Covid-19 severity prediction

ML phase IV report

Moneer Zaki
Computer Engineering
The American University in Cairo
moneerzaki@aucegypt.edu

Mario Ghaly
Computer Engineering
The American University in Cario
mariomamdouh@aucegypt.edu

ABSTRACT

In the previous phase of this machine learning project, the top 3 models for the problem at hand were KNN, Neural Networks(NN), and Support Vector Machine(SVM). After thoughtful analysis, although NN did not yield the best results yet, it was found to be the most suitable model based on this project, data nature, and the drawbacks of the dataset mentioned in the prior report. All the aforementioned prompted the choice of Neural Networks. Thus, the next step is to start designing the model and the utility application. This includes the process of modifying the parameters of the model and fine-tuning it till the model reaches the best performance metrics. These parameters include but are not limited to the number of hidden layers, the number of neurons in each hidden layer, the type of activation function in each hidden layer, the loss function used, class weights in the loss functions for handling class imbalance in the dataset, and overfitting analysis. Moreover, the final utility application design is going to be discussed in this report with different diagrams and a prototype to brainstorm and have an idea before the implementation of the final step that is going to make this project practical for real-life usage.

MODEL DESIGN

Model Choice

Choosing Neural Networks out of the best 3 models retained from the pilot study is due to the fact that NN has the ability to tackle non-linear relationships between the features and each other and between the label. Since most of the models that tackle only linear complexities did not perform well, this motivates the choice of NN. Although KNN yielded better Recall values and was at first picked as the best one, it is not scalable beyond the current dataset size, which is around 1 million rows, making it difficult to choose larger values for K. Also, it would be very slow in prediction. On the other hand, NN has way more parameters to be fine-tuned. These parameters will be discussed in the upcoming parts. Regarding SVM, it is very slow -- it was the model that took the most time during the pilot study -- and NN has more parameters that could be changed in various ways to make the model suitable for the problem.

Model Design Process

After model choice, the next step is to design the model in a way that best fits the data. For a comprehensive model design, the chosen model would be experimented with using the Tensorflow library as it is faster than sklearn, it has more manipulation over different parameters, and it allows different activation functions in different layers. The model will be tried with different hyperparameters, and the performance metrics of each choice of hyperparameter values will be compared -- focusing on F1-score and Recall -- and the best hyperparameters will be chosen. K-cross validation would be applied like in the prior study, with k = 5, not 10 to be able to experiment with more combinations of the hyperparameters. Furthermore, in each choice of hyperparameters, overfitting would be put into consideration by comparing the training and validation performance metrics as would be illustrated next. The target is to tune the following parameters as following:

1. **NN Architecture:**
    a. **The input layer** should have 30 neurons, for the number of features in the dataset.
    b. **The hidden layers** will be experimented with 1 and 2 layers since the more layers, the more overfitting could occur. This is why in every choice of hyperparameters, there would be reporting for performance metrics on the training set and the validation set to compare if the training performance is way better than the validation performance, this would indicate overfitting in some sense. A final note is that 3 layers could be tried in case no satisfactory results were obtained, but this is highly unlikely.
    c. **The first hidden layer** would be tried for different values of neurons starting from 1 and increasing by multiplying by 2(e.g., 2,4,8) as it is more common that the number of perceptrons is a power of 2. The second layer would be the same until it reaches the same number of neurons as

the first hidden layer to avoid overfitting, reduce training time, and for hierarchical representations. For both hidden layers, the number of neurons would not be increased when the training performance is increasing and the validation performance is decreasing.

d. **The output layer** is going to be 1 perceptron since there is only one binary label. However, it is going to be 2 only in case of applying the softmax activation function instead to get the probability for each label value instead.

e. The activation functions to be experimented with in the hidden layers are sigmoid, Rectified Linear Unit(ReLU), and identity functions. For the output layer, it could be ReLU, sigmoid since it is convenient for binary classification problems, or softmax to play with the threshold values of each probability.

2. **Learning Rate:** Learning rate is an important parameter as it affects how fast the model converges. This parameter would be tried when constant and when adaptive for the same values to compare and find which converges faster and produces better results without overfitting.
3. **Optimizer:** Optimizer is the algorithm used to update the weights of the model during training to minimize the loss function. The optimizer algorithms that are going to be tested are stochastic gradient descent(SGD) since it is simple, efficient, and tries to avoid getting stuck at local minima, adaptive gradient(AdaGrad) which is good when features have varied importance, and adaptive moment estimation(Adam) as it is commonly used for distinct datasets and it is the default.
4. **Regularization:** This parameter is a set of techniques used to prevent a machine-learning model from overfitting, which is important since NN is prone to overfitting. NN tends to memorize the training data instead of generalizing patterns that can be applied to unseen data. Regularization methods introduce additional constraints or penalties to the optimization to prevent this by encouraging the model to learn simpler patterns.
   The regularization algorithms going to be tested are L1 and L2 only since the rest are believed to be irrelevant to the problem at hand.
5. **Batch Size & Epochs:** The batch size refers to the number of training examples used in each iteration of the optimization algorithm. Instead of updating the model parameters/weights after computing the gradients for the entire training dataset, they are updated after processing a small subset of the data called the batch. Smaller batches can lead to noisy gradients and slower convergence, while larger ones provide stable gradients but require more memory and computation.
   Regarding epochs, they represent the number of times the entire dataset is going to be iterated on during the training process. This hyperparameter needs to be determined based on the convergence behavior of the model. Too few epochs may result in underfitting, while too many epochs may lead to overfitting.
   For the current project, batch sizes are going to start with values ranging from 32 to 256, in powers of 2, and epochs ranging from 10 to 100, with increments of 10.
6. **Loss Function:** This hyperparameter is very critical as it determines how the model's performance is evaluated and how the gradients are computed during backpropagation. Moreover, the loss functions have the power to address the class imbalance, which is a problem in this dataset. For the previous reasons, focal loss function is going to be tested, which is a variant of cross-entropy designed to address class imbalance by down-weighting easy examples and focusing on hard examples. Secondly, weighted cross-entropy would be tested as it is similar to binary cross-entropy, but allows for assigning different weights to positive and negative classes to account for class imbalance as well. Assigning higher weights to the minority class increases their contribution to the loss function and makes them more effective in weight choice.
7. **callbacks=[early_stopping]:** This is the last parameter to be changed from its default value. it is going to be monitoring a specific performance metric of the model on a validation dataset and stopping training when the performance starts to degrade. The performance metric to be monitored is the F1-score, as it is the most representative metric for the performance in this problem.

UTILITY APPLICATION DESIGN

Technology Choice

Having a **website** for a COVID-prediction service offers several advantages **over a mobile application**. Firstly, it provides accessibility across various devices without the need for downloading and installation. Secondly, web applications are easier to update and maintain, ensuring that the AI model stays current with the latest medical data and research. Additionally, a website allows for seamless integration with other online resources such as medical databases or information portals, enhancing the service's overall utility and reliability.

For implementing the website, Django was chosen as a high-level web framework written in Python that enables rapid development of web applications.

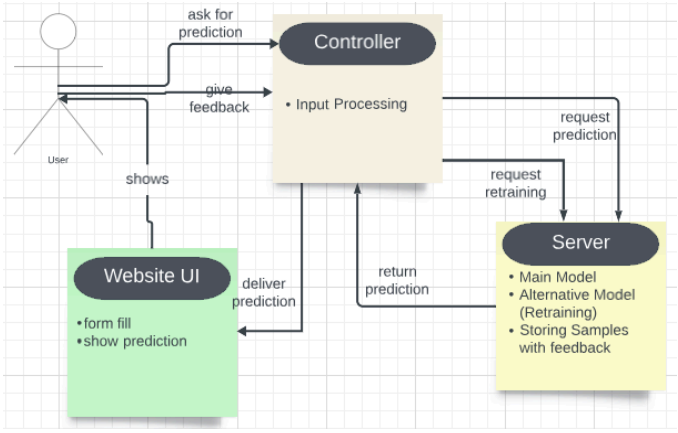
Django Choice Reasons

Using Django as a server offers several advantages. Firstly, it's Python-based, providing a familiar and powerful language for development. Django's utilization of SQLite for database management streamlines database handling. Being a full-stack framework, Django covers both front-end and back-end development aspects. It seamlessly integrates with machine learning models for predictive analytics, making it suitable for data-driven applications. Furthermore, Django's ORM simplifies database interaction by allowing developers to work with Python objects. Moreover, Django comes with built-in security features to safeguard against common web vulnerabilities. Additionally, it offers scalability, effortlessly handling large volumes of traffic and data, making it an excellent choice for growing applications, like this one -- hopefully.

Architecture (Client/Server)

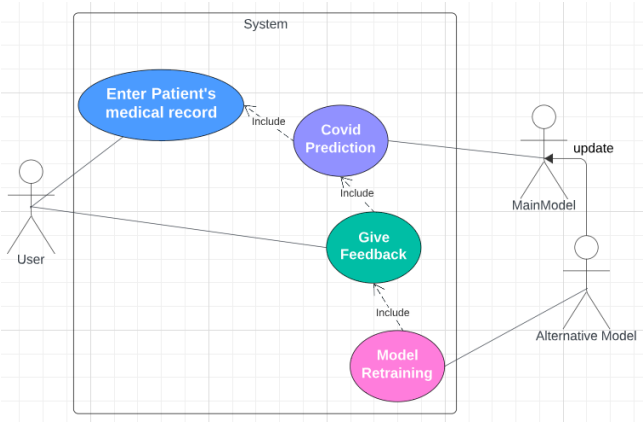
In this implementation using Django, **client/server architecture** was opted for to facilitate seamless interaction between users and the AI-powered prediction system. By having the client fill in a medical form and request predictions from the server, there is the advantage of centralizing data processing and leveraging the computational power of the server to execute complex AI algorithms efficiently(NN). This architecture enables users to access the prediction service from various devices with ease, without requiring heavy processing on their end. Moreover, the server's role in using the uploaded AI model ensures consistency and accuracy in predictions across all clients.

Employing the MVC design pattern within this architecture enhances maintainability and scalability. The MVC design for this project is displayed in the figure below. The model component manages the AI model and handles data manipulation, ensuring the prediction system remains up-to-date and adaptable to new information. The view component presents the user interface for form filling and result display, providing a seamless experience for users. Finally, the controller component orchestrates communication between the client and server, facilitating data exchange, prediction requests, and model retraining. Not only does this separation of concerns enhances code organization and readability, but also simplifies future modifications and updates to the system, making it robust and efficient for both users and developers alike. Here, there is UI for filling the form, and displaying things like the prediction. There is also a controller to connect the GUI with the server, and it takes the requests for prediction and the feedback given to send them to the server. For the server, it has the main model, which is always responsible for prediction to be always available for users to get a prediction. The server also has an alternative model that uses the stored samples with feedback to retrain using transfer learning, and then the main model is updated to be like the alternative when it finishes retraining.

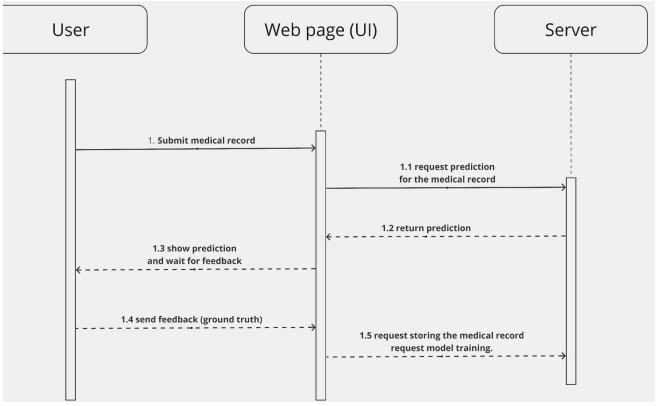


Human Interaction Flow

A use case diagram outlines the system's functionalities and user interactions. It helps in understanding user requirements and system behavior by visualizing scenarios like form filling, prediction processing, and model retraining. In this case, the diagram below facilitates the interaction between the user and the server model. It shows all possible options that could be done by the user, which is submitting his/her medical record and giving feedback on the prediction received by the model. On the other hand, the server is responsible for predicting the client case, covid holder or not, and requests for retraining the model after a periodical interval of time.

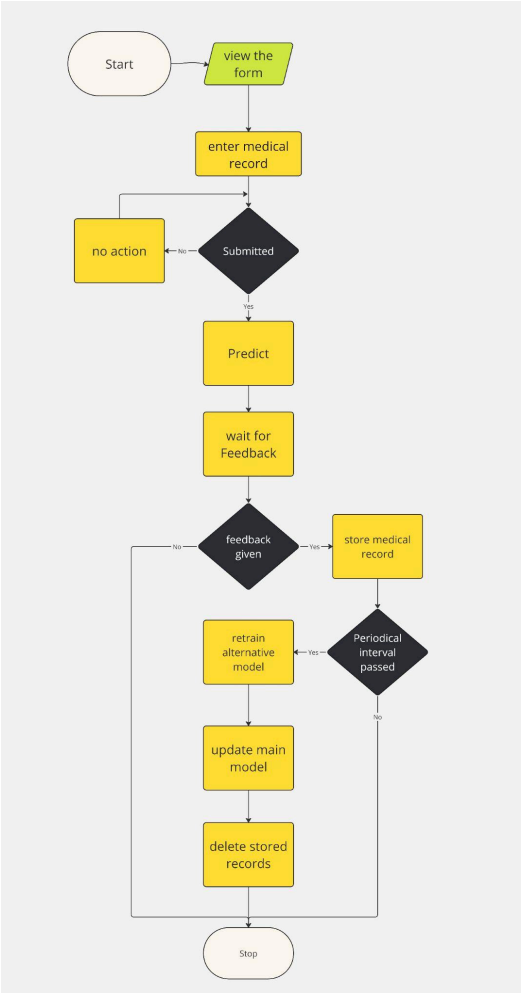


Furthermore, a Sequence Diagram depicts the interactions between system components over time. It visualizes the sequence of messages exchanged between objects or entities, illustrating the dynamic behavior of the system. This diagram clarifies the order of operations in the prediction process, from user input to result delivery and model retraining. It assists in understanding complex interactions and identifying potential issues or optimizations in the system's workflow. It is added to highlight the human interaction flow more.



Data Flow

The Data Flow Diagram illustrates the flow of data within the system. It shows how user input moves through processes, data stores, and external entities, aiding in understanding data movement and processing. It provides a structured representation of data flow, aiding in system documentation and design. The below diagram shows this for the project desired. The user will view the form, enter the medical record, and when they submit it, the model will then predict if they have COVID-19 or not. After that, there will be a question for the user if they want to give feedback, and if not given, nothing happens. Otherwise, the medical record would be stored for future retraining of the model when the periodical interval passes.

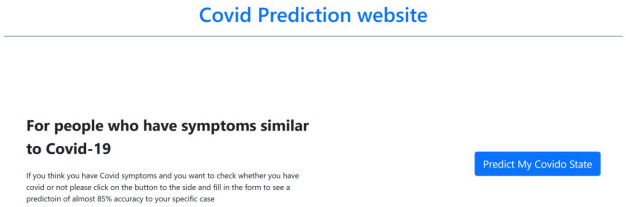


Prototype

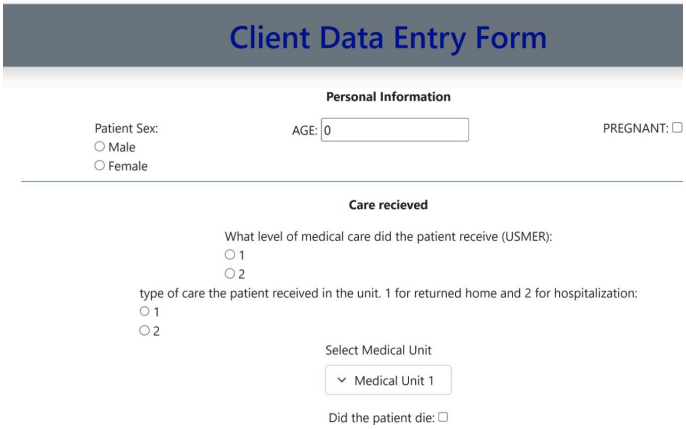
The website is currently being finalized with all of its facilities. Equally important, the user experience is taken into consideration to be as good as possible by making the website accessible and user-friendly. This website will be published online for as many users as possible to use and get a very high percentage prediction on their cases.

So far, the website starts with a very simple Homepage with no registration nor login needed. There is no need to create a user profile or ask for any login details to

keep the process of Covid prediction as fast and simple as possible. The image below illustrates the homepage.



Once the patient enters the form, a Medical Record form is presented in the web page as follows categorized into sections and subsections to facilitate the process for the user and make it more user-friendly.



The rest of the medical record with a submit button is shown next. Once the user submits the medical record the web page requests a prediction from the server. As a response to that, the server uses the AI model, which in our case would be Neural Networks AI model to predict the patient's case and returns the prediction back to the website to be presented to the user. Once the prediction is out, the website waits for the user to enter the ground truth value of its covid case, if known by the user.