

# A Case of Study for Classifications Algorithm to a Tabular Vector Borne Disease Dataset

Alberto Arath Figueroa Salomon

December 12, 2024

## 1 Introduction

Vector-borne diseases have become relevant because human activity creates imbalance in ecosystems, and this family of diseases has become a recurring problem in warm climates. These diseases have a common set features, in addition to the fact that they are transmitted bites of a blood-sucking arthropod.

Such common features are the prognosis, where the symptoms between each other could be almost indistinguishable

The purpose of the study is to build a machine learning model that can discern classification patterns for each set of shared symptoms.

## 2 Data

### 2.1 Instance composition

Data set is composed by labeled data, 707 rows and 64 features. Training data has a synthetic origin so there is really no need for data preparation which is good as we can focus solely in the application of Machine Learning techniques

Muscle Pain	Fatigue	Weakness	Prognosis
Present	Not Present	Present	Present
Not Present	Not Present	Present	Present

Table 1: Sample training set with a subset of features.

### 2.2 Predictor Set

Is composed of binary variables specifying if given symptom is present in this prognosis instance it represents the presence or absence of a specific condition, The variable  $X$  takes values from the set:

$$X \in \{0, 1\},$$

where:

$$X = \begin{cases} 1 & \text{if the condition is **present**,} \\ 0 & \text{if the condition is **not present**.} \end{cases}$$

### 2.3 Target Variable

The target variable is called Prognosis containing the Vector-borne disease classifier

The target variable, denoted as  $Y$ , is a categorical variable representing the prognosis outcome it can take values from the following finite set of classes:

$$Y \in \{C_1, C_2, \dots, C_k\},$$

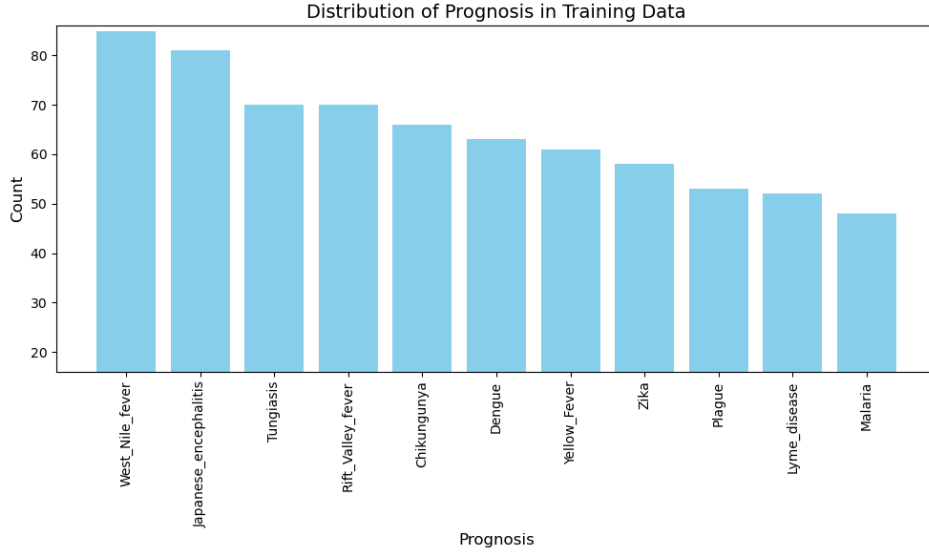


Figure 1: Distribution of Prognosis in Training Data.

where  $C_i$  represents the  $i^{\text{th}}$  class label, for  $i = 1, 2, \dots, k$ .

In this study,  $k = 11$ , and the classes are defined as follows:

- $C_1$ : Dengue,
- $C_2$ : Zika,
- $C_3$ : Malaria.

To allow the application of machine learning algorithms label encoder variables as in following example

$$Y = \begin{cases} 0 & \text{if the class is } C_1 \text{ (Malaria),} \\ 1 & \text{if the class is } C_2 \text{ (Dengue),} \\ 2 & \text{if the class is } C_3 \text{ (Zika).} \end{cases}$$

This encoding To allows the application of machine learning algorithms such as softmax regression, decision trees, or neural networks to predict the likelihood of each class based on the input features.

### 3 Exploratory Data Analysis (EDA)

#### 3.1 Class distribution

Class distribution shows there is no significant class imbalance, Figure 1 show West Nile Fever with 80 instances and Malaria falls short with 50 as part of study, class imbalance could have an impact but we expect not to be significant. Just for the sake of the study we will apply oversampling techniques to improve model.

#### 3.2 Features correlation matrix

Correlation matrix shows there is little information between features. So dimensionality reduction techniques are not likely to be effective, in Model section we will place this hypothesis to test.

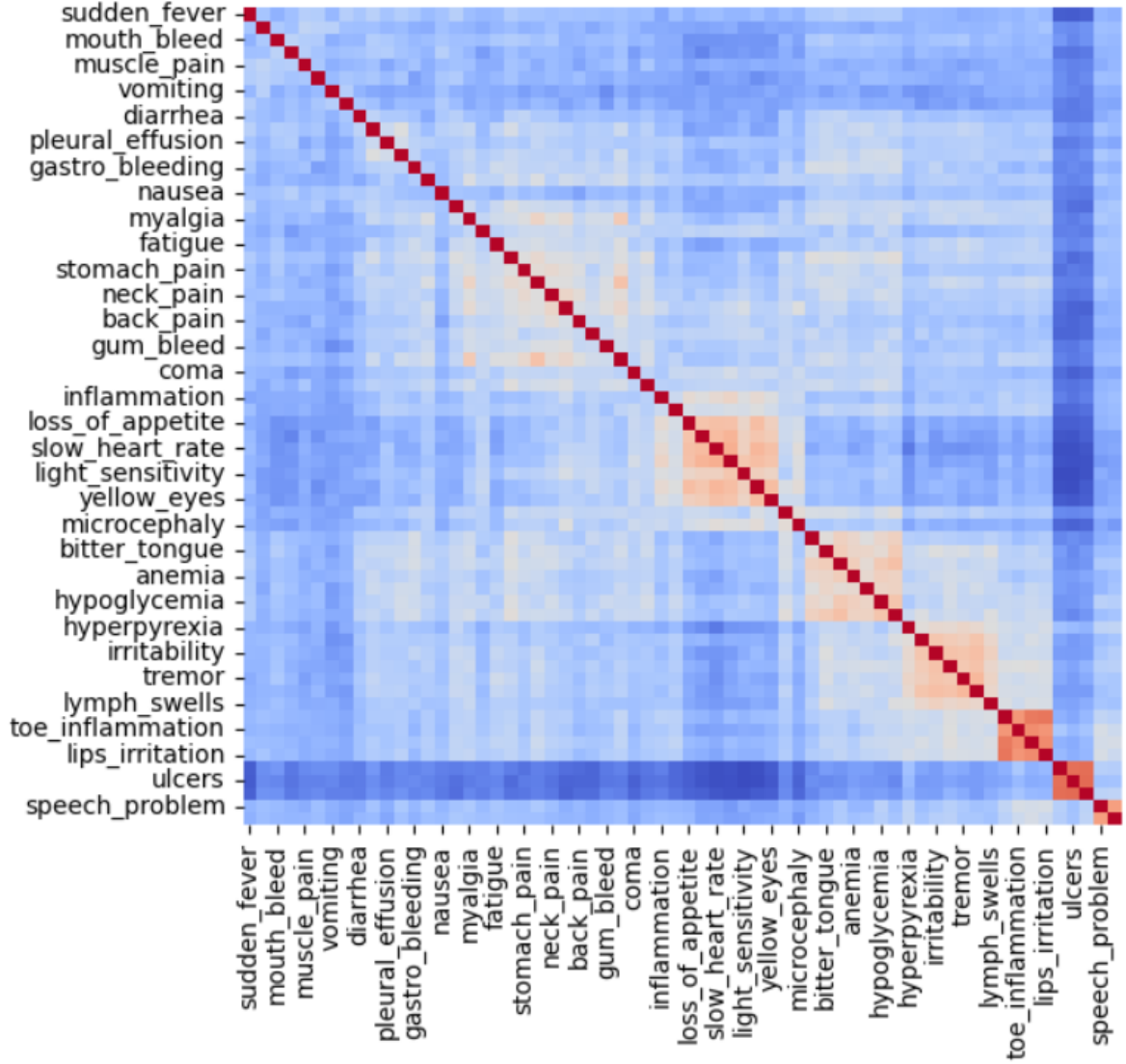


Figure 2: Features Correlation Heat Map. Blue areas indicate low correlation.

## 4 Methodology

In this study, we performed hyperparameter tuning using Grid Search, testing multiple machine learning algorithms to identify the best-performing model. The following algorithms were evaluated:

### 4.1 Cross Algorithms Grid Search Approach

Criteria for model selection and study is: collect results for a set of 4 algorithms, benchmark score will be collected based on different scores apply techniques to improve model performance

A nested grid search will be applied, outer loop being the algorithm under test and the inner loops will be defined by the hyperparameter of the given algo.

In this study we won't explain the inner workings of machine learning model as is it of scope and a waste of time we will instead try to explain:

### 4.2 Hyperparameter Tuning

Grid search was expanded based on fixing all hyperparameters while increasing the promised onecolumn

- Parameters used for each algorithm

- Explain performance results of each algorithm
- An Engineer decision for which model will be best for this data set

### 4.3 K-Nearest Neighbors (KNN)

The **K-Nearest Neighbors (KNN)** algorithm is a non-parametric, instance-based learning method. It classifies a data point based on the majority class of its  $k$  nearest neighbors in the feature space. Hyperparameters tuned include:

- **n\_neighbors**: Number of neighbors considered.
- **weights**: Weighting function for neighbors (*uniform* or *distance*).

Table 2: KNeighborsClassifier Grid Search Results

mean_test_score	rank_global_score	classifier and hyperparameters
0.230935	11	KNeighborsClassifier, n_neighbors=7, weights='uniform'
0.219302	14	KNeighborsClassifier, n_neighbors=5, weights='uniform'
0.214970	18	KNeighborsClassifier, n_neighbors=7, weights='distance'
0.200214	22	KNeighborsClassifier, n_neighbors=5, weights='distance'
0.121799	27	KNeighborsClassifier, n_neighbors=3, weights='distance'
0.121095	28	KNeighborsClassifier, n_neighbors=3, weights='uniform'

### 4.4 Random Forest Classifier

The **Random Forest Classifier** is an ensemble learning method that constructs a collection of decision trees and combines their predictions. It reduces overfitting compared to individual trees. Hyperparameters tuned include:

- **n\_estimators**: Number of trees in the forest.
- **max\_depth**: Maximum depth of each tree.
- **min\_samples\_split**: Minimum number of samples required to split an internal node.

Table 3: Random Forest Grid Search Results

mean_test_score	rank_test_score	classifier	hyperparam_combination
0.277801	1	RandomForestClassifier	RandomForestClassifier, max_depth=5, min_samples_split=2, n_estimators=50
0.256374	3	RandomForestClassifier	RandomForestClassifier, max_depth=5, min_samples_split=2, n_estimators=100
0.251517	4	RandomForestClassifier	RandomForestClassifier, max_depth=None, min_samples_split=2, n_estimators=100
0.244632	5	RandomForestClassifier	RandomForestClassifier, max_depth=None, min_samples_split=5, n_estimators=100
0.239188	8	RandomForestClassifier	RandomForestClassifier, max_depth=5, min_samples_split=5, n_estimators=100
0.213390	19	RandomForestClassifier	RandomForestClassifier, max_depth=10, min_samples_split=5, n_estimators=50
0.210364	20	RandomForestClassifier	RandomForestClassifier, max_depth=None, min_samples_split=5, n_estimators=50
0.207067	21	RandomForestClassifier	RandomForestClassifier, max_depth=10, min_samples_split=2, n_estimators=50
0.193830	23	RandomForestClassifier	RandomForestClassifier, max_depth=5, min_samples_split=5, n_estimators=50
0.192695	24	RandomForestClassifier	RandomForestClassifier, max_depth=10, min_samples_split=2, n_estimators=100
0.190548	25	RandomForestClassifier	RandomForestClassifier, max_depth=None, min_samples_split=2, n_estimators=50
0.166299	26	RandomForestClassifier	RandomForestClassifier, max_depth=10, min_samples_split=5, n_estimators=100

## 4.5 Support Vector Classifier (SVC)

The **Support Vector Classifier (SVC)** is a kernel-based method that finds an optimal hyperplane to separate classes in a high-dimensional space. Hyperparameters tuned include:

- **C**: Regularization parameter.
- **kernel**: Kernel function (*linear*, *rbf*, or *poly*) *rbf* was just included just for benchmark as it is widely used kernel

Table 4: SVC Grid Search Results

mean_test_score	rank_test_score	classifier	hyperparam_combination
0.276434	2	SVC	SVC, C=0.1, kernel='linear'
0.243171	6	SVC	SVC, C=1, kernel='rbf'
0.235429	10	SVC	SVC, C=10, kernel='rbf'
0.215171	16	SVC	SVC, C=10, kernel='linear'
0.215171	16	SVC	SVC, C=1, kernel='linear'
0.007571	29	SVC	SVC, C=0.1, kernel='rbf'

## 4.6 Logistic Regression

The **Logistic Regression** model is a linear classifier used for binary and multi-class classification problems. It models the probability of a class using the logistic function. Hyperparameters tuned include:

- **penalty**: Type of regularization (*l1*, *l2*, or *elasticnet*).
- **C**: Inverse of regularization strength.
- **solver**: Optimization algorithm for model fitting.

Table 5: Logistic Regression Grid Search Results

mean_test_score	rank_test_score	hyperparam_combination
0.242251	4	maxiter=500, multiclass='ovr', solver='saga', C=10, penalty='l1'
0.240382	5	maxiter=500, multiclass='ovr', solver='saga', C=1, penalty='l1'
0.238665	7	maxiter=500, multiclass='ovr', solver='saga', C=0.1, penalty='l2'
0.226524	13	maxiter=500, multiclass='ovr', solver='saga', C=1, penalty='l2'
0.216889	17	maxiter=500, multiclass='ovr', solver='saga', C=10, penalty='l2'
0.006944	30	maxiter=500, multiclass='ovr', solver='saga', C=0.1, penalty='l1'

## 4.7 Best cross validation score training set

Test set best cross validation Score

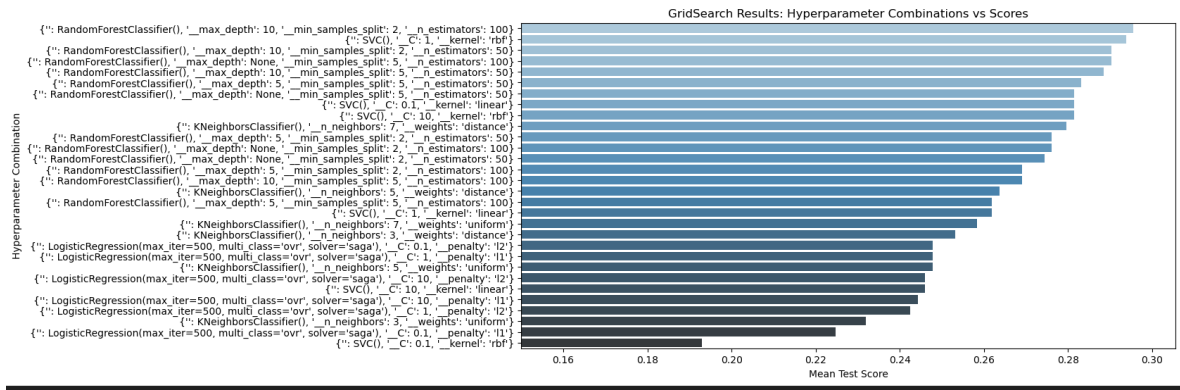


Figure 3: Distribution of Prognosis in Training Data.

Table 6: Classifier Results with Rank and Scores

Classifier	Mean Test Score	Rank
KNeighborsClassifier	0.295575	1
RandomForestClassifier	0.293805	2
RandomForestClassifier	0.290265	3
KNeighborsClassifier	0.290265	4
RandomForestClassifier	0.288496	5

## 4.8 Oversampling and Dimensionality Reduction

## 5 Results

### 5.1 Score and prediction

### 5.2 Confusion Matrix

## 6 Conclusions

## 7 Anex

## References