Adverse Reaction Cluster of the COVID-19 Vaccine: Potential Clinical Prediction Tool

Authors: Andrea Gomez, Dung Mai, Mariana Maroto

Graduate Center, CUNY - Machine Learning CSCI 740 Spring 2021

Table of Contents

Relevant Links

Github Link: https://github.com/marianamaroto/VaccineReactions

Demo: https://marianamaroto.shinyapps.io/Covid-19AdverseVaccineReaction/

Poster:

https://docs.google.com/presentation/d/1aB99idxXNbtjTwLgdSuC8szyNL4FLV6U/edit#slide=id.p1

Problem Description

After the COVID-19 virus emerged in late 2019, it triggered an urgent international response to prepare for an outbreak and develop a preventive COVID-19 vaccine. On December 11th 2020, United States Food and Drug Administration (FDA) granted Emergency Use Authorization of the COVID-19 Vaccine. Since then, there has been a massive vaccination campaign across the United States. Although generally safe, no prescription drug or biological product, such as a vaccine, is completely free from side effects. Vaccines protect many people from dangerous illnesses, but for a very small percentage its side effects may be serious.

This project aims to cluster COVID-19 vaccine adverse reactions. The purpose of the project is one, having a detailed understanding of the common types of adverse reactions and two, identifying depending on the person's adverse reactions, medical history, allergies, gender, age, and vaccine manufacturer, if they are at a higher risk of suffering a serious effect that could be life threatening.

This project uses the data from the Vaccine Adverse Event Reporting System (VAERS), which was created by the Food and Drug Administration (FDA) and Centers for Disease Control and Prevention (CDC) to receive reports about adverse events that may be associated with vaccines. VAERS is used by these government agencies to determine whether any vaccine has a higher than expected rate of rare events. Doctors, vaccine providers, and medical professionals are encouraged to report adverse events, even if they are not certain that the vaccination was the cause for the event. Since it is difficult to distinguish a coincidental event from one truly caused by a vaccine, the VAERS data will contain events of both types. Additionally, due to the human element in the creation of this dataset we can expect incomplete and imperfect data.

We hypothesized that symptoms reported after the COVID-19 vaccine would cluster individuals into distinct subtypes with different collections of symptoms. And that we could use this symptom information combined with demographic information, medical history, and vaccine manufacturers to create a predictive tool for medical support that could be used to identify unsuitable candidates for vaccination.

State-of-the-Art

Previous research has been conducted in a similar matter for COVID-19 symptoms, *Symptom clusters in COVID-19: A potential clinical prediction tool from the COVID Symptom Study app* [1]. This project follows a similar approach: first clustering, and second, predicting the need of urgent care. The main difference is that this paper uses early COVID-19 symptoms, with the purpose of it being a clinical prediction tool, while our project is focused on a more relevant problem in mid-2021 for the United States, COVID-19 vaccine adverse reactions. However, both problems are particularly challenging due to lack of data and pre-existing research from the novel virus. Additionally, the data used for this research, came with its own set of challenges, as it was self-reported (via smartphones) by infected patients, meaning if the patient becomes too unwell, they were unable to determine if that was the reason to stop recording symptoms. Comparing it to ours, our data is reported by healthcare providers, but still comes with errors due to its human aspect.

Interestingly in this base paper, due to the nature of the data, they were able to cluster not only based on common COVID-19 symptoms, but also by the days in which these appear. Providing more detailed insights into how the virus develops and the turning points for needing urgent care.

The clustering method they used was Mc2PCA, which allows for the clustering of time series with unequal duration, using a covariance matrix. The optimization of the clustering was done using a K-means iterative process series. Lastly, The Bayesian information criteria (BIC) was applied to balance model fit and model complexity, which led to identifying the number of clusters. For the prediction analysis, Random Forest was used. Needing respiratory support was the target variable, which led to a precision of 84.9% [83.5; 86.3] with a recall of 84.6% [83.2; 86.1], using bootstrapping to produce the confidence intervals.

The results distinguished six different clusters of symptoms. The authors cited that further work should be evaluated to implement their tool correctly and efficiently. The challenge lies in how to practically adapt such a tool into actionable insights for health-care providers and infected patients. Possible implementations and benefits will allow primary health-care providers to monitor patients from a distance and alert high-risk patients that need proactive care.

Approach

Our approach is divided into four distinct parts: 1) data cleaning, 2) data preparation, 3) classification, and 4) clustering.

Data cleaning:

Our project is based on data from the Vaccine Adverse Event Reporting System (VAERS), FDA and CDC that receives reports about adverse events that may be associated with vaccines. The data may also contain coincidental events that are not caused by the COVID-19 vaccines. It can also be incomplete and imperfect in some cases, as stated before due to the nature of the data. In order to prepare the data for classification and clustering analysis, extensive cleaning and quality check had to be implemented.

There are three datasets that are being used: 2021VAERSDATA, 2021VAERSSYMPTOMS, and 2021VAERSVAX. 2021VAERSDATA contains the event's IDs (VAERS_ID) and the information about the individual such as age, sex, symptom, allergy type, medical history, died, hospitalization, life threat. The dataset has 40348 rows and 35 columns. We removed the entries with age less than 16, as COVID-19 vaccines in the United States are only given to those who are age 16 and older. As well we removed those entries with incomplete information on demographic variables: gender, and age.

2021VAERSSYMPTOMS encloses the events' IDs and symptom's keywords extracted from the symptom text in 2021VAERSDATA. It has 56533 rows and 11 columns. There are 4407 unique adverse reaction symptoms from the vaccine across all entries. Due to memory size and to not over-complex the models, we kept only the symptoms with a frequency of more than 1000 across all cases (or symptoms that are frequent in more than 2% of cases). That left us with 33 common symptoms for COVID-19 vaccines. The symptoms are shown in Table 1.

2021VAERSVAX inscribes vaccine types and vaccine manufacturers. It has 40937 rows and 8 columns. We removed entries that specified vaccines for other diseases other than COVID-19. As well, there were also 13 cases of individuals who took two different types of COVID-19 (first dose with one manufacturer and a second dose with another manufacturer). We removed these cases from our dataset. Finally, before combining the three datasets by the event's IDs, we removed duplicate individuals from each of the datasets.

Symptoms	Frequency
Headache	8881
Pyrexia	7204
Chills	6865
Fatigue	6418
Pain	6034
Nausea	5039
Dizziness	4229
Pain in extreme	3678
Myalgia	3416
Injection site	3320
Injection site	2655
Arthralgia	2505
Dyspnea	2407
Vomiting	2050
Pruritus	2044
Injection site	1975
Rash	1934
Asthenia	1799
Injection site	1619
Paranesthesia	1508
Malaise	1495
Erythema	1490
Diarrhea	1456
SARS-CoV-2 test positive	1416
Injection site	1399
Urticaria	1379
Hypoesthesia	1280
Hyperhidrosis	1213
Lymphadenopathy	1212
COVID-19	1190
Cough	1141
Feeling abnormal	1101
SARS-CoV-2 test negative	1005

Figure 1: Common symptoms and their corresponding frequency.

Our data cleaning process also involved Natural Language Processing to be able to use the textual attributes from the dataset: allergy and history. We removed punctuations, converted text

to lower cases, and tokenize the texts into individual words. Then, we discarded stop-words (words that are meaningless such as *the*, *and*, *with*, etc.) and converted each word to its base form by using stemming technique (e.g., *allergy* and *allergies* become *allergi*). Lastly, we kept only the allergies and types of medical history which appeared in more than 2% of the cases.

Data Preparation:

Data preparation process involved One-hot encoding, feature scaling, and balancing the dataset with Synthetic Minority Oversampling Technique (SMOTE). One-hot encoding means integer representation for our categorical features such as symptoms, allergy, history, gender, and vaccine manufacturers. For feature scaling, we utilized the Standardization technique, which scales each feature by subtracting the mean and dividing by the standard deviation. This technique converts the mean of the feature to be zero and the standard deviation of the feature to be one.

Our target (referred as *y*) was binary and the positive class (denoted as 1) was for the individuals that either died, had a life-threatening experience, or were hospitalized according to the dataset. As shown in Figure 2, our dataset has 25358 cases of people who did not suffer a life threat experience due to the vaccine and 3695 cases which did (12.6% over the total cases). This means our dataset is highly unbalanced.

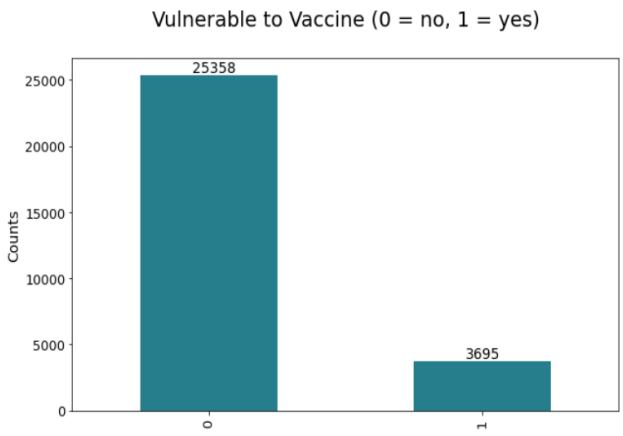


Figure 2: The number of cases which are vulnerable to the COVID-19 vaccines is 12.7% over the total cases (refer as *y*)

SMOTE transforms the imbalanced dataset by oversampling the minority class. Our training set was modified by adding synthetically generated minority class instances, resulting in a balanced class distribution. New instances created are called synthetic, because they were created out of existing minority class instances. The algorithm works by first select a minority class instance a at random and finds its k nearest minority class neighbors. Then, the synthetic instance is created by choosing one of the k nearest neighbors b at random and connecting a and b to form a line segment in the feature space. The synthetic instances are generated as a convex combination of the two chosen instances a and b [5]. In other words, synthetic instances are generated by taking the difference between a and b. Then, the difference is multiplied by a random number between 0 and 1, and is added to a. This causes the selection of a random point along the line segment between two specific features [6]. After splitting our dataset into training set and testing set, Our training dataset has 20276 instances that belong to the negative class and 2966 instances that belong to the positive class. After oversampling with SMOTE in our training dataset, the number of instances of the positive class equals the number of instances of the negative class as shown in Figure 3.

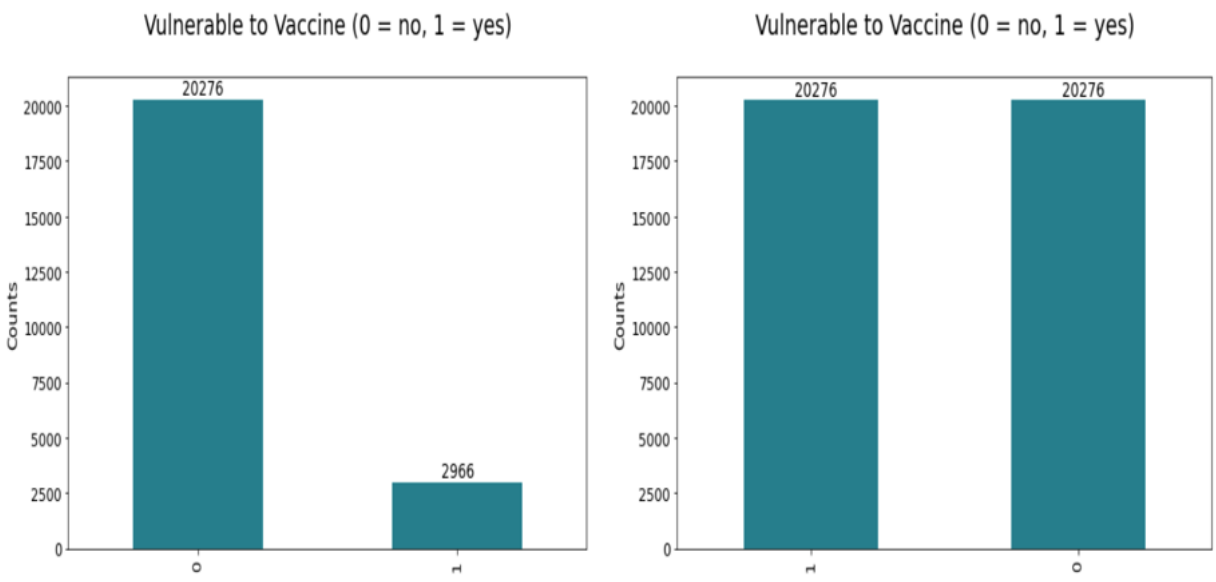


Figure 3: Training dataset target variable before and after applying SMOTE.

Classification models:

For the classification goal, we trained 7 classification algorithms using the Scikit-Learn library to predict if an individual will need urgent medical care after receiving the COVID-19 vaccines and finding unsuitable candidates for vaccination, skip if familiar with algorithms.

K-nearest neighbors (KNN):

KNN makes predictions for a new instance by searching through the entire training set for the k most similar instances (nearest neighbors) and summarizing the label classes for those k instances. Some of distance measures that are used to calculate the distances between the new input instance and k instances are Euclidean Distance, Hamming Distance, Manhattan Distance, and Minkowski Distance. The output class is determined by the class with the highest frequency from the k-nearest neighbors [10].

The advantages of KNN are that it works well with a small number of features. In addition, being a lazy learner is also its advantage, because it only begins to work at the

time when a prediction is requested. The disadvantages of KNN are not being able to work well with high dimensional data, because as the number of features increases the number of required training data increases at an exponential rate. Thus, calculating the distance between the new instance and the k-most similar instances in high dimensional data are slow and expensive. Furthermore, the data needs to be rescaled otherwise, KNN might predict poorly. KNN also does not perform well when the data has missing values, because the distance between instances cannot be calculated [10].

After tuning the algorithm with Grid Search, the optimal parameters we obtain are:

- n_neighbors (number of neighbors) : 5
- metric (distance metric) : Manhattan
- weight: uniform
- p: 1 (p=1 indicates Manhattan distance metric)

Naive Bayes (NB):

Naive Bayes classifier makes a strong assumption that the features are conditionally independent given the class. NB represents class probabilities and conditional probabilities. NB makes predictions using Bayes theorem: $MAP(h) = max(P(d|h) \times P(h))$. $P(d|h)$ is the probability of hypothesis h given the data d. This is called the posterior probability. $P(h)$ is the probability of hypothesis h being true, which is the prior probability of h. The new instances are assigned to the class with the highest probability. This is called maximum a posteriori (MAP) hypothesis [11].

The success of the Naive Bayes algorithm depends on redundant features being removed, feature selection is performed, and numerical stability is ensured. The performance of

Naive Bayes can devalue if data contains highly correlated features. This is because highly correlated features are voted twice in the model which over-inflating their importance. Removing irrelevant features in feature selection can cope with overfitting. Moreover, underflow can occur when doing computation of the likelihood of feature components. This can cause trouble to the model. Adding log into the computation can solve the problem. Feature extraction is required before implementing the Naïve Bayes algorithm.

We implement Bernoullli NB (binary features) and Gaussian NB (continuous features).

The optimal parameters we obtain for Bernoulli NB are:

- alpha (smoothing parameter): 2

The optimal parameters we obtain for Bernoulli NB are:

- var_smoothing (Portion of the largest variance of all features that is added to variances for calculation stability): 1

Logistic Regression (LR):

Logistic regression targets for binary classification. The weights in logistic regression can be estimated using maximum-likelihood estimation. It uses gradient descent to update the weights. The model predicts the class by plugging in coefficients to the sigmoid function ($S(z) = \frac{1}{1+e^{-z}}$), where $S(z)$ is the output between 0 and 1 and z is the input to the function. Given a decision boundary, if the output is greater than the decision boundary, it belongs to the positive class. Otherwise, it belongs to the negative class [12].

For logistic regression model to work well, the training data should remove noises, because LR assumes no error in the output variable; follow Gaussian distribution, because it expects a linear relationship between the input variables with the output; and

remove correlated features, because the model can overfit with highly-correlated features. In addition, the training dataset which has many highly correlated features or is very sparse can make the likelihood estimation process that learns the weights to fail to converge [12].

The optimal parameters we obtain for LR are:

- C (inverse of regularization strength, smaller values specify stronger regularization): 5
- max_iter (Maximum number of iterations taken for the solvers to converge): 150
- solver (algorithm to use in the optimization problem): liblinear
- tol (tolerance for stopping criteria): 0.001
- class_weight: Balanced (n_samples / (n_classes * np.bincount(y)))

Support Vector Machine:

The Maximal-Margin Classifier provides a theoretical model of SVM. In SVM, a hyperplane is selected to separate the instances in the input feature space by their class. The distance between the hyperplane and the closest data points is called the margin. The optimal hyperplane has the largest margin. In Practice, SVM is implemented using a kernel. Some of the kernels are linear kernel, polynomial kernel, and Gaussian radical kernel. SVM model learns the weights of the hyperplane using a variation of gradient descent called sub-gradient descent. The downside of the SVM algorithm is that the training time is slow with large datasets [13].

We train our model with the linear kernel, polynomial kernel, and Gaussian radical kernel. Linear kernel gives us the best model. The optimal parameters we obtain for LinearSVC are:

- C (regularization parameter): 1
- max_iter (the maximum number of iterations): 1000
- tol (tolerance for stopping criteria): 0.001
- class_weight: balanced (n_samples / (n_classes * np.bincount(y)))
- penalty: l1
- dual (select the algorithm to either solve the dual or primal optimization problem): False (prefer dual=False when n_samples > n_features)

Multi-layer Perceptron:

Backpropagation algorithm is used to implement the Multi-Layer Perceptron (MLP) model. The model is trained using gradient descent. The gradients are calculated using backpropagation. The backpropagation algorithm is a method for training the weights in a multilayer feedforward neural network. It requires a network structure to be defined of one or more layers where one layer is fully connected to the next layer. A standard network structure is one input layer, one hidden layer, and one output layer [14].

The issues with MLP are that its hidden layers have a non-convex loss function where there exists more than one local minimum, therefore, different random weight initializations can lead to different validation accuracy; MLP requires tuning a number of hyperparameters such as the number of hidden neurons, layers, and iterations; and it is sensitive to feature scaling.

The optimal parameters we obtain for MLP are:

- hidden_layer_sizes: (50,)
- activation: logistic
- alpha: 0.1

- tol: 0.0001
- learning_rate_init: 0.01
- learning_rate: adaptive

Random Forest (RF):

Random Forest is a type of ensemble learning, which builds a large number of decision trees from bootstrap samples from the training dataset. The algorithm works by creating a bootstrapped dataset of the training set. Then, a decision tree is built by selecting a random subset of input features from the bootstrapped dataset. The process is repeated to create n number of trees. The new instance is classified into the label class which wins the majority vote after it runs through all the n trees in the random forest.

The advantages of Random Forest are being able to handle well with high dimensional data and missing data; being insensitive to outliers; and having a lower chance to be overfitting because it considers only a subset of features, thus resulting in a wide variety of trees [15]. The disadvantages of Random Forest are slow training and might not give good performance on an imbalanced dataset, because it is possible that the bootstrap sample has very few or none of the minority class [16].

The optimal parameters we obtain from hyperparameter tuning are:

- max_depth (the maximum depth of the tree): 50
- max_features: auto (squared-root the number of features)
- min_samples_leaf (the minimum number of samples required to be at a leaf node): 1
- min_samples_split (the minimum number of samples required to split an internal node): 10

- n_estimators (the number of trees in the forest): 2000

Gradient Boosting:

Gradient Boosting is an ensemble learning method. It involves a loss function to be optimized, a weak learner (decision trees) to make predictions, and an additive model to add weak learners to minimize the loss function. In Gradient Boosting, after calculating the loss and before computing the gradient descent, a tree must be added to the model that reduces the loss. This is called functional gradient descent. Gradient boosting can overfit a training dataset easily because it is a greedy algorithm. Therefore, regularization methods such as L-1 and L-2 regulation of weights can be necessary [17].

The optimal parameters we obtain from hyperparameter tuning are:

- learning_rate: 0.1
- max_depth (the maximum depth limits the number of nodes in the tree): 8
- min_samples_leaf (The minimum number of samples required to be at a leaf node): 30
- min_samples_split (the minimum number of samples required to split an internal node): 2
- n_estimators (The number of boosting stages to perform): 300

AdaBoost:

AdaBoost works by weighing the instances and putting more weight on instances that have the difficulty of being classified. Weak learners (decision trees with a single split) are added sequentially until there is no more improvement that can be made on the training dataset. The predictions are made by the majority vote of the weak learners'

predictions which are weighted by their individual accuracy. AdaBoost is insensitive to outliers and noises [17].

Clustering:

For our clustering goal, we implemented 7 clustering algorithms using the Scikit_learn library to segment individuals based on their COVID-19 vaccine symptoms. With the purpose again, of understanding what are the common symptoms, which symptoms usually appear together, how large is each segment, and the life risk within each segment, skip if familiar with algorithms.

K-means:

K-means is an iterative process that involves in choosing the number of clusters k. Then, the inputs are partitioned into sets S_1, S_2, ..., S_k in the way that minimizes the total sum of squared distances from each point to the mean of its assigned cluster [3]. In other words, the algorithm has 4 steps. First is manually choosing the number of k clusters. Second is randomly choosing k different cluster centers (k-means) from the input dataset. Third, assigning each point to the closest cluster center (mean) and that will form k clusters. Fourth, re-calculating each cluster center (new mean) based on the points that were assigned to it. It is done by calculating the mean all the points currently assigned to that cluster center. Then, reassign each data points to the new closest center. The iteration stops when there are no more changes in the cluster centers. K-means is sensitive to noise and outliers in the data.

BIRCH:

BIRCH stands for Balanced Iterative Reducing and Clustering using Hierarchies. The algorithm consists of two steps. The first step is to build the *cluster feature* (CF) tree. The algorithm scans the data one record at a time, and determines whether a given record

should be assigned to an existing cluster, or a new cluster should be constructed. The second step is to apply any clustering algorithms on the leaves of the CF tree. BIRCH is known to work very well with large datasets, because of the time complexity and space efficiency [4].

DBSCAN:

DBSCAN (Density-Based Spatial Clustering and Application with Noise) is a density-based clustering algorithm, which can be used to identify clusters of any shape in a data set containing noise and outliers. Two important parameters are: epsilon ("eps") and minimum points ("MinPts"). "Esp" is the radius of the neighborhood around a point and"MinPts" is the minimum number of neighbors within "eps" radius. The algorithm follows 3 steps. Step 1 is to compute the distance between each point x_i and the other points. Then, find all neighboring points within distance "eps" of the starting point (x_i). Each point, with a neighbor count greater than or equal to MinPts, is called core point. Step 2 is to create a new cluster if any core point is not yet assigned to a cluster. Step 3 is to iterate through the remaining unvisited points in the dataset. Those points that do not belong to any cluster are treated as outliers or noise [18].

Spectral Clustering:

Spectral Clustering uses the eigenvalues and vectors of the graph Laplacian matrix to find clusters. This algorithm is sensitive to noise and it is expensive for large datasets, because of the computation of the eigenvectors [19].

Gaussian Mixture:

Gaussian mixture model clustering fits a set of Gaussian distributions to the data and estimates the probability of the data coming from each Gaussian. The

expectation-maximization (EM) algorithm is used to iteratively update the model until the likelihood of the data converges. Gaussian Mixture gives a probability of each point belonging to each cluster. The drawbacks of Gaussian Mixture are that It cannot select the optimal number of clusters automatically; it can converge to a locally optimal model because to the randomness of the initial Gaussians; and it is sensitive to outliers [20].

Affinity Propagation:

Affinity Propagation does not require manually assigning the number of clusters. Instead, it can find the number of clusters automatically from a dataset. This algorithm performs clustering by passing messages between data points. In other words, it takes

a similarity matrix between data points as the input and and treat all data points as potential exemplars, representatives of clusters. Clusters are formed when the similarity between data points and their exemplars are maximized. The issue with Affinity Propagation is time complexity. It is very expensive to run this algorithm on large datasets. It is also sensitive to noise [21].

Agglomerative Clustering:

In Agglomerative, each data point is its own cluster and the two most similar data points are clustered. The similarity is calculated by a distance measure. This process repeats until all data points are joined to one cluster. Agglomerative is a greedy algorithm, because it only focuses on the current state and does not consider any other alternative that could be better. This also means that merging cannot be reserved. The issues with this algorithm are being sensitive to noise and high in time complexity [22].

Evaluation

Model evaluation is a key factor of any machine learning task or project. Model evaluation allows us to compare the performance of different algorithms and methods used to find the optimal solution that addresses the problem of a learning task. The analysis of evaluation metrics is also useful to understand how a learning algorithm is failing and its weakness. This information can be used to improve the algorithm or its parameters, or to implement a new model that will address the weakness of the current model and provide a better solution to the problem. General metrics have been established across the different types of learning, supervised and unsupervised. In our project, we present two independent approaches that aim to acquire a better understanding of the COVID-19 vaccine reactions and to conduct a threat-risk classification task across patients, given certain features. Each approach evaluation was carried out independently. To evaluate the performance of the trained models in each of our approaches we used common metrics used in machine learning.

Classification metrics

The metrics used to measure performance of the classifiers were recall, specificity, f1 score, confusion matrix and AUC-ROC (Area under the ROC). These metrics are derived from the type of classification or errors that a learning algorithm produces. Once a classifier predicts a class for each sample, depending on the assigned class, the sample could be classified as True Positive (TP), False Positive (FP), True Negative (TN) or False Negative (FN), forming the confusion matrix of a classifier.

The most common metric to assess performance of a classification task is accuracy, which quantifies the number of correct classifications that are made, given a dataset. As mentioned by Novakovic[8], the main disadvantage of accuracy is that it neglects the differences between the

types of errors (FN and FP) and it is dependent on the distribution of class in the dataset. Due to the nature of our dataset (imbalanced distribution of classes) and the aim of the classification task of our project (prediction of high risk patients) we decided that we will focus on improving recall (sensitivity or True Positive Rate) over precision and leave aside accuracy. Finding a model that has a higher recall comes at the cost of a lower precision (allowing more FP type of errors), this is known as the precision-recall trade-off.

Recall

Recall is defined as TP/(TP+FN), the ratio of the samples that were correctly classified as True Positives, and the samples that truly belong to the positive class, True Positives and False Negatives. The positive class corresponds to high-risk patients and the negative class to non-high risk patients. Setting up our models to obtain a higher recall measure meant that we allowed a lower number of False Negatives in our models. False Positive errors can be seen as less important or less consequential errors in healthcare diagnoses, however this type of error still 'raises an alarm' on the patient. Predicting whether or not a patient is at high risk of presenting a threatening situation from the vaccine reactions is a task where having a False Positive error or false alarm could be disregarded after following up with a doctor's visit or healthcare professional diagnosis. In contrast, if a patient's prediction is a False Negative, meaning the patient is at high risk but the model predicted not at high risk, there could be more room for unwanted consequences, resulting in an untreated patient that could lose his/her life.

$$Recall = \frac{TP}{TP + FN}$$

Specificity and Balanced Accuracy

Following the research *Evaluation of machine learning algorithms for health and wellness applications: A tutorial*. [7] practices, we decided to also include specificity (TNR - True

Negative Rate) and balanced accuracy as metrics for the performance of our models. Specificity quantifies the number of the True Negatives cases that were correctly classified by the algorithm. It is defined as the ratio of the samples that were classified as True Negatives, and the samples that truly belong to the negative class, True Negatives and False Positives. Specificity is basically the recall of the negative class. Balanced accuracy is a metric used when data is imbalanced, it accounts for the positive and negative class and doesn't inflate performance in imbalance datasets. Balanced accuracy (in the binary classification case) is defined as the arithmetic mean of sensitivity (Recall) and specificity._The preference of allowing more False Positives over False Negative type of errors, and considering recall and specificity as evaluation metrics, is perhaps one of the most common practices used in healthcare contexts when predicting patients' diagnoses.

$$Specificity = \frac{TN}{TN + FP} \qquad Balanced\ Accuracy = \frac{Sensitivity + Specificity}{2}$$

F1 score, Precision-Recall AUC and precision were also used as second support metrics to observe and maintain a good algorithm's performance in general and to make sure the model was not making too many misclassifications.

Clustering metrics

Clustering is an unsupervised learning technique whose objective is to find a structure in a collection of unlabeled data. Clustering techniques allow the identification of useful patterns and feature relationships from data, even when the data is highly dimensional. Some clustering techniques require the user to specify the number of clusters. In some cases, the number of clusters can be specified by the nature of the dataset or by domain knowledge. In our case, we

had a rough estimate of the number of clusters we wanted to group our data. Based on the symptoms, 3 - 5 clusters could be sufficient to obtain additional knowledge about the group of symptoms that patients presented together and other patterns. When domain knowledge is not accessible, an approximation to the appropriate number of clusters can also be established by mathematical and statistical methods like the elbow method, using the inertia/distortion of a cluster, and metrics like the Silhouette Coefficient [9]. The elbow method uses the sum of the square distances between each sample to the closest cluster centroid as a performance measure over different values of clusters. The Silhouette coefficient is used to measure the separation distance between clusters. It is a measure of how close each point in a cluster is to sample points in the neighbour clusters. A Silhouette coefficient close to 1 means the clusters are clear and well separated. We used the Silhouette Coefficient and DB Score only as a comparison measure across the different clustering models used, we already had an idea of the numbers of clusters we needed.

Evaluating clustering techniques can be a very subjective problem. We mainly used domain knowledge to interpret and evaluate the results we obtained from different clustering algorithms. After applying different models, we compared the symptoms that were present together in the clusters obtained from the models. If the group of symptoms in each cluster were very similar with the ones from different models and made sense we assumed that the results were valid. We did comparisons using 3 and 5 clusters to find different patterns and relationships across these groups of patients.

Results

Classification results

As discussed in the Approach, we evaluated 7 different classification algorithms to predict whether or not a patient would present high-risk adverse reactions (in need of immediate care) from the COVID-19 vaccine. The algorithms we trained were KNN, Naive Bayes, Logistic Regression, Support Vector Machines, Multilayer Perceptron, Random Forest and GDBoosting. Some other ensemble algorithms like AdaBoost and Easy Ensemble were implemented, but no hyperparameter tuning was performed on them. The results are shown in Figure 4.

Model	Precision	Recall	Specificity	F1-score	PR -AUC Score	Balanced Accuracy
KNN	0.67	0.40	0.97	0.50	0.46	0.68
KNN SMOTE	0.38	0.61	0.85	0.47	0.38	0.73
Gaussian NB	0.40	0.65	0.86	0.50	0.45	0.76
Gaussian NB SMOTE	0.28	0.83	0.69	0.41	0.55	0.76
Bernoulli NB	0.56	0.52	0.94	0.54	0.54	0.73
Bernoulli NB SMOTE	0.32	0.76	0.77	0.45	0.54	0.77
Logistic Regression	0.36	0.78	0.80	0.49	0.60	0.79
Logistic Reg SMOTE	0.36	0.77	0.80	0.49	0.61	0.79
SVM	0.37	0.77	0.81	0.50	-	0.79
SVM SMOTE	0.36	0.77	0.81	0.49	-	0.79
MLP	0.80	0.43	0.98	0.56	0.64	070
MLP SMOTE	**0.38**	**0.76**	**0.82**	**0.51**	**0.61**	**0.79**
Random Forest	0.60	0.61	0.94	0.61	0.64	0.78
RF SMOTE	0.60	0.56	0.95	0.58	0.62	0.75
Gradient Boost	0.69	0.49	0.97	0.57	0.63	0.73
GB SMOTE	0.62	0.57	0.95	0.59	0.64	0.76
AdaBoost	0.73	0.44	0.98	0.55	0.62	0.71
AdaBoost SMOTE	0.49	0.64	0.90	0.56	0.61	0.77
Easy Ensemble	**0.38**	**0.77**	**0.82**	**0.51**	**0.62**	**0.80**

Figure 4. Classifiers performance

The best model's performance was the Multilayer perceptron with SMOTE with recall 0.76, specificity 0.82, and PR-AUC Score 0.61. As discussed in the evaluation metrics, we choose to

improve recall over precision, leaving us with low precision models and higher false positives rate, thus lowering F1-score. However, a higher false positive rate (patients who were predicted as high-risk but are not at high-risk) in this specific application is not as consequential as having a high rate of false negative values or higher precision. SVM (LinearSVC with SMOTE) and Logistic Regression scored very similar values with a lower precision than MLP with SMOTE. Even though we focused on recall and sensitivity, the precision couldn't be dropped dramatically (close to 0.10) causing the model to have less correct classifications. That is why we included f1-score and PR-AUC scores, to maintain the model away from only classifying one class, to maintain a balance in the model. Surprisingly, the Easy ensemble algorithm, without hyperparameter tuning, performs almost the same as the MLP classifier with SMOTE. This algorithm is not included in the sklearn library so we considered MLP our best model. The confusion matrix and results report of the MLP algorithm are shown in Figure 5 and Figure 6.

	Precision	Recall	F1-score
0	0.96	0.82	0.89
1	0.38	0.76	0.51
Weighted avg	0.89	0.81	0.84

Figure 5. Classification Report MLP Classifier

N = 6162	Predicted 0	Predicted 1	
True 0	4180	902	5082
True 1	177	552	729
	4357	1454	4732

Figure 6. Confusion Matrix MLP Classifier

- PR-AUC: 0.61
- Balanced Accuracy: 0.79

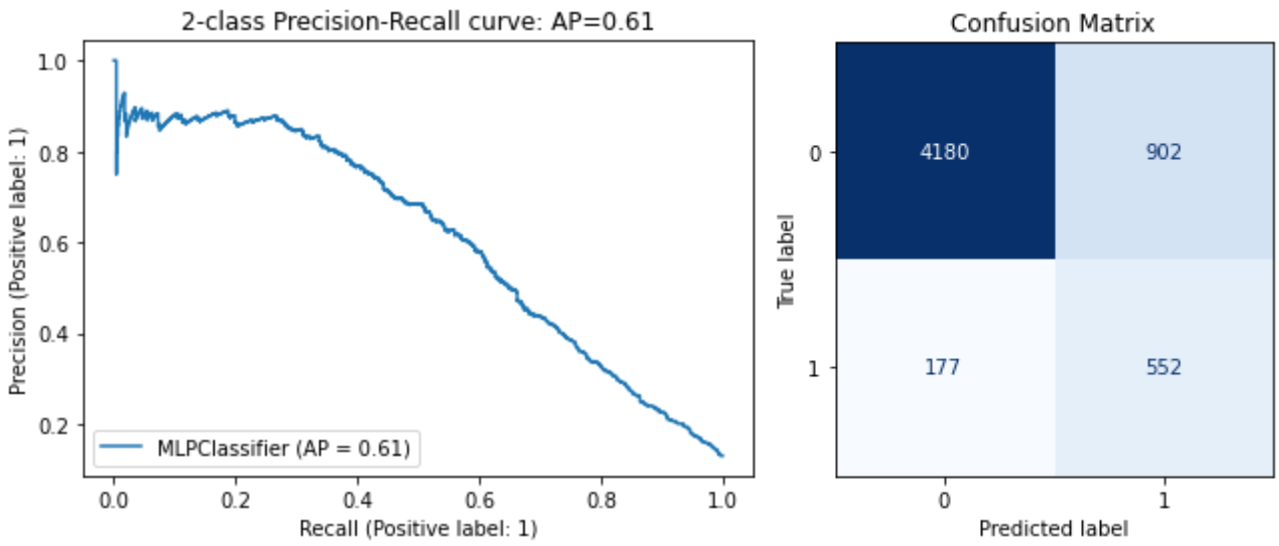


Figure 7. Left: Precision Recall Curve of MLP Classifier. Right: Confusion Matrix of MLP Classifier

As we can see in Figure 7 in the left image, a higher recall, the ability of the model to make more correct predictions about the minority/positive class, comes at the risk of low precision and less correct classifications. Out of 729 true positives samples our model correctly classified 552 and 902 as false positives. The model presents a high number of false positives cases. However, false positive patients are patients who will raise an alarm in the model as in need of immediate care, but after a professional diagnosis by a healthcare worker the patient's prediction by the model could be disregarded and confirmed to be a false positive.

To avoid having a model that underfits/overfits the dataset we used the *cross_val_score* function from sklearn to measure the f1-score and balance accuracy of our model in the training set. The package *make_pipeline* from *imblearn.pipeline* was also used as a tool to handle the validation of our training set with cross validation without including the synthetic data added by the SMOTE technique in our train dataset. The samples added from SMOTE are only used to train

the learning algorithm, but they should not be used for validation, testing or evaluation purposes. The results obtained from the cross validation are:

- F1 score validation set: 0.50
- F1-score test set: 0.50
- Balance accuracy validation set: 0.78
- Balance accuracy validation set: 0.76

Clustering results

The analysis of the clusters was based on the knowledge we acquired from the data while handling it and analyzing it. There was no measure that strictly determined what algorithm delivered the best results. We followed the patterns that appear repeatedly across different algorithms techniques. From analyzing the results from the clustering techniques we opted to use the results that were common after using 3 clusters across different models and a more detailed analysis was included using 5 clusters. The results discussed here were the ones obtained from the K-Means++ algorithm. However, we will show a comparison of very similar patterns found using other algorithms.

The clustering algorithms used were K-Means++, BIRCH, DBSCAN, Spectral Clustering, Gaussian Mixtures, Affinity Propagation Clustering and Agglomerative clustering. DBSCAN and Gaussian Mixture did not produce any meaningful patterns, so we disregarded their results.

As stated earlier, we clustered our data into groups of three and five according to the results of the Elbow Method (Figure 8) and our own interpretation of the clusters.

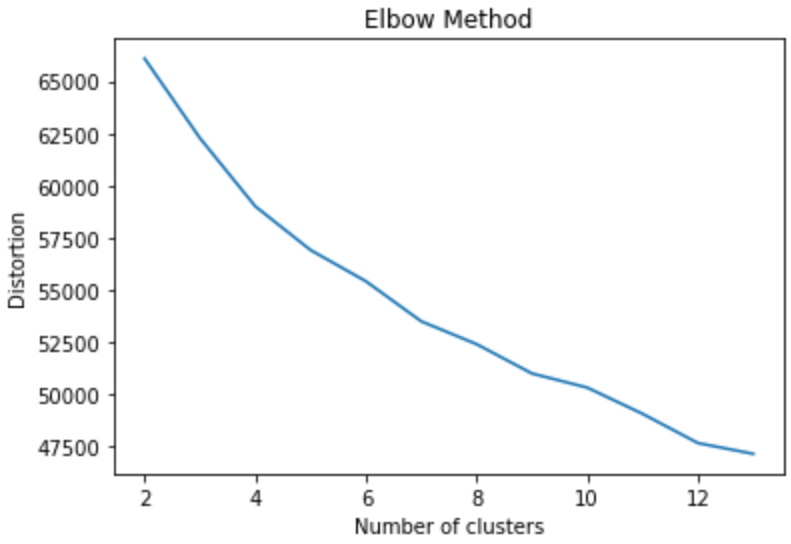
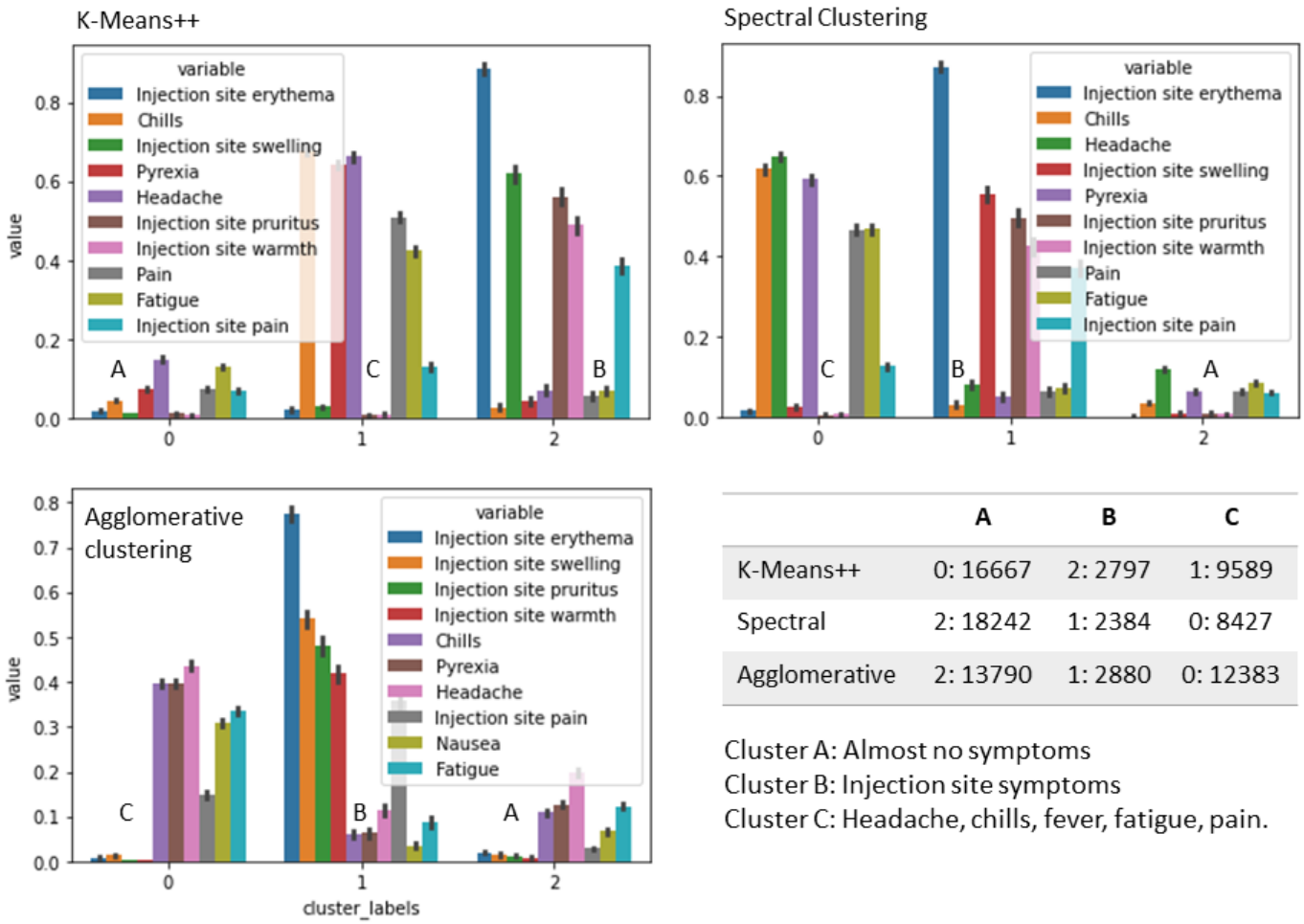


Figure 8: Elbow Method to Determine Optimal Clusters for K-Means++ Algorithm

When clustering our data into three clusters we noticed that there was a strong similarity between the results obtained from KMeans++, Spectral clustering, Agglomerative clustering and BIRCH (some of the clusters from BIRCH shared the same features from the other three algorithms). Figure 8 shows the most recurrent symptoms that were grouped together from KMeans++, Spectral clustering and Agglomerative clustering. The y-axis in the figures below represent the percentage of patients from that cluster that presents a specific symptom. The x-axis represents the cluster label assigned by the algorithm.



	A	B	C
K-Means++	0: 16667	2: 2797	1: 9589
Spectral	2: 18242	1: 2384	0: 8427
Agglomerative	2: 13790	1: 2880	0: 12383

Cluster A: Almost no symptoms
Cluster B: Injection site symptoms
Cluster C: Headache, chills, fever, fatigue, pain.

Figure 9. Cluster results from KMeans++, Spectral Clustering and Agglomerative clustering.

The table in the bottom right of Figure 9 contains the number of samples/patients per cluster in each algorithm and the type of clusters found. The three clusters can be defined as:

- Cluster A: Patients developed none of the common symptoms and less than 20% of the cluster presented a headache. The "no common symptoms" cluster represents the majority of the patients, it represents between 47% - 63% of the dataset. The percentage varies with the algorithm.

- Cluster B: Patients developed mainly injection site related symptoms like itch, redness, swelling, warmth and pain around the area where the vaccine injection was applied. This is the smallest cluster, it's represented by 7% - 10% of the dataset. The percentage varies with the algorithm.
- Cluster C: Patients developed stronger symptoms including headache, fever, chills, fatigue, and nausea. This cluster is represented by 30% - 42% of the dataset. The percentage varies with the algorithm.

We finally decided to select 5 clusters to obtain more detailed characteristics across clusters. Just as we found similar patterns from different algorithms when using 3 clusters, we found similar results using 5 clusters from KMeans++ and Agglomerative clustering . The results are shown in Figure 10.

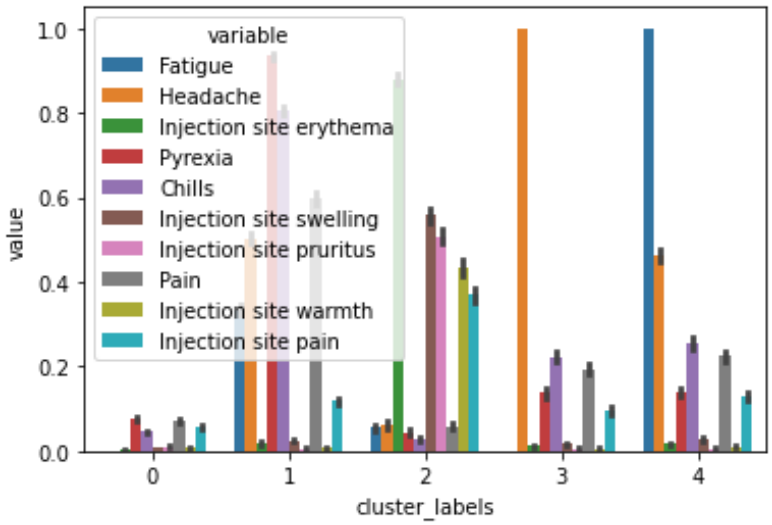


Figure 10. Common Symptoms in Each Cluster KMeans++ with k=5

Additionally we looked at how our additional demographics attributes, vaccine manufacturers, and our target variable (life threat) from the classification to further understand the composition of the clusters.

	K-Means 3 Clusters				K-Means 5 Clusters					
	0	1	2	Total	0	1	2	3	4	Total
% Life Threat	7%	1%	18%	13%	19%	7%	1%	10%	10%	13%
% Female	76%	94%	75%	77%	74%	75%	94%	77%	76%	77%
% Moderna	40%	88%	46%	48%	47%	40%	88%	42%	42%	48%
% Pfizer	44%	11%	48%	43%	48%	42%	10%	48%	48%	43%
% Janssen	15%	2%	5%	8%	5%	18%	2%	9%	10%	8%
% Allergies	4%	3%	3%	4%	3%	3%	3%	4%	4%	4%
% Hypertension	4%	5%	6%	6%	7%	4%	5%	4%	5%	6%
% Dyspnoea Symptom	6%	1%	10%	8%	11%	5%	1%	6%	8%	8%
Average Age	46	47	51	49	51	45	47	48	49	49
Average Days from Vaccine	6	12	18	13	17	3	12	20	10	13

Comparisons of Column Proportions[b]

	K-Means 3 Clusters			K-Means 5 Clusters				
	0	1	2	0	1	2	3	4
	(A)	(B)	(C)	(A)	(B)	(C)	(D)	(E)
% Life Threat	B		A B	B C D E	C		B C	B C
% Female	C	A C				A B D E	A	
% Moderna		A C	A	B D E		A B D E		
% Pfizer	B		A B	B C	C		B C	B C
% Janssen	B C		B	C	A C D E		A C	A C
% Allergies								A C
% Hypertension			A B	B C D E				
% Dyspnoea Symptom	B		A B	B C D E	C		C	B C D

Results are based on two-sided tests. For each significant pair, the key of the category with the smaller column proportion appears in the category with the larger column proportion.
Significance level for upper case letters (A, B, C): .05

a. This category is not used in comparisons because its column proportion is equal to zero or one.

b. Tests are adjusted for all pairwise comparisons within a row of each innermost subtable using the Bonferroni correction.

Figure 11. Proportions of demographic, life threat (target in classification), and vaccine information for each of the clusters. Additional stat testing at 95% CI table below.

Our final clusters are defined as follows:

- Cluster 0: Patients developed almost no common symptoms (and instead rarer ones such as dyspnoea/breathing difficulty). The “no common symptoms” cluster represents the majority of the patients, it represents 45.88% of the dataset. [samples in cluster:13332]. This cluster also has higher chances of life threat, tends to be of older age and more likely to have hypertension. As the Pfizer vaccine was approved first, and therefore given to the most vulnerable and older population first. This cluster also over-indexes in the Pfizer vaccine.
- Cluster 1: Patients developed multiple symptoms like fever, chills, pain, headache and fatigue. This cluster represents 16.84% of the dataset. [samples in cluster:4893]. This cluster has a higher male proportion than others, slightly younger base, and the symptoms tend to occur after just a few days from the shot.
- Cluster 2: Patients developed mainly injection site related symptoms like itch, redness, swelling, warmth and pain around the area where the vaccine injection was applied. This cluster represents 9.26% of the dataset. [samples in cluster:2691]. As expected this cluster has a very low life threat, only 2%. It over-samples for the Moderna vaccine and for females.
- Cluster 3: Patients developed mainly headaches and about 20% of patients presented chills, fever and pain. This cluster represents 13.12% of the dataset. [samples in cluster:3813]. This cluster of symptoms occurs later after the vaccine (on average 20 days after). It over-samples as well for Pfizer vaccine.

- Cluster 4: Patients developed mainly fatigue and headache. And about 20% of patients presented chills, fever and pain. This cluster represents 14.88% of the dataset. [samples in cluster:4324]. This cluster skews towards male compared to other clusters, and are more likely to have a pre-existing condition of allergies.

Discussion

In this section we discuss the results and the reasons why the Multilayer perceptron has the best performance. MLP ability to learn complex patterns is due to its multilayer structure of stacked logistic regressors. The MLP is able to find/learn non-linear relationships between features that other linear algorithms could not do itself. MLP does automatic feature extraction in the hidden layers without actually augmenting the data, unlike Logistic Regression or SVM where the basis function is used to create new features. In feed-forward neural networks (with backpropagation) information in propagated forward and the errors are propagated backwards. While the network learns the weights (backpropagation) it can also apply regularization, which allows the neural network to give less or more weight to different features depending on how much they contribute to the neural network. Other regularization techniques like dropout and early stopping are used to avoid overfitting.

Logistic Regression and SVM scored very close scores after MLP. The performance of these models in high dimensional data could be attributed to the use of regularization penalties applied to features. Ensemble algorithms were also trained, and from the results Table 2 we could see that they do a good job maintaining a good balance between classes in the classifier. Ensemble methods like random forest and GBoost performed better in the negative class, making them not as useful for our medical application.

Challenges

A general problem regarding public datasets, such as the one used for this project, is sparse data. Sparse data refers to data that is incomplete and can play an effect on the ability to train the classifier and produce accurate predictions [2]. For example, our dataset had 4,407 unique vaccine adverse reactions. However, only 33 of those reactions were present in more than 2% of the cases. This means that health-care professionals, responsible for filling the entries in this dataset, might be focusing on only the most relevant reactions and omit those that they consider less important. This not only affects the classification but also the clustering algorithms.

Similarly, not only the adverse reactions attributes suffered similar sparsity, as well, when dealing with the medical history and allergies information of each individual, the lack of multi-select data and instead use of textual data presented the same challenge.

An additional challenge was that for the classification we were dealing with an imbalanced dataset. Imbalanced datasets, which are prevalent in many fields, and this includes healthcare, are those where there is a disparity of classes. The challenge appears in machine learning, when identifying these rare cases in big datasets. The algorithm will tend to categorize into the majority class (that one with more instances, in our case the class of not having a life threat reaction), while at the same time giving the false sense of a highly accurate model. To tackle this challenge we decided first to balance our sample using SMOTE, as discussed in the Approach section, which made our training data synthetically balanced. Additionally, when evaluating model performance we focused on recall and sensitivity to not be misled by the sense of high accuracy.

More on the dataset, it would be very helpful for this kind of analysis to have longitudinal data for each individual. Symptoms and reactions from a vaccine vary from the time taken the shot to

weeks or months after. An individual could have different types of symptoms across different days. This information would allow for a time-series clustering. In this dataset we know the number of days between the shot and the date of data entry, but we only have one snapshot for each individual.

Regarding memory complexity of the techniques used, we ran into out of memory problems when running Affinity propagation clustering. The same happened when trying to use a different affinity matrix (Jaccard difference and cosine difference) as input for a precomputed affinity matrix for Spectral Clustering. Computational complexity was also experienced. Finding the optimal parameters for learners like Random Forest, MLP, SVM and GBoost required a long time because of the dimension of our dataset.

Lessons Learned

This project was very helpful in showing us how to apply what we learned in our Machine Learning course to a real life example. Although, using a dataset such as this one was very time consuming in the cleaning and preparing phase. The nature of the data made us implement Natural Language Processing and Synthetic Minority Oversampling Technique. Additionally, we were careful to determine the right metric to evaluate our classification model with a highly unbalanced dataset.

Our main two goals of the project were achieved. First, we are now able to identify unsuitable candidates for a vaccine with a Precision-Recall AUC of 0.61 using MLP with regularization. In the future, implementing more sophisticated neural networks for this highly dimensional and sparse dataset such as the MLP in this project would yield probably even better results. Second, we were able to meaningfully cluster individuals based on symptoms and reactions from the

COVID-19 vaccine. This gives a holistic and comprehensive understanding to this new and relevant vaccine. Implementing the clustering algorithms was straight-forward, however, as with unsupervised methods, interpreting and visualizing the results consumed most of the efforts. Further work in the area of clustering both symptoms and time lapsed from the vaccine could result in a more detailed presentation of symptom clustering.

References

1. C. H. Sudre, K. A. Lee, M. N. Lochlainn, T. Varsavsky, B. Murray, M. S. Graham, C. Menni, M. Modat, R. C. E. Bowyer, L. H. Nguyen, D. A. Drew, A. D. Joshi, W. Ma, C.-G. Guo, C.-H. Lo, S. Ganesh, A. Buwe, J. C. Pujol, J. L. du Cadet, A. Visconti, M. B. Freidin, J. S. El-Sayed Moustafa, M. Falchi, R.Davies, M. F. Gomez, T. Fall, M.J. Cardoso, J. Wolf, P. W. Franks, A. T. Chan, T.D. Spector, C. J. Steves, S. Ourselin, *Symptom clusters in COVID-19: A potential clinical prediction tool from the COVID Symptom Study app*. Sci. Adv. 7, eabd4177 (2021).
2. Adomavicius, Gediminas, and Jingjing Zhang. "*Stability of Collaborative Filtering Recommendation Algorithms.*" Uiowa.edu, University of Iowa, 2010, dollar.biz.uiowa.edu/~street/adomavicius11.pdf.
3. Grus, J. (2019). *Data Science from Scratch*. (2nd ed., p.338). O'Reilly.
4. Larose, T. D. & Larose, D. C (2015). Data Mining and Predictive Analytics. (1st ed., p560 - p.562). Wiley.
5. He, H., & Ma, Y. (2013). *Imbalanced Learning: Foundations, Algorithms, and Applications* (1st ed., p. 47). John Wiley & Sons.

6. Chawla, N., & Bowyer, K., & Hall, L., & Kegelmeyer, P. (2011). *SMOTE: Synthetic Minority Over-sampling Technique*. https://arxiv.org/pdf/1106.1813.pdf.
7. Jussi Tohka, Mark van Gils. *Evaluation of machine learning algorithms for health and wellness applications: A tutorial.* Computers in Biology and Medicine, Volume 132 (2021). https://doi.org/10.1016/j.compbiomed.2021.104324.
8. Novaković, J. D., Veljović, A., Ilić, S. S., Papić, Željko, & Milica, T. (2017). *Evaluation of Classification Models in Machine Learning.* Theory and Applications of Mathematics & Computer Science, 7(1), Pages: 39 -. https://uav.ro/applications/se/journal/index.php/TAMCS/article/view/158
9. Yuan, Chunhui & Yang, Haitao. (2019). *Research on K-Value Selection Method of K-Means Clustering Algorithm*. J. 2. 226-235. 10.3390/j2020016.
10. Brownlee, J. (2017). *Master Machine Learning Algorithms: Discover How They Work and Implement Them From Scratch*. (1st ed., p. 99 - p.102). Machine Learning Mastery.
11. Brownlee, J. (2017). *Master Machine Learning Algorithms: Discover How They Work and Implement Them From Scratch*. (1st ed., p. 83 - p.88). Machine Learning Mastery.
12. Brownlee, J. (2017). *Master Machine Learning Algorithms: Discover How They Work and Implement Them From Scratch*. (1st ed., p. 52 - p.56). Machine Learning Mastery.
13. Brownlee, J. (2017). *Master Machine Learning Algorithms: Discover How They Work and Implement Them From Scratch*. (1st ed., p. 114 - p.119). Machine Learning Mastery.
14. Brownlee, J. (2019). *Master Machine Learning Algorithms From Scratch with Python*. (1st ed., p. 154). Machine Learning Mastery.
15. Brownlee, J. (2021, April 26). *How to Develop a Random Forest Ensemble in Python*. Machine Learning Mastery.

https://machinelearningmastery.com/random-forest-ensemble-in-python/.

16. Brownlee, J. (2020). *Imbalanced Classification with Python: Choose Better Metrics, Balance Skewed Classes, and Apply Cost-Sensitive Learning*. (1st ed., p.281 - p.283). Machine Learning Mastery.
17. Brownlee, J. (2018). *XGBoost with Python: Gradient Boosted Trees with XGBoost and Scikit-learn*. (1st ed., p.10 - p.13). Machine Learning Mastery.
18. Kassambara, A. (2017). *Practical Guide To Cluster Analysis in R*. (1st ed., p.177 - p.182). sthda.
19. Quinn, S. *Spectral Clustering*. http://cobweb.cs.uga.edu/~squinn/mmd_s15/lectures/lecture10_feb4.pdf
20. Rhys, I. H. (2020). Machine Learning with R, the tidyverse, and mlr. (1st ed., p.467). Manning.
21. Iliassich, L. (2016, May 19). *Clustering Algorithms: From Start To State Of The Art. Toptal Engineering Blog*. https://www.toptal.com/machine-learning/clustering-algorithms.
22. *Hierarchical Clustering*. https://online.stat.psu.edu/stat555/node/85/.