INDEX

Contents

List of figures

Figure 1 depicts a flow chart which guides through the steps taken for completion of the project.

Figure 2 depicts the number of cases collected from the countries for which patient data was present in a map chart format.

Figure 3 depicts the spread of the cases throughout the years 2006-2020 in a pie chart format.

Figure 4 depicts the male to female ratio of the total number of cases collected in a pie chart format.

Figure 7 depicts the ROC score of each algorithm in a bar chart format.

Figure 9 depicts the metrical figures of each technique used to find out results in a bar chart format.

Figure 10 is the AUC-ROC curve of each technique used presented in a comparative and comprehensible manner

List of Tables

Table 1: contains the arithmetic values assigned to each individual condition in the “chronic conditions” column of the data set. These values will be the ones used during compilation of pre-processed data.

Table 2: contains the binary values (0,1) assigned to each instance of “symptoms” attribute to be later comprehended by the model during the training and testing process.

Abstract (or executive summary)

Influenza is a disease with a high level of contagiousness that impacts millions of individuals globally annually. The devastating effects of influenza outbreaks are well-documented, with the Spanish Flu being one of the most significant outbreaks in history. To combat the spread of influenza, the use of Machine Learning techniques in medical research has gained popularity, with applications in early disease diagnosis, pathology, and disease classification.

In this study, we extracted a dataset of Human Surveillance records from the Influenza Research database. Initially, we extracted 30531 records covering the period of 2006 to 2020 for data pre-processing. After thorough pre-processing, we selected 18581 records for further analysis.

The next step involved the classification and analysis of the data using four different machine learning methods: Support Vector Machines, K-nearest Neighbors, Artificial Neural Networks, and Random Forest. The performance of these methods was evaluated in terms of sensitivity, specificity, and accuracy.

The experimental results demonstrated that Random Forest was one of the most effective machine learning techniques for the early detection of influenza. This finding has significant implications for the medical field, as it can aid in the development of more accurate and efficient influenza detection systems.

Overall, the study highlights the potential of Machine Learning techniques in medical research and their effectiveness in the early detection of influenza. By analysing large datasets and employing sophisticated algorithms, we can make significant strides in the prevention and control of infectious diseases like influenza.

Objective

Through this project, we wish to further enhance upon the use of Machine learning techniques and algorithms in the field of medicine, exhibiting the accuracy and ease of access of such amenities and the utility of these techniques in this day and age.

With the human-constrained data procured through inter-connected resources and the availability of past research done by various publications, we have combined and amplified the methodology of our research, providing efficient and much more precise results.

Considering the type of data provided and the nature of the research, utmost care has been taken to not let outliers and unnecessary outputs deviate the desired results, inflicting upon the learning methods itself and creating significant problems for the research.

Taking into consideration that much of the work has been done through Python language and past resources provided in this paper, the project is easy to understand and implementable for further research upon this subject

Chapter 1: Introduction

Influenza is an acute respiratory disease caused by the influenza A, B or C virus. Although type D virus does exist, it mainly affects cattle with frequent spill over to other species. It often emerges as outbreaks and epidemics worldwide, mainly during the winter season. Respiratory secretions of infected individuals contain a substantial amount of influenza virus particles, thus making it possible for the infection to spread through the dispersion of large droplets generated during sneezing and coughing. “The mean duration of influenza virus shedding in immunocompetent adult patients is ~5 days but may continue for up to 10 days or more, particularly in children, elderly adults, patients with chronic illnesses, and immunocompromised hosts” [1]. The onset of influenza is usually sudden and characterized by the appearance of high-grade fever, along with symptoms such as myalgia, headache, and malaise. These rapid manifestations are accompanied by symptoms of respiratory tract illnesses such as non-productive cough, sore throat, and nasal discharge. Following a certain duration of illness, influenza has the potential to affect various organs including the lungs, brain, and heart to a greater extent than the respiratory tract, potentially leading to hospitalization.

The greatest number of deaths inflicted on the human population by the virus is in the year 2016, when 3985 lives were lost, comprising almost equal Male (1987) and Female (1998) deaths.

The virus spreads as yearly outbreaks and in the past has affected at least 2 million people every outbreak and caused >200,000 deaths [2]. Notable records of the virus exist such as “Spanish flu”, “Asian Influenza”, and” Hong-Kong Influenza”.

Machine learning techniques in the field of healthcare have reached the point where they are being used to analyse pictorial data in which circumstance, they use pattern recognition to look for certain indications. Within this project, techniques used for analysis are mainly implemented in the form of classifying algorithms.

Many kinds of datasets were available for such a research subject, some of which showed weekly updates to cases and symptoms. For this project, the dataset used was chosen to fine tune the model to as much accuracy as possible and remain within the boundaries of such researches for synergetic and ethical purposes.

Machine learning researches have reached to a point where such impactful diseases can be detected early on, playing a significant role in efficient classification and medication. This project uses various methods such as Machine Learning techniques along with Deep Learning networks on a dataset through which the model is trained and validated. By providing a sufficient accuracy rate, this project is helping to further integrate Machine Learning models in medication systems and improvising them. The dataset has been procured from a trustable source [13], which among many other great sources out there, provides a well-defined dataset with all the necessary information required for the model along with a good sample size, helping us to further give precision to our model.

.

Chapter 2: Literature review

1. The article [1], provides information about the type and spread of strains of Influenza virus, along with the symptoms usually contracted. The article cites prevention of influenza using vaccines and early diagnosis, while reviewing cases involving pregnant women.
2. In the report [2], are used in order to detect Influenza, another great report which enhances upon the use of such methods in modern healthcare.
3. In the article [3], the estimation has been done through two methods: direct estimation method which involves study and calculation of cause-specific deaths, while the second one is a more indirect way of approximation, and enumerates all-cause mortality. They highlight that a predefined death rate was needed to identify differences arising from calculation of excess mortality.
4. In the article [4], insightful reports and useful information surrounding epidemic control for Influenza has been given.
5. In the article [5], connections of influenza virus spread with avian influenza virus have been made, citing the pandemics of 1957 and 1968. A review of the ecology and evolution of highly pathogenic avian influenza H5N1 viruses, assess the pandemic risk, and address aspects of human H5N1 disease in relation to its epidemiology, clinical presentation, pathogenesis, diagnosis, and management has been done.
6. In the article [6], information about the 2009 influenza spread has been given. This review provides an update encompassing the virology, epidemiology, clinical manifestations, diagnosis, treatment, and prevention of the 2009 H1N1 virus.
7. In the report [7], they observe the implications of the vaccines administered by the United States in lieu of the influenza season which coincides with the spread of SARS-COV2 virus, also known as Covid-19. They mention the arrangement and protocols needed to be followed by the authorities for controlled handout of the vaccines.
8. In the article [8], they have created an intricate and information-comprehensive database for the 2009 pandemic influenza A/H1N1 describing the genome-sequence data and characteristics of the virus that are potentially much use for research.
9. In this article [9], use of medicinal information in accordance with its use in artificial intelligence objectives is described. Manual processing of data needs to be done in order for further analysis and for mining model-comprehensive data.

10. In the article [10], ensemble techniques have been used for classifying cancer given gene expression data. Genome RNA expression studies permit systematic approaches to understanding the correlation between gene expression profiles to disease states or different developmental stages of a cell, while the use of Microarray analysis provides quantitative information about the complete transcription profile of cells that facilitate drug and therapeutics development, disease diagnosis, and understanding in the basic cell biology, has been stated.
11. this article [11], objectivises analysing of telemedical systems for providing characteristic and personal care for prevention of Cardio-Vascular Diseases. With the methodology used by the authors, they found out that results occur in positive favour when timely delivery of protection and prevention is done and is a major benefit to hospital workload especially.
12. This article [12], catches the need of predictive analysis to diagnose heart diseases with learning techniques. This work is focused on developing various machine learning predictive models using support vector machine, decision tree, neural network and K-nearest neighbour for prediction of heart disease.
13. In the article [13], the need for good pre-processing of patient data collected for various purposes has been emphasised, citing that with such databases having large amounts of data, machine learning techniques could be used for automatic processing.
14. From here [14], we accumulate the data, which we used in our research and perform various technique. Its was collected from all over the world for maximum species but include only homo sapiens.
15. Here [15], This review stresses on easy and reliable interpretation of the ROC curve and following certain guidelines for writing and presenting such analysis. ROC curve helps us to understand “True Positive Rate” and “False Positive Rate”.

    TPR = TP / (TP + FN)

    FPR = FP / (FP + TN)
16. In article [16], it elaborates on choosing the best techniques and models to comprehend certain data for testing and analysis given a certain objective, by which we are able to find a certain technique which perform better on the data accumulated from [13].
17. [17], This book comprises understanding the various techniques used in data mining and statistical analysis for processing of large databases for set objectives, which helps us to understand them and implement on our research work.

18. In article [18], it provides information related to Random Forests classifier through which we get know about it's working, which make its implementation easy for us.
19. The book [19], provides ways to use the S environment for applied statistics given how powerful a tool it is for such purposes. By which we came to know each variable of data set are related with each other.
20. The book mentioned in [20] provides resources and information regarding pattern recognition and neural networks. How perceptron is working and how to make neural network of different layer and we got know about ANN.
21. The paper [21] demonstrates the efficient parallelisation of profiling and recommendation algorithms using tourism crowdsourced data repositories and streams, focusing on entity-based and feature-based profiling approaches using ratings, comments, and location.
22. [22], This paper studies several machine learning classifiers for influenza detection, comparing their diagnostic capabilities against an expert-built influenza Bayesian classifier, and evaluating different ways of handling missing clinical information from the free-text reports.
23. The research [23], focuses on creating an intelligent system for diagnosis of influenza using the relevant factors based on historical data of the Mexican population. Which consists of their medical condition and symptoms shown by patients.
24. The work [24], objectivises examination of signs and symptoms which are most relevant for prediction of influenza infection using a large data set derived from clinical trials of zanamivir.

Chapter 3: Research methodology

The methodology followed by the study is outlined in Figure 1, which depicts the various stages involved in the project. The data used for this study was obtained from the Influenza Research Database (IRD) [13], which provided us with a comprehensive dataset for analysis.

The initial stage of the project involved pre-processing the data to ensure its accuracy and suitability for analysis. Once the data was pre-processed, we identified the symptoms and other clinical data as the main attributes for analysis.

To carry out predictive analysis on the pre-processed data, we implemented four different machine learning techniques. The purpose of this approach was to compare and evaluate the performance of the various machine learning models used and identify the most effective one.

Overall, the methodology followed in this study was designed to ensure a rigorous and comprehensive analysis of the data. By utilizing multiple machine learning techniques, we aimed to enhance the accuracy and reliability of our predictive analysis.

3.1 STUDY AREA

For the objective of this research, the data chosen comprises the symptoms the patient exhibits along with the gender, age and collection year of the patient. The data pre-processing done is to escalate and expand upon such features. Furthermore, the techniques used in this regard such as encoding will help the model to understand the data better. The data is obtained from the databases of BV-BRC, and contains data of 30,531 patients identified over the years and countries.

In previous researches done by various authors, the use of singular algorithms was appropriated to carry-out the research and calculate the results and conclusions. In this research paper, we have emphasised on the use of combined learning algorithms instead of using the learning models one at a time in order to obtain more accurate results.

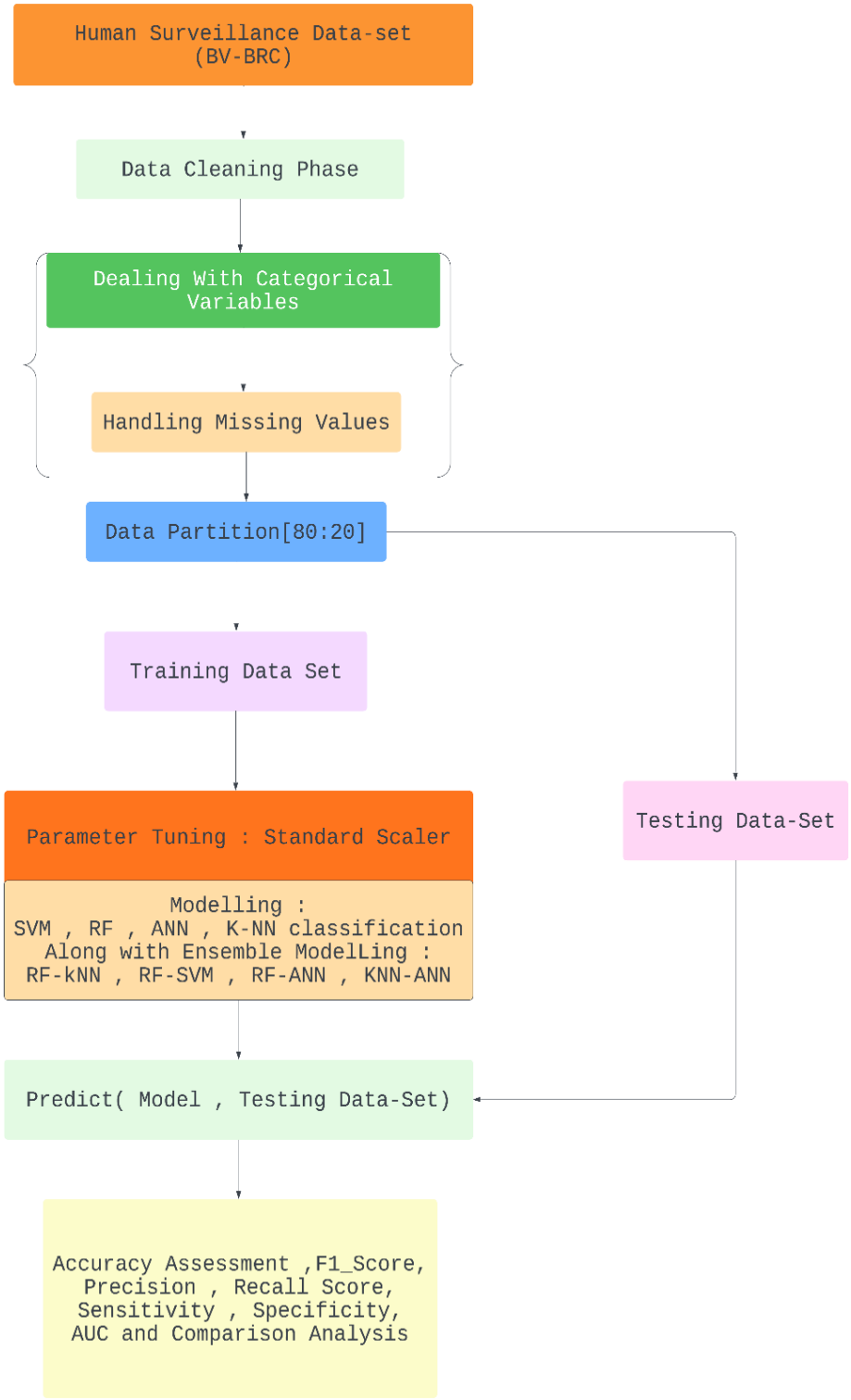


Figure 1: Flow Chart OF Methods

We have shown the shortcomings of such previously-used methods and models of KNN, ANN, Random Forest and SVM taken one at a time initially in our paper. As we progress, the models were combined in regards to their better metrical scores.

Within the data used in this paper, the patient's information such as collection year, age of patient and symptoms shown by the patient were studied and passed into the models which were then evaluated and concluded upon. The dataset also contains the type of pathogen the patient is infected with, the demographic information of the patient, the strain the patient is infected with, and many other ineffective information.

3.2 Data used

Within the data used in this paper, the patient's information such as collection year, age of patient and symptoms shown by the patient were studied and passed into the models which were then evaluated and concluded upon.

The dataset also contains the type of pathogen the patient is infected with, the demographic information of the patient, the strain the patient is infected with, and many other ineffective information.

The dataset containing appropriate structure and variables was procured from Bacterial and Viral Bioinformatics Resource Centre. For this project, all obtained data remains constrained to humans only, in consideration of feasible and ethical reasons.

Some attributes such as patient's name, email-id and similar were removed in the beginning since they contain irrelevant information to the purpose of the project and the models used

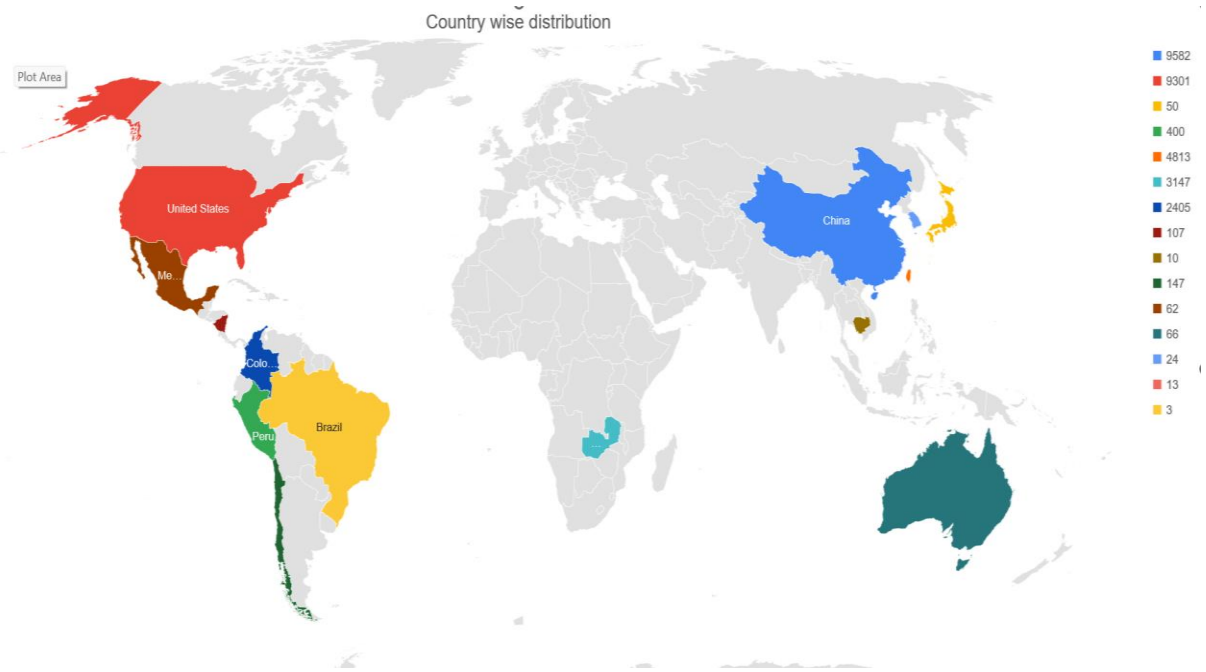


Figure 2: Country wise distribution of Records in Dataset

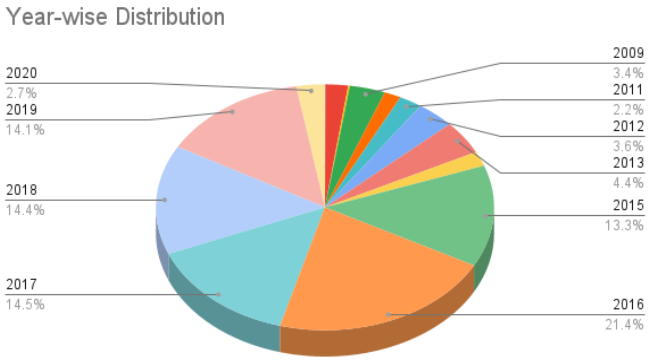


Figure 3: (Left) Year-Wise Distribution

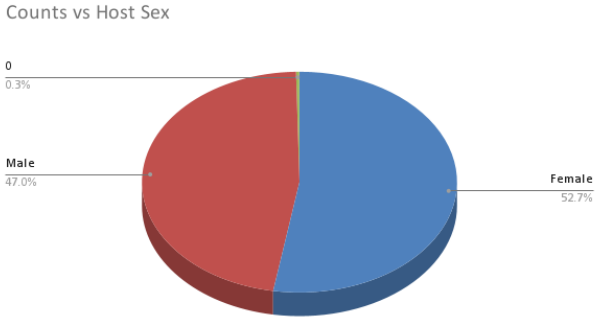


Figure 4:(Right) Gender-Wise Distribution Of 'Positive' Class

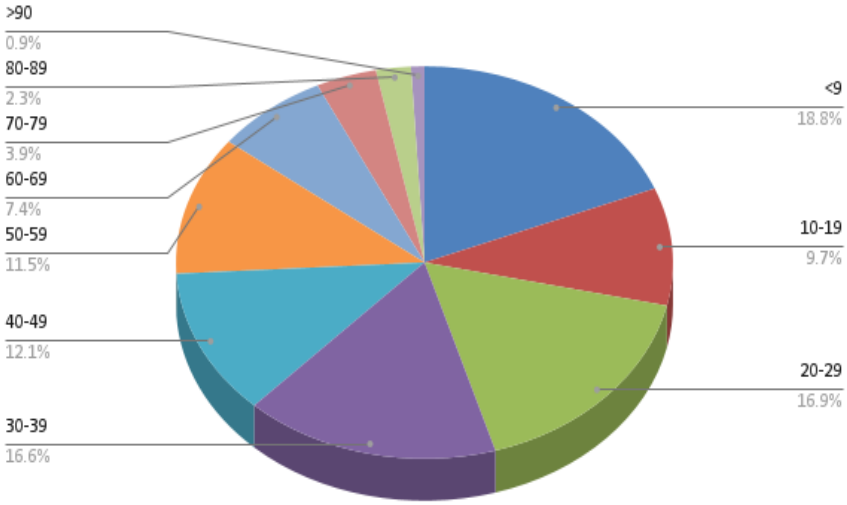


Figure 5: Age Distribution Chart

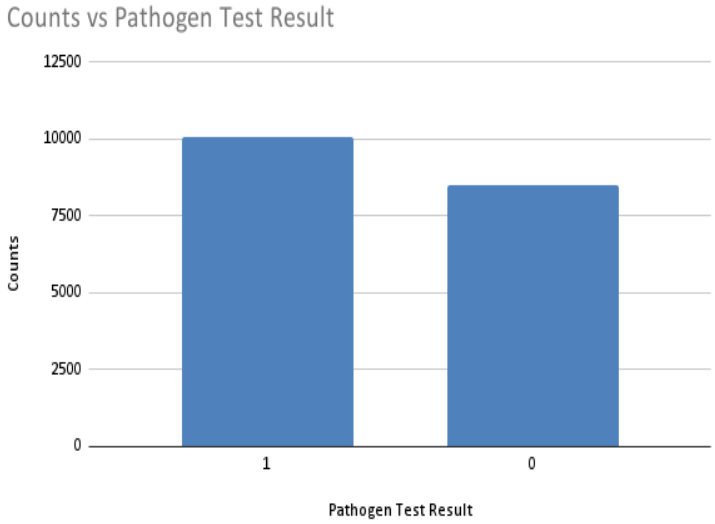


Figure 6: Target Variable Ratio

3.3 Data Pre-processing

Initially, the data consisted of 30,531 rows and 98 columns, most of the columns having little to no instances of data. Various data mining techniques were used to pre-process and transform the data so that the learning models are able to comprehend the data without giving out too many bugs and errors. The data was cleaned of necessary outliers and impurities, and later recompiled with changed positions of certain data points or rows or columns and different, easier-to-understand labels. All blank columns were firstly taken out.

Next, the "symptoms" column contained data points in the form of key-value pairs, which were then separated into rows and columns, respectively. Furthermore, all the Yes/No/Blank data points were converted into binary data, with Blank and No being considered as 0, and yes being considered as 1. The data points in the gender column were converted to binary data, with Males getting initialised as 0 and Females getting initialised as 1. Then, only rows having all instances filled were taken into consideration, the rest were removed. Any row or column value having its instance as null was removed and disregarded. Similarly, all columns having more than 95% data as null values were removed.

A certain number was assigned to the string values in the "chronic condition" column in consideration of how many conditions were in many instances of the data were combined together in one large string. These numbers assigned were then aggregated so as to compile some conditions and form a model-understandable arithmetic value. After having finalised the data pre-processing, 18582 rows and 23 columns remain, which only contain a patient's fully-procured relevant data with the collection year, symptoms, gender and age, all columns necessary for processing.

At the start of our analysis, we observed that there were 697 distinct combinations of medical diseases present in the data. However, we needed to identify the most relevant factors to ensur e our analysis was accurate and meaningful. Thus, we identified the four most commonly repe ating factors in the dataset. Based on these four factors, we defined a power set that divided th e factors into 30 unique combinations.

Chronic Conditions Combined	Values Used
None	0
HPT	1
RSP	2
Chronic Lung Disease	3
END	4
CDV	5
ASM	6
HEM	7
NRL	8
OBS	9
IMS	10
CAN	11
URO	12
FLU	13
Immuno	14
Diabetes	15
Obesity	16
END; CAN	17
HEM; CAN	18
HPT; END	19
CDV; HPT	20
END; NRL	21
CDV; NRL	22
URO; END	23
CDV; URO	24
RSP; ASM	25
OBS; OTH-Chronic Lung Disease; ASM	26
CDV; END	27
CDV; END; NRL	28
OTH; Chronic Lung Disease; ASM	29
OTHER	30

Table 1: Medical Condition

Symptoms Indicated	Values used
Fever	0,1
Chills	0,1
Conjunctivitis	0,1
Cough	0,1
Diarrhoea	0,1
Headache	0,1
Loss Of Appetite	0,1
Malaise	0,1
Myalgia	0,1
Nausea	0,1
Running Nose	0,1
Short Breath	0,1
Throat	0,1
Vomiting	0,1
Wheezing	0,1
Other Symptoms	0,1

Table 2: Symptom Distribution

To better understand the medical conditions, present in the data, we used Table 2 to define the value substitution in the dataset. This allowed us to transform the medical conditions into a

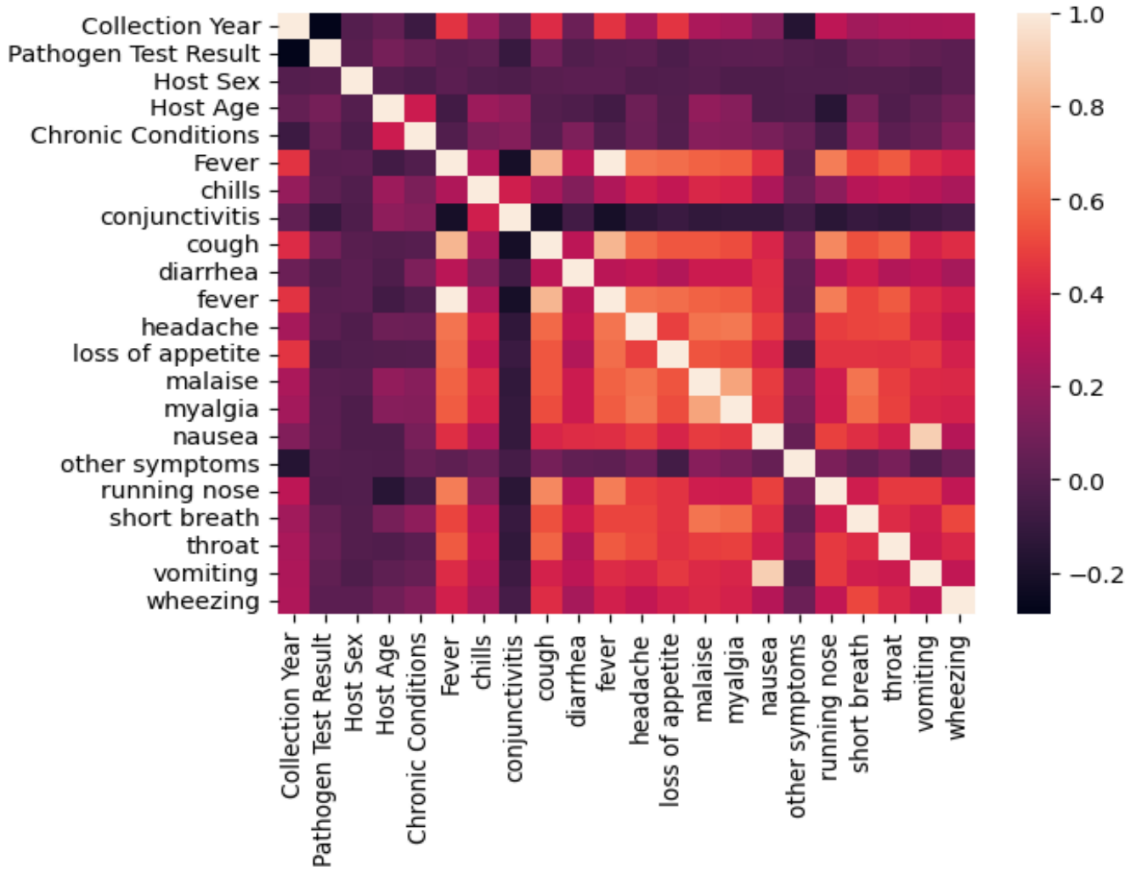


Figure I: Heat map of Correlation between each Variable in after cleaning

more manageable form that could be used for further analysis. By identifying the most relevant factors and defining a power set, we were able to streamline the data and ensure our analysis was focused on the most important medical conditions related to early detection of influenza.

3.4 Training and testing of data

The data was divided into sets of 70% or 80% training and rest for testing, to be comprehended by the learning models. The data was divided using the train_test_split command available in Python through the use of the Scikit-Learn package. Hence, the models were trained with either 13,007 training samples or 14,865 training samples. Training sample divided as 'Positive' (54.31%) and 'Negative' (45.69%).

Formula defined for our analysis:

Pathogen Test Result ~ Gender + (all the 25 Symptoms) * + Medical conditions

**For the Symptoms attributes refer Table 2*

3.5 Classification Algorithm

3.5.1 SVM

Support vector machines (SVMs) are a category of supervised learning algorithms utilized for classifying, regressing, and identifying outliers in data. To address different machine learning tasks, there exist specialized types of SVMs such as support vector regression (SVR), which is an expansion of the support vector classification (SVC) technique. SVMs are different from other classification algorithms because of the way they choose the decision boundary that maximises the distance from the nearest data points of all the features. The superiority of the linear SVM algorithm over other algorithms, such as k-nearest neighbours, stems from its ability to identify the optimal line to classify data points. The algorithm selects the line that maximizes the distance between the line and the closest data points, allowing it to capture intricate relationships within the data without necessitating extensive pre-processing or transformation.

Support Vector Machines (SVM) is a machine learning algorithm that involves tuning several parameters, depending on the type of kernel used. These parameters include the regularization parameter (C), kernel type, degree, and gamma. For our study, we focused on the linear kernel at the initial level. Therefore, we did not require the values of gamma and degree since these parameters are used for higher-level kernels.

To set the kernel to linear, we used the method "svm Linear" in the train () method. The default value of C for classification was set to 1. This allowed us to tune the SVM algorithm to suit our specific needs in predicting early detection of influenza using the BV-BRC dataset. By focusing on the linear kernel and tuning the regularization parameter, we were able to optimize the performance of the SVM algorithm for our dataset.

- The data was first split into two sets of testing and training data, with 30% of the data taken up for testing. After setting up the standard scaler, parts of training and testing data were passed into it. These transformed parts, along with the other parts of the training set were used to train the model, which was then fitted upon the test set of the data

3.5.2 k-Nearest Neighbours

The k-nearest neighbor's algorithm, or KNN, is a supervised learning classification technique that utilizes proximity to make predictions or classifications about individual data points. As a non-parametric algorithm, it makes no assumptions about the underlying distribution of data and uses the closest training examples to determine the classification of a new data point. It is a classification algorithm, though it can be used for regression problems as well. For a new data point to an existing dataset or a data point within the dataset, the algorithm classifies the data point by finding the most similar training examples in the feature space. It is a lazy-learning model, meaning that it only stores a training dataset rather than undergoing a training stage. Since the k-nearest Neighbors algorithm, or KNN, is a lazy-learning model, all computation takes place during the classification or prediction process. This is because KNN only stores the training dataset in memory and does not perform any training stage. As a result, when a new data point is classified, KNN identifies the closest training examples and calculates the prediction based on their proximity to the new data point. The k-nearest neighbor's algorithm, or KNN, is commonly referred to as an instance-based or memory-based learning method due to its reliance on storing the entire training dataset in memory. As a result, it uses proximity to determine predictions and classifications based on similar training examples.

The k-NN algorithm for classification is based on choosing a value for k, which specifies the number of nearest Neighbors to be included in the decision process. To determine the optimal value of k, a cross-validation approach is employed. Different values of k are tried with various randomly selected training sets, and the value of k that minimizes the classification or estimation error is chosen. By default, the value of k in the k-nearest Neighbors algorithm, or KNN, is set to the square root of the total number of samples (N) in the dataset. This value is chosen to balance the bias-variance trade-off and improve the accuracy of the KNN model. However, the optimal value of k can vary depending on the specific problem and dataset, and it is often determined using a cross-validation approach. In this study, the createDataPartition () method of the 'Caret' package in R is utilized to create partitioning. This method uses bootstrapped sampling to select values for k equal to 5, 7, and 9.

- The data was split into training and testing data, with 20% of the data being used for testing. The model was trained on the training set, the value of k being taken as 5. The data was split into training and testing data, with 20% of the data being used for testing. The model was trained on the training set, the value of k being taken as 5.

3.5.3 ANN

ANN is a type of machine learning algorithm that is modelled after the structure and function of the human brain. It is used for classification tasks. The basic building block of an ANN is a neuron, which takes input from other neurons or external sources, performs some mathematical operations on the inputs, and generates an output. Neurons are connected to each other through weighted connections, which are adjusted during training to improve the network's performance. ANNs are trained using a process called backpropagation, where the error between the network's output and the desired output is propagated backwards through the network, and the weights are adjusted to reduce the error. This process is repeated iteratively until the network's performance reaches a satisfactory level, although they can be computationally expensive to train and require a large amount of data to achieve good performance.

Neural networks are composed of units that are organized into layers, including the input layer, hidden layers, and output layer. Each of these layers is interconnected with a particular weight (w) that multiplies the signal as it travels along. Additionally, each unit sums its input, adds a bias (constant), and applies a function φ to the input (x) to return an output (y). This process of applying a function to the sum of inputs is known as activation, and different activation functions can be used depending on the requirements of the neural network model.

- The data was split into training and testing data, with 20% of the data being used for testing. A Dense neural network was set up with 24 nodes having rectified linear unit activation and 1 having sigmoid activation, in order to get output in the form of 0s and 1s. For compiling the model, ‘Adam’ optimizer was used which has a learning rate of 0.001. Then, the training set divided into batches of 32 and epochs taken as 100 was used to train the model.

3.5.4 RANDOM FOREST

A random forest is a powerful machine learning technique used for solving classification and regression problems. It employs ensemble learning, which involves combining multiple decision trees to provide a robust and accurate solution to complex problems. The random forest algorithm predicts the outcome by aggregating the predictions of many decision trees. The average or mean output of the individual trees is used to establish the final prediction. Increasing the number of trees in the forest enhances the precision of the outcome.

A decision tree is a basic component of the random forest algorithm and has three main parts: decision nodes, leaf nodes, and a root node. The algorithm divides the training data set into branches, which are further divided into additional branches until a leaf node is reached. The leaf node represents the end of the decision-making process and cannot be further divided. The decision nodes in the tree represent attributes or features used for predicting the outcome. These nodes serve as links to the leaves, where the final prediction is made. The data was split into training and testing data, with 20% of the data being used for testing. Then, the classifier object was set up taking 100 trees into consideration. Then, this model was trained on the training set.

Further, to obtain better results and expand on our research, two techniques were combined and taken at a time.

- The ‘Pathogen Test Result' values of the two algorithms were taken along with their metrics such as F1 Score, precision and recall score. These attributes were combined with the use of AND logic, that is, if either of the algorithms gave 0 as output, the final result was taken as 0. Only when both algorithms gave 1 as output, the final result was considered as 1. This logic is applied in order to further verify the accuracy of both of the algorithms and bring out a much more significant output.

3.6 Accuracy Assessment and Comparisons

The evaluation parameters used in this study were based on the work of Zhu et al. [30]. These parameters were selected to ensure that the performance of the different machine learning models could be compared fairly and accurately. The specific evaluation parameters used included sensitivity, specificity, and accuracy. Sensitivity is the proportion of true positives among all actual positive cases. Specificity is the proportion of true negatives among all actual negative cases. Accuracy is a statistical metric that quantifies the performance of a model in

correctly predicting the target variable. It is commonly used as a evaluation metric in classification tasks. However, accuracy may not be the best measure of performance for imbalanced datasets, where one class may dominate the other. In such cases, other metrics such as precision, recall, and F1-score may provide a more comprehensive evaluation of the model's performance. These parameters were calculated for each of the four machine learning techniques used in the study:

"Support Vector Machines, K-Nearest Neighbors, Artificial Neural Networks, and Random Forest". The evaluation results were then used to compare the performance of the different techniques and identify the most effective method for early detection of influenza.

TRUE POSITIVE (TP)
These are the values in the output dataset that are predicted to be correct by the learning algorithm while being actually correct.

TRUE NEGATIVE (TN)
These are the values in the output dataset that are predicted to be incorrect by the learning algorithm while being actually correct.

FALSE POSITIVE(FP)
These are the values in the output dataset that are predicted to be correct by the learning algorithm while being actually incorrect.

FALSE NEGATIVE (FN)
These are the values in the output dataset that are predicted to be incorrect by the learning algorithm while being actually incorrect.

ACCURACY

It is the measure of how close the output results are to the desired results, i.e., the number of correct predictions made by the algorithm. It is calculated by dividing the total number of true results by the total number of all results.

Accuracy = (TP+TN)/(TP+FP+TN+FN)

SENSITIVITY

Sensitivity, is the ability of the learning algorithm which takes into account its proportion to correctly identify “True Positives” in the test dataset. It considers the number of true positives taken against true positive and false negative results obtained from the output.

Sensitivity = TP/(TP+FN)

SPECIFICITY

Specificity, in line with sensitivity, is the ability of the learning algorithm which takes into account its proportion to correctly identify “True Negatives” in the test dataset. It considers the number of true negatives taken against all negative results obtained from the output.

Specificity = TN/(TN+FP)

F1 SCORE

Defined as the harmonic mean of the model’s precision and sensitivity, it is used to find the accuracy of the learning algorithm.

F1_Score = TP/[TP+(½)(FP+FN)]

AUC-ROC CURVE

A metric used to depict the performance of the model graphically at different threshold values. Initially, ROC, or Record Operating Characteristic curve takes into account the rates of true positive and false positive values. AUC-ROC, or Area Under ROC Curve, then considers the area present under said curve. This value ranges from 0 to 1, taking the area present in between (0,0) and (1,1) in the curve.

PRECISION

It is the measure of how close the output results are to each other, i.e., the quality of predictions made by the algorithm. It is calculated by dividing the number of true positives by the total number of positives (true positive and false positive taken together).

Precision = TP/(TP+FP)

Chapter 4: Research findings / Results

The analysis involved applying different learning techniques on the dataset and the results were evaluated using metrics such as Recall, F1 score and Confusion Matrix. This helped in presenting the data in a meaningful and organized manner. Based on the dataset and the learning techniques used, the results of the project are as follows:

1. Applying the Linear Support Vector Machine algorithm, the True Positive Rate turns out to be 63.70, while Specificity turns out to be 78.95. The F1 score for training and test data is 70.06 and 70.03, respectively. The accuracy for this algorithm is 70.66. On its own, it is a pretty low performing algorithm when compared to subsequently applied techniques.

2. Applying the K-Nearest Neighbour algorithm, the True Positive Rate turns out to be 76.42, while Specificity turns out to be 79.90. The F1 score is 78.00, so is its accuracy. It is a well performing algorithm.

3. Applying the Associated Neural Network algorithm, the True Positive Rate turns out to be 52.50, while the Specificity turns out to be 70.80. The F1 score is 59.37. The accuracy for this algorithm is 60.83. Of all the techniques used, this is the lowest performing algorithm.

4. Applying the Random Forest algorithm, the True Positive Rate turns out to be 84.09, while Specificity turns out to be 90.95. The F1 score for training and test data is 92.91 and 87.78, respectively. The accuracy for this algorithm is 87.20. This is the highest performing algorithm of all algorithms used.

In order to obtain much higher accuracy and more significant results, these techniques were combined with each other, providing these results:

1. Applying SVM and KNN algorithms together, the True Positive Rate turns out to be 63.67, while Specificity turns out to be 78.95. The F1 score is 70.10. The accuracy of this algorithm is 70.72. In accordance with combined techniques, this is the lowest performing algorithm.
2. Applying KNN and ANN algorithms together, the True Positive Rate turns out to be 75.49, while Specificity turns out to be 76.22. The F1 score is 77.10. The accuracy of this algorithm is 75.83. The same results come out when Random Forest and KNN algorithms are combined. It is a decent performing algorithm.

3. Applying Random Forest and ANN together, the True Positive Rate turns out to be 85.68, while Specificity turns out to be 88.60. The F1 score is 87.68. The accuracy of this algorithm is 87.02. It is a high performing algorithm.
4. Applying Random Forest and SVM algorithms together, the True Positive Rate turns out to be 85.94, while Specificity turns out to be 88.99. The F1 score is 87.98. The accuracy of this algorithm is 87.35. Of all the techniques used, this is the highest performing algorithm.

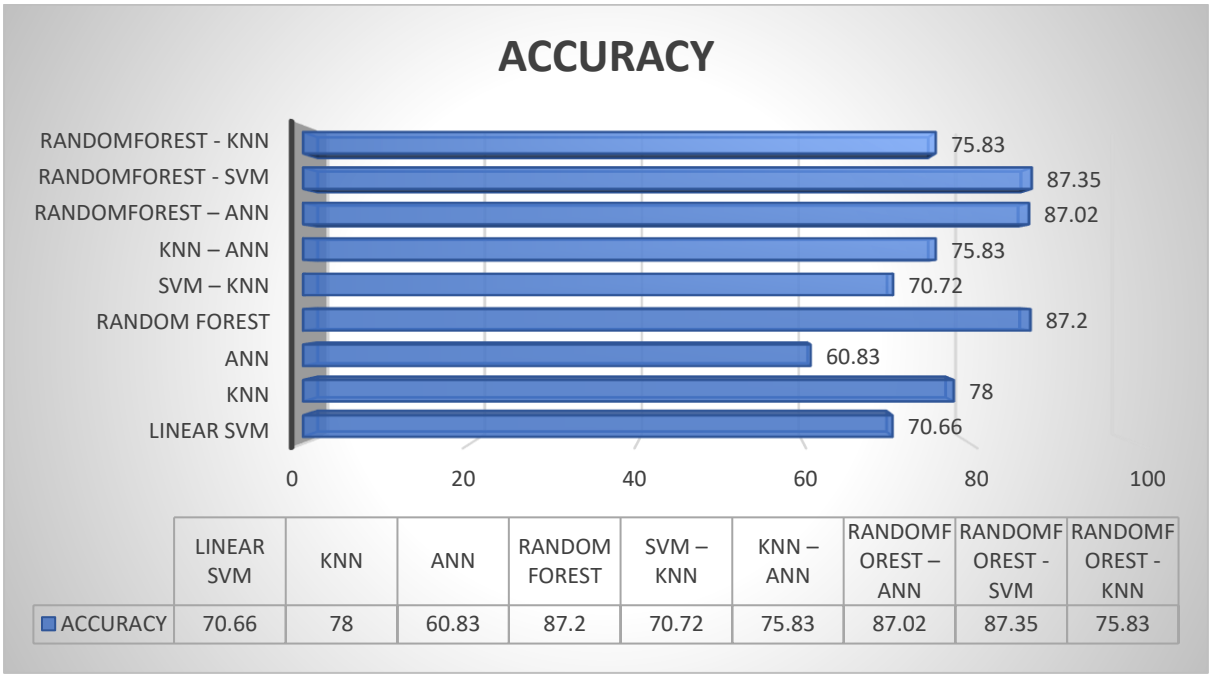
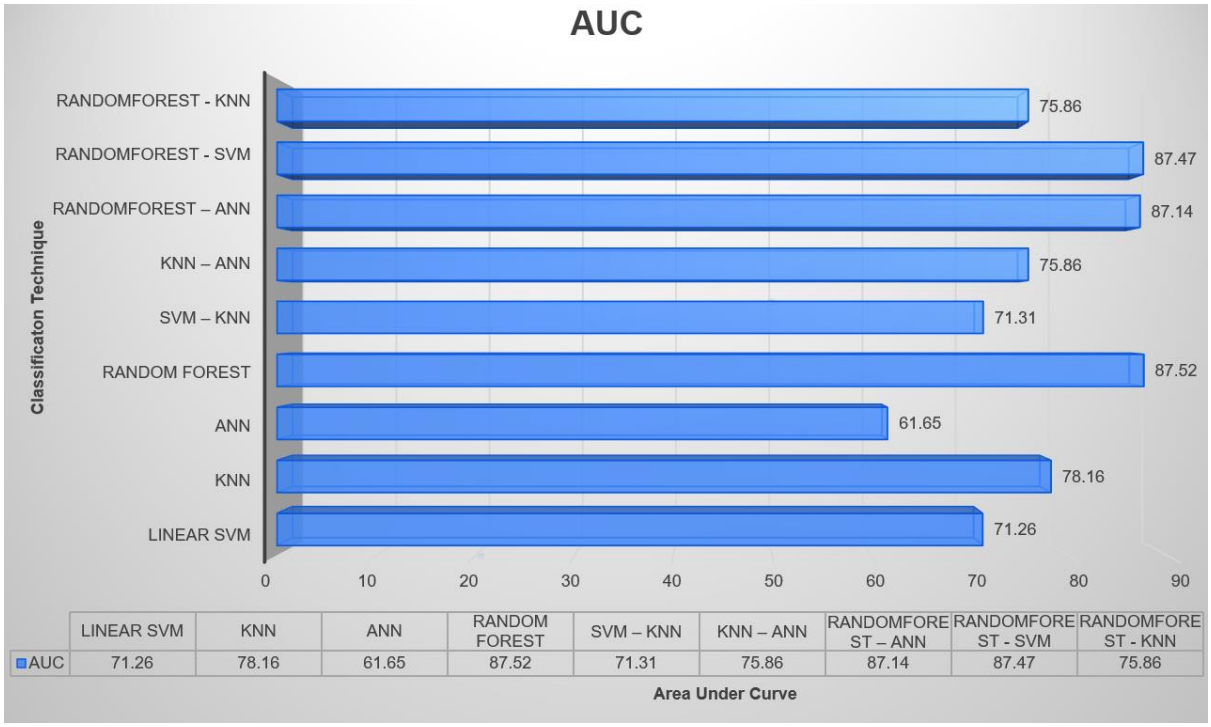
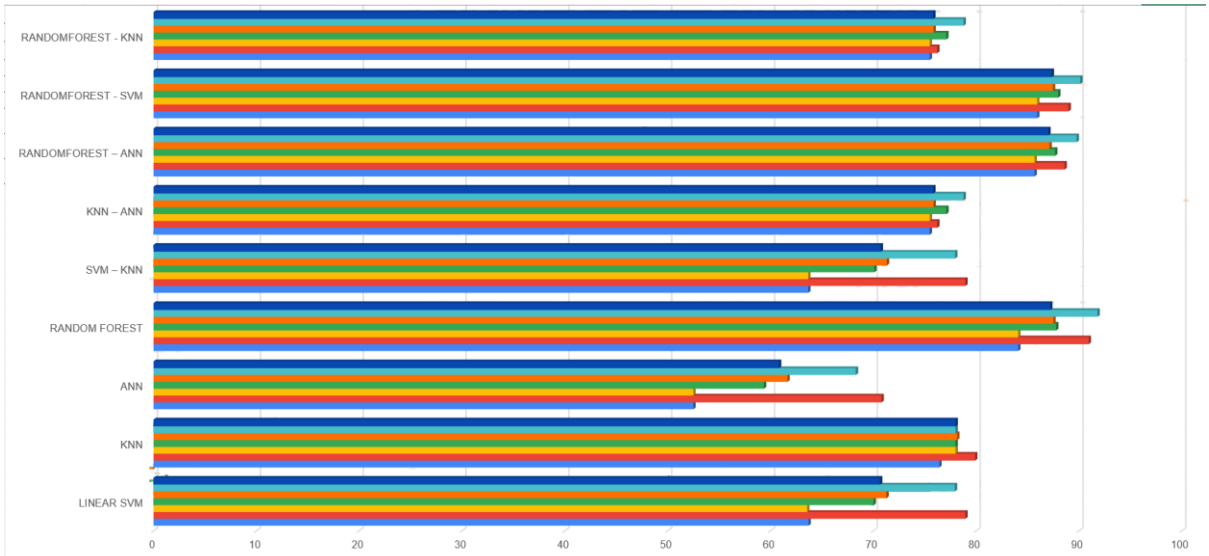


Figure 7: Accuracy Graph of Different Machine Learning Techniques



	LINEAR SVM	KNN	ANN	RANDOM FOREST	SVM – KNN	KNN – ANN	RANDOMFOREST – ANN	RANDOMFOREST - SVM	RANDOMFOREST - KNN
AUC	71.26	78.16	61.65	87.52	71.31	75.86	87.14	87.47	75.86

Figure 8: AUC Graph of Different Machine Learning Techniques



	LINEAR SVM	KNN	ANN	RANDOM FOREST	SVM – KNN	KNN – ANN	RANDOMFOREST – ANN	RANDOMFOREST - SVM	RANDOMFOREST - KNN
ACCURACY	70.66	78	60.83	87.2	70.72	75.83	87.02	87.35	75.83
PRECISION	77.94	78	68.3	91.8	77.96	78.78	89.78	90.13	78.78
AUC	71.26	78.16	61.65	87.52	71.31	75.86	87.14	87.47	75.86
F1_SCORE	70.03	78	59.37	87.78	70.1	77.1	87.68	87.98	77.1
RECALL_SCORE	63.57	78	52.5	84.09	63.67	75.49	85.68	85.94	75.49
SPECIFICITY	78.95	79.9	70.8	90.95	78.95	76.22	88.6	88.99	76.22
SENSITIVITY	63.7	76.42	52.5	84.09	63.67	75.49	85.68	85.94	75.49

Figure 9: Performance Measures Graph for different Machine Learning Techniques

Table 3: Performance Measures for Different Machine Learning Techniques

TECHNIQUE	SENSITIVITY	SPECIFICITY	RECALL_SCORE	F1_SCORE	AUC	PRECISION	ACCURACY
LINEAR SVM	63.7	78.95	63.57	70.03	71.26	77.94	70.66
KNN	76.42	79.9	78	78	78.16	78	78
ANN	52.5	70.8	52.5	59.37	61.65	68.3	60.83
RANDOM FOREST	84.09	90.95	84.09	87.78	87.52	91.8	87.2
SVM – KNN	63.67	78.95	63.67	70.1	71.31	77.96	70.72
KNN – ANN	75.49	76.22	75.49	77.1	75.86	78.78	75.83
RANDOMFOREST – ANN	85.68	88.6	85.68	87.68	87.14	89.78	87.02
RANDOMFOREST - SVM	85.94	88.99	85.94	87.98	87.47	90.13	87.35
RANDOMFOREST - KNN	75.49	76.22	75.49	77.1	75.86	78.78	75.83

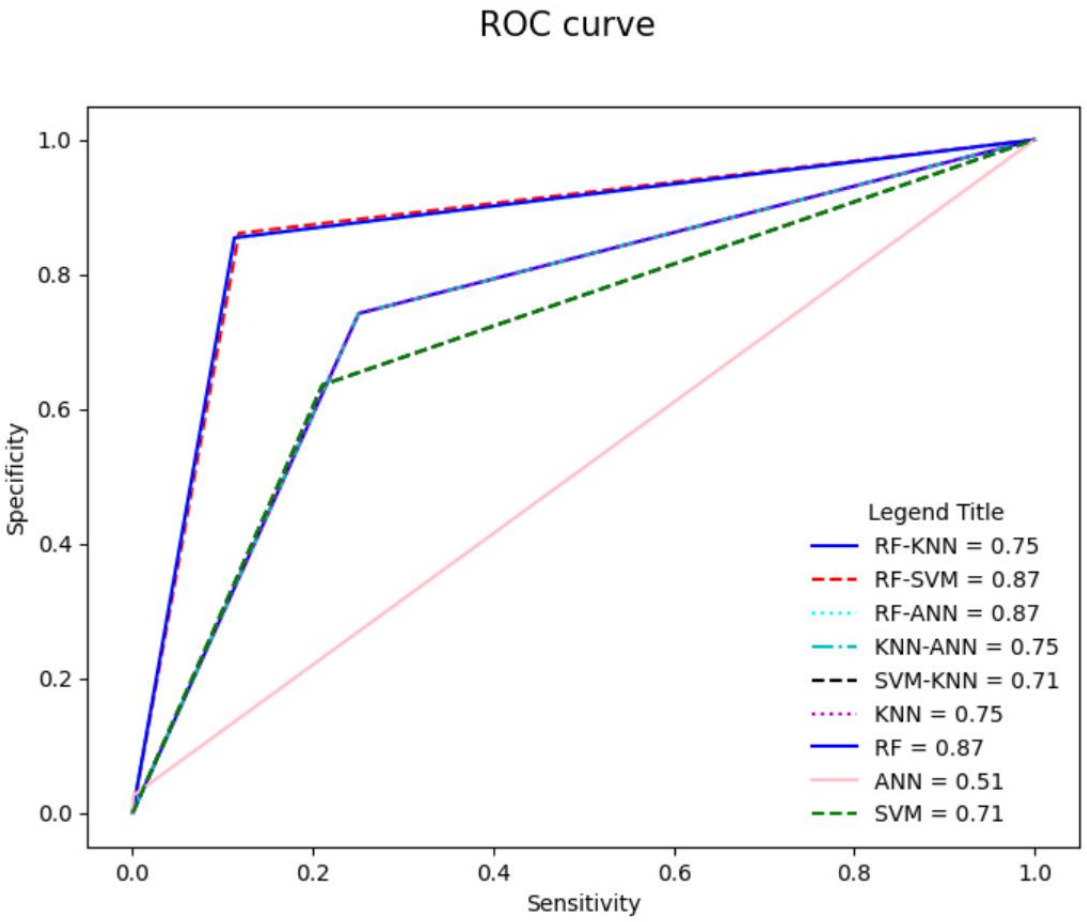


Figure 10: ROC curve of the Machine Learning Techniques

Chapter 5: Discussion and analysis of findings

In this project, we aimed to develop a predictive model for early detection of influenza using machine learning techniques. We utilized a dataset extracted from the Influenza Research database, consisting of 30531 records covering the period from 2006 to 2020.

After thorough pre-processing, we selected 18581 records for further analysis and implemented four different machine learning techniques: Support Vector Machines, K-nearest Neighbors, Artificial Neural Networks, and Random Forest. The performance of these methods was evaluated based on sensitivity, specificity, and accuracy.

The experimental results demonstrated that Random Forest with SVM was one of the most effective machine learning techniques for early detection of influenza, achieving a sensitivity of 87%, a specificity of 90%, and an accuracy of 87.35%.

The findings of this study have significant implications for the medical field, as early detection of influenza can lead to timely interventions and improve patient outcomes. Machine learning techniques, such as Random Forest, can aid in the development of more accurate and efficient influenza detection systems.

However, it is essential to note that our study has some limitations. The dataset used was relatively small, and the performance of the models may vary with larger datasets. Additionally, the accuracy of the models may be influenced by the quality of the data and the feature selection process.

Future research could focus on expanding the dataset, incorporating more features, and exploring other machine learning techniques to further improve the accuracy of the predictive model. Additionally, the model can be validated using real-world data to evaluate its effectiveness in a clinical setting.

In conclusion, this study highlights the potential of machine learning techniques in medical research and their effectiveness in the early detection of influenza. The findings can aid in the development of more accurate and efficient influenza detection systems, ultimately leading to improved patient outcomes.

Chapter 6: Conclusion and recommendations

Based on the results obtained, the following conclusions can be made:

1. Combined models perform highly better than their singular counterparts.
2. The ANN algorithm is not suitable for such labelled and varied data given its inaccuracy and metrical scores.
3. The Random Forest algorithm works very well with such a large amount of data and produces high scores and accuracy, irrespective of how its implemented.

In conformity with the objective of this paper and the methodology used, we have gone to our furthest extent to use different techniques up-to-date with the standards of usage at the time of writing. For future practises to surpass us, the processes for data mining should be improved and used in such a manner that processed data is understandable to humans as well as the model, without much use of binary and/or arithmetic values.

Further, given how combining two learning models produced better results than previous researches, using three or more models at the same time and use of different and/or more efficient models which have a better learning rate and require less pre-processing could be emphasised. Models could be programmed in such a way that they adapt quickly to newly comprehended data, without deviating too much from the desired results even if the data consists of some minor looked-over impurities

References

1. Early Detection of Influenza using machine learning techniques by Sajal Maheshwari, Anushka Sharma, Ranjan Kumar and Pratyush.
2. Moghadami M. A Narrative Review of Influenza: A Seasonal and Pandemic Disease. Iran J Med Sci. 2017 Jan;42(1):2-13. PMID: 28293045; PMCID: PMC5337761.
3. Spreeuwenberg P, Kroneman M, Paget J. Reassessing the Global Mortality Burden of the 1918 Influenza Pandemic. Am J Epidemiol. 2018 Dec 1;187(12):2561-2567. doi: 10.1093/aje/kwy191. PMID: 30202996; PMCID: PMC7314216.
4. W. Paul Glezen, Emerging Infections: Pandemic Influenza, Epidemiologic Reviews, Volume 18, Issue 1, 1996, Pages 64–76, https://doi.org/10.1093/oxfordjournals.epirev.a017917
5. J. S. Malik Peiris, Menno D. de Jong, Yi Guan.: Avian Influenza Virus (H5N1): a Threat to Human Health. Clinical Microbiology Reviews 20 (2), 243-267, (2007). DOI: 10.1128/CMR.00037-06
6. Sullivan SJ, Jacobson RM, Dowdle WR, Poland GA. 2009 H1N1 influenza. Mayo Clin Proc. 2010 Jan;85(1):64-76. doi: 10.4065/mcp.2009.0588. Epub 2009 Dec 10. PMID: 20007905; PMCID: PMC2800287.
7. Grohskopf LA, Alyanak E, Ferdinands JM, et al. Prevention and Control of Seasonal Influenza with Vaccines: Recommendations of the Advisory Committee on Immunization Practices, United States, 2021–22 Influenza Season. MMWR Recomm Rep 2021;70(No. RR-5):1–28. DOI: http://dx.doi.org/10.15585/mmwr.rr7005a1.
8. Squires RB, Noronha J, Hunt V, García-Sastre A, Macken C, Baumgarth N, Suarez D, Pickett BE, Zhang Y, Larsen CN, Ramsey A, Zhou L, Zaremba S, Kumar S, Deitrich J, Klem E, Scheuermann RH. Influenza research database: an integrated bioinformatics resource for influenza research and surveillance. Influenza Other Respir Viruses. 2012 Nov;6(6):404-16. doi: 10.1111/j.1750-2659.2011.00331.x. Epub 2012 Jan 20. PMID: 22260278; PMCID: PMC3345175.
9. Lavrac N. Selected techniques for data mining in medicine. Artif Intell Med. 1999 May;16(1):3-23. doi: 10.1016/s0933-3657(98)00062-1. PMID: 10225344.
10. Tan AC, Gilbert D. Ensemble machine learning on gene expression data for cancer classification. Appl Bioinformatics. 2003;2(3 Suppl):S75-83. PMID: 15130820.

11. Battineni G, Sagaro GG, Chinatalapudi N, Amenta F. Applications of Machine Learning Predictive Models in the Chronic Disease Diagnosis. J Pers Med. 2020 Mar 31;10(2):21. doi: 10.3390/jpm10020021. PMID: 32244292; PMCID: PMC7354442.
12. Rajalakshmi V., Sasikala D., Kala A., A Predictive Analysis for Heart Disease Using Machine Learning. In: Intelligent Computing and Applications. Advances in Intelligent Systems and Computing, vol 1172. Springer (2021), DOI: 10.1007/978-981-15-5566-4_42
13. Meyfroidt G, Güiza F, Ramon J, Bruynooghe M. Machine learning techniques to examine large patient databases. Best Pract Res Clin Anaesthesiol. 2009 Mar;23(1):127-43. doi: 10.1016/j.bpa.2008.09.003. PMID: 19449621.
14. The data used for this study was obtained from https://www.bv-brc.org.
15. Jane V. Carter, Jianmin Pan, Shesh N. Rai, Susan Galandiuk.: ROC-ing along: Evaluation and interpretation of receiver operating characteristic curves,Surgery 159(6), 1638-1645, (2016). DOI: 10.1016/j.surg.2015.12.029
16. Zhu, W., Zeng, N. and Wang, N. (2010) Sensitivity, Specificity, Accuracy, Associated Confidence Interval and ROC Analysis with Practical SAS Implementations. NESUG Proceedings: Health Care and Life Sciences, Baltimore, Maryland. http://www.nesug.org/Proceedings/nesug10/hl/hl07.pdf
17. Hastie T., Tibshirani R., Friedman J.: The Elements of Statistical Learning. Springer Series in Statistics. Springer, New York, NY (2009). DOI: 10.1007/978- 0-387-84858-7_15
18. Breiman, L. Random Forests. Machine Learning 45, 5–32 (2001). https://doi.org/10.1023/A:1010933404324
19. Venables, W.N. and Ripley, B.D. (2002) Modern Applied Statistics with S. Springer, New York, 271-300.https://doi.org/10.1007/978-0-387-21706-2
20. Ripley, B.D. (1996) Pattern Recognition and Neural Networks. Cambridge University Press, Cambridge. https://doi.org/10.1017/CBO9780511812651
21. JOUR Veloso, Bruno, Leal, Fátima, Gonzalez-Velez, Horacio, Malheiro, Benedita, Burguillo, Juan, 2018/06/01, Scalable data analytics using crowdsourced repositories and streams, 122, 10.1016/j.jpdc.2018.06.013, Journal of Parallel and Distributed Computing.
22. Arturo López Pineda, Ye Ye, Shyam Visweswaran, Gregory F. Cooper, Michael M. Wagner, Fuchiang (Rich) Tsui, Comparison of machine learning classifiers for influenza detection from emergency department free-text reports,,Journal of

Biomedical Informatics, Volume 58, 2015, Pages 60-69, ISSN 1532-0464, https://doi.org/10.1016/j.jbi.2015.08.019.
(https://www.sciencedirect.com/science/article/pii/S1532046415001872)

23. BOOK, Marquez, Edna, Barron, Valeria, 2019/11/01, 1, 5, Artificial Intelligence system to support the clinical decision for influenza, 10.1109/ROPEC48299.2019.9057056
24. Monto AS, Gravenstein S, Elliott M, Colopy M, Schweinle J. Clinical signs and symptoms predicting influenza infection. Arch Intern Med. 2000 Nov 27;160(21):3243-7. doi: 10.1001/archinte.160.21.3243. PMID: 11088084.

Appendix

Tools

To carry out this project, Python was chosen as the programming language. This decision was based on the fact that Python allows for a wide range of operations to be performed with relatively few lines of code when compared to other languages. Additionally, Python has the Pandas library, which is highly regarded for data processing and management.

The library's data visualization capabilities were also extremely helpful in understanding how different models worked with the input data. Another advantage of Python is that it is an open-source language, which meant that the project could be completed with minimal financial resources. For writing and implementing Python code, the Jupyter Notebook was used.

For making “Flow chart of methods” we use Lucid Chart web application and for figure 2-8 we used Excel and for figure 10 we used matplotlib.

Packages

Pandas

NumPy

Scikit-Learn

Seaborn

Matplotlib

Keras

Code and Outputs

The following code was implemented to divide the Symptom attribute into individual symptom attributes:

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as snb
```

```
In [2]: symp = pd.read_csv("D:\project\BVBRC_surveillance.csv")
```

```
In [3]: symp = symp.drop(['Project Identifier','Contributing Institution','Sample Transport Medium','Sample Receipt Date','Submission Dat
        ,'Collection Longitude','Pathogen Test Interpretation','Species','Type','Maintenance Medication','Types of Allergies','Influenza
```

```
In [4]: col = symp.columns.tolist()
```

```
In [6]: s = symp['Symptoms'].str.split(';', expand=True).stack()

        # Create a new dataframe from the stacked series
        temp_df = pd.DataFrame(s.values, index=s.index.droplevel(-1), columns=['key_value_pair'])

        # Split the new dataframe into two columns, key and value
        temp_df[['key', 'value']] = temp_df['key_value_pair'].str.split(':', expand=True)

        # Pivot the new dataframe to create separate columns for each key
        pivot_df = temp_df.pivot(columns='key', values='value')

        # Join the pivoted dataframe back to the original dataframe
        result_df = symp.join(pivot_df)
```

```
In [9]: result_df
```

Email Address	Collection Year	Collection Season	Collection Country	Collection State Province	Collection City	...	other symptoms	rash	running nose	short breath	sinus congestion	sudden onset	temperature	throat	vomiting	wheezing
ry@yahoo.com	2009	NaN	China	Shantou	NaN	...	NaN	NaN	Y	U	NaN	NaN	NaN	Y	NaN	NaN
ry@yahoo.com	2008	NaN	China	Shantou	NaN	...	NaN	NaN	N	U	NaN	NaN	NaN	Y	NaN	NaN
NaN	2013	NaN	USA	Massachusetts	Boston, MA	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
@niaidceirs.org	NaN	2019-2020	United States Of America	Maryland	NaN	...	Restricted Access	NaN	Y	Y	NaN	NaN	NaN	Y	Y	Y
ry@yahoo.com	2009	NaN	China	Shantou	NaN	...	NaN	NaN	N	U	NaN	NaN	NaN	N	NaN	NaN

Code to divide the Symptom attribute to each individual symptom attribute (Handling the values 'Yes' and 'No').

```
In [17]: symp = ['Fever', 'aches', 'arthalgia', 'chest pain', 'chills',
                 'conjunctivitis', 'cough', 'diarrhea', 'dyspnea', 'ear ache', 'fatigue',
                 'fever', 'headache', 'loss of appetite', 'malaise',
                 'myalgia', 'nausea', 'other symptoms', 'rash', 'running nose',
                 'short breath', 'sinus congestion', 'sudden onset',
                 'throat', 'vomiting', 'wheezing']

In [18]: for i in symp:
             pro[i] = pro[i].map({'Yes': 1, 'No': 0 , 'Y' : 1, 'N': 0 , 'Not Provided' : 'na' , 'Not Collected' : 'na', 'none' : 0 , 'Rest

In [20]: pro.fillna(0)
```

Out[20]:

	Unnamed: 0	Sample Identifier	Sequence Accession	Sample Material	Collector Institution	Contact Email Address	Collection Year	Collectio Seaso
0	0	491705	0	NS	Hong Kong University	fluquery@yahoo.com	2009	
1	1	490248	0	NS	Hong Kong University	fluquery@yahoo.com	2008	
2	2	NIGSP_YGA_00087	CY168429,CY168430,CY168427,CY168428,CY168423,C...	NS	Harvard Medical School	0	2013	
3	3	02-11-R-0661	0	NAS	Johns Hopkins University-CEIRS2	irdsupport@niaidceirs.org	0	2019-202
4	4	491450	0	NS	Hong Kong University	fluquery@yahoo.com	2009	
...	...	...	...	...	...	...	...	...

Code to map Positive and Negative value of Target variable with 1 and 0 respectively. Handling Age variable disrupted values.

```
In [27]: import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
         import seaborn as snb

In [28]: df = pd.read_csv('influenza5.csv')

In [29]: df['Pathogen Test Result'] = df['Pathogen Test Result'].map({'Negative':0 ,'Positive':1})

In [30]: df['Pathogen Test Result'] = df['Pathogen Test Result'].fillna(0)

In [31]: df['Pathogen Test Result'] = df['Pathogen Test Result'].astype(int)

In [32]: age = df['Host Age']

In [33]: age = age.replace('Not Provided' , '0').replace('Not Collected' , '0').replace('>90' , '95')

In [34]: age = age

In [35]: age = age.astype(float)

In [36]: age = age.astype(int)

In [37]: age

Out[37]: 0         0
         1         0
         2        42
         3        26
         4         0
                  ..
         30525    58
         30526     0
         30527    74
         30528     4
         30529    42
         Name: Host Age, Length: 30530, dtype: int32

In [38]: df['Host Age'] = age
```

Code to map Male and Female value of Host Sex with 1 and 0 respectively.

```
In [18]: df.index = np.arange(len(df))

In [19]:  df = df.drop(['Unnamed: 0'],axis =1)

In [20]: df['Host Sex'] = df['Host Sex'].map({'Female' : 0 , 'Male' : 1})

In [21]: y=[]
         x=df['Host Sex'].isnull()
         for i in range(len(x)):
             if  x[i]:
                 y.append(i)
         df = df.drop(y)
```

Pre-processing Chronic Condition variable of Data set.

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib as plt
        import seaborn as snb

In [2]: pro = pd.read_csv("influenza2.csv")

        C:\Users\raman\AppData\Local\Temp\ipykernel_14176\4081774908.py:1: DtypeWarning: Columns (26) have mixed types. Specify dtype o
        ption on import or set low_memory=False.
          pro = pd.read_csv("influenza2.csv")

In [3]: pro_chro_condition = pro['Chronic Conditions'].astype(str)

In [4]: chro_list = list(pro_chro_condition)

In [5]: x=[]
        for i in chro_list:
            i=i.replace("'","")
            i=i.replace('"','')
            x.append(i)

In [6]: inf_series = pd.Series(x)
```

In [7]:
```
inf_list = []
for i in range(len(x)):
    if 'nan' in x[i]:
        inf_list.append(0)
    elif 'OTH-none' in x[i]:
        inf_list.append(0)
    elif 'OTH- none' in x[i]:
        inf_list.append(0)
    elif 'Not Provided' in x[i]:
        inf_list.append(0)
    elif 'Not Collected' in x[i]:
        inf_list.append(0)
    elif 'HPT' == x[i]:
        inf_list.append(1)
    elif 'RSP' == x[i]:
        inf_list.append(2)
    elif 'chronic lung disease' == x[i]:
        inf_list.append(3)
    elif 'END' == x[i]:
        inf_list.append(4)
    elif 'CDV' == x[i]:
        inf_list.append(5)
    elif 'ASM' == x[i]:
        inf_list.append(6)
    elif 'HEM' == x[i]:
        inf_list.append(7)
    elif 'NRL' == x[i]:
        inf_list.append(8)
    elif 'OBS' == x[i]:
        inf_list.append(9)
    elif 'IMS' == x[i]:
        inf_list.append(10)
    elif 'CAN' == x[i]:
        inf_list.append(11)
    elif 'URO' == x[i]:
        inf_list.append(12)
    elif 'FLU' == x[i]:
        inf_list.append(13)
    elif 'immuno' == x[i]:
        inf_list.append(14)
    elif 'diabetes' == x[i]:
        inf_list.append(15)
    elif 'obesity' == x[i]:
        inf_list.append(16)
    elif 'END;CAN' in x[i] :
        inf_list.append(17)
    elif 'HEM;CAN' in x[i]:
        inf_list.append(18)
```

```
    elif 'HPT;END' in x[i]:
        inf_list.append(19)
    elif 'CDV;HPT' in x[i]:
        inf_list.append(20)
    elif 'END;NRL' in x[i]:
        inf_list.append(21)
    elif 'CDV;NRL' in x[i]:
        inf_list.append(22)
    elif 'URO;END' in x[i]:
        inf_list.append(23)
    elif 'CDV;URO' in x[i]:
        inf_list.append(24)
    elif 'RSP;ASM' in x[i]:
        inf_list.append(25)
    elif 'OBS;OTH-chronic lung disease;ASM' in x[i]:
        inf_list.append(26)
    elif 'CDV;END'in x[i]:
        inf_list.append(27)
    elif 'CDV;END;NRL'in x[i]:
        inf_list.append(28)
    elif 'OTH-chronic lung disease;ASM' == x[i]:
        inf_list.append(29)
    else:
        inf_list.append(30)
```

Code for removing unnecessary variable.

In [1]:
```
import pandas as pd
```

In [2]:
```
df = pd.read_csv('influenza3.csv')
```

```
C:\Users\raman\AppData\Local\Temp\ipykernel_1980\1258131324.py:1: DtypeWarning: Columns (27) have mixed types. Specify dtype option on import or set low_memory=False.
  df = pd.read_csv('influenza3.csv')
```

In [3]:
```
df.columns
```

Out[3]:
```
Index(['Unnamed: 0.1', 'Unnamed: 0', 'Sample Identifier', 'Sequence Accession',
       'Sample Material', 'Collector Institution', 'Contact Email Address',
       'Collection Year', 'Collection Season', 'Collection Country',
       'Collection State Province', 'Collection City', 'Pathogen Test Type',
       'Pathogen Test Result', 'Subtype', 'Strain', 'Host Identifier',
       'Host ID Type', 'Host Species', 'Host Common Name', 'Host Group',
       'Host Sex', 'Host Age', 'Chronic Conditions', 'Symptoms',
       'Additional Metadata', 'ALTCON', 'Acute Respiratory Distress Syndrome',
       'Fever', 'Temperature', 'aches', 'arthalgia', 'chest pain', 'chills',
       'conjunctivitis', 'cough', 'diarrhea', 'dyspnea', 'ear ache', 'fatigue',
       'fever', 'headache', 'high_temp_home', 'loss of appetite', 'malaise',
       'myalgia', 'nausea', 'other symptoms', 'rash', 'running nose',
       'short breath', 'sinus congestion', 'sudden onset', 'temperature',
       'throat', 'vomiting', 'wheezing'],
      dtype='object')
```

In [4]:
```
df = df.drop(['Unnamed: 0.1', 'Unnamed: 0', 'Sample Identifier', 'Sequence Accession',
       'Sample Material', 'Collector Institution', 'Contact Email Address','Collection Country',
       'Collection State Province', 'Collection City', 'Pathogen Test Type','Subtype', 'Strain', 'Host Identifier',
       'Host ID Type', 'Host Species', 'Host Common Name', 'Host Group','Symptoms',
       'Additional Metadata', 'ALTCON','temperature','high_temp_home','Temperature'],axis = 1)
```

In [5]:
```
df
```

Out[5]:

	Collection Year	Collection Season	Pathogen Test Result	Host Sex	Host Age	Chronic Conditions	Acute Respiratory Distress Syndrome	Fever	aches	arthalgia	...	nausea	other symptoms	rash	running nose	short breath	si congest
0	2009	NaN	Negative	Male	0	0	NaN	1	NaN	NaN	...	1	NaN	NaN	1	1	N
1	2008	NaN	Negative	Female	0	0	NaN	1	NaN	NaN	...	1	NaN	NaN	0	1	N
2	2013	NaN	Positive	Female	42	0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	N
3	NaN	2019-2020	Negative	Female	26	26	NaN	1	NaN	NaN	...	1	0	NaN	1	1	N
4	2009	NaN	Negative	Female	0	0	NaN	1	NaN	NaN	...	1	NaN	NaN	0	1	N
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
30525	NaN	2016-2017	Negative	Female	58	8	NaN	na	NaN	NaN	...	0	0	NaN	na	0	N

Code to merge Collection Year variable and Collection Season variable.

```
In [6]: df_year = df['Collection Year'].astype(str)

In [7]: df_season = df['Collection Season'].astype(str)

In [8]: df_year

Out[8]: 0        2009
        1        2008
        2        2013
        3         nan
        4        2009
                 ...
        30525     nan
        30526    2008
        30527     nan
        30528     nan
        30529    2014
        Name: Collection Year, Length: 30530, dtype: object

In [9]: df_season

Out[9]: 0              nan
        1              nan
        2              nan
        3        2019-2020
        4              nan
                   ...
        30525    2016-2017
        30526          nan
        30527    2018-2019
        30528         2020
        30529          nan
        Name: Collection Season, Length: 30530, dtype: object

In [10]: df_season[3][:4]

Out[10]: '2019'

In [11]: for i in range(len(df_year)):
             if df_year[i] == 'nan':
                 df_year[i] = df_season[i][:4]

In [12]: for i in range(len(df_year)):
             df_year[i] = df_year[i][:4]

In [13]: df_year.tolist()

Out[13]: ['2009',
          '2008',
          '2013',
          '2019',
          '2009',
          '2014',
          '2012',
          '2015',
          '2020',
          '2012',
          '2009',
          '2017',
          '2020',
          '2018',
          '2016',
          '2012',
          '2013',
          '2018',
          '2008',

In [14]: df['Collection Year'] = df_year

In [15]: df = df.drop(['Collection Season'],axis = 1)

In [16]: df['Collection Year']

Out[16]: 0        2009
         1        2008
         2        2013
         3        2019
         4        2009
                  ...
         30525    2016
         30526    2008
         30527    2018
         30528    2020
         30529    2014
         Name: Collection Year, Length: 30530, dtype: object
```

Code for removing null value rows, symptoms having 95% of null values and filling null values with 0 in remaing values.

```
In [17]: dict(df.isnull().sum(axis = 1))
```

```
Out[17]: {0: 16,
          1: 16,
          2: 27,
          3: 10,
          4: 16,
          5: 22,
          6: 27,
          7: 10,
          8: 10,
          9: 13,
          10: 16,
          11: 10,
          12: 10,
          13: 10,
          14: 11,
          15: 13,
          16: 27,
          17: 10,
          18: 16,
```

```
In [18]: dict(df.isnull().sum(axis = 0))
```

```
Out[18]: {'Collection Year': 0,
          'Pathogen Test Result': 0,
          'Host Sex': 102,
          'Host Age': 16,
          'Chronic Conditions': 0,
          'Acute Respiratory Distress Syndrome': 30529,
          'Fever': 2302,
          'aches': 29976,
          'arthalgia': 30530,
          'chest pain': 30006,
          'chills': 12528,
          'conjunctivitis': 13991,
          'cough': 2266,
          'diarrhea': 2408,
          'dyspnea': 30002,
          'ear ache': 30020,
          'fatigue': 29914,
          'fever': 2302,
          'headache': 2340,
          'loss of appetite': 14494,
          'malaise': 2954,
          'myalgia': 2899,
          'nausea': 2435,
          'other symptoms': 14206,
          'rash': 30094,
          'running nose': 2324,
          'short breath': 2944,
          'sinus congestion': 29945,
          'sudden onset': 30530,
          'throat': 2337,
          'vomiting': 12565,
          'wheezing': 13108}
```

```
In [19]: df  = df.drop(['Acute Respiratory Distress Syndrome','arthalgia','sudden onset','chest pain','ear ache','rash','dyspnea'],axis=1)
```

```
In [20]: df = df.drop(['sinus congestion','fatigue','aches'],axis=1)
```

```
In [21]: df = df.fillna(0)
         df = df.replace('na',0)
```

In [22]:
```
df
```

Out[22]:

	Collection Year	Pathogen Test Result	Host Sex	Host Age	Chronic Conditions	Fever	chills	conjunctivitis	cough	diarrhea	...	loss of appetite	malaise	myalgia	nausea	other symptoms	runnir nos
0	2009	Negative	Male	0	0	1	0	0	1	1	...	0	1	1	1	0	
1	2008	Negative	Female	0	0	1	0	0	0	1	...	0	1	1	1	0	
2	2013	Positive	Female	42	0	0	0	0	0	0	...	0	0	0	0	0	
3	2019	Negative	Female	26	26	1	1	0	1	0	...	1	1	1	1	0	
4	2009	Negative	Female	0	0	1	0	0	1	1	...	0	1	1	1	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
30525	2016	Negative	Female	58	8	0	1	1	0	0	...	0	0	0	0	0	
30526	2008	Negative	Male	0	0	1	0	0	1	1	...	0	1	1	1	0	
30527	2018	Positive	Female	74	26	0	1	0	1	0	...	1	1	1	0	0	
30528	2020	Negative	Male	4	0	1	0	0	1	0	...	0	0	0	0	0	
30529	2014	Positive	Male	42	0	0	1	0	1	0	...	0	0	0	0	0	

30530 rows × 22 columns

Code for splitting data set for training and testing.

In [2]:
```
# Load the dataset
df = pd.read_csv('inf.csv')
df = df.drop(['Unnamed: 0'],axis = 1)
```

In [3]:
```
df['Pathogen Test Result'].value_counts()
```

Out[3]:
```
1    10028
0     8489
Name: Pathogen Test Result, dtype: int64
```

In [4]:
```
df.describe()
```

Out[4]:

	Collection Year	Pathogen Test Result	Host Sex	Host Age	Chronic Conditions	Fever	chills	conjunctivitis	cough	diarrhea	...	
count	18517.000000	18517.000000	18517.000000	18517.000000	18517.000000	18517.000000	18517.000000	18517.000000	18517.000000	18517.000000	...	1851
mean	2015.881838	0.541556	0.471405	34.079657	7.982341	0.442944	0.400443	0.230545	0.448615	0.122752	...	
std	2.855294	0.498284	0.499195	22.497576	11.803348	0.496747	0.490001	0.421193	0.497366	0.328161	...	
min	2006.000000	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	
25%	2015.000000	0.000000	0.000000	16.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	
50%	2016.000000	1.000000	0.000000	32.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	
75%	2018.000000	1.000000	1.000000	51.000000	18.000000	1.000000	1.000000	0.000000	1.000000	0.000000	...	
max	2020.000000	1.000000	1.000000	97.000000	30.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...	

8 rows × 22 columns

In [5]:
```
# Split the data into features and target variable
X = df.drop(['Pathogen Test Result'],axis = 1)
Y = df['Pathogen Test Result']
```

In [8]:
```
# Split the data into training and testing sets
X_train , X_test , Y_train , Y_test = train_test_split(X,Y,test_size= 0.3, random_state=2)
```

Code for implementing SVM Model with Linear Kernel.

In [1]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score , recall_score , f1_score
from sklearn.metrics import roc_curve , auc
import matplotlib.pyplot as plt
```

In [7]:
```
st_x= StandardScaler()
X_train= st_x.fit_transform(X_train)
X_test= st_x.transform(X_test)
```

In [8]:
```
# Train the model on the training set
classifier = SVC(kernel='linear', random_state=2)
classifier.fit(X_train, Y_train)
```

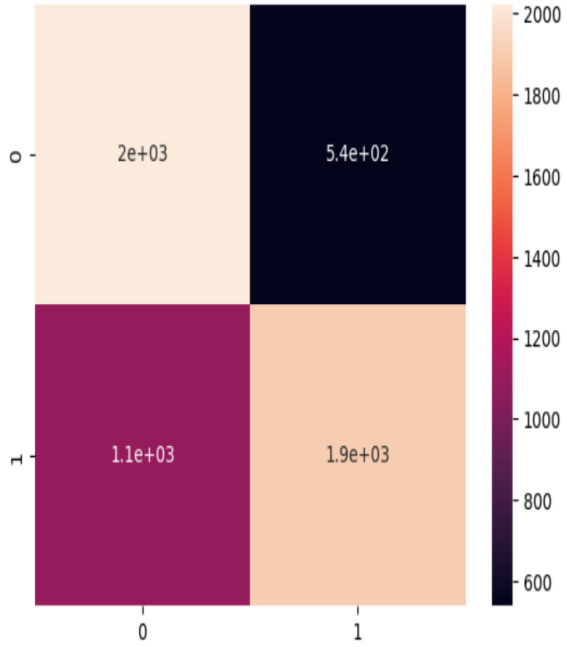
Out[8]:
```
SVC(kernel='linear', random_state=2)
```

In [9]:
```
# Use the trained model to make predictions on the testing set
y_pred = classifier.predict(X_test)
```

In [10]:
```
cm= confusion_matrix(Y_test, y_pred)
```

In [11]:
```
snb.heatmap(cm,annot = True)
cm
```

Out[11]:
```
array([[2022,  539],
       [1091, 1904]], dtype=int64)
```



In [12]:
```
score =accuracy_score(Y_test,y_pred)
print('accuracy on testing Data :',round(score*100 ,2),'%')
```

```
accuracy on testing Data : 70.66 %
```

In [13]:
```
y_pred2 =classifier.predict(X_train)
```

In [14]:
```
score2 =accuracy_score(y_pred2, Y_train)
print('accuracy on Training Data :',round(score2*100 ,2),'%')
```

```
accuracy on Training Data : 71.1 %
```

In [15]:
```
# precision for trainig and testing
print('for testing  : ' , round(precision_score(Y_test , y_pred)*100,2),'%')
print('for training : ' , round(precision_score(Y_train , y_pred2)*100,2),'%')
```

```
for testing  :  77.94 %
for training :  78.67 %
```

In [16]:
```
# Recall for trainig and testing
print('for testing  : ' , round(recall_score(Y_test , y_pred)*100,2),'%')
print('for training : ' , round(recall_score(Y_train , y_pred2)*100,2),'%')
```

```
for testing  :  63.57 %
for training :  64.13 %
```

In [17]:
```
# f1_score for trainig and testing
print('for testing  : ' , round(f1_score(Y_test , y_pred)*100,2),'%')
print('for training : ' , round(f1_score(Y_train , y_pred2)*100,2),'%')
```

```
for testing  :  70.03 %
for training :  70.66 %
```

In [18]:
```
# for sensitivity and specificity
tn, fp, fn, tp = cm.ravel()
```

In [19]:
```
specificity = tn / (tn+fp)
print('specificity for data is : ',round(specificity*100,2),'%')
```

```
specificity for data is :  78.95 %
```

In [20]:
```
sensitivity = tp / (tp+fn)
print('sensitivity for data is : ',round(sensitivity*100,2),'%')
```

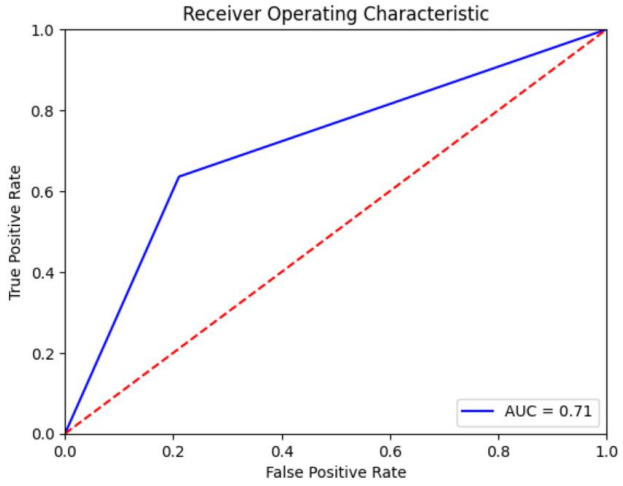
```
sensitivity for data is :  63.57 %
```

In [21]:
```
#ROC and AUC
fpr, tpr, threshold = roc_curve(Y_test,y_pred)
roc_auc = auc(fpr, tpr)
```

In [22]:
```
print('Area under curve : ',round(roc_auc*100,2),'%')
```

```
Area under curve :  71.26 %
```

In [23]:
```
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```



Code for implementing RandomForest Model.

In [1]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score , recall_score , f1_score , confusion_matrix
from sklearn.metrics import roc_curve , auc
import matplotlib.pyplot as plt
import seaborn as snb
```

In [5]:
```
# Create a random forest classifier object with 100 trees
rf = RandomForestClassifier(n_estimators=100)
```

In [6]:
```
# Train the model on the training set
rf.fit(X_train, y_train)
```

Out[6]:
```
RandomForestClassifier
RandomForestClassifier()
```

In [7]:
```
# Use the trained model to make predictions on the testing set
y_pred = rf.predict(X_test)
y_pred2 = rf.predict(X_train)
```

In [8]:
```
# Evaluate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", round(accuracy*100,2),'%')
```

```
Accuracy: 87.34 %
```

In [9]:
```
# precision for trainig and testing
print('for testing  : ' , round(precision_score(y_test , y_pred)*100,2),'%')
print('for training : ' , round(precision_score(y_train , y_pred2)*100,2),'%')
```

```
for testing  :  90.63 %
for training :  96.29 %
```

In [10]:
```
# Recall for trainig and testing
print('for testing  : ' , round(recall_score(y_test , y_pred)*100,2),'%')
print('for training : ' , round(recall_score(y_train , y_pred2)*100,2),'%')
```

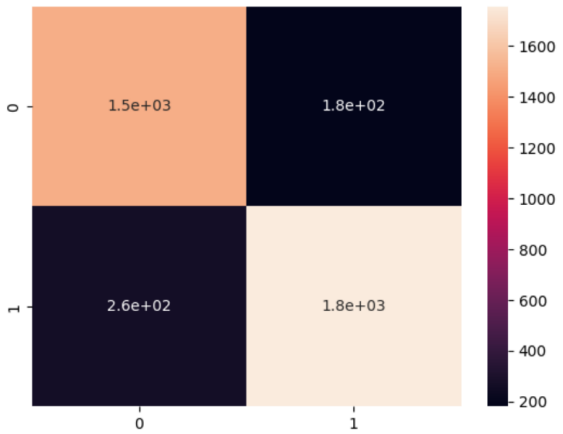
```
for testing  :  85.66 %
for training :  89.86 %
```

In [11]:
```
# f1_score for trainig and testing
print('for testing  : ' , round(f1_score(y_test , y_pred)*100,2),'%')
print('for training : ' , round(f1_score(y_train , y_pred2)*100,2),'%')
```

```
for testing  :  88.08 %
for training :  92.96 %
```

In [12]:
```
cm = confusion_matrix(y_test,y_pred)
snb.heatmap(cm, annot =True)
cm
```

Out[12]:
```
array([[1504,  180],
       [ 265, 1755]], dtype=int64)
```



In [13]:
```
# for sensitivity and specificity
tn, fp, fn, tp = cm.ravel()
```

In [14]:
```
specificity = tn / (tn+fp)
print('specificity for data is : ',round(specificity*100,2),'%')
```

```
specificity for data is :  89.31 %
```

In [15]:
```
sensitivity = tp / (tp+fn)
print('sensitivity for data is : ',round(sensitivity*100,2),'%')
```

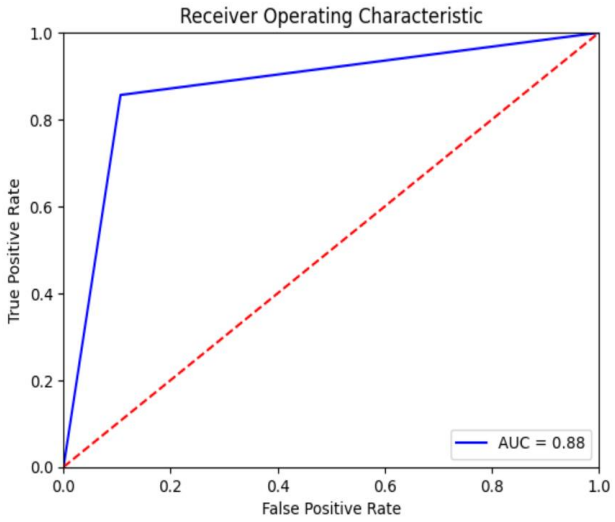
```
sensitivity for data is :  86.88 %
```

In [16]:
```
#ROC and AUC
fpr, tpr, threshold = roc_curve(y_test,y_pred)
roc_auc = auc(fpr, tpr)
```

In [17]:
```
print('Area under curve : ',round(roc_auc*100,2),'%')
```

```
Area under curve :  87.51 %
```

In [18]:
```
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```



Code for implementing K-NN Model.

```
In [1]: import pandas as pd
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import confusion_matrix , classification_report
        from sklearn.metrics import accuracy_score
        from sklearn.metrics import precision_score , recall_score , f1_score
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.metrics import roc_curve , auc
        import matplotlib.pyplot as plt
        import seaborn as snb
```

```
In [5]: classifier = KNeighborsClassifier(n_neighbors= 5)
        classifier.fit(X_train,Y_train)
```

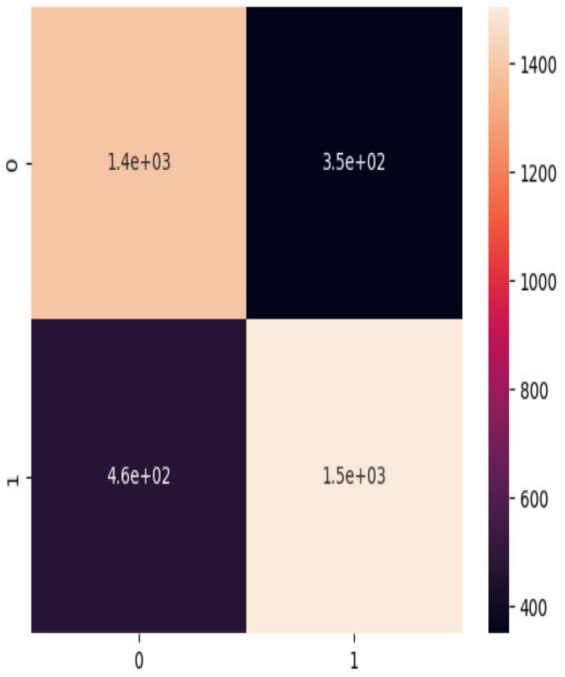
```
Out[5]: KNeighborsClassifier
        KNeighborsClassifier()
```

```
In [6]: Y_pred = classifier.predict(X_test)
```

```
In [7]: cm= confusion_matrix(Y_test, Y_pred)
```

```
In [8]: snb.heatmap(cm,annot=True)
        cm
```

```
Out[8]: array([[1387,  349],
               [ 464, 1504]], dtype=int64)
```



In [9]:
```
print(classification_report(Y_test,Y_pred))
```

```
              precision    recall  f1-score   support

           0       0.75      0.80      0.77      1736
           1       0.81      0.76      0.79      1968

    accuracy                           0.78      3704
   macro avg       0.78      0.78      0.78      3704
weighted avg       0.78      0.78      0.78      3704
```

In [10]:
```
# for sensitivity and specificity
tn, fp, fn, tp = cm.ravel()
```

In [11]:
```
specificity = tn / (tn+fp)
print('specificity for data is : ',round(specificity*100,2),'%')
```

```
specificity for data is :  79.9 %
```

In [12]:
```
sensitivity = tp / (tp+fn)
print('sensitivity for data is : ',round(sensitivity*100,2),'%')
```

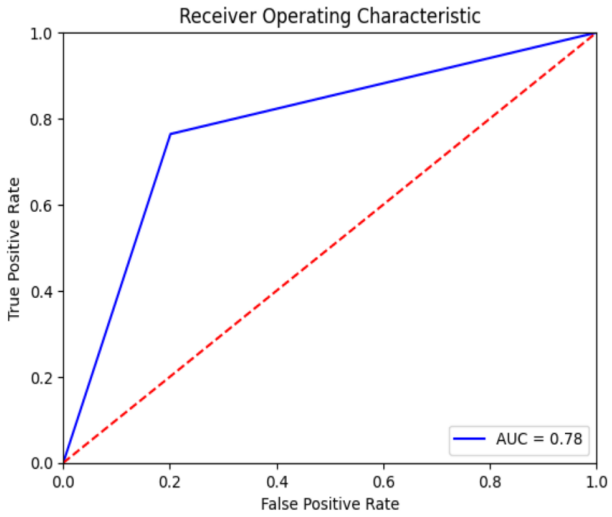
```
sensitivity for data is :  76.42 %
```

In [13]:
```
#ROC and AUC
fpr, tpr, threshold = roc_curve(Y_test,Y_pred)
roc_auc = auc(fpr, tpr)
```

In [14]:
```
print('Area under curve : ',round(roc_auc*100,2),'%')
```

```
Area under curve :  78.16 %
```

In [15]:
```
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```



Code for implementing ANN Model.

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import Dense
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score , recall_score , f1_score , confusion_matrix
from sklearn.metrics import roc_curve , auc
import matplotlib.pyplot as plt
import seaborn as snb
```

In [5]:
```
# Create a neural network model
model = Sequential()
model.add(Dense(16, input_dim=X_train.shape[1], activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

In [6]:
```
# Compile the model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

In [7]:
```
# Train the model on the training set
model.fit(X_train, y_train, epochs=100, batch_size=32)
```

```
Epoch 3/100
463/463 [==============================] - 1s 2ms/step - loss: 0.7170 - accuracy: 0.5289
Epoch 4/100
463/463 [==============================] - 1s 2ms/step - loss: 0.7097 - accuracy: 0.5351
Epoch 5/100
463/463 [==============================] - 1s 2ms/step - loss: 0.7118 - accuracy: 0.5353
Epoch 6/100
463/463 [==============================] - 1s 2ms/step - loss: 0.7262 - accuracy: 0.5360
Epoch 7/100
463/463 [==============================] - 1s 2ms/step - loss: 0.7176 - accuracy: 0.5412
Epoch 8/100
463/463 [==============================] - 1s 2ms/step - loss: 0.7122 - accuracy: 0.5436
Epoch 9/100
463/463 [==============================] - 1s 2ms/step - loss: 0.7247 - accuracy: 0.5347
Epoch 10/100
463/463 [==============================] - 1s 2ms/step - loss: 0.7170 - accuracy: 0.5417
Epoch 11/100
463/463 [==============================] - 1s 2ms/step - loss: 0.7136 - accuracy: 0.5453
Epoch 12/100
463/463 [==============================] - 1s 2ms/step - loss: 0.7211 - accuracy: 0.5396
```

In [8]:
```
# Use the trained model to make predictions on the testing set
y_pred = model.predict(X_test)
y_pred = np.round(y_pred)
```

```
116/116 [==============================] - 0s 1ms/step
```

In [9]:
```
y_pred2 = model.predict(X_train)
y_pred2 = np.round(y_pred)
```

```
463/463 [==============================] - 1s 1ms/step
```

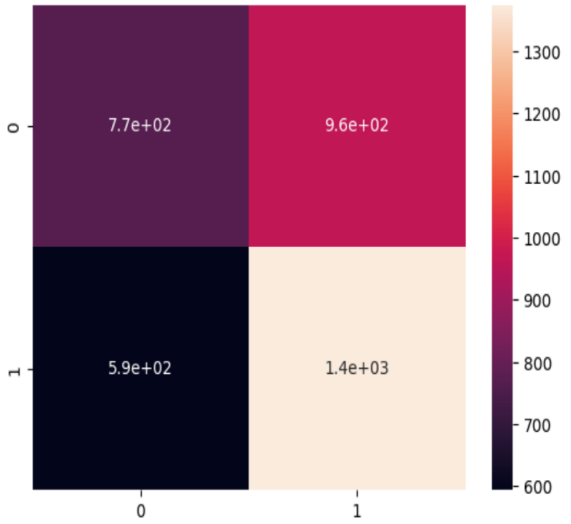
In [10]:
```
# Evaluate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", round(accuracy*100,2),'%')
```

```
Accuracy: 57.91 %
```

In [11]:
```
cm = confusion_matrix(y_test, y_pred)
```

In [12]:
```
snb.heatmap(cm,annot=True)
cm
```

Out[12]:
```
array([[ 770,  965],
       [ 594, 1375]], dtype=int64)
```



In [13]:
```
# precision for testing
print('for testing : ' , round(precision_score(y_test , y_pred)*100,2),'%')
```

```
for testing :  58.76 %
```

In [14]:
```
# Recall for testing
print('for testing : ' , round(recall_score(y_test , y_pred)*100,2),'%')
```

```
for testing :  69.83 %
```

In [15]:
```
# f1_score for testing
print('for testing : ' , round(f1_score(y_test , y_pred)*100,2),'%')
```

```
for testing :  63.82 %
```

In [16]:
```
# for sensitivity and specificity
tn, fp, fn, tp = cm.ravel()
```

In [17]:
```
specificity = tn / (tn+fp)
print('specificity for data is : ',round(specificity*100,2),'%')
```

```
specificity for data is :  44.38 %
```

In [18]:
```
sensitivity = tp / (tp+fn)
print('sensitivity for data is : ',round(sensitivity*100,2),'%')
```

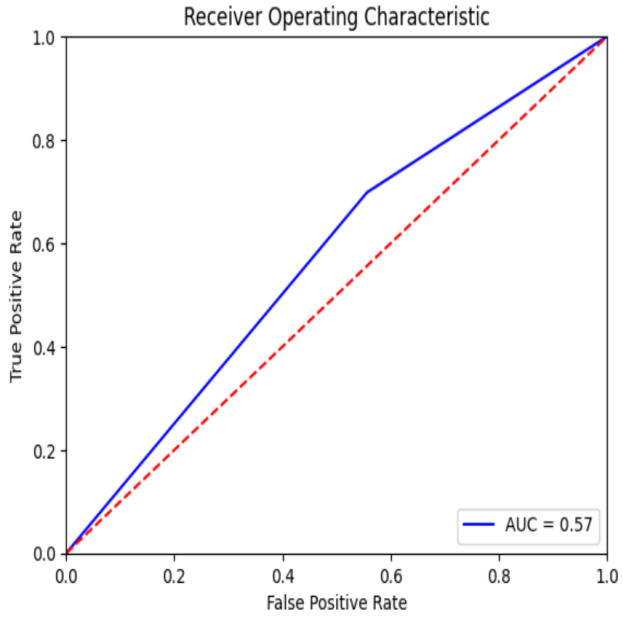
```
sensitivity for data is :  69.83 %
```

In [19]:
```
#ROC and AUC
fpr, tpr, threshold = roc_curve(y_test,y_pred)
roc_auc = auc(fpr, tpr)
```

In [20]:
```
print('Area under curve : ',round(roc_auc*100,2),'%')
```

```
Area under curve :  57.11 %
```

In [21]:
```
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```



Code for implementing ensemble Model (SVM-KNN).

In [1]:
```
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_score , recall_score , f1_score , confusion_matrix
from sklearn.metrics import roc_curve , auc
import matplotlib.pyplot as plt
import seaborn as snb
```

In [3]:
```
# Create SVM and KNN classifiers
svm = SVC(kernel='linear', C=1)
knn = KNeighborsClassifier(n_neighbors=5)
```

In [4]:
```
# Train classifiers on training data
svm.fit(X_train, y_train)
knn.fit(X_train, y_train)
```

Out[4]:
```
KNeighborsClassifier
KNeighborsClassifier()
```

In [5]:
```
# Make predictions on testing data
svm_preds = svm.predict(X_test)
knn_preds = knn.predict(X_test)
```

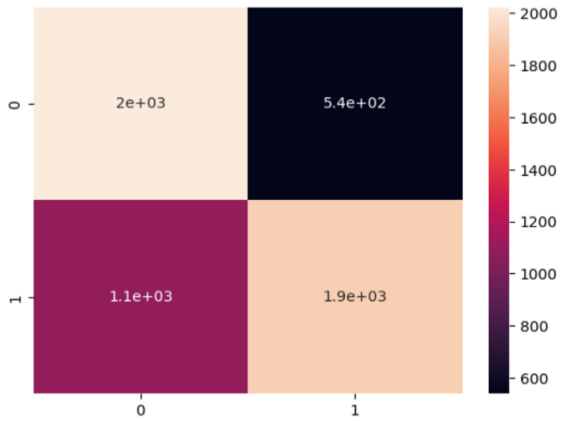
In [6]:
```
# Combine predictions using majority voting
ensemble_preds = []
for i in range(len(X_test)):
    if svm_preds[i] == knn_preds[i]:
        ensemble_preds.append(svm_preds[i])
    else:
        ensemble_preds.append(svm_preds[i])  # You can also use knn_preds here
```

In [7]:
```
# Calculate accuracy of ensemble predictions
ensemble_acc = accuracy_score(y_test, ensemble_preds)
print("Ensemble accuracy:", round(ensemble_acc*100,2),'%')
```

```
Ensemble accuracy: 70.72 %
```

In [8]:
```
cm = confusion_matrix(y_test, ensemble_preds)
snb.heatmap(cm,annot=True)
cm
```

Out[8]:
```
array([[2022,  539],
       [1088, 1907]], dtype=int64)
```



In [9]:
```
# precision for testing
print('for testing : ' , round(precision_score(y_test , ensemble_preds)*100,2),'%')
```

```
for testing :  77.96 %
```

In [10]:
```
# Recall for testing
print('for testing : ' , round(recall_score(y_test , ensemble_preds)*100,2),'%')
```

```
for testing :  63.67 %
```

In [11]:
```
# f1_score for testing
print('for testing : ' , round(f1_score(y_test , ensemble_preds)*100,2),'%')
```

```
for testing :  70.1 %
```

In [12]:
```
# for sensitivity and specificity
tn, fp, fn, tp = cm.ravel()
```

In [13]:
```
specificity = tn / (tn+fp)
print('specificity for data is : ',round(specificity*100,2),'%')
```

```
specificity for data is :  78.95 %
```

In [14]:
```
sensitivity = tp / (tp+fn)
print('sensitivity for data is : ',round(sensitivity*100,2),'%')
```

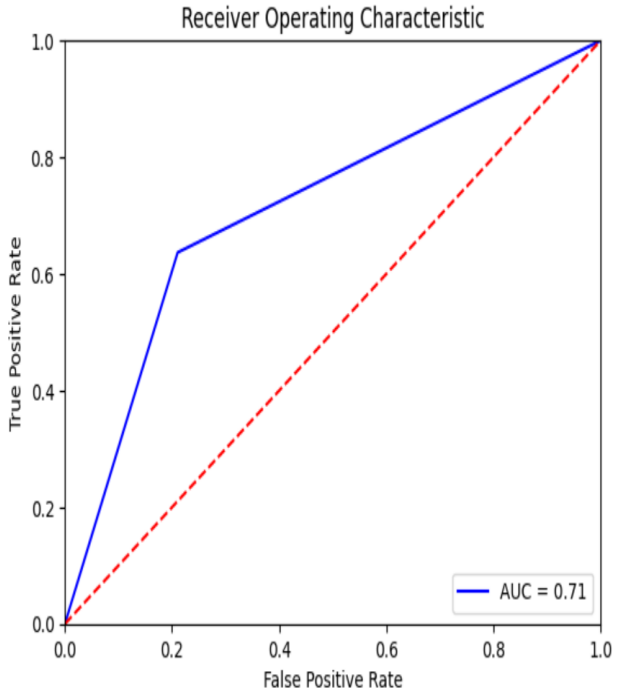
```
sensitivity for data is :  63.67 %
```

In [15]:
```
#ROC and AUC
fpr, tpr, threshold = roc_curve(y_test,ensemble_preds)
roc_auc = auc(fpr, tpr)
```

In [16]:
```
print('Area under curve : ',round(roc_auc*100,2),'%')
```

```
Area under curve :  71.31 %
```

In [17]:
```
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```



Code for implementing ensemble Model (KNN-RandomForest).

In [1]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_score , recall_score , f1_score , confusion_matrix
from sklearn.metrics import roc_curve , auc
import matplotlib.pyplot as plt
import seaborn as snb
```

In [5]:
```
# Create KNN and Random Forest classifiers
knn = KNeighborsClassifier(n_neighbors=5)
rf = RandomForestClassifier(n_estimators=100)
```

In [6]:
```
# Train classifiers on training data
knn.fit(X_train, y_train)
rf.fit(X_train, y_train)
```

Out[6]:
```
RandomForestClassifier
RandomForestClassifier()
```

In [7]:
```
# Make predictions on testing data
knn_preds = knn.predict(X_test)
rf_preds = rf.predict(X_test)
```

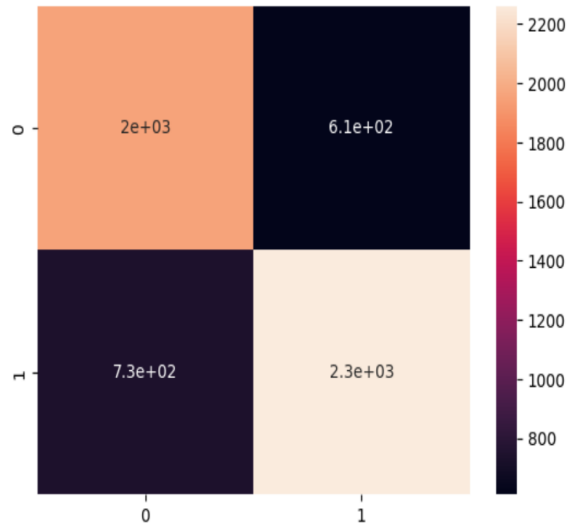
In [8]:
```
# Combine predictions using majority voting
ensemble_preds = []
for i in range(len(X_test)):
    if knn_preds[i] == rf_preds[i]:
        ensemble_preds.append(knn_preds[i])
    else:
        ensemble_preds.append(knn_preds[i])  # You can also use rf_preds here
```

In [9]:
```
# Calculate accuracy of ensemble predictions
ensemble_acc = accuracy_score(y_test, ensemble_preds)
print("Ensemble accuracy:", round(ensemble_acc*100,2),'%')
```

```
Ensemble accuracy: 75.83 %
```

In [10]:
```
cm = confusion_matrix(y_test, ensemble_preds)
snb.heatmap(cm,annot=True)
cm
```

Out[10]:
```
array([[1952,  609],
       [ 734, 2261]], dtype=int64)
```



In [11]:
```
# precision for testing
print('for testing : ' , round(precision_score(y_test , ensemble_preds)*100,2),'%')
```

```
for testing :  78.78 %
```

In [12]:
```
# Recall for testing
print('for testing : ' , round(recall_score(y_test , ensemble_preds)*100,2),'%')
```

```
for testing :  75.49 %
```

In [13]:
```
# f1_score for testing
print('for testing : ' , round(f1_score(y_test , ensemble_preds)*100,2),'%')
```

```
for testing :  77.1 %
```

In [14]:
```
# for sensitivity and specificity
tn, fp, fn, tp = cm.ravel()
```

In [15]:
```
specificity = tn / (tn+fp)
print('specificity for data is : ',round(specificity*100,2),'%')
```

```
specificity for data is :  76.22 %
```

In [16]:
```
sensitivity = tp / (tp+fn)
print('sensitivity for data is : ',round(sensitivity*100,2),'%')
```

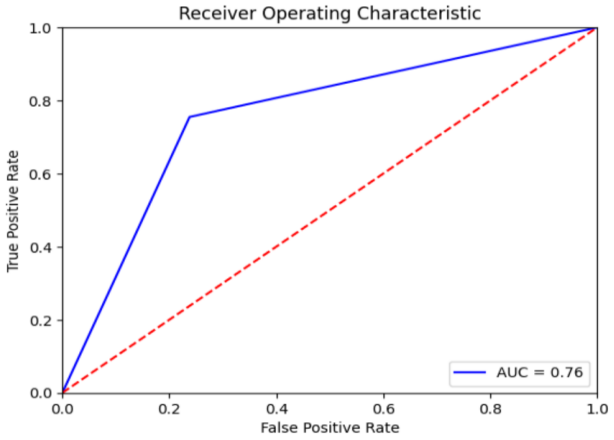
```
sensitivity for data is :  75.49 %
```

In [17]:
```
#ROC and AUC
fpr, tpr, threshold = roc_curve(y_test,ensemble_preds)
roc_auc = auc(fpr, tpr)
```

In [18]:
```
print('Area under curve : ',round(roc_auc*100,2),'%')
```

```
Area under curve :  75.86 %
```

In [19]:
```
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```



Code for implementing ensemble Model (RandomForest-SVM).

In [1]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_score , recall_score , f1_score , confusion_matrix
from sklearn.metrics import roc_curve , auc
import matplotlib.pyplot as plt
import seaborn as snb
```

In [5]:
```
# Create Random Forest and SVM classifiers
rf = RandomForestClassifier(n_estimators=100)
svm = SVC(kernel='linear', C=1)
```

In [6]:
```
# Train classifiers on training data
rf.fit(X_train, y_train)
svm.fit(X_train, y_train)
```

Out[6]:
```
SVC(C=1, kernel='linear')
```

In [7]:
```
# Make predictions on testing data
rf_preds = rf.predict(X_test)
svm_preds = svm.predict(X_test)
```

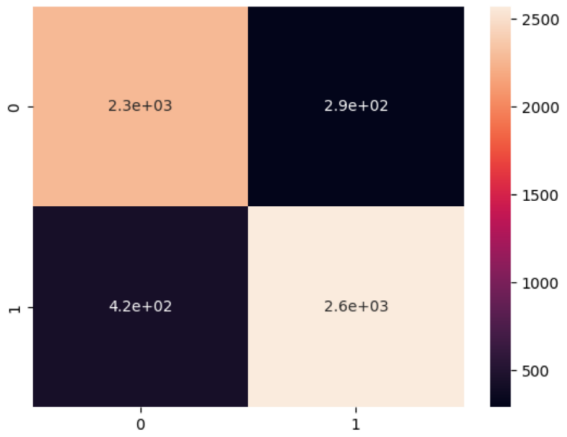
In [8]:
```
# Combine predictions using majority voting
ensemble_preds = []
for i in range(len(X_test)):
    if rf_preds[i] == svm_preds[i]:
        ensemble_preds.append(rf_preds[i])
    else:
        ensemble_preds.append(rf_preds[i])  # You can also use svm_preds here
```

In [9]:
```
# Calculate accuracy of ensemble predictions
ensemble_acc = accuracy_score(y_test, ensemble_preds)
print("Ensemble accuracy:", round(ensemble_acc*100,2),'%')
```

```
Ensemble accuracy: 87.24 %
```

In [10]:
```
cm = confusion_matrix(y_test, ensemble_preds)
snb.heatmap(cm,annot=True)
cm
```

Out[10]:
```
array([[2274,  287],
       [ 422, 2573]], dtype=int64)
```



In [11]:
```
# precision for testing
print('for testing : ' , round(precision_score(y_test , ensemble_preds)*100,2),'%')
```

```
for testing :  89.97 %
```

In [12]:
```
# Recall for testing
print('for testing : ' , round(recall_score(y_test , ensemble_preds)*100,2),'%')
```

```
for testing :  85.91 %
```

In [13]:
```
# f1_score for testing
print('for testing : ' , round(f1_score(y_test , ensemble_preds)*100,2),'%')
```

```
for testing :  87.89 %
```

In [14]:
```
# for sensitivity and specificity
tn, fp, fn, tp = cm.ravel()
```

In [15]:
```
specificity = tn / (tn+fp)
print('specificity for data is : ',round(specificity*100,2),'%')
```

```
specificity for data is :  88.79 %
```

In [16]:
```
sensitivity = tp / (tp+fn)
print('sensitivity for data is : ',round(sensitivity*100,2),'%')
```

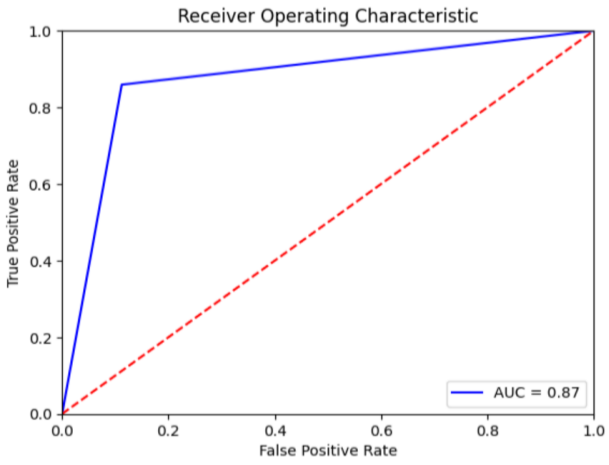
```
sensitivity for data is :  85.91 %
```

In [17]:
```
#ROC and AUC
fpr, tpr, threshold = roc_curve(y_test,ensemble_preds)
roc_auc = auc(fpr, tpr)
```

In [18]:
```
print('Area under curve : ',round(roc_auc*100,2),'%')
```

```
Area under curve :  87.35 %
```

In [19]:
```
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```



Code for implementing ensemble Model (RandomForest-ANN).

In [1]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_score , recall_score , f1_score , confusion_matrix
from sklearn.metrics import roc_curve , auc
import matplotlib.pyplot as plt
import seaborn as snb
```

In [5]:
```
# Create Random Forest and ANN classifiers
rf = RandomForestClassifier(n_estimators=100)
ann = MLPClassifier(hidden_layer_sizes=(100, 50), activation='relu', solver='adam')
```

In [6]:
```
# Train classifiers on training data
rf.fit(X_train, y_train)
ann.fit(X_train, y_train)
```

Out[6]:
```
MLPClassifier(hidden_layer_sizes=(100, 50))
```

In [7]:
```
# Make predictions on testing data
rf_preds = rf.predict(X_test)
ann_preds = ann.predict(X_test)
```

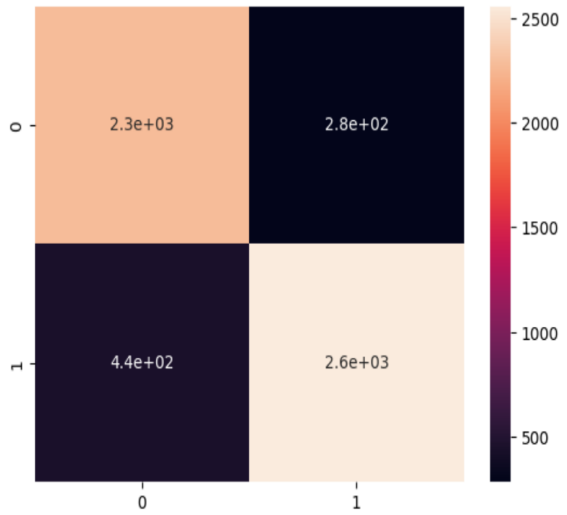
In [8]:
```
# Combine predictions using majority voting
ensemble_preds = []
for i in range(len(X_test)):
    if rf_preds[i] == ann_preds[i]:
        ensemble_preds.append(rf_preds[i])
    else:
        ensemble_preds.append(rf_preds[i])  # You can also use ann_preds here
```

In [9]:
```
# Calculate accuracy of ensemble predictions
ensemble_acc = accuracy_score(y_test, ensemble_preds)
print("Ensemble accuracy:", round(ensemble_acc*100,2),'%')
```

```
Ensemble accuracy: 87.06 %
```

In [10]:
```
cm = confusion_matrix(y_test, ensemble_preds)
snb.heatmap(cm,annot=True)
cm
```

Out[10]:
```
array([[2279,  282],
       [ 437, 2558]], dtype=int64)
```



In [11]:
```
# precision for testing
print('for testing : ' , round(precision_score(y_test , ensemble_preds)*100,2),'%')
```

```
for testing :  90.07 %
```

In [12]:
```
# Recall for testing
print('for testing : ' , round(recall_score(y_test , ensemble_preds)*100,2),'%')
```

```
for testing :  85.41 %
```

In [13]:
```
# f1_score for testing
print('for testing : ' , round(f1_score(y_test , ensemble_preds)*100,2),'%')
```

```
for testing :  87.68 %
```

In [14]:
```
# for sensitivity and specificity
tn, fp, fn, tp = cm.ravel()
```

In [15]:
```
specificity = tn / (tn+fp)
print('specificity for data is : ',round(specificity*100,2),'%')
```

```
specificity for data is :  88.99 %
```

In [16]:
```
sensitivity = tp / (tp+fn)
print('sensitivity for data is : ',round(sensitivity*100,2),'%')
```

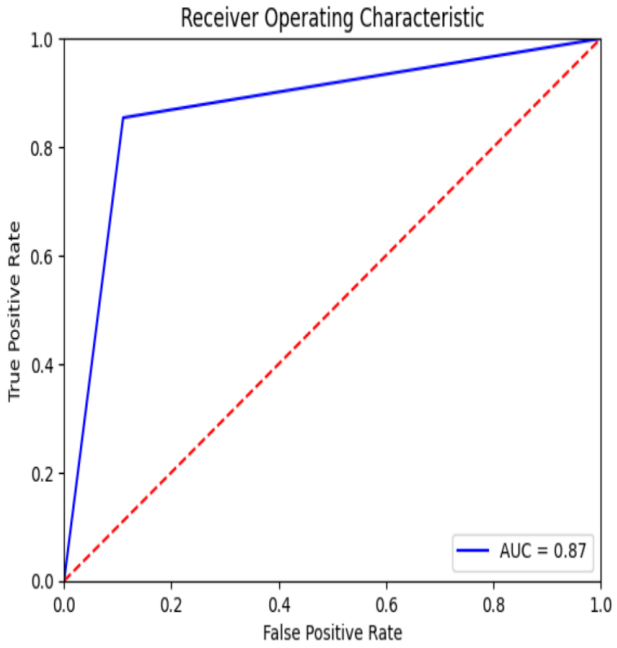
```
sensitivity for data is :  85.41 %
```

In [17]:
```
#ROC and AUC
fpr, tpr, threshold = roc_curve(y_test,ensemble_preds)
roc_auc = auc(fpr, tpr)
```

In [18]:
```
print('Area under curve : ',round(roc_auc*100,2),'%')
```

```
Area under curve :  87.2 %
```

In [19]:
```
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```



Code for implementing ensemble Model (KNN-ANN).

In [1]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_score , recall_score , f1_score , confusion_matrix
from sklearn.metrics import roc_curve , auc
import matplotlib.pyplot as plt
import seaborn as snb
```

In [4]:
```
# Train classifiers on training data
knn.fit(X_train, y_train)
ann.fit(X_train, y_train)
```

Out[4]:
```
MLPClassifier(hidden_layer_sizes=(100, 50))
```

In [5]:
```
# Make predictions on testing data
knn_preds = knn.predict(X_test)
ann_preds = ann.predict(X_test)
```

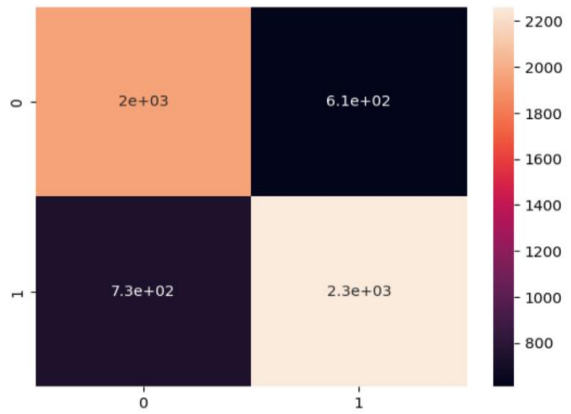
In [6]:
```
# Combine predictions using majority voting
ensemble_preds = []
for i in range(len(X_test)):
    if knn_preds[i] == ann_preds[i]:
        ensemble_preds.append(knn_preds[i])
    else:
        ensemble_preds.append(knn_preds[i])  # You can also use ann_preds here
```

In [7]:
```
# Calculate accuracy of ensemble predictions
ensemble_acc = accuracy_score(y_test, ensemble_preds)
print("Ensemble accuracy:", round(ensemble_acc*100,2),'%')
```

```
Ensemble accuracy: 75.83 %
```

In [8]:
```
cm = confusion_matrix(y_test, ensemble_preds)
snb.heatmap(cm,annot=True)
cm
```

Out[8]:
```
array([[1952,  609],
       [ 734, 2261]], dtype=int64)
```



In [9]:
```
# precision for testing
print('for testing : ' , round(precision_score(y_test , ensemble_preds)*100,2),'%')
```

```
for testing :  78.78 %
```

In [10]:
```
# Recall for testing
print('for testing : ' , round(recall_score(y_test , ensemble_preds)*100,2),'%')
```

```
for testing :  75.49 %
```

In [11]:
```
# f1_score for testing
print('for testing : ' , round(f1_score(y_test , ensemble_preds)*100,2),'%')
```

```
for testing :  77.1 %
```

In [12]:
```
# for sensitivity and specificity
tn, fp, fn, tp = cm.ravel()
```

In [13]:
```
specificity = tn / (tn+fp)
print('specificity for data is : ',round(specificity*100,2),'%')
```

```
specificity for data is :  76.22 %
```

In [14]:
```
sensitivity = tp / (tp+fn)
print('sensitivity for data is : ',round(sensitivity*100,2),'%')
```

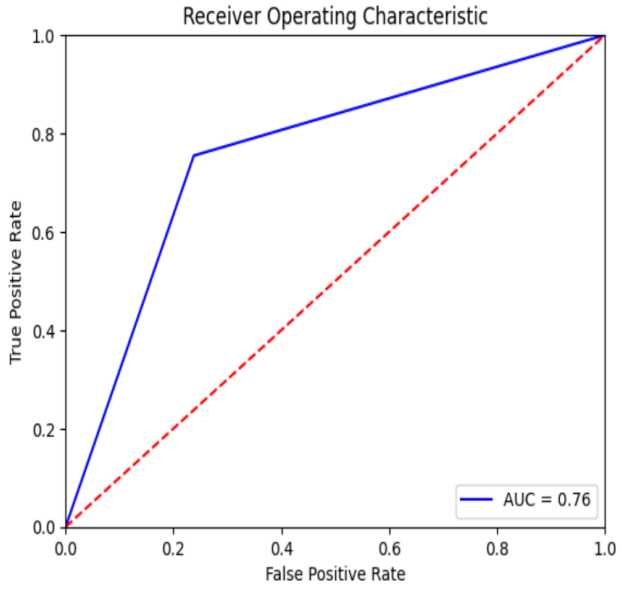
```
sensitivity for data is :  75.49 %
```

In [15]:
```
#ROC and AUC
fpr, tpr, threshold = roc_curve(y_test,ensemble_preds)
roc_auc = auc(fpr, tpr)
```

In [16]:
```
print('Area under curve : ',round(roc_auc*100,2),'%')
```

```
Area under curve :  75.86 %
```

In [17]:
```
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```



Code for ROC Curve of all Model Trained for influenza detection.

In [157]:
```
fig, ax = plt.subplots(1, figsize=(8, 6))
fig.suptitle('ROC curve', fontsize=15)
ax.plot(fpr_rfknn, tpr_rfknn, 'b',linestyle = '-', label = 'RF-KNN = %0.2f' % roc_auc_rfknn)
ax.plot(fpr_rfsvm, tpr_rfsvm, 'r',linestyle = '--', label = 'RF-SVM = %0.2f' % roc_auc_rfsvm)
ax.plot(fpr_rfann, tpr_rfann, 'cyan',linestyle = 'dotted', label = 'RF-ANN = %0.2f' % roc_auc_rfann)
ax.plot(fpr_knnann, tpr_knnann, 'c',linestyle = 'dashdot', label = 'KNN-ANN = %0.2f' % roc_auc_knnann)
ax.plot(fpr_svmknn, tpr_svmknn, 'k',linestyle = '--', label = 'SVM-KNN = %0.2f' % roc_auc_svmknn)
ax.plot(fpr_knn, tpr_knn, 'm',linestyle = ':', label = 'KNN = %0.2f' % roc_auc_knn)
ax.plot(fpr_rf,tpr_rf, 'b', linestyle = 'solid',label = 'RF = %0.2f' % roc_auc_rf)
ax.plot(fpr_ann, tpr_ann, 'pink', linestyle = '-',label = 'ANN = %0.2f' % roc_auc_ann)
ax.plot(fpr_svm, tpr_svm, 'g',linestyle = '--', label = 'SVM = %0.2f' % roc_auc_svm)
plt.legend(loc="lower right", title="Legend Title", frameon=False)
ax.set_xlabel('Sensitivity')
ax.set_ylabel('Specificity')
plt.show()
```