

Information system module for analysis viral infections data based on machine learning

Nickolay Rudnichenko¹, Vladimir Vychuzhanin¹, Tetiana Otradska¹ and Igor Petrov²

¹Odesa Polytechnic National University, 1 Shevchenko Ave., Odesa, 65001, Ukraine

²National University "Odessa Maritime Academy", 8 Didrichson Str., Odesa, 65029, Ukraine

Abstract

The article presents results of the development information system module for analysis viral infections data. The relevance of the problem of automating the process of analyzing large volumes of data based on the use of intelligent technologies and machine learning methods is considered. The structure of the system has been developed and described, the results of design modeling of the key functionality and capabilities of the system based on the use of the UML language are presented, the basic components and technologies for implementing software are described, allowing for modularity and dynamic expandability of the potential for conducting data analysis research. The process of creating, training and testing the created machine learning models is detailed, the results of assessing the significance of the input features of the collected data set on viral diseases and the obtained values of the error matrices are described. The profiling of the operation process of the created models was carried out, the most productive and efficient of them were determined in terms of the consumption of computing resources and overall accuracy, taking into account their generalization ability.

Keywords

data analysis, data visualization, machine learning, viral infections, information systems development

1. Introduction

In the modern world, information technology (IT) allows for the improvement and acceleration of almost any sphere of human activity, automating key aspects and calculations [1].

Systems and software enable us to eliminate complex tasks and go straight to the results, reducing the time needed to make decisions [2]. Automation of data analysis is one of the main aspects of the practical application of modern IT.

Organizations in various fields focus on collecting and accumulating diverse large volumes of data that need to be gathered, analyzed, and stored in a structured and convenient format for further analysis [3].

The use of modern tools and approaches for data analysis makes it possible not only to research and evaluate statistical indicators but also to perform more complex tasks, including


CS&SE@SW 2023: 6th Workshop for Young Scientists in Computer Science & Software Engineering, February 2, 2024, Kryvyi Rih, Ukraine

✉ nickolay.rud@gmail.com (N. Rudnichenko); vint532@gmail.com (V. Vychuzhanin); tv_61@ukr.net (T. Otradska); firmn@gmail.com (I. Petrov)

🆔 0000-0002-7343-8076 (N. Rudnichenko); 0000-0002-6302-1832 (V. Vychuzhanin); 0000-0002-5808-5647 (T. Otradska); 0000-0002-8740-6198 (I. Petrov)



© 2024 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

predicting future values using machine learning (ML). In the context of data analysis, one of the current research directions is the issue of viral diseases [4].

Although COVID-19 has become less critical due to regular vaccinations, infection prevention, and the use of preventive measures such as quarantines, viruses have not disappeared from the planet and continue to mutate [5, 6, 7].

Identifying disease patterns, forecasting, and other analyses can help prevent the further spread of the virus. Automation of symptom detection and human body reactions to viruses is essential, as it enables faster modification of effective vaccines and the introduction of various protective measures into the virus transmission environment [8].

The relevance of this work lies in the search for existing methods of data analysis, for their further application to the data visualization and the identification of various patterns based on obtained graphical representations.

The goal of this work is to investigate the dependencies of various indicators in virology using data analysis models, creating a cross-platform information system module for practical purposes.

Assessing and automating data analysis for various diseases can help identify the source of the virus, its symptoms, and prevent its further spread. Collecting such information is a challenging aspect due to the need for access to medical databases or conducting surveys of infected individuals. This complexity reduces the speed of information updates, and the human factor may compromise data reliability, especially when dealing with a significant number of input features [9].

To solve this task effectively, it is important to use the right methods and algorithms during data collection and preprocessing. Specifically, among contemporary approaches in this direction, dimensionality reduction methods are relevant to facilitate the transformation of data into a form suitable for sequential analysis and interpretation of results [10].

The efficiency of data analysis for such data largely depends on the models used and their hyperparameter values [11]. Some of the less deterministic ML models may perform data analysis quickly but produce inaccurate values and lack sufficient generalization capability [12]. On the other hand, more stochastic models may take longer to perform computational operations, but their precision will be higher as a result [13].

For both cases, setting the right attributes for the models and their relationships with each other is a critical factor. This can reduce the likelihood of full-scale quarantines and expedite the development of vaccine modifications [14].

2. Results

2.1. Project structure

During the system development process, UML modeling language was used. The system is built using a client-server architecture and includes a relational database (DB) for storing data from experiments. The developed use case diagram for the system's operation is shown in figure 1.

Within the scope of this diagram, the user can perform the following actions:

- Authenticate in the system using personal data by verifying requests from the database;

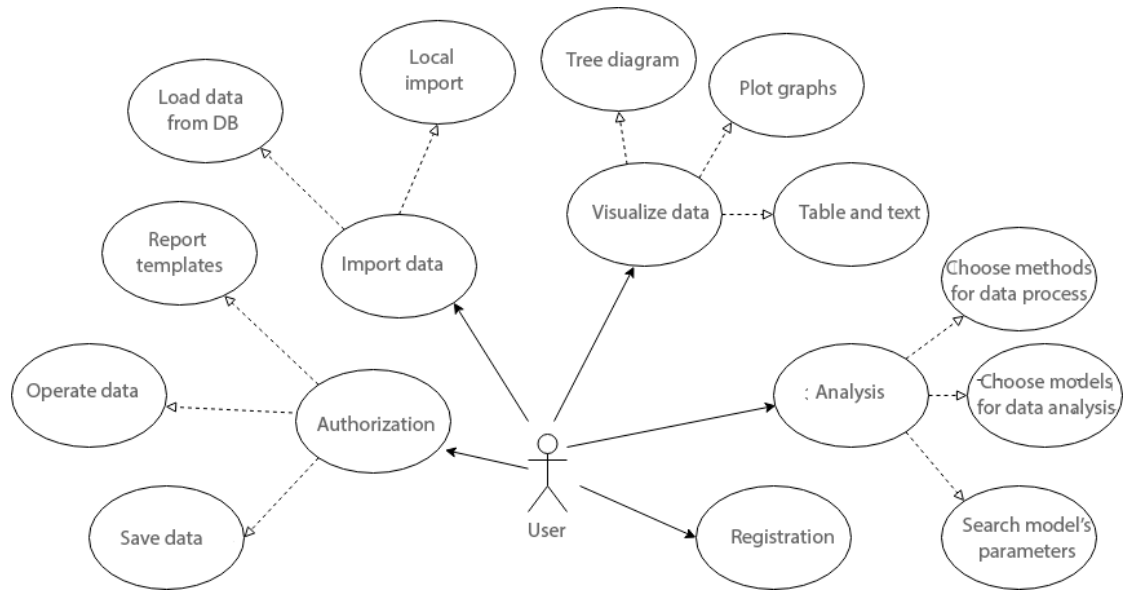


Figure 1: System use-case diagram.

- Import datasets into the system for further analysis;
- Visualize processed data in the form of diagrams, graphs, tables, and textual information;
- Modify arguments and parameters of ML models during their training;
- Register in the system to gain access to data analysis tracking functions.

Let's formalize the main sequence of system operation by applying a sequence diagram for this purpose. For demonstration, we'll take a user who uploads a dataset for conducting exploratory analysis by creating graphical visualizations of static data calculation results (in the form of a small image with statistics and a report with obtained results in a separate file). The user uploads data, selects ML models for analysis, and can set parameter values. The sequence diagram is shown in figure 2.

The user's first action is to access the data analysis model selection form for loading features with parameter editing visualization. After that, they can upload a new dataset in the form of a CSV file or use one of the previously imported data sets serialized to the database. The system user confirms their actions, after which the system dynamically changes the parameter values in the form's interface, depending on which data the user wants to input.

Next, the user fills in all the necessary parameters and data processing algorithms, after which the generated data is sent from the form to the system's handler. Using a REST request, the server-side of the system processes and manages the data to perform further control actions.

In the case of a POST request, the server activates the process of building an ML model based on the specified parameters. The process of using models, in general terms, is as follows:

- Data processing involves receiving input data that will be further processed. Afterward, the system selects software methods for their processing;

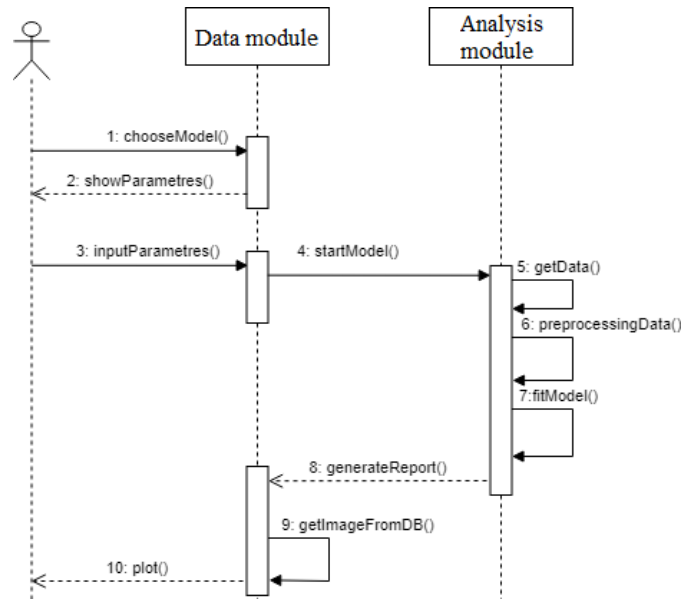


Figure 2: Diagram representing the process of the intelligence analysis of data.

- Creating an instance of an ML model, all imported and processed data are then passed on, and they are processed internally using the fit function;
- Output of a concise report on the evaluation of the performance metrics of the ML model.

After generating the ML model and processing the data, they need to be sent to the system's data visualization module, where, through a GET request, images and other graphical elements specified by the user for the report are displayed. For more detailed planning and formalization of the main relationships between the system components and their structure, it is advisable to construct a diagram of its main components. A generalized component diagram is provided in figure 3.

The main module of the system is flsite. Given that all system modules are developed in Python using the Flask framework, flsite is responsible for the operability and integration of the entire system's functionality. It utilizes the Flask library, allowing the system to be used in the form of a web application, which is convenient due to its compactness and speed.

flsite also contains a set of REST requests that are sent to or received by the system during its operation for rendering authorization pages, user profiles, and model management.

The user profile page contains information about the results of analytical experiments performed by users and datasets uploaded by them. This is necessary for repeating or correcting data analysis.

On the authorization page, the user needs to enter login and password data, as well as a personal token generated by the system after the registration process through a separate modular window. Additionally, for introductory actions in working with the system, the user can choose a limited version in which they don't need to enter authentication data, but they will have restrictions in terms of uploading their data; in other words, they will be able to work

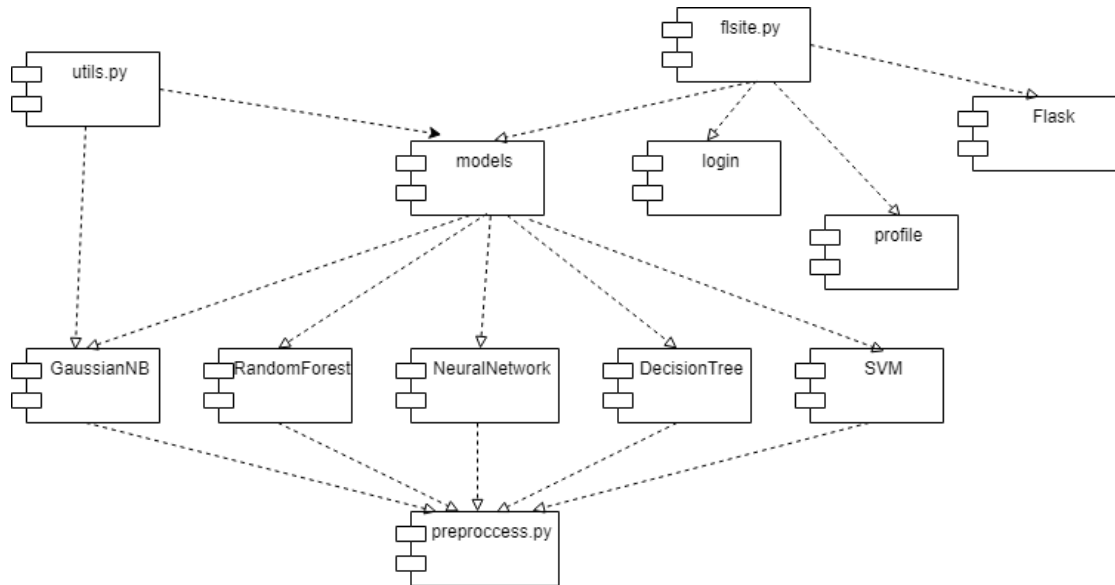


Figure 3: The diagram of the system components in a general form.

with only preinstalled test datasets.

The scikit-learn library is used for the programmatic implementation of ML models due to its convenience, good documentation, and integration with the Python language. The advantage of using this dependency is its support for a significant set of objects for conducting ML model research, performing statistical modeling, including classification, regression, clustering, and dimensionality reduction, through a sequential Python interface. Additionally, libraries such as Pandas, NumPy, SciPy, and Matplotlib were applied for data processing and visualization.

It's worth noting the preprocess.py class, responsible for obtaining input model values and their transformations. This is implemented by integrating functionality from the imbalanced-learn library, which is based on scikit-learn and provides a range of objects for simplified and fast classification work in cases of class imbalance detection.

The utils.py class is responsible for retrieving data from the database, writing them to a file, generating graphical visualizations for reports, transforming data structures and data into matrices of various dimensions, and a variety of other utility functions.

We use SQLite to create the database, which comes bundled with Python3. Its convenience is due to its implementation of support for autonomous and transactional relational mechanisms. The relationship between the database and the system is depicted in figure 4.

First, a database with a set of tables is created separately (without launching the web server). Based on this, request handlers access the construction of database tables, reading and writing information to them. Afterward, the process of creating a general function for establishing a connection to the database and an auxiliary function that will initiate the construction process of a relational database model with the required tables is carried out. At this stage, the open_resource programmatic method is introduced, which opens the 'sq_db.sql' file for reading, located in the project's system working directory. Then, for the open database, the

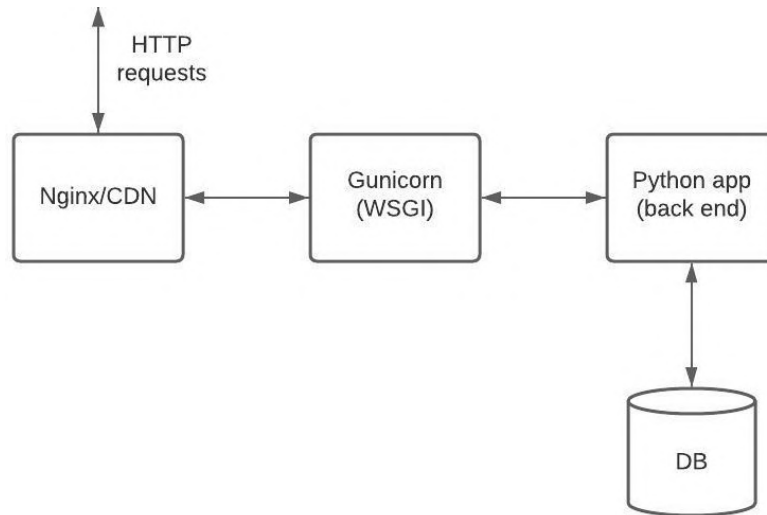


Figure 4: Connection between the system and the database.

script contained in the 'sq_db.sql' file is executed. Finally, the commit method is called to apply the changes to the current database, and the close method closes the established connection.

2.2. ML models implementation

As part of using the system's data analysis module, the following 5 ML models have been programmatically implemented: Gaussian Naive Bayes (GNB), Decision Tree, Random Forest, SVM, and Neural Network (NN) based on a perceptron model. In the system's interface, these models can be assigned various hyperparameter values, allowing clients to input data manually from the web page through corresponding fields. The steps of implementing the data analysis process are as follows:

- Importing input dependencies and libraries.
- Fetching input data into the working environment by downloading them via links.
- Splitting data into training and testing subsets. To perform this step, a cross-validation splitter has been defined, which divides the data into training/test sets according to a specified scheme. Each sample can be assigned to no more than one fold of the test set, as indicated by the user using the test_fold parameter.
- Creating ML model objects.
- Tuning hyperparameter values of ML models using the GridSearchCV method for optimal selection. This is essentially an object implementing a systematic approach to hyperparameter optimization, which involves defining a grid of possible values for each hyperparameter and then exhaustively searching this grid to find the best combination. This approach is called grid search because it involves creating a grid of hyperparameter values and evaluating the algorithm's performance for each combination of values in the grid. It's worth noting that there's no way to know the best hyperparameter values in advance, so ideally, you should try all possible values to find the optimal ones.

- Doing this manually can be time-consuming and resource-intensive, so we use Grid-SearchCV for automating hyperparameter tuning. The following models have similar code but different parameters within.
- Training models and assessing their accuracy metrics based on the use of methods such as `classification_report`, taking into account particularities including permutations (`permutation_importance`). Permutation importance is defined as the reduction in model score when the values of a single feature are randomly shuffled. This procedure breaks the relationship between the feature and the target variable, so a drop in the model's score indicates how much the model depends on that feature. One advantage of this method is that it's model-agnostic and can be calculated many times with different feature shuffles. At this stage, we obtain a finished model for further interaction.

2.3. Data analysis

In order to conduct research, a dataset was created based on information from the European Centre for Disease Prevention and Control regarding the most common COVID-19 symptoms in 2020. The dataset contains information about patients who underwent COVID-19 testing, with a total of 5434 rows and 211 columns. Each row corresponds to a patient's symptoms, where each column represents a symptom. To properly preprocess the data, a dimensionality reduction procedure was performed using PCA. Some features, such as 'had contact with someone with the virus' or 'recently traveled', were either removed or combined into more meaningful ones. This allows us to focus on significant symptoms and identify which symptoms indicate early stages of COVID-19.

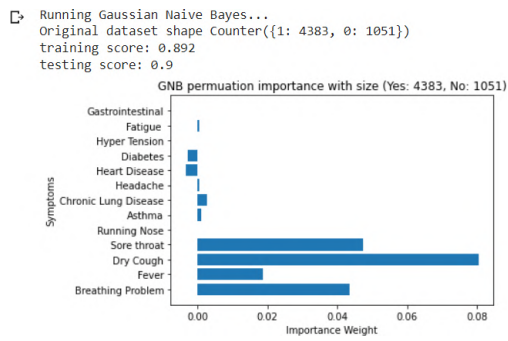
Out of 5434 patients, 81% later tested positive for the virus. To address the dataset's imbalance, the `imblearn` library was used, which helps minimize the dataset's class imbalance. We chose the resampling method, which aggregates repetitions and reduces data imbalance. The dataset's target variable is expressed in a binary form, with two values: "yes" and "no". Essentially, this allows us to frame the task as binary classification. The data was split into training, validation, and testing sets, with 60%, 20%, and 20% of the data in each part, respectively. Below is an example of the data processing results and model creation with the described datasets.

The results of building the Gaussian Naive Bayes model are shown in figure 5.

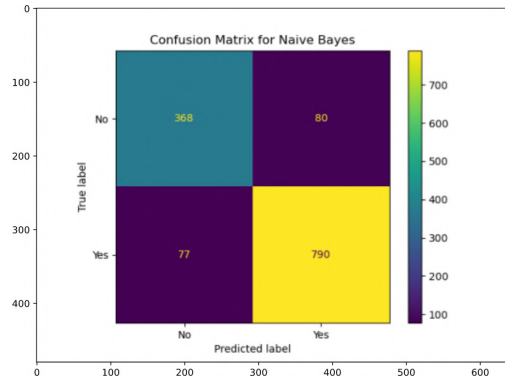
As we can see, the top three symptoms for the Naive Bayes model are Dry cough, Sore Throat, and Breathing Problems. Fever was close to the third position, but it's worth noting that its weight is significantly higher than other symptoms. As observed from the matrix, there are 77 False Negatives (FN) and 80 False Positives (FP). Considering that GNB is a basic model, its accuracy is quite high. It's evident that we aim to have a stronger model, and we hope to achieve this with better models ahead. In terms of metrics, the F1 score for negative cases is 0.824, while the F1 score for positive cases is 0.910.

Let's examine the results of the Decision Tree model (figure 6).

As we can see, the top three symptoms for the Decision Tree model are Breathing Problems, Sore Throat, and Dry Cough. Once again, Fever was close to the third position, but, unlike GNB, other symptoms have more weight. In this case, we used the output of the `feature_importances` from the Decision Tree. There is an interesting shift in the importance of Breathing Problems and Sore Throat. After analyzing the data, we observe an increase in the importance of Sore

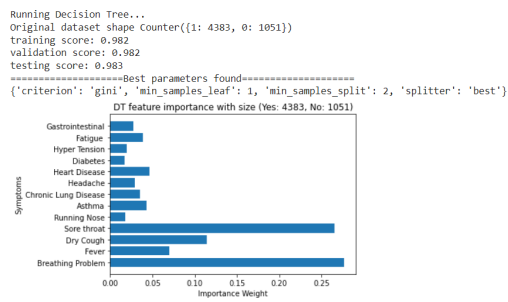


(a)

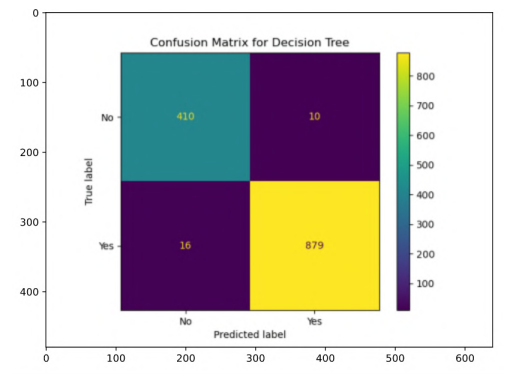


(b)

Figure 5: Outputs of GNB model (a) and the corresponding confusion matrix (b).



(a)



(b)

Figure 6: Outputs of DT model (a) and the corresponding confusion matrix (b).

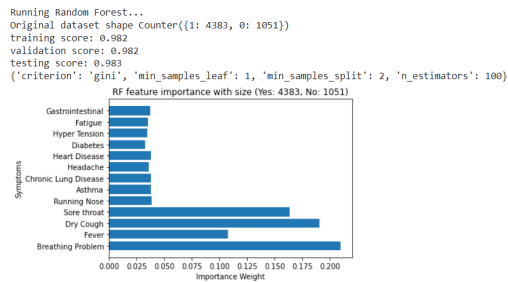
Throat, while Breathing Problems decrease. This suggests that Sore Throat is much more important than Breathing Problems. As we can see, the Decision Tree works much better than GNB. Previously, there were 77 FN and 80 FP, and now there are 16 FN and 10 FP. In terms of metrics, the F1 score for negative cases is 0.969, and the F1 score for positive cases is 0.985.

The most important features for the Decision Tree are Breathing Problems, Sore Throat, and Dry Cough. Currently, the symptoms are consistent with each other in every model we have built. Next, we will conduct research and analyze the Random Forest models (figure 7).

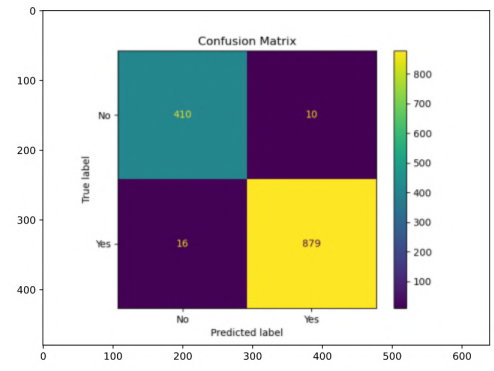
As we can see, the top three symptoms for the scaled Random Forest model are Dry Cough, Breathing Problems, and Sore Throat.

Overall, the Random Forest model produces results similar to the Decision Tree. Previously, there were 16 FN and 10 FP, and we still have 16 FN and 10 FP. In terms of metrics, the F1 score for negative cases is 0.968, while the F1 score for positive cases is 0.985.

The most important features for the Random Forest are Dry Cough, Breathing Problems, and Sore Throat. Currently, the symptoms are consistent in every built and analyzed model.



(a)



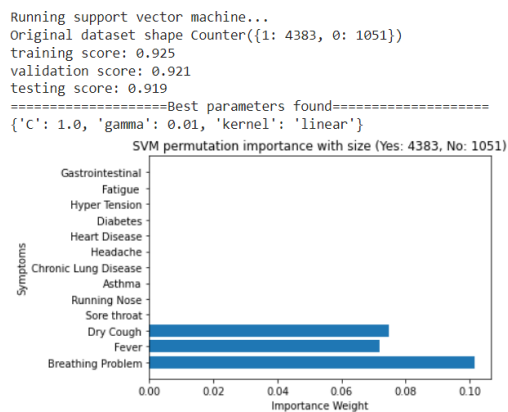
(b)

Figure 7: Outputs of the Random Forest model (a) and the corresponding confusion matrix (b).

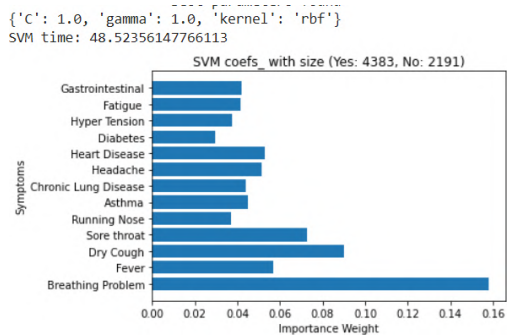
Next, let's analyze the performance of the Support Vector Machine (SVM) model. The demonstration of SVM is shown in figure 8. In our case, SVM outputs a hyperplane that separates both classes. The weight coefficients, represented by coef, form a hyperplane orthogonal to the original division boundary. If the hyperplane finds a feature useful for separating the data, the plane will be orthogonal to that axis. Therefore, the coefficient shows how important it was in dividing the two datasets.

We will move on to using more complex kernels (rbf, poly) and use permutation importance to compute feature importance (since coef_ is valid only for linear kernels). As we can see, the top features are Breathing Problems, Dry Cough, and Sore Throat.

In our case, using the RBF kernel works faster and provides a more balanced analysis of feature importance.



(a)



(b)

Figure 8: Outputs of the SVM model using linear kernel (a), and using RBF kernel (b).

Let's conduct an investigation of the created neural network model (figure 9). For the neural

network, we compare functions using permutation importance.

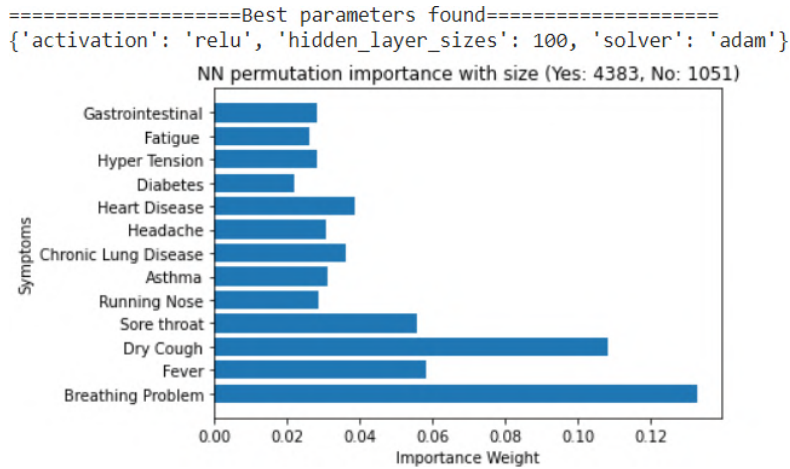


Figure 9: Outputs of the Neural Network.

From the parameter grid, we used a validation set to find the best combination. For the activation function, we used ReLU, the number of hidden layers is 80, and we utilized the Adam optimizer for better performance.

We wanted to understand whether our network prefers having one larger bank of hidden layers or several smaller banks of hidden layers. The artificial neural network model achieved the most significant results with only 6 cases of TN errors. The most important features are “breathing problems”, “dry cough”, and “sore throat”, depending on the dataset used.

Based on the results obtained, let’s analyze the performance speed of all the ML models discussed above. The speed of the two neural networks (ReLU and Adam) is presented through profiling in figure 10. In this case, we can see that GNB and DT have the minimum operating speed. RF, due to its complexity, operates more slowly. SVM and NN1 work significantly faster, even with input value analysis.

Therefore, the developed module is sufficiently functional and allows for the exploration of datasets to uncover hidden patterns within the data.

3. Summary

As a result of this research, a software module for the system has been developed, capable of performing data preprocessing and analysis steps related to human virus-related illnesses using machine learning methods. The following evaluation criteria for the models were determined: speed, accuracy, predictions, best hyperparameters for the models, error matrix assessments.

The research findings have established that the most effective models are artificial neural networks, but decision trees also showed one of the best results, considering that this model is based on a basic algorithm. The study also revealed the varying speeds of different models, with a significant workload dependence on the chosen hyperparameter values. Additionally, data inaccuracies can complicate the process of determining the best model.

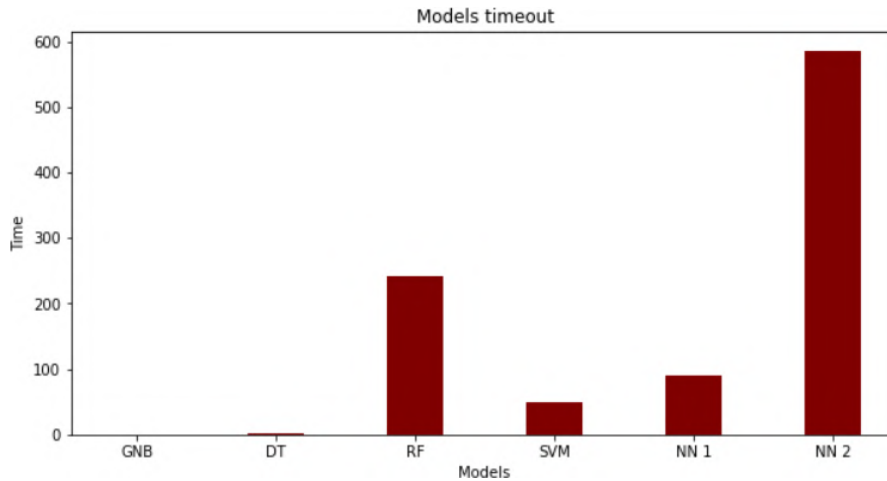


Figure 10: Comparison of the time performance (cost) of the created models.

Therefore, it can be concluded that the best model for working with the data collected during the development of the system is the neural network. However, its speed is significantly lower than that of the decision tree model. Therefore, future research in this area could involve exploring more efficient models and forming a more extensive dataset for a more balanced analysis.

References

- [1] C. Peng, Digital Inclusive Finance Data Mining and Model-Driven Analysis of the Impact of Urban-Rural Income Gap, *Wireless Communications and Mobile Computing* 7 (2022) 1–8. doi:10.1155/2022/5820145.
- [2] N. Rudnichenko, V. Vychuzhanin, I. Petrov, D. Shibaev, Decision Support System for the Machine Learning Methods Selection in Big Data Mining, *CEUR Workshop Proceedings* 2608 (2020) 872–885. URL: <https://ceur-ws.org/Vol-2608/paper65.pdf>.
- [3] I. M. Shpinareva, A. A. Yakushina, L. A. Voloshchuk, N. D. Rudnichenko, Detection and classification of network attacks using the deep neural network cascade, *Herald of Advanced Information Technology* 4 (2021) 244–254. doi:10.15276/hait.03.2021.4.
- [4] J. Andry, H. Tannady, G. Rembulan, D. Dinata, Analysis of the Omicron virus cases using data mining methods in rapid miner applications, *Microbes and Infectious Diseases* 4 (2023). doi:10.21608/mid.2023.194619.1469.
- [5] H. Wijaya, S. Syaifudin, T. Siswanto, Visualization of corona virus disease 2019 deoxyribonucleic acid data analysis, *AIP Conference Proceedings* 2659 (2022) 090005. doi:10.1063/5.0118893.
- [6] A. D. Kubegenova, E. S. Kubegenov, Z. M. Gumarova, G. A. Kamalova, G. M. Zhazykbaeva, Using Data Mining Technology in Monitoring and Modeling the Epidemiological Situation of the Human Immunodeficiency Virus in Kazakhstan, in: A. Gibadullin (Ed.), *Information*

- Technologies and Intelligent Decision Making Systems, Springer Nature Switzerland, Cham, 2022, pp. 57–65. doi:10.1007/978-3-031-21340-3_6.
- [7] P. Srinivas, D. Bhattacharyya, D. MidhunChakkaravarthy, Prediction of Swine Flu (H1N1) Virus Using Data Mining and Convolutional Neural Network Techniques, in: A. Kumar, G. Ghinea, S. Merugu, T. Hashimoto (Eds.), Proceedings of the International Conference on Cognitive and Intelligent Computing, Springer Nature Singapore, Singapore, 2023, pp. 557–573. doi:10.1007/978-981-19-2358-6_51.
- [8] H. Gunawan, V. Purwayoga, Data mining menggunakan algoritma k-means clustering untuk mengetahui potensi penyebaran virus corona di kota cirebon, Jurnal Sisfokom (Sistem Informasi dan Komputer) 11 (2022) 1–8. doi:10.32736/sisfokom.v11i1.1316.
- [9] V. Dev, P. Singh, Effect of Corona Virus on Multi-Disease Patients using Association Rule Mining, 2023. URL: <https://www.researchgate.net/publication/372312135>.
- [10] E. Luna-Ramírez, J. Soria-Cruz, I. Castillo-Zúñiga, J. I. López-Veyna, COVID-19 Social Lethality Characterization in some Regions of Mexico through the Pandemic Years Using Data Mining, in: Y. P. Rybarczyk (Ed.), Research Advances in Data Mining Techniques and Applications, IntechOpen, Rijeka, 2023. doi:10.5772/intechopen.113261.
- [11] A. Y. O. Allmuttar, S. K. D. Alkhafaji, Using data mining techniques deep analysis and theoretical investigation of COVID-19 pandemic, Measurement: Sensors 27 (2023) 100747. doi:10.1016/j.measen.2023.100747.
- [12] M. Pandey, Machine Learning, International Journal for Research in Applied Science and Engineering Technology 11 (2023) 864–869. doi:10.22214/ijraset.2023.55224.
- [13] D. P. F. Möller, Machine Learning and Deep Learning, in: Guide to Cybersecurity in Digital Transformation: Trends, Methods, Technologies, Applications and Best Practices, Springer Nature Switzerland, Cham, 2023, pp. 347–384. doi:10.1007/978-3-031-26845-8_8.
- [14] R. Thakur, P. Panse, P. Bhanarkar, Machine Learning and Deep Learning Techniques, in: U. N. Dulhare, E. H. Houssein (Eds.), Machine Learning and Metaheuristics: Methods and Analysis, Springer Nature Singapore, Singapore, 2023, pp. 235–253. doi:10.1007/978-981-99-6645-5_11.