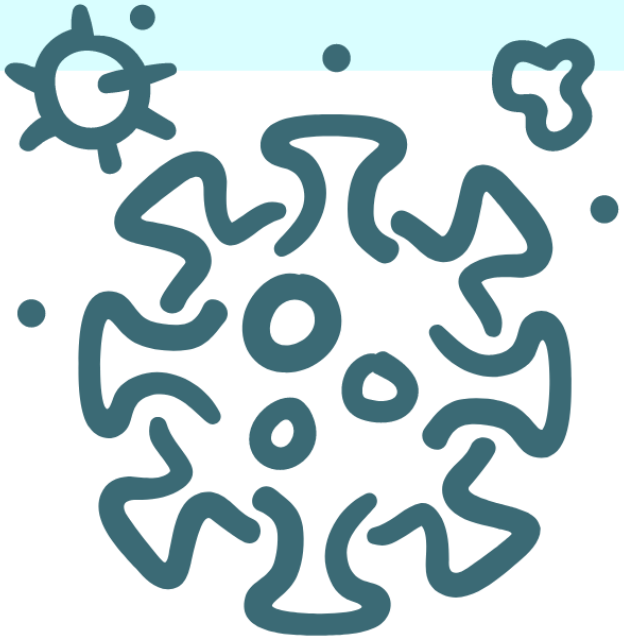
Project Report for Disease Prediction



This report examines the application of disease prediction tool in society and our team's approaches to build this tool. It details the reasonings behind choosing KNearest Neighbors machine learning model for disease prediction. The report concludes with possible improvements for the model as suggested by the team.

Project Report for Disease Prediction

A Report to Health Insights Regarding a Disease Prediction Tool

Prepared for:

Health Insights
Supervisors

Prepared by:

Maize & Blue Data Analytics, Team YES WAY
Sachit Soni, *Data Scientist*
Eric Yuan, *Data Scientist*
Yashoditya Watal, *Data Scientist*
Anny Wu, *Data Scientist*

April 19, 2022

Table of Contents

List of Figures

1. Executive Summary

Health Insights, a healthcare startup in Silicon Valley, has asked our organization, Maize & Blue Data Analytics, to build a tool to predict a disease that a patient is fighting given a list of their exhibited symptoms. We were provided with a dataset of real past prognoses and symptoms. Using this dataset, our team used various machine learning methods to create an accurate predictive model. In addition to this, we conducted additional analysis to visualize diseases with similar symptoms.

Our team built our model using *Python*, an interpreted programming language. We utilized the *Python* libraries *Pandas, NumPy, Sci-Kit Learn,* and *Matplotlib* to import the dataset, create our models, and visualize the results.

Our path of action was largely influenced by the data science lifecycle. As such, our work consisted of the following tasks:

- Cleaning and organizing the dataset
- One-hot encoding the dataset for our computer to process
- Performing PCA feature reduction to create simpler, quicker models
- Using ML methods to create models from both the PCA reduced dataset and full-featured dataset
- Performed clustering analysis to visualize diseases with similar symptoms
- Analyzing our findings and evaluate our best performing models

In the production of this tool, we produced a series of models and conducted extensive tests on all of them to determine the “best” model. Following this process, our team recommends that the PCA-Reduced K-Nearest Neighbors should be used as the predictive tool requested by Health Insights. This model produced the best, with an accuracy of 93.4 %, of our models and is capable of taking in new data efficiently, making it both accurate and adaptable. The model’s main flaw, however, is that it does not scale well with larger datasets, but this is not a concern given the current dataset. Finally, our team built clustered groupings for diseases with similar symptoms which can be used as a reference for medical staff who are using the tool.

This report gives details about our project’s goals and requirements, solution concepts, technical approaches, findings, results, and possible future improvements.

2. Introduction

Health Insights, a healthcare startup in Silicon Valley, approached our organization, Maize & Blue Data Analytics (MBDA) to develop a disease prediction tool. Our subteam of MBDA, YES WAY, was tasked to build this tool, which will assist medical personnel worldwide. This tool will be used to predict a diagnosis that a patient is fighting given their symptoms. Health Insights has provided a data set of real medical diagnosis and their symptoms. Using this dataset, our team used machine learning methods and models to build this predictive tool.

When applied to the medical field, the predictions that this tool will create will help professionals and patients respond to illnesses and afflictions more efficiently, and effectively, pointing them in the right direction of care and treatment. This report documents the development process of the tool, going over the following:

- The project's goals and objectives to fulfill Health Insight's requests.
- Our approach to the project and potential challenges.
- Our findings and the final model
- Analysis of results and future applications/updates

3. Background

Given the importance of this project and its applications, our team decided to familiarize ourselves with other modern examples of disease prediction tools. This would provide us with a basic understanding of the benefits and limitations of such a model, from the perspective of other developers. It would also highlight any potential points of difficulty that we would face during development. Our team also wanted to understand the direct and indirect effects that the model's application may have when deployed for professional use. This understanding would enable the team to pay special attention to points of interest that we would have to address throughout development, preventing the team from unintentionally neglecting important factors.

3.1. Solutions to Disease Prediction in Use Today

Disease prediction through ML models has been key to streamlining the diagnosis process. The usage of these models on large datasets, similar to the one given to us by Health Insights, is widespread in the medical field today. In addition, similar studies have explored the accuracy of ML models to predict diseases. Specifically, in a paper written by Y. Deepthi and his colleagues, the implementation of the random forest, decision tree, and Naive Bayes classifiers all formed possible solutions, accurately predicting up to 91% of the test data fed into it [1].

Using ML in disease prediction is not just limited to these three models. In another paper by Dong Jin Park and his colleagues, a new neural network accurately predicted 91% of diseases [2]. ML models may even be better at predicting disease than medical practitioners. A Bayes classifier trained by was used to predict diseases better than 72% of practitioners [3].

3.2. Advantages and Impact of Our Tool

Many human lives can be saved by the efficiency and fast responses initiated by the use of an accurate disease predictive tool, such as the one Health Insights proposed. In the fast-paced and hectic world of the medical field, a split-second decision could make or break the lives of those suffering from diseases. By having a second opinion, through this tool, it is possible that doctors can make more accurate and efficient decisions while on the job. The tool can also serve as an organization tool, pointing a medical professional in the general direction of an affliction's treatment and saving time in the initial searching process. It is important to stress, however, that the production of this tool must be made with the utmost care, as the stakeholders directly impacted by it may face terrible consequences if it were to prove to be inaccurate. If a doctor were to wrongly diagnose a disease based on this tool, then lives could potentially be ruined. This is something to take incredibly seriously and must be considered when training the model.

The majority of the stakeholders for our project revolve around medical treatments as well. As stated earlier, patients benefit from more accurate and efficient predictions. They can also avoid unnecessary medical tests by reducing the number of tests ordered for diseases with lower probabilities as identified by the model. Researchers can rely on the model to decide their focus areas and can dedicate more time to treating symptoms that are associated with more diseases. Overall, this tool has the potential to make a doctor's job much simpler by providing a faster method of identification, thus saving more lives in the process.

Finally, two large stakeholders impacted by the development of this tool are Maize and Blue data Analytics, and Health Insights. Health Insights is dependent on MBDA for the production of this tool. As an up-and-coming startup in the healthcare industry, Health Insights may rely heavily on the production of such a tool to utilize and/or monetize. The future development, and the very existence of the startup, may be dependent on the careful production of tools such as this. As Health Insights is a client for MBDA, their success is also inextricably linked with MBDA's. In addition to this, if this tool were to be inaccurate or was to be made hastily, then that would reflect poorly on both MBDA and Health Insights (in addition to the medical harm that may be caused). The production of this tool is to be held to the highest standards to keep from reflecting negatively on both of these companies in the short and long run.

4. Project Goals

The goal of this project is to clean and analyze a set of data with information regarding certain diseases and the symptoms that are exhibited in patients. The purpose of this product will be to build a predictive tool that can classify a disease based on the symptoms exhibited by a patient. The steps in order to do so, are as follows:

1. Clean the data so that it is correctly partitioned into the diseases and the symptoms each patient exhibits with that disease
2. One hot encode the cleaned dataset so that the model can be trained on a series of Machine learning models
3. Apply Principal Component Analysis (PCA) to the one-hot-encoded symptoms set to reduce the symptom dimensions to their optimal amount (determined by the team and data on the model gathered by them)
4. Create and train multiple machine learning models that can accurately classify diseases based on the symptoms given using the principal components as the result of PCA
5. Analyze the hyperparameters used in each model and thoroughly test other such hyperparameters to have as accurate a model as possible. (Using GridsearchCV when possible)
6. Repeat steps 4 and 5 using the feature set with all the symptoms rather than the PCA reduced feature set
7. Analyze the results using a series of different metrics on all the models we create, such as training and testing accuracy, precision, recall, f1-score, etc.
8. Apply K-means clustering on diseases with corresponding symptoms, therefore visualizing the diseases that have similar, and different, sets of symptoms. (Showing results visually on 3D-plane)

While these goals broadly encompass the steps required to thoroughly complete the development of this tool, not all points are examined. Each step outlined above will be gone through in the following report in more detail.

5. Project Overview

The data science lifecycle (as shown below) largely influenced our process in the creation of our tool. The business understanding stage is covered by Health Insights and the data collection stage was provided by the dataset of diagnoses and symptoms.

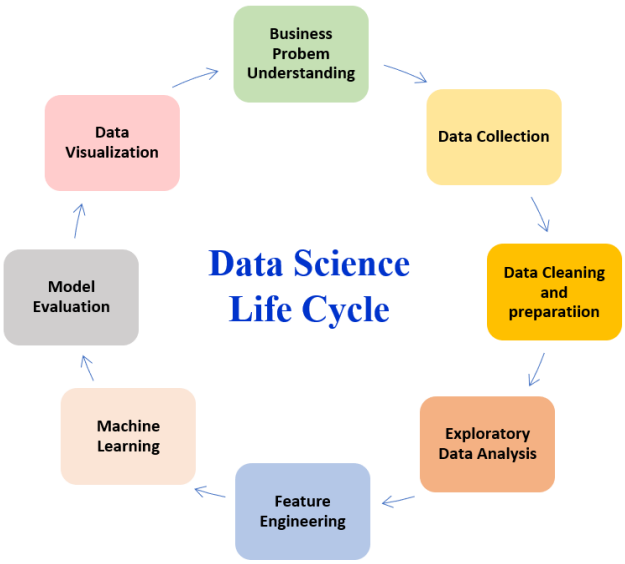


Figure 1: The Data Science Lifecycle

Figure 1: Outline of a standard data science lifecycle. Data scientists need to first understand a business problem, collect, clean, prepare, and analyze the data, do feature engineering, use machine learning to train models, evaluate the model, and then visualize the data.

[4]

Following this model in Figure 1, our team performed a stakeholder analysis (outlined in section *3.2 Advantages and Impact of our tool*) and listed possible ways to approach the project. We began by analyzing the primary foci that would be used to assess the performance and desired qualities of a final product. The evaluation of our models was largely based on two areas of focus: Accuracy, and Adaptability.

5.1. Accuracy

It is imperative that the tool we produce can provide consistent, reliable, and accurate predictions. As our tool will be practically applied to patients awaiting treatment or advice, inaccurate predictions could prove detrimental to a patient's condition and prolong suffering. In extreme cases, this could even endanger entire treatment plans and human lives. With this in mind, we wanted to ensure that our model's prediction accuracy was at least 90%. In addition to this, our model's recall (True Positive / (True Positive + False Negative)), which represents the model's rate of predicting a disease when the patient actually has it, should be maximized in order to avoid false negatives (a sick patient is not diagnosed with the disease they have) in

diagnoses. In addition, we wanted to avoid models that are more prone to overfitting, which could potentially create a less accurate model.

We understand that our solution may not be perfectly accurate, and as such, we have created groupings of diseases with similar symptoms. In a situation where the model predicts a disease inaccurately, these groupings can be used to determine a potential second diagnosis.

Nevertheless, MBDA wishes to stress that this tool will simply be an aid to doctors looking to diagnose ill patients. This tool should not be taken as a 100% guarantee or predictor of any diseases or afflictions using a set of symptoms presented by a patient. This model is not expected to be completely infallible, and MBDA recognizes that perfect accuracy is nigh impossible. We wish to get as close as possible to correct classification; however, any potential inaccuracies, no matter how unlikely, will inevitably need to be dealt with by experienced doctors in the field.

5.2. Adaptability

Healthcare services may want to expand the use of the tool to larger datasets by adding new patients with different diseases and symptoms. As such, we want our model to be adaptable and be able to integrate different datasets. Therefore, the model should be able to handle the addition of new data and ensure that its accuracy and efficiency do not fall by any significant margin. In addition, the model should be able to retrain itself without a lengthy training time.

6. Solution Concepts

Our team chose the PCA-Reduced K-Nearest Neighbors (KNN) Classifier as the model to represent our tool. This model had great accuracy on the test dataset - 93.38% - tied for the best among our models, among the Multi-layer Perceptions (MLP), the logistic regression, and the decision tree models. In addition, the model can be trained in 0.04 seconds. Furthermore, KNN is a lazy learning algorithm, which means it requires no training time. The major downside to KNN is its inability to scale well with larger datasets. However, we do not believe this is a major issue due to the relatively small size of the original dataset. It is still possible that larger datasets may be used to train the tool. In this event, we recommend the use of an MLP or Multi-layer-perceptron, with the PCA reduced feature set, as this model had the same accuracy and test metric values as the KNN model. MLP model can handle larger datasets more reliably at the expense of efficiency. Ultimately, the KNN model displayed both great accuracy and adaptability for the data-set MBDA was given, both of which are imperative in the fast-paced healthcare system.

7. Technical Approaches - Model Creation

Our approach largely consisted of steps in the Data Science Lifecycle (see *Figure 1*): exploratory data analysis, data cleaning and preparation, and PCA Reduction (feature engineering). In addition, we also performed test/train split optimization and hyperparameter optimization.

7.1. Exploratory Data Analysis

Our team first imported the dataset into a DataFrame from a CSV file. The DataFrame showed the data we received included 5,362 diagnoses and their symptoms (with repetitions). We found that there were a total of 41 unique diagnoses and 131 unique symptoms with differing frequencies as shown in Figure 2.

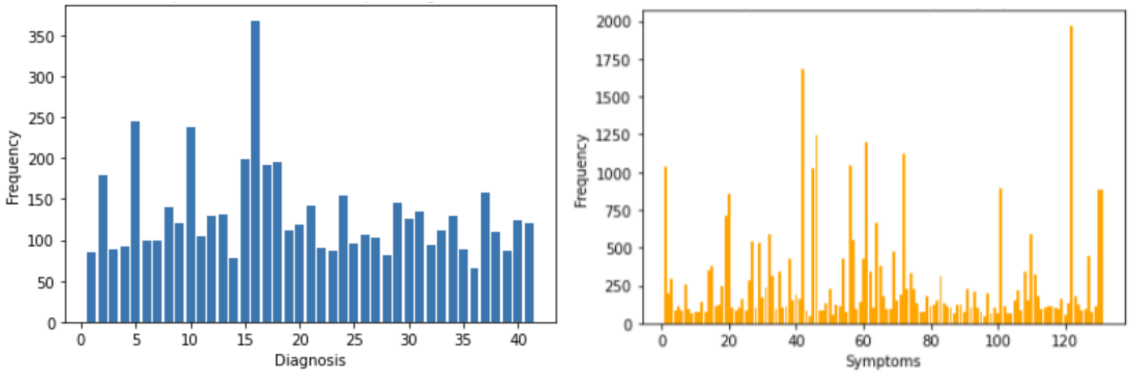


Figure 2: Frequency of Unique Diagnosis and Symptoms

Figure 2: Frequency of 41 unique diagnoses (on the left) and 131 unique symptoms (on the right). Our most common diagnosis was fungal infection and our most common symptom was vomiting, which are shown by the two tallest bars. As an example, the most frequent symptoms in the table are detailed in Appendix F.

7.2 Data Cleaning and Preparation

Our initial step in data cleaning was to split the data into columns of patient symptoms and their respective diagnoses. Each symptom that is originally stored in a list is then stored in a separate column on the same row as their diagnosis. Following this step, we used the DataFrame with those columns to one-hot encode the data into a feature matrix, which is used for the computer to read into machine learning libraries. The process of one-hot encoding converts each symptom written into text into 1s and 0s. The number 1 shows that a symptom is associated with a specific diagnosis and 0 otherwise.

One-hot encoding allows our data to be trained efficiently by machine learning libraries. However, prior to training and testing our dataset, we performed PCA-feature reduction, find an optimal test-train set split, and perform hyperparameter optimization on the following models, which are discussed in detail further below:

1. K-Nearest Neighbors
2. Multilayer Perceptron (Neural Network)
3. Logistic Regression
4. Gradient Boosting Classifier

5. Random Forest Classifier
6. Decision Tree Classifier
7. Perceptron

7.3 PCA Feature Reduction

After completing one-hot encoding, we performed Principal Component Analysis (PCA) to reduce the number of features (symptoms) our model has to handle. Reducing the number of features can lower the risk of overfitting as well as speed up our model's training time. We chose to reduce our model to 40 features as the explained variance ratio at 40 features does not change much with an increase in feature size (the turning point is at around 94% of the explained variance ratio). The variance ratio is effectively the value of a feature in its predicting power. The value of a feature in its variance ratio diminishes as there are more features. Using this reduced feature (which still gives the majority of the explained variance ratio) set allows us to save time for model training. Figure 3 below shows the relation between the principal component index, which are the features, and the explained variance ratio in PCA.

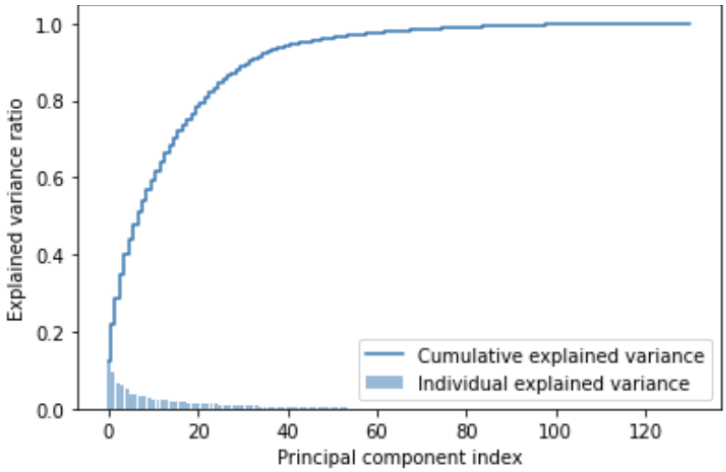


Figure 3: Explained Variance Ratio vs Principal Component Index (# features)

Figure 3: The principal component index ranges from 0 to 131. The explained variance ratio increases exponentially from the index of 0 to 40. After the index of 40, the change in variance ratio effectively flattens out, and there is little value in adding another feature.

7.4 Train/Test Split Optimization

Following PCA feature reduction, we found an optimal train/test split, which we used to split the data into sets that our models will train on (training set) and the data that our models will be evaluated on (test set).

To do this, the team trained 9 random forest models (explained in detail below), each with a differing train-test split, ranging from a training size of 0.1 (10%) to 0.9 (90%). Figure 4 shows the cross-validation scores (CV scores) and test accuracy scores for each of the models trained.

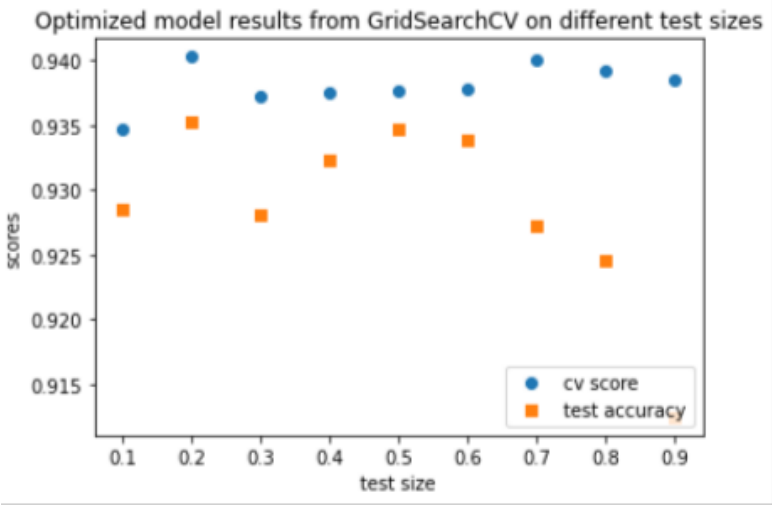


Figure 4: Train/Test Split Optimization for Different Test Sizes on Random Forests

Figure 4: Test Size is plotted with the CV Scores and Test Accuracy. The test size is the percentage of the test size in decimal form. The first Random Forest started at a 10% test size, the following eight were each incremented by an additional 10%.

We chose a testing size of 40% due to the closeness of the cross-validation score and test accuracy. This indicates that this test size would lead to less overfitting, leading to a more adaptable and generalizable model. Although a testing size of 40-60% all produces similar scores, the number 0.4 guarantees that we still have a relatively large training size. As a result, we applied this train/test split to use on all of our models to be developed in the future.

7.5 Hyperparameter Optimization

After optimizing our train/test split, we wanted to optimize each hyperparameter to produce better accuracy. Our team used GridSearchCV, a *Sci-Kit Learn* function, to perform this task. GridSearchCV works by creating every combination of hyperparameters the user inputs and producing a validation score for each combination. The combination with the highest score is the combination of hyperparameters used on the final model. The final optimized hyperparameters that each model trained used can be found in *Appendix C: Performance Investigation Results*

8. Overview of Models Chosen

We performed the above three tasks on all seven models (see *pg.10*). Our hyperparameter optimization and motivation for each model are detailed below.

8.1 K-Nearest Neighbors

The KNN Classifier works by first plotting out all the data points. The classifier then takes in an input test data point and searches for the k, a predetermined number, nearest training data points to it. Due to the nature of KNN, it requires no training time - making it very efficient. Because it requires no training time, the training data can be changed without any major time concerns. In the situation where more diseases could be added to the training dataset, the KNN classifier would have no problem adapting to the new data. However, this comes with drawbacks as the KNN classifier does not scale well with larger datasets. However, since the dataset provided was relatively small (~5, 300 cases), the KNN classifier would be a great model for this dataset.

In addition to the size of the dataset, determining the k-value is also critical to the accuracy of the KNN classifier. A low-k value may be too specific whereas a high k-value may be too generalizable. Our classifier used GridSearchCV (detailed below) to determine our k-value to be 9.

8.2 Multi-Layered Perceptron (Neural Network)

Development then focused attention on creating a multi-layered perceptron model. Historically, this learning algorithm is well known for its adaptability to large quantities of data, its learning efficiency, and its ability to develop fast-performing, accurate, models. The models it creates are also distinctly adaptable to non-linearly separable data, which proved beneficial when accounting for diseases with overlapping symptoms in their lists. The only notable limitation of the model is the computational expenses this model requires. Using *Sci-Kit Learn's* neural network classifier, we created a model with 300-neurons and 200-neuron layers.

8.3 Logistic Regression

A logistic regression classifier utilizes a set of defined parameters to make its predictions. The algorithm generates sets of linear partitions between points of data, using them to generate probabilities of each point being in each respective category. The learning algorithm is an industry classic, proven to be reliable in both linear and nonlinearly separable classifications. Its simplicity also makes it very reliable for predicting unlabeled data within a set of parameters. A notable flaw with this algorithm is its inability to differentiate dependent and independent variables which could have led to inaccurate results.

8.4 Gradient Boosting Classifier

A gradient boosting classifier is an ensemble algorithm that utilizes the averaged results of multiple models to create a balanced classification. By utilizing a collection of models - both

good and bad - one model's biases are balanced by the biases of another. While classifiers produced using this learning algorithm have returned high predictive accuracies, the model is prone to overfitting and requires a significantly longer training period. In our experimentation, even with a PCA-reduced dataset, training time lasted well over an hour. Its reputation, however, still made it worth investigating, and we trained the models accordingly.

8.5 Random Forest Classifier

The random forest classifier is an ensemble method that involves using many decision trees, classifying with the majority result of each decision tree. Because a random forest uses multiple different decision trees to classify, its predictive power is almost always greater than that of a single decision tree. A study done by Manuel Fernández Delgado and his colleagues showed that out of 179 classifiers from 17 families of classifiers, random forests were the most likely to perform the best **[5]**. Our motivation behind choosing this classifier was due to its consistent and accurate results.

GridSearcCV determined our hyperparameters. It chose the criterion as entropy and the depth of the trees as 5 for both the PCA-Reduced models and full-featured models.

8.6 Decision Tree Classifier

The decision tree classifier trains its dataset by learning a series of questions through the training data. The decision tree algorithm is built in a way that prioritizes the maximum information gain from a question. Decision trees are not very practical for large models such as this because they are very sensitive to small variations in the data. This makes them much more likely to overfit, causing lower accuracy. Our motivation behind using this model is to compare it to the Random Forest classifier, which uses many decision trees to classify, to compare their results.

GridSearchCV determined our hyperparameters. It chose the criterion to determine how the features of the dataset should be split as 'entropy'. In addition, it chose the depth of our tree as 10 for the PCA-Reduced version of the decision tree and 15 for the full-featured version of the decision tree.

8.7 Perceptron

The perceptron algorithm is a binary classifier and one of the earliest forms of a neural network. However, this algorithm has many disadvantages: it can only converge if the data is linearly separable, and even when it can converge it takes much longer than other models. Our main motivation for choosing this model is to compare it to the other models and to set a benchmark value for our accuracy.

9. Technical Approach - Clustering

Following our development and analysis of a satisfactory prediction model, the team conducted additional analysis by visualizing the results using K-means clustering. After generating average feature vectors for each disease using the original one-hot encoded data, the team utilized these vectors to categorize the diseases within general "clusters." These clusters highlighted diseases with similar sets of symptoms, revealing underlying patterns.

9.1. Producing Average Feature Vectors

In order to develop accurate feature vectors for the K-means Clustering model, we began with the original one-hot encoded data that was not subjected to dimensionality reduction. In this process, every instance of each disease is "averaged," removing the inconsistencies in symptoms by generating an average feature vector. An average feature vector provides the probability that a symptom is associated with a specific disease. A series of these vectors, one for each disease, was then stored in a dataset for further analysis.

9.2. Optimizing Clusters Value

Using the feature vector dataset and ScikitLearn's KMeans learning algorithm, our team experimented with a range of 39 K-cluster values. We then plotted the distortion of each resulting model with its average silhouette coefficient. This coefficient determines how good a specific clustering technique is, with the value 1 being the best. We analyzed Figure 5 for the k-value with the high average silhouette coefficient while maintaining low distortion. We found this value to be 11 distinct clusters - the contents of which can be found in Appendix D.

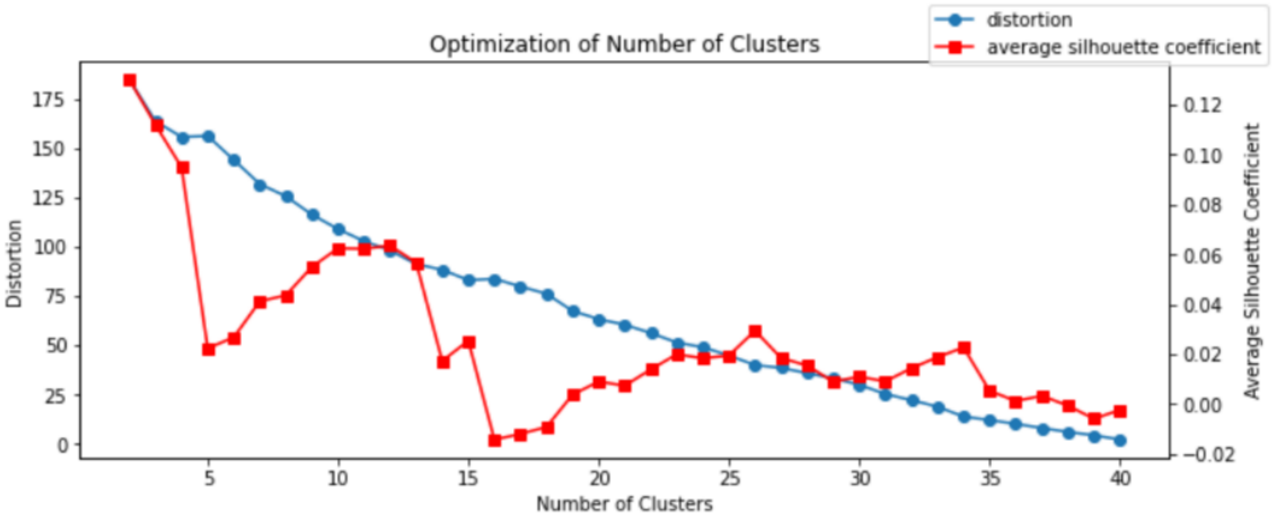


Figure 5: Distortion and Average Silhouette Coefficient Influenced by The Number of Clusters

Figure 5: The number of clusters is plotted against distortion and average silhouette coefficient. Distortion decreases as the number of clusters increases. The average silhouette coefficient fluctuates with a change in the number of clusters which indicates a change in cluster performance.

9.3. Reducing Dimensions for Visualization

In order to better visualize and understand the clusters, we performed dimensionality reduction using PCA on the dataset of average feature vectors. We reduce the dimensionality down to three components. Figure 10 provides a visualization of our dataset in three dimensions with the same colors indicating the same cluster group. Some clusters contain 19 diagnoses while others only contain one. This difference can be explained by the explained variance ratio of the three components, 0.31. It indicates that only 31% of the variance is described by three components which means that the distribution of our actual data will look quite different than what is shown below. Additional information on the clusters themselves can be found in Appendix D.

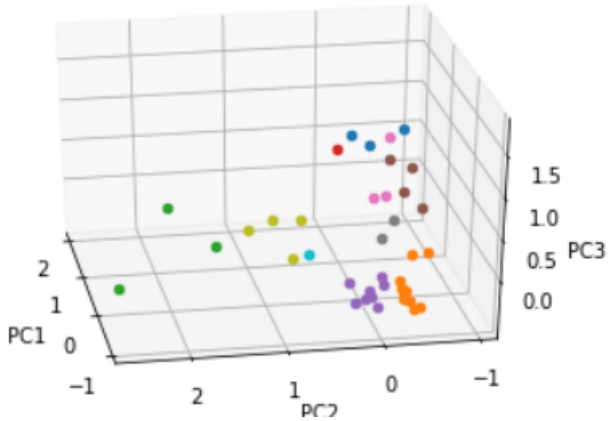


Figure 6: Clusters of Average Feature with 11 Clusters

Figure 6: Each unique diagnosis is plotted in a 3D graph. Each data point has one of the eleven colors assigned according to its cluster

From this model, some of the data's limitations become very evident. The similarities among the clusters, such as instances of Bronchial Asthma and tuberculosis, demonstrate how this model may be unreliable when diseases have multiple common symptoms. Therefore, it should be applied mostly as a beneficial tool rather than a steadfast diagnosing model. It does reveal a secondary use - organizing symptoms to point towards a general group of diseases for a medical professional to consider. This could very well be extremely beneficial, as individual symptoms may point to a variety of ailments. The system can efficiently parse through these patterns and point medical professionals in the right direction.

10. Results & Analysis

After the preparation of our k-means clustering model, we ran this model to produce groupings of similar diseases. See *Appendix D* for groupings.

In addition, after performing PCA reduction, test-train split optimization, and hyperparameter optimization on our classification models, we trained the predetermined models using both our PCA-reduced and full-featured dataset. We then tested these models on several metrics, such as balanced accuracy, precision, recall, and f-1 score. On the training set, our PCA-Reduced models were relatively similar in accuracy to our full-featured models. However, on the test set our PCA-Reduced models were more accurate overall than the full-featured models. We believe this is because the full-featured model is more likely to overfit to the training set than the PCA-Reduced dataset.

When evaluating models, we took training set accuracies into negligible consideration as the models were trained on that dataset so they would surely perform well on that them. Our main focus was the models' performance on the testing set accuracy. Overall, the PCA-Reduced Logistic Regression, Decision Tree, K-Nearest Neighbors, Multi-Layered Perceptron and the full-featured Logistic Regression, K-Nearest Neighbors, and Multi-Layered Perceptron tied for the best accuracy on the test dataset - 93.34%. The accuracies are displayed in Figure 7 below**.**

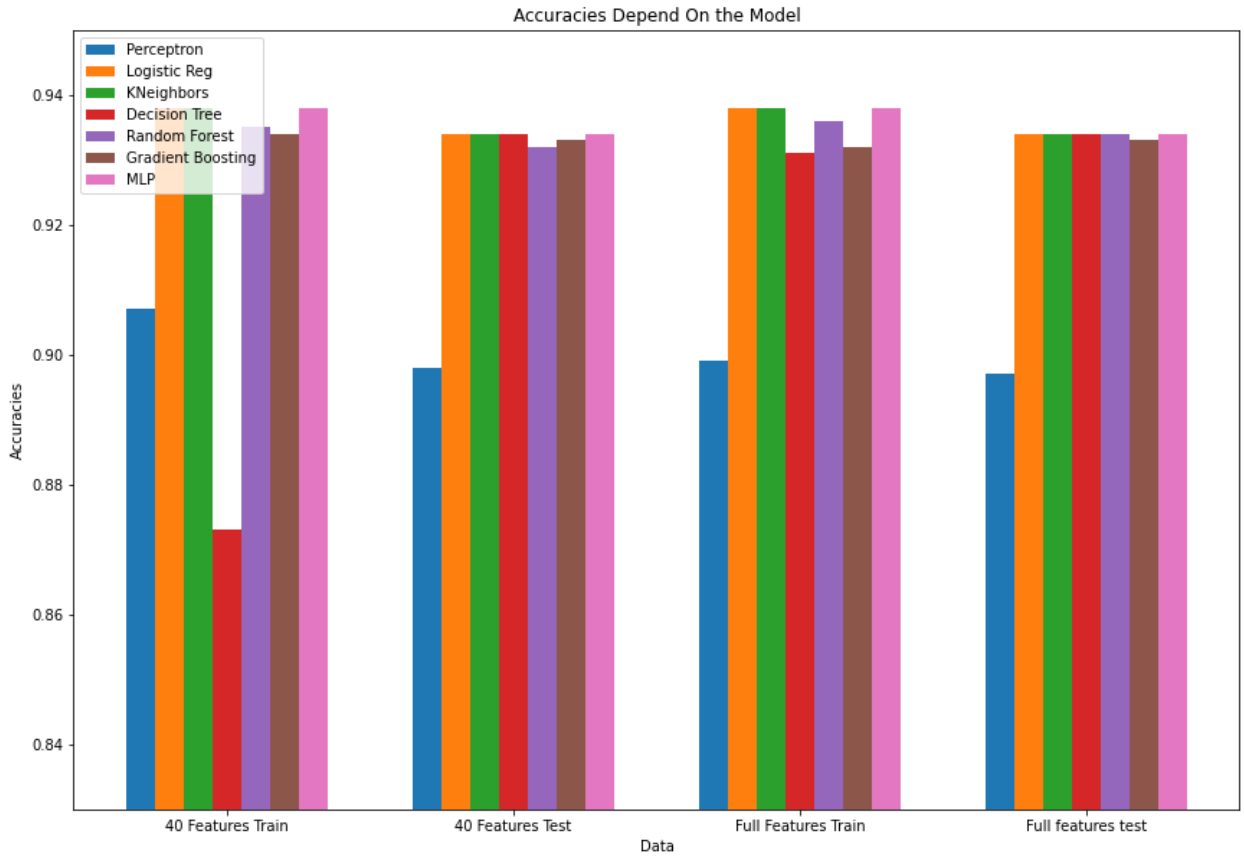


Figure 7: Train and Test Set Accuracies of Our Models

*Our *PCA-Reduced models overall performed much better than the full-featured models on the test dataset, likely due to the full-featured models being more likely to overfit. We took test set accuracy into consideration when evaluating the model and as such our best performing models are the models with the best test set accuracy, 93.4 % (listed above).*

**Note: PCA-Reduced is labeled as 40-features*

In addition to accuracy, our team also measured precision, balanced accuracy, recall, and F-1 score for each model. Our primary area of focus was a high recall (True Positive / True Positive + False Negative). We wanted our model to have a high recall since we would want to reduce false negatives in diagnoses that could potentially slow the patient treatment process. Eight different models tied for the highest recall - 93.4%, including all seven of the models with the highest accuracy (discussed above). Our advanced metrics are shown in Figure 8 below.

	Models	Balanced Accuracy	Precision	Recall	F-1 score
0	Perceptron	0.891	0.895	0.897	0.89
1	Logistic Regression	0.926	0.936	0.934	0.933
2	KNeighbors Classifier	0.926	0.936	0.934	0.933
3	Decision Tree Classifier	0.896	0.924	0.913	0.915
4	Random Forest Classifier	0.857	0.89	0.893	0.884
5	Gradient Boosting Classifier	0.924	0.935	0.933	0.932
6	Multilayer Perceptron Classifier	0.926	0.936	0.934	0.933

Models	Balanced Accuracy	Precision	Recall	F-1 score
Perceptron	0.838	0.852	0.86	0.841
Logistic Regression	0.926	0.936	0.934	0.933
KNeighbors Classifier	0.926	0.936	0.934	0.933
Decision Tree Classifier	0.925	0.935	0.934	0.933
Random Forest Classifier	0.923	0.934	0.932	0.931
Gradient Boosting Classifier	0.925	0.935	0.934	0.933
Multilayer Perceptron Classifier	0.926	0.936	0.934	0.933

Figure 8: Model Accuracy, Precision, Recall, and F-1 Score.

The left table lists the details of models trained with the Full-Features dataset. The right table lists the details of models trained with a PCA-Reduced dataset. The Logistic Regression, K-Nearest Neighbors, Multilayer Perceptron classifiers for both their PCA-Reduced and full-featured versions of their models all had the same advanced metric percentages.

While we evaluated our models on having high balanced accuracy, precision, and F-1 scores, we placed a greater emphasis on models with higher recall and test accuracy. We considered these to be the most important metrics, as the rate of true positives essentially showcases the predictive accuracy of the model, while the test accuracy of the models are exactly how well they predict diseases given real-life input that would resemble how the tool would be used in the real world. An issue we encountered was that the metrics of six of our best-performing models (Logistic Regression, K-Nearest Neighbors, Multilayer Perceptron for both PCA-reduced and full-featured) were the same. Our team believes this is due to the simplicity of the original dataset. Due to these equivalent performances, our model evaluation is based on its adaptability (*see Adaptability)*.

10.1 Final Model Choice

Given our results, our team's recommendation to support the predictive tool is the PCA-Reduced K-Nearest Neighbors (KNN) classifier. As discussed above, this model tied for the best test accuracy and recall of all our models. Since this KNN is PCA-reduced, there is less risk of this model overfitting compared to the full-featured version of it. Furthermore, KNN requires no training time, which makes it a highly adaptable model in any case where the training data is updated. The only downside to this model is its inability to scale well with large data. However, this does not appear to be an obstacle on this dataset as this model was able to converge within 22 seconds - the fastest out of all our models.

We chose KNN over the similar performing Logistic Regression due to the fact that Logistic Regression has difficulty differentiating between dependent and independent variables which could lead to inaccurate results. Our choice of KNN over Multilayer Perceptron was mainly due to KNN being less computationally expensive than the Multilayer Perceptron algorithm.
We understand that our model is not perfectly accurate, however, as discussed previously, this model is meant to be taken into account as a suggestion rather than as an answer for disease diagnosis.

11. Future Updates

As the model is increasingly utilized within the medical field and more labeled data becomes available, we want to ensure that the model can continue learning in order to become increasingly accurate and reliable. The incorporation of new data could also introduce new diseases and symptoms that the classifier would be capable of utilizing in its algorithm, further increasing its accuracy and expanding its catalog of classifications.

As stated in the *Results* section, KNN does not scale well with larger sets of data. While the current dataset is small enough for the impact to be negligible, as the total amount of accessible data increases, the firm may want to consider resorting to a Multilayer-Perceptron model. Multilayer-Perceptron models have proven to be capable of handling large amounts of data, and have the flexibility to handle large numbers of non-linear features. While this comes at the expense of computational complexity, and time, the model itself need only be trained once. After training a MLP to a new dataset, while it may take significantly longer, the tool could continue to reliably give accurate results based on the data fed into it regarding patient symptoms.

12. Conclusion

A healthcare startup in Silicon Valley, Health Insights, approached our firm, Maize & Blue Data Analytics, to build a predictive disease tool. Given a set of symptoms, this tool is to be able to provide an accurate diagnosis for a disease that a patient exhibiting those symptoms is facing.

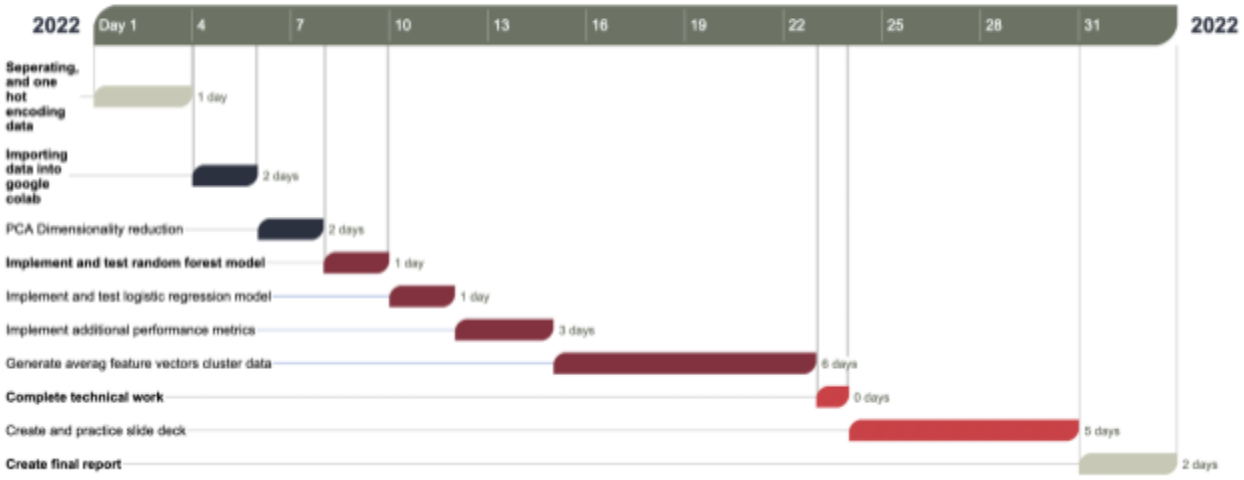
Given the dataset and information we received, our team, YES WAY, believes that our PCA-Reduced K-Nearest Neighbors is the best classifier for the tool requested by Health Insights. This model produced, and tied for, the best accuracy and recall of our models and is capable of taking in new data efficiently, given that the dataset itself is not too large. While the model does not scale well with large datasets, this is not a concern given the current dataset, and if needed, a multi-layer-perceptron may be utilized for prediction.

Our team followed the data science lifecycle closely - data cleaning and preparation, feature engineering, testing, and training - to build and create this K-Nearest Neighbors algorithm. In addition, our team followed this lifecycle to build a grouping of similar diseases in situations where the algorithm's classification is potentially inaccurate. Given our team's approach and findings, we believe that this PCA-Reduced K-Nearest neighbor is the best classifier to diagnose a disease given a list of symptoms.

13. Appendix

Appendix A: Project Timeline

Figure 3: Project Gantt Chart



Appendix B: Accuracy Statistics (with and without PCA)

Models	40 Features Test	Full Features Test
Perceptron	0.898	0.897
Logistic Reg	0.934	0.934
KNeighbors	0.934	0.934
Decision Tree	0.934	0.934
Random Forest	0.932	0.934
Gradient Boosting	0.933	0.933
MLP	0.934	0.934

Appendix C: Performance Investigation Results

Model Name	**Best Parameters**	**Number of Fits**	**Seconds per Fit (s)**
KNeighbors (PCA)	{'algorithm': 'auto', 'n_neighbors': 9, 'weights': 'uniform'}	480	0.04
KNeighbors	{'algorithm': 'auto', 'n_neighbors': 9, 'weights': 'uniform'}	480	0.13
Multilayer Perceptron (PCA)	{'activation': 'identity', 'alpha': 0.0001, 'hidden_layer_sizes': (300, 200), 'learning_rate': 'constant', 'solver': 'sgd'}	480	5.18
Multilayer Perceptron	{'activation': 'identity', 'alpha': 0.0001, 'hidden_layer_sizes': (300, 200), 'learning_rate': 'constant', 'solver': 'sgd'}	480	4.64
Logistic Regression (PCA)	{'C': 0.1, 'max_iter': 100, 'multi_class': 'auto', 'solver': 'newton-cg'}	480 (440 successful)	2.60
Logistic Regression	{'C': 0.1, 'max_iter': 100, 'multi_class': 'auto', 'solver': 'newton-cg'}	480 (440 successful)	4.17
Gradient Boosting Classifier (PCA)	'criterion': 'absolute_error'}	10	744.64
Gradient Boosting Classifier	'criterion': 'absolute_error'}	10	958.16
Random Forest (PCA)	{'criterion': 'entropy', 'max_depth': 5, 'n_estimators': 150}	480	0.21
Random Forest	{'criterion': 'entropy', 'max_depth': 12, 'n_estimators': 50}	480	0.15
Decision Tree (PCA)	{'criterion': 'entropy', 'max_depth': 9}	480	0.02
Decision Tree	{'criterion': 'entropy', 'max_depth': 24}	480	0.04
Perceptron (PCA)	{'alpha': 0.0005, 'max_iter': 100, 'penalty': 'elasticnet'}	270	0.04
Perceptron	{'alpha': 0.0001, 'max_iter': 100, 'penalty': 'l1'}	270	0.33
Note: Model training time can varied based on devices			

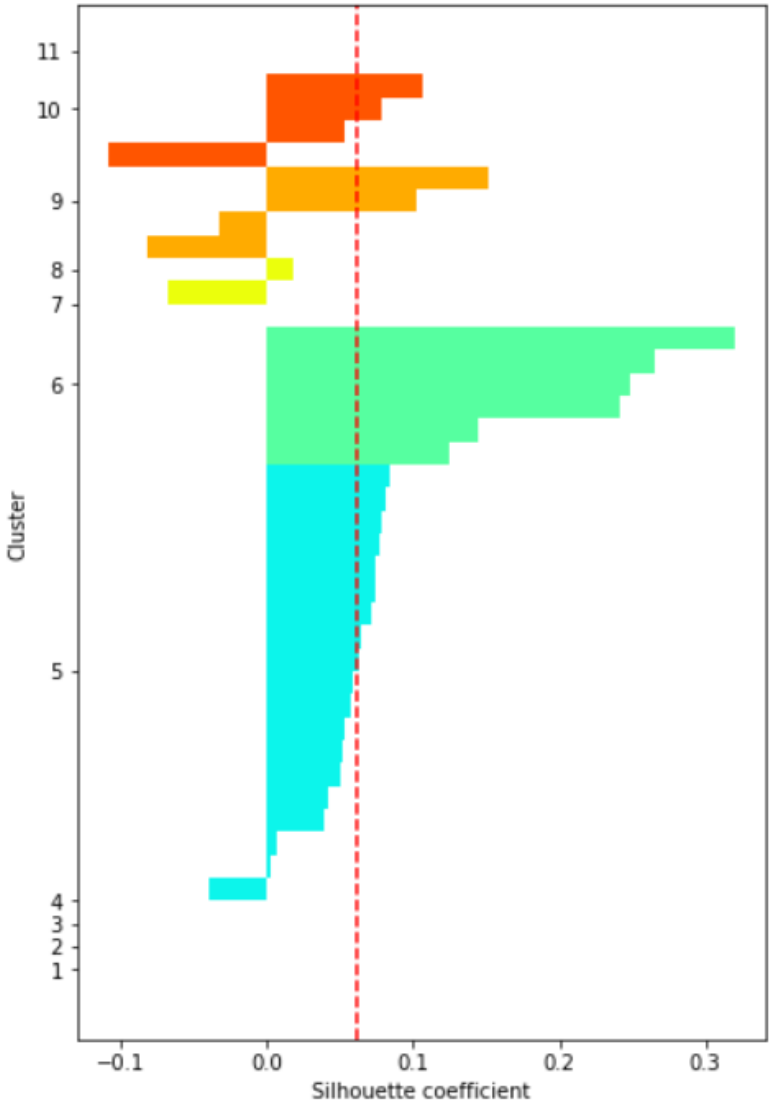
Appendix D: Clusters Details

```
Cluster  1 :  Hyperthyroidism
Cluster  2 :  Dimorphic hemmorhoids(piles)
Cluster  3 :  Chicken pox
Cluster  4 :  Hypoglycemia

Cluster 5: AIDS, Acne, Alcoholic hepatitis, Allergy, Arthritis, Cervical spondylosis, Drug Reaction, Fungal infection, GERD,
           Gastroenteritis, Heart attack, Hypertension, Impetigo, Migraine, Osteoarthristis, Paralysis (brain hemorrhage),
           Peptic ulcer diseae, Psoriasis, Urinary tract infection

Cluster  6 :  Chronic cholestasis, Hepatitis C, Hepatitis D, Hepatitis E, Jaundice, hepatitis A
Cluster  7 :  Hepatitis B
Cluster  8 :  Diabetes, Varicose veins
Cluster  9 :  (vertigo) Paroymsal  Positional Vertigo, Dengue, Malaria, Typhoid
Cluster  10 :  Bronchial Asthma, Common Cold, Pneumonia, Tuberculosis
Cluster  11 :  Hypothyroidism
```

Appendix E: Silhouette Coefficient for Each Cluster



Appendix F: Most Common Symptoms

	Symptoms	Frequency
121	vomiting	1970
41	fatigue	1679
45	high_fever	1248
60	loss_of_appetite	1201
71	nausea	1121
55	itching	1046
0	abdominal_pain	1039
44	headache	1025
100	skin_rash	897
129	yellowing_of_eyes	887

Appendix G: Technical Work
https://colab.research.google.com/drive/1tXC62y29PbTkSnwIgECqk6gEZf4wGb06?usp=sharing

Appendix H: Hyperparameter Tuning
https://colab.research.google.com/drive/18IeRT4Hogv-xzr-zEqeMjXhtjJBByQbE?usp=sharing

14. Works Cited

[1] Deepthi, Y., et al. “Disease Prediction Based on Symptoms Using MachineLearning, in *Lecture Notes in Electrical Engineering*, vol 664. Singapore: Springer, 2020, pp. 561-569.https://link.springer.com/chapter/10.1007/978-981-15-5089-8_55.

[2] Park, D.J., Park, M.W., Lee, H. *et al.* Development of machine learning model for diagnostic disease prediction based on laboratory tests. *Sci Rep* 11, 7567 (2021). https://doi.org/10.1038/s41598-021-87171-5

[3] Richens, J.G., Lee, C.M. & Johri, S. Improving the accuracy of medical diagnosis with causal machine learning. *Nat Commun* 11, 3923 (2020). https://doi.org/10.1038/s41467-020-17419-7

[4] P. Umathe, “Data Science Life Cycle,” *Medium*, 30-Jun-2020. [Online]. Available: https://medium.com/@pumathe/data-science-life-cycle-4b0b6c4dfeef. [Accessed: 19-Apr-2022].

[5] Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, Dinani Amorim; 15(Oct):3133−3181, 2014.