

Project Report
on
Animal Disease Prediction
Using Machine Learning techniques
Submitted to
Amity University, Uttar Pradesh



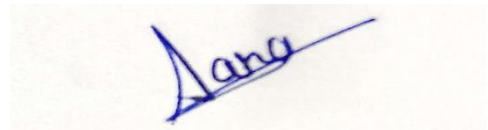
In partial fulfilment of the requirements for the award of the degree of
Bachelor of Technology
in
Computer Science and Engineering
By
Sana Rehman(A2305219623)
Bhanushikha Rathore(A2305219617)
Under the guidance of
Mr. Roshan Lal

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
AMITY SCHOOL OF ENGINEERING AND TECHNOLOGY
AMITY UNIVERSITY, UTTAR PRADESH

DECLARATION

I Sana Rehman student of B.Tech (7CSE9) hereby declare that the project titled "Animal Disease Outbreak Prediction" which is submitted by us to Department of Computer Science and Engineering, Amity School of Engineering and Technology. Amity University Uttar Pradesh, Noida, in partial fulfilment of requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering has not been previously formed the basis for the award of any degree, diploma or other similar title or recognition.

Date: 17/05/23



Name and Signature

Sana Rehman

(A2305219623)

Name and Signature

Bhanushikha Rathore

(A2305219617)

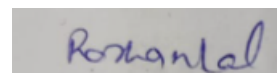
CERTIFICATE

On the basis of declaration submitted by SANA REHMAN student of B. Tech (CSE)-9Y, I hereby certify that the project titled "Animal Disease Prediction using ML techniques" which is submitted to Department of Computer Science and Engineering, Amity School of Engineering and Technology, Amity University Uttar Pradesh, Noida, in partial fulfilment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering, is an original contribution with existing knowledge and faithful record of work carried out by him/her under my guidance and supervision.

To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Noida

Date: 17/05/23



Mr. Roshan Lal

Assistant Professor-III

Department of Computer Science and Engineering

Amity School of Engineering and Technology

Amity University Uttar Pradesh

ACKNOWLEDGEMENT

It is a high privilege for me to express my deep sense of gratitude to those entire faculty members who helped me in the completion of the project, especially my internal guide who was always there in need.

My special thanks to all other faculty members, batchmates & seniors of Amity School of Engineering and Technology, Amity University Uttar Pradesh for helping me in the completion of project work and its report submission.

Sana Rehman
(A2305219623)

Bhanushikha Rathore
(A2305219617)

ABSTRACT

A great uprising has been witnessed in numerous Animal Diseases in the past many years. Many of these diseases have the tendency to transform into zoonotic diseases which can turn out to be very infectious and impact both animals as well as humans. Machine Learning is the field of study that deals with making machines/computers learn on their own so that further predictions can be made for varied Applications. Human disease detection using Machine Learning Techniques has been there from quite a while but very few advancements have been made for Animal Diseases.

Through this research paper we make a new contribution in the aforementioned field by deploying ML techniques to predict certain Animal Diseases along with predicting the spread of the disease. Animal disease when turn into Zoonosis can have a huge scale impact on both Human and Animal species. So, through this project we also used few techniques to measure the level of zoonosis in such diseases. All the results are well formulated. To assess all possible number of zoonotic disease outbreaks related with animal exhibits and identify published suggestions for preventing zoonotic disease transmission from animals to people in exhibit settings.

Zoonotic illnesses account for a sizeable portion of infectious ailment outbreaks and burden on public health packages to keep surveillance and preventative measures. To facilitate facts collection, evaluation, and choice-making, the wide variety of spatial selection support systems said within the ultimate 10 years has multiplied. This systematic review objectives to describe characteristics of spatial decision support structures evolved to help public fitness officials within the management of zoonotic disease outbreaks.

INDEX

	Page No.
1. Introduction.....	1-2
2. Related Work.....	3
3. Research Methodology.....	4
4. About the dataset.....	5
5. Data Exploration.....	6
6. ML Models.....	7-17
6.1 Linear regression	
6.2 Logistic Regression	
6.3 Support Vector Machine	
6.4 Decision Trees	
6.5 Random Forest	
6.6 Naïve Bayes	
6.7 XG-Boost	
6.8 ANN (Artificial Neural network)	
6.9 CNN (Convolution Neural Network)	
7. Implementation and Results.....	18-40
7.1 linear regression to predict the outbreak of animal deaths as per the cases observed	
7.2 Logistic regression and Support vector machine to predict human deaths as per the affected cases	
7.3 Implementation of XGBoost	
7.4 Animal Disease classification using different Algorithms.	
7.5 CNN to predict if Disease Zoonotic or not	

8. Limitations.....	41
9. Future Work.....	42
10.Conclusion.....	43

FIGURES AND TABLES

List of figures

Figure 1: Top 4 most common disease across the world

Figure 2: Top 4 species affected by diseases across the world

Figure 3: Representing the demographic distribution of humans and how they are affected by Zoonotic diseases.

Figure 4: Research methodology.

Figure 5: SVM Hyperplane illustration

Figure 6: XGBoost algorithm illustration

Figure 7: ANN layers illustration

Figure 8: Backpropagation work flow.

Figure 9: Regression line obtained by implementing linear regression:
Describing the linear relationship between number of deaths and number of cases

Figure 10: Confusion matrix of logistic regression predicting the human deaths

Figure 11: Confusion matrix of SVM in predicting the human deaths

Figure 12: SVM for predicting human deaths for testing dataset as per the affected cases to measure the severity of zoonosis

Figure 13: Comparison between Logistic Regression and SVM to Predict human deaths as per the affected cases to measure the severity of zoonosis in those diseases.

Figure 14: Depicts the transformation of dataset into Boolean column to fetch in XGBoost.

Figure 15: Data transformation for XGBoost

Figure 16: XGBoost results monitoring

Figure 17: Examination of XGBoost model

Figure 18: Results from XGBoost

Figure 19: Performance comparison for different techniques in classifying the disease.

Figure 20: Proposed architecture for our CNN model

List of tables

Table 1: Performance Measures: The below table contains the performance summary of the above-mentioned algorithms and their specific purpose.

Table 2: Performance results for naïve model used in XGBoost

Table 3: Performance results for XGBoost model after tweaking some parameters.

Table 4: Performance Measures of algorithms in classifying animal disease asper the observed symptoms

Table 5: Confusion matrix for CNN Classifier

Table 6: Additional Performance Measure for CNN classifier

1. INTRODUCTION

Within the past so many years we have witnessed a great uprising in the spread of numerous infectious diseases both in animals and humans. In addition to that, spread of zoonotic diseases also serves as a great threat to both the species. The outbreak of COVID-19 is a perfect example of how can zoonotic diseases can leave such a large-scale devastating impact on the world. Apart from COVID, diseases like bird flu and swine fever have caused so many deaths in past years. With all the aforementioned reasons, it has become imperial for Healthcare Industry to deploy systems that can predict the outbreak of such deadly diseases so that we can be prepared for any pandemic like situation or at least have strong combat mechanism to fight against them.

Machine learning algorithms have become effective tools for understanding and forecasting zoonotic diseases, which are illnesses that can spread from animals to people. These methods make use of the enormous amounts of data that are accessible from numerous sources, including records of animal and human health, environmental conditions, and genetic information. Machine learning algorithms can find patterns and relationships in these complicated datasets that would not be seen using conventional statistical techniques.

Supervised learning is a well-known machine learning technique for forecasting zoonotic illnesses, where models are trained on labelled data to identify patterns and generate predictions. For instance, researchers can train algorithms to recognise early warning signs and possible hotspots for disease transmission using historical data on zoonotic disease outbreaks and associated factors.

Unsupervised learning is a different method that involves finding structures and patterns in data without the aid of labels. Large datasets can contain hidden links or clusters that unsupervised learning algorithms can help reveal, providing researchers with the opportunity to find possible risk factors for the spread of zoonotic diseases. Unsupervised learning can offer important insights into the intricate dynamics of zoonotic illnesses by examining the relationships between many variables, such as animal migration patterns, ecological interactions, and environmental factors.

By improving decision-making procedures, reinforcement learning can also aid in the prediction of zoonotic diseases. In this method, an algorithm is trained to make judgements sequentially based on input from the environment. For instance, resource allocation or vaccination plans can be optimised using reinforcement learning models in the event of an outbreak.

Through this paper we attempt to use various Machine Learning techniques in order to predict the outbreaks of such diseases. There are several models that help in predicting chronic diseases in human like Heart Diseases, Diabetes etcetera, however very less development has been made for animal diseases in this area. But, especially after COVID has struck the world its high time that we start monitoring Animal Diseases that have the tendency to transform into zoonosis.

2. RELATED WORK

‘Ensemble Approach for Zoonotic Disease Prediction Using Machine Learning Techniques ’Authors: Rama Krishna Singh and Vikash Chandra Sharma compares the efficiency of traditional techniques with that of Machine Learning Techniques in predicting the impact of Zoonotic Diseases. [1]

Authors compare two important Cluster Analysis techniques namely ANN and K-means clustering for Dairy Cattle breed. [2]

Authors talk about how Machine Learning can help in animal epidemic situation, followed by application of machine learning in animal disease analysis and prediction. [3]

Paper points out how farm animal movement can be an influential factor for the spread of disease in animals at a faster rate also since the same animals are used as means of food for humans it is a possibility the infections can transform into zoonosis. The paper uses techniques like random forest to compute the probability of swine movements in two regions. [4]

In another paper a multi-scale big data-model integration approach was applied using human-guided machine learning to develop EWS for vesicular stomatitis (VS), a widespread viral vectorborne vesicular disease affecting livestock throughout the Americas. The goal is to objectively assess the significance of a large suite of spatially distributed environmental variables (>400). [6]

3. RESEARCH METHODOLOGY

The aim through this research is to first carry out an in-depth analysis of the gathered data. Secondly, we will deploy machine learning techniques to perform the following tasks:

- i. Firstly, in order to justify our project, we wanted to investigate on the fact that how many Animals all over the world are dying by a particular disease. If the relation between number of cases and number of deaths is linear, this means that we need to have more robust practices regarding sustaining Animal's health. So, we used Linear regression to predict the number of animal deaths caused by the specified disease.
- ii. The second most important aim of our project is to monitor the diseases which has the tendency to transform into zoonotic. As a part of this, we first will drive a comparative analysis between two important techniques to predict human deaths as per the affected cases to measure the severity of zoonosis in those diseases.
- iii. After obtaining knowledge on what all diseases have tendency to be zoonotic, we will check that which species of animal is most likely to spread diseases to humans. For this purpose, we have introduced XGBosst algorithm which is discussed in detail in the further sections of this paper.
- iv. To expand our project scope, we further have taken a new dataset which was extracted from WOAHA (World organisation of animal Health). Further we used the dataset to classify the Animal Disease on the basis of the symptoms observed in them. Logistic Regression, Naïve Byes, Decision tree, Random Forest, SVM, ANN were the few techniques used for this purpose.
- v. Lastly, we will use CNN to predict if the disease is zoonotic or not on the basis of symptoms of the disease.

4. ABOUT THE DATASET

The first dataset is gathered from EMPRES Global Animal Disease Information System which is run by food and agricultural organization of united nations. It maintains the record of various animal diseases and its distribution across the world. The dataset contains 24 features in total where disease, number of cases and deaths are few of them.

The second dataset was extracted from WOAHA (world organisation of animal health) which has over 17000 entries and contains the records of over 63 diseases observed across different species of animals. This dataset contains 38 columns in total which corresponds to the varied features of dataset. 35 of these features represents different symptoms observed for different diseases, they are categorical in nature i.e., if 'x' disease has 'y' symptom, so the value will be '1' in that particular cell for the corresponding disease and value will be '0', if 'y' not a symptom of 'x'. The rest three columns contain details regarding disease name, species in which disease is found and if disease is zoonotic or not respectively.

5. DATA EXPLORATION:

Before implementing ML Models, we explored the dataset to understand the parameters and figure out some meaningful trends in the dataset.

- Firstly, we studied the correlation between different variables using correlation matrix and scatter plots for both datasets.
- Next, we observed the top 4 most common diseases in terms of the number of cases observed:

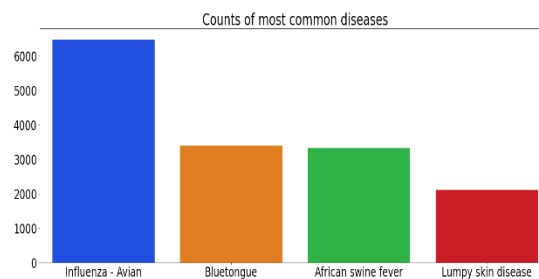


Fig. 1

- We then observed the top 4 species that are most affected by the diseases worldwide:

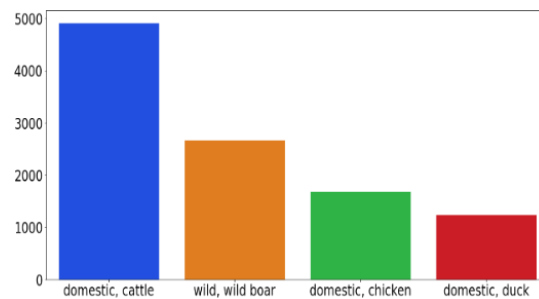


Fig. 2

- Age distribution of humans affected by zoonotic diseases:

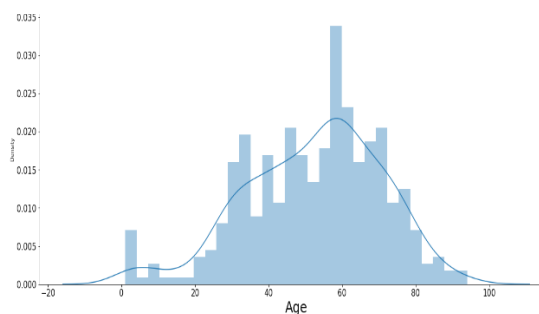


Fig. 3

6. ML MODELS

Machine Learning is the field of study which deals with making computers learn and perform tasks without explicitly programming them. When it comes to building a ML model, it basically consists of two phases training and testing. In training phase, we make the machine learn using training data, once trained, the model is tested against the new data so as to measure its performance. Machine learning is further broadly categorized into supervised and unsupervised. In supervised machine learning the model is trained using labelled dataset whereas in unsupervised the model itself learns by recognizing patterns in the data.

Below is the representation of how the models will be built and flow chart of entire process:

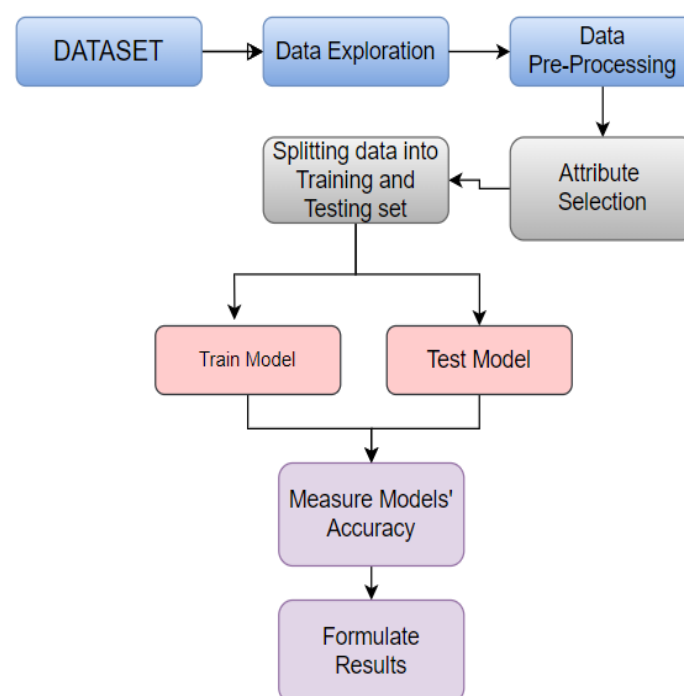


Fig. 4 Representing the work flow and methodology adapted in this project

In the upcoming section we will discuss all the ML models that have been used as a part of our Research.

6.1 Linear Regression:

It is one of the most important machine learning algorithms which is used to perform prediction functions. Linear regression deals with continuous form of data like a person's age or employee's salary. Linear regression basically studies and observes the relationship between two different variables namely dependent or target variable i.e., the parameter that we predict using the independent variable. This relationship is linear in nature i.e., the graph has a straight line. The variables can either be positively linearly related or negatively linearly related. The equation of this straight line is given as follows:

$$Y=mx +c \quad (1)$$

Where 'Y 'is the target variable 'x 'is the independent variable 'c 'is the intercept of line and 'm 'is the slope of line.

The aim of linear regression is to find a regression line which best fits the data such that the difference between actual values and predicted values is as minimum as possible. This aforementioned task is performed by using a cost function which measures the accuracy that how well the input variables are mapped to the output variable.

6.2 Logistic Regression

It is another most important supervised machine learning algorithm. Unlike Linear Regression, logistic regression deals with categorical kind of data. The target variable has to be categorical in nature i.e., it should attain a discrete value like 'Yes 'and 'No 'or '0 'and '1'. Logistic regression makes use of Sigmoid function to predict the values of output variable and map those values between '0 'and '1 'so as to classify which class the output belongs to. In multinomial and ordinal logistic regression, we can have more than 2 classes as well.

6.3 Support Vector Machine (SVM)

This algorithm is another popular supervised machine learning algorithm which is used for both regression and classification problems but mostly for the latter. The goal is to find a decision boundary which separates the data into different classes. The line which is able to segregate the data most efficiently is known as hyperplane. SVM sets two extreme values to create this hyperplane which are known as support vectors.

The SVM equation is:

Given a set of training data:

$$\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$$

where x_i is the input data and y_i is the class label (+1 or -1):

- First, transform the input data to a higher dimensional space using the nonlinear mapping function $\phi(x)$ so that the data is hyperplane separable.
- Next, find the hyperplane that maximizes the distance between the two classes. This can be expressed as the following optimization problem:

Maximize ρ considering

$$y_i(w^T \phi(x_i) + b) \geq \rho$$

for all $i=1,2,\dots,n$ $\|w\| = 1$ where w is the weight vector, b is the bias, ρ is the margin and $\|w\|$ is the norm of w .

- Once we have found the optimal hyperplane, we can use it to predict the class of new data points by computing the sign of the function

$$f(x) = w^T \phi(x) + b \tag{2}$$

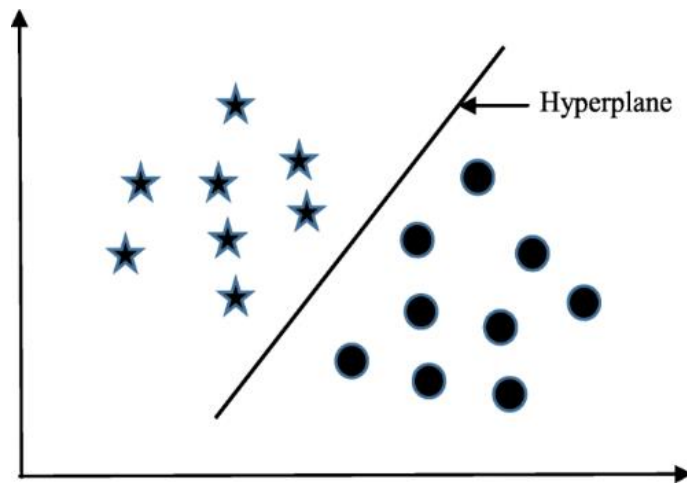


Fig. 5

6.4 Decision Trees

Decision trees are a well-known machine learning algorithm utilized for classification and regression purposes. A decision tree embodies a hierarchical structure wherein every internal node signifies a property test, each branch symbolizes a test result, and every leaf node signifies a class label or numerical value.

The basic idea behind decision trees is to split the data recursively depending on the values of the input features until all data points belong to the same class or yield pure nodes with the same numeric value. The tree is built top-down, starting at the root node and partitioning the data at each internal node based on features that maximize class separation.

Here are the steps involved in constructing a decision tree:

- Choose the function that works best for you:

The first step is to choose the best feature that divides the data into the purest subsets. This is done by computing the information gain or Gini index for each feature, which measures the amount of entropy or contamination in the data before and after splitting.

- Split the data.

Once the best features are selected, the data are divided into sub-sets based on the values of the selected features. Each subset is then used to create a new branch of the tree.

- Appeal:

The mentioned steps are repeated recursively for each subset until the stopping criteria are met. This criterion can be the maximum tree depth, the minimum number of data points per leaf node, or the minimum information gained.

- Forecast:

To predict the class or numeric value of a new data point, start at the root node and traverse the tree based on the values of the input features until you reach the leaf nodes. Classes or values associated with leaf nodes are then used as predictions.

Decision trees can steer both categorical and numeric data and can be utilized for both classification and regression tasks. They are easy to interpret and can capture complex decision boundaries. However, decision trees are prone to overfitting if not properly pruned, where small changes in data can lead to large changes in tree structure.

6.5 Random Forest:

Random forest is an ensemble learning method that enhances prediction accuracy and resilience by amalgamating multiple decision trees. In a random forest, numerous decision trees are trained using diverse subsets of the training data, and the ultimate prediction is derived by averaging the predictions of all the trees. Here are the steps involved in training a Random Forest:

- Bootstrap example:

Firstly, bootstrap samples are extracted from the original training data using a surrogate. This means that some data points in the bootstrap sample may appear multiple times and some may be omitted.

- Random feature selection:

For each tree, a random subset of features is chosen. This is done to ensure that each tree contains a diverse set of features and to reduce correlations between trees.

- Train the decision tree.

A decision tree is trained on the bootstrap samples using the selected traits. The tree grows until stopping criteria such as maximum tree depth or minimum number of data points per leaf node are met.

- Repeat above 3 steps several times to form a forest of decision trees.

- Forecast:

To make assumptions for new data points, all trees in the forest are used to make predictions, and the last prediction is obtained by averaging the predictions of all trees. The Random Forest algorithm is a modification of the standard Decision Tree algorithm. The splitting criterion for each node in the tree is estimated based on the Gini impurity or entropy of the data.

The splitting criterion for each node in the tree is intended on the Gini impurity or entropy of the data.

The Gini entropy is calculated using the following equations:

Gini impurity:

$$\text{Gini}(p) = 1 - \sum (p_i^2) \quad (3)$$

where p is the vector of class probabilities, and p_i is the probability of class 'i'.

Entropy:

$$H(p) = - \sum (p_i \log_2(p_i)) \quad (4)$$

where p is the vector of class probabilities, and p_i is the probability of class 'i'.

The Random Forest algorithm is a powerful machine learning algorithm that can handle both regression and classification problems. Random forests can be used to identify the most important features in a data set and handle missing data. Random forests typically perform better than single decision trees because they are less prone to overfitting and can capture more complex decision boundaries.

6.6 Naive Bayes

Naive Bayes is a probabilistic machine learning algorithm widely employed for classification tasks. It relies on Bayes' theorem, which establishes that the probability of a hypothesis (class) given evidence (input characteristics) is proportionate to the probability of the evidence given the hypothesis, multiplied by the prior probability of the hypothesis. The "naive" aspect of Naive Bayes lies in the assumption that the input features are conditionally independent when given the class labels. Consequently, the presence or absence of one attribute does not impact the likelihood of the presence or absence of another attribute, provided the class designation. This assumption simplifies the computation of conditional probabilities and enables the algorithm to effectively handle high-dimensional data.

Here are the steps involved in training a Naive Bayes classifier:

- Compute prior probabilities. The prior probability for each class is computed as the percentage of training examples belonging to that class.
- Calculate conditional probabilities. The conditional probability for each feature given a class is computed as the proportion of training examples that have that feature and belong to that class divided by the proportion of training examples.
- Forecast: To generate a prediction for a new data point, the algorithm calculates the probability of each class given the evidence (input features) by utilizing Bayes' theorem. Subsequently, the prediction is made by selecting the class with the highest probability among all the classes.

6.7 XGBoost:

XG Boost is an implemented of the Gradient Boosted Decision Trees algorithm. The algorithm works by going over cycles that builds a new model after end of every cycle and then all these models are combined together to form an ensemble model. Before the cycle we take a base model 'x' which is naïve in nature and use it to make predictions. The predictions can be incorrect in nature since model is not that robust yet. Then this base model is fetched into the boost cycle where it further calculates errors made in prediction for each observation. Then a new model 'y' also called 'Error predicting model' is built which is used to predict the errors that was calculated by model 'x'. Further the predictions made by model 'y' are then added to the ensemble of models. This process keeps repeating by making use of previous predictions in calculating new errors, building a new error-predicting model, and then, adding it to the ensemble model. A more detail version on implementation of this algorithm is explained in further sections of this paper.

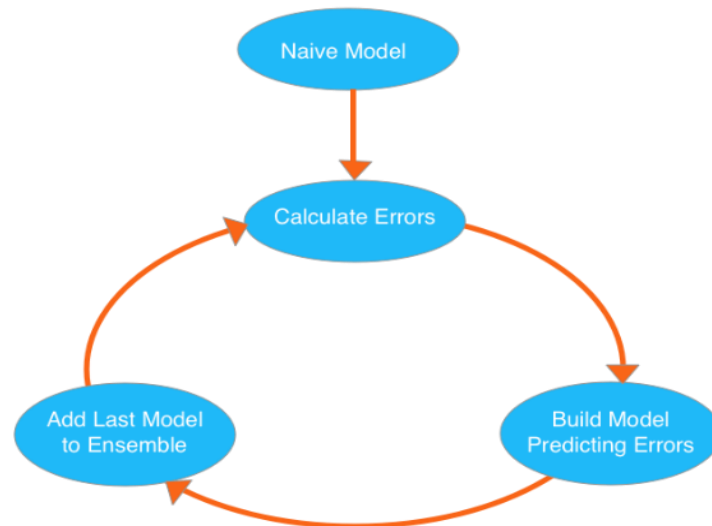


Fig. 6

6.8 ANN (artificial Neural network)

It is an approach through which we try to simulate the working of a human brain. Just like biological cells that exist in human brain, ANN contains a network of artificial neurons that gives machines the capability to think like human and make decisions like humans. Just like humans who make some decision on the basis of studying the inputs properly and then giving some output, ANN also studies the variable relationship between input and outputs and then makes some decision. ANN can be taken as weighted directed graph where the artificial neurons correspond to the nodes of graph and the weighted edges represent the relation between the input and the output.

Further, ANN has a varied architecture where the execution is spread among different layers of ANN:

Input Layer:

Layer is responsible for fetching the input in different formats from some external source and then these inputs are mathematically given a notation like $x(n)$ for every n number of input.

Hidden Layer:

The hidden layer is actually responsible for the computational part of ANN. As a part of it, first the inputs are taken from previous layers and they are multiplied with the corresponding weights of the edges. This computation is done for every ‘n’ number of input and then all the results are added to give the weighted sum of all inputs plus some bias.

$$\sum_{i=1}^n W(i) * X(i) + b \quad (5)$$

This sum if turns out to be ‘0’ then a certain bias is introduced into the total weighted inputs to make it non-zero. The total weighted inputs value can range over 0 to +ve infinity but some threshold value is set so that the output comes as desired. For this purpose, we pass our total weighted inputs through some activation function whose job is to transfer the input to desired output by choosing the nodes which makes to the output layer by firing them.

Output Layer:

The input after going through a series of computations in the hidden layer finally results into an output, the nodes which gets fired by the activation layer reaches the output layer.

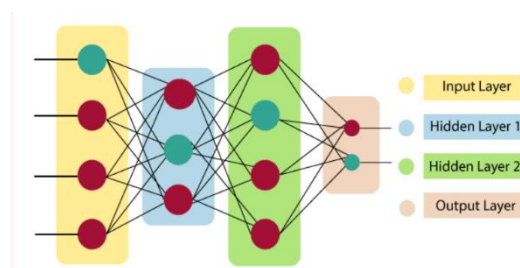


Fig. 7

6.9 CNN (Convolutional Neural network)

It is again a type of feed-forward neural network where it contains some additional layers as compared to the traditional neural network. For instance, these layers are:

- **Input layer**
- **Convolution Layers:** This is the major part of CNN where the input is scanned and then passed through some non-linear activation function for generating output. This output is fed to the next convolution layer until all are completed.
- **Pooling Layer:** This layer is basically used to reduce the dimensionality of the input images.
- **Dense layer:** Neurons in this layer takes input from neurons from all previous layers.
- **Output Layer**

CNN uses backpropagation to train the input nodes, this is done in order to reduce the value of cost function by iteratively adjusting weights, which ultimately helps in minimizing cost function. Backpropagation starts by forwarding the weighted sums through the layers and then gradient is calculated of cost function which is nothing but mean squared error for corresponding input-output pairs. After that, while propagating backwards the weights of first hidden layer are updated by the gradient value computed earlier, and this how this process keeps on propagating backwards until the starting point of network. This process stops when the gradient value has not changed much from the previous one.

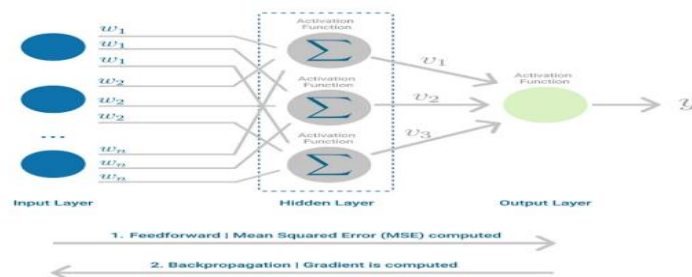


Fig. 8

7. IMPLEMENTATION AND RESULTS

In earlier section of this paper, we laid down five different goals that we proposed to achieve through this research work. In the following sections the executions and results of these goals are described:

7.1 As discussed in the earlier section of this paper we used linear regression to predict the outbreak of animal deaths as per the cases observed. The independent variable is 'sumCases' i.e., the number of affected cases and the target variable is 'sumDeaths' i.e., the number of deaths due to the specified disease. The variables shared a positive relationship which is demonstrated by the following figure:

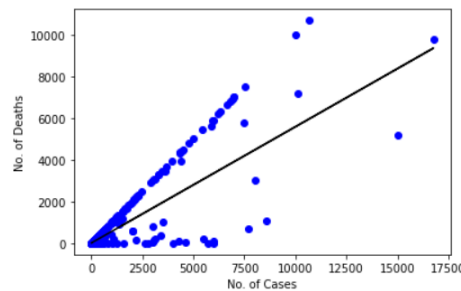


Fig. 9

The results of outbreak as well as the positive relationship between the animal and the cases observed stands for the fact that we need more robust practices when it comes to Animal health conditions all across the world because a huge number of animal deaths are observed yearly.

7.2 Next, we used Logistic regression and Support vector machine to predict human deaths as per the affected cases to measure the severity of zoonosis in those diseases. The independent variables were ‘Humans Age’ and ‘Humans Affected’ were extracted by performing PCA (Principal Component Analysis) on entire set of features and then finding the top two features that are most related with output or target variable ‘Human Deaths’, which further had integer values so we transformed these values and divided into two classes ‘0’ (If no human died of the diseases) and ‘1’ (if human deaths observed due to the diseases).

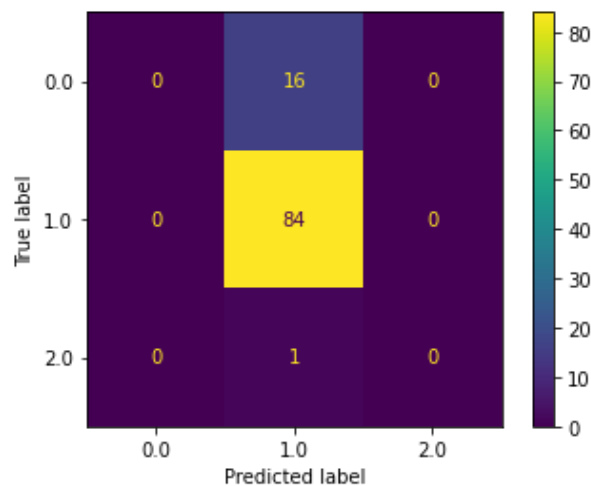


Fig. 10 Confusion matrix of logistic regression predicting the human deaths

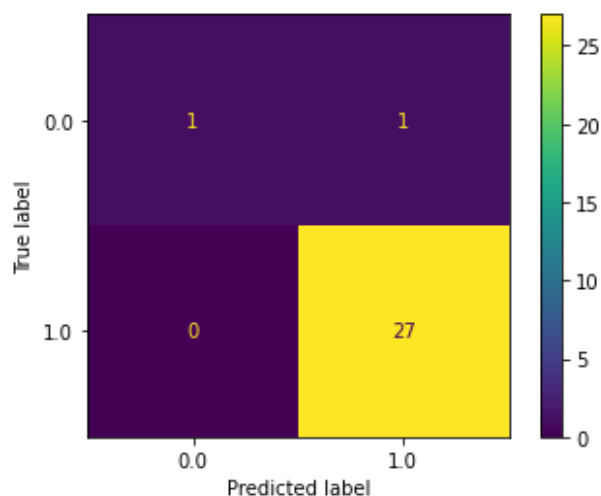


Fig. 11 Confusion matrix of SVM in predicting the human deaths

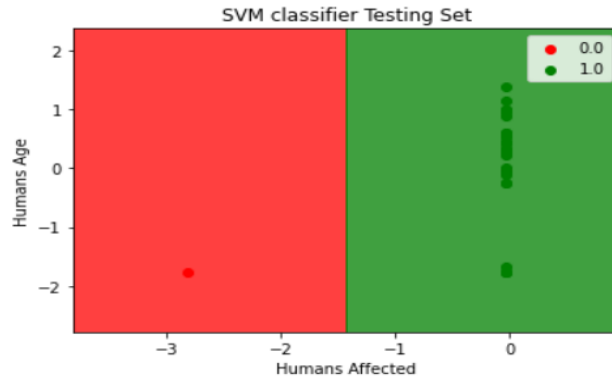


Fig. 12 SVM for predicting human deaths for testing dataset as per the affected cases to measure the severity of zoonosis

TABLE I. **PERFORMANCE MEASURES:** The below table contains the performance summary of the above-mentioned algorithms and their specific purpose

Aim	Technique	Accuracy	Precision	Recall	F1-Score
Animal Deaths outbreak Prediction	Linear Regression	74%	-	-	-
Predict human deaths to measure severity of zoonosis	Logistic Regression	83%	82%	99%	90%
	Support Vector Machine (SVM)	96%	96%	100%	98%

In addition to this we drew a comparative analysis between Logistic Regression and SVM, the results are depicted as follows, Clearly SVM performed better than Logistic regression:

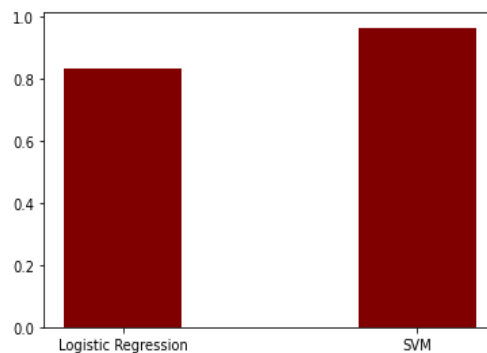


Fig. 13

7.3 Implementation of XGBoost to predict which outbreaks of animal diseases are more likely to get humans sick.

XGBoost algorithm can't just take a data-frame. It requires data to be in the form of matrix. So before executing XGBoost algorithm we first prepared data-frames in the form of DMatrix which were then fetched to the boosting algorithm. Before preparing the data, we did some basic data cleaning:

- **Step-1: Data cleaning**

First, we removed all the features from the dataset that contained information regarding the target variable. So, features like "humansGenderDesc", "humansAge", "humansAffected" and "humansDeaths" were dropped from the dataset. Next, we created a column 'diseaseInfo_numeric', which contains Boolean values: '0' if humans were affected by the disease and '1' if humans not affected. Furthermore, all the unnecessary information like 'latitude', 'longitude', 'id' etcetera from the dataset was removed, as it won't help in accomplishing the goal.

humansAffected
TRUE
FALSE
FALSE
FALSE
FALSE
FALSE

Fig. 14

- **Step-2: Data-frames Preparation**

We started by converting the features that were categorical in nature into numerical format. To serve this purpose, we used One-hot encoding method which creates separate column for each unique category, then each observation is tested against each column and that cell value is marked with ‘1’ if that observation belongs to that column and ‘0’ if it does not. The feature ‘country’ was converted into a hot-matrix ‘region’ using above method.

countryChina	countryFrance	countryIndonesia	countryMontenegro	countryRepublic of Korea	countryRussian Federation
0	1	0	0	0	0
0	0	1	0	0	0
0	0	0	0	1	0
0	0	0	1	0	0
0	0	0	0	0	1
1	0	0	0	0	0

Fig. 15

Similarly for feature ‘speciesDescription’ we created separate columns for each animal species and created a one-hot matrix named ‘species’, of different species that was listed under the ‘speciesDescription’ feature, this was done to discover that if some specific specie is more likely to spread disease to human. For instance, it can be a possibility that domestic species are more likely to transfer disease to human so we created a Boolean column ‘isdomestic’ and for each observation that contained the keyword domestic under ‘speciesDescription’ in it was marked ‘1’ in ‘is_domestic’ column and ‘0’ if otherwise.

- **Step-3: Combining all Data-Frames:**

From the previous steps we prepared three separate data-frames namely: 'diseaseInfo_numeric', 'region' and 'species'. We will bundle all these data-frames and convert them into a matrix.

- **Step-4:**

Dividing dataset into two subsets which is training and testing.

We split the dataset into the ratio of 70:30 where 70 is for training and 30 is for testing.

- **Step-5:**

Converting the clean data -frame to Dmatrix because our data-frame contains binary set of values that are '0' and '1' and once converted to a matrix it will be known as a sparse matrix, and Dmatrix helps to store and access sparse matrix values more easily which will in turn improve model training process.

- **Step-6: Training the model:**

Three inputs that were provided for training the model:

- i. Training data: passing the data in the form of Dmatrix generated in previous step.
- ii. Number of training rounds:
Number of training rounds means number of times the boosting cycle will go on and will keep adding models to the ensemble of models and hence improving the naïve model which was used as a base model for starting point.
- iii. Objective Function:
This help in determining the results. In our case since we are predicting something which is binary in nature (Humans get affected or not) so we have used 'binary logistic' function which is actually logistic regression for binary classification.

Code snippet:

```
# train an xgboost model
model_tuned <- xgboost(data = dtrain, # the data
                      max.depth = 3, # the maximum depth of each decision tree
                      nround = 2, # max number of boosting iterations
                      objective = "binary:logistic") # the objective function

# generate predictions for our held-out testing data
pred <- predict(model_tuned, dtest)

# get & print the classification error
err <- mean(as.numeric(pred > 0.5) != test_labels)
print(paste("test-error=", err))
```

Now let's examine the results of our naïve or base model:

TABLE II. **PERFORMANCE RESULTS FOR NAÏVE MODEL USED IN XGBOOST**

Error on training data	Number of training rounds	1	0.014698
		2	0.014698
Error on testing data	----	-- -	0.0121520972167777

As we can see no improvement was observed after second round of training. Now that we have built our base model and tested its performance, we can go ahead and tune or improve the model by making some changes as discussed in further section.

- **Step-7: Tuning the model:**

After we having our base model, we made an attempt to improve the performance of model by:

- i. Avoiding overfitting: More the number of layers in decision trees more likely is the chance for the model to capture randomness in the data rather than important variation, which can land the model into overfitting problem, hence decreasing the depth of trees by keeping a smaller number of layers in gradient decision tress.
- ii. Avoid imbalance classes
- iii. Training for more rounds
- iv. Early stopping while training: this means that if no improvement is observed in model's performance after a certain number of rounds so, we will stop the training, again this is done to prevent overfitting our model.

Code snippet:

```
# train a model using our training data
model_tuned <- xgboost(data = dtrain, # the data
  max.depth = 3, # the maximum depth of each decision tree
  nround = 10, # number of boosting rounds
  early_stopping_rounds = 3, # if we dont see an improvement in this many rounds, stop
  objective = "binary:logistic", # the objective function
  scale_pos_weight = negative_cases/postive_cases, # control for imbalanced classes
  gamma = 1) # add a regularization term

# generate predictions for our held-out testing data
pred <- predict(model_tuned, dtest)

# get & print the classification error
err <- mean(as.numeric(pred > 0.5) != test_labels)
print(paste("test-error=", err))
```

XGBoost model 'M' with training rounds 'n'=10

```
False_cases=sum(label =FALSE) #when no humans affected
True_cases=sum(label=TRUE)    #when humans are affected
While n≤10:
  M.XGBoost( data->training data
    Max.depth=3 #maximum depth of each decision tree
    es_rounds=3 #if improvement is not observed in the model's performance for
                these many rounds then we'll stop training
    objective_func= 'binary_logistic'
    scaling=False_cases/True_cases #avoiding imbalance classes
  )
  predicted ->pred(M)
Err ->mean(test)
```

RESULTS:

TABLE III. **PERFORMANCE RESULTS FOR XGBOOST MODEL AFTER TWEETING SOME PARAMETRES.**

Error on training data	Number of training Rounds	1	0.016126
		2	0.014866
		3	0.014866
		4	0.014866
		5	0.014614
		6	0.014530
		7	0.014530
		8	0.014530
		9	0.014614
Error on testing data	----	--	0.0121520972

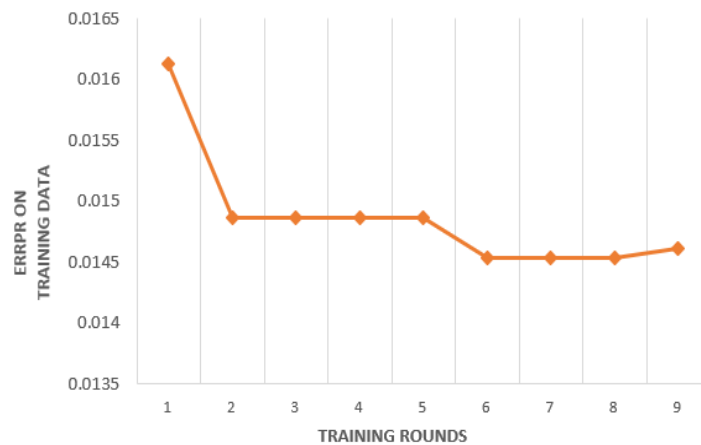


Fig. 16

Observations:

In the first training round the error comes out to be higher as compared to the error in the first round of training for naïve model, this is because we decreased the depth of decision tree for the boosting rounds to avoid overfitting. From second training rounds onwards, we observe that as the number of training rounds increases the error decreases this is because more the number of training or boosting round more accurately the model captures the variation in the dataset. In the 9th round of training suddenly error increases because this might be the point that our model has started overfitting the data, so as discussed in the previous section we stopped the training rounds as soon as no improvement seen in the performance.

Final error on train data: 0.014530

Final error on test data : 0.0121520972

- **Step-8: Examining the Model:**

We can examine our model by stacking all the gradient Decision trees used on top of each other and pick up the feature that shows up the most in each node for every separate tree.

```
# plot them features! what's contributing most to our model?
xgb.plot.multi.trees(feature_names = names(diseaseInfo_matrix),
                     model = model)
```

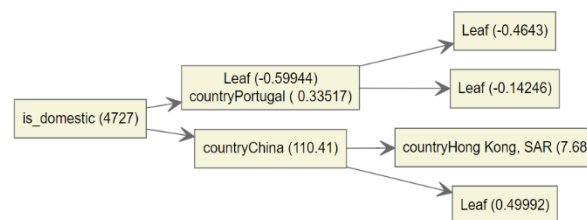


Fig. 17

The leftmost side of the picture is the top of the tree and the right-most is the bottom of tree. So, we can conclude that ‘is_domestic’ is the most important feature across all the trees in our ensemble model as it holds the highest position in the tree and also because the numeric value mentioned against the feature which is the quality factor is also the highest when compared to the quality factor of other features.

A visual representation of how important and informative the features were to determine if humans get affected by disease or not.

```
# get information on how important each feature is
importance_matrix <- xgb.importance(names(diseaseInfo_numeric), model = model)

# and plot it!
xgb.plot.importance(importance_matrix)
```

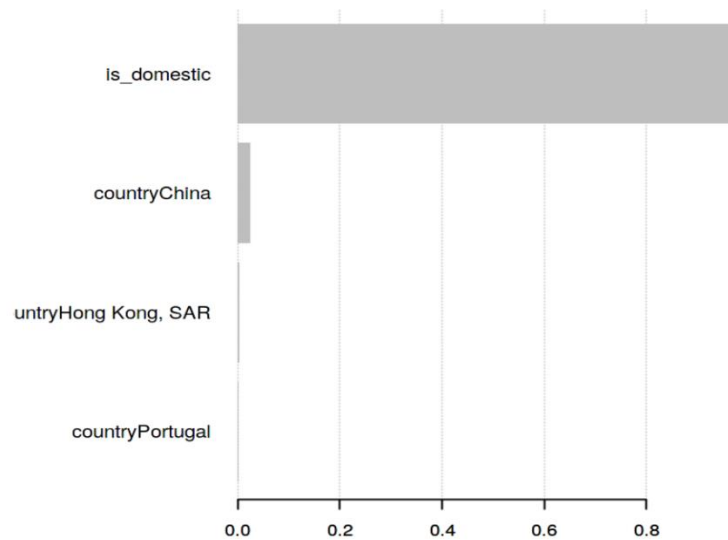


Fig. 18

The above plot shows that whether the affected animal was domestic or not was the most important factor to determine if human get affected by the same disease as well. Actually, it makes sense, because humans are more likely to get affected by certain zoonotic disease by coming in contact to a domestic animal rather than a wild or equine animal.

7.4 Animal Disease classification:

With the help of second dataset that was collected from WOAHA (World organisation of animal health), we used various models Logistic Regression, Naïve Byes, Decision tree, Random Forest, SVM, ANN to classify the diseases as per their symptoms. In ANN (Artificial Neural network) one input layer, two hidden layers and one output layer is added. 'ReLU' (rectifying linear unit) is the activation function deployed at hidden layer and the activation function used at output layer is 'softmax'.

Code Snippets:

Logistic Regression

```
lr=LogisticRegression(C=0.2,random_state=42, penalty='l2')
lr.fit(X_train,y_train)
print("Logistic Train score with ",format(lr.score(X_train, y_train)))
```

Logistic Train score with 0.9114350654440003

```
print("Logistic Test score with ",format(lr.score(test_set.iloc[:,1:], test_set['Disease'])))
```

Logistic Test score with 0.9079239117750029

```
y_pred = lr.predict(test_set.iloc[:,1:])
class_names=encoder.classes_
fig, ax = plt.subplots(figsize = (20,10))
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
cm = confusion_matrix(test_set['Disease'], y_pred)
sns.heatmap(cm, annot=True, cmap="YlGnBu",fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

Decission tree

```
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)
print("Decision Tree Train score with ",format(dt.score(X_train, y_train)))
```

Decision Tree Train score with 0.9148052355200251

```
print("Decision Tree Test score with ",format(dt.score(test_set.iloc[:,1:], test_set['Disease'])))
```

Decision Tree Test score with 0.9113082039911308

```
y_pred = dt.predict(test_set.iloc[:,1:])
class_names=encoder.classes_
fig, ax = plt.subplots(figsize = (20,10))
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
cm = confusion_matrix(test_set['Disease'], y_pred)
sns.heatmap(cm, annot=True, cmap="YlGnBu",fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

Text(0.5, 637.6, 'Predicted label')

Random Forest

```
rf = RandomForestClassifier(max_depth=6,oob_score=True,random_state=42,criterion='entropy',max_features
rf.fit(X_train, y_train)
y_pred=rf.predict(X_valid)
print("Random Forest Train score with ",format(rf.score(X_train, y_train)))
```

```
perm_imp1 = PermutationImportance(rf, random_state=42,scoring='accuracy').fit(test_set.iloc[:,1:], test
eli5.show_weights(perm_imp1, feature_names = test_set.iloc[:,1:].columns.tolist(),top=50)
```

```
print("Random Forest Test score with ",format(rf.score(test_set.iloc[:,1:], test_set['Disease'])))
```

```
y_pred = rf.predict(test_set.iloc[:,1:])
class_names=encoder.classes_
fig, ax = plt.subplots(figsize = (20,10))
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
cm = confusion_matrix(test_set['Disease'], y_pred)
sns.heatmap(cm, annot=True, cmap="YlGnBu",fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

```
print(classification_report( test_set['Disease'], y_pred))
```


SVM

```
svm = SVC()
svm.fit(X_train, y_train)
y_pred=svm.predict(X_valid)
print("SVM Train score with ",format(svm.score(X_train, y_train)))
```

```
print("SVM Test score with ",format(svm.score(test_set.iloc[:,1:], test_set['Disease'])))
```

```
y_pred = svm.predict(test_set.iloc[:,1:])
class_names=encoder.classes_
fig, ax = plt.subplots(figsize = (20,10))
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
cm = confusion_matrix(test_set['Disease'], y_pred)
sns.heatmap(cm, annot=True, cmap="YlGnBu",fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

```
print(classification_report( test_set['Disease'], y_pred))
```

Naive Bayes

```
bayes = GaussianNB()
bayes.fit(X_train, y_train)
y_pred=bayes.predict(X_valid)
print("Naive Bayes Train score with ",format(bayes.score(X_train, y_train)))
```

```
print("Naive Bayes Test score with ",format(bayes.score(test_set.iloc[:,1:], test_set['Disease'])), '%')
```

```
y_pred = bayes.predict(test_set.iloc[:,1:])
class_names=encoder.classes_
fig, ax = plt.subplots(figsize = (20,10))
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
cm = confusion_matrix(test_set['Disease'], y_pred)
sns.heatmap(cm, annot=True, cmap="YlGnBu",fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

```
print(classification_report( test_set['Disease'], y_pred))
```

Neural Network

ANN

```
# transform into dummies for y_train (prognosis variable)
y_train_dum = pd.get_dummies(y_train)

classifier = Sequential()

classifier.add(Dense(64, activation = "relu", input_dim = x_train.shape[1]))
# adding second hidden layer
classifier.add(Dense(48, activation = "relu"))
# adding last layer
classifier.add(Dense(y_train_dum.shape[1], activation = "softmax"))

classifier.compile(optimizer = "adam", loss = "categorical_crossentropy", metrics = ["accuracy"])
classifier.summary()

history = classifier.fit(X_train, y_train_dum, epochs = 5, batch_size = 30)

print("ANN Train score with ",format(history.history['accuracy'][-1]))

prediction = classifier.predict(test_set.iloc[:,1:])
prediction = [np.argmax(i) for i in prediction ]

print("ANN Test score with ",format(accuracy_score(test_set['Disease'], prediction)*100),'%')

class_names=encoder.classes_
fig, ax = plt.subplots(figsize = (20,10))
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
cm = confusion_matrix(test_set['Disease'], prediction)
sns.heatmap(cm, annot=True, cmap="YlGnBu",fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
```

TABLE IV. **PERFORMANCE MEASURES:** The below table contains the performance summary of the abovementioned algorithms in classifying animal disease as per the observed symptoms

Technique	Accuracy	Precision	Recall	F1-Score
Logistic Regression	90.8%	91%	91%	91%
Decision Tree	91.1%	91%	91%	91%
Support Vector Machine (SVM)	91.13%	91%	91%	91%
Random Forest	90.9%	91%	91%	91%
Naïve Bayes	78.8%	76%	79%	75%
ANN	91.2%	91%	91%	91%

Visual representation of the above performance measures:

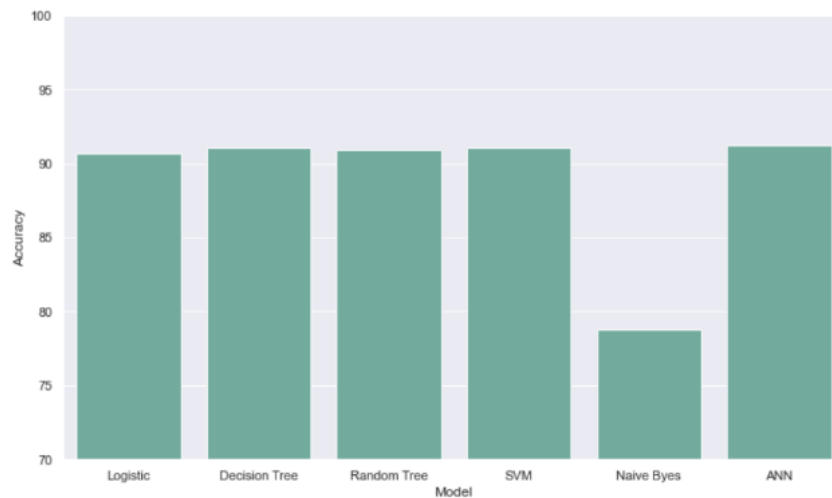


Fig. 19

As depicted in the above graph, the worst performance was shown by Naïve Bayes, this is because naïve bayes is more suitable for classifying data into a smaller number of categories but here we are trying to classify 63 different diseases. ANN gave the best performance in classifying the animal disease followed by SVM, Decision Tree, Random Forest and logistic regression.

7.5 CNN to predict if Disease Zoonotic or not:

In the previous section of this paper, we used Logistic regression and SVM to predict the number of human deaths as per the affected cases to measure the severity of zoonosis in those particular diseases.

We propose a new model where we will predict if the disease is zoonotic or not on the basis of symptoms observed for that disease.

Proposed model:

- **Data pre-processing and cleaning:**

Before jumping to build a CNN model we first performed a LASSO (Least Absolute Shrinkage) regression technique on our dataset. LASSO help in selecting important features from the dataset and eliminating the parameters that are not important, this is done to increase the accuracy of the model and is achieved by shrinking the data towards a central point. The idea is to sub-sample our dataset and run LASSO on it multiple times. So, for 'n' number of times that LASSO is performed over 'm' number of sub-samples of data and 37 variables we will get a set of values like $\mathbf{y}_i = [\mathbf{y}_{i,1} \mathbf{y}_{i,2} \dots \mathbf{y}_{i,37}]$ where 'i' is the observation. So, for nay variable 'a' we count the number of observations for which the variable was non-zero and if this count is greater or equal to the threshold value manually set then that variable 'a' is selected for further analysis.

Feature Nomination using LASSO

```

: from sklearn.linear_model import Lasso, LogisticRegression

featureVote = np.zeros(Data.shape[1])
print(featureVote.shape)

iteR = 100

for num in range(iteR):
    label0_idx = np.where(opLabel==0)[0] #not zoonotic
    label1_idx = np.where(opLabel==1)[0] #zoonotic
    numTrainData0 = 1300
    numTrainData1 = 1300
    np.random.shuffle(label0_idx)
    np.random.shuffle(label1_idx)

    label0_idx_train = label0_idx[0:numTrainData0-1]
    label1_idx_train = label1_idx[0:numTrainData1-1]
    label0_idx_test = label0_idx[numTrainData0-1:]
    label1_idx_test = label1_idx[numTrainData1-1:]

    testIdx = np.append(label0_idx_test, label1_idx_test)
    trainIdx = np.append(label0_idx_train, label1_idx_train)
    trainData = Data[trainIdx]
    trainLabel = opLabel[trainIdx]
    testData = Data[testIdx]
    testLabel = opLabel[testIdx]

    ### data standardization
    scaler = preprocessing.StandardScaler().fit(trainData)
    trainData_scaled = scaler.transform(trainData)
    testData_scaled = scaler.transform(testData)

    ### Elastic net and Lasso from scikit
    #regr = ElasticNet(random_state=0, alpha=1, l1_ratio=0.03, tol=0.000001, max_iter=100000)
    regr = Lasso(random_state=0, alpha=0.006, tol=0.000001, max_iter=100000)
    #regr = LogisticRegression(penalty='l1', random_state=0, C=100, tol=0.000001, max_iter=100, class_weight='balanced')
    regr.fit(trainData_scaled, trainLabel)
    cof = np.abs(regr.coef_)
    colIdx = np.where(cof != 0)[0]
    for col in colIdx:
        featureVote[col] += 1

(37,)

# feature nomination via Lasso (from feature 1 to 30)
# We keep the dummy variables

#thresH = iteR//5. Pick features occurring more than 5 times
thresH = 0
featureIdx = np.where(featureVote[0:30] >= thresH)[0]
featureIdx = np.append(featureIdx, np.arange(30, Data.shape[1]))
print(varb[featureIdx])

```

- **Proposed CNN Model:**

With the help of CNN model, we will be predicting the status of disease in regard to zoonosis. So, the prediction will be done for two classes that are zoonosis class: ‘1’ and no-zoonosis class: ‘0’.

Below is the visual representation of the model architecture:



Fig. 20

The input layer accepts a ‘1D-Numerical’ array which contains all the 37 features that were selected through LASSO and Majority Voting process. So, the dimension of input layer will be $N \times 37$ where ‘N’ is the number of training rounds or examples. The dense layer just after the input layer contains 64 neurons and it linearly combines the 37 variables with some bias factor. Further the activation function rectified linear unit (ReLU) is used to perform the non-linear transformation of the input. Also, a dropout factor is introduced in the model where 20% of data is dropped just to avoid overfitting scenario. After the first dense layer or the fully connected layer we have two consecutive Convolution layers. In convolution layer1 we have applied two filters where kernel width=3 and stride=1. So, this first Convolution layer converts the output block of dense layer i.e. $N \times 64$ to a tensor having dimensions $N \times 64 \times 1$. This tensor further undergoes batch-normalization, non-linear transformation

by ReLu and average pooling to give an output tensor of dimension Nx31x2.

The filters in second convolution layers are applied with kernel width =5 and stride=1 and the output given by second convolution layers is tensor of dimension Nx13x4, this tensor will serve as an input to the next dense layer. The final output of the disease belonging to class ‘zoonosis’ or class ‘no-zoonosis’, is given at the end of SoftMax layer where categorical cross -entropy is used as the loss function. In the next section we can see the performance results shown by our model.

MLP+ CONV: Input => Dense(64) => Conv(2) => Conv(4) = Dense(512) => Dense (2) [Best model]

```
inputs = keras.layers.Input(shape=(x_train.shape[1],1))
RS0 = keras.layers.Reshape((x_train.shape[1], ))(inputs)
FC0 = keras.layers.Dense(64, bias_initializer=keras.initializers.VarianceScaling())(RS0)
BN0 = keras.layers.BatchNormalization(axis=-1)(FC0)
AC0 = keras.layers.Activation('relu')(BN0)
DP0 = keras.layers.Dropout(0.2)(AC0)

RS1 = keras.layers.Reshape((64,1))(DP0)
FC1 = keras.layers.Conv1D(2,3, strides=1)(RS1)
BN1 = keras.layers.BatchNormalization(axis=-1)(FC1)
AC1 = keras.layers.Activation('relu')(BN1)
Pool1 = keras.layers.AveragePooling1D(pool_size=2)(AC1)

FC2 = keras.layers.Conv1D(4,5, strides=1)(Pool1)
BN2 = keras.layers.BatchNormalization(axis=-1)(FC2)
AC2 = keras.layers.Activation('relu')(BN2)
Pool2 = keras.layers.AveragePooling1D(pool_size=2)(AC2)

FL1 = keras.layers.Flatten()(Pool2)

FC3 = keras.layers.Dense(512, bias_initializer=keras.initializers.VarianceScaling())(FL1)
BN3 = keras.layers.BatchNormalization(axis=-1)(FC3)
AC3 = keras.layers.Activation('relu')(BN3)
DP3 = keras.layers.Dropout(0.2)(AC3)

FC4 = keras.layers.Dense(2)(DP3)
outputs = keras.layers.Activation('softmax')(FC4)

myCNN1D4 = keras.Model(inputs=inputs, outputs=outputs)
myCNN1D4.compile(optimizer=keras.optimizers.Adam(),
                 loss='categorical_crossentropy',
                 metrics=['accuracy'])

myCNN1D4.summary()
```

- **Results of our model:**

Confusion matrix:

TABLE V. CONFUSION MATRIX FOR CNN CLASSIFIER

Out of 6065 true non-zoonotic diseases the classifier was able to predict all 6065 correctly. Out of 9732 true zoonotic diseases 9716 diseases were correctly predicted as zoonotic

Total		True Condition	
		Non-zoonotic	Zoonotic
Disease predicted	Non-zoonotic	True positive (TP) 6065	False Negative (FN) 0
	Zoonotic	False Positive (FP) 16	True Negative (TN) 9716

```
class_weight = {0: 1, 1: 2.9}

myCNN1D4.fit(x_train,y_train,epochs=1,verbose=1, class_weight=class_weight)

171/171 [=====] - 8s 10ms/step - loss: 0.2246 - accuracy: 0.9400

<keras.callbacks.History at 0x1dce8443af0>
```

```
test_loss,test_acc = myCNN1D4.evaluate(x_test,y_test)
print(test_acc)

494/494 [=====] - 3s 4ms/step - loss: 0.1515 - accuracy: 0.9990
0.9989871382713318
```

```
from sklearn import metrics
predlabel = myCNN1D4.predict(x_test)
f = np.argmax(predlabel,axis=1)
confMat = metrics.confusion_matrix(np.argmax(y_test,axis=1),f)
print(confMat)

494/494 [=====] - 2s 4ms/step
[[6065  0]
 [ 16 9716]]
```


Observation: when it comes to diagnosing a medical condition, the false negatives should be less because if our model will predict that human is not affected when actually the human is affected by the disease, it will lead us into thinking that we are not affected hence no prevention would be taken which in turn can turn out to be very dangerous situations.

As we can see in the above results false negatives are zero which stands for the fact that our model has done a good job in predicting the disease as zoonotic if it actually is, due to this people can start taking precautions from early stage.

Some additional performance measures of the model are listed below:

TABLE VI. **ADDITIONAL PERFORMANCE MEASURE**
 $\text{TPR (total positive rate)} = \text{TP}/\text{TP}+\text{FN}$,
 $\text{Total negative Rate (TNR)} = \text{TN}/\text{TN}+\text{FP}$,
 $\text{Accuracy} = \text{TP}+\text{TN}/\text{TP}+\text{TN}+\text{FP}+\text{FN}$, $\text{Precision} = \text{TP}/\text{TP}+\text{FP}$

Measure	Class Weight	Score
TPR%	2.9 : 1	100
TNR%	2.9 : 1	100
Accuracy%	2.9 : 1	99.8
Precision%	2.9 : 1	99.7
Train Accuracy %	2.9 : 1	94
Test Accuracy%	2.9 : 1	99.9
Training Loss	2.9 : 1	0.1515

8. LIMITATIONS

There are a number of constraints that need to be taken into account, despite the fact that machine learning approaches have showed considerable promise in the prediction of zoonotic illnesses. First off, the reliability of predictions is largely dependent on the accessibility of data. The reliability of machine learning models can sometimes be impacted by the inadequate, inconsistent, or biased nature of zoonotic disease data. Additionally, due to differences in data formats and standards, it might be difficult to integrate data from many sources.

The complexity of zoonotic illnesses is another drawback. The complex connections between animals, people, and the environment that these diseases entail make it challenging to include all important variables in a predictive model. These complexities are frequently simplified by machine learning models, which may leave out important factor. Additionally, machine learning models are challenged by the zoonotic diseases' dynamic nature. Numerous factors, such as climate change, urbanisation, or changes in human behaviour, can cause outbreak patterns and disease dynamics to shift quickly. It may be difficult for machine learning models developed using past data to adjust to these changing circumstances, producing predictions that are not as accurate.

Moreover, machine learning models are typically trained to make predictions based on trends seen in the past. But zoonotic illnesses might display unusual or uncommon patterns that have never been seen before, making it difficult for models to precisely forecast such occurrences. This constraint becomes especially clear when addressing newly or recently reemerging zoonotic illnesses for which there is insufficient historical data.

9. FUTURE WORK

Machine learning shows enormous promise for influencing the direction of preventive healthcare in the field of zoonotic disease prediction. The importance of early detection and preventative measures in lessening the effects of zoonotic illnesses is becoming increasingly clear as technology develops. Machine learning algorithms have the potential to completely change how we forecast and prevent certain diseases because of their capacity to analyse enormous volumes of data and spot trends. The integration of several datasets is a critical component of upcoming work in machine learning-based zoonotic disease prediction. Machine learning models can develop a thorough grasp of the intricate interactions between animal hosts, environmental conditions, human health data, and genetic information by incorporating data from a variety of sources.

The creation of sophisticated machine learning algorithms that can manage high-dimensional and heterogeneous data will also be crucial. For example, deep learning algorithms may extract subtle features and relationships from complicated datasets, allowing for more precise forecasts and a greater comprehension of the dynamics of zoonotic diseases.

The development of machine learning-powered real-time surveillance systems will be another area of emphasis. These systems have the capacity to continuously analyse data coming from sources including environmental sensors, animal monitoring devices, and health records for people and animals. These models can alert medical practitioners and policymakers to impending zoonotic disease outbreaks by spotting early warning indicators and aberrant patterns, enabling quick response and containment.

10. CONCLUSION

Machine Learning Techniques have proved to be useful in many areas. After COVID it was very imperial to have built certain systems which can predict the outbreak of such infectious diseases which not only has the tendency to threaten the livelihood of animal species but humans as well. With the help of the predictions made using ML techniques we can be prepared to combat any pandemic like situation that is likely to become a great threat to living beings.

Overall, by utilising a variety of data sources, spotting hidden patterns, and streamlining decision-making procedures, machine learning approaches show tremendous potential in the prediction of zoonotic illnesses. These methods support global efforts to lessen the negative effects of zoonotic illnesses on human and animal health by improving our understanding of disease dynamics and identifying efficient preventative and control measures.

REFERENCES

- [1] Singh, R. K., & Sharma, V. C. (2015). Ensemble Approach for Zoonotic Disease Prediction Using Machine Learning Techniques.
- [2] Atil, H., & Akilli, A. (2016). Comparison of artificial neural network and K-means for clustering dairy cattle. *International Journal of Sustainable Agricultural Management and Informatics*, 2(1), 40-52.
- [3] Zhang, S., Su, Q., & Chen, Q. (2021). Application of machine learning in animal disease analysis and prediction. *Current Bioinformatics*, 16(7), 972-982.
- [4] Valdes-Donoso P, VanderWaal K, Jarvis LS, Wayne SR, Perez AM. Using Machine Learning to Predict Swine Movements within a Regional Program to Improve Control of Infectious Diseases in the US. *Front Vet Sci*. 2017 Jan 19;4:2. doi: 10.3389/fvets.2017.00002. PMID: 28154817; PMCID: PMC5243845.
- [5] Morota, G., Ventura, R. V., Silva, F. F., Koyama, M., & Fernando, S. C. (2018). Big data analytics and precision animal agriculture symposium: Machine learning and data mining advance predictive big data analysis in precision animal agriculture. *Journal of animal science*, 96(4), 1540-1550.
- [6] Peters, D. P., McVey, D. S., Elias, E. H., Pelzel-McCluskey, A. M., Derner, J. D., Burruss, N. D., ... & Rodriguez, L. L. (2020). Big data–model integration and AI for vector-borne disease prediction. *Ecosphere*, 11(6), e03157.
- [7] Corley, C. D., Pullum, L. L., Hartley, D. M., Benedum, C., Noonan, C., Rabinowitz, P. M., & Lancaster, M. J. (2014). Disease prediction models and operational readiness. *PloS one*, 9(3), e91989.
- [8] Buza, T., Arick, M., Wang, H., & Peterson, D. G. (2014). Computational prediction of disease microRNAs in domestic animals. *BMC Research Notes*, 7(1), 1-13.

- [9] Kasbohm, E., Fischer, S., Küntzel, A., Oertel, P., Bergmann, A., Trefz, P., ... & Köhler, H. (2017). Strategies for the identification of disease-related patterns of volatile organic compounds: prediction of paratuberculosis in an animal model using random forests. *Journal of Breath Research*, 11(4), 047105.
- [10] Ortiz-Pelaez, Á., & Pfeiffer, D. U. (2008). Use of data mining techniques to investigate disease risk classification as a proxy for compromised biosecurity of cattle herds in Wales. *BMC Veterinary Research*, 4(1), 1-16.
- [11] Li, X., Zhang, Z., Liang, B., Ye, F., & Gong, W. (2021). A review: Antimicrobial resistance data mining models and prediction methods study for pathogenic bacteria. *The Journal of Antibiotics*, 74(12), 838-849.
- [12] Huang, S., Cai, N., Pacheco, P. P., Narrandes, S., Wang, Y., & Xu, W. (2018). Applications of support vector machine (SVM) learning in cancer genomics. *Cancer genomics & proteomics*, 15(1), 41-51.
- [13] Schuldt, C., Laptev, I., & Caputo, B. (2004, August). Recognizing human actions: a local SVM approach. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.* (Vol. 3, pp. 32-36). IEEE.
- [14] Tsang, I. W., Kwok, J. T., Cheung, P. M., & Cristianini, N. (2005). Core vector machines: Fast SVM training on very large data sets. *Journal of Machine Learning Research*, 6(4).
- [15] Salazar, D. A., Vélez, J. I., & Salazar, J. C. (2012). Comparison between SVM and logistic regression: Which one is better to discriminate?. *Revista Colombiana de Estadística*, 35(SPE2), 223-237.
- [16] Musa, A. B. (2013). Comparative study on classification performance between support vector machine and logistic regression. *International Journal of Machine Learning and Cybernetics*, 4(1), 13-24.

- [17] Sanson, R. L., Pfeiffer, D. U., & Morris, R. S. (1991). Geographic information systems: their application in animal disease control. *Rev sci tech*, 10(1), 179-195.
- [18] De La Rocque, S., Rioux, J. A., & Slingenbergh, J. (2008). Climate change: effects on animal disease systems and implications for surveillance and control. *Rev Sci Tech*, 27(2), 339-54.
- [19] Hästein, T., Hill, B. J., & Winton, J. R. (1999). Successful aquatic animal disease emergency. *Rev. sci. tech. Off. int. Epiz*, 18(1), 214-227.
- [20] Morgan, N., & Prakash, A. (2006). International livestock markets and the impact of animal disease. *Rev Sci Tech*, 25(2), 517-528.
- [21] Beard, Rachel, Elizabeth Wentz, and Matthew Scotch. "A systematic review of spatial decision support systems in public health informatics supporting the identification of high risk areas for zoonotic disease outbreaks." *International journal of health geographics* 17.1 (2018): 1-19.
- [22] Smith, K.F., Goldberg, M., Rosenthal, S., Carlson, L., Chen, J., Chen, C. and Ramachandran, S., 2014. Global rise in human infectious disease outbreaks. *Journal of the Royal Society Interface*, 11(101), p.20140950.
- [23] Chowdhury, S., Aleem, M. A., Khan, M. S. I., Hossain, M. E., Ghosh, S., & Rahman, M. Z. (2021). Major zoonotic diseases of public health importance in Bangladesh. *Veterinary Medicine and Science*, 7(4), 1199-1210.
- [24] Smith, Katherine F., Michael Goldberg, Samantha Rosenthal, Lynn Carlson, Jane Chen, Cici Chen, and Sohini Ramachandran. "Global rise in human infectious disease outbreaks." *Journal of the Royal Society Interface* 11, no. 101 (2014): 20140950.

- [25] Morand, Serge, and Claire Lajaunie. "Outbreaks of vector-borne and zoonotic diseases are associated with changes in forest cover and oil palm expansion at global scale." *Frontiers in veterinary science* (2021): 230.
- [26] Winck, Gisele R., et al. "Socioecological vulnerability and the risk of zoonotic disease emergence in Brazil." *Science Advances* 8.26 (2022): eabo5774.
- [27] Ostfeld, Richard S., and Robert D. Holt. "Are predators good for your health? Evaluating evidence for top-down regulation of zoonotic disease reservoirs." *Frontiers in Ecology and the Environment* 2.1 (2004): 13-20.
- [28] Harrison, Mark E., et al. "Tropical peatlands and their conservation are important in the context of COVID-19 and potential future (zoonotic) disease pandemics." *PeerJ* 8 (2020): e10283.
- [29] Salata, Cristiano, et al. "Coronaviruses: a paradigm of new emerging zoonotic diseases." *Pathogens and disease* 77.9 (2019): ftaa006.
- [29] Narrod, Clare, Jakob Zinsstag, and Marites Tiongco. "A one health framework for estimating the economic costs of zoonotic diseases on society." *EcoHealth* 9.2 (2012): 150-162.
- [30] Cascio, Antonio, et al. "The socio-ecology of zoonotic infections." *Clinical microbiology and infection* 17.3 (2011): 336-342.
- [31] Kshirsagar, D. P., et al. "Disease alerts and forecasting of zoonotic diseases: an overview." *Veterinary World* 6.11 (2013): 889.

