. Discussion

4.1. Are the Models Good Enough?

The lead question was 'What is a good model?' Three crucial aspects were focused on: generalizability, explainability, and usability. The two main ingredients of a model are the training dataset and the algorithm (including its implementation). Shortcomings of the dataset were addressed through feature selection and balancing by resampling. Unfortunately, that does not appear to have been sufficient. Fully grown trees in a random forest cause overfitting. Balancing did not solve imbalance. Feature selection did not make the models more intelligible. Most models are very complex, which causes them to score low in terms of generalizability, explainability and usability. They are not a meaningful simplification that captures the essential features that will be found in other datasets. The models showed large differences between training dataset classifications and test classifications.

Assessing the models individually shows that even when lenient thresholds are employed, all models fall short in at least one critical area. These shortcomings cannot be compensated, as they are all individually vital parts. Consider a car with three tires; a better motor will not compensate for the missing tire. See the list of critical requirements for a model in [Box2].

Box 2 ***Critical Requirements that must be met to make a good model***

1. Biology: If it is established that a bacterium has intermediate replicons that will not fit into the binary chromosome/plasmid model, a binary model should not be used to model this bacterium.
2. Dataset Duplicates: Having duplicate contigs in a dataset, which have identical features, means that the random forest will either overfit (in case of consistent labels) or be internally inconsistent (in case opposite labels for the features).
3. Performance: If less than 50% of the plasmid classifications are correct (as seen from the Threat Score), then a model cannot be reliably used for classifying plasmid contigs. Any contig that is classified as a plasmid contig needs additional external validation, so that the model cannot be used as a stand-alone tool.
4. Overfitting: If there is a big difference in scores between test and training data, the model is overfit and does not generalize. Note that because of duplicates in the data, the scores are artificially increased/decreased.
5. Complexity: If a model is so complex that it is larger than the dataset, it is not a meaningful simplification and misses the purpose of having a model. Also, the larger the model, the harder it is to explain how it makes classifications.

The outcome of the assessment for all 18 models are found in [Table4]. The worst model is Vibrio, which fails on all 5 points. The "best" models are Campylobacter, Bacillus, Enterobacteriaceae and Bacteria, they have just 2 disqualifying points. Overall, the conclusion is that none of the balanced models passed all tests. Many models look acceptable or even good from one or another perspective, but none is good in a holistic sense of having all acceptable metrics. If Accuracy above 85% had been the only requirement for model assessment, then they would all have passed, but as seen above, more is needed to be a good model to classify contigs than high Accuracy.

4.2. Why do these models not meet expectations?

There are several core reasons why the models do not work as intended. All major components have critical issues; the dataset, the algorithm, the models and even the underlying model of bacterial replicons. See [Table5] for a summary of the issues and [Box4] for a summary of the major take-aways.

Critical Dataset Issues

The dataset is reflecting a binary model of bacterial replicons [Fig1]. This is the simple model that one finds in schoolbooks. Microbial research shows bacterial genome biology is more complex. The examples listed in [Box3] defy the idea that there is a clear-cut distinction between chromosomes and plasmids. By assuming that long contigs must be originating from chromosomes, the datasets are not realistic and needlessly binary. Whereas it still is possible to make a meaningful distinction between chromosomes and plasmids based on their mode of reproduction, one needs to be careful to not fit the data to an oversimplified model.

Box 3 ***Ways in which bacterial replicons are not so binary***

1. Plasmids can get integrated into a chromosome.
2. Essential and non-essential genes can be found on both plasmids and chromosomes.
3. There is a lot of between-strains variation, what is on a plasmid in one strain might be on a chromosome in the next.
4. Plasmids can be transferred to bacteria of a different genus or even domain.
5. Plasmids can be lost.
6. Whole chromosomes can be transferred between bacteria[21].
7. There exist intermediate type replicons called chromids[22] (also referred to as secondary chromosomes or long essential plasmids), which look part plasmid part chromosome. At least 4 taxons represented in the datasets are known to have chromids[22]: *Burkholderia, Vibrio, Rhizobium,* and *Cyanothece.*
8. Some bacteria have differing genome architectures. *Borrelia* which causes Lyme disease has a linear chromosome and a mix of linear and circular plasmids of which some are essential for survival[23].

The datasets are not representative of real-world contig datasets. Error-free simulated chromosomal and plasmid contigs from reference genomes are the best-case scenario. Any realistic dataset will have contaminants, a class for contigs that map to both chromosomes and plasmids, biases introduced by sequencing and assembly techniques. A realistic model also needs at least four classes, 'chromosomal contig', 'plasmid contig', 'maps to both chromosome and plasmid sequences' and 'none of the above'. Bacteria known to have intermediate replicons additionally need a class for the plasmids that look like chromosomes. The current binary models would only be usable in a setting where the contigs are already pre-processed in the same way as the training dataset (including the mapping of long plasmid contigs to chromosomes and filtering out all contaminants and sequences that do not map to reference genomes) before classification. As it stands now, if no preprocessing occurs, any sequence will be classified as a chromosomal or plasmid contig, even if it is eukaryotic DNA. But if this much preprocessing is required, there is no more need for a model, as the preprocessing could be a classification in and of itself.

Another critical issue concerning the datasets, is that they contain many duplicates [Table3], making it unfit for the random forest algorithm. Having duplicate data defeats the idea of bootstrapping and bagging as the trees will be highly correlated (when duplicates have consistent labels) and makes the trees and the forests needlessly complex (when duplicates have contradicting labels). Duplicates in combination with the (sometimes extreme) imbalance of the datasets makes the problem worse. Balancing through resampling was shown not to solve the imbalance problem, instead, it merely displaced the problem and made it worse for the most imbalanced datasets. Additionally, the kmers made the datasets needlessly wide and noisy, which gave the random forests ample room to overfit.

Critical Algorithm Issues

Both the random forest algorithm and its implementation in SKLearn did not meet expectations. They have not been found to be plug-and-play as advertised both by Breiman[8] and the Scikit-Learn Documentation. Two major downsides of the SKLearn implementation that were encountered are especially concerning, namely that it is not clear that the default settings must be adjusted so much to get a good model and that there is no default class specific resampling feature like there is in the R randomForest package. To have options called 'balanced' in the main documentation without mentioning the need for a separate add-on package for resampling gives the impression that there is an easy solution to the balancing problem.[7] The Imbalanced-Learn package suggests in its documentation that imbalance is solved when the balanced accuracy goes up, but 'balanced' accuracy does not take into account the Precision and is therefore not a sufficient measure of model improvement.[13] As was seen above in section 3.1, a higher Recall (which plays into Balanced Accuracy) came at the cost of Precision for the most imbalanced datasets.

The random forest, when trees are fully grown, overfits. The bigger the dataset, the bigger the forest. That should not be the case, a model should be a simplification. More data should mean it is easier to detect meaningful features as in more conventional models, but the random forest instead grows bigger trees. The random forest trains on the noise and the signal gets lost. This is not to say that the random forest picks up no signal at all. Breiman's Random Forests often seem to robustly pick the best features, and the worst features get filtered out (or rather, made 'less important'). The question becomes what the added value of using a random forest is compared to using a simpler model that also robustly picks the best and worst features. Because there is no adequate penalty applied in the algorithm to compensate for the complexity of the trees, they grow very deep. This in combination with giving equal importance to leaves containing few samples and those that contain many, the random forest appears to overfit by design, contrary to what Breiman claims in his paper[8]. There are many options to limit tree growth (called pruning) before they start to overfit, but this is not part of the original Breiman algorithm and the trees in the RFPlasmid models are also fully grown.

Maybe random forests can be used in the exploratory phase to detect features to use in a simpler model. But considering the computational cost of random forests, it would be worthwhile to first determine whether simpler means of feature selection are not just as good or even better. A look into other algorithms such as boosting or more regression-based models is also an option, however these models often require adding assumptions such as normality and other distribution-based requirements which random forests are not subject to. Boosting methods overfit even faster than random forests, which is why Breiman thought of his version of the random forest in the first place[8]. Combining different methods in an ensemble model is also an option. But all these methods and algorithms have assumptions that must be considered and require separate analysis.

Critical Model Issues

As telling from [Table4], all of the models have some major flaw after internal validation. If the models were to be validated with separate datasets, the outcome is expected to be much worse. The combination of random forests with datasets full of duplicates yield models that look good when seen from the majority perspective, but a closer look reveals that they are either poor at predicting plasmids or overfit, and therefore not fit for predicting plasmids because they are not generalizable. These models defeat the purpose of having a model, as they only work within the very narrow constraints of their training dataset, and it does not appear reasonable based on these results to use the models in real-life applications (such as for risk assessment, decision making or as medical devices) as the confidence in their correctness and reliability is low. Additionally, the models and code would have to be thoroughly tested and validated before use. Overfit models also tend to be less explainable and usable, as overfit random forests are complex, making them hard to understand and in some cases the model file size is so big that they might not be able to be used on all computers.

4.3. Recommendation

These models were found to not be good enough to make reliable predictions for plasmid detection. This does not mean that models must be 'perfect' to be used. But they should be 'good enough', fit for purpose and transparent as to their shortcomings. These models do not appear to be acceptable for real-life applications. Every issue discussed is a potential showstopper in and of itself. Some models look good when considering the Threat Score (The Top 5 for Threat Score in [Fig11]: Borrelia, Enterobacteriaceae, Rhizobium and Campylobacter) and all models have Accuracy scores of over 85%. This should be especially alarming, because they all show fatal flaws in other areas that indicate overfitting. When this is the case, a model appears to be 'accurate' even when it is not generalizable, the high Accuracy scores are due to memorizing the dataset (overfitting) but will not work on other datasets. In this situation, selective reporting of the "best metrics" might not only overestimate the performance but could make potential users overconfident [Box1]. If the choice is to make the models available (not recommended) there must be very clear warnings and disclaimers that predicted plasmids often cannot be trusted and disclose the known limitations of the models and training data. These warnings must be understandable by the end-user so they can make an informed decision to use a model.

These findings have implications for other models as well. As RFPlasmid models are nearly identical to the models described here (same datasets, algorithm, and balancing approach, with only a different package used to build the random forests), all listed issues are expected to substantially apply to them as well. Other kmer based models for plasmid detection are also expected to have problems of overfitting, low explainability and high complexity. In general, fully grown random forest models are expected to overfit to the training data. All binary plasmid classifiers cannot be used as standalone tools, as they require additional classes to deal with contaminants, long chromosome-like plasmid contigs, and contigs that map to both chromosomes and plasmids. They require additional tools before classification to preprocess the contigs and after classification to validate the results. (Extremely) imbalanced datasets are the norm in Bioinformatics (for example in genome-wide association studies), meaning most findings are spurious correlations and need additional lines of evidence.

The goal was to improve RFPlasmid by making an implementation in SKLearn. Instead, many critical issues came to light that cast doubt on the approach altogether. There are many possible minor interventions to improve a random forest, by pruning for example. The dataset can be cleaned and clustered beforehand, the kmers removed, and multiple classes added. But the problems here are so substantial, that there is no quick fix. The datasets seem to need a complete makeover to make them more realistic and remove noise and duplicates. The type of data does not fit the biology, the random forest does not fit the data, the models are too overfit to be useful. This is an example where machine learning might not be the way to go. Mapping to references in combination with sequence graph approaches appear to be a better fit for plasmid detection; without the dataset issues, problems handling duplicate data, and less affected by class imbalance. This approach would allow to go beyond the binary classes and be more realistic through the inclusion of known contaminant sequences. Contigs would not be forced into having to be either a plasmid or chromosome contig. This mapping pipeline could also be automated. In the meantime, based on the insights from this analysis, using SKLearn, kmers or fully grown random forests for plasmid detection cannot be recommended.

Box 4 ***Main Takeaways***

1. Random Forests with fully grown trees overfit, especially on noisy and duplicate data.
2. The datasets are wide, noisy, imbalanced, unrealistic, and have many duplicate features & contigs.
3. The combination of these datasets and the fully grown random forest produce overfit models that are not generalizable.
4. The datasets and models have only 2 classes, where at least 4 are required to be realistic.
5. The models look good in terms of Accuracy and Recall, but the Precision is very low for the most imbalanced datasets.
6. Accuracy is not a good summary metric to use for model evaluation, as it can be high but hide poor performance of other metrics.
7. The models have poor performance, are overfit and/or are complex. This makes them not generalizable, explainable, or usable.
8. The models have not been externally validated, therefore their true performance on real data is unknown, and is expected to be much lower.
9. Because the models are not generalizable, they are not expected to make reliable and correct classifications on real-world contig datasets. Therefore, these models should not be used for real-world applications.

5. Final Remarks

5.1. Reported Findings

The analysis presented in this report is representative of the many analyses that were carried out and not shown in this report. Much care was given to report the general trends, and not the exceptions. Many more models were trained and assessed with a variety of metrics with the same general findings.

5.2. Code Availability

The training pipeline and contig datasets can be found at:
https://github.com/e-gill/RFPlasmid2_prototype

6. Bibliography

1. aldertzomer/RFPlasmid: Predicting plasmid contigs from assemblies using single copy marker genes, plasmid genes, kmers - Developed by Linda van der Graaf. https://github.com/aldertzomer/RFPlasmid.
2. Edgar, R. C. & Bateman, A. Search and clustering orders of magnitude faster than BLAST. *Bioinformatics* **26**, 2460–2461 (2010).
3. Parks, D. H., Imelfort, M., Skennerton, C. T., Hugenholtz, P. & Tyson, G. W. CheckM: assessing the quality of microbial genomes recovered from isolates, single cells, and metagenomes. *Genome Res.* **25**, 1043–1055 (2015).
4. Andy Liaw, by, Wiener, M. & Andy Liaw, M. Package 'randomForest' Title Breiman and Cutler's Random Forests for Classification and Regression. (2018) doi:10.1023/A:1010933404324.
5. Pedregosa FABIANPEDREGOSA, F. *et al.* Scikit-learn: Machine Learning in Python Gaël Varoquaux Bertrand Thirion Vincent Dubourg Alexandre Passos PEDREGOSA, VAROQUAUX, GRAMFORT ET AL. Matthieu Perrot. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011).
6. Bloois, L. van der G. van, Wagenaar, J. A. & Zomer, A. L. RFPlasmid: Predicting plasmid sequences from short read assembly data using machine learning. *bioRxiv* 2020.07.31.230631 (2020) doi:10.1101/2020.07.31.230631.
7. sklearn.ensemble.RandomForestClassifier — scikit-learn 0.24.2 documentation. https://scikit-learn.org/0.24/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier.
8. Breiman, L. Random forests. *Mach. Learn.* **45**, 5–32 (2001).
9. Decision tree learning - Wikipedia. https://en.wikipedia.org/wiki/Decision_tree_learning.
10. Oshiro, T. M., Perez, P. S. & Baranauskas, J. A. How Many Trees in a Random Forest? (2012) doi:10.1007/978-3-642-31537-4_13.
11. Smialowski, P., Frishman, D. & Kramer, S. Pitfalls of supervised feature selection. *Bioinformatics* **26**, (2010).
12. BalancedRandomForestClassifier — Version 0.8.1. https://imbalanced-learn.org/stable/references/generated/imblearn.ensemble.BalancedRandomForestClassifier.html#imblearn.ensemble.BalancedRandomForestClassifier.
13. Fitting model on imbalanced datasets and how to fight bias — Version 0.8.1. https://imbalanced-learn.org/stable/auto_examples/applications/plot_impact_imbalanced_classes.html.
14. Strobl, C., Boulesteix, A. L., Kneib, T., Augustin, T. & Zeileis, A. Conditional variable importance for random forests. *BMC Bioinformatics* **9**, (2008).
15. Beware Default Random Forest Importances. https://explained.ai/rf-importance/.
16. Permutation Importance vs Random Forest Feature Importance (MDI) — scikit-learn 1.0.1 documentation. https://scikit-learn.org/stable/auto_examples/inspection/plot_permutation_importance.html.
17. sklearn.feature_selection.SelectFromModel — scikit-learn 1.0.1 documentation. https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectFromModel.html.
18. sklearn.feature_selection.RFECV — scikit-learn 0.24.2 documentation. https://scikit-learn.org/0.24/modules/generated/sklearn.feature_selection.RFECV.html?highlight=rfecv#sklearn.feature_selection.RFECV.
19. Confusion matrix - Wikipedia. https://en.wikipedia.org/wiki/Confusion_matrix.
20. Chicco, D. & Jurman, G. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics* **21**, (2020).
21. Hall, J. P. J. Is the bacterial chromosome a mobile genetic element? *Nat. Commun.* **12**, 6400 (2021).
22. Harrison, P. W., Lower, R. P. J., Kim, N. K. D. & Young, J. P. W. Introducing the bacterial 'chromid': not a chromosome, not a plasmid. *Trends Microbiol.* **18**, 141–148 (2010).
23. Stewart, P. E., Byram, R., Grimm, D., Tilly, K. & Rosa, P. A. The plasmids of Borrelia burgdorferi: essential genetic elements of a pathogen. *Plasmid* **53**, (2005).

7. Figures

INTRODUCTION

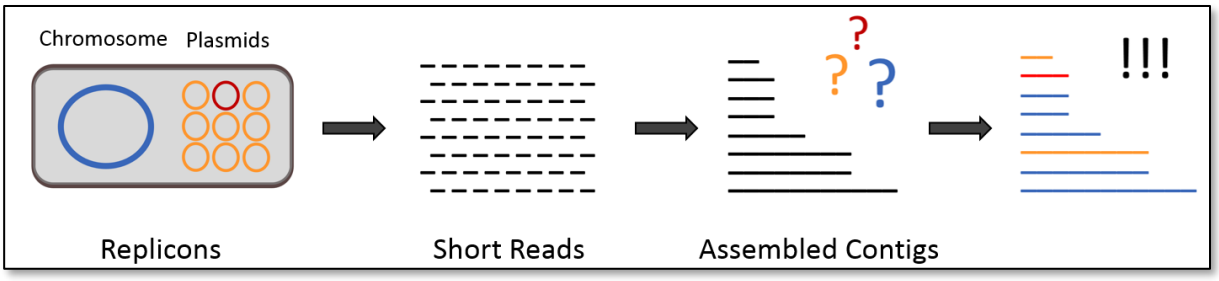


Figure 1. The Contig Classification Problem.

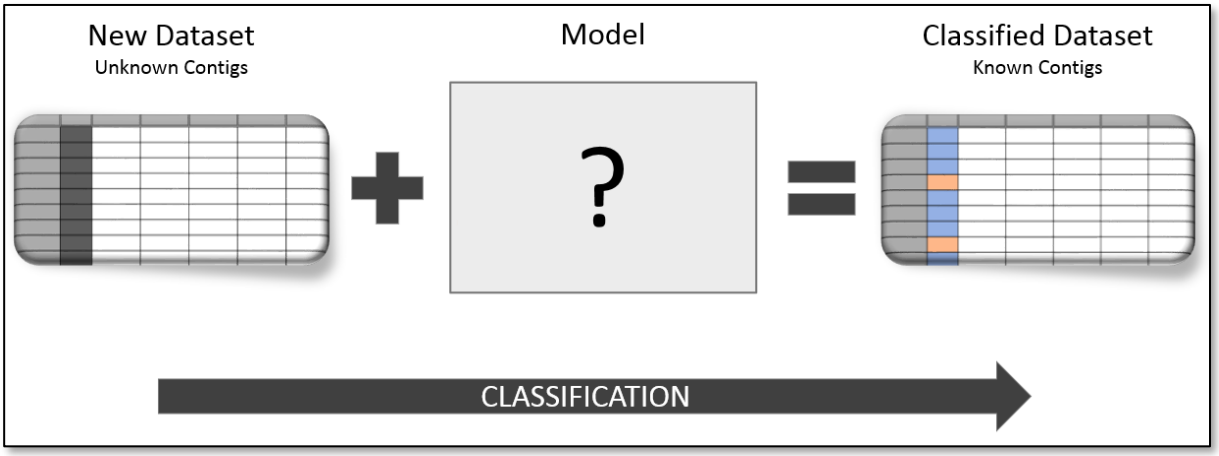


Figure 2. Goal of the Project: Develop Models to Classify Contigs of Unknown Origin.

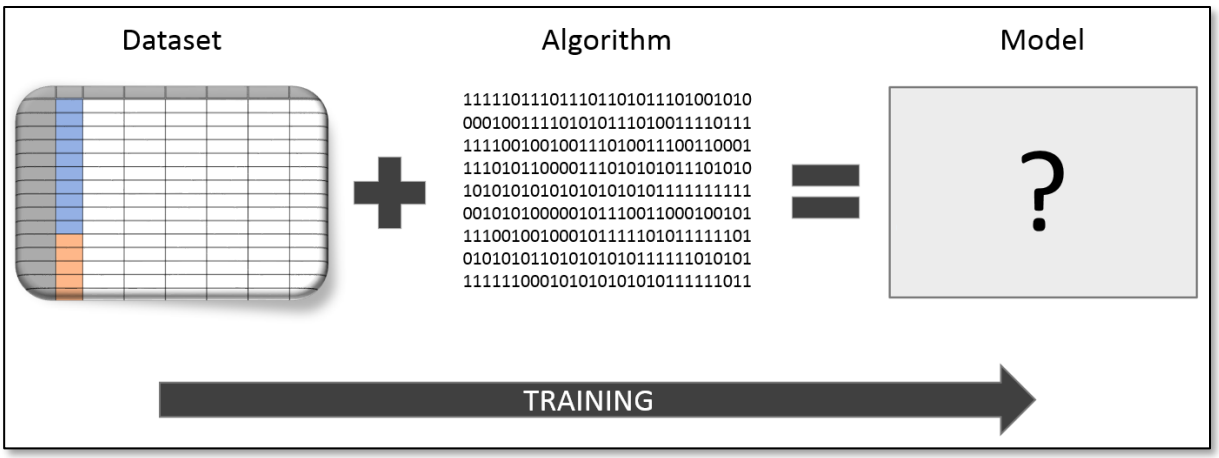
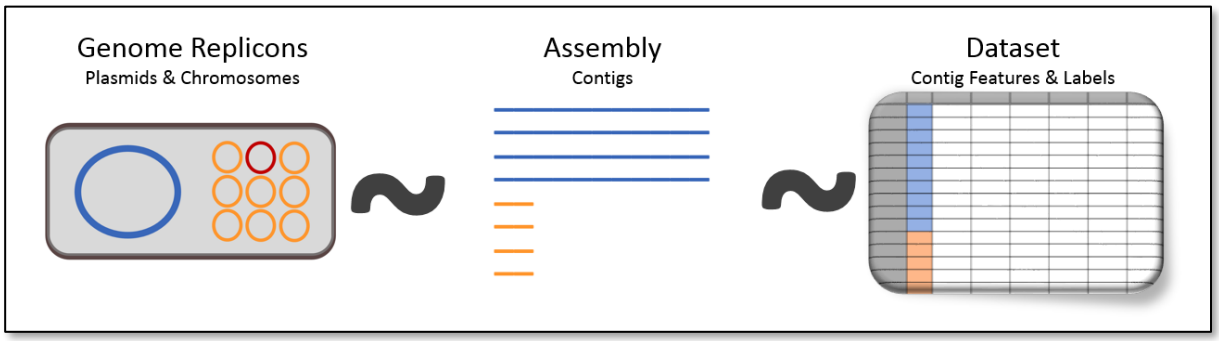
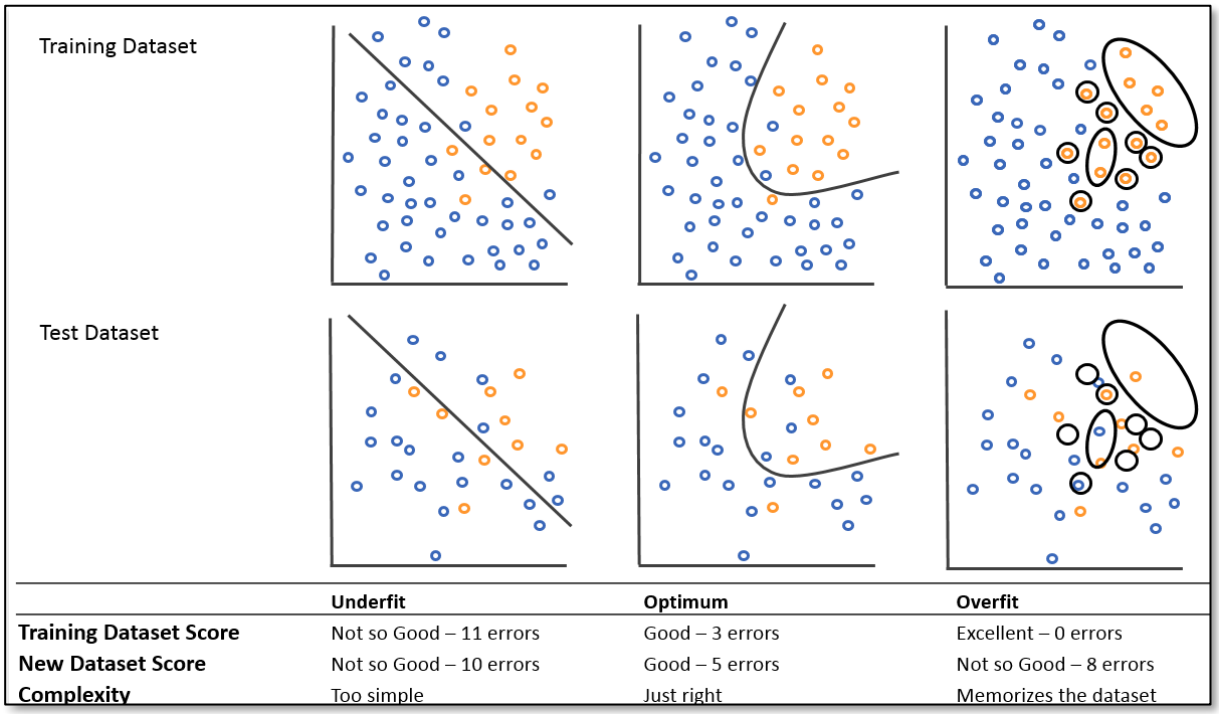


Figure 3. Models are made up of an Algorithm that is trained on a Dataset with known labels.

GENERAL APPROACH

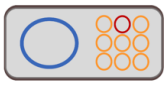
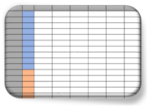
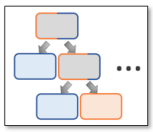
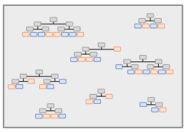




	Underfit	Optimum	Overfit
Training Dataset Score	Not so Good – 11 errors	Good – 3 errors	Excellent – 0 errors
New Dataset Score	Not so Good – 10 errors	Good – 5 errors	Not so Good – 8 errors
Complexity	Too simple	Just right	Memorizes the dataset

Figure 4. Basic Modeling Assumptions. Top: The dataset is realistically representing the biology, so that a model trained on that dataset makes good distinctions that are true, reliable and relevant. **Bottom:** Goal is not for a model to perfectly fit the dataset, leading to an overfit model that cannot be generalized, but a model that extracts the relevant features of the dataset in a meaningful way, that can be applied to other datasets.

Table 1. Requirements for the pipeline parts.

	Binary Model of Bacterial Replicons	Simulated Contig Features Dataset	SKLearn Random Forest	Trained Model
				
Generalizable	Realistic • Representative of real-world biology in a relevant sense for the question	Realistic • representative sample	• Not easily overfitting • Good fit for data & question	Good Models ... • make distinctions that are true, reliable and relevant • have been validated internally and externally on many levels • robust
Explainable	• Not too simple / not too complex	• Relevant features • Reproducible	• Understandable • Transparent • Good documentation	Explainable • not too complex • transparent • understandable for expert, developer AND user
Usable	• Good fit to reality and question	• Good fit to reality, question and algorithm • Confidence in reliability • Not too big/small • Statistically sound	Developer-friendly • Good implementation • Not too big / slow / too many dependencies • reliable, production grade code	User-friendly • not requiring super computer • reliable • safe to use • fit for purpose

DATASET

Table 2. Dataset metrics, sorted by Plasmid percentage.

		Total Contigs	Chromosome Contig	Plasmid Contig	Chromosome Contig	Plasmid Contig	Imbalance
Taxon	Dataset ID	#	#	#	% contig count	% contig count	% contig count
Clostridium	CLOM	6,537	6,393	144	98	2	98
Pseudomonas	PSES	18,645	18,200	445	98	2	98
Vibrio	VIBO	11,265	10,965	300	97	3	97
Corynebacterium	CORM	4,614	4,487	127	97	3	97
Burkholderia	BURA	26,256	25,291	965	96	4	96
Streptomyces	STRS	6,449	6,104	345	95	5	95
Staphylococcus	STAS	9,124	8,594	530	94	6	94
Listeria	LISA	2,685	2,480	205	92	8	92
Lactobacillus	LACB	19,412	17,633	1,779	91	9	91
BACTERIA	AB17	222,723	194,597	28,126	87	13	87
Campylobacter	CAMR	5,423	4,596	827	85	15	85
Bacillus	BACS	20,055	16,903	3,152	84	16	84
Lactococcus	LACC	3,423	2,661	762	78	22	78
Enterococcus	ENTS	6,270	4,543	1,727	72	28	73
Cyanothece	CYAE	634	416	218	66	34	66
Enterobacteriaceae	ENTE	28,544	17,031	11,513	60	40	60
Rhizobium	RHIM	4,241	1,825	2,416	43	57	57
Borrelia	BORA	1,564	110	1,454	7	93	93

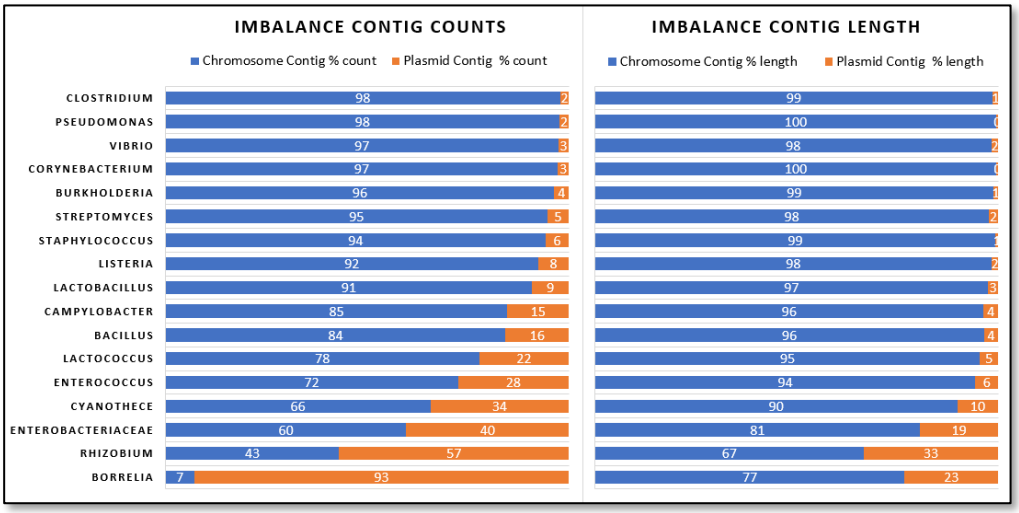
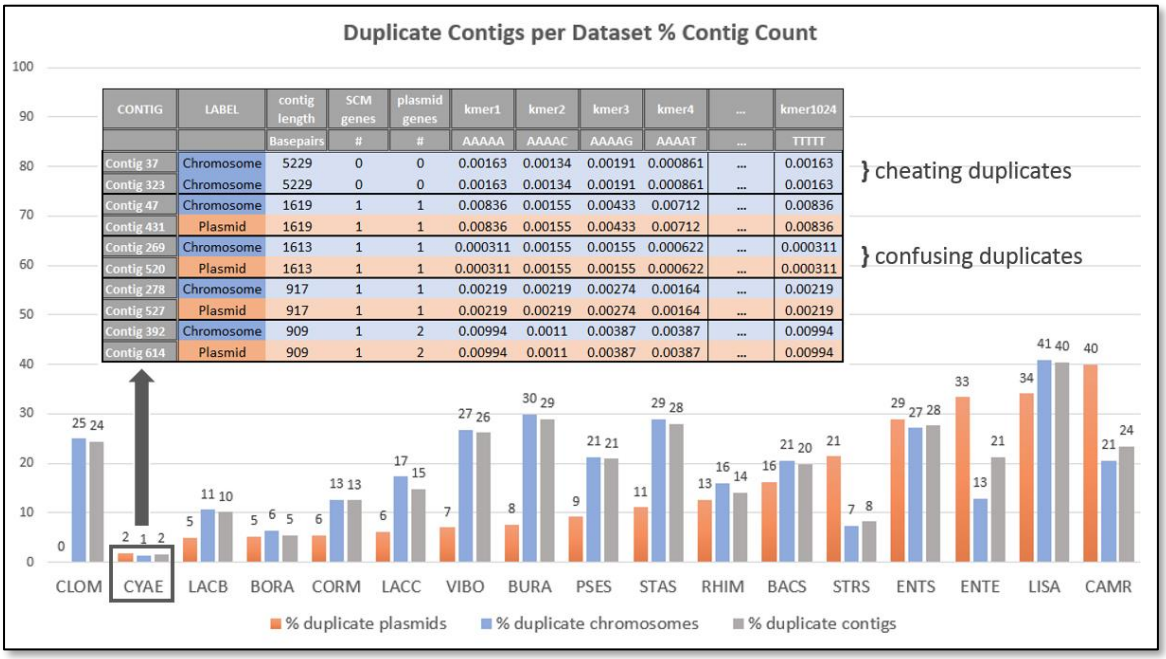


Figure 5. Imbalance of datasets in terms of contig count (left) and length (right). Imbalance based on length is even more imbalanced and biased towards the chromosomal contigs.

DATASET Continued

Table 3. Subsets of the smallest dataset (Cyanothece) for illustration. **Top:** Rows are contigs, columns are the contig label (chromosomal or plasmid) and 1031 features. **Middle:** The rows contain duplicates. **Bottom:** There are correlated and identical features in the datasets.

CONTIG	LABEL	contig length	SCM genes	plasmid genes	kmer1	kmer2	kmer3	kmer4	...	kmer1024
		Basepairs	#	#	AAAAA	AAAAC	AAAAG	AAAAT	...	TTTTT
Contig 1	Chromosome	390177	205	0	0.00473	0.00242	0.00233	0.00437	...	0.00473
Contig 2	Chromosome	178352	94	0	0.00459	0.00233	0.0022	0.00421	...	0.00459
Contig 3	Chromosome	155656	90	0	0.00482	0.00245	0.00228	0.00457	...	0.00482
Contig 4	Chromosome	137379	70	1	0.00435	0.00237	0.00218	0.00398	...	0.00435
Contig 5	Chromosome	134629	79	0	0.00458	0.00224	0.00229	0.0042	...	0.00458
Contig 6	Chromosome	128570	74	0	0.0043	0.00239	0.00238	0.00403	...	0.0043
Contig 7	Chromosome	108010	36	0	0.005	0.00273	0.00282	0.00443	...	0.005
Contig 8	Chromosome	102479	52	0	0.00443	0.00236	0.00219	0.00393	...	0.00443
Contig 9	Chromosome	94603	62	3	0.00509	0.00251	0.00252	0.00446	...	0.00509
Contig 10	Chromosome	80019	41	0	0.00446	0.00228	0.00235	0.00424	...	0.00446
Contig 11	Chromosome	73588	36	1	0.0044	0.00232	0.00234	0.00432	...	0.0044
Contig 12	Chromosome	360727	208	2	0.00486	0.00248	0.00227	0.00445	...	0.00486
...	...	...	...	...	...	...	...	...	...	...
Contig 629	Plasmid	495	0	1	0	0	0.00102	0.00102	...	0
Contig 630	Plasmid	492	0	1	0.0154	0.0041	0.00205	0.0143	...	0.0154
Contig 631	Plasmid	466	1	0	0.0184	0.00325	0.00866	0.00649	...	0.0184
Contig 632	Plasmid	353	0	0	0.00143	0	0.00143	0.00143	...	0.00143
Contig 633	Plasmid	300	0	0	0.00169	0	0.00169	0.00169	...	0.00169
Contig 634	Plasmid	42100	6	12	0.0041	0.00241	0.00249	0.00355	...	0.0041



CONTIG	LABEL	contig length	contig length	SCM genes	SCM genes	plasmid genes	plasmid genes	kmer 1	kmer 1024
		Basepairs	kmer #	#	fraction	#	fraction	AAAAA	TTTTT
Contig 37	Chromosome	5229	10450	0	0	0	0	0.00163	0.00163
Contig 323	Chromosome	5229	10450	0	0	0	0	0.00163	0.00163
Contig 47	Chromosome	1619	3230	1	1	1	1	0.00836	0.00836
Contig 431	Plasmid	1619	3230	1	1	1	1	0.00836	0.00836
Contig 269	Chromosome	1613	3218	1	1	1	1	0.000311	0.00031
Contig 520	Plasmid	1613	3218	1	1	1	1	0.000311	0.00031
Contig 278	Chromosome	917	1826	1	1	1	1	0.00219	0.00219
Contig 527	Plasmid	917	1826	1	1	1	1	0.00219	0.00219
Contig 392	Chromosome	909	1810	1	0.5	2	1	0.00994	0.00994
Contig 614	Plasmid	909	1810	1	0.5	2	1	0.00994	0.00994

Essentially the same | Highly correlated | Highly correlated | **Identical !**

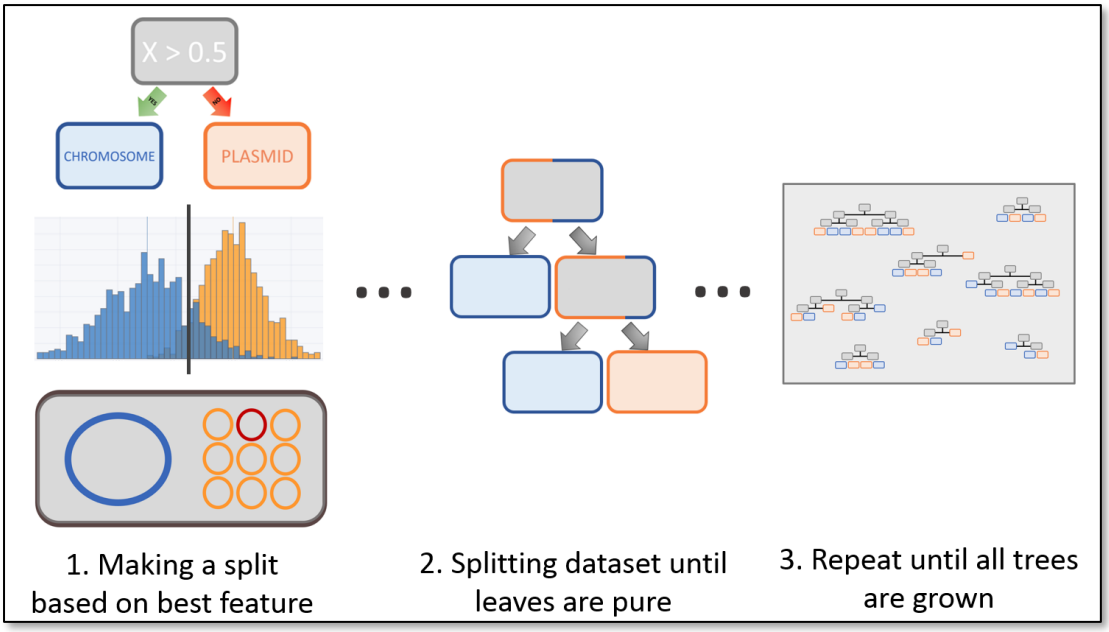


Figure 6. Building a Random Forest by making splits at each node of a Tree until the leaves of the tree are pure.

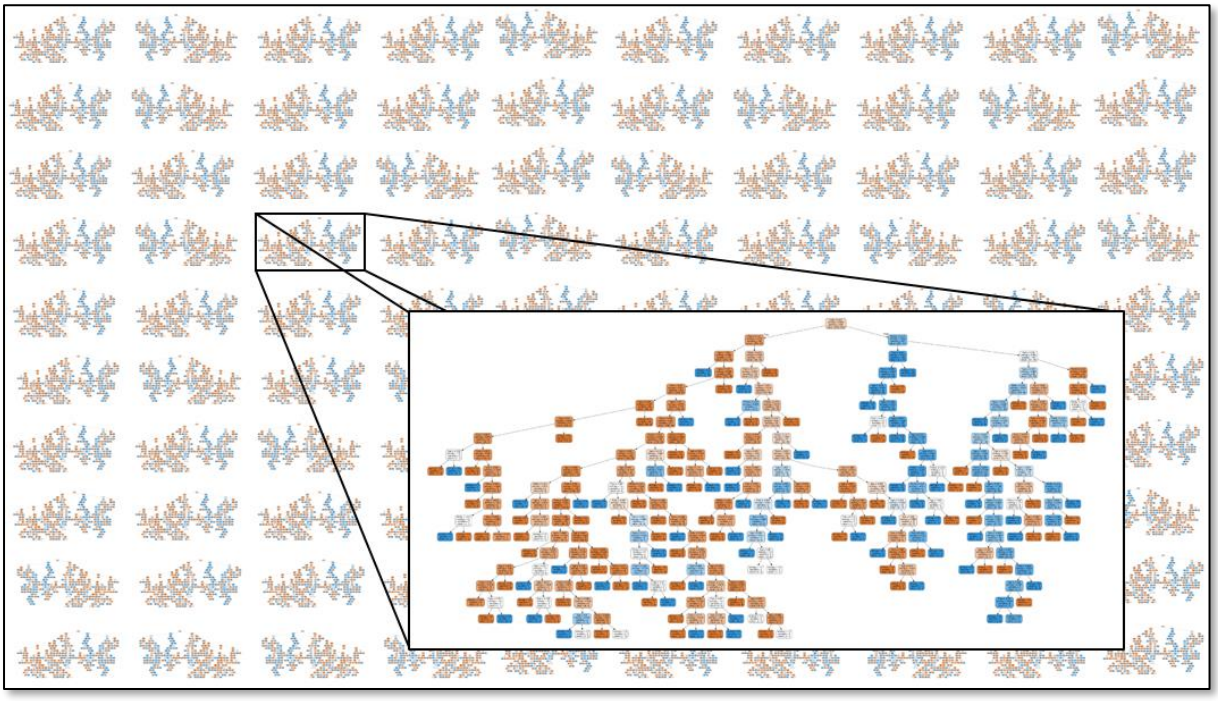


Figure 7. A random Forest made up of 100 trees of depth 20.

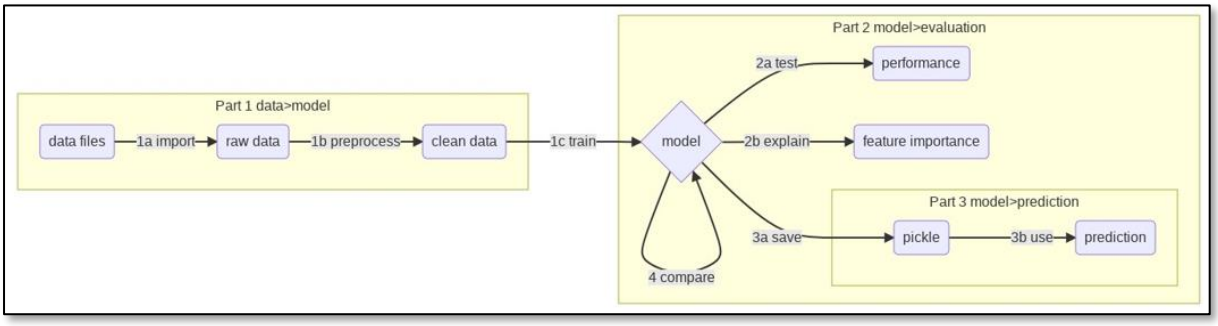


Figure 8. The SKLearn pipeline.

MODEL METRICS

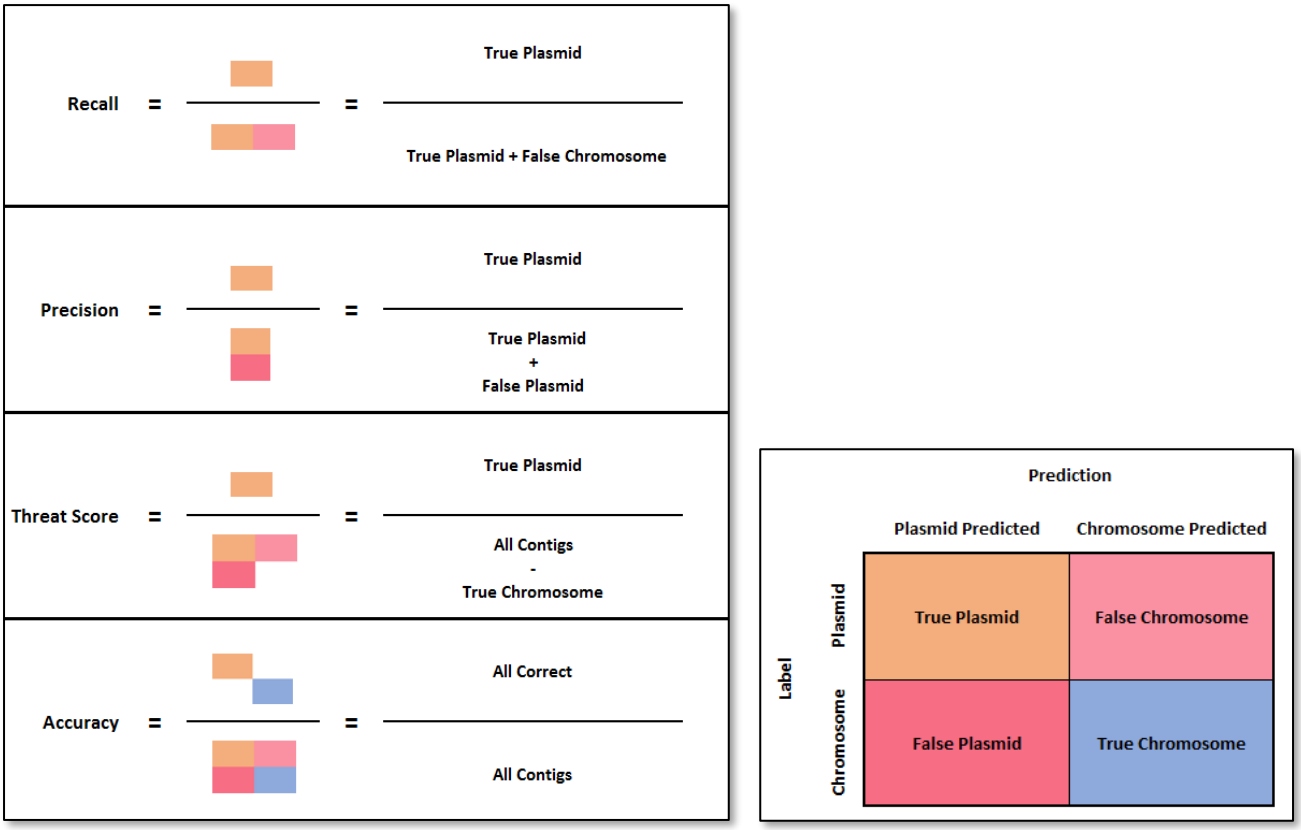


Figure 9. The Confusion Matrix and Derived Metrics. Recall, Precision and Threat Score are important for Plasmid classification. Accuracy is biased towards the chromosomal contigs because they are in the majority (except for Rhizobium and Borrelia).

MODEL PERFORMANCE

	Raw Confusion Matrix				Summary Prediction Metrics		
Model	True Positive	False Negative	False Positive	True Negative	Accuracy	MCC	Threat Score
Balanced	True Plasmid	False Chromosome	False Plasmid	True Chromosome		Matthew's Correlation Coefficient	
Borrelia	1,419	35	9	101	97	81	97
Enterobacteriaceae	11,021	492	431	16,600	97	93	92
Rhizobium	2,328	88	301	1,524	91	81	86
Campylobacter	814	13	175	4,421	97	88	81
Enterococcus	1,641	86	366	4,177	93	83	78
Cyanothece	203	15	42	374	91	81	78
Lactococcus	720	42	252	2,409	91	78	71
Bacillus	3,096	56	1,370	15,533	93	79	68
BACTERIA	26,567	1,559	11,810	182,787	94	78	67
Listeria	196	9	98	2,382	96	78	65
Staphylococcus	513	17	715	7,879	92	61	41
Lactobacillus	1,650	129	2,300	15,333	87	57	40
Streptomyces	316	29	697	5,407	89	50	30
Burkholderia	939	26	3,904	21,387	85	40	19
Corynebacterium	116	11	543	3,944	88	37	17
Vibrio	273	27	1,464	9,501	87	35	15
Clostridium	131	13	730	5,663	89	35	15
Pseudomonas	419	26	2,492	15,708	86	34	14

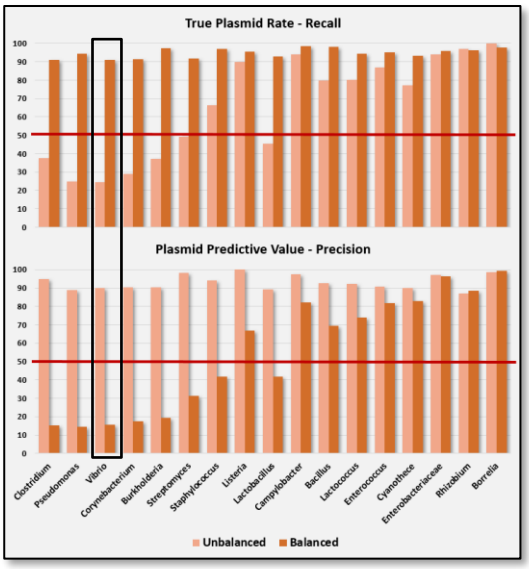


Figure 10. Confusion Matrix and Performance metrics for the Out-of-Bag of the Balanced Models.

MODEL GENERALIZABILITY

	Summary Prediction Metrics			Test - Train		
Model	Accuracy	MCC	Threat Score	Accuracy	MCC	Threat Score
Balanced		Matthew's Correlation Coefficient			Matthew's Correlation Coefficient	
Borrelia	97	81	97	-3	-17	-3
Campylobacter	97	88	81	-1	-4	-5
Burkholderia	85	40	19	-4	-7	-6
Enterobacteriaceae	97	93	92	-3	-6	-7
Pseudomonas	86	34	14	-7	-15	-12
Bacillus	93	79	68	-3	-9	-12
Staphylococcus	92	61	41	-3	-11	-12
Enterococcus	93	83	78	-4	-10	-13
Clostridium	89	35	15	-6	-16	-13
Vibrio	87	35	15	-7	-17	-13
BACTERIA	94	78	67	-3	-10	-14
Rhizobium	91	81	86	-9	-18	-14
Corynebacterium	88	37	17	-6	-17	-14
Streptomyces	89	50	30	-6	-18	-18
Cyanothece	91	81	78	-8	-16	-18
Lactococcus	91	78	71	-6	-15	-19
Lactobacillus	87	57	40	-7	-19	-22
Listeria	96	78	65	-3	-17	-26

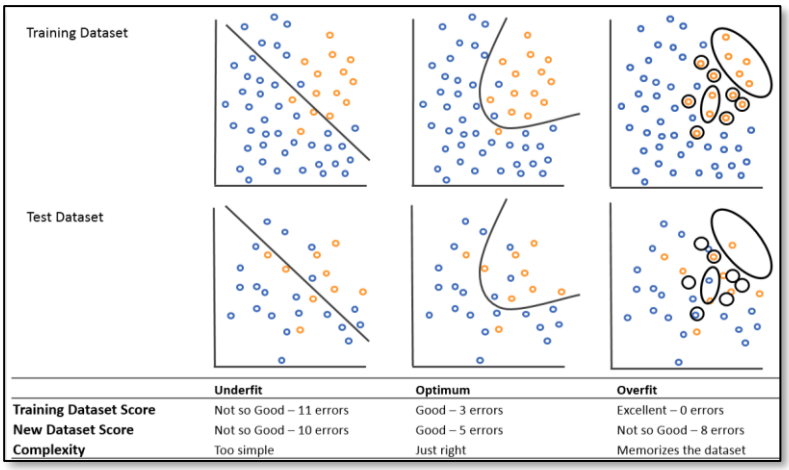


Figure 11. LEFT Overfitting Summary Metrics. the more red, the greater the difference between training performance and the test performance in percentage points (here the test performance is the out-of-bag performance). Threshold 0.5. RIGHT The greater the difference between performance on test and training data, the more overfit the model looks.

MODEL COMPLEXITY

		Model Complexity		Forest Complexity		
Model	Dataset ID	Model size	Model size / Dataset size	Trees per Forest	Average Tree Depth	Average Nodes per Tree
		MB	ratio	#	#	#
Rhizobium	RHIM	11	0.3	500	6	23
Streptomyces	STRS	18	0.4	500	8	38
Burkholderia	BURA	48	0.2	2000	7	41
Corynebacterium	CORM	54	1.5	2000	7	44
Cyanothece	CYAE	19	3.8	500	9	72
Staphylococcus	STAS	104	1.5	2000	10	97
Listeria	LISA	59	2.8	1000	11	108
Vibrio	VIBO	89	1.1	1000	11	110
Pseudomonas	PSES	150	1.0	2000	11	117
Bacillus	BACS	68	0.4	500	14	155
Lactococcus	LACC	63	2.3	500	13	199
Clostridium	CLOM	158	3.1	1000	15	247
Enterococcus	ENTS	140	2.9	500	18	445
Borrelia	BORA	148	14.0	500	16	467
Lactobacillus	LACB	144	1.0	500	17	507
Campylobacter	CAMR	254	6.3	500	18	579
Enterobacteriaceae	ENTE	922	4.2	500	28	2103
BACTERIA	AB17	2253	1.1	500	36	5796

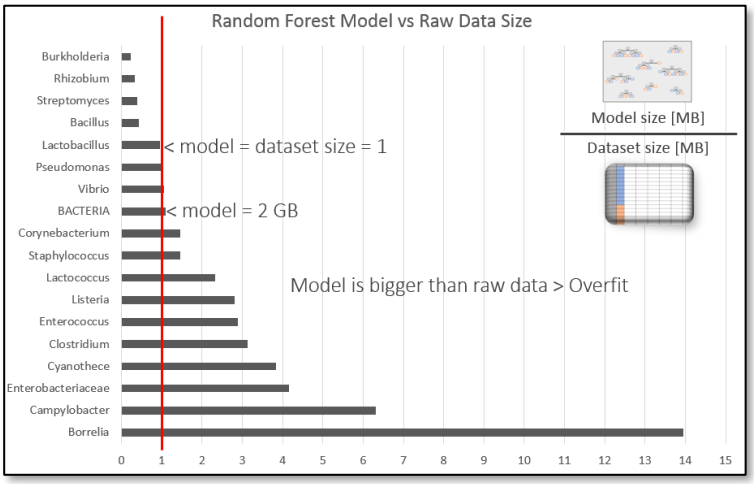


Figure 12. Complexity measures.

MODEL EXPLAINABILITY

Cyanothece					Campylobacter					BACTERIA				
Feature	Rank	No Selection	Simple Selection	Recursive Selection	Feature	Rank	No Selection	Simple Selection	Recursive Selection	Feature	Rank	No Selection	Simple Selection	Recursive Selection
plasmid genes %	1	9	19	35	plasmid genes %	1	6	12	16	plasmid genes %	1	14	27	27
plasmid genes #	2	6	13	20	plasmid genes #	2	4	9	9	plasmid genes #	2	6	13	13
SCM genes #	3	1	2	4	TATTC	3	2	4	3	SCM #	3	1	2	2
contig length	4	1	2	4	GAATA	4	1	4	4	contig length	4	1	2	2
GGGTA	5	1	2	3	AATTA	5	1	3	3	kmer #	5	1	2	2
SCM genes %	6	1	2	3	TAATT	6	1	3	3	plasmidcge	6	1	1	1
kmer #	7	1	2	4	GCATC	7	1	1	1	AAAAT	7	0	2	2
GGGTT	8	1	1		AATAT	8	1	1	2	SCM %	8	0	1	1
TACCC	9	1	2	3	GATGC	9	1	1	2	CGCGC	9	0	1	1
TTACT	10	1	1	3	ATAAT	10	1	1	1	ATTTT	10	0	1	1
ACTTT	11	0	1		AGAAT	11	1	1	1	CACGC	11	0	1	1
AAGTA	12	0	1		ATTCT	12	1	1	1	GCGCG	12	0	1	1
TTTCA	13	0	1	2	TTAAT	13	1	1	1	GCGTG	13	0	1	1
AGTAA	14	0	1	3	GAGAA	14	1	1	1	CCAAG	14	0	1	1
CCCCA	15	0	1		CTTTT	15	1	2	2	CTTGG	15	0	1	1
GATCC	16	0	1		ATTGT	16	1	1	1	CCCCT	16	0	1	1
GTGGG	17	0	1	2	AAAAG	17	1	1	2	GGCGT	17	0	1	1
TACTT	18	0	1		ATTAT	18	1	1	1	CAGAG	18	0	1	1
GGATC	19	0	1		GCTGC	19	1	1	1	AAAAG	19	0	1	1
TGAAA	20	0	1	2	GCAGC	20	1	1	1	CTCTG	20	0	1	1

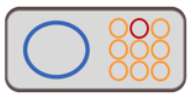
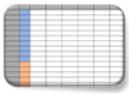
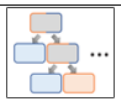
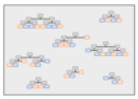
Figure 13. Top 20 Feature importances of balanced models %, sorted by feature rank of model without feature selection.

FINAL ASSESSMENT

Table 4. Final Assessment of Critical Issues of Model Performance. 5 critical categories were assessed and for each, a model could get 2 points, 0 points = disqualifying (red), 1 point = sufficient (yellow), 2 points = good (green). If any of these 5 catagories scores 0 points, it disqualifies the entire model, as these are essential requirements, which cannot be compensated.

	Scoring Category	Biology	Dataset Duplicates	Performance	Overfitting	Complexity	Total Score	Total disqualifying
	Critical Requirement	no known intermediate replicons	under 5%	> 50% TS	TS < 10 or MCC < 15	model size < dataset size	>= 50% Score	# disqualifying = 0
Model	Campylobacter	1	0	2	2	0	50%	2
	Bacillus	1	0	1	0	2	40%	2
	Enterobacteriaceae	1	0	2	1	0	40%	2
	BACTERIA	1	1	1	0	0	30%	2
	Rhizobium	0	0	2	0	2	40%	3
	Streptomyces	1	0	0	0	2	30%	3
	Burkholderia	0	0	0	1	2	30%	3
	Cyanothece	0	1	2	0	0	30%	3
	Lactococcus	1	0	2	0	0	30%	3
	Enterococcus	1	0	2	0	0	30%	3
	Borrelia	0	1	2	0	0	30%	3
	Listeria	1	0	1	0	0	20%	3
	Pseudomonas	1	0	0	0	1	20%	3
	Lactobacillus	1	0	0	0	1	20%	3
	Corynebacterium	1	0	0	0	0	10%	4
	Staphylococcus	1	0	0	0	0	10%	4
	Clostridium	1	0	0	0	0	10%	4
	Vibrio	0	0	0	0	0	0%	5

Table 5. Summary of Requirements and Outcomes of the most important parts of the pipeline.

	Binary Model of Bacterial Replicons	Simulated Contig Features Dataset	SKLearn Random Forest	Trained Model
				
Generalizable	**Requirement**	**Requirement**	**Requirement**	**Requirement**
	Realistic • Representative of real-world biology in a relevant sense for the question	Realistic • representative sample	• Not easily overfitting • Good fit for data & question	Good Models ... • make true, reliable & relevant distinctions • validated on many levels
	Outcome	**Outcome**	**Outcome**	**Outcome**
	Not realistic • different bacterial genome architectures • long essential plasmids (chromids) • dynamic • contigs that can be part of both chromosomes and plasmids	Not realistic ... • Not representative of microbial diversity • No contaminants • No intermediate replicons • No contigs of unknown origin • No error in simulated assembly	Overfits by design • Overfits on similar data • Trains to noise • Requires non-duplicate data • Requires a lot of tuning	Models: • Look good... • But very Overfit • do not pass first round of internal validation
Explainable	**Requirement**	**Requirement**	**Requirement**	**Requirement**
	• Not too simple / not too complex	• Relevant features • Reproducible	• Understandable • Transparent • Good documentation	Explainable • for expert, developer and user • not too complex • transparent
	Outcome	**Outcome**	**Outcome**	**Outcome**
	• too simple	• Kmers too wide, too low signal • Kmers need further external validation	Not developer-friendly • misleading documentation • requires going to source code at every step to see what it does	Not very explainable • plasmid genes best feature • kmers drown out signal • identical features not consistently same importance
Usable	**Requirement**	**Requirement**	**Requirement**	**Requirement**
	• Good fit to reality and question	• Good fit to reality, question and algorithm • Confidence in reliability • Not too big/small • Statistically sound	Developer-friendly • Good implementation • Not too big / slow / too many dependencies • reliable, production grade code	User-friendly • not requiring super computer • reliable • safe to use • fit for purpose
	Outcome	**Outcome**	**Outcome**	**Outcome**
	• useful as toy model / first step • not for real-life applications	Statistical Flaws • Noisy & Wide • Low Signal-to-Noise Ratio • many Duplicate Features (columns) • many Duplicate Contigs (rows) • imbalanced	Not developer-friendly • buggy • requires separate package for resampling • not production-grade • not as advertised (not plug-and-play)	Not user-friendly • "best models" so big, they are unusable • not generalizable • unreliable predictions • not for real-life applications • potentially dangerous to make recommendations
Conclusions	Dataset appears to not fit the question, the biology, or the algorithm. Models are overfit to the dataset and not generalizable, hence not fit for real life epidemiological purpose. Machine learning might not be the best approach for plasmid detection.			