Applied Machine Learning Project Report: Diseases Prediction Dataset

Tom Bourjala, Wiktoria Ciasnocha, Patricia List, Adéla Ondrouchová

Abstract

The rapid advancement of artificial intelligence and machine learning has created new possibilities in various fields, including healthcare. Misdiagnosis errors are common and can lead to mistreatment, delayed care, or even death. Research in this field may provide reliable tools to reduce the chance of misdiagnosis.

This study investigates the efficiency of diverse machine learning algorithms for predicting diseases based on symptoms, aiming to facilitate timely diagnosis and treatment. The performance of Logistic regression, K-Nearest Neighbors, Decision tree, Random forest and Neural network algorithms is analyzed using on a dataset containing 4921 patients' records. Further, the study explores the use of probabilities with Neural network and Random forest algorithms to assess the confidence of the model in its predictions and to offer alternative diagnoses.

The dataset, procured from Pranay Patil and Pratik Rathod, includes a list of 2 to 17 symptoms from 133 distinct symptoms, associated with one of 42 diseases. Data preprocessing methods, such as one-hot encoding and removal of duplicate data, are employed to optimize the dataset for machine learning.

The study concludes by comparing the performance of the various algorithms, identifying the most effective approach and techniques for predicting diseases based on symptoms and discussing the potential of probability-based predictions in enhancing diagnostic accuracy.

Email addresses: `tobou23@student.sdu.dk` (Tom Bourjala), `wicia23@student.sdu.dk` (Wiktoria Ciasnocha), `palis23@student.sdu.dk` (Patricia List), `adond22@student.sdu.dk` (Adéla Ondrouchová)

Contents

1. Introduction

According to a report published by the US Department of Health and Human Services' Agency, an estimated 7.4 million misdiagnosis errors occur annually in the United States alone [1]. These errors can lead to incorrect treatment, deterioration of patient's health, and increased healthcare costs. Therefore, there is a need to develop accurate disease prediction models that can assist healthcare professionals in making decisions and reducing the occurrence of misdiagnosis.

Although machine learning models do not match human intelligence, they can be extremely useful tools to support human work. Predicting diseases based on symptoms can help healthcare professionals and researchers identify potential diseases and provide better treatment options. With the increasing availability of data and the development of machine learning algorithms, disease prediction based on symptoms will become more accurate and efficient.

The aim of this project is to create and assess a model that assists in predicting and diagnosing diseases. This research paper investigates the efficacy of different machine learning algorithms in predicting disease symptoms. The algorithms' performance is analyzed and compared on a relevant dataset.

The paper includes a section on related work, as well as descriptions of the data, experiments, and results. The experiments section covers pre-processing techniques, while the processing section discusses several machine learning algorithms, such as **Logistic regression**, **KNN**, **Decision tree**, **Random forest** and **Neural network**.

Finally, the performance of different algorithms is compared, and the most effective method for predicting disease symptoms is identified.

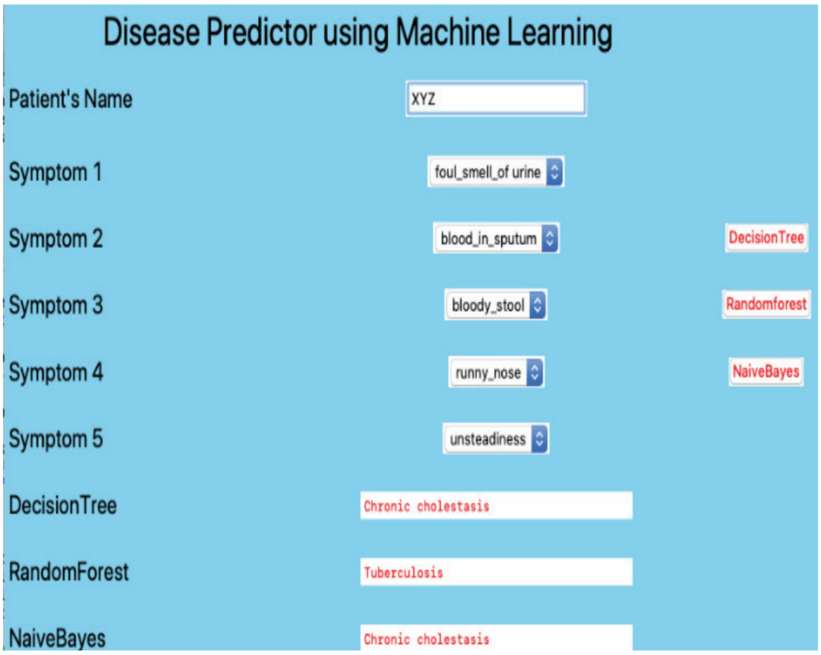
2. Related Work

Multiple studies and methodologies, dealing with the problem of disease prediction using machine learning techniques, have been already proposed. This section will review three related papers based on the used dataset and their approaches to predicting diseases from symptoms.

The study by P. Hema et al. [2] proposes a model for disease prediction using decision trees, Naive Bayes, and Random forest classifiers. Information from 4921 patients and 41 disease were selected. The model takes various symptoms as input, and the final result is shown to users via a graphical user interface. The final disease prediction was made by taking into account the mode of the outputs from all the three classifiers. The model shows an accuracy of 93 %, which is much higher compares to existing models. In the future P. Hema et al. want to test the system on a wider dataset.

K. S. Kumar et al. [3] also developed a machine learning system within their study that predicts diseases based on entered symptoms. For their predictive model they chose the Random forest algorithm. To display the findings they used a graphical user interface (GUI). Their model offers additionally a convenient alternative to visiting a doctor and contributes to the initiative of treating diseases at an earlier stage. The system uses the same dataset with 132 symptoms and 42 diseases. Although the system achieves an accuracy of 95 %, the computation time is long. To improve accuracy and reduce computation time, the authors propose a pipeline model of three algorithms, and suggests improvements to the user interface, including providing basic health information and guidelines.

Disease prediction model using machine learning algorithms was also proposed by Sneha Grampurohit and Chetan Sagarnal [4]. To predict the disease based on the symptoms entered by the user they used machine learning algorithms such as Decision tree, Random forest, and Naïve Bayes classifiers. Finally they compared the results of all used algorithms (see Figure (1)).



Disease Predictor using Machine Learning

Patient's Name: XYZ

Symptom 1: foul_smell_of urine

Symptom 2: blood_in_sputum

Symptom 3: bloody_stool

Symptom 4: runny_nose

Symptom 5: unsteadiness

DecisionTree: Chronic cholestasis

RandomForest: Tuberculosis

NaiveBayes: Chronic cholestasis

Figure 1: Resulting disease prediction GUI. After the patient enters the symptoms, the GUI displays the prediction results for the selected algorithms

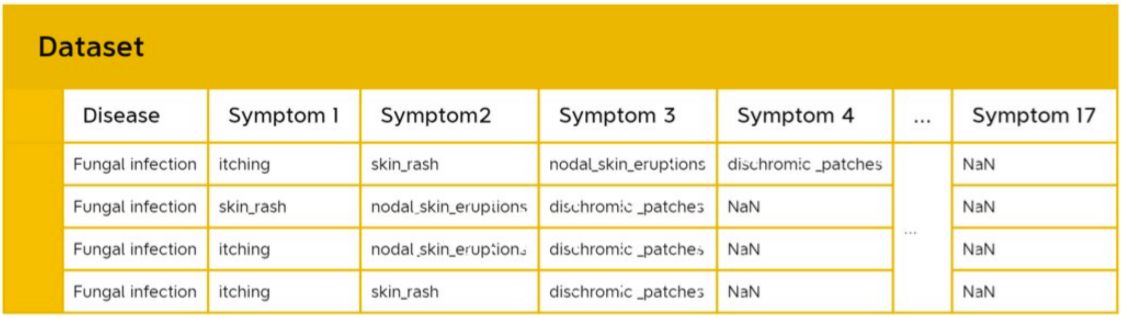
To avoid the problem of overfitting they eliminated independent variables that had little or no impact on the target variable. In this way the size of the database was reduced from 132 to 95 symptoms. All three algorithms performed well, with Naïve Bayes performing slightly better than the other two. The final predictive model reached the accuracy of up to 95 %.

3. Description of the Data

This paper is based on the **Diseases symptom prediction** dataset [5]. The data was collected from various web sources and processed for unification by a team of researchers.

The dataset contains information on 4921 patients, which had 133 different symptoms linked to 41 diseases. All the diseases and symptoms are listed in the Appendix. Each entry in the dataset associates a **disease (outcome)** with a list of **symptoms (features)**. The dataset is formed of categorical features where each disease has anywhere between 1 to 17 symptoms.

In Figure (2) it is shown that some diseases have more symptoms than others, which leads to a lot of NaN fields.

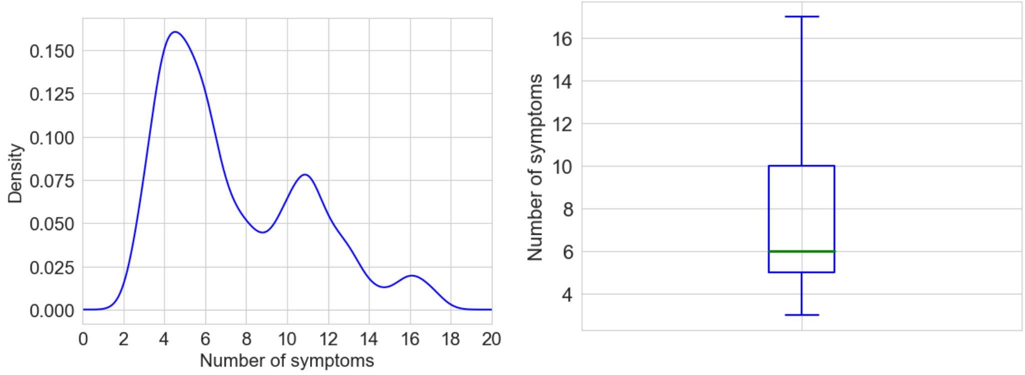


Dataset

Disease	Symptom 1	Symptom2	Symptom 3	Symptom 4	...	Symptom 17
Fungal infection	itching	skin_rash	nodal_skin_eruptions	dischromic _patches	...	NaN
Fungal infection	skin_rash	nodal_skin_eruptions	dischromic _patches	NaN		NaN
Fungal infection	itching	nodal_skin_eruptions	dischromic _patches	NaN		NaN
Fungal infection	itching	skin_rash	dischromic _patches	NaN		NaN

Figure 2: Short overview of the Dataset

The density plot in Figure (3a) shows that the number of symptoms per disease ranges between 2 and 17. The plot also reveals a peak between 4 and 6 symptoms, indicating that most patients had this amount of symptoms. The same results are shown in the box plot in Figure (3b), where the median is shown to be at 6.



(a) Density of symptoms per entry

(b) Box plot of symptoms per entry

Figure 3: Dataset and Sampling

4. Experiments

Due to the nature of the dataset being related to health, accuracy is a particularly important feature of the proposed model. Therefore, five different algorithms were tested to find the one that produces the best results in disease forecasting. Figure (4) displays the pipeline of the proposed algorithm.

The data were **pre-processed** and then split into **training and testing datasets**. The `CustomGridSearch` function was implemented, which required another split of the testing data into **test** and **GridSearchTest** sets for hyperparameter validation. The training dataset was used to train the various models considered for this pipeline. These trained models were then evaluated using the testing dataset.

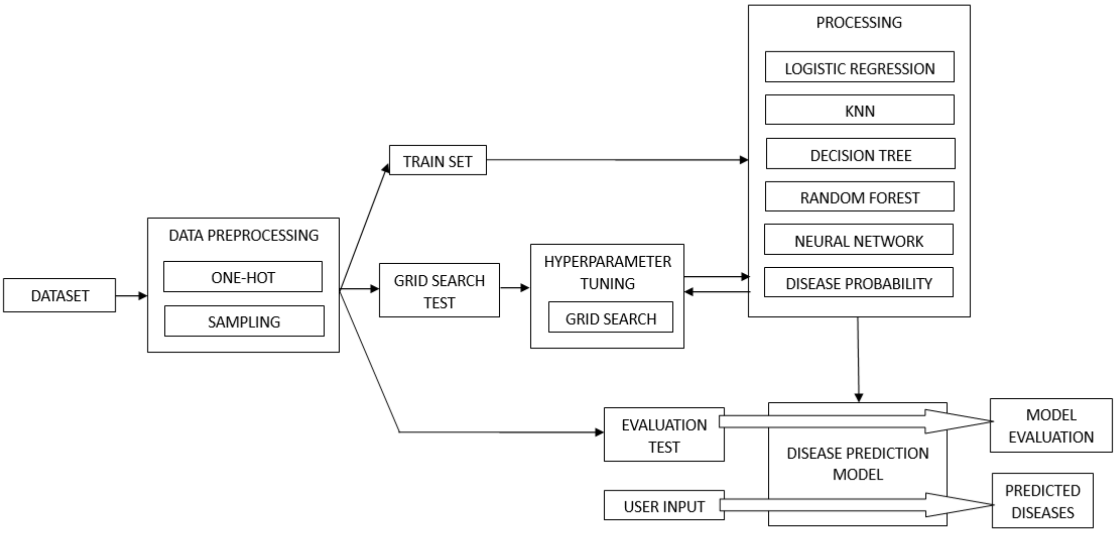


Figure 4: Project pipeline

4.1. Pre-processing

Data pre-processing is essential in preparing data for machine learning models. In this case it involved one-hot encoding of diseases and symptoms, addressing the issue of duplicate data and lastly, splitting into train and test sets.

4.1.1. One-hot encoding

In order to use categorical data for prediction using machine learning, one-hot encoding for both diseases and symptoms was needed. This technique allows replacing categorical variables with numerical features. For each possible value new column is created. Based on whether the feature is present, a binary value of 1 or 0 is assigned to each row. [6]

First, **all symptoms** for each entry were combined into a list. This created a data frame with only two columns, `Diseases` for the disease and `Symptoms` for list of all symptoms the patient had. This modification was necessary, because in the original data frame one symptom could be in different columns for different entries, which would later result in multiple columns for the same symptom.

Next, the `Symptoms` column was split into separate Pandas Series and stacked as multiple rows per entry, where each row was one single symptom. This created a data frame with **multi-level indexing**, where all of the symptoms are in one column.

One-hot encoding is applied on the new data frame using `get_dummies` function from the Pandas library. This function automatically transforms all categorical columns.

Final step is to merge all the symptoms from one entry back together using the `group_by` and `sum` methods. The grouping is performed by first level of the multi-level indexing (meaning the original row, which represents an entry) and summing the values in one-hot encoded columns. The resulting value represents if a symptom occur in an entry (i.e. 1 or 0).

Since the **diseases** are already in one column named `Disease`, there is no need for further altering. One-hot encoding is again performed using `get_dummies` function from Pandas library.

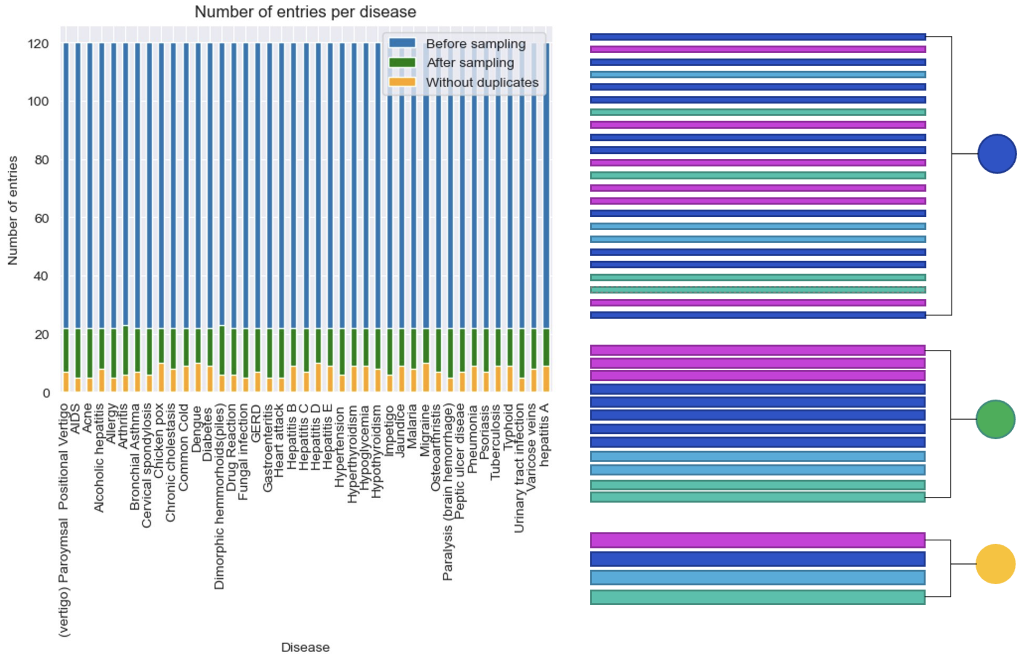
The last step is to combine the one-hot encoded symptoms with the diseases. The result is the final data frame with rows representing the entries and the columns representing the different diseases and symptoms. This data frame can now be used for training models to predict diseases based on their associated symptoms.

4.1.2. Duplicate data

This database contains a lot of duplicate data, with around 90 % of entries for each disease being identical. This leads to an **overfitting** problem. In the initial testing, prediction accuracy was almost always 100 % due to the training and test datasets being nearly identical after random splitting. Another issue is the size of the dataset, as having a lot of data can slow down the training process.

Although deleting all duplicates was considered, it would result in a loss of information about the distribution. This is because some duplicates are more common than others.

To address this issue, a solution was implemented which involves dividing the data into subsets of identical entries (Figure (5b)). By dividing the number of entries in each subset by the same number (number 5 was chosen in this case), the same distribution of duplicates is maintained while reducing the overall dataset volume (see in Figure (5a)).



(a) Dataset comparison before and after sampling

(b) Sampling

Figure 5: Dataset and Sampling

4.1.3. Split train and test

The data was split into a training set and two testing sets using a custom train-test split function. This function takes the subsets mentioned in the previous section and divides them into two equal groups. Then, it splits the second set into two parts. The first, larger group is used for training, one part of the second group is used for grid search validation and the last one is used for the final model evaluation. This splitting ensures that the sets are sufficiently dissimilar and do not contain common entries, which cannot be achieved using the `train_test_split` function from sklearn.

4.2. Processing

Various machine learning models were employed in this project. In addition to typical classification algorithms such as Logistic regression, KNN, Decision tree, and Random forest, more advanced techniques were used, such as the Neural network algorithm. Ultimately, a probability table for diseases was created using some of the trained models.

In some cases a hyperparameter optimisation was needed. Because of the dataset used, it was not possible to use `GridSearchCV` function from **sklearn**, since this function uses a **k-fold Cross-Validation**. This random split gives similar train-test data and therefore 100 % accuracy for almost every value of the hyperparameters. For this reason a `CustumGridSearch` function was implemented. This grid search function takes two datasets, training and testing, as an input. One part of the testing dataset (as mentioned in 4.1.3) is used for grid search validation. Doing grid search this way eliminates the similarities in hyperparameters validation while it does not contaminate the model evaluation.

4.2.1. Logistic regression

The first implemented predictive model was `LogisticRegression`. It is one of the most commonly used linear classification algorithms. Because the dataset used in this project is multiclass, the type of Logistic regression used for prediction is `multinomial`.

Multinomial Logistic regression examines the influence of an independent variable on a multinomial dependent variable [6]. It extends the Logistic regression algorithm to solve multiclass possible outcome problems. In contrast to the binary Logistic regression, there are more than two response categories. With multinomial variables, more than one comparison can be made. Which response categories are compared depends on how the analysis is specified.

The Logistic regression model implemented in this project predicts the probability of each possible class. Once the model is trained, predictions are made on the test data. The accuracy of the model is then evaluated by comparing the predicted labels with the true labels using the `accuracy_score()` function. The final accuracy score of this model on testing data is 100 %.

4.2.2. KNN

Another prediction model used in this project is the `KNeighborsClassifier`. This supervised learning algorithm classifies every data point by finding its **k** nearest neighbors in the training set and assigning it the most frequent class among those neighbors.

In order to find the optimal value of the hyperparameter **k** (= **n_neighbors**) and maximise the models accuracy the `CustomGridSearch` function was used. The accuracy was plotted over the number of neighbours hyperparameter in Figure (6).

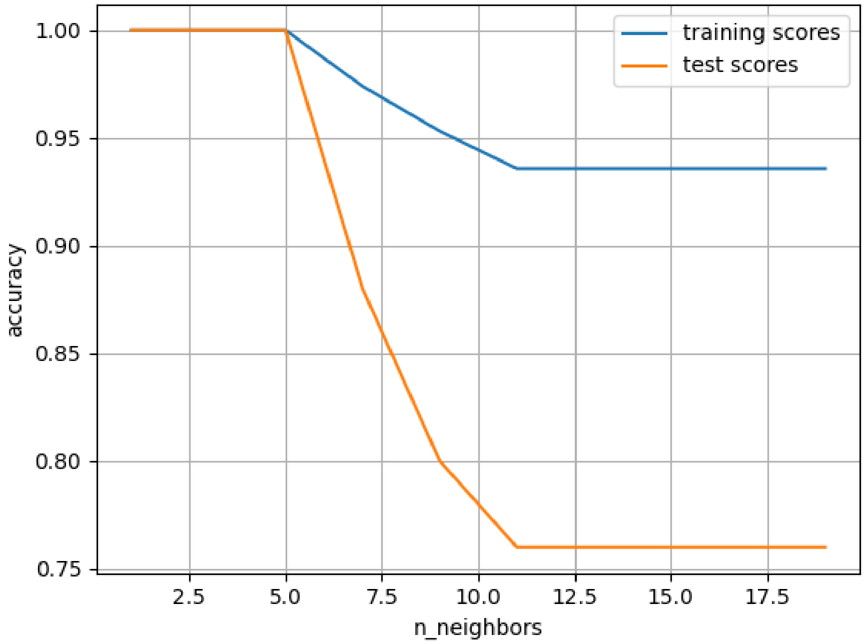


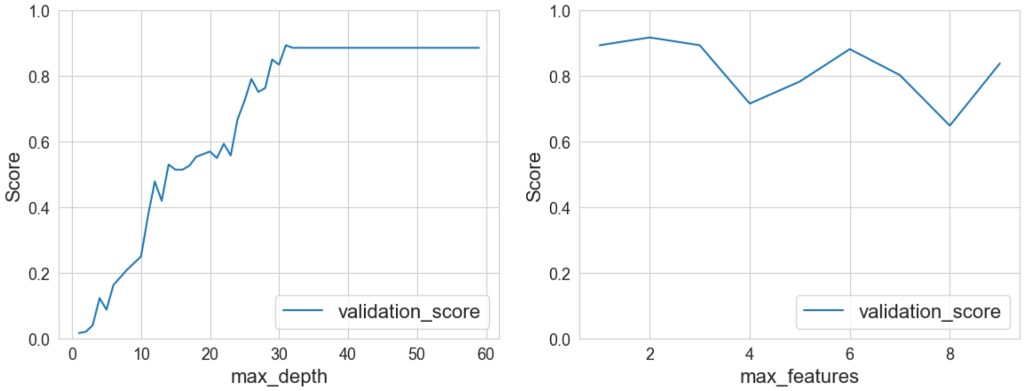
Figure 6: Training and test score based on number of neighbors for KNN

The plot shows that the training and the test score generally decrease as the number of neighbors increases. Initially, both functions are constant and the accuracy is 100 %. Both lines start decreasing as soon as the **n_neighbors** exceed value of 5. This suggests that the model becomes less flexible and shows signs of **underfitting** when using more than 5 neighbours.

Based on the highest mean score of the evaluation, the best number of neighbors is found to be 1. Then the model was trained using this parameter. Finally, the `KNeighborsClassifier` was used to predict the class labels for the test set and the accuracy of the model was calculated. The final accuracy of this model is 100 %.

4.2.3. Decision tree

The last simple model for class prediction is the Decision tree. At first, the parameters were set to `max_features=1` which determines the number of features considered at each split and `max_depth=none` which means there is no limit on the number of nodes, this model gives an accuracy of 69 %. Those two hyperparameters were then optimised using the custom grid search function implemented previously. First, the best value for `max_depth` was found and then another grid search was performed with this optimised value to find the best `max_features`. The values for the hyperparameters of the Decision tree have been plotted over the accuracy. The results are shown in Figure (7a) and Figure (7b). The optimal values of `max_features` and `max_depth` are those that correspond to the highest score.



(a) Scores for **max_depth** grid search

(b) Scores for **max_feature** grid search

Figure 7: Dataset and Sampling

Hyperparameters found by the grid search were `max_depth=31` and `max_features=2`. This final model gives the accuracy of 76 %, so the optimisation led to 7 % increase in accuracy score.

4.2.4. Random forest

Random forest is a learning method which is used for classification and regression. It is a model which is based on a Decision tree. The Random forest function creates multiple decision trees and combines their results to make more accurate predictions. The way to implement this model to predict a disease outcome for the test data is the using the `RandomForestClassifier`. The `classification_report()` function used in the code summarises the precision, recall, f1-score and support for each class. The precision is 100 % for all classes.

4.2.5. Neural network

Neural network is a machine learning algorithm which is inspired by the way how human neurons work. They are used to solve complex, non-linear, problems or work with high volume datasets. The network is constructed from input layer, output layer and one or more **hidden layers**. These layers are made of nodes called **artificial neurons** or just **neurons** for short. Architecture of the whole network can be seen in Figure (8).

The connection between every neuron has a specific **weight**, which is used to multiply the value associated with this specific connection. Each neuron then sums the input values together with a **bias** and applies an **activation function** to the computed value, which adds non-linearity. During the training process, the values for weights and for biases are adjusted using optimisation algorithms such as **gradient descent** or **Adam (Adaptive Moment Estimation)**.

To prevent overfitting during training, a **drop out rate** can be introduced to the model. During each iteration of the training process, there is a chance for each neuron and its connections to be temporarily removed. This encourages the network to learn more generalizable features and not rely too heavily on a specific set of neurons.

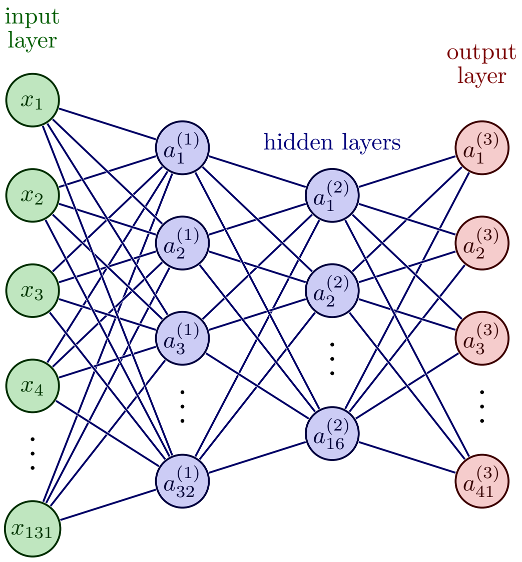


Figure 8: Neural network

Neural network have recently gained success in various fields, including medical diagnosis [7]. In this project, a Neural network classifier was used for disease prediction. The model architecture in this case is composed of 2 hidden layers (Figure (8)) and uses a drop out rate due to overfitting problems. Activation function used in both layers is **ReLU**. The optimisation function for training is **Adam**, because it is suitable for use in data sets with high number of parameters. The values for numbers of neurons in both hidden layers and the ideal drop out rate were found using `CustumGridSearch`. The results are visualised as heat-maps in Figure (9). Model with the best score uses **32 neurons** in first hidden layer, **16 neurons** in the second hidden layer and **drop-out rate of 0.1** and has the **accuracy of 100 %**.

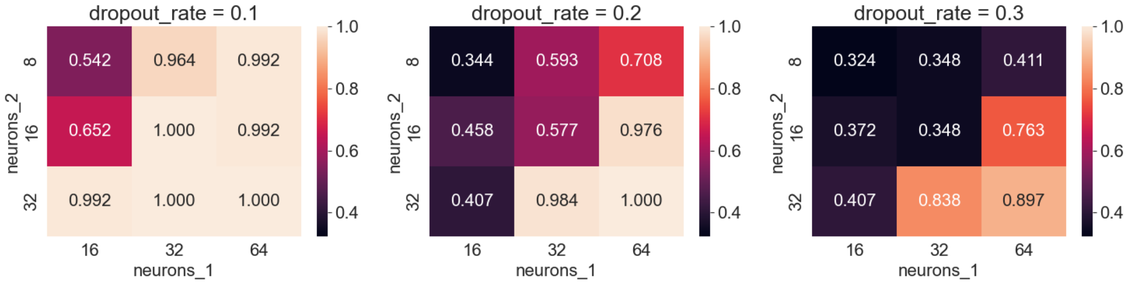


Figure 9: Neural network: Heat-maps for best parameters

4.2.6. Disease probability

After training the dataset to predict diseases based on symptoms, the Random forest and Neural network algorithms were used to generate a probability table for diseases based on the patient's symptoms.

Both the Random forest and Neural network models have the `predict_proba()` method, which estimates the probabilities of each class based on the provided symptoms. This functionality was implemented into the trained models.

For the symptoms `itching` and `skin_rush` the top five diseases with the highest probabilities are shown for Random forest in Table (1) and for the Neural network in Table (2).

Random forest	
Disease	**Probability**
Fungal infection	0.51
Drug Reaction	0.14
Acne	0.08
Impetigo	0.04
AIDS	0.03

Table 1: Diseases and Probabilities from Random forest

Neural network	
Disease	**Probability**
Fungal infection	0.31
Dengue	0.24
Drug Reaction	0.17
Chicken pox	0.05
Hypothyroidism	0.04

Table 2: Diseases and Probabilities from Neural network

The most common disease predicted by both Random forest and Neural network models is Fungal infection. The difference between the models' estimations is obvious already for the second most likely disease, with Dengue being the second most likely for the Neural network and Drug reaction for Random forest.

5. Discussion

This paper introduces five different models for disease prediction. Table (3) compares the performance of various methods used to create disease prediction models. The accuracy scores show that all methods have strong predictive capabilities. Logistic regression, KNN, Random forest, and Neural network models all have perfect accuracy, indicating their potential for accurate disease prediction. Although slightly less accurate than the other methods, the Decision tree model still demonstrates promising results. This may be due to the high complexity of the data used. The Decision tree relies on a series of if-else conditions to make predictions, which may make it challenging to classify the data correctly.

Method	**Logistic regression**	**KNN**	**Decision tree**	**Random forest**	**Neural network**
Accuracy in %	100	100	76	100	100

Table 3: Comparison of the methods accuracy

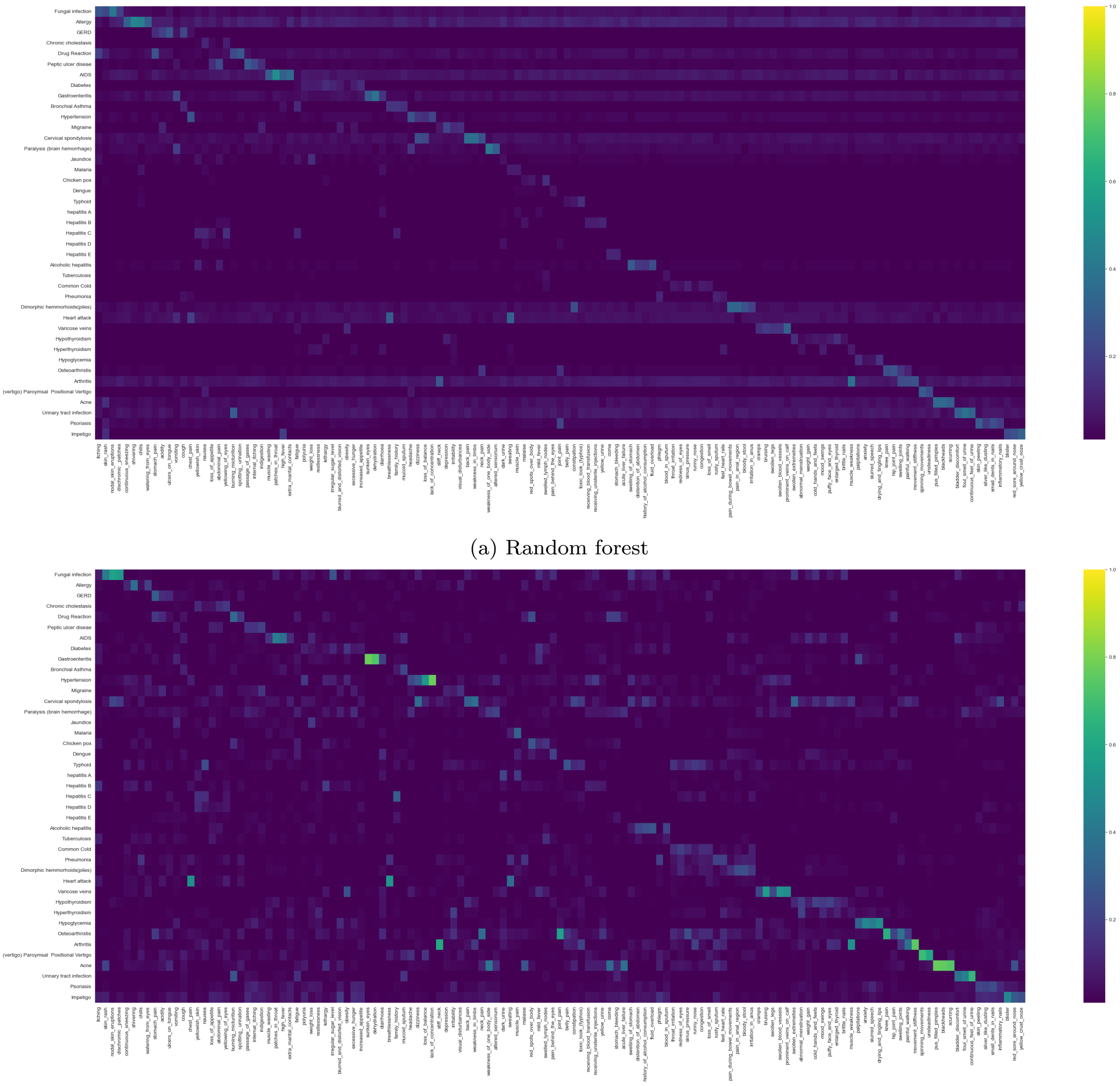
The 100 % accuracy of all algorithms can be attributed to a large number of duplicate and similar entries in the dataset. To overcome this problem, data sampling was attempted, a **custom train-test split** function, and a custom grid search function. However, even with these precautions, the issue of overfitting has only been reduced to a certain extent. As a result, it is difficult to reliably and realistically evaluate the performance of the disease prediction models.

The `CustomGridSearch` algorithm can introduce biases in the tuning of hyperparameters, as it is less methodical and rigorous than a typical stratified **k-fold**. One possible solution is to implement an equivalent of **k-fold** that prevents duplicates from being shared between the train and test sets.

Two algorithms, Random forest and Neural network, were used to create **probability-based predictions**. This approach provides a more transparent prediction tool by displaying the certainty of each model and possible alternative diseases.

The prediction probabilities of Random forest and Neural network models can be compared by creating the following heatmaps. A prediction is made for each individual symptom, and the probability associated with each disease outputted by each model is displayed here. This gives insight into the models' thought processes, as the dataset does not contain entries with only one symptom.

While the Neural network seems to make organic predictions (Figure (10a)), there are lateral line artifacts present in the Random forest graph (Figure (10b)). This suggests that in unknown territories, Random forest may favor certain diseases over others.



(a) Random forest

(b) Neural network

Figure 10: Single symptom probability heatmap

It should be noted that this prediction only shows the confidence in the models as if there is exactly one disease causing the symptoms, because of the dataset it was trained on. The models are unable to estimate if there is no disease that fits the symptoms, nor if there may be multiple concurrent diseases.

6. Conclusions

The experiment found that all algorithms, including **Logistic regression**, **KNN**, **Random forest**, and **Neural network** achieved high accuracy scores. However, the **Decision tree** model had a lower accuracy score and is therefore not as effective for this task. Additionally, the issue of duplicate entries in the dataset led to an overfitting problem, which resulted in inflated accuracy scores.

The experiment also involved the use of probabilities in Decision tree and Neural network models. By displaying the ”certainty” of the model, the estimations become more transparent and exploitable. A tool could be designed to display both models' predictions or take the mean of the two, benefiting from their joint processes.

The developed models could, to some extent, contribute to the development of healthcare tools to reduce the risk of misdiagnosis.

7. Contributions

Adela Ondrouchova, Patricia List, Tom Bourjala, and Wiktoria Ciasnocha participated in and contributed equally to this project with the support from Abdolrahman Peimankar. The custom grid search algorithm was made by Tom Bourjala.

References

[1] Diagnostic errors in the emergency department: A systematic review (16.05.2023).
URL https://effectivehealthcare.ahrq.gov/products/diagnostic-errors-emergency/research#field_report_title_1

[2] P. Hema, N. Sunny, R. Venkata Naganjani, A. Darbha, Disease prediction using symptoms based on machine learning algorithms, in: 2022 International Conference on Breakthrough in Heuristics And Reciprocation of Advanced Technologies (BHARAT), 2022, pp. 49–54. doi:10.1109/BHARAT53139.2022.00021.

[3] K. S. Kumar, M. Sai Sathya, A. Nadeem, S. Rajesh, Diseases prediction based on symptoms using database and gui, in: 2022 6th International Conference on Computing Methodologies and Communication (ICCMC), 2022, pp. 1353–1357. doi:10.1109/ICCMC53470.2022.9753707.

[4] Sneha Grampurohit, Chetan Sagarnal, 2020 international conference for emerging technology (incet): Belgaum, india, jun 5-7, 2020, Ph.D. thesis, Piscataway, NJ. doi:10.1109/INCET49848.2020.
URL https://ieeexplore.ieee.org/servlet/opac?punumber=9145687

[5] P. Patil, Disease symptom prediction (24.05.2020).
URL https://www.kaggle.com/datasets/itachi9604/disease-symptom-description-dataset

[6] A. C. Müller, S. Guido, Introduction to machine learning with Python: A guide for data scientists, first edition Edition, O'Reilly Media, Sebastopol, CA, 2017.
URL https://search.ebscohost.com/login.aspx?direct=true&scope=site&db=nlebk&db=nlabk&AN=1361381

[7] C.-H. Weng, T. C.-K. Huang, R.-P. Han, Disease prediction with different types of neural network classifiers, Telematics and Informatics 33 (2) (2016) 277–292. doi:10.1016/j.tele.2015.08.006.
URL https://www.sciencedirect.com/science/article/pii/S0736585315001057

Appendix

Appendix .1. List of all diseases

1. Acne
2. AIDS
3. Alcoholic hepatitis
4. Allergy
5. Arthritis
6. Bronchial Asthma
7. Cervical spondylosis
8. Chicken pox
9. Chronic cholestasis
10. Common Cold
11. Dengue
12. Diabetes
13. Dimorphic hemmorhoids (piles)
14. Drug Reaction
15. Fungal infection
16. Gastroenteritis
17. GERD
18. Heart attack
19. Hepatitis A
20. Hepatitis B
21. Hepatitis C
22. Hepatitis D
23. Hepatitis E
24. Hypertension
25. Hyperthyroidism
26. Hypoglycemia
27. Hypothyroidism
28. Impetigo
29. Jaundice
30. Malaria
31. Migraine
32. Osteoarthristis
33. Paralysis (brain hemorrhage)
34. Peptic ulcer diseae
35. Psoriasis
36. Pneumonia
37. Typhoid
38. Tuberculosis
39. Urinary tract infection
40. Varicose veins
41. (vertigo) Paroymsal Positional Vertigo

Appendix .2. List of all symptoms

1. abdominal_pain
2. abnormal_menstruation
3. acidity
4. acute_liver_failure
5. altered_sensorium
6. anxiety
7. back_pain
8. belly_pain
9. blackheads
10. bladder_discomfort
11. blister
12. blood_in_sputum
13. bloody_stool
14. blurred_and_distorted_vision
15. breathlessness
16. brittle_nails
17. bruising
18. burning_micturition
19. chest_pain
20. chills
21. cold_hands_and_feets
22. coma
23. congestion
24. constipation
25. continuous_feel_of_urine
26. continuous_sneezing
27. cough
28. cramps
29. dark_urine
30. dehydration
31. depression
32. diarrhoea
33. dischromic _patches
34. distention_of_abdomen
35. dizziness
36. drying_and_tingling_lips
37. enlarged_thyroid
38. excessive_hunger
39. extra_marital_contacts
40. family_history
41. fast_heart_rate
42. fatigue
43. fluid_overload
44. foul_smell_of urine
45. headache
46. high_fever
47. hip_joint_pain
48. history_of_alcohol_consumption
49. increased_appetite
50. indigestion
51. inflammatory_nails
52. internal_itching

53. irregular_sugar_level
54. irritability
55. irritation_in_anus
56. itching
57. joint_pain
58. knee_pain
59. lack_of_concentration
60. lethargy
61. loss_of_appetite
62. loss_of_balance
63. loss_of_smell
64. malaise
65. mild_fever
66. mood_swings
67. movement_stiffness
68. mucoid_sputum
69. muscle_pain
70. muscle_wasting
71. muscle_weakness
72. nausea
73. neck_pain
74. nodal_skin_eruptions
75. obesity
76. pain_behind_the_eyes
77. pain_during_bowel_movements
78. pain_in_anal_region
79. painful_walking
80. palpitations
81. passage_of_gases
82. patches_in_throat
83. phlegm
84. polyuria
85. prominent_veins_on_calf
86. puffy_face_and_eyes
87. pus_filled_pimples
88. receiving_blood_transfusion
89. receiving_unsterile_injections
90. red_sore_around_nose
91. red_spots_over_body
92. redness_of_eyes
93. restlessness
94. runny_nose
95. rusty_sputum
96. scurring
97. shivering
98. silver_like_dusting
99. sinus_pressure
100. skin_peeling
101. skin_rash
102. slurred_speech
103. small_dents_in_nails
104. spinning_movements
105. spotting_ urination
106. stiff_neck
107. stomach_bleeding
108. stomach_pain
109. sunken_eyes
110. sweating
111. swelled_lymph_nodes
112. swelling_joints
113. swelling_of_stomach
114. swollen_blood_vessels
115. swollen_extremeties
116. swollen_legs
117. throat_irritation
118. toxic_look_(typhos)
119. ulcers_on_tongue
120. unsteadiness
121. visual_disturbances
122. vomiting
123. watering_from_eyes
124. weakness_in_limbs
125. weakness_of_one_body_side
126. weight_gain
127. weight_loss
128. yellow_crust_ooze
129. yellow_urine
130. yellowing_of_eyes
131. yellowish_skin