THE OHIO STATE UNIVERSITY

Treat First What Kills First: Predicting Disease Severity with DDXPlus

Project Category: Table (Numerical/Categorical)
Physics 5680, Autumn 2024

Author: Leandra Hogrefe

December 12, 2024

Abstract

The accurate and fast diagnosis of patients is a crucial part of ensuring they get the best possible care and highest chance of recovery. With machine learning on the rise, there has been an increasing effort to support medical staff with algorithms' abilities to analyze connections and patterns within data and to recognize these trends when presented with new data. Especially when diseases are potentially life-threatening it is important to act fast to recognize the risk. This project focuses on using the DDXPLUS data set to classify disease severity to be able to rule out severe diseases over harmless ones. To achieve this, I will trained a Random Forest Classifier and a Convolutional Neural Network (CNN) on the DDXPlus patient data. In terms of performance metrics, the CNN outperformed the Random Forest significantly but required significantly more ressources to do so. In the context, of medical diagnosis accuracy is very important but so is time. This opens up space for future projects to find ways to create models that are both fast while still being accurate to support doctors and save patients' lives.

1 Introduction

In the medical field, the accurate and timely prediction of disease is essential to ensure patients have the best possible chances of getting healthy. At the same time, medical professionals are needed more than ever and many medical institutions are short-staffed increasing the work load of each doctor and nurse. With machine learning on the rise, there have been many efforts to support medical professionals in their diagnostic tasks to reduce said workload. While these algorithm could never replace the interpersonal tasks that medical staff perform from day to day, they can learn be supportive in some of the knowledge skills, for example by coming up with differential diagnoses or predicting/ruling out severe diseases. This project makes use of the DDXPLUS data set to explore exactly this opportunity. How well can a machine learning model predict a disease's severity or even the exact pathology? The data set comes with a wide variety of evidence assigned to each virtual patient. I will use this evidence as input features for a Random Forest Classifier as well as a Convolutional Neural Network to examine how well they can classify patients into disease severity classes.

2 Related Work

Due to the high relevance of medical diagnostics, there have been many papers looking into ways too find algorithms to better assist with the diagnostic process. Some of these studies have been using DDXPlus because it contains a large amount of data and does not only list the pathology (final diagnosis) but also a differential diagnosis assigning probabilities to different diseases as well as having a severity index.

Nassiwa et al. uses DDXPLUS as one of the data sets for developing a "Predictive Diagnosis Assistant", a machine learning tool to assist doctors' diagnoses. They used different approaches of Principal Component Analysis, Correlation Analysis, Feature Importance and Chi Squared to identify the most relevant features. These features were then used to train their model [4]. Their detailed look into the features of DDXPlus before even fitting a model stands out. While I will not be able to do something to this extent, I will look into what the symptoms are and if that makes sense. They particularly marked that advanced methods such as deep learning or reinforcement learning will be necessary to realistically implement these techniques.

Tchango et al. uses a deep reinforcement learning framework to generate a differential diagnosis with a prioritization of severe pathologies [5]. In their differential diagnosis process they include through a exploration-confirmation process mimicking the interaction a doctor would have their patient, asking questions based on the assumptions in which direction the diagnosis could go. As opposed to other papers, they emphasize the particular importance of identifying diseases of high severity which is what I am looking to do with this project.

Zhou et al (2023) uses a Convolutional Neural Network in combination with ensemble learning for chronic disease diagnosis [1]. Their data set is the real-world Electronic Medical Records dataset (C-EMRs). Like DDXPlus this data set is imbalanced, so there is a wide variety in the number of records for each disease. They point out that CNN's Softmax classifier has the flaw that it "will directly select the category with the highest classification results, without considering other categories, which will lead to other potentially correct disease diagnosis results being directly ignored by the classifier, increasing the possibility of misdiagnosis". To lessen the impact of this shortfall, they combined the CNN with ensemble learning. In this method, an ensemble of weaker classifiers comes together to build one strong classifier. Their methods resemble mine in the way that I am implemnenting a CNN and a Random Forest which is an ensemble learning method, however they go the extra step to combine the two for increased performance.

Overall, almost all work that aims to implement these diagnostic algorithms for real-life application seems to go towards using more advanced machine learning methods. I read the keywords deep learning, reinforcement learning and ensemble learning many timess.

3 Dataset

The DDXPLUS data set consists of five files: three files of patients with their personal features and medical conditions (split into 1,025,602 train patients, 132,448 validation patients and 134,529 test patients set totaling about 1.3 million patients) as well as one file each that lists the conditions and the evidence to get some background information on the pathology and to decode the ID numbers associated with each symptom and antecedent.

The patients in the dataset are artificially generated to ensure individual anonymity. Fig. 1 shows how the "patients" are generated. The dataset is imbalanced in pathologies as well as disease severity. Fig. 2 shows the occurrence of diseases by age group and the different colors show the fraction of each severity class. The differential diagnosis in the patient tables lists possible pathologies and possibilities that the respective patient has this disease. These results are generated by "a real-world telemedicine platform" [3]. The pathology then lists the name of the actually diagnosed disease. In most cases, these two agree showing that the model used for this prediction did a good job.

Apart from several interesting tendencies associated with different age groups, e, g, extremely strong representation of patients less than 1 year old, Fig. 2 also shows the distribution of pathology severity. The intermediately severe diseases are most common (especially 3 and 4 are well represented) while very severe.

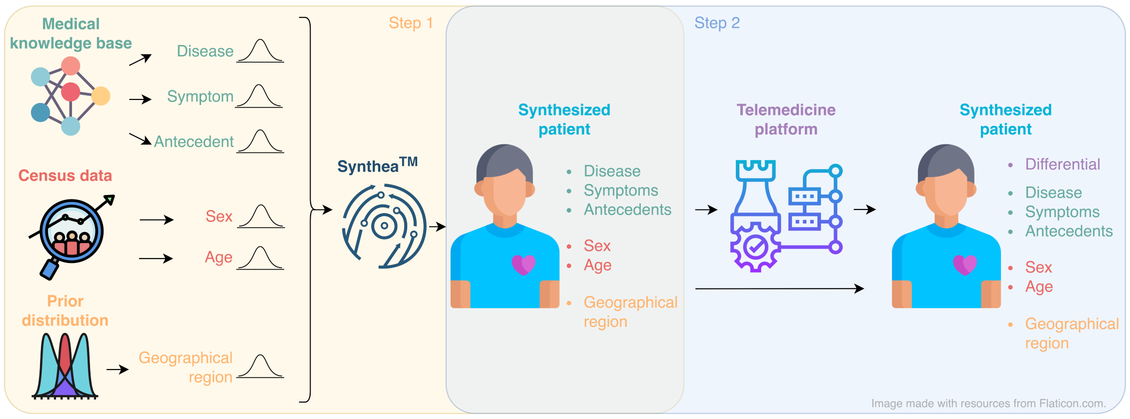


Figure 1: Creation of the DDXPlus Patient based on real Mediacl Data [2]

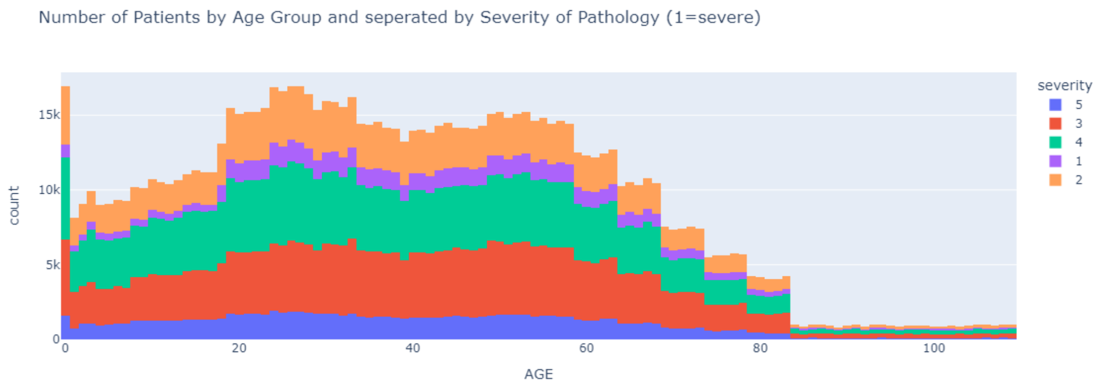


Figure 2: Histogram of Training data by age and severity separation

3.1 Converting the features

There are 223 symptoms and antecedents (evidence) in the DDXPLUS data set. Out of those 208 are binary (shows evidence yes/no) and 15 can take different values (e. g. pain on a scale from 0-10, classifying the type of pain, etc.). To use these as features, we one-hot encode the evidence. This process leads to a different shape for every patient table as soon as one piece of evidence is missing added. Too keep a consistent shape, we will only use the train file from DDXPlus because one million patients leaves us with enough to train even when splitting it into train and test sets. The binary evidence was already normalized from 0 to 1 but the numerical evidence initially went from -1 to 10. Here, I adjusted the range from -1 to 1.

4 Methods

The DDXPlus data set offers a large number of patients with symptoms as well as their labeled severity and pathology. Since we have these known classes, we will use supervised machine learning algorithms to assign new data to these same classes.

4.1 Random Forest

Just like a forest is a collection of many trees in nature, the Random Forest is a collection of many Decision Trees in Machine Learning. A Decision Tree contains a number of criteria based on which the data is separated into different categories. With each layer of the Decision Tree, the number of nodes increases and the separated categories become cleaner making for a better classification. However, this method is very prone to closely adapting to the training data it is given. The over-fitting leads to a relatively poor performance when applying to unseen test data. The Random Forest alleviates these issues by combining many different decision trees [6]. This leads to an increasingly complex structure of the model which comes at the cost of requiring more computational resources during the model fitting process.

To train the different trees a method named "Bootstrapping" is first applied to randomize the data that every tree sees. This makes sure that the different trees capture different structures in the data and they do not all over-fit to the same data leading to bad performance results of the forest. To further increase the differences between the trees, the trees "grow" based on different features. The number of features is the square root of features (default) or the log based 2.

At the end of each patient, all trees (that saw the data) output a predicted class. The random forest then uses the result decided by the majority of trees as the final output (hard voting). Alternatively, the output probabilities associated with each class can also be averaged and the final class is the one that is most probable on average (soft voting). While random forests are complex and a lot more difficult to picture than decision trees, Fig. 3 can show how the trees come up with the class prediction resulting in a final classification though it does not capture that not every tree sees the whole data set and that the trees can be structures differently, for example, because of the different features that they are using.

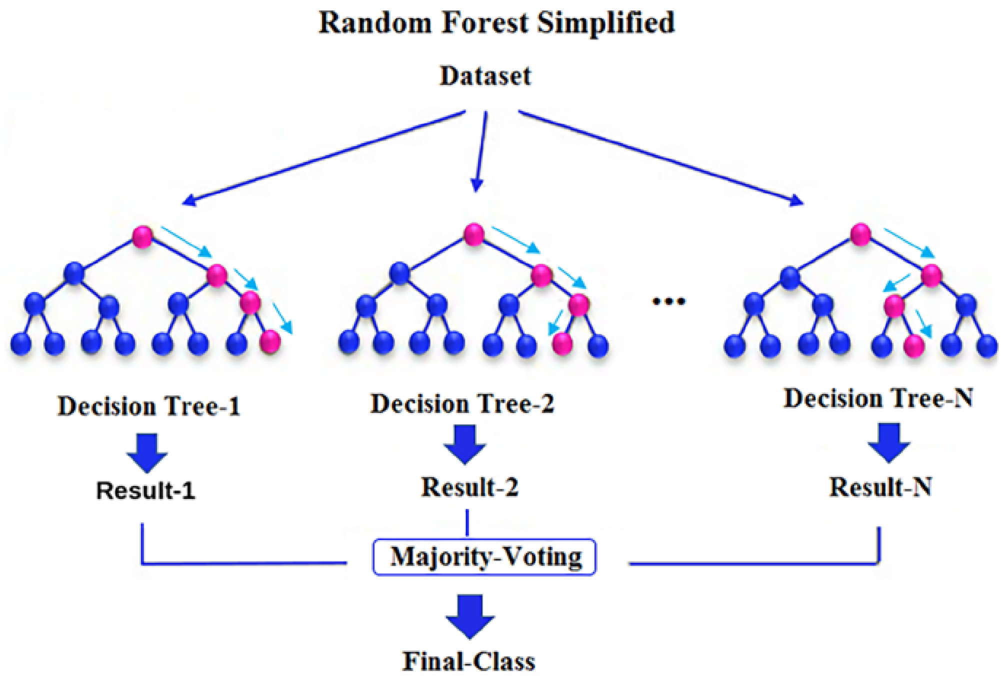


Figure 3: Simplified Illustration of the classification process pf a Random Forest Classifier via its decision trees [7]

4.2 Convolutional Neural Network

As a second approach, I chose to use a Keras network to also classify the disease severity within DDXPlus. I decided to use a CNN rather than a FCN because CNNs usually require a lot of data until they perform well and I have a very large data set available. CNNs are very good at learning patterns which we illustrated in class based on the example of the numbers in the MNIST data set [8]. The idea for this application is that the one-dimensional vectors will have similar patterns for similarly severe diseases.

As the name suggests, CNNs take advantage of a process called "convolution". In this process a kernel moves across the input multiplying the features by the kernel resulting in a convolved feature. The kernel can be sensitive to different structures in the data which is how it can recognize patterns. However, the kernel is not something that can be set. The model learns what the kernels are during the fitting process and can then apply them to find the same structures in the testing data. Another important feature of the network's architecture are pooling layers. The pooling layers further decrease the size of the vector by conserving only the max value out of a number of features. This identifies where high as opposed to low values are increasing translational invariance, so it saves the structure but not the exact position. At the end of the model, there are a global average pooling layer and a dropout layer both of which effectively resuce overfitting, one by averaging, one by setting as fraction of values to 0. The dropout layer can also lead to the validation accuracy being higher than for the training because some training data is dropped.

Another benefit of the CNN is that it allows to look into feature importance which is interesting in the context of determining the disease severity. I chose to normalize the data before handing it to the CNN. Most columns are binary and therefore normalized on a 0 to 1 scale immediately after one-hot encoding. The numerical evidence however was previously on a scale of -1 to 10 (where -1 is not applicable and 0 to 10 is least to most for the respective symptom). I scaled this to be from -1 to 1 by dividing all positive values by 10.

4.2.1 Activation Functions

Activation functions play an important role in neural networks. They introduce non-linearity allowing the networks to describe more complex structures than a linear regression [9].

The CNN uses the ReLU (Rectified Linear Unit) function for the hidden layers and the Softmax function for the output layer.

Eq. 1 shows the mathematical definition of the ReLU function:

$$A(x) = max(0, x) \tag{1}$$

ReLU outputs positive values when the input is also positive but all negative input leads to an output of 0. The mathematical operations for this functions are relatively simple making it less computationally extensive than the sigmoid and tanh functions.

The output's activation function Softmax is defined as follows:

$$softmax(z_i) = \frac{e^{z_i}}{\Sigma_j e^{z_i}} \tag{2}$$

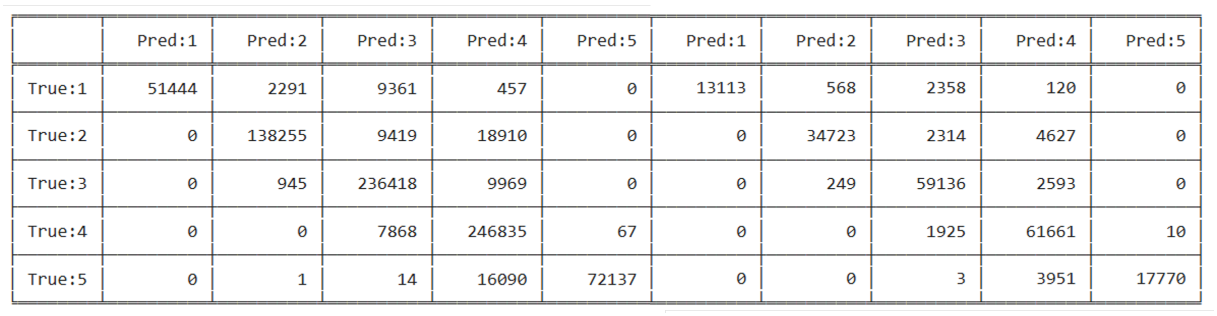
As an activation function for the output layer, it has the important takes of putting the results of previous layers in a context that makes sense. The sum in the denominator of in Eq. 2 already gives an idea of what the function does: it returns a fractional probability. This means that the final result can be interpreted as a fraction of 1 making it an immediately obvious probability [10].

5 Results/Discussion

When designing the models for this task, I tried to keep the models relatively simple while maintaining good performance metrics (precision or accuracy over 90%).

5.1 Random Forest Classifier

I used a Random Forest Classifier with the number of trees set relatively low at 50 to keep the model simple and a max depth of 7. I initially tried several max depths for a subsample and 7 was where the error for recall and f1 score stops decreasing significantly. I could not reproduce that result later with larger samples but was satisfied with the performance of the model. For the severity classification, the classifier immediately did a good job. Precision, recall and F1 score were all in the lower 90% range with a time of under 10 minutes. I then decided to fit the same structured model to the pathologies (49 instead of 5 classes). Unfortunately, these results were not as good. For some classes the model performed very well, however for other pathologies not a single patient got sorted into right category. This also led to problems with the metrics because they often require to divide by the number of correct identifications which led to some divisions through 0. This was likely because some diseases ”look” similar resulting in almost all patients being misclassified into the same category. The model seemed not advanced enough to capture the nuances differentiating these diseases.



	Pred:1	Pred:2	Pred:3	Pred:4	Pred:5	Pred:1	Pred:2	Pred:3	Pred:4	Pred:5
True:1	51444	2291	9361	457	0	13113	568	2358	120	0
True:2	0	138255	9419	18910	0	0	34723	2314	4627	0
True:3	0	945	236418	9969	0	0	249	59136	2593	0
True:4	0	0	7868	246835	67	0	0	1925	61661	10
True:5	0	1	14	16090	72137	0	0	3	3951	17770

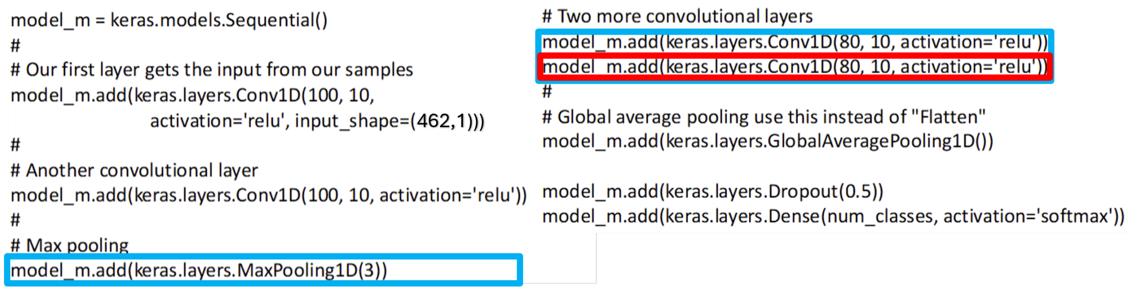
Figure 4: Combined Confusion Matrix for Random Forest train (left) and test sets (right), labels range from most severe/1 to least severe/5

Additionally, I decided to look into feature importance. The number of features for the model is large which results in a low importance of a singular feature. The combination of all features makes it possible for the classifier to perform well. The most important features had an importance of about 2-3%. I was surprised to see that these classifiers came from many different areas which I am attributing to the fact that these different questions can narrow down the ”field” of diseases significantly.

The metrics were very similar for train and test data. I also ran a k-fold validation with 5 folds to get another metric but kept the number of folds low to keep the model simple. Finally, I ran the model with the whole DDXPlus train file to get the two confusion matrices shown in Fig. 4.

5.2 CNN

As an alternative approach, I chose to use a Convolutional Neural Network (CNN). I chose an epoch of 20 which is lower than for previous cases but given the large data set, the fitting time is already going to be long and it turned out that the models were able to achieve good results within 20 epochs without a problem.



```
model_m = keras.models.Sequential()
#
# Our first layer gets the input from our samples
model_m.add(keras.layers.Conv1D(100, 10,
                  activation='relu', input_shape=(462,1)))
#
# Another convolutional layer
model_m.add(keras.layers.Conv1D(100, 10, activation='relu'))
#
# Max pooling
model_m.add(keras.layers.MaxPooling1D(3))
# Two more convolutional layers
model_m.add(keras.layers.Conv1D(80, 10, activation='relu'))
model_m.add(keras.layers.Conv1D(80, 10, activation='relu'))
#
# Global average pooling use this instead of "Flatten"
model_m.add(keras.layers.GlobalAveragePooling1D())

model_m.add(keras.layers.Dropout(0.5))
model_m.add(keras.layers.Dense(num_classes, activation='softmax'))
```

Figure 5: Sequential Standard Model adjusted from[11]. The red box marks the layer removed for the 2nd model in Table 1 and the blue boxes are the layers removed for the model in column 3

Here, I spend some more time with the network architecture. Once again I tried to design a minimalist network. However, the accuracy for test and validation sets suffered significantly from removing convolutional and max pooling layers. I started with the Standard Sequential Model described in Module 8 [11], adjusted the input and output shapes, then proceeded to remove layers as shown in Fig. 5 and recorded the impact on the performance metrics. Table 1 shows the validation accuracy for the three architecture choices. All three start at the same accuracy but while the first model increases very quickly (note that not every epoch is shown in the table), the third column shows much less increase in accuracy.

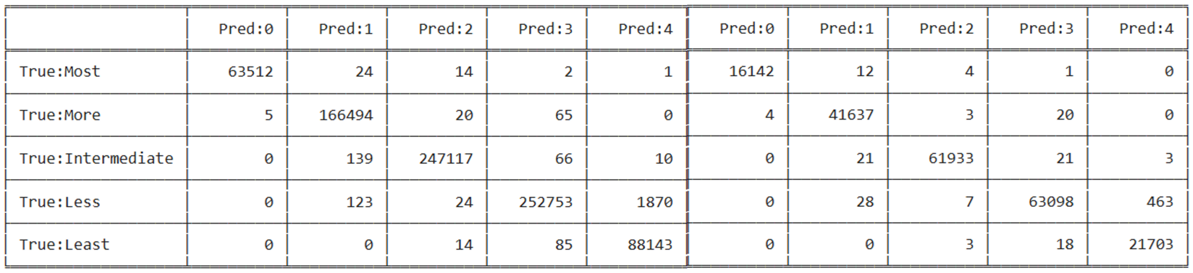
Epoch	Sequential Standard Model	- 1 Conv1D	-2 Conv1D & - MaxPooling
1	0.315	0.315	0.315
2	0.321	0.326	0.320
4	0.563	0.496	0.354
8	0.907	0.683	0.451
12	0.931	0.816	0.502
16	0.971	0.905	0.512
20	0.975	0.932	0.548

Table 1: Validation Accuracy at select epochs for different CNN architectures fitted to a subset of 8,000 patients with 2,000 validation patients. The first is the most complex is the left, the most simple is on the right. Fig. 5 shows where layers were removed.

Even though I expected (and after training on 800,000 patients I can confirm this) that the accuracy would increase when being trained on more data, the subset-based performance of model 3 led to the decision not to test it on the entire dataset. I did however test model 1 and 2 with the following results: The training on 820,000 patients took about 2h 53min (20 epochs) for model 1 and 1h 47min (14 epochs, early stopping) for model 2. The final validation accuracies were about 0.9977 and 0.9970.

I chose the loss function of "sparse_categorical_crossentropy" because it does not require one-hot encoded labels. I did change the severity labels from 0 to 4 instead of 1 through 5 because the function expected the five values to start at 0. This does not have any impact, except the labels of the confusion matrices for test and train sets in Fig. 6.

For my optimization criterion I chose the validation accuracy which in my case is really the test accuracy because I passed the test data as validation data. The accuracy is the fraction of the correct predictions out of all predictions made. The validation accuracy ended up being higher than the test accuracy during a test run of the model. This was due to the Dropout component of the model which **does something**.



	Pred:0	Pred:1	Pred:2	Pred:3	Pred:4	Pred:0	Pred:1	Pred:2	Pred:3	Pred:4
True:Most	63512	24	14	2	1	16142	12	4	1	0
True:More	5	166494	20	65	0	4	41637	3	20	0
True:Intermediate	0	139	247117	66	10	0	21	61933	21	3
True:Less	0	123	24	252753	1870	0	28	7	63098	463
True:Least	0	0	14	85	88143	0	0	3	18	21703

Figure 6: Combined Confusion Matrix for Model 2 train (left) and test (right) sets, labels range from most severe/0 to least severe/4

T

6 Conclusions/Future Work

In predicting disease severity classes, the CNN outperformed the Random Forest. However, it required significantly more resources to do so. Using a simple and faster to train CNN architecture led to results that were significantly worse. The Random Forest performed not quite as good but was significantly quicker to train (10min compared to 1h 47min) and to evaluate test data. Here, we have to not that the performance metrics can be misleading: a precision of 92% does not seem that much lower than 99% but comparing the confusion matrices in Fig. 4 & 6 show where the misidentification are and that the Random Forest more frequently is far off. In a medical context this raises the question: How long would you be able to wait for a diagnosis to make sure it is as accurate as possible? And how many resources are you able to use to make sure you are not missing a severe disease? A goal for a continuing project would be to experiment with models more that can achieve a high precision/recall quickly. To increase performance, there are also some aspects in the contexts of this medical situation that could be reflected better by the data/model structure.

With more time, I would be interested to look into how to get more out of the data. Currently, the models do not know that the 5 classes are on a severity scale and that identifying a 1 as a 5 is much worse than misidentifying it as a 2. The confusion matrices for the Random Forest in Fig. 4 show some of these ”far off” classifications and implementing a higher penalization for those could boost performance. Additionally, the current one-hot encoding method caused the fitting to fail if testing data had more or less evidence columns due to the expected input shape. I avoided this by doing a train-test split after the one-hot encoding. In a real application it would not be unlikely though that a new symptom could show up and the model would still need to perform. Finding a more flexible model structure or developing a pre-modeling process to deal with these features would be an interesting project to tackle this problem.

References

[1] Zhou, Huan et al. ”Chronic disease diagnosis model based on convolutional neural network and ensemble learning method”. Digital health, 9, 20552076231198643. https://doi.org/10.1177/20552076231198643, 2023.

[2] Tchango, Arsene F. et al. DDXPlus Dataset https://github.com/mila-iqia/ddxplus, 2022.

[3] Tchango, Arsene F. et al. ”DDXPlus: A new dataset for automatic medical diagnosis.” Advances in neural information processing systems 35: 31306-31318, 2022.

[4] Nassiwa, Faith et al. ”A Predictive Diagnosis Assistant in an Electronic Medical Records Platform.” http://dx.doi.org/10.2139/ssrn.4878046, 2024.

[5] Tchango, Arsene F. et al. ”Towards trustworthy automatic diagnosis systems by emulating doctors' reasoning with deep reinforcement learning.” Advances in Neural Information Processing Systems 35: 24502-24515, 2022.

[6] Random Forest. Carmen Canvas: AU24 PHYSICS 5680 - Big Data Analytics. module4_random_forests_only.pdf, 2024.

[7] Yousefi, Masoud et al. ”Random forest classifier for high entropy alloys phase diagnosis”. Afr. Mat. 35, 57 . https://doi.org/10.1007/s13370-024-01198-1, 2024.

[8] Convolutional Neural Networks or CNNs. Carmen Canvas: AU24 PHYSICS 5680 - Big Data Analytics. module6_cnn_overview.pdf, 2024.

[9] Activation Functions in Neural Networks.https://www.geeksforgeeks.org/activation-functions-neural-networks/, 2024.

[10] Baheti, Pragati. ”Activation Functions in Neural Networks [12 Types & Use Cases]”. https://www.v7labs.com/blog/neural-networks-activation-functions, 2021.

[11] Classifying Sequences. Carmen Canvas: AU24 PHYSICS 5680 - Big Data Analytics. module8_sequences_2024.pdf, 2024.