

PROFESSIONAL TRAINING REPORT
at
Sathyabama Institute of Science and Technology
(Deemed to be University)

Submitted in partial fulfillment of the requirements for the
award of Bachelor of Engineering Degree in Information
Technology

By
Edupuganti Srivalli
REG. NO. 39120028



DEPARTMENT OF INFORMATION TECHNOLOGY

SCHOOL OF COMPUTING

**SATHYABAMA INSTITUTE OF SCIENCE AND
TECHNOLOGY**

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119, TAMILNADU**

NOVEMBER 2021



SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade “A” by NAAC
(Established under Section 3 of UGC Act, 1956)
JEPPIAAR NAGAR, RAJIV GANDHI SALAI
CHENNAI– 600119
www.sathyabama.ac.in



DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of
Edupuganti Srivalli (Reg. No: 39120028) who carried out the
project entitled

“Covid – 19 VACCINE USAGE PREDICTION” under my
supervision from June 2021 to November 2021.

Internal Guide

Dr. Nirmalrani, M.E., Ph.D.,

Head of the Department

Dr. Subhashini, M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **Edupuganti Srivalli** hereby declare that the project report entitled “**Covid – 19 VACCINE USAGE PREDICTION**” done by me under the guidance of **Dr.Nirmalrani**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Information technology.

DATE: 09-11-2021

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D., Dean**, School of Computing, **Dr. R. Subhashini, M.E., Ph.D. Head of the Department of Information technology** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.Nirmalrani, M.E., Ph.D.,** for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Information Technology** who were helpful in many ways for the completion of the project

TRAINING CERTIFICATE



Certificate of Completion

Awarded to

Srivalli Edupuganti

Upon successfully Completed the Boot Camp Training in Machine Learning
for 45 Hrs from 01st Sep 2021 To 27th Sep 2021



Nikhil Barshikar
Director

ABSTRACT

From fun to finance, from science to security, from prediction to protein, the expanse of machine learning algorithms has reached every stratum. Humans are vulnerable and are open to over or under estimation, but not machines. For providing appropriate results and making effective decisions on data, some advanced data mining techniques can be used in many fields. Healthcare is an inevitable task to be done in human life. In today's era the main cause of death is COVID. The most efficient way to reduce the risk of COVID is getting vaccinated. But not everyone is showing interest to vaccinate so to know the usage of vaccinations we can use Machine learning. In the study, we proposed a machine-learning based diagnosis system by getting data from government and cleaning the data and finally to know how much percent people are showing interest to get vaccinated. Taking vaccination on time is very essential to live. The obtained results have illustrated that the designed diagnostic system can effectively predict the usage of vaccination

TABLE OF CONTENTS

CHAPTER No.	TITLE	PAGE No
1.	INTRODUCTION	1
	BACKGROUND	1
	OBJECTIVE	2
2.	LITERATURE REVIEW	3
3.	TECHNOLOGIES LEARNT	4
	PYTHON	4
	ADAVANTAGES IN PYTHON	4
	DISADAVANTAGES IN PYTHON	8
	MACHINE LEARNING	
	CHALLENGES IN MACHINE LEARNING	
	TYPES OF MACHINE LEARNING	
	ADVANTAGES OF MACHINE LREANING	
	DISADAVANTAGES OF MACHINE	
	LEARNING	8
	APPLICATIONS OF MACHINE LEARNING	8
4.	MODELS AND MODULES USED IN	
	PROJECT	9
	SUPERVISED MACHINE LEARNING	9
	WORKINF OF SUPERVISED LEARNING	
	STEPS INVOLVED IN SUPVERVISED	
	LEARNING	11
	TYPES OF SUPERVISED MACHINE	
	LEARNING ALGORITHMS	11

	REGRESSION	12
	CLASSIFICATION	12
	ADVANTAGES OF SUPERVISED	
	LEARNING	12
	DISADVANTAGES OF SUPERVISED	
	LEARNING	13
	LOGISTIC REGRESSION WITH MLE	13
	MAXIMUM LIKELIHOOD ESTIMATION	
	(MLE)	15
	STOCHASTIC GRADIENT DESCENT	
	(GSD)	17
	STEPS INVOLVED IN ALGORITHM	18
	MODULES USED IN PROJECT	
5.	RESULT AND CONCLUSIONS	21
	REFERENCES	22
	APPENDIX	23
	A. SOURCE CODE	23
	B. SCREENSHOTS	31

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE NO.
01.	Working of logistic regression	14
02.	Types of Supervised Learning	16
03.	Sigmoid curve	17
04.	Probabilities of predict values	

VACCINE USAGE PREDICTION BASED ON MACHINE LEARNING

CHAPTER 1

Introduction

Vaccination plays a vital role in eradicating and controlling the wide spread of diseases like covid - 19 throughout the world. It is also considered to be cost effective method in public health services and save millions of lives. Different vaccines could be suggested for any particular disease there by necessitating comparative vaccine trial to estimate the potency of vaccines. Numerous studies have been conducted in the estimation of vaccine efficacy using case-control and prospective cohort studies; and these studies offered varied information.

1.1 Back Ground

In December 2019, an outbreak of pneumonia of unknown origin was reported in Wuhan, Hubei Province, China. Pneumonia cases were epidemiologically linked to the Huanan Seafood Wholesale Market. Inoculation of respiratory samples into human airway epithelial cells, Vero E6 and Huh7 cell lines, led to the isolation of a novel respiratory virus whose genome analysis showed it to be a novel coronavirus related to SARS-CoV, and therefore named severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2). SARS-CoV-2 is a betacoronavirus belonging to the subgenus Sarbecovirus. The global spread of SARS-CoV-2 and the thousands of deaths caused by coronavirus disease (COVID-19) led the World Health Organization to declare a pandemic on 12 March 2020. To date, the world has paid a high toll in this pandemic in terms of human lives lost, economic repercussions and increased poverty. The development of vaccines, the role of artificial intelligence in the management of the pandemic and limiting the spread of the virus, the impact of the COVID-19 epidemic on our lifestyle, and preparation for a possible second wave. Vaccination is one of the cornerstones of controlling a covid pandemic. To optimize vaccination rates in the general population, ways of identifying determinants that influence decisions to have or not to have a vaccination need to be understood. Therefore, this study aimed to predict intention to have a covid vaccination in an adult population.

CHAPTER 2

AIM, SCOPE, OBJECTIVE

Aim

The main aim of the project is To know the vaccination status in a country or state to enhance non vaccinated to get vaccinated.

Scope

The purpose of this project mainly is to get the status of vaccination. This can be known from websites of government by considering a dataset. Data should be cleaned by splitting it into categorical and numerical sets and performing the models. Finally know which model best predicted it and use it to get accurate results

Objective

- To Develop a Machine Learning algorithm for COVID-19 Vaccine prediction
- Getting maximum knowledge on Machine Learning Algorithm
- Learning to increase accuracy of the model
- To gain complete Knowledge on Python Libraries
- To become Ready to solve any classifier Algorithm Problems
- To be able to use Jupiter Note book or Goggle Colab
- Able to Decrease Time Consumption

Hardware requirements

- **Processor:** intel core i5
- **Speed:** 1.19GHz
- **RAM:** 8.0GB
- **Memory usage:** 7.2kb
- **Keyboard:** Standard Keyboard
- **Monitor:** 15VGA color

Software Requirements

- **Operating System:** Windows
- **Coding Language:** Python 3.7
- **Online Services:** Google Colaboratory

CHAPTER 3

TECHONOLOGIES LEARNT

PYTHON

Python is currently the most widely used multi-purpose, high-level programming language. Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java. Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time. Python language is being used by almost all tech-giant companies like Google, Amazon, Facebook, Instagram, Dropbox, Uber... etc. The biggest strength of Python is huge collection of standard library which can be used for the following –

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like OpenCV, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia

ADVANTAGES OF PYTHON

- Extensive Libraries
- Extensible
- Embeddable
- Improved Productivity
- IOT Opportunities
- Readable
- Object-Oriented
- Free and Open Source
- Portable
- Interpreted

DISADVANTAGES OF PYTHON

So far, we've seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let's now see the downsides of choosing Python over another language.

- Speed Limitations
- Weak in Mobile Computing and Browsers
- Design Restrictions
- Underdeveloped Database Access layers
- Simple

MACHINE LEARNING

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of building models of data.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models tunable parameters that can be adapted to observed data; in this way the program can be considered to be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is similar to the "learning" exhibited by the human brain.

CHALLENGES IN MACHINE LEARNING

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is that ML has not been able to overcome number of

challenges. The challenges that ML is facing currently are –

Quality of data

Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.

Time-Consuming task

Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.

Lack of specialist persons

As ML technology is still in its infancy stage, availability of expert resources is a tough job.

No clear objective for formulating business problems

Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.

Issue of overfitting & underfitting

If the model is overfitting or underfitting, it cannot be represented well for the problem.

Curse of dimensionality

Another challenge ML model faces is too many features of data points. This can be a real hindrance.

Difficulty in deployment

Complexity of the ML model makes it quite difficult to be deployed in real life.

TYPES OF MACHINE LEARNING

Supervised Learning

This involves learning from a training dataset with labeled data using classification and regression models. This learning process continues until the required level of performance is achieved.

Unsupervised Learning

This involves using unlabelled data and then finding the underlying structure in the data in order to learn more and more about the data itself using factor and cluster

analysis models.

Semi-supervised Learning

This involves using unlabelled data like Unsupervised Learning with a small amount of labeled data. Using labeled data vastly increases the learning accuracy and is also more cost-effective than Supervised Learning.

Reinforcement Learning

This involves learning optimal actions through trial and error. So the next action is decided by learning behaviors that are based on the current state and that will maximize the reward in the future.

ADVANTAGES OF MACHINE LEARNING

Easily identifies trends and patterns

Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, it serves to understand the browsing behaviors and purchase histories of its users to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.

No human intervention needed (automation)

With ML, you don't need to babysit your project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and also improve the algorithms on their own. A common example of this is anti-virus software's, they learn to filter new threats as they are recognized. ML is also good at recognizing spam.

Continuous Improvement

As ML algorithms gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say you need to make a weather forecast model. As the amount of data you have keeps growing, your algorithms learn to make more accurate predictions faster.

Handling multi-dimensional and multi-variety data

Machine Learning algorithms are good at handling data that are multi-dimensional and multi-variety, and they can do this in dynamic or uncertain environments.

Wide Applications

You could be an e-tailer or a healthcare provider and make ML work for you. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.

DISADVANTAGES OF MACHINE LEARNING

Acquisition

Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.

Time and Resources

ML needs enough time to let the algorithms learn and develop enough to fulfill their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function. This can mean additional requirements of computer power for you.

Interpretation of Results

Another major challenge is the ability to accurately interpret results generated by the algorithms. You must also carefully choose the algorithms for your purpose.

High error-susceptibility

Machine Learning is autonomous but highly susceptible to errors. Suppose you train an algorithm with data sets small enough to not be inclusive. You end up with biased predictions coming from a biased training set. This leads to irrelevant advertisements being displayed to customers. In the case of ML, such blunders can set off a chain of errors that can go undetected for long periods of time. And when they do get noticed,

it takes quite some time to recognize the source of the issue, and even longer to correct it.

APPLICATIONS OF MACHINE LEARNING

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach.

Following are some real-world applications of ML.

- Emotion analysis
- Sentiment analysis
- Error detection and prevention
- Weather forecasting and prediction
- Stock market analysis and forecasting
- Speech synthesis
- Speech recognition
- Customer segmentation
- Object recognition
- Fraud detection
- Fraud prevention
- Recommendation of products to customer in online shopping.

CHAPTER 4

MODELS AND MODULES USED IN PROJECT

Supervised Machine Learning

Supervised learning is the types of machine learning in which machines are trained using well "labelled" training data, and on basis of that data, machines predict the output. The labelled data means some input data is already tagged with the correct output. In supervised learning, the training data provided to the machines work as the supervisor that teaches the machines to predict the output correctly. It applies the same concept as a student learns in the supervision of the teacher. Supervised learning is a process of providing input data as well as correct output data to the machine learning model. The aim of a supervised learning algorithm is to find a mapping function to map the input variable(x) with the output variable(y). In the real-world, supervised learning can be used for Risk Assessment, Image classification, Fraud Detection, spam filtering, etc.

Working of Supervised Learning

In supervised learning, models are trained using labelled dataset, where the model learns about each type of data. Once the training process is completed, the model is tested on the basis of test data (a subset of the training set), and then it predicts the output. The working of Supervised learning can be easily understood by the below example and diagram:

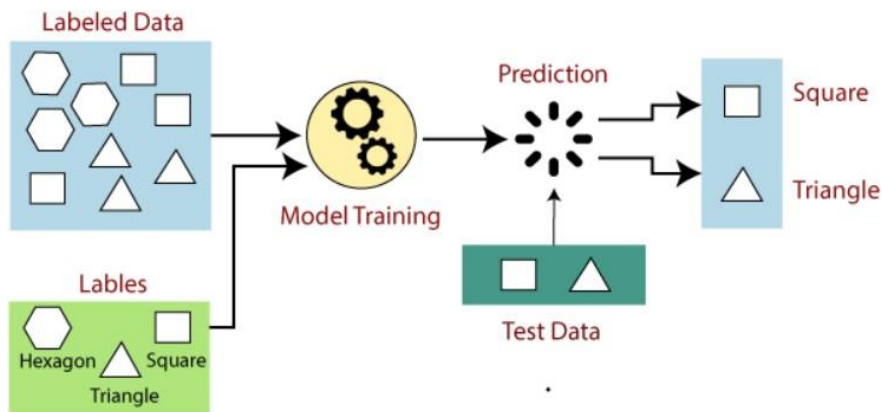


Fig.1. working of Supervised learning.

Suppose we have a dataset of different types of shapes which includes square, rectangle, triangle, and Polygon. Now the first step is that we need to train the model for each shape.

- If the given shape has four sides, and all the sides are equal, then it will be labelled as a Square.
- If the given shape has three sides, then it will be labelled as a triangle.
- If the given shape has six equal sides then it will be labelled as hexagon.

Now, after training, we test our model using the test set, and the task of the model is to identify the shape.

The machine is already trained on all types of shapes, and when it finds a new shape, it classifies the shape on the bases of a number of sides, and predicts the output.

Steps Involved in Supervised Learning:

- First Determine the type of training dataset.
- Collect/Gather the labelled training data.
- Split the training dataset into training dataset, test dataset, and validation dataset.
- Determine the input features of the training dataset, which should have enough knowledge so that the model can accurately predict the output.
- Determine the suitable algorithm for the model, such as support vector machine, decision tree, etc.
- Execute the algorithm on the training dataset. Sometimes we need validation sets as the control parameters, which are the subset of training datasets.
- Evaluate the accuracy of the model by providing the test set. If the model predicts the correct output, which means our model is accurate.

Types of supervised Machine learning Algorithms:

Supervised learning can be further divided into two types of problems:

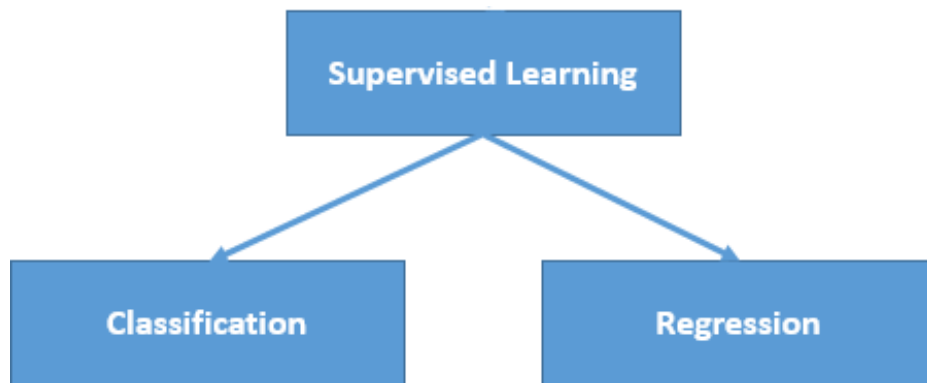


Fig.2. Types of supervised learning

Regression

Regression algorithms are used if there is a relationship between the input variable and the output variable. It is used for the prediction of continuous variables, such as Weather forecasting, Market Trends, etc. Below are some popular Regression algorithms which come under supervised learning:

- Linear Regression
- Regression Trees
- Non-Linear Regression
- Bayesian Linear Regression
- Polynomial Regression

Classification

Classification algorithms are used when the output variable is categorical, which means there are two classes such as Yes-No, Male-Female, True-false, etc.

- Random Forest
- Decision Trees
- Logistic Regression
- Support vector Machines

Advantages of Supervised learning

- With the help of supervised learning, the model can predict the output on the

basis of prior experiences.

- In supervised learning, we can have an exact idea about the classes of objects.
- Supervised learning model helps us to solve various real-world problems such as fraud detection, spam filtering, etc.

Disadvantages of supervised learning

- Supervised learning models are not suitable for handling the complex tasks.
- Supervised learning cannot predict the correct output if the test data is different from the training dataset.
- Training required lots of computation times.
- In supervised learning, we need enough knowledge about the classes of object.

Logistic Regression

Logistic Regression is a classification algorithm of Machine Learning where the output variable is categorical. It falls under the Supervised Learning method where the past data with labels is used for building the machine learning model.

Essentially Logistic Regression model outputs probabilities (or log odds ratios in the logit form) that have a linear relationship with the predictor variables. When you attach a threshold to these probability values, it classifies the outcomes as 1 or 0 (Binomial Logistic Regression). Hence even if Logistic Regression is a classification algorithm, it has the word regression in it.



Fig.3. Sigmoid curve

The S-shaped curve shown in the figure above is a sigmoid curve. Logistic regression function is also called sigmoid function. The expression for logistic regression function is:

$$f(y) = \frac{1}{1 + e^{-y}}$$

Where:

$y = \beta_0 + \beta_1 x$ (in case of univariate Logistic regression)

$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 \dots + \beta_n x_n$ (in case of multivariate logistic regression)

Univariate Logistic Regression means the output variable is predicted using only one predictor variable, while Multivariate Logistic Regression means output variable is predicted using multiple predictor variables.

The logistic regression function converts the values of logits also called log-odds that range from $-\infty$ to $+\infty$ to a range between 0 and 1.

Now let us try to simplify what we said. Let P be the probability of occurrence of an event. So probability the the event will not occur is $1-P$.

Odds is defined as the ratio of the probability of occurrence of a particular event to the probability of the event not occurring.

$$Odds = \frac{P}{1 - P}$$

We know that logistic regression function gives us probability value. So we can write:

$$P = \frac{1}{1 + e^{-y}}$$

Now since we mentioned log odds, let us take the natural log of both sides of the Odds equation and substitute the value of P .

$$\ln(Odds) = \ln\left(\frac{P}{1-P}\right) = y$$

$$\ln\left(\frac{P}{1-P}\right) = \beta_0 + \beta_1 x$$

Thus, we get a more simplified form of logistic regression function equation and we can say that log odds has linear relationship with the predictor variable x.

Maximum Likelihood Estimation (MLE)

In order that our model predicts output variable as 0 or 1, we need to find the best fit sigmoid curve, that gives the optimum values of beta co-efficient. That is we need to create an efficient boundary between the 0 and 1 values.

Now a cost function tells you how close your values are from actual. So here we need a cost function which maximizes the likelihood of getting desired output values. Such a cost function is called as Maximum Likelihood Estimation (MLE) function.

Let this cost function be represented as $P(Y; z)$. We have some sample data points represented by Y. So Y represents 'n' observations say $Y_1, Y_2, \dots, Y_n$. We need to find this unknown parameter z, such that the probability of observing Y is maximized. z is the bias which decides whether the variable will take a value of 0 or 1.

Remember this is a supervised learning algorithm. So we will have actual observations and predicted observations. Our cost function should be such that it maximizes the probability of predicted values being close to the actual ones.

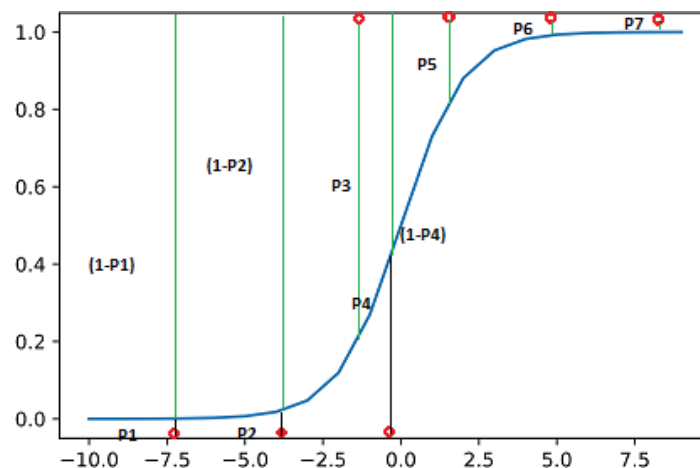


Fig.4. Probabilities of predict values

From the above figure, we can see the points being classified as 0 or 1 and the respective probabilities associated with them. There are 7 points and seven associated probabilities P1 to P7.

For points to be 0, we need the probabilities P1, P2 and P4 to be as minimum as possible and for points to be 1, we need the probabilities P3, P5, P6 and P7 to be as high as possible, for correct classification. We can also say that (1-P1), (1-P2), P3, (1-P4), P5, P6 and P7 should be as high as possible.

The joint probability is nothing but the product of probabilities. So the product: $[(1-P1)*(1-P2)* P3*(1-P4)*P5*P6*P7]$ should be maximum.

This joint probability function is nothing but our cost function which should be maximized in order to get a best fit sigmoid curve. Or we can say predicted values to be close to the actual values.

Now coming to our cost function, let $J(z)$ be a function of z such that

$$J(z) = P(Y ; z) = P(Y_1, Y_2 \dots Y_n ; z)$$

The assumption here is that all Y are independent.

$$J(z) = \prod_{i=1}^n P(Y_i ; z)$$

Taking natural log on both sides :

$$\ln(J(z)) = \ln\left(\prod_{i=1}^n P(Y_i ; z)\right)$$

Since log of product becomes summation:

$$\ln(J(z)) = \sum_{i=1}^n \ln (P(Y_i ; z))$$

$J(z)$ can also be written as $L(z|Y_i)$ (L for Likelihood). For a given value of z and observed sample Y_i , this function gives the probability of observing the sample values. So if $Y_i=1$ the expression becomes z and if Y_i is 0 the expression becomes $1-z$:

$$\ln(J(z)) = L(z | Y_i) = \sum_{i=1}^n \ln (z^{Y_i} * (1 - z)^{1-Y_i})$$

Solving this equation further we get :

$$\ln(J(z)) = L(z | Y_i) = \sum_{i=1}^n Y_i * \ln(z) + (1 - Y_i) * \ln(1 - z)$$

Differentiating this equation with respect to z and setting the derivative to zero, we calculate the maxima using closed form solution:

$$\frac{d}{dz} (\sum_{i=1}^n Y_i * \ln(z) + (1 - Y_i) * \ln(1 - z)) = 0$$

Solving further, this equation becomes:

$$\frac{z}{1-z} = \sum_{i=1}^n \frac{Y_i}{1-Y_i}$$

The right side term represents the ratio of number of 1s to number of 0s. Thus the function achieves a maximum at :

$$z = \frac{\sum_{i=1}^n Y_i}{n}$$

Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD) is a simple yet very efficient approach to fitting linear classifiers and regressors under convex loss functions such as (linear) Support Vector Machines and Logistic Regression. Even though SGD has been around in the machine learning community for a long time, it has received a considerable amount of attention just recently in the context of large-scale learning.

SGD has been successfully applied to large-scale and sparse machine learning problems often encountered in text classification and natural language processing. Given that the data is sparse, the classifiers in this module easily scale to problems with more than 10^5 training examples and more than 10^5 features.

Strictly speaking, SGD is merely an optimization technique and does not correspond to a specific family of machine learning models. It is only a way to train a model. Often, an instance of SGD Classifier or SGD Regressor will have an equivalent estimator in the scikit-learn API, potentially using a different optimization technique. For example, using `SGDClassifier(loss='log')` results in logistic regression, i.e. a model equivalent to Logistic Regression which is fitted via SGD instead of being fitted by one of the other solvers in

Logistic Regression. Similarly, `SGDRegressor(loss='squared_error', penalty='l2')` and Ridge solve the same optimization problem, via different means.

The advantages of Stochastic Gradient Descent are:

- Efficiency.
- Ease of implementation (lots of opportunities for code tuning).

The disadvantages of Stochastic Gradient Descent include:

- SGD requires a number of hyperparameters such as the regularization parameter and the number of iterations.
- SGD is sensitive to feature scaling.

Steps involved in algorithm

- Find the slope of the objective function with respect to each parameter/feature. In other words, compute the gradient of the function.
- Pick a random initial value for the parameters. (To clarify, in the parabola example, differentiate “y” with respect to “x”. If we had more features like x_1 , x_2 etc., we take the partial derivative of “y” with respect to each of the features.)
- Update the gradient function by plugging in the parameter values.
- Calculate the step sizes for each feature as : $\text{step size} = \text{gradient} * \text{learning rate}$.
- Calculate the new parameters as : $\text{new params} = \text{old params} - \text{step size}$
- Repeat steps 3 to 5 until gradient is almost 0.

MODULES USED IN PROJECT

NUMPY

NumPy is a general-purpose array-processing package. It provides a high performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. It contains various

features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using NumPy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

PANDAS

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

MATPLOTLIB

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.

Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits.

Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.^c For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of

functions familiar to MATLAB users.

SCIKIT – LEARN

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

5. RESULT AND CONCLUSION

This study has indicated that by using a machine learning it is possible to extract different patterns based on different attributes, input keys and predict attributes which impart useful knowledge with great accuracy. The purpose of this work was to compare algorithms with different performance measures using machine learning. All data were pre-processed and used for test prediction. Each algorithm worked better in some situations and worse in others. The resulting models generated of this research are worthy for clinical testing. In the data, missing values, noisy data, inconsistencies, and outliers showed a big challenge during the process of mining of data. LR, LDA, KNN, Cart, NB, SVM, Decision tree, Random forest are the models likely to work best in the data set used in the study. Random forest gives the best result of 83 percent as an accuracy. So this model can be a big help of predicting vaccine usage.

Future scope

The scope of our project is to provide an efficient and effective software for the users and government to know the vaccination status in country or state. This pandemic has took many lives but we shouldnt stop fighting. The only way to eradicate is hetting vaccinated so this software tool really gonna help to know vaccination status and get non vaccinated persons to get vaccinated in time and stop this pandemic.

REFERENCES

- [1] Yang, J., Jiang, H., & Zhang, H. (2011). Teaching Assistant Evaluation Based on Support Vector Machines with Parameters Optimization. *Information Technology Journal*, 10(11), 2140-2146.
- [2] Badur, B., & MARDIKYAN, S. (2011). Analyzing teaching performance of instructors using data mining techniques. *Informatics in Education-An International Journal*, (Vol 10_2), 245-257.
- [3] Bao, Q., Newport, D., Chen, M., Stout, D. B., & Chatziioannou, A. F. (2009). Performance evaluation of the inveon dedicated PET preclinical tomograph based on the NEMA NU-4 standards. *Journal of Nuclear Medicine*. 50(3), 401-408.
- [4] Baradwaj, B. K., & Pal, S. (2012). Mining educational data to analyze students' performance. *arXiv preprint arXiv:1201.3417*.
- [5] Yadav, S. K., & Pal, S. (2012). Data mining: A prediction for performance improvement of engineering students using classification. *arXiv preprint arXiv:1203.3832*

SOURCE CODE

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn

data = pd.read_csv('/content/h1n1_vaccine_prediction.csv')
data.dtypes
data = data.drop(columns=['unique_id'])
data.columns
data_category_column = data.columns[data.dtypes=='object'] #Which of the
columns are category type
data_numeric_column = data.columns[data.dtypes!='object'] #Which of the
columns are of numeric type
print(data_category_column) #Names of category columns
print(data_numeric_column) #Names of numeric columns

Create data sub sets
data_category = data[data_category_column] #Category Data
data_numeric = data[data_numeric_column] #Numeric Data
data_category.shape
data_numeric.shape
data_category.head()
data_numeric.head()

Check for missing data
data_category.isna().sum().sort_values(ascending=False)
data_numeric.isna().sum().sort_values(ascending=False)

Categorical imputation using lambda function
lambda x: x.fillna(x.value_counts().index[0])
#For Sample
y = pd.DataFrame(['Male','Male',np.NaN,np.NaN,np.NaN,'Female','Male'])
```

```
print(y.value_counts())
```

```
y=y.apply(lambda x: x.fillna(x.value_counts().index[0]))
```

```
y
```

```
data_clean_category = data_category.apply(lambda  
x:x.fillna(x.value_counts().index[0]))
```

```
data_clean_category.isna().sum().sort_values(ascending=False)
```

```
data_clean_category.dtypes
```

Use a Loop to convert all object columns to category

```
for column in data_clean_category.columns:
```

```
    data_clean_category[column] = data_clean_category[column].astype('category')
```

```
data_clean_category.dtypes
```

```
data_clean_category.head()
```

Numeric columns missing value imputation

```
data_numeric.head()
```

```
data_numeric.dtypes
```

```
data_clean_numeric = data_numeric.apply(lambda  
x: x.fillna(x.value_counts().index[0]))
```

```
data_clean_numeric.head()
```

```
data_clean_numeric.isna().sum()
```

```
data_clean_numeric.dtypes
```

```
data_clean_numeric = data_clean_numeric.astype('category')
```

```
data_clean_numeric.head()
```

```
data_clean_numeric['h1n1_worry'].value_counts()
```

```
category_count = data_clean_category.columns.value_counts()
```

```
sns.barplot(category_count.index,category_count.values,alpha=0.9)
```

```
plt.show()
```

```
numeric_count = data_clean_numeric.columns.value_counts()
```

```

sns.barplot(numeric_count.index,numeric_count.values,alpha=0.9)

plt.show()
data_clean_category.columns
data_clean_numeric.columns
data_clean_category.dtypes
data_clean_numeric.dtypes
data_clean_category
data_clean_category['age_bracket'] = data_clean_category['age_bracket'].cat.codes

data_clean_category['qualification'] =data_clean_category['qualification'].cat.codes

data_clean_category['race']=data_clean_category['race'].cat.codes

data_clean_category['sex']=data_clean_category['sex'].cat.codes

data_clean_category['income_level']=data_clean_category['income_level'].cat.codes

data_clean_category['marital_status']=data_clean_category['marital_status'].cat.codes
s
data_clean_category['housing_status']=data_clean_category['housing_status'].cat.codes
es
data_clean_category['employment']=data_clean_category['employment'].cat.codes

data_clean_category['census_msa'] = data_clean_category['census_msa'].cat.codes

data_clean_category.head()
data_clean_numeric.isna().sum()
data_clean_numeric = data_clean_numeric.apply(lambda x:
x.fillna(x.value_counts().index[0]))
data_clean_numeric.isna().sum()

```

```
print(type(data_clean_category))
print(type(data_clean_numeric))
```

Merging all data

Integrating category data with numeric data

```
covid_data = pd.concat([data_clean_category,data_clean_numeric],
                        join='outer',
                        axis=1)
```

```
covid_data.head()
covid_data.corr()
X = covid_data.iloc[:, :-1]
```

```
y = covid_data.iloc[:, -1]
X.shape,y.shape
```

Independent Data X has category values. This has to be one hot encoded for numeric computation

```
X = pd.get_dummies(X)
X.shape
X.head()
sns.heatmap(X)
covid = pd.concat([X,y],axis=1)
covid.head()
type(covid)
```

Converting X and y to arrays

```
X = covid.iloc[:, :-1].values #features

y = covid.iloc[:, -1].values #target
X,y,X.shape,y.shape
```

```

from sklearn.model_selection import train_test_split

Xtrain,Xtest,ytrain,ytest = train_test_split(X,y,test_size=0.2)
Xtrain.shape,Xtest.shape,ytrain.shape,ytest.shape
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model
model.fit(Xtrain,ytrain)
ypred=model.predict(Xtest)
ytest
ypred
model.coef_ #Weights or coefficient
model.intercept_

from sklearn import model_selection

from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
print(sklearn.metrics.accuracy_score(ytest,ypred))
print(sklearn.metrics.classification_report(ytest,ypred))
print(sklearn.metrics.confusion_matrix(ytest,ypred))
(4049+430) / (4049+194+669+430)
Vaccinators = pd.DataFrame(ypred) #ypred is in array.It needs to be converted to
DataFrame

Vaccinators.value_counts()
print('Default Logistic Regression Accuracy Training
Score',model.score(Xtrain,ytrain))
print('Default Logistic Regression Accuracy Test Score',model.score(Xtest,ytest))
# Logistic Regression using MLE

```

```
covid.shape #Checking records and variables
import statsmodels.api as sm
from statsmodels.formula.api import logit #We want to use Logistic Regression in
StatsModels
covid_float = covid.astype('float') #statsmodels require dataset to have float values
X =covid_float.iloc[:,S-1]
```

```
y = covid_float.iloc[:,-1]
mle_model = logit("y~X",covid_float,).fit() #Equation of MLE using statsmodels
print(mle_model.summary())
print(mle_model.summary2())
mle_model.pred_table() #Build Confusion Matrix
accuracy = (19942+2398)/(19942+1091+3276+2398)
accuracy
mle_model.aic
```

Logistic Regression using SGD Model

```
from sklearn.linear_model import SGDClassifier
sgd_classifier = SGDClassifier()
sgd_classifier
SGDClassifier(def __init__(loss='hinge', penalty='l2', alpha=0.0001, l1_ratio=0.15,
fit_intercept=True, max_iter=1000, tol=0.001, shuffle=True, verbose=0,
epsilon=DEFAULT_EPSILON, n_jobs=None, random_state=None,
learning_rate='optimal', eta0=0.0, power_t=0.5, early_stopping=False,
validation_fraction=0.1, n_iter_no_change=5, class_weight=None,
warm_start=False, average=False))
sgd_classifier = SGDClassifier(loss='hinge',
                             penalty='l2',
                             max_iter=1000)
sgd_classifier
sgd_classifier.fit(Xtrain,ytrain)
```

```

print('Training Accuracy Score',sgd_classifier.score(Xtrain,ytrain))
print('Validation Accuracy Score',sgd_classifier.score(Xtest,ytest))
sgd_classifier.predict(Xtest)
sgd_classifier.coef_
sgd_classifier.intercept_
sgd_classifier.decision_function(Xtest)

```

Changing the loss function to log and max iterations to 5

```

sgd_classifier = SGDClassifier(alpha=0.0001, average=False,
class_weight=None,
                                early_stopping=False, epsilon=0.1, eta0=0.0,
fit_intercept=True,
                                l1_ratio=0.15, learning_rate='optimal', loss='log',
                                max_iter=5, n_iter_no_change=5, n_jobs=None, penalty='l2',
                                power_t=0.5, random_state=None, shuffle=True, tol=0.001,
                                validation_fraction=0.1, verbose=0,
warm_start=False).fit(Xtrain,ytrain)
sgd_classifier.predict_proba(Xtest)
print(sgd_classifier.predict_proba(Xtest))
sgd_ypred = sgd_classifier.predict(Xtest)
sklearn.metrics.accuracy_score(ytest,sgd_ypred)

```

Compare Various Models

#Step 5: Build models

```

from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.SS.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier

```

```

# Spot Check Algorithms
models = []
models.append(('LR', LogisticRegression()))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC()))
models.append(('DecisionTree', DecisionTreeClassifier()))
models.append(('Random Forest', RandomForestClassifier()))
# evaluate each model in turn
results = []
names = []
for name, model in models:
    kfold = model_selection.KFold(n_splits=10, random_state=None)
    cv_results = model_selection.cross_val_score(model, Xtrain, ytrain, cv=kfold,
scoring='accuracy')
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
    print(msg)
#Now we can make prediction on most accurate model
# Make predictions on validation dataset
rf = RandomForestClassifier()
rf.fit(Xtrain, ytrain)
predictions = rf.predict(Xtest)
print(accuracy_score(ytest, predictions))
print(confusion_matrix(ytest, predictions))
print(classification_report(ytest, predictions))

```

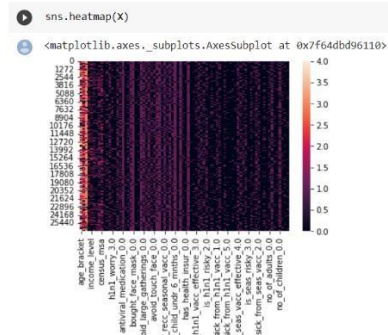
SCREENSHOTS

POSSIBLE DATA LOSS - Some features might be lost if you save this workbook in the comma-delimited CSV format. To preserve these features, save it in an Excel file format. Don't show again Save As...																								
A1	unique_id																							
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V		
1	unique_id	h1n1_wor	h1n1_awa	antiviral_n	contact_av	bought_fa	wash_han	avoid_larg	reduced_o	avoid_tou	dr_recc_h	dr_recc_s	chronic_m	cont_child	is_health	has_health	is_h1n1_v	is_h1n1_r	sick_from	is_seas_v	is_seas_r	sick_from	age	
2	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	1	3	1	2	2	1	2	55	
3	1	3	2	0	1	0	1	0	0	1	1	0	0	0	0	1	5	4	4	4	2	4	35	
4	2	1	1	0	1	0	0	0	0	0	0	0	1	0	0	3	1	1	4	1	2	18		
5	3	1	1	0	1	0	1	0	1	0	0	0	1	1	0	0	3	3	5	5	4	1	65	
6	4	2	1	0	1	0	1	1	0	1	0	0	0	0	0	0	3	3	2	3	1	4	45	
7	5	3	1	0	1	0	1	0	0	1	0	1	0	0	0	0	5	2	1	5	4	4	65	
8	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	1	1	4	2	1	55	
9	7	1	0	0	1	0	1	0	0	0	1	1	0	1	0	0	1	5	2	1	4	2	1	45

	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK	AL	AM	AN	AO	AP	AQ
1	sick_from	age_brack	qualification	race	sex	income_level	marital_status	housing_status	employment	census_msa	no_of_adults	no_of_children	h1n1_vaccine									
2	2 55 - 64 Yrs	< 12 Years	White	Female	Below Pov	Not Marri	Own	Not in Lab	Non-MSA	0	0	0										
3	4 35 - 44 Yrs	12 Years	White	Male	Below Pov	Not Marri	Rent	Employed	MSA, Not i	0	0	0										
4	2 18 - 34 Yrs	College Gr	White	Male	<= \$75,000	Not Marri	Own	Employed	MSA, Not i	2	0	0										
5	1 65+ Years	12 Years	White	Female	Below Pov	Not Marri	Rent	Not in Lab	MSA, Princ	0	0	0										
6	4 45 - 54 Yrs	Some Coll	White	Female	<= \$75,000	Married	Own	Employed	MSA, Not i	1	0	0										
7	4 65+ Years	12 Years	White	Male	<= \$75,000	Married	Own	Employed	MSA, Princ	2	3	0										
8	1 55 - 64 Yrs	< 12 Years	White	Male	<= \$75,000	Not Marri	Own	Employed	MSA, Not i	0	0	0										
9	1 45 - 54 Yrs	Some Coll	White	Female	<= \$75,000	Married	Own	Employed	Non-MSA	2	0	1										
10	1 45 - 54 Yrs	College Gr	White	Male	> \$75,000	Married	Own	Employed	MSA, Not i	1	0	0										
11	2 55 - 64 Yrs	12 Years	White	Male	<= \$75,000	Not Marri	Own	Not in Lab	MSA, Not i	0	0	0										

+ Code + Text

data.dtypes	
unique_id	int64
h1n1_worry	float64
h1n1_awareness	float64
antiviral_medication	float64
contact_avoidance	float64
bought_face_mask	float64
wash_hands_frequently	float64
avoid_large_gatherings	float64
reduced_outside_home_cont	float64
avoid_touch_face	float64
dr_recc_h1n1_vacc	float64
dr_recc_seasonal_vacc	float64
chronic_medication	float64
cont_child_under_6_mths	float64
is_health_worker	float64
has_health_insurance	float64
is_h1n1_vaccine_effective	float64
is_h1n1_risky	float64
sick_from_h1n1_vacc	float64
is_seas_vaccine_effective	float64
is_seas_risky	float64
sick_from_seas_vacc	float64
age_bracket	object
qualification	object
race	object
sex	object
income_level	object
marital_status	object
housing_status	object
employment	object
census_msa	object
no_of_adults	float64
no_of_children	float64
h1n1_vaccine	int64
dtype:	object



```
[ ] from sklearn import model_selection

from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score

[ ] print(sklearn.metrics.accuracy_score(ytest,ypred))

0.8384500187195807

[ ] print(sklearn.metrics.classification_report(ytest,ypred))

              precision    recall  f1-score   support

     0       0.86       0.95       0.90       4243
     1       0.69       0.39       0.50       1099

 accuracy_
macro avg       0.77       0.67       0.70       5342
weighted avg       0.82       0.84       0.82       5342

[ ] print(sklearn.metrics.confusion_matrix(ytest,ypred))

[[4049  194]
 [ 669 430]]

[ ] (4049+430) / (4049+194+669+430)

0.8384500187195807
```

```
[ ] mle_model.pred_table() #Build Confusion Matrix

array([[19942., 1091.],
       [ 3276., 2398.]])

[ ] accuracy = (19942+2398)/(19942+1091+3276+2398) #logistic regression using MLE
accuracy

0.836484816714719
```

```
1 #How we can make prediction on most accurate model
# Make predictions on validation dataset
rf = RandomForestClassifier()
rf.fit(Xtrain, ytrain)
predictions = rf.predict(Xtest)
print(accuracy_score(ytest, predictions))
print(confusion_matrix(ytest, predictions))
print(classification_report(ytest, predictions))

0.8328341445151629
[[4042  201]
 [ 692 407]]

              precision    recall  f1-score   support

     0       0.85       0.95       0.90       4243
     1       0.67       0.37       0.48       1099

 accuracy_
macro avg       0.76       0.66       0.69       5342
weighted avg       0.82       0.83       0.81       5342
```