International Academic Institute for Science and Technology

International Academic Journal of Innovative Research
Vol.11, No.1, 2024, pp.06-16.

ISSN 2454-390X

International Academic Journal of Innovative Research

www.iaiest.com

Covid -19 Prediction

Dr.N. Baggyalakshmi [a*], K. Jayasri [b], Dr.R. Revathi [c]

[a] Assistant Professor, Department of Computer Science, PSGR Krishnammal College for Women, Coimbatore.*
E-mail: baggyanethra@gmail.com
[b] Student, B.sc Computer Science, PSGR Krishnammal College for Women, Coimbatore.
E-mail: jayasrikumar0006@gmail.com
[c] Assistant Professor, Department of Computer Science, Karpagam Academy of Higher Education, Coimbatore.
E-mail: revathilakshay@gmail.com

Abstract

A global emergency has emerged in the form of the COVID-19 pandemic. If governments around the world are serious about lowering the impact of this worldwide epidemic, they must devote significant resources to studying this disease. This research looks into the disease's outbreak, trains datasets for the Indian region, and tests the number of cases over the following three weeks. Based on data acquired from the official portal of the Government of India throughout the crutial time period, we have utilised various machine learning algorithms, including a logistic regression model, to make predictions. When testing the models' efficacy, several regression models were used. In the three-week period of test data, the expected instances are anticipated to be approximately 175K--200K, which closely matches the actual figures. Both the government and medical professionals can use this information to inform their future plans. The goal of data science is to discover new patterns and relationships in large datasets by applying statistical and computational methods. Cleansing and preparing data, visualising data, statistical modelling, machine learning, and many other tasks are all part of it. Discovering trends and patterns in data, creating forecasts, and providing decision-making support are all possible with the use of these methods. Data types that they might encounter include both structured (like dates and numbers in a spreadsheet) and unstructured (like text, photos, or audio). Many different sectors make use of data science, including banking, medicine, retail, and many more.

Keywords: COVID19, Machine Learning, Logistic Regression, Cleaning, Preparation, Visualization.

DOI:10.9756/IAJIR/V11I1/IAJIR1102
Received: 03 January 2024/Accepted: 29 January 2024/Published: 08 February 2024

Introduction

Python data scientists also collect data from many sources and keep a database up to date. Data analysts are tasked with discovering, analysing, and interpreting patterns or trends in large datasets. They find this useful for data cleaning and filtering.

A data analyst uses data to answer the mundane queries that arise in company operations, but a data scientist uses data to draw important conclusions. In contrast to data scientists, who focus on the future using existing data, data analysts are more concerned with the present. As a result, data analysts need to be familiar with spreadsheet programmes, and data scientists should have strong business acumen.

Data science is no different from any other field in that it requires initial inspiration to begin. Clearly and exactly stating your situation is of the utmost importance [1]. Your assertion is crucial to the functioning of your entire model. This is the first and most crucial stage in date science, according to many scientists. So, be careful with your problem statement and think about how it may benefit a company or other group.

Finding data that could be useful for your model is the next logical step after stating the problem. Find everything you need by doing thorough investigation. Any type of data can be stored, whether it's organised or not. Media such as films, spreadsheets, programmed forms, etc., might be included. Gathering all of these types of sources is essential.

The following phase, after formulating your motive and collecting facts, is cleaning. I agree! For data scientists, cleansing data is the most enjoyable part of their job. Eliminating duplicate, irrelevant, or missing data is the main goal of data cleaning [2]. If you know how to code in R or Python, you can use any number of tools to accomplish this. Picking one of them is entirely up to you.

With the rise of the COVID-19 pandemic, data science has emerged as a potent tool in the fight against infectious disease epidemics, both past and future. To tackle the biggest pandemic of the century, virologists and public health workers have teamed up with data scientists, computer programmers, physicists, and mathematicians to use the massive amounts of "big data" created and used to fight the COVID-19 pandemic. New data science methods for dealing with COVID-19 are discussed in this article. These methods include digital contact tracing, diagnosis, policymaking, resource allocation, risk assessment, social media analytics, mental health surveillance, medication repurposing, and drug development [3]. Using the COVID-19 pandemic as an example, we examine the advantages and disadvantages of data science approaches to epidemiological research, compare the new methods to traditional methods, and talk about what we found.

Big data from human mobility, medical imaging, virology, drug screening, bioinformatics, electronic health records, scientific literature, and ever-increasing computing power have all contributed to the rise of data science approaches [4]. Recent developments, the immense interest from both academics and practitioners, and the paramount importance of data-driven insights have all contributed to data science's meteoric rise in prominence in the fight against the 2019 coronavirus disease (COVID-19) pandemic.

A number of countries have elevated the severe acute respiratory syndrome Coronavirus 2 (SARS-CoV2) pandemic to the level of a top national security concern. If we want to know how this infectious disease will spread and what effects it will have, we need better prediction models for when it will happen. A comparison of ML and soft computing models for COVID-19 outbreak prediction is presented in this research. Preliminary findings from two ML models (ANFIS and MLP) demonstrated strong capacity for long-term prediction generalisation. Based on the findings presented here, and considering how the COVID-19 pandemic has varied from country to country and how complicated it has been, this study recommends ML as a useful method for modelling the outbreak's time series [5]. To showcase machine learning's potential for future study, the current work offers an early benchmarking.

Literature Review

Kewat and Kanojiya [6] proposed new goal of the study paper is to forecast used car prices. The dataset utilized to forecast the price was obtained from Kaggle. They will use supervised machine learning, specifically linear regression, to analyze the dataset, which included variables such as car name, company, year, kilometres driven, fuel type, and price, which is a dependent variable. The pre-processed dataset and linear regression algorithm will aid in determining the accurate price of used cars.

Kikteva et.al [7] In this paper, they investigate the effects of incorporating contextual information for the task and study how the reconstruction of the actual debate contributions—that is, utterances containing pronouns, ellipses, and fuzzy language—into full-fledged propositions that can be understood without context. They engage in extremely

intricate impromptu discussions with over ten speakers covering an extensive range of subjects. Unlike what we first thought, we discover that context only enhances predictions when it is utilized in conjunction with propositions; reconstruction does not improve predictions at all.

Hall et.al [8] This study uses statistical analysis to examine the literature in the developing subject of welfare and poverty forecasts derived from the integration of satellite images and machine learning. They utilize an integrative review approach to gather important information on variables associated with welfare's capacity for prediction. We discovered that the number of pre-processing stages used, the number of datasets used, the type of welfare indicator targeted, and the AI model chosen are the most significant parameters connected with the predictive capacity of wellbeing. Studies that targeted flow measures (income and consumption) as targets performed worse in forecasting welfare than those that used stock measure indicators (assets), doing 17 percentage points better. Furthermore, we discovered that combining deep learning and machine learning.

Topaloğlu et. Al [9] Three distinct methods are used for this purpose in order to precisely estimate the revenue. Using Random Forest (RF), forecast models are created in the first method following a few basic preparation procedures on the dataset. In the second method, the dataset is examined for outliers using Isolation Forest (IF) and the features that have the greatest impact on revenue prediction quality are accurately selected using minimum Redundancy Maximum Relevance (mRMR). The last method involves first performing the feature selection procedure, after which the dataset is clustered using Density-Based Spatial Clustering and Application with Noise (DBSCAN). Following the completion of these procedures, forecast models are created using RF. The daily revenue of a seller is one of the many variables included in the dataset. Mean Absolute Percentage.

Viswanatha et.al [10] This work presents the development of a tuberculosis prediction model using a machine learning methodology. One of the leading causes of death from an infectious disease that affects the lungs and yet poses a hazard to the human population overall is tuberculosis. After HIV/AIDS, tuberculosis poses a major threat to human health, according to the WHO. The World Health Organization (WHO) estimates that seven to eight million new cases of tuberculosis (TB) are reported worldwide each year, accounting for one-third of the world's population that is infected with the disease. It takes a while to determine whether a patient has the condition because it is hard to distinguish it from the ordinary cold.

Proposed Methodology

Module Description

You can organise your Python code logically using modules. Code is more comprehensible and useful when grouped together into modules. In Python, an object with bindable and referenceable characteristics can be defined as a module. A module is just a Python script file. Modules have the ability to define variables, classes, and functions. Executable code is another type of module.

Example:

The Python code for a module named aname normally resides in a file named aname.py. Here's an example of a simple module, support.py

def print_func(par):

print "Hello: ", par

return

The import Statement

The import has the following syntax:

import module1[, module2[, moduleN]

The module is imported by the interpreter whenever it finds an import statement, provided that the module is on the search path. The interpreter will look through the specified directories in the search path before importing a module. To import the support.py module, for instance, you would need to include the following command at the beginning of the script.

There is a single loading of a module, independent of the number of imports. In the case that numerous imports take place, this will stop the module execution from occurring repeatedly.

Packages in Python

A Python application environment is defined by a package, which is a hierarchical file directory structure that contains modules, sub packages, and sub-sub packages.

Consider a file Pots.py available in Phone directory.

This file has following line of source code −

def Pots ():
print "I'm Pots Phone"
Similar way, we have another two files having different functions with the same name as above –
• Phone/Isdn.py file having function Isdn()
• Phone/G3.py file having function G3()
Now, create one more file __init__.py in Phone directory −
• Phone/__init__.py
To make all of your functions available when you've imported Phone,to put explicit import statements in

__init__.py as follows −
from Pots import Pots
from Isdn import Isdn
from G3 import G3
After you add these lines to __init__.py, you have all of these classes available when you import the Phone package.

# Now import your Phone Package.
import Phone
Phone.Pots()
Phone.Isdn()
Phone.G3()
RESULT:
I'm Pots Phone
I'm 3G Phone
I'm ISDN Phone
While we've only used one function per file in this example, you're free to include as many as you like. Additionally, those files can be used to define various Python classes, which can then be used to create packages.

Python Files I/O

This chapter covers all the basic I/O functions available in Python.

Printing to the Screen

The simplest way to produce output is using the print statement where you can pass zero or more expressions separated by commas. This function converts the expressions you pass into a string and writes the result to standard output as follows − print "Python is really a great language,", "isn't it?"

Result

Python is really a great language, isn't it?

Reading Keyboard Input

Python provides two built-in functions to read a line of text from standard input, which by default comes from the keyboard. These functions are −

- raw_input
- input

Theraw_inputFunction:
The raw_input([prompt]) function reads one line from standard input and returns it as a string (removing the trailing newline).
str = raw_input ("Enter your input: ");
print "Received input is: ", str
This prompts you to enter any string and it would display same string on the screen. When I typed “Hello Python!", its output is like this −
Enter your input: Hello Python

Received input is: Hello Python
The input Function:
The input([prompt]) function is equivalent to raw_input, except that it assumes the input is a valid Python expression and returns the evaluated result to you.
str = input ("Enter your input: ");
print "Received input is: ", str
This would produce the following result against the entered input −
Enter your input: [x*5 for x in range (2,10,2)]
Received input is: [10, 20, 30, 40]

Dataflow Diagram

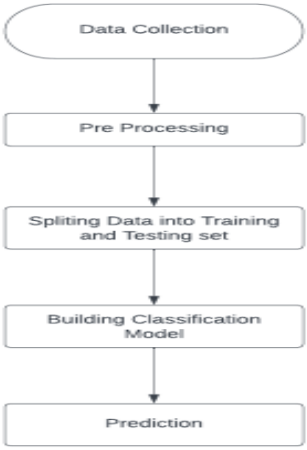

Fig. 1: Data Flow Diagram for Proposed Methodology

Classes and Objects

From the very beginning, Python has been designed to be an object-oriented language. This makes it incredibly simple to create and use classes and objects. Learn the ins and outs of Python's object-oriented programming features with the help of this chapter. In order to understand the fundamentals of object-oriented (OO) programming, those who have never done it before might look into taking an introductory course or reading a tutorial.

Overview of OOP Terminology

- **Class**: A user-defined object prototype that specifies the characteristics that all objects of the class must have. Dot notation allows access to the attributes, which are data members (instance variables and class variables) and methods.
- **Class variable**: A class variable is one that every instance of that class uses. Unlike variables defined in methods of a class, class variables are defined inside the class itself. When compared to instance variables, class variables are much less common.
- **Data member**: A variable that stores information about a class and its objects; it can be an instance variable or a class variable.
- **Function overloading**: The practice of giving a single function control over multiple actions. The kinds of objects or arguments used determine the operation's execution.
- **Instance variable**: A variable that can only be used by the current instance of a class and is defined inside a method.
- **Inheritance**: Passing on a class's traits to its descendants is what inheritance is all about.
- **Instance**: One particular item belonging to a specific class. For instance, the class Circle has instances of objects obj that are members of the class.
- **Instantiation**: The creation of an instance of a class.
- **Method:** A unique kind of function that is defined in a class definition.
- **Operator overloading**: The assignment of more than one function to a particular operator.
- **Creating Classes:** The class statement creates a new class definition. The name of the class immediately follows the keyword class followed by a colon as follows −

class ClassName: 'Optional class documentation string'
class_suite

- The class has a documentation string, which can be accessed via ClassName. __doc__.
- The class_suite consists of all the component statements defining class members, data attributes and functions.

Class Inheritance

Instead of starting from zero when creating a class, it is possible to "derived" from an existing class. Enclosed in parenthesis after the new class name is the list of its parents. With inheritance, you can access and use the properties of the parent class in the child class as if they were declared in the parent class. A child class can also override the parent class's methods and data members.

Syntax

Derived classes are declared much like their parent class; however, a list of base classes to inherit from is given after the class name −

class SubClassName (ParentClass1[, ParentClass2, ...]):

'Optional class documentation string'

class_suite

Overriding Methods

You can always override your parent class methods. One reason for overriding parent's methods is because you may want special or different functionality in your subclass.

Example

class Parent: # define parent class
def myMethod(self):
print 'Calling parent method'
class Child (Parent): # define child class
def myMethod(self):
print 'Calling child method'
c = Child () # instance of child
c.myMethod() # child calls overridden method
When the above code is executed, it produces the following result −
Calling child method

Base Overloading Methods

Following table lists some generic functionality that you can override in your own classes −

SN Method, Description & Sample Call
1 __init__ (self [, args...])
Constructor (with any optional arguments)
Sample Call: obj = className(args)
2 __del__ (self)
Destructor, deletes an object
Sample Call: del obj
3 __repr__ (self)
Evaluatable string representation
Sample Call: repr(obj)
4 __str__ (self)
Printable string representation
Sample Call: str(obj)
5 __cmp__ (self, x)
Object comparison
Sample Call: cmp(obj, x)

Attribute Error

To ensure the safety of certain members, Python adds the class name to the name internally. Attributes like object. _className__attrName are accessible. This should work for you if you replace the last line with the following.

NumPy

" A Python package called NumPy adds support for big, multi-dimensional arrays and matrices and a plethora of high-level mathematical functions to work with them. Jim Hugunin and a number of other developers established Numeric, a programming language that is comparable to NumPy. By heavily modifying Numeric and adding features from the competitor Numarray, Travis Oliphant built NumPy in 2005. You can get it for free because it's an open source library.

Pandas

Pandas is also a library or a data analysis tool in python which is written in python program- ming language. It is mostly used for data analysis and data manipulation. Useful for data structures and time series as well. Python has numerous potentials uses in various domains, including but not limited to: economics, recommendation systems (e.g., Spotify, Netflix, and Amazon), advertising, analytics, neurology, statistics, and stock prediction. Data can be analyzed in pandas in two ways -

Data frames - In this data is two dimensional and consist of multiple series. Data is always represented in rectangular table.

Series - In this data is one dimensional and consist of single list with index.

Matplotlib

" Matplotlib is a plotting library for the Python programming language and its numerical math ematics extension NumPy" [11]. Matlab provides an application that is used in graphical user interface tool kits. Another such libraby is pylab which is almost same as MATLAB.

It is a library for 2D graphics, it finds its application in web application servers, graphical user interface toolkit and shell. Below is the example of a basic plot in python.

Sklearn

Among Python's machine learning libraries, Scikit-learn (Sklearn) stands head and shoulders above the others. Classification, regression, clustering, and dimensionality reduction are just a few of the effective techniques provided by this Python package for machine learning and statistical modelling. The foundation of this Python package is Matplotlib, NumPy, and SciPy.

What is Scikit-Learn (Sklearn)

Scikit-learn (Sklearn) is Python's best and most powerful machine learning package. Classification, regression, clustering, and dimensionality reduction are just a few of the effective techniques provided by this Python package for machine learning and statistical modelling. The foundation of this Python package is Matplotlib, NumPy, and SciPy.

Installation

If you already installed NumPy and Scipy, following are the two easiest ways to install scikit-learn –

1 Using pip

Following command can be used to install scikit-learn via pip −

pip install -U scikit-learn

2 Using conda

Following command can be used to install scikit-learn via conda −conda install scikit-learn.

On the other hand, if NumPy and Scipy is not yet installed on your Python workstation then, you can install them by using either pip or conda.

Another option to use scikit-learn is to use Python distributions like Canopy and Anaconda because they both ship the latest version of scikit-learers.

Features

Rather than focusing on loading, manipulating and summarising data, Scikit-learn library is focused on modeling the data. Some of the most popular groups of models provided by Sklearn are as follows –

- **Supervised Learning algorithms:** Almost all the popular supervised learning algorithms, like Linear Regression, Support Vector Machine (SVM), Decision Tree etc., are the part of scikit-learn.
- **Unsupervised Learning algorithms:** On the other hand, it also has all the popular unsupervised learning algorithms from clustering, factor analysis, PCA (Principal Component Analysis) to unsupervised neural networks.
- **Clustering:** This model is used for grouping unlabeled data.
- **Cross Validation:** It is used to check the accuracy of supervised models on unseen data.
- **Dimensionality Reduction:** It is used for reducing the number of attributes in data which can be further used for summarisation, visualisation and feature selection.
- **Ensemble methods:** As name suggest, it is used for combining the predictions of multiple supervised models.
- **Feature extraction:** It is used to extract the features from data to define the attributes in image and text data.
- **Dataset Loading:** A collection of data is called dataset. It is having the following two components:
- **Features:** The variables of data are called its features. They are also known as predictors, inputs or attributes.
- **Feature Matrix:** It is the collection of features, in case there are more than one.
- **Feature Names:** It is the list of all the names of the features.
- **Response:** It is the output variable that basically depends upon the feature variables. They are also known as target, label or output.
- **Response Vector:** It is used to represent response column. Generally, we have just one response column.
- **Target Names:** It represents the possible values taken by a response vector

System Specification

Predicting the individual mortality rate is equally important as estimating the maximum number of infected patients, which is the more challenging of the two. When trying to predict how many patients will need critical care unit beds, the death rate is crucial. It is critical for countries to model the mortality rate in order to plan for new facilities. In order to improve upon the current conventional epidemiological models, we suggested a system that would include machine learning models. This would allow for longer lead times and more accuracy.

Table 1: Hardware Specification & Software Specification

Processor	Pentium IV
Hard Disk	512GB or more
RAM	8GB or more
Operating System	Windows 7, 10, 11, Linux
Programming Language	Python, HTML, CSS, Js Bootstrap, Django
IDE/Workbench	Pycharm, visual Studio code

Python has dynamic semantics and is a high-level, interpreted, object-oriented programming language. Its dynamic typing, binding, and high-level built-in data structures make it a great choice for scripting, rapid application development, and glueing together different components. The emphasis on readability in Python's basic, easy-to-learn syntax lowers the cost of programme maintenance. Because of its modules and packages, Python promotes code reuse and programme modularity. Python and its vast standard library are freely distributable and available in source or binary form for all major platforms.

Python Features

Python is known for its straightforward syntax, simplified structure, and small number of keywords. Python code is easier to see and understand because of its better coding standards. The source code of Python is not too difficult to keep up to date. The majority of Python's library is interoperable with UNIX, Windows, and Mac OS X, making it very portable. Interactive testing and debugging of code snippets is made possible using Python's support for an interactive mode. Because it uses the same interface across all supported hardware platforms, portable Python is compatible with a broad range of devices.

Extendable

The Python interpreter can have low-level modules added to it. To make their tools more efficient, programmers might add or customise these modules.

Databases

Python provides interfaces to all major commercial databases.

GUI Programming

Python allows for the creation and porting of graphical user interface programmes to a wide variety of platforms, libraries, and operating systems, including Mac OS X, Windows MFC, and Unix's X Window system.

Object-Oriented Approach

In Python, the object-oriented approach is a crucial feature. This essentially means that Python understands object encapsulation and class encapsulation, which makes programmes more efficient overall.

Highly Dynamic

When it comes to modern programming languages, Python is among the most dynamic options. You can save time and increase efficiency by not having to specify the variable's type while coding.

Extensive Array of Libraries

Python comes inbuilt with many libraries that can be imported at any instance and be used in a specific program.

Open Source and Free

Anyone can make and contribute to Python because it is an open-source programming language. Any operating system, whether Windows, Mac, or Lin, may use Python, and it's free to download and use.

Research and Disscussion

System Design

While the level of enthusiasm for each idea has fluctuated over the past year, they have all managed to remain relevant. With each, the software designer has a solid base upon which to build more advanced design techniques. The foundation for "getting it right" is provided by basic design principles. Design models that detail the data structures, system architecture, interface, and components are created from the software requirements model during the design process. Before going to the next stage of software development, every design product is checked for quality.

Input Design

Keeping the procedure simple, minimising time, and managing the amount of dataset as input are the main goals of the input design. Security is ensured via the input's design. Input design will consider the following steps:

- The dataset should be given as input.
- The dataset should be arranged.
- Methods for preparing input validations.

Output Design

The output is considered high-quality if it satisfies the user's requirements and provides clear information. The output design process decides how the data will be shown for quick reference. A well-planned and methodical approach is required when designing computer output; not only must the correct output be created, but every aspect of the output must be built in a way that the user will discover the system to be easy to use and effective.

Dataset Design

This phase contains the attributes of the dataset which are maintained in the database table. The dataset collection can be of two types namely train dataset and test dataset.

Feasibility Study

For example, to make sure a project is technically and legally possible and economically justified, one can conduct a feasibility study to find out if a concept is viable. A developer can anticipate the project's success and the suggested system's utility based on its feasibility through a feasibility study. It has an effect on the company's efficiency, its capacity to satisfy customers, and the way it uses its resources. As a result, a feasibility study is typically conducted on newly submitted applications prior to their approval for development. Three key consideration involved in the feasibility analysis are,

- Technical Feasibility
- Operational Feasibility

- Economic Feasibility

Technical Feasibility

In this stage, the organization's technical resources are the centre of attention. This aids businesses in figuring out if their technical resources are sufficient and if their ideas can be turned into a functional system model. An further step in determining the system's technical feasibility is to assess its software, hardware, and other technical needs.

Operational Feasibility

During this stage, a study is conducted to assess the extent to which the project can fulfil the needs of the organisation. An operational feasibility assessment will also look at how well a project plan meets the needs of the system development phase.

Economic Feasibility

Before allocating financial resources, this phase usually entails a cost-benefit analysis of the project to help the organisation establish the project's sustainability. Improving the project's legitimacy, it also acts as an impartial evaluation. It aids decision-makers in figuring out how much money the organisation would save thanks to the planned project.

System Testing

The goal of the system testing phase of implementation is to verify the system's correctness and efficiency prior to going live. The system's success depends on testing. The underlying premise of system testing is that the objective can only be satisfied if every component is functioning as intended. The testing of a system entails putting it through its paces, both during user training and during actual operation. After the user reviews the generated system, any necessary adjustments are done. Using different types of data, the constructed system is tested during the testing phase. Mistakes are identified and fixed during the testing process. We have also made note of the corrections for future reference.

Unit Testing

With unit testing, the emphasis is on verifying the tiniest possible piece of code, component, or module. In order to find bugs within the module's boundaries, control pathways are evaluated using the component-level design description. With a narrow focus, unit testing is able to contain both the relative complexity of tests and the problems they reveal. The data structures and internal processing logic of a component are the primary targets of the unit test. In most cases, this is seen as supplementary to the coding process. Prior to starting to code, you can complete the design of the unit tests.

Black Box Testing

The primary goal of black box testing, also known as behavioural testing, is to ensure that the programme meets its functional requirements. All of a program's functional requirements can be derived from this testing's set of input conditions. Deriving test cases by dividing a program's input and output, this technique focuses on the software's information domain.

White Box Testing

White box testing, sometimes known as glass box testing, is a method of creating test cases that derives them from the control structures that are described in component level design. In order to guarantee that all programme statements have been performed at least once and that all logical conditions have been exercised, this test case is derived.

Integration Testing

Integration testing is a methodical approach to building software architecture with the purpose of conducting mistakes related to interface. An iterative technique to building the software architecture is top-down integration testing. Beginning with the primary control module and working one's way down the control hierarchy, modules are integrated. The process of building and testing with atomic modules starts with bottom-up integration testing. Processing for components below a specific level is constantly available due to the bottom-up integration.

Validation Testing

Once integration testing is complete and all software components have been tested individually, the programme is combined as a package and validation testing can begin. Actions and results that the user can see and interact with are the primary targets of the testing. The testing has been carried out under various conditions, such as ensuring that the

function characteristic meets the specification, and an error or deviation has been found. Users themselves perform the alpha and beta testing on the developer site.

Logistic Regression

Coding

```
from sklearn.linear_model import LogisticRegression model
= LogisticRegression()
#Fit the model
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
#Score/Accuracy
acc_logreg=model.score(x_test,      y_test)*100
acc_logreg
```

Conclusion

The pharmaceutical industry makes use of AI technologies, such as ML algorithms and deep learning techniques, in its day-to-day operations. There have been many disagreements on the use of ML approaches in areas of medicine and drug development, particularly with regard to image analysis and omics data. As an alternative to more conventional techniques, ML models trained on known data can predict the structure of compounds, such as PPT inhibitors and macrocycles, which can be useful in the medical field. Furthermore, due to the high success rate in clinical trials, deep learning models can be seen as chemical structure and QSAR models extracted from pharmaceutical data. These models are relevant for molecules with suitable attributes. To reacquire its formidable data-mining skills, artificial intelligence has advanced to the point where it can now be used in computer-aided medication creation.

Reference

Bacchi, S., Teoh, S.C., Lam, L., Schultz, D., Casson, R.J., & Chan, W. (2023). Bias, coronavirus, nationality, gender and neurology article citation count prediction with machine learning. *Neurology Perspectives*, *3*(1), 1-3.

Rastogi, R., Goyal, A., Rastogi, A.R., & Gupta, N. (2023). A Comparative Intelligent Environmental Analysis of Air-Pollution in COVID: Application of IoT and AI Using ML in a Study Conducted at the North Indian Zone. *Innovative Engineering with AI Applications*, 189-207.

Jelinek, C., Orozco-Bersoza, L., Pini, S., Melloy, S., O'Connell, S., Shaikh, H., & Laghi, F. (2023). A Home-based, Remotely Monitored Program to Improve Physical Activity in Patients with Long COVID. *In A64. Post-Covid19 Rehabilitation*, A2169-A2169. American Thoracic Society.

Fitria, T.N. (2023). Lexical Meaning of Medical Terms Found in English Book End of Days: Predictions and Prophecies about the End of the World. *Vivid: Journal of Language and Literature*, *12*(2), 153-162.

Hajiabbasi, A. (2023). The Role of Emotional Regulationin in Predicting Academic Depression in female high school During the corona virus. *Journal of Psycho Research and Behavioral Studies*, *1*(1).

Kewat, A., & Kanojiya, N. (2023). Price Prediction of Used Cars using Linear Regression. *Journal of Online Engineering Education*, *14*(1s), 27-31.

Kikteva, Z., Trautsch, A., Katzer, P., Oest, M., Herbold, S., & Hautli, A. (2023). On the Impact of Reconstruction and Context for Argument Prediction in Natural Debate. In *Proceedings of the 10th Workshop on Argument Mining*, 100-106.

Hall, O., Dompae, F., Wahab, I., & Dzanku, F.M. (2023). A review of machine learning and satellite imagery for poverty prediction: Implications for development research and applications. *Journal of International Development*, 1753-1768.

Topaloğlu, G., Kalaycı, T.A., Pekel, K., & Akay, M.F. (2023). Revenue forecast models using hybrid intelligent methods. *International Journal of Mathematics and Computer in Engineering*, *2*(1), 117-122.

Viswanatha, V., Ramachandra, A.C., Togaleri, A.R., & Gowda, N.S. (2023). Tuberculosis Prediction using KNN Algorithm. *International Journal of Engineering and Management Research*, *13*(4), 58-71.