MACHINE LEARNING PROJECT REPORT

Medication Recommendation System

GITHUB LINK: HTTPS://GITHUB.COM/ANASTACIA04/RECOMMENDED_MEDICATION

Imane-Nourah KONE
Anastacia BOIS
Chloé LE

December 10, 2025

Contents

1 Introduction **3**
1.1 Context 3
1.2 Description of the Dataset 3
1.3 Discriminant Features 3

2 Data Exploration and Preprocessing **4**
2.1 Overview of the Data 4
2.2 Handling Missing Values 4
2.3 Transforming categorical data 4
2.4 Key Insights from Visual Exploration 5

3 Formalization of the Problem **6**

4 Addressing Modeling Challenges **6**

5 Presentation of the Models **7**
5.1 Test 1 7
5.2 Test 2 8
5.3 Test 3 9
5.4 Pycaret classification 10

6 Comparison of Results **10**

7 Conclusion **12**

8 References **13**

1 Introduction

1.1 Context

In modern healthcare, prescribing medication requires clinicians to analyze multiple factors simultaneously, such as symptoms, allergies, chronic conditions, demographic information, and medical history.Choosing an inappropriate treatment can lead to adverse reactions or reduced effectiveness.The aim of this project is to develop a machine-learning system capable of recommending the most suitable medication based on patient characteristics. This task is formulated as a multiclass classification problem with four possible outcomes: amlodipine, ibuprofen, amoxicillin, or no medication.

1.2 Description of the Dataset

The dataset, sourced from the Personalized Medication Dataset on Kaggle, contains 1,000 patients described by 17 features. It includes demographic information (age, gender, weight, height, BMI), medical history (chronic conditions, allergies, genetic disorders), and detailed symptom descriptions.

```
data.head()
```

Age	Gender	Weight_kg	Height_cm	BMI	Chronic_Conditions	Drug_Allergies	Genetic_Disorders	Diagnosis	Symptoms	Recommended_Medication
78	Other	88.7	196.3	21.1	NaN	Penicillin	Cystic Fibrosis	Inflammation	Fever	Amlodipine
57	Female	90.5	195.6	30.2	Hypertension	NaN	Cystic Fibrosis	Depression	Fatigue, Headache, Dizziness	Amoxicillin
29	Female	87.0	168.2	27.0	NaN	Sulfa	NaN	Inflammation	Joint Pain, Headache, Nausea	NaN
56	Female	81.4	188.9	26.9	Hypertension	Penicillin	Cystic Fibrosis	Infection	Joint Pain	Ibuprofen
90	Male	64.2	157.0	33.3	NaN	Sulfa	Sickle Cell Anemia	Inflammation	Fatigue, Fever, Headache	Amlodipine

Figure 1: Preview of the Dataset

1.3 Discriminant Features

A discriminant feature is a variable whose values help distinguish between classes. Such features exhibit clear differences across target categories and contribute significantly to accurate classification. In the context of our project, however, most patient characteristics did not behave as discriminant features. Medication choices appeared almost uniformly distributed across symptoms, diagnoses, and medical histories, meaning that these features lacked the necessary predictive power. This absence of discriminant information suggested that the prediction task would be challenging.

2 Data Exploration and Preprocessing

2.1 Overview of the Data

The exploratory data analysis (EDA) revealed well-distributed numerical variables and diverse categorical attributes. Symptom descriptions, initially recorded as comma-separated text, were transformed into binary indicators to better capture frequency and co-occurrence. This preprocessing improved structure without altering the underlying meaning.

2.2 Handling Missing Values

Although the dataset appeared complete in its raw form, several entries were converted into `NaN` values during the DataFrame import process. Importantly, these `NaN` values did not correspond to missing information. For the columns *Chronic_Conditions*, *Drug_Allergies*, and *Genetic_Disorders*, the `NaN` entries reflected patients who simply had zero conditions, allergies, or disorders. They were therefore replaced with the string `None` to restore the intended meaning. Similarly, in the *Recommended_Medication* column, `NaN` values were replaced by `Nothing`, as these patients were originally labeled as receiving no medication. This correction did not involve interpretation but merely reinstated the dataset's true structure.

2.3 Transforming categorical data

In this project, the dataset contains different types of categorical variables, which required different encoding strategies. Simple categorical attributes such as *gender*, *chronic conditions*, or *genetic disorders* take only a few discrete values with no internal structure. For these variables, Label Encoding is appropriate, as it provides a compact numerical representation without altering their meaning. Since the number of categories is small and well-defined, Label Encoding does not introduce artificial relationships that could mislead the model.

However, other fields in the dataset, most notably the *Symptoms* column, consist of free-text descriptions containing multiple medical terms. These entries cannot be meaningfully transformed using Label Encoding, which would assign arbitrary numbers to entire text strings and ignore the internal semantic content. For this reason, we used TF–IDF (Term Frequency–Inverse Document Frequency) for the symptoms data. TF–IDF captures the relative importance of each symptom across the dataset, producing a numerical representation that highlights informative terms while down-weighting very common or uninformative ones.

Using both encoding methods allowed us to treat each variable in a way that preserves its meaning and maximizes its contribution to the predictive model: Label Encoding for structured categorical features, and TF–IDF for unstructured text-based medical descriptions.

2.4 Key Insights from Visual Exploration

Visualization helped identify important patterns. Numerical features showed plausible distributions, and categorical variables varied widely among patients. The most notable finding was that medication choices remained almost uniformly distributed across all patient groups. Since no feature strongly correlated with the target class, the dataset lacked discriminant power, explaining the low accuracy observed in early modeling attempts.

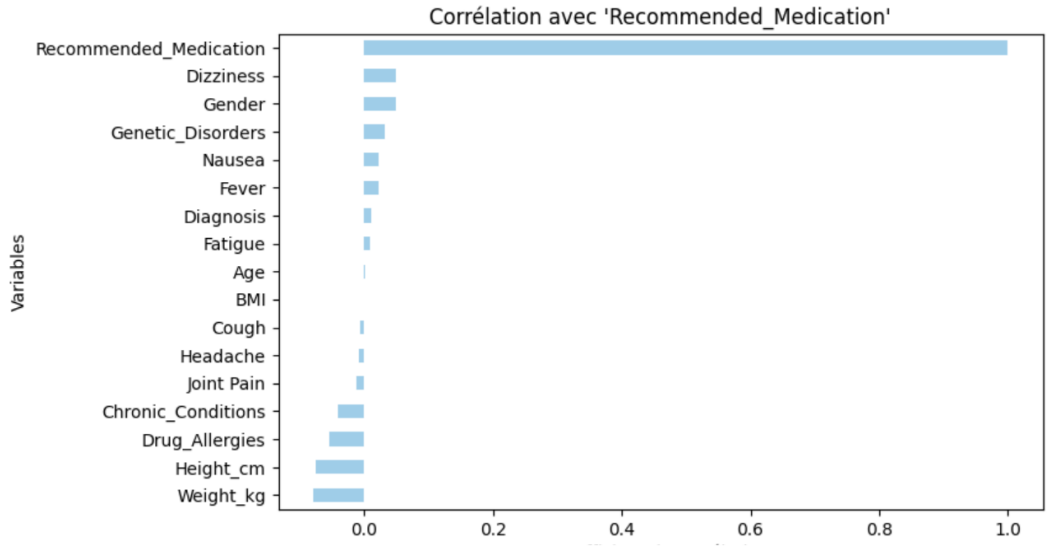

Figure 2: Features correlation with recommended medication

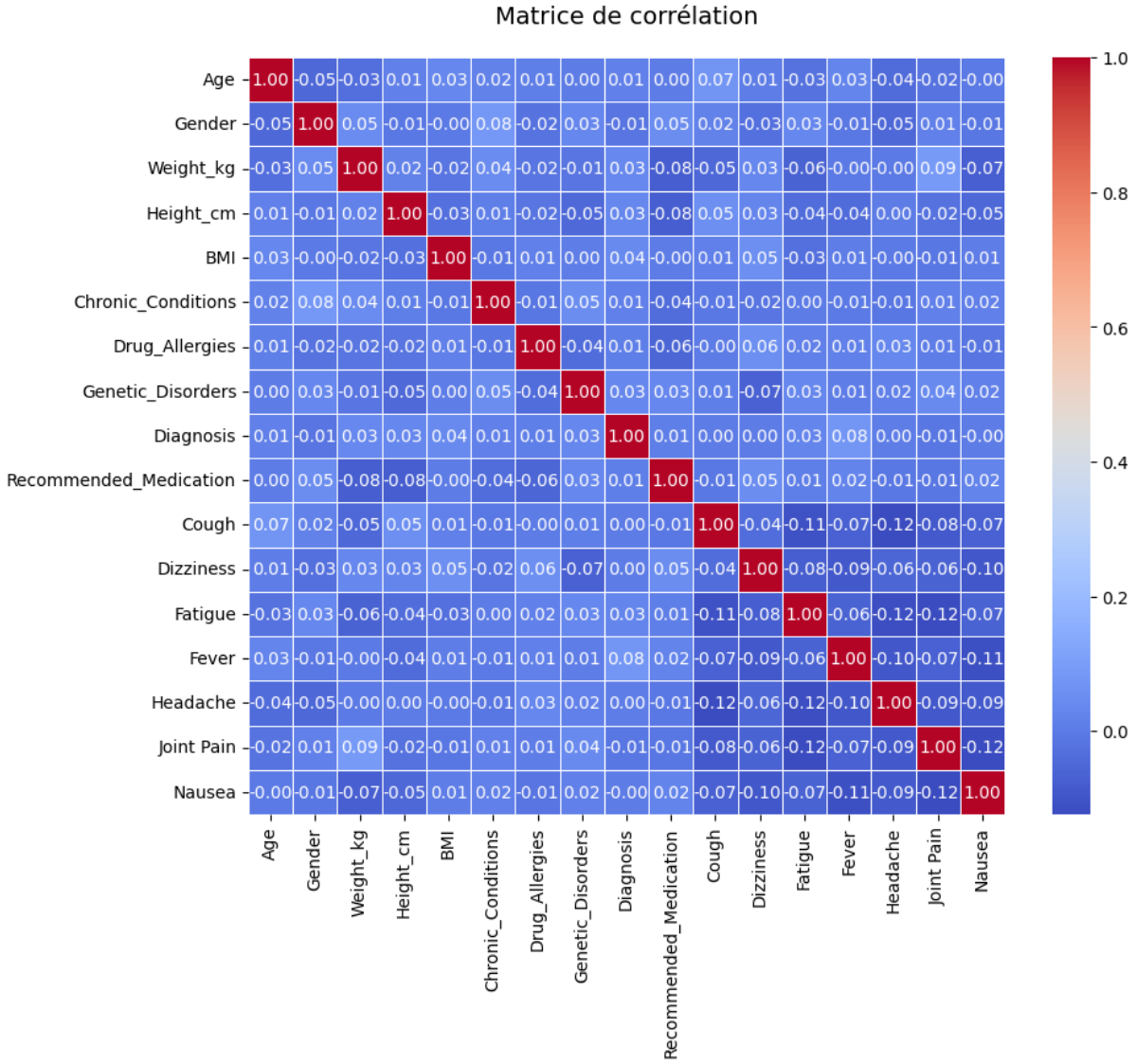

Figure 3: Correlation matrix

3 Formalization of the Problem

The task is formulated as a supervised multiclass classification problem. Given a patient described by a feature vector X, the goal is to predict the recommended medication y among the four available classes. The objective is to learn a decision function capable of generalizing to new patient cases while providing interpretable and medically meaningful outputs.

4 Addressing Modeling Challenges

A major challenge was the low predictive accuracy observed during both training and validation. Further analysis confirmed that the four medications appeared in nearly equal proportions across similar patient profiles. Without discriminant features, the

models could not learn clear decision boundaries, inherently limiting accuracy.
Due to this structural limitation in the data, we conducted multiple experimental variations to determine whether modifying specific preprocessing or modeling components could improve performance. This led to the development of three separate testing pipelines,*Test 1*, *Test 2*, and *Test 3*, each designed to evaluate different aspects of the workflow. In the following sections, we describe the particularities and motivations behind each of these tests.

5 Presentation of the Models

5.1 Test 1

Models We first removed the `diagnosis` column because it could leak information directly related to the target. Then, we separated the dataset into features (X) and the target variable (y). Using `train_test_split`, we created a training set and a test set while keeping the class proportions balanced thanks to `stratify = y`. Next, we standardized all numerical features with `StandardScaler` to ensure that the model receives scaled inputs. We then trained a Random Forest classifier with 500 trees and no limit on tree depth. After fitting the model on the training data, we predicted the medication on the test set and evaluated the model's performance. The accuracy obtained (0.26) indicates that the model correctly predicted 26% of the test samples.

Thus, we tried another model and performed hyperparameter tuning on a Decision Tree using `GridSearchCV`. First, we defined a set of possible values for several Decision Tree parameters (such as the criterion, max depth, minimum samples to split, and maximum number of features to consider). `GridSearchCV` then trained multiple models, each using a different combination of these parameters, and evaluated them using 5-fold cross-validation. After testing all combinations, it selected the model with the best accuracy. Finally, the best model was used to make predictions on the test set, and we obtained a test accuracy of 0.21 and a cross-validation accuracy of 0.267.

We performed hyperparameter tuning on a Random Forest using GridSearchCV. We defined a grid of possible values for important parameters such as the number of trees, maximum depth, minimum samples per split and per leaf, the number of features considered at each split, and whether bootstrap sampling is used. GridSearchCV trained and evaluated all parameter combinations using 5-fold cross-validation to identify the best model. The search returned the configuration with the best cross-validation accuracy (0.25). You then evaluated this optimized model on the test set, obtaining a test accuracy of 0.24, meaning that even after tuning, the Random Forest only slightly improved and still struggled to generalize well on this dataset.

Then we trained an XGBoost classifier configured for multiclass classification using the softmax objective. We specified several key hyperparameters: the learning rate (0.1), tree depth (6), number of boosting rounds (300), and subsampling ratios for rows and features to reduce overfitting. After fitting the model on the training data, we used it to predict the medication on the test set. Finally, we calculated the test accuracy, which reached 0.205, indicating that XGBoost performed worse than the other models on this

dataset despite its advanced boosting strategy.

We applied `GridSearchCV` to optimize the XGBoost classifier by testing different values for key hyperparameters, including the learning rate, tree depth, number of boosting rounds, and subsampling ratios. `GridSearchCV` trained multiple XGBoost models using 5-fold cross-validation, evaluating each combination based on accuracy. After exploring the full parameter grid, it selected the configuration that achieved the best cross-validation accuracy (0.2575). We then used this best model to make predictions on the test set, obtaining a test accuracy of 0.25. This shows that hyperparameter tuning improved XGBoost slightly compared to the default version, but its performance remained limited on this dataset.

Next, we used bagging: we split the dataset into training and test sets while preserving class proportions. After scaling the numerical features, we trained a `BaggingClassifier` using 50 Decision Trees as base estimators. Bagging works by training each tree on a different random subset of the data, which helps reduce variance and improve robustness. After fitting the ensemble on the training data, we used it to predict the medication on the test set and computed the accuracy. The model reached an accuracy of 0.225, showing that bagging provided slightly better performance than a single decision tree but remained limited on this dataset, linked to the complexity of the dataset.

5.2 Test 2

Preprocessing. The preprocessing stage transforms the heterogeneous clinical dataset into a coherent numerical representation suitable for machine learning. The feature matrix contains 1,000 samples and six explanatory variables (Shape $X = (1000, 6)$), including four numerical features (Age, Weight_kg, Height_cm, BMI), one categorical feature (Gender), and a combined textual clinical description. Missing values in both the target variable and the medical text fields are replaced by the string "None" to avoid vectorization errors. The textual descriptors (Symptoms, Diagnosis, Chronic_Conditions, Drug_Allergies, Genetic_Disorders) are concatenated into a single feature, *text_medical*, which encodes the full medical context of each patient. The dataset is then split into a training set of 800 samples and a test set of 200 samples $(800, 6) \mid (200, 6)$, with class stratification preserved across the four target labels (Amlodipine, Amoxicillin, Ibuprofen, None). A multi-modal preprocessing pipeline is constructed using a *ColumnTransformer*: numerical variables are standardized with *StandardScaler*, the categorical feature is encoded with a *OneHotEncoder*, and the clinical text is transformed into a high-dimensional representation using TF–IDF (up to 1000 features, with unigram and bigram extraction). This unified pipeline ensures consistent preprocessing for any model trained subsequently.

Models Two supervised learning models are trained using the preprocessing pipeline: a Decision Tree and a Support Vector Machine (SVM). For each model, a grid search with 3-fold cross-validation is performed, optimizing for the macro F1-score to address the four-class structure of the target variable. The Decision Tree evaluates 72 hyperparameter configurations and achieves its best performance using entropy as the splitting criterion, unlimited depth, and a minimum of two samples per leaf. However, its results

remain limited, with an accuracy of 0.28 and a macro F1-score of 0.2797 on the test set of 200 samples. The confusion matrix indicates substantial misclassification across all medication categories. In contrast, the SVM tested over 12 hyperparameter combinations obtains its optimal configuration with a linear kernel and $C = 10$, resulting in superior predictive performance. It reaches an accuracy of 0.325 and a macro F1-score of 0.3255, outperforming the Decision Tree. These results highlight the advantage of SVMs when dealing with high-dimensional, sparse TF–IDF representations, whereas Decision Trees struggle to capture such complex text-based patterns.

5.3 Test 3

Bagging-based ensembles. In this optimization phase, we construct ensemble models based on the best individual classifiers previously identified, namely the Decision Tree and the linear SVM. The optimal hyperparameters found in Part 2 are reused to define two strong base learners: a Decision Tree with entropy as the splitting criterion, unlimited depth and a minimum of two samples per leaf, and an SVM with a linear kernel and $C = 10$. These base estimators are then wrapped inside *BaggingClassifier* objects, each embedded in a full pipeline that first applies the same multimodal preprocessing as before (standardization of numerical features, one-hot encoding of the gender variable, and TF–IDF vectorization of the concatenated medical text), followed by the bagging ensemble. For both the Decision Tree and the SVM, a grid search with 3-fold cross-validation is performed over the number of estimators, the proportion of samples used in each bootstrap (`max_samples`), and the use of bootstrap sampling itself. The best bagging configuration for each base learner is selected according to the macro F1-score. In addition to predictive performance, the code also estimates the complexity of the resulting ensembles by inspecting the individual trees' depths and leaf counts for the bagged Decision Tree, and the average number of support vectors for the bagged SVM, providing an approximate notion of model size.

Hybrid voting ensemble and evaluation. After tuning the two bagging ensembles, a hybrid model is constructed using a *VotingClassifier* that combines the bagged Decision Tree and the bagged SVM in a soft-voting scheme, i.e., by averaging their predicted class probabilities. All three models Bagging Decision Tree, Bagging SVM, and the hybrid Voting ensemble are then evaluated on the same held-out test set. For each model, several metrics are computed: accuracy, macro F1-score, macro precision, macro recall, as well as the confusion matrix, which provides a detailed view of the distribution of correct and incorrect predictions across the four medication classes. The inference time on the test set is also measured to quantify the computational cost of each approach. The results are visualized through confusion matrix heatmaps and a bar chart comparing test accuracies, and are finally summarized in a comprehensive table that reports performance metrics, approximate model size, inference time, and a qualitative assessment of interpretability. This analysis allows us to assess whether the additional complexity introduced by bagging and hybrid voting actually leads to a significant improvement over the best single SVM model obtained in Part 2.

5.4 Pycaret classification

After experimenting with our three custom models and consistently observing low predictive accuracy, we decided to use the PyCaret Python package for classification. PyCaret provides an easy way to compare the performance of a wide range of machine learning models on the same dataset with minimal setup. Running the dataset through PyCaret confirmed our previous results: all models performed poorly, reinforcing the conclusion that the low accuracy is due to the lack of discriminant features in the dataset rather than the choice of a particular model or algorithm.

	Model	Accuracy	AUC	Recall	Prec.	F1
svm	SVM - Linear Kernel	0.2900	0.0000	0.2900	0.2881	0.2820
lda	Linear Discriminant Analysis	0.2886	0.0000	0.2886	0.2872	0.2836
ridge	Ridge Classifier	0.2857	0.0000	0.2857	0.2828	0.2750
lr	Logistic Regression	0.2843	0.0000	0.2843	0.2912	0.2839
qda	Quadratic Discriminant Analysis	0.2814	0.0000	0.2814	0.2770	0.2627
nb	Naive Bayes	0.2771	0.5150	0.2771	0.2801	0.2748
gbc	Gradient Boosting Classifier	0.2714	0.0000	0.2714	0.2691	0.2658
dt	Decision Tree Classifier	0.2671	0.5110	0.2671	0.2653	0.2629
ada	Ada Boost Classifier	0.2657	0.0000	0.2657	0.2709	0.2637
lightgbm	Light Gradient Boosting Machine	0.2500	0.5088	0.2500	0.2466	0.2439
rf	Random Forest Classifier	0.2486	0.5086	0.2486	0.2491	0.2439
dummy	Dummy Classifier	0.2443	0.5000	0.2443	0.0597	0.0959
et	Extra Trees Classifier	0.2357	0.4988	0.2357	0.2330	0.2309
knn	K Neighbors Classifier	0.2286	0.4885	0.2286	0.2335	0.2250

Figure 4: Models comparison

6 Comparison of Results

This section presents a detailed comparison of the ensemble methods evaluated in Test 3, namely Bagging Decision Trees, Bagging SVMs, and a hybrid Voting Classifier. Their performance is contrasted with the single SVM model from Part 2, which serves as the strongest baseline with an accuracy of 0.325 and a macro F1-score of 0.325. The

objective is to assess whether ensemble learning provides meaningful improvements in predictive performance for this multiclass medical recommendation problem.

Bagging Decision Trees The Bagging Decision Tree achieved an accuracy of 0.25 and a macro F1-score of 0.248, which is lower than the single Decision Tree from Part 2. Although bagging typically stabilizes high-variance models, the ensemble of ten deep trees (average depth ≈ 25) did not significantly enhance performance. The confusion matrix shows substantial misclassification across the four medication classes, suggesting that the Decision Tree structure struggles to capture patterns in the high-dimensional TF–IDF text features. While model interpretability remains moderate, predictive performance remains limited.

Bagging SVM The Bagging SVM obtained the weakest results, with an accuracy of 0.185 and a macro F1-score of 0.153. This considerable performance drop compared to the single SVM (accuracy 0.325) can be explained by the loss of information caused by bootstrap sampling. Each SVM in the ensemble is trained on only 70% of the available data, which is detrimental in a setting with sparse TF–IDF features and only 1000 samples. As SVMs are already low-variance, stable models, bagging provides no benefit and instead degrades performance. Interpretability is also very low due to the accumulation of support vectors across estimators.

Voting Classifier The hybrid Voting Classifier, combining Bagging Decision Trees and Bagging SVMs, achieved the best ensemble performance with an accuracy of 0.265 and a macro F1-score of 0.262. Although this represents a slight improvement over Bagging DT, it remains inferior to the single SVM of Part 2. Soft voting offers more stable predictions by averaging probabilistic outputs, but since both constituent models are individually weak in this setting, the ensemble inherits their limitations. Interpretability is the lowest among the tested models due to the compounded complexity of combining two ensembles.

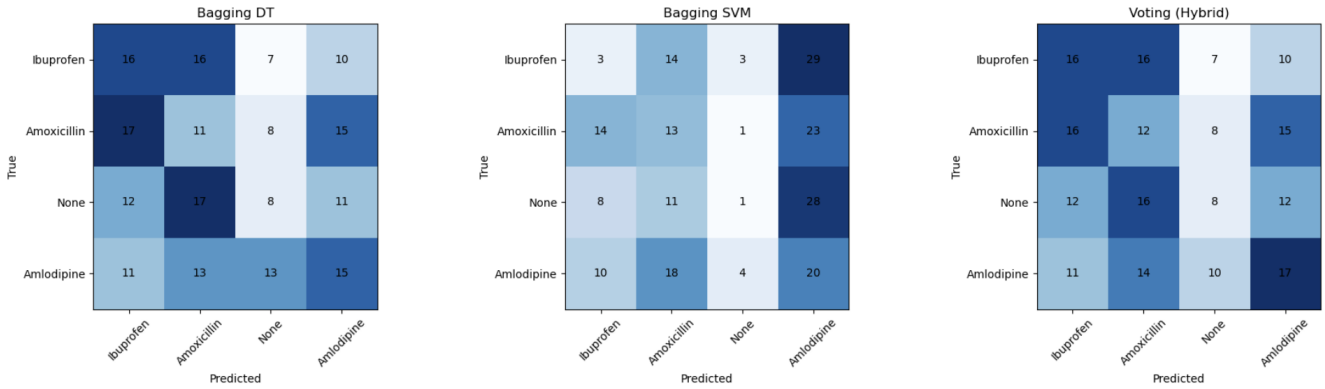

Figure 5: Confusion matrices for the Bagging DT, Bagging SVM, and Voting Classifier models in Test 3.

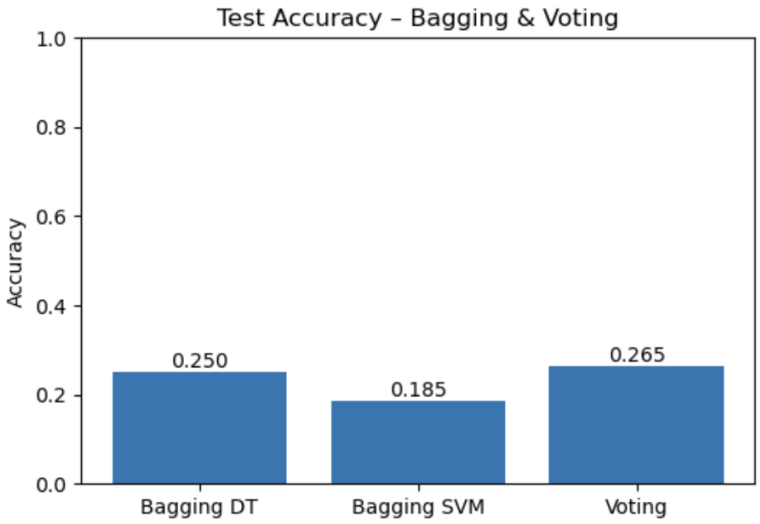

Figure 6: Comparison of test accuracy across ensemble models in Test 3.

As illustrated in Figures 5 and 6, none of the ensemble models outperform the single SVM from Part 2. This suggests that, for this dataset, where text features dominate and sample size is limited, ensemble methods such as bagging and hybrid voting add computational complexity without yielding substantial performance gains.

7 Conclusion

This project explored the use of machine learning for personalized medication recommendation through a complete pipeline including preprocessing, exploratory analysis, model development, and evaluation. Despite the low predictive accuracy, the work provided valuable insights into healthcare analytics, the challenges of clinical datasets, and the importance of meaningful features for medical prediction. The project illustrates how data-driven tools can complement medical expertise, even when limited by underlying data patterns.

8 References

- Personalized Medication Dataset (Kaggle)
- Documentation and resources on machine learning, classification, and healthcare decision-support systems:
- https://scikit-learn.org/stable/modules/cross_validation.html
- https://scikit-learn.org/stable/modules/grid_search.html
- https://scikit-learn.org/stable/modules/tree.html
- https://www.ibm.com/fr-fr/think/topics/xgboost
- https://medium.com/@renjiniag/from-prediction-to-perfection-how-machine-learning