A Case of Study for Classification Algorithms Applied to a Tabular Vector-Borne Disease Dataset

Alberto Arath Figueroa Salomon

December 13, 2024

1 Introduction

Vector-borne diseases have become a major concern due to human activity that creates imbalances in ecosystems. These diseases, transmitted through the bites of blood-sucking arthropods, have become a recurring issue in warm climates. They share common features, including similar symptoms, which can be nearly indistinguishable from one another.
The purpose of this study is to build a machine learning model capable of discerning classification patterns based on these shared symptoms.

2 Data

2.1 Instance Composition

The dataset consists of labeled data with 707 rows and 64 features. The training data has a synthetic origin, so no additional data preparation is required. This allows us to focus solely on applying 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

The predictor set consists of binary variables indicating whether a given symptom is present in the prognosis instance. These binary variables represent the presence or absence of a specific condition. The variable X takes values from the set:

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

where:

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

2.3 Target Variable

The target variable, called Prognosis, represents the vector-borne disease classifier. The target variable, denoted as Y, is categorical and takes values from the following set of classes:

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

where C_i represents the i^{th} class label, for $i = 1, 2, \ldots, k$.
In this study, $k = 11$, and the classes are defined as follows:

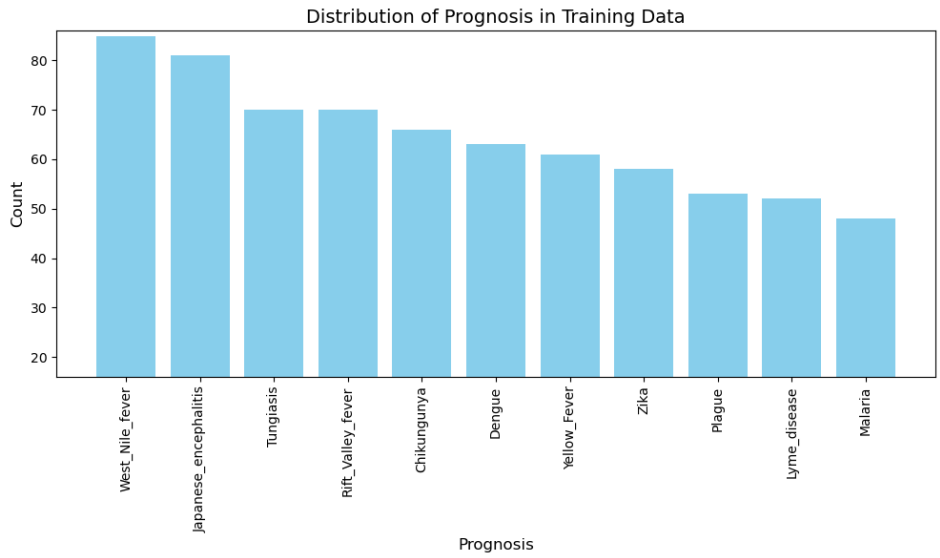

Figure 1: Distribution of Prognosis in Training Data.

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

To apply machine learning algorithms, we use label encoding as follows:

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

This encoding allows for 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

The class distribution shows no significant imbalance, as shown in Figure 1. West Nile Fever has 80 instances, while Malaria has 50. Although class imbalance could have an impact, we expect it to be non-significant. For the sake of this study, we will apply oversampling techniques to improve the model.

3.2 Feature Correlation Matrix

The correlation matrix shows little correlation between features, suggesting that dimensionality reduction techniques may not be effective. We will test this hypothesis in the modeling section.

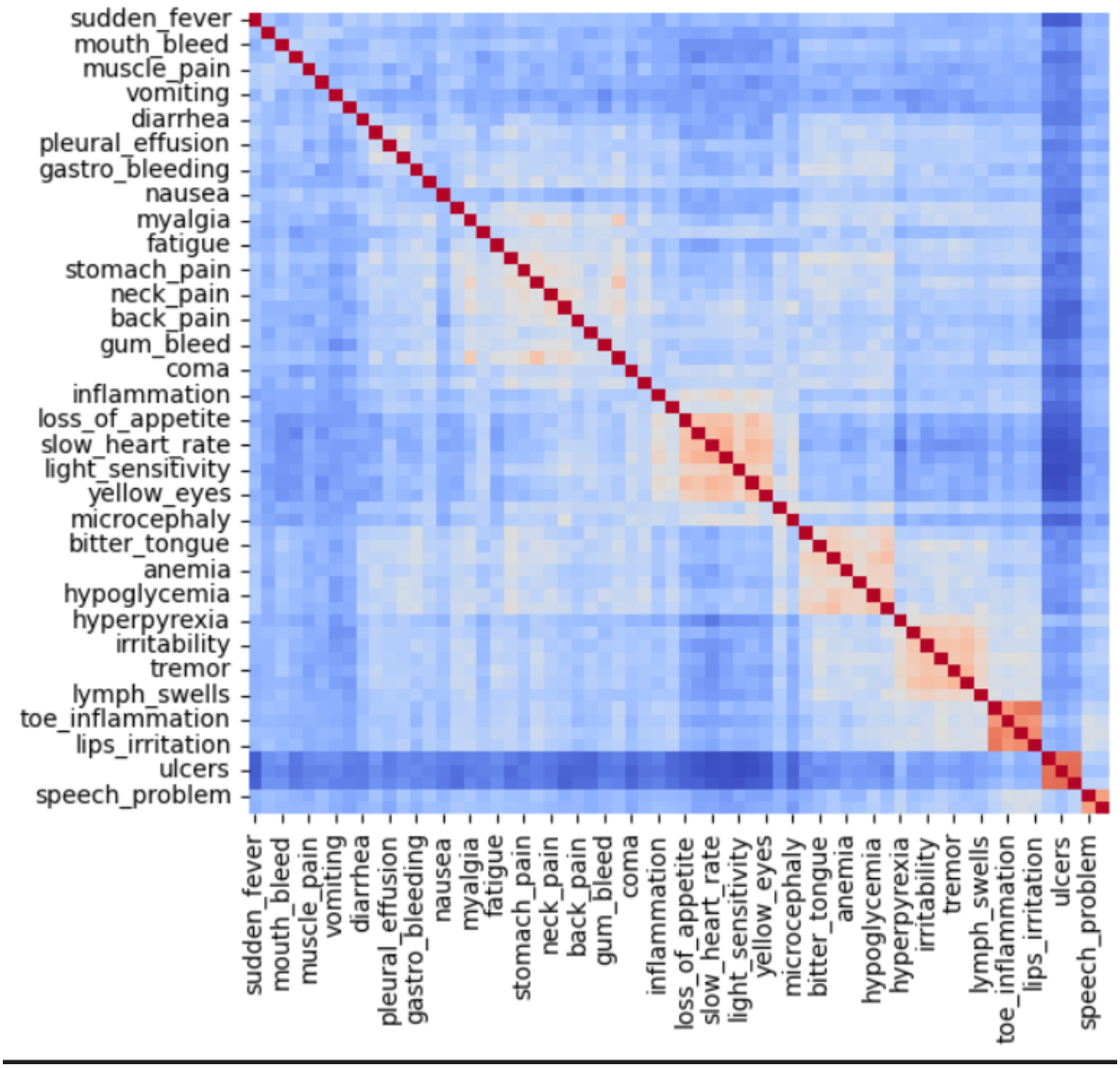

Figure 2: Features Correlation Heat Map. Blue areas indicate low correlation.

4 Methodology

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

4.1 Cross-Algorithms Grid Search Approach

The metric used to evaluate the models is the precision metric. Other metrics, such as false negatives, are outside the scope of this study.

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}}$$

4.2 Model Selection and Study

The study will collect results for four algorithms and benchmark the scores based on different techniques to improve model performance. A nested grid search will be applied, with the outer loop being the algorithm under test and the inner loop defined by the hyperparameters of the given algorithm.

We will not explain the inner workings of the models, as this is outside the scope of the study.

4.3 Hyperparameter Tuning

Grid search was expanded by fixing all hyperparameters and gradually increasing the chosen hyperparameter.

- Parameters used for each algorithm.
- Explanation of the performance results for each algorithm.
- Engineer's decision for selecting the best model for this dataset.

4.4 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'

4.5 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.

4.6 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 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

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'

4.7 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'

4.8 Best cross validation score training set

Test set best cross validation Score

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

5 Results

Table 7: Training Set best results

mean_test_score	rank_test_score	hyperparam_combination
0.304661	1	RandomForestClassifier, max_depth=10, min_samples_split=5, n_estimators=100
0.295298	2	RandomForestClassifier, max_depth=10, min_samples_split=2, n_estimators=50
0.292155	3	RandomForestClassifier, max_depth=None, min_samples_split=5, n_estimators=100
0.288419	4	SVC, C=10, kernel='rbf'
0.283026	5	SVC, C=1, kernel='rbf'

Table 8: Test Set best results

mean_test_score	rank_test_score	hyperparam_combination
0.276434	1	SVC, C=0.1, kernel='linear'
0.251400	2	RandomForestClassifier, max_depth=10, min_samples_split=2, n_estimators=100
0.246152	3	RandomForestClassifier, max_depth=10, min_samples_split=5, n_estimators=100
0.243171	4	SVC, C=1, kernel='rbf'
0.242251	5	LogisticRegression, maxiter=500, multiclass='ovr', solver='saga', C=10, penalty='l1'

5.1 Confusion Matrix

The matrix shows significant confusions among classes that have similar symptom sets. For example: "Japanese Encephalitis" and "Lyme Disease" have off-diagonal misclassifications. "Tungiasis" has the strongest performance with 12 correct predictions.

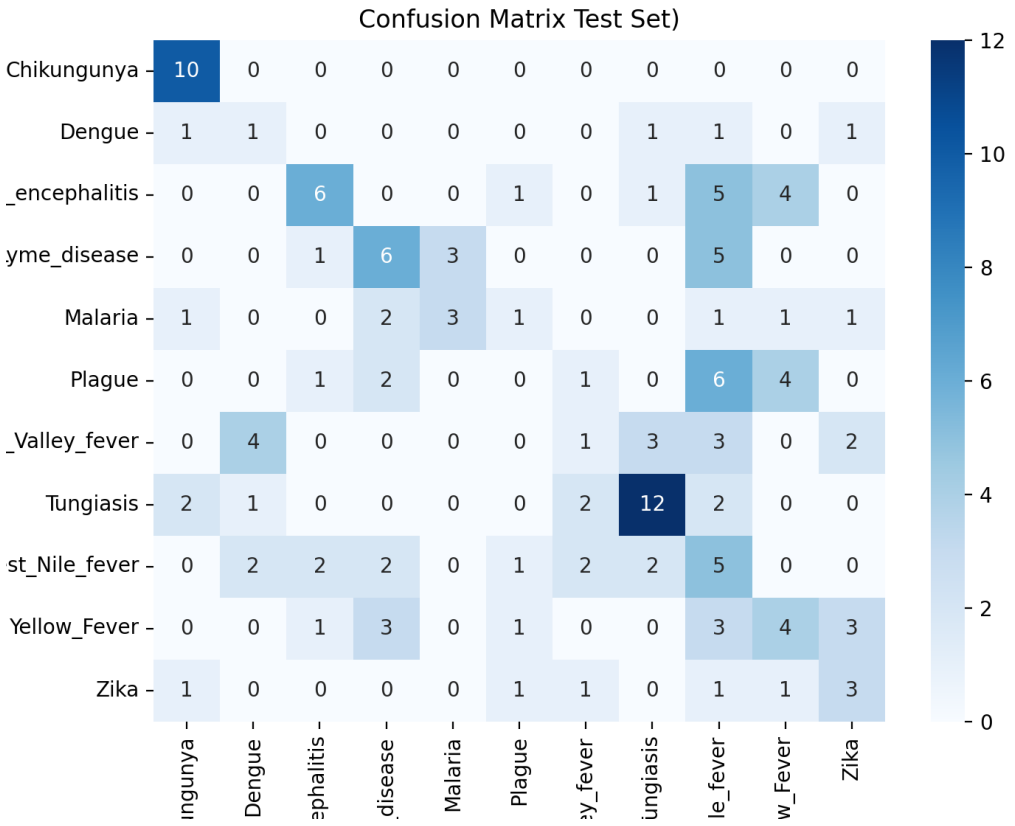

Figure 3: Features Correlation Heat Map. Blue areas indicate low correlation.

6 Discusion and further analysis

We will apply a clustering algorithm to determine the optimal number of clusters within the dataset. If the clusters are not evenly distributed, or if certain features exhibit high compactness within a cluster, it may suggest that the dataset lacks sufficient discriminative features to effectively differentiate between distinct patterns."

6.1 Clustering sympoms pattern

We will employ the k-means clustering algorithm and expect the k-means plateau to be as high as possible, or at least close to the expected value based on the characteristics of our datasete this hypothesis the following subsequent analysis is performed:

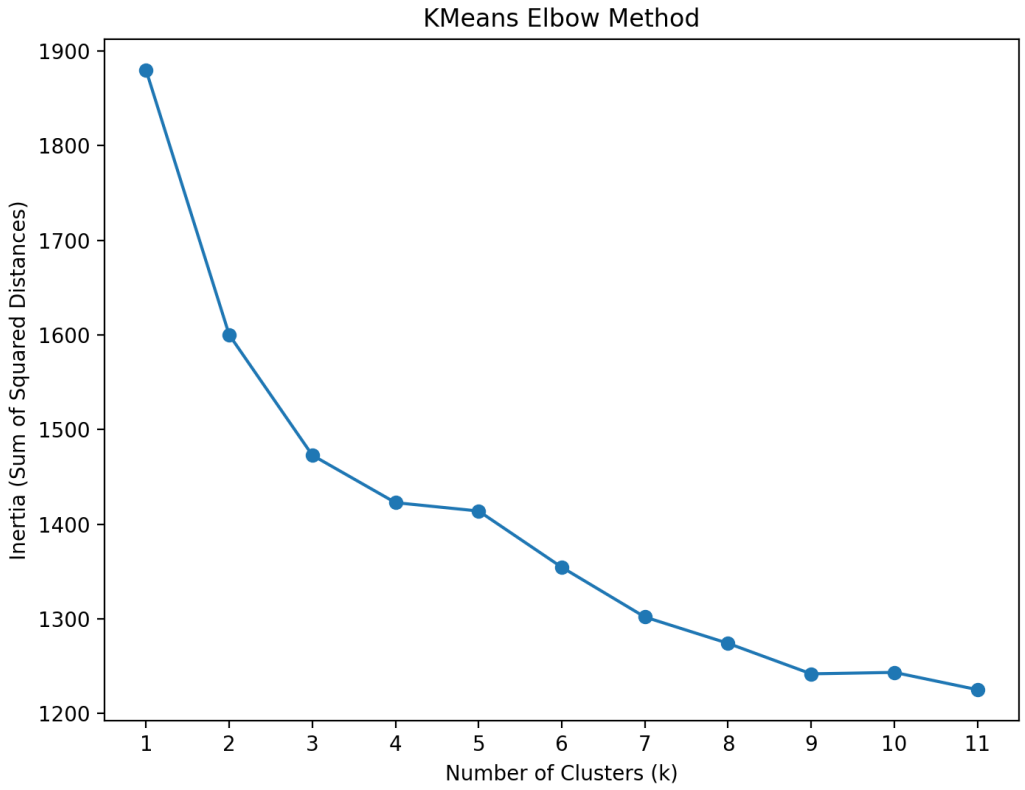

Figure 4: Features Correlation Heat Map. Blue areas indicate low correlation.

6.2 Clustering Results

The elbow inflection point is found between 4 and 6 clusters. This suggests that most of the points can be effectively grouped within this range. Beyond 6 clusters, diminishing returns occur, indicating that the model stared to clump with too few cluster centers.

7 Conclusions and Further analysis

7.1 Further feature engineering

Clustering optimal suggests data is hard to separate We could start to create features with a cross interaction machanis between sympoms for example if sympom A and B happen then make another switch This Aproach could in principle improve our classification, and we could repeat our clustering analysis to see if the features are enough

A Additional tables

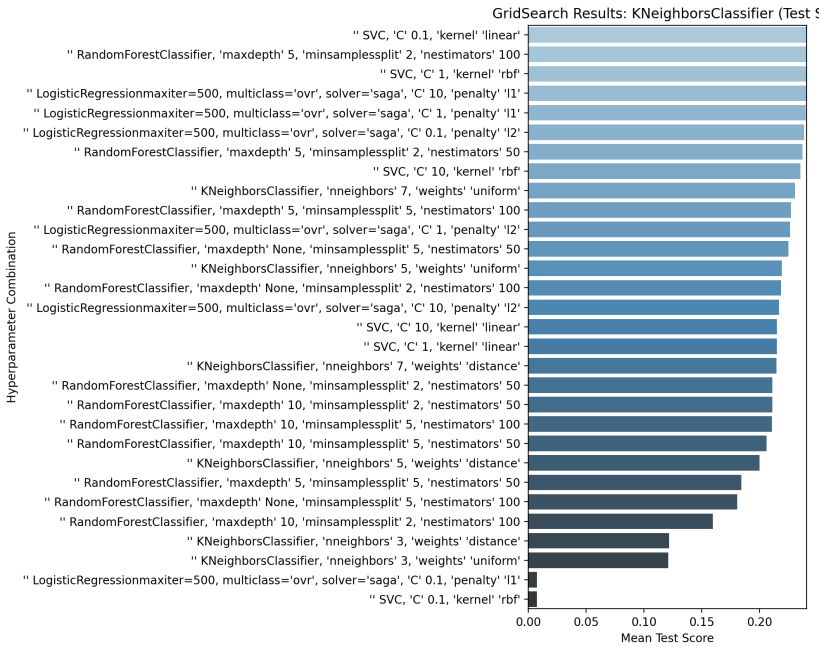

Figure 5: Performace Test Set

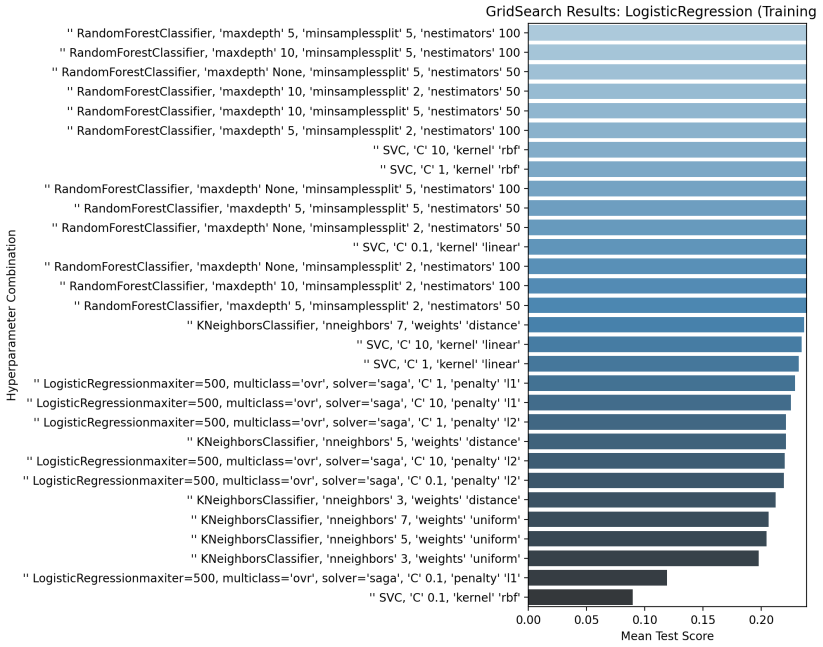

Figure 6: Performace Training Set

B Jupyter notebook

Code used for this analysis [Sal24].

References

[Sal24] Alberto Arath Figueroa Salomon. Machine learning clustering project. https://github.com/AlbertoArath/IA/tree/main/MachineLearning/Projects/Clustering, 2024. Accessed: 2024-12-13.