KAGGLE COMPETITION TEAM MIDTERM_FAILURE

Mateo Mangialomini
2483752
mateo.mangialomini@polymtl.ca

Maxime Tran
2488206
maxime-anh-duy.tran@etud.polymtl.ca

Adrien Marie Saouma
2486687
saoumadrien@gmail.com

CONTENTS

1 INTRODUCTION

Antimicrobial resistance (AMR) is recognized by the World Health Organization as one of the top ten global public health threats. The rapid emergence of resistant bacterial strains, particularly in *Escherichia coli,* compromises the effectiveness of standard antibiotic treatments. While whole-genome sequencing has become routine, translating raw genomic data into accurate predictions of phenotypic resistance remains a major bioinformatics challenge.

This Kaggle competition provides a real-world classification task: predict whether an *E. coli* isolate is resistant or susceptible to a specific antibiotic using pre-computed k-mer count matrices derived from its genome, supplemented by rich metadata (geographical origin, isolation source, collection date, etc.). The evaluation metric is the macro-averaged F1-score, which appropriately handles the expected class imbalance.

In this work, we propose a robust model combining TF–IDF transformation of high-dimensional k-mer profiles, variance-based and SVD feature selection, integration of metadata features, and an ensemble of Xgboost, Catboost, LightGBM, Random Forest and regularized logistic regression models.

2 FEATURE SELECTION

The training set is composed of 1939 samples with 1 million features. In order to avoid the curse of dimensionality and to reduce computational cost, we processed our data. Also, we believed the metadata to be of use for AMR detection.

2.1 VARIANCE

First we decided to reduce the number of k-mers to 100 000 using variance selection. We believe that k-mers with low variance do not have an important impact for the classification. On one hand, k-mers present or absent in most genomes have low variance and don't give information regarding antibiotic mutations. On the other hand, k-mers linked to resistance mechanisms show higher variance.

Fig.1 presents histograms comparing the differences in means and standard deviations for the 10 000 highest variance features and the 10 000 lowest variance features. The high variance features show a positive skew in mean differences (up to 8), indicating significant shifts in average values across classes. In contrast, low variance k-mers cluster around zero, showing low discriminative power. The standard deviation differences plot shows a similar trend with high variance features displaying more class specific variability than the low variance. Therefore, this analysis confirms that variance filtering is a great way of retaining the most informations while greatly reducing the number of features.

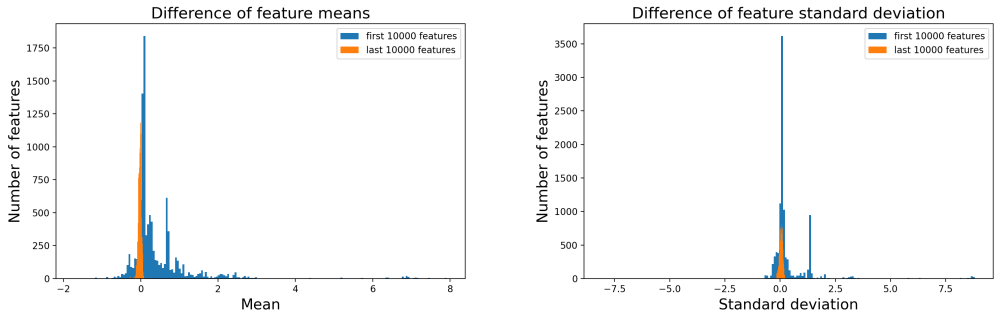

Figure 1: Difference histograms between Class 1 and Class 0 (Means and Standard deviations)

2.2 TF-IDF TRANSFORMATION

To reduce the impact of genome length variation and to emphasize k-mers that truly matter for the prediction, we applied a TF–IDF (Term Frequency–Inverse Document Frequency) normalization. In this formulation, each genome is treated as a "document" and each k-mer as a "term" within a shared vocabulary. Let's first describe how this normalization works and then its impact on the model.

For k-mer j, let c_{ij} denote the count of that k-mer in genome i.The term frequency is given by

$$\mathrm{tf}_{ij} = \frac{c_{ij}}{\mathrm{row\_sum}_i}.$$

This converts absolute counts into relative frequencies, preventing longer genomes from dominating simply due to larger total k-mer counts.

Let N be the number of genomes in the training set and df_j the number of genomes containing k-mer j. The IDF is given by

$$\mathrm{idf}_j = \log\left(\frac{1+N}{1+\mathrm{df}_j}\right) + 1.$$

This downweights k-mers that are widespread across genomes and therefore less informative.

The final TF–IDF is given by

$$\mathrm{tfidf}_{ij} = \mathrm{tf}_{ij} \cdot \mathrm{idf}_j,$$

which emphasizes k-mers that are frequent within a specific genome but rare across all. The point was to give lower weigths to k-mers that are frequently present in the genomes as they probably represent core sequences that don't trigger resistance when mutated and to give higher weights to those who are scarer accross all genomes but still frequent in some genomes.

In order to support the use of this normalization, we did an ablation study: our model was trained once with TF–IDF and another without it. The results are displayed in Table.1

TF–IDF	Public F1 score	Private F1 score
Excluded	0.76222	0.72490
Included	0.78890	0.72093

Table 1: Ablation Study of TF–IDF.

One can observe that the F1 score increased by approximately 3.5% on the public test using the TF–IDF, showing its importance. However, on the private set, it showed a slight decrease of 0.55%. A hypothesis would be that the transformation correctly highlighted the discriminative k-mers in the training data but might have overspecialized on it. Indeed, the test set having a phylogenetic split, introduced genomes that are not considered in the training set. While thinking of this transformation, we focused too much on the ablation results and did not think it would impact the robustness of our model. Hopefully, robustness was correctly covered by our ensemble model (Section.3). Moreover, we employed the default hyperparameters in our implementations. To increase performance, it would have been interesting to tune the following: sublinear TF scaling, IDF smoothing, and vector normalization.

2.3 TRUNCATEDSVD

Even after variance filtering and TF–IDF, the dataset still contains far more features (100,000) than samples (1,939). Many of these k-mers are also correlated, which increases the feature space without adding much useful information and can affect how the model learns.

TruncatedSVD helps to solve these issues by reducing the dimensionality of the TF–IDF matrix. Unlike PCA, it does not center the data before computing the decomposition, which makes it suitable for high-dimensional matrices like ours.

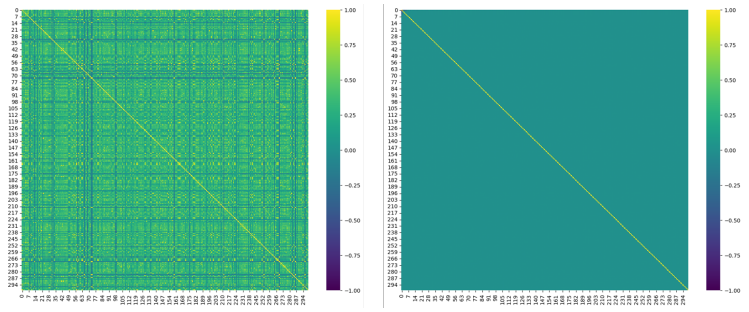

Figure 2: Correlation matrices of a subset of 300 features from the TF–IDF matrix before the TruncatedSVD transformation (left), and after (right). As the features are different, so are the subsets

The left plot of Figure 2 shows the correlation structure in a random subset of 300 k-mers. One can observe that several groups of features are strongly correlated, illustrating the redundancy present in the original matrix.

We keep 1,000 components after applying TruncatedSVD. These components are orthogonal, meaning they are uncorrelated and capture distinct directions of variation in the data. This can be seen on the right plot of Figure 2. This process reduces redundancy and gives the model a more compact representation to work with.

2.4 METADATA

The provided metadata contains six categorical fields (Organism group, Isolation type, Location, Isolation source, Laboratory typing platform, Testing standard) and a collection timestamp (Create date). Missing categorical values were explicitly encoded as a dedicated "Unknown" category to avoid information loss. The creation date was converted to a numerical feature "Days Since Ref Date" using 2010-01-01 as reference (predating all samples), then standardized. All categorical variables were one-hot encoded.
We believe that this data is truly important and can have an effect on the model. As a matter of fact, it is known that different regions expose the bacterias to different environmental factors such as temperature, UV radiations and mutagen. Depending on these factors, mutations may be more frequent and thus impact the AMR.

After all this data processing, we are left with a matrix of 1939 samples and 1105 features.

3 ENSEMBLE METHOD

We chose an ensemblist model for this competition as we were primarly focusing on robustness. We want to combine the advantages of each of the model we trained in the hope of filling the gaps left by any model.

3.1 MODELS

3.1.1 MODELS CONSIDERATIONS

While developing our model, we focused on robustness primarily. Our first model performed well on the public set (XGBoost: 0.80771). However, we knew it would not perform as well on the private set because of the phylogenetic split. Some patterns appearing in the training data do not show up in the private set and vice versa. This assumption turned out to be correct, as our first model overfitted the training data and performed poorly on the private set (0.69467). This is one of the reasons why we integrated logistic regression on top of the gradient boosting models (See Section 3.1.2). Combining different types of models and averaging their predictions (see Section 3.2) helped us get a more balanced and robust final prediction. We also relied on strong regularization to further reduce overfitting.

Another important point was the class imbalance. The training set contains 1666 samples of class 0 and only 273 of class 1, so the ratio is more than 6 to 1. To deal with this, we used cross-validation with stratification to ensure each folds maintains the same proportion of samples for each class as the original full dataset. Moreover, we implemented class weights in the models in order to assign greater weights to the minority class and lower to the other. Interestingly, the competition metric is the F1 score instead of accuracy, most likely because of this imbalance. As a matter of fact, with such a skewed dataset a model can reach a high accuracy by mostly predicting the majority class. This is what we observed during cross-validation: accuracies around 0.87, which do not reflect the difficulty of predicting the resistant class.

3.1.2 MODELS CHOICE

XGBoost, CatBoost & LighGBM XGBoost, CatBoost and LightGBM are amongst the most widely used gradient boosting frameworks in Machine Learning and have achieved excellent performances in past competitions. They excel in handling sparse, high-dimensional data, and offer built-in regularization, and robust handling of imbalanced datasets, which makes them well-suited for capturing the nonlinear interactions among the k-mers.

Random Forest Random Forest (from sklearn) is an ensemble of decision trees that provides robustness to our model.

Logistic Regression Logistic Regression works well as a secondary voter as it handles high dimensional data well. Also using regularization is good in order to avoid overfitting. Moreover, with balanced class weights, it addresses imbalance between resistant and susceptible samples, penalizing errors on the minority class more heavily. As a linear model, it adds diversity to XGBoost's non-linear trees, reducing correlated errors in the soft voting setup.Therefore its main importance is to increase the robustness of the model by adding some stability.

3.2 MODEL VOTE

The goal of having multiple models is to leverage their complementary strengths and improve the overall predictions to get the best results. Rather than relying on a simple majority vote, we chose to combine the predictions of our models through a weighted average because our first submissions showed that boosting frameworks (XGBoost, LightGBM, CatBoost) could independtly achieve good performance on the public leaderboard, so the primary purpose of including additional models such as Random Forest and Logistic Regression was to enhance prediction in the cases where boosting was failing.
To determine the optimal contribution of each model to the ensemble, we performed an optimization using cross-validated out-of-fold predictions using 4 folds. We use the predictions on the validations folds to estimate the model performance for each fold and using scipy.optimize we compute the optimal weights for each model. Quite surprisingly XGBoost seems to be completely outperformed by LightBGM and CatBoost (see Table 2) which use slightly different boosting methods and optimizations. Random Forest is probably too simple in comparison to boosting methods so it doesn't capture anything meaningful. On the other hand the Logistic Regression clearly fulfills its role and, although it is assigned a small weight, it performs as we expected, it provides a small improvement where boosting is less performant.
Our initial approach had been to manually set the weights to give a higher importance to XGBoost because we had obtained a good F1 score on the public leaderboard in our first submissions with it. The results in Table 2 clearly show that this method is completely wrong because we would have added a bias on the meta-classifier that would have lowered its performance.

3.3 RANDOM SEARCH

For the hyperparameter optimization, we chose to use random search rather than the exaustive grid search because we wanted to explore the hyperparameter space as much as possible without having to try a lot of values for the hyperparameters that had a small impact on the accuracy.

Model	Weight
XGBoost	0.0000
LightGBM	0.4692
CatBoost	0.4560
Random Forest	0.0000
Logistic Regression	0.0747

Table 2: Weights assigned to each model in the ensemble based on cross-validated F1 scores.

4 RESULTS

4.1 HYPERPARAMETER TUNING

Hyperparameter optimization was performed independently for five candidate models using `RandomizedSearchCV` (50 trials, 4-fold stratified cross-validation). The search spaces deliberately spanned both high-capacity and strongly regularized configurations.

Table 3 reports the best configurations identified for each model.

Parameter	LightGBM	CatBoost	XGBoost	Random Forest	LogReg
learning rate / depth	0.0213 / 6	0.164 / 3	0.0133 / 5	– / 5	–
n estimators / iterations	291	246	247	282	–
num leaves / –	79	–	–	–	–
subsample / bagging temp.	0.810	0.577	0.876	–	–
colsample bytree / border count	0.860	223	0.836	–	–
min child weight / l2 leaf reg	10.39	6.72	3	–	–
reg alpha	0.183	–	0.645	–	–
reg lambda / gamma	0.608	–	0.349 / 0.203	–	–
max features / bootstrap	–	–	–	sqrt / True	–
C / l1 ratio / penalty	–	–	–	–	6.83 / 0.61 / elasticnet

Table 3: Best hyperparameter configurations identified by `RandomizedSearchCV` for the five models evaluated. XGBoost and Random Forest achieved significantly lower cross-validation F1 scores and were assigned zero weight in the final ensemble.

The search confirmed the dominance of LightGBM and CatBoost, both converging toward low learning rates and strong regularization. The optimal CatBoost configuration is remarkably shallow (depth 3), indicating that the TF-IDF + SVD representation is already highly discriminative. Logistic regression consistently favored ElasticNet regularization with an L1 ratio of approximately 0.61, confirming that sparse feature selection remains beneficial after dimensionality reduction. Despite extensive tuning, XGBoost and Random Forest lagged behind in internal validation and contributed nothing to the final prediction. The retained models were retrained on the full training set using the hyperparameters above before weighted ensemble construction.

4.2 SCORE EVOLUTION

Figure 3 shows the macro F1 scores on the public and private leaderboards across our eleven submissions (chronological order). During the competition, only public scores were available. Our first submissions, based on a single XGBoost trained on variance-selected k-mers and metadata, quickly exceeded 0.80 on the public leaderboard. However, we strongly suspected severe overfitting: a model that learns spurious correlations present in the random split would inevitably fail on phylogenetically distant isolates. Rather than continuing to optimize for the public score, we deliberately chose to sacrifice it when necessary to improve robustness.

We therefore systematically favored stronger regularization, model diversity (introduction of LightGBM and CatBoost), and conservative blending, even when these changes temporarily lowered the public score, sometimes by more than 4 points. One particularly telling decision was to test an extremely regularized version without TF-IDF: despite a public score of only 0.761, we submitted it because we believed it would reduce specialization to the training distribution. The final phase consisted of letting an automated weighting on out-of-fold predictions decide the ensemble composition:

XGBoost and Random Forest were naturally assigned zero weight, while LightGBM, CatBoost and a small contribution from logistic regression formed the final model.

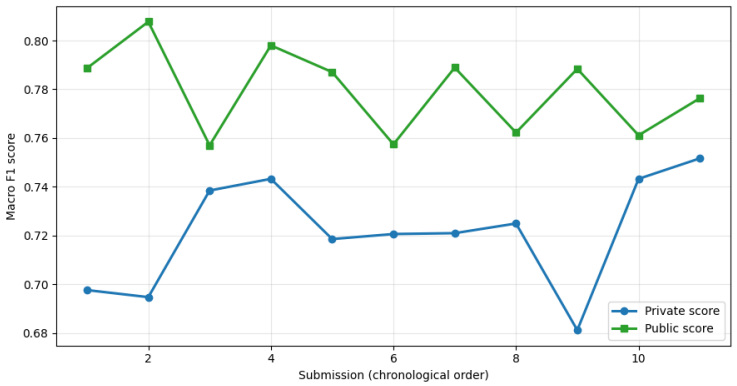

Figure 3: Evolution of public (green) and private (blue) macro F1 scores over the eleven submissions. Only public scores were visible during the competition.

The private leaderboard, revealed only at the end, fully validated this strategy: our final submission achieved 0.776 publicly and 0.752 privately, a gain of more than 5.5 points on the phylogenetic split compared to our initial baseline, with the public/private gap reduced from over 0.11 to 0.024. This confirms that our constant focus on robustness, even at the cost of short-term public performance, was the right approach for this phylogenetically challenging task.

5 Discussion

The results of the private leaderboard clearly indicates the generalization capabilities of our models on unseen bacterias from a different genetic evolutionary branch. More precisely, our selected and last prediction achieved 77.632% score on the public leaderboard and 75.163% on the private one. The main area of improvement of our model would be to include more models in the ensemblist meta-model. Our choice to use Random Forest while also using Boosting methods is redundant and we should have focused on models radically different that could have modelled the data differently, such as SVM, Neural Networks or Linear Regression. For the preprocessing pipeline, we could have explored different feature selection algorithms and methods such as Boruta, Mutual information, Chi-square which could have yielded some limited improvement.

6 Statement of contribution

We hereby state that all the work presented in this report is that of the authors. Mateo mainly focused on the metadata and data loading and exploration, and the running of the code. Adrien engineered our feature selection and Maxime created the meta-model. We all collaborated on the models choices and tuning.