Application of Multi-Label Classification for Prediction of Adverse Reactions to COVID-19 Vaccinations

Grace Baek
Department of Management Science and Engineering
Stanford University
Stanford, CA 94305
graceb99@stanford.edu

Hee Jung Choi
Department of Data Science
Stanford University
Stanford, CA 94305
cheejung@stanford.edu

1 Introduction

As the world continues to grapple with the COVID-19 pandemic, vaccination has emerged as a critical tool in curbing the spread of the virus. Vaccines have historically been one of the most effective ways to prevent the spread of infectious diseases and save lives. However, vaccination hesitancy due to potential side effects has emerged as a major barrier in controlling the spread of COVID-19.

In our project, we build multi-label classification models that take in 49 features related to the type of vaccine administered, key patient measurements (e.g., age, height, weight, blood type, etc.), past COVID-19 symptoms, and underlying health conditions. These features are then used to predict whether a patient will experience the following 8 possible side effects of vaccination: fever, fatigue, headache, nausea, chills, joint pain, muscle pain, and injection site reactions. Given that there are various types of vaccines available worldwide, by providing patients with personalized information about the potential side effects of each vaccine, we allow patients to make an informed decision on their COVID-19 vaccination, potentially increasing vaccination rates.

2 Related Work

There has been a substantial amount of research done on utilizing machine learning on patient health data. For example, Jamshidi et al., using the same dataset as the one used in our project, also utilize similar machine learning models to create personalized predictions [1]. The paper presents a multi-label classification model for each vaccine type, while our project builds a single multi-label classification model that additionally takes the vaccine type as an input feature. The paper unfortunately achieved suboptimal results, arriving at an average ROC AUC score of 0.63. For instance, the paper utilizes stochastic gradient descent, which may not be the most effective optimizer for imbalanced datasets. Our algorithms aim to address some of the shortcomings in this paper.

In addition, Gonzalez-Dias et al. present a general data analysis procedure for predicting vaccine-induced immunity and reactogenicity using machine learning methods [2]. Specifically, the authors emphasize the importance of cleaning and normalizing the data, along with applying best machine learning practices to patient data. We aim to adapt some of these steps in our project. Finally, Ahamad et al. utilize another similar dataset with patient symptoms and characteristics as input features to produce a three-label classification model [3]. The model produced accurate results and utilized several metrics that were appropriate for a multi-label classification model on an imbalanced dataset. Combining the insights from the three papers proved to be fruitful.

3 Dataset

Our project utilized a dataset that was derived from the paper, *Personalized Predictions of Adverse Side Effects of the COVID-19 Vaccines* [1]. The survey conducted in the paper involved 90 hospitals located in Iran and Switzerland in 2020-2021 and was used to gather information about the adverse reactions experienced by patients who received COVID-19 vaccinations. For our project, we only used data from the first vaccination dosage.

The original dataset contained 59 columns: 8 columns were response variables (for the 8 different labels corresponding to a post-vaccination adverse effect) and the remaining columns were predictor variables, most of which were binary features. We observed that most of the data was from Sputnik V, AstraZeneca, and Sinopharm vaccines; there were 7,394 data points for Sputnik, 6,118 for AstraZeneca, 5,460 for Sinopharm, 814 for Covaxin, 556 for Moderna, and 280 for Pfizer vaccines.

3.1 Data Preprocessing

Data Cleaning Highly correlated features like `BMI` (correlated with the `height` and `weight`) and unknown features (labeled `Other`) were dropped. Multiple features—needle pain, needle redness, and needle swelling—were related to injection site reactions and were combined into one feature to reduce dimensionality. Continuous variables were normalized accordingly. We used one-hot encoding to indicate the type of vaccine that was administered. No missing values were in the data.

Iterative Stratification Train/Test Split Regular train/test splitting, which uses random sampling, may not be the best option for multi-label classification models for imbalanced data. In our project, we used iterative stratification to conduct train/test splitting so that label distribution was maintained, sampling bias was reduced, and model evaluation was improved. We split the data into training and test sets using a 75/25 split through iterative stratification, which gave us 15,466 training examples and 5,156 test examples.

Sparse Features Since the dataset is from Iran and Switzerland, where drug use is illegal or negatively stigmatized and alcohol consumption is considered forbidden for certain dominant religions in the region, `narcotics` and `alcohol` features were dropped, as these features were significantly sparse to have meaningful impact on the learning process. Additional sparse features like `Cancer` (0.6%), `Paresis` (0.8%), `Hematologic` (1.1%), `Consciousness Disorder` (1.2%), `Mental` (1.5%), `Hepatic` (1.8%) were identified. While sparse features can pose a challenge, these features were not removed, as they may be rare but critical events and can be handled with appropriate tuning and customization, as discussed in the next section.

The final dataset had a dimension of (20,622, 57). We sliced this data into predicting features (X) and response features (y), where X had a dimension of (20,622, 49) and y had a dimension of (20,622, 8), indicating that we have 49 predicting features and 8 potential output labels for the 20,622 examples.

4 Method

We implemented 2 machine learning algorithms, namely decision trees (including XGBoost) and neural networks, to build the best multi-label classification model for predicting the potential side effects of COVID-19 vaccines.

4.1 Decision Tree Algorithms

Decision tree algorithms build a tree-like model of decisions by recursively partitioning the input space into smaller regions based on the values of the input features. At each node of the tree, a decision is made based on the value of a particular feature, and the input space is split into two or more subspaces. The process continues until a stopping criterion is reached, such as a maximum depth.

Information gain evaluates the quality of a split on a particular feature and measures the reduction in entropy—or the measure of the impurity of a set of labels—resulting from a split on a given attribute. A higher information gain indicates that splitting on a particular feature results in a greater reduction in the entropy of the labels and is therefore a more effective way to partition the data. In our experiments, we used information gain as the criterion for selecting the best feature to split on at each node of our decision trees. The equation for information gain can be defined by:

$$\text{Information Gain} = \text{Entropy}(S) - \sum_{i=1}^{n} \frac{|S_i|}{|S|} \text{Entropy}(S_i),$$

where S is the set of labels at the parent node, n is the number of child nodes resulting from splitting on the feature, and S_i is the subset of labels at the i-th child node.

Simple Decision Trees When building our simple decision trees, we implemented the One-vs-Rest (OvR) method, where a separate decision tree model was trained for each label, dividing the multi-label classification problem into 8 smaller binary classification subproblems, where each decision tree model predicted whether a person will have certain COVID-19 vaccination side effects. The OvR method with decision trees has the advantage of being interpretable, as each label is associated with a separate decision tree model, but may not be as effective as other multi-label classification algorithms that consider the dependencies between labels. However, we favored the interpretability of using OvR classification in our models and made the assumption that the 8 response variables were uncorrelated. For this model, we tuned our trees on just one hyperparameter—max depth of the tree (10, 20, ... , 70)—using a manually implemented training loop that applied 5-fold cross-validation.

XGBoost XGBoost is an optimized implementation of gradient boosting decision trees that uses a gradient descent algorithm to minimize the loss function and combines the strengths of both decision tree and gradient boosting algorithms to achieve high predictive accuracy and speed. Instead of finding a set of optimal hyperparameters for all 8 response variables, we implemented a training loop that tunes hyperparameters for each response label individually for more accurate results. Our XGBoost model was trained using 4 hyperparameters—`max_depth`, `learning_rate`, `n_estimators`, and `scale_pos_weight`—which were selected using the grid search method.

4.2 Neural Networks

For a multi-label classification problem, neural networks offer several advantages that make them a suitable choice for this task. Neural networks can effectively learn non-linear relationships between input features and output labels and can handle high-dimensional sparse data with multiple input features. Our neural network consisted of the following layers:

- A dense layer with 256 units and ReLU activation function.
- A dropout layer with a rate of 0.5.
- A dense layer with 128 units and ReLU activation function.
- A dropout layer with a rate of 0.5.
- A dense layer with 64 units and ReLU activation function.
- A dropout layer with a rate of 0.3.
- An output dense layer with 8 units and sigmoid activation function.

In our case, we used the binary cross entropy as the loss function, which measures the dissimilarity between the true labels and the predicted probabilities. For a single training example, the binary cross entropy loss can be defined as:

$$L(y, \hat{y}) = -\frac{1}{N} \sum_{i=1}^{N} \Big[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \Big],$$

where y_i is the true label, $\hat{y}_i$ is the predicted probability of the label being 1 for the i-th adverse effect, and N is the number of adverse effects. The binary cross entropy loss is calculated for each of the 8 adverse effects independently, and the total loss is obtained by averaging the individual losses. This loss function is particularly appropriate for our problem, as it is designed to handle binary classification problems and can be easily extended to multi-label scenarios.

5 Results

5.1 Evaluation Metrics

We used 5 metrics to evaluate model performance: *precision*, *recall*, *F1 score* (all three of which were micro-averaged), *Hamming loss*, and *AUC ROC*. These metrics offer a balanced evaluation by emphasizing the importance of minimizing false positives and false negatives while accounting for overall model discrimination capabilities and the imbalanced nature of the dataset.

5.2 Decision Tree Algorithms

After tuning the decision trees, we determined that 10 was the optimal value for the `max_depth` hyperparameter. In Figure 1, both the F1 score and Hamming loss are highest when max depth is 10. Though these scores may seem contradictory, we determined the best max depth using the F1 score, as the F1 score is generally considered a better metric than Hamming loss for multi-label classification when the class distribution is imbalanced (since Hamming loss does not take into account the relative importance of each class).

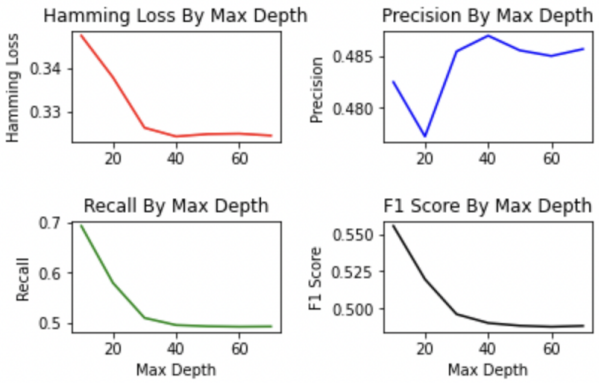

Figure 1: Evaluating *simple decision tree models* for different values of max depth

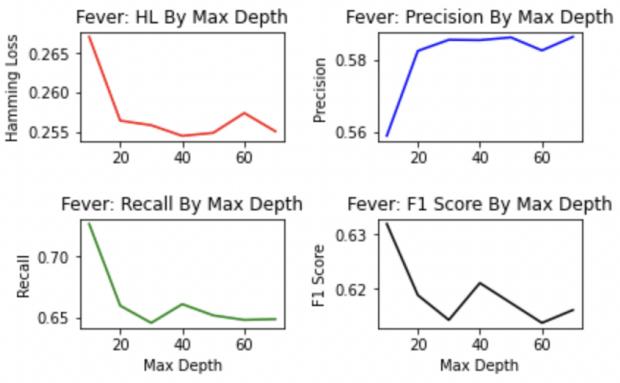

Figure 2: Evaluating *XGBoost models* (fever classifier) for different values of max depth

We performed tuning on four hyperparameters for the XGBoost model using the `GridSearchCV` function. Due to time constraints and limits to computing power, we did not run grid search on all possible combinations of the hyperparameters we tested but instead used the grid search function on a range of values for one hyperparameter at a time (while fixing the value for other hyperparameters), allowing us to efficiently search for optimal hyperparameters. Figure 2 visualizes how we tuned for the `max_depth` hyperparameter for one of the 8 XGBoost classifiers (in this example, we evaluated the model that predicted the fever symptom).

The best parameters for each of the 8 labels, which were validated using XGBoost's cross validation, are shown below in Table 1. Our models performed 5-fold cross validation and used accuracy—a ratio of correctly predicted observations to the total observations—to score each model tuned on a specific parameter value.

	Fever	Fatigue	Headache	Nausea	Chills	Joint Pain	Muscle Pain	Needle Reaction
max_depth	30	10	50	30	50	30	70	10
n_estimators	200	50	100	200	100	150	150	100
learning_rate	0.15	0.15	0.3	0.3	0.15	0.3	0.1	0.15
scale_pos_weight	2.17	0.95	1.96	8.95	2.74	2.56	1.51	0.72

Table 1: Optimally tuned XGBoost hyperparameters for each label classifier

5.3 Neural Networks

The layers in the neural network aimed to gradually reduce the dimensionality of the input data while extracting and refining abstract representations. ReLU activation functions in the hidden layers introduces non-linearity, allowing the model to capture complex patterns in the data and ensuring reasonable computation times. These additional hidden layers were beneficial, as training performance increased. To optimize the model, we employed the Adam optimizer with a learning rate of 0.001, as it provided an appropriate convergence speed and model stability with an adapative learning rate.

To account for the imbalanced nature of the dataset, class weights were computed and applied to provide a balanced representation of each class so that the model did not become biased toward the majority class. The model assigned a higher importance to the minority class instances, penalizing the model more when it made mistakes in classifying these instances. As shown in Figure 3, the best epoch number was found to be around 10, as beyond this, the neural network was over-fitting very heavily. For the general prediction threshold value, we determined that 0.3 was the best. In fact, to further improve the model, custom prediction thresholds were applied for each class by finding the threshold that maximized the micro F1 score for each class. While these custom thresholds were close to 0.3, they were in fact found to be 0.4, 0.25, 0.3, 0.2, 0.35, 0.3, 0.35, and 0.3 (from class 0 to 7, respectively).

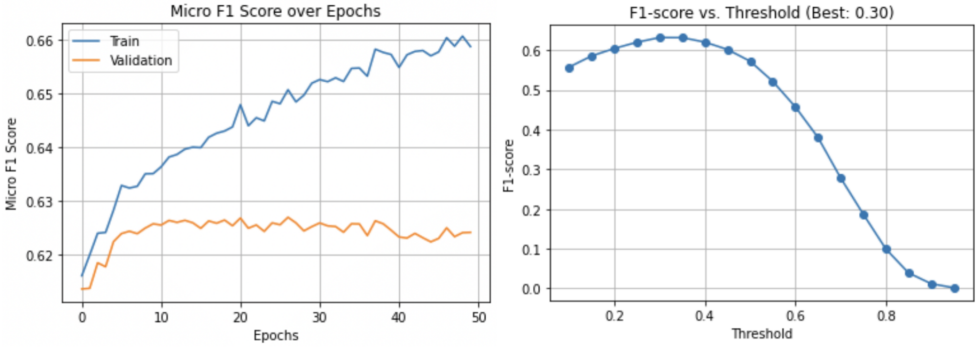

Figure 3: Training/validation F1 scores over epochs and prediction thresholds

As for the batch size, there were no significant findings, as shown in Figure 4. Nevertheless, the best micro F1 score was reached for a batch size of 16. In addition, as the model was clearly over-fitting, regularization was applied by having dropout for each hidden layer. Different dropout combinations were tested, and applying a dropout of 0.5, 0.5, and 0.3 for the respective hidden layers found to maximize the micro F1 score.

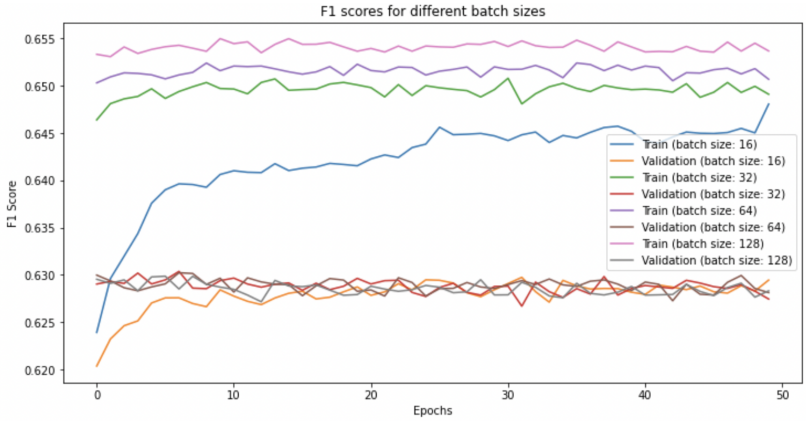

Figure 4: Training/validation F1 scores over epochs for various batch sizes

5.4 Model Comparison

The simple decision trees, XGBoost, and neutral network models we trained on yielded the results below.

	Precision	Recall	F1 Score	Hamming Loss	AUC ROC
Simple Decision Trees	0.4987	0.7423	0.5966	0.3505	0.6636
XGBoost	0.5976	0.6335	0.6150	0.2769	0.7023
Neural Network	0.5164	0.8236	0.6348	0.3317	0.7436

Table 2: Evaluation metrics from the test data for all three models (micro-averaged)

The neural network performed best in terms of recall, F1 score, and AUC ROC. These metrics are generally more important for multi-label classification, especially in the medical domain, where missing true positive cases can have severe consequences [4]. Although XGBoost had a slightly better precision and hamming loss, the neutral network's performance was more balanced.

5.5 Error Analysis

We conducted our error analysis on the neural network. The confusion matrices for each class are shown below in Figure 5.

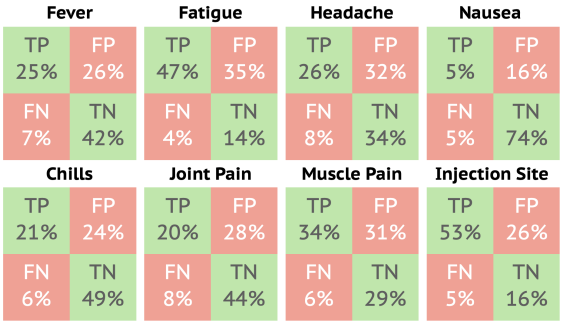

Figure 5: Confusion matrix for each class

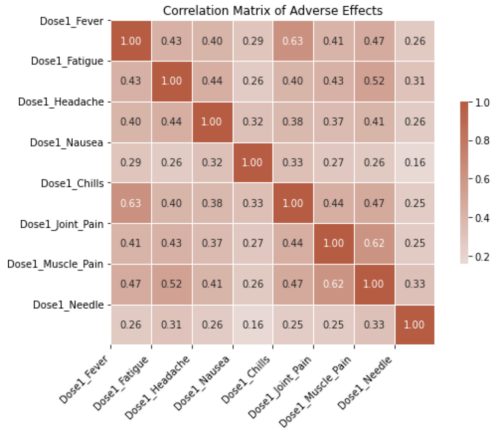

Figure 6: Pearson correlation matrix

The neural network faced difficulties in accurately distinguishing between certain adverse effects, especially those that are general symptoms or share similarities with other symptoms. The model demonstrated a relatively high number of false positives for most symptoms, except nausea (at a relative level), suggesting that it may be challenging to discern certain symptoms due to their frequent co-occurrence. Figure 6 implies that fever might co-occur with chills and, at times, muscle pain. The model might have learned that the presence of certain input features is associated with fever and other adverse effects. As a result, when the model encountered similar input features during prediction, it may have predicted fever even when it was not present, leading to false positives.

6 Conclusion

In this project, simple decision trees, XGBoost, and neural networks were employed to predict the potential side effects of COVID-19 vaccines, and their performance was evaluated using micro-averaged precision, recall, F1 scores, Hamming loss, and AUC ROC. The model performance was evaluated primarily based on the F1 scores, and the results indicated that the neural network outperformed the decision tree and XGBoost models, with an overall F1 score of 0.6348. The use of custom thresholds and class weights, as well as regularization techniques such as dropout, helped to address imbalanced data and prevented overfitting. Our project demonstrated that neural networks can be effective in predicting personalized side effects of COVID-19 vaccines.

6.1 Future Work

We obtained promising results in our project, with a neural network model achieving an AUC ROC score of 0.74, which is higher than the average score of 0.63 achieved by Jamshidi et al. [1] on their previous study using the same dataset. However, there are several avenues for future work to explore further. Future work might include exploring other machine learning algorithms, ensembling techniques, feature importance analysis, and tuning other hyperparameters, such as `subsample` and `colsample_bytree` for XGBoost and number of neurons per layer for neural networks. Collecting more data on the relatively sparse vaccine types, such as Moderna and Pfizer, would help create a more balanced dataset, and thus construct more robust models.

Contributions

Grace focused on developing decision trees, while Hee Jung worked on neural networks. Both Grace and Hee Jung contributed to data processing, result interpretation, and final report writing.

References

[1] Jamshidi, E., Asgary, A., Kharrazi, A. Y., Tavakoli, N., Zali, A., Mehrazi, M., Jamshidi, M., Farrokhi, B., Maher, A., von Garnier, C., Rahi, S. J.,& Mansouri, N. (2023). Personalized predictions of adverse side effects of the COVID-19 vaccines. *Heliyon*, *9*(1), e12753. https://doi.org/10.1016/j.heliyon.2022.e12753.

[2] Gonzalez-Dias, P., Lee, E. K., Sorgi, S., de Lima, D. S., Urbanski, A. H., Silveira, E. L., & Nakaya, H. I. (2020). Methods for predicting vaccine immunogenicity and reactogenicity. *Human vaccines & immunotherapeutics*, *16*(2), 269-276.

[3] Ahamad, M. M., Aktar, S., Uddin, M. J., Rashed-Al-Mahfuz, M., Azad, A. K. M., Uddin, S., ... & Moni, M. A. (2022, December). Adverse effects of COVID-19 vaccination: machine learning and statistical approach to identify and classify incidences of morbidity and postvaccination reactogenicity. In *Healthcare* (Vol. 11, No. 1, p. 31). MDPI.

[4] Parikh, R., Mathai, A., Parikh, S., Chandra Sekhar, G., & Thomas, R. (2008). Understanding and using sensitivity, specificity and predictive values. *Indian journal of ophthalmology*, *56*(1), 45–50. https://doi.org/10.4103/0301-4738.37595