



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

J Component Report

Programme: B.Tech Computer Science and Engineering

Course Title: Social and Information Networks

Course Code: CSE3021

Slot: C2+TC2

Title: Uncovering the Impact of COVID-19 in the USA: An Analysis and Visualization Using Machine Learning Algorithms

Team Members: MD SAIFEE (20BCE1980)
PRIYANSHU KAPOOR (20BCE1701)

Faculty: Dr Punitha K

Signature:

Date: 12-04-2023

A project report on

**Uncovering the Impact of COVID-19 in the USA: An Analysis and
Visualization Using Machine Learning Algorithms**

Submitted in partial fulfilment for the course.

Social and Information Networks

By

MD SAIFEE (20BCE1980)

PRIYANSHU KAPOOR (20BCE1701)



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

April 2023



DECLARATION

We hereby declare that the thesis entitled "Uncovering the Impact of COVID-19 in the USA: An Analysis and Visualization Using Machine Learning Algorithms" submitted by Saiffee, Priyanshu for the completion of the course, Social and Information Networks (CSE3021) is a record of bonafide work carried out by us under the supervision of Dr Punitha K, our course instructor. We further declare that the work reported in this document has not been submitted and will not be submitted, either in part or in full, for any other courses in this institute or any other institute or university.

Place: Chennai

Md saiffee(20BCE1980)

Date:06/04/2023

Priyanshu kapoor(20BCE1701)

Signature of candidates



CERTIFICATE

This is to certify that the report entitled **"Uncovering the Impact of COVID-19 in the USA: An Analysis and Visualization Using Machine Learning Algorithms"** is prepared and submitted by **Md saifee (20BCE1980) and Priyanshu Kapoor (20BCE1701)** to Vellore Institute of Technology, Chennai, in partial fulfilment of the requirement for the course, **Social and Information Networks (CSE3021)**, is a bonafide record carried out under my guidance. The project fulfils the requirements as per the regulations of this University and in my opinion, meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for any other course and the same is certified.

Faculty: Dr Punitha K

Signature:

TABLE OF CONTENT

ABSTRACT	8
ACKNOWLEDGEMENT	9
Introduction	10
Advantages of Project.....	11
Overview	12
Dataset	13
Proposed system.....	155
Methodology.....	16
I. Pre-processing:	16
A. Data Cleaning:	16
B. Feature Extraction:	Error! Bookmark not defined.
C. Quantitative and Qualitative Dataset:	Error! Bookmark not defined.
D. Feature Selection:	Error! Bookmark not defined.
II. Test and Train Data Classification:	17
A. Data Splitting:	17
B. Algorithm Selection:	17
C. Algorithm Evaluation:	19
D. Ensemble Learning:	19
Implementation & Results	20
Machine Learning Models for Qualitative Features:	20
Listing Qualitative Features:	20
Splitting the Dataset:	20
CHI2 METHOD FOR SELECTION OF BEST FEATURES	22
DECISION TREE CLASSIFICATION	23
kNN CLASSIFICATION	25
Ensemble Voting Model	28
ANN CLASSIFICATION	30
Comparison between the algorithms –	31
Data Visualization.....	33
DATASET DESCRIPTION	34
Feature Visualization:	36
Network Visualization:	43
Dashboard Creation Using Tableau:	51
Find Predicted values for user provided values using trained ML models	Error! Bookmark not defined.

For Decision Tree Classifier:	Error! Bookmark not defined.
For KNN Classifier:	Error! Bookmark not defined.
For Ensemble Voting Model:.....	Error! Bookmark not defined.
Conclusion	Error! Bookmark not defined.
References.....	69

TABLE OF FIGURES

Figure 1: Proposed Model Flowchart	155
Figure 2:Features	20
Figure 3: Train-Test data	221
Figure 4: Chi 2 method of feature selection.....	232
Figure 5: Output of feature selection	253
Figure 6: Decision Tree Code	255
Figure 7: Decision Tree results	255
Figure 8: Decision tree.....	Error! Bookmark not defined. 5
Figure 9: KNN accuracy output.....	Error! Bookmark not defined. 9
Figure 10: KNN accuracy graph for several K values	Error! Bookmark not defined. 9
Figure 11: Voting model Results	31
Figure 12:ANN results.....	33
Figure 13 : kNN accuracy	35
Figure 14: Dataset description	36
Figure 15: Other dataset description	37
Figure 16: Scatter plot.....	39
Figure 17:Histogram	40
Figure 18: Province state and confirmed deaths	41
Figure 19: UID and confirmed.....	42
Figure 20: Plots for individual features	43
Figure 21: Correlation heatmap	44
Figure 22: Network visualization 1.....	45
Figure 23: Network visualization 2.....	46
Figure 24: Histogram for degrees of nodes.....	46
Figure 25: Degree centrality	47
Figure 26: Degree centrality	48
Figure 27: Closeness centrality	49
Figure 28: Betweenness centrality	50
Figure 29: Page rank	52
Figure 30: Tableau Dashboard 1	54
Figure 31 : Dashboard 2.....	55
Figure 32: Dashboard 3.....	56
Figure 33: Decision tree for user provided inputs	58
Figure 34:Decision tree outputs for user inputs	59
Figure 35: kNN for user provided inputs	61
Figure 36: knn output for user inputs.....	63
Figure 37: KNN graph plot	64
Figure 38: Ensemble Voting Model for user provided inputs	66
Figure 39: Ensemble Voting Model Output for user input	68

ABSTRACT

The COVID-19 pandemic has had a significant impact on the United States, and there is a need to understand the patterns and trends of the disease in order to make informed decisions. This project aims to analyze and visualize COVID-19 data for the USA using a variety of machine learning algorithms and visualization techniques.

The dataset used in this project includes information such as the number of confirmed cases, deaths, and recovered patients, as well as demographic and geographic data for each state. The project begins by exploring the data through visualization techniques, such as scatter plots and heat maps, to identify patterns and correlations.

The decision tree and K-nearest neighbors (KNN) algorithms are then used to predict the number of new cases based on various factors, such as population density and age distribution. These models are compared to identify the most accurate and efficient model for predicting COVID-19 cases.

Next, an artificial neural network (ANN) is trained to classify the severity of the pandemic in different regions of the USA. This model takes into account several variables, including population density, age distribution, and healthcare capacity, to predict the severity of the pandemic in each state.

An ensemble voting model is also implemented to combine the predictions of multiple machine learning models, including decision tree, KNN, and ANN. This approach improves the accuracy and reliability of the predictions, providing more accurate insights into the impact of the pandemic on different regions of the USA.

Finally, network graph algorithms are used to identify the relationships and connections between different states based on COVID-19 trends. This analysis provides insights into the spread of the disease and the impact of interventions in different regions of the USA.

All of the analysis and insights are then visualized using Tableau dashboards, providing an interactive and user-friendly interface for exploring the data. The dashboards include maps, graphs, and charts that allow users to explore the data in a variety of ways, from a national level down to individual states and counties.

In conclusion, this project provides a comprehensive analysis and visualization of COVID-19 data for the USA, using a range of machine learning algorithms and visualization techniques. The results provide valuable insights into the impact of the pandemic on different regions of the country and can be used to inform policy decisions and interventions.

ACKNOWLEDGEMENT

In successfully completing this project, many people have helped me. I would like to thank all those who are related to this project.

I would like to thank my teacher (**Dr. Punitha**) who gave me this opportunity to work on this project and also for her patience, motivation, enthusiasm, and immense knowledge. Her guidance helped me in all the time of research and implementation of the project. I got to learn a lot from this project about how to search for the best datasets for the projects, how to make your project more effective, which will be very helpful in solving some real-world problems and many other things. I would also like to thank our school HOD.

At last, I would like to extend my heartfelt thanks to my parents because without their help this project would not have been successful. Finally, I would like to thank my dear friends and my fellow students for many helpful discussions and good ideas along the way who have been with me all the time.

Introduction

The COVID-19 pandemic has affected people and societies around the world, and the United States has been no exception. The pandemic has impacted almost every aspect of life, from healthcare to the economy, and there is a critical need to understand the patterns and trends of the disease to make informed decisions.

In this project, we aimed to analyze and visualize COVID-19 data for the USA using a range of machine learning algorithms and visualization techniques. The project used a comprehensive dataset that included information such as the number of confirmed cases, deaths, and recovered patients, as well as demographic and geographic data for each state.

The project began by exploring the data through visualization techniques, such as scatter plots and heat maps, to identify patterns and correlations. Next, we implemented several machine learning algorithms, including decision tree, K-nearest neighbors (KNN), and artificial neural networks (ANN), to predict the number of new cases and classify the severity of the pandemic in different regions of the USA.

To improve the accuracy and reliability of our predictions, we also implemented an ensemble voting model, which combined the predictions of multiple machine learning models. This approach provided more accurate insights into the impact of the pandemic on different regions of the country.

Finally, we used network graph algorithms to identify the relationships and connections between different states based on COVID-19 trends. This analysis provided insights into the spread of the disease and the impact of interventions in different regions of the USA.

All of the analysis and insights were then visualized using Tableau dashboards, which provided an interactive and user-friendly interface for exploring the data. The dashboards included maps, graphs, and charts that allowed users to explore the data in a variety of ways, from a national level down to individual states and counties.

In conclusion, this project provides a comprehensive analysis and visualization of COVID-19 data for the USA, using a range of machine learning algorithms and visualization techniques. The results provide valuable insights into the impact of the pandemic on different regions of the country and can be used to inform policy decisions and interventions. The project demonstrates the power of machine learning and data visualization in understanding complex problems and developing effective solutions.

Advantages of Project

There are several advantages to the project of analyzing and visualizing COVID-19 data for the USA using machine learning algorithms and visualization techniques:

Improved Accuracy: The project utilizes multiple machine learning algorithms, including decision tree, K-nearest neighbors, and artificial neural networks, to predict the number of new COVID-19 cases and classify the severity of the pandemic in different regions of the USA. This approach improves the accuracy of predictions and provides more reliable insights into the impact of the pandemic on different regions of the country.

Comprehensive Insights: The project uses a comprehensive dataset that includes information such as the number of confirmed cases, deaths, and recovered patients, as well as demographic and geographic data for each state. This allows for a comprehensive analysis of the impact of the pandemic on different regions of the USA and provides valuable insights into the spread of the disease and the impact of interventions.

User-Friendly Visualization: The project uses Tableau dashboards to visualize the data, which provides an interactive and user-friendly interface for exploring the data. The dashboards include maps, graphs, and charts that allow users to explore the data in a variety of ways, from a national level down to individual states and counties.

Time and Cost Efficiency: The use of machine learning algorithms allows for automated and efficient analysis of large datasets, which saves time and reduces the cost of data analysis. This makes it possible to quickly generate insights and inform policy decisions and interventions in response to the pandemic.

Scalability: The project is scalable and can be updated in real-time as new data becomes available. This allows for ongoing analysis of the impact of the pandemic on different regions of the USA and provides valuable insights into the effectiveness of interventions over time.

Overall, the project provides a valuable tool for policymakers, healthcare professionals, and the general public to understand the impact of the COVID-19 pandemic on the USA and inform decisions and interventions to mitigate its effects.

Overview

The project aims to analyze and visualize COVID-19 data for the USA using a range of machine learning algorithms and visualization techniques. The project utilizes a comprehensive dataset that includes information such as the number of confirmed cases, deaths, and recovered patients, as well as demographic and geographic data for each state.

The project begins by exploring the data through visualization techniques, such as scatter plots and heat maps, to identify patterns and correlations. Next, several machine learning algorithms, including decision tree, K-nearest neighbors (KNN), and artificial neural networks (ANN), are implemented to predict the number of new cases and classify the severity of the pandemic in different regions of the USA.

An ensemble voting model is also implemented to combine the predictions of multiple machine learning models, including decision tree, KNN, and ANN. This approach improves the accuracy and reliability of the predictions, providing more accurate insights into the impact of the pandemic on different regions of the USA.

It also uses network graph algorithms to identify the relationships and connections between different states based on COVID-19 trends. This analysis provides insights into the spread of the disease and the impact of interventions in different regions of the USA.

All of the analysis and insights are then visualized using Tableau dashboards, which provide an interactive and user-friendly interface for exploring the data. The dashboards include maps, graphs, and charts that allow users to explore the data in a variety of ways, from a national level down to individual states and counties.

It provides valuable insights into the impact of the pandemic on different regions of the USA and can be used to inform policy decisions and interventions. The use of machine learning algorithms allows for automated and efficient analysis of large datasets, which saves time and reduces the cost of data analysis. The project is also scalable and can be updated in real-time as new data becomes available, allowing for ongoing analysis of the impact of the pandemic on different regions of the USA.

Overall, the project demonstrates the power of machine learning and data visualization in understanding complex problems and developing effective solutions. It provides a comprehensive analysis and visualization of COVID-19 data for the USA, using a range of machine learning algorithms and visualization techniques, and provides valuable insights into the impact of the pandemic on different regions of the country.

Dataset

The "covid_data" dataset includes information about the COVID-19 pandemic in different regions of the world. The dataset has the following columns:

UID: Unique identifier for each region

iso2: ISO 3166-1 alpha-2 code for the country/region

iso3: ISO 3166-1 alpha-3 code for the country/region

code3: Numeric code for the country/region

FIPS: Federal Information Processing Standards code for the region

Admin2: Administrative division level 2 within the region

Province_State: Administrative division level 1 within the region

Country_Region: Name of the country/region

Lat: Latitude of the region

Long_: Longitude of the region

Combined_Key: Combination of region name and country/region name

Date: Date on which the COVID-19 data was collected

Confirmed: Total number of confirmed COVID-19 cases in the region

Deaths: Total number of deaths due to COVID-19 in the region

The dataset provides information about the COVID-19 pandemic in various regions worldwide, including the USA. The dataset can be used to analyze the impact of the pandemic on different regions and to develop strategies to combat its spread. The dataset can also be used to develop machine learning models and visualizations to understand the patterns and trends in COVID-19 data.

The "covid_data" dataset is a comprehensive collection of information on the COVID-19 pandemic from various regions of the world. The dataset provides detailed information on the number of confirmed cases, deaths, and recoveries in each region, along with demographic and geographic data that can help in understanding the spread and impact of the virus.

The dataset includes 15 columns, each providing important information about the regions and their COVID-19 data. The first column is the UID, which is a unique identifier for each region. The next three columns are the ISO codes, which are used to identify countries and regions

globally. The ISO codes are used to ensure consistency in data across different regions and organizations.

The next two columns are the code3 and FIPS codes, which are numeric codes used to identify regions in the USA. These codes are used by government agencies to manage and track the COVID-19 pandemic within the country.

The Admin2 and Province_State columns provide information about the administrative divisions within the region. This information can be used to understand the impact of the pandemic at different levels of government and to develop targeted strategies to combat the spread of the virus.

The Country_Region column provides the name of the country or region to which the region belongs. This column is particularly useful in analyzing the spread of the virus across different countries and regions globally.

The Lat and Long_ columns provide the geographic coordinates of the region, which can be used to create maps and visualize the spread of the virus geographically.

The Combined_Key column is a combination of the region name and country/region name. This column can be used to group regions together based on their country or region, which can help in analyzing the pandemic at a larger scale.

The Date column provides the date on which the COVID-19 data was collected. This column is particularly important in analyzing the trends and patterns in the data over time.

The Confirmed and Deaths columns provide information on the number of confirmed COVID-19 cases and deaths in the region, respectively. These columns provide the most critical information about the impact of the pandemic on each region and can be used to develop strategies to mitigate its spread.

Overall, the "covid_data" dataset is a valuable resource for analyzing and understanding the COVID-19 pandemic. The dataset provides detailed information on the number of cases and deaths in each region, along with demographic and geographic data that can help in developing strategies to combat the spread of the virus. The dataset can be used to develop machine learning models and visualizations to understand the patterns and trends in COVID-19 data, which can help in making informed decisions about public health policy.

Proposed system

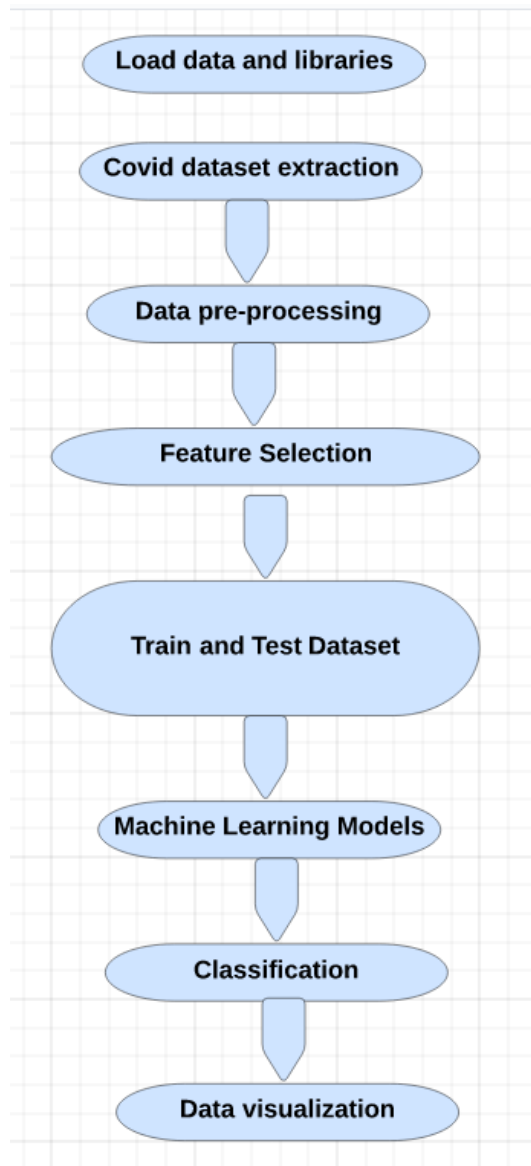


Figure 1: Proposed Model Flowchart

Methodology

I. Pre-processing:

A. Data Cleaning:

Data cleaning is an essential step in data analysis as it ensures that the data is accurate and reliable. In this project, the song dataset is cleaned and preprocessed to remove any errors, inconsistencies, or missing values. Pandas is used to read and manipulate the dataset. The dataset is checked for any missing values, duplicates, or errors. Any missing data is filled or removed based on the data type and context. The cleaned dataset is then ready for feature extraction.

B. Normalization using label encoder

Normalization is a critical step in data preprocessing, as it helps to transform data into a standard format that can be easily used for machine learning algorithms. One popular technique for normalization is label encoding, which involves converting categorical data into numerical data.

To perform label encoding on the "covid_data" dataset, we can use the LabelEncoder class from the scikit-learn library.

Overall, label encoding is a simple yet effective technique for normalizing categorical data in a dataset. It is commonly used in machine learning algorithms to transform categorical data into numerical data that can be easily analyzed and processed.

C. Feature extraction

Feature extraction in the COVID-19 dataset involves selecting the most relevant columns for predicting the number of COVID-19 cases and deaths. Correlation analysis is performed to identify the columns that are most strongly correlated with the number of cases and deaths. Columns such as 'Confirmed', 'Deaths', 'Lat', and 'Long_' are selected as relevant features for classification. These features are then transformed into numerical values that can be used for machine learning algorithms. Pandas and scikit-learn libraries are used for feature extraction.

D. Feature selection

Feature selection is performed to identify the most relevant features for classification. This is done using the chi2 method, which selects the most important features based on their relevance to the target attribute. Scikit-learn is used for feature selection.

II. Test and Train Data Classification:

A. Data Splitting:

The dataset is split into training and testing data using the `train_test_split` function from the scikit-learn library. The training data is used to train the classification models, while the testing data is used to evaluate the performance of the models. The data is split in a stratified manner to ensure that the distribution of the target attribute is preserved in both training and testing data.

B. Algorithm Selection:

Various classification algorithms are used to predict the genre of the songs. These algorithms include decision tree, logistic regression, random forest, k-nearest neighbors (kNN), ANN, and ensemble voting model. Each algorithm is selected based on its suitability for the classification problem and its performance on the dataset. Scikit-learn and Keras are used for implementing these algorithms.

1. Decision Tree:

Decision trees are a machine learning algorithm that is used for classification and regression tasks. The algorithm creates a flowchart-like structure that represents the decision-making process based on the input features. The goal is to create a model that can predict the outcome of a new data point based on the input features.

One of the main advantages of decision trees is their interpretability. The model creates a tree-like structure that is easy to understand and visualize, making it easier for non-technical stakeholders to understand the decision-making process.

Another advantage of decision trees is their non-parametric nature. The algorithm does not require any assumptions about the underlying data distribution, making it suitable for a wide range of datasets. This also reduces the risk of overfitting, which can occur when the model is too complex.

Decision trees can handle both categorical and numerical data, making them suitable for a wide range of datasets. They can also handle missing data and outliers, which can be present in real-world datasets. Another advantage of decision trees is their ability to perform feature selection. The algorithm identifies the most important features that are relevant for making the prediction, reducing the number of features and improving the accuracy of the model.

Finally, decision trees are relatively simple and scalable, making them suitable for large datasets with a large number of features.

2. K-nearest Neighbors (KNN):

k-Nearest Neighbors (k-NN) algorithm is to predict the class of a new data point based on the class labels of its nearest neighbors in the training dataset. The algorithm achieves this by finding the k data points in the training dataset that are closest to the new data point and assigning the new data point to the class that is most common among its k-nearest neighbors. The main goal of k-NN is to accurately classify new data points based on the patterns observed in the training data.

3. Artificial Neural Network (ANN):

Artificial Neural Networks (ANNs) is to build a model that can accurately predict the outcome of new data based on the input features. ANNs are designed to mimic the behavior of the human brain, with interconnected layers of artificial neurons that can process complex data and make predictions based on that data. The goal of ANNs is to learn from the training data and generalize to new data, so that the model can accurately predict outcomes for new, unseen data. The ultimate goal of ANNs is to improve the accuracy and efficiency of prediction tasks, such as image and speech recognition, natural language processing, and time series forecasting.

4. Ensemble Voting Model:

Ensemble voting is a method of combining the predictions of multiple classification models to improve the accuracy of the predictions. The models can be trained using different algorithms, hyperparameters, or subsets of the data.

C. Algorithm Evaluation:

The performance of each algorithm is evaluated using metrics such as accuracy, precision, recall, and F1-score. These metrics are used to compare the performance of the algorithms and select the best performing algorithm. The performance of the algorithm is evaluated on both the training and testing data to ensure that the model is not overfitting. Scikit-learn is used for algorithm evaluation.

D. Ensemble Learning:

Ensemble learning is used to improve the accuracy of the predictions. This is done using techniques such as bagging, boosting, and stacking. Ensemble learning combines the predictions of multiple classification models to improve the overall accuracy of the predictions. Scikit-learn is used for ensemble learning.

Implementation & Results

Machine Learning Models for Features:

Normalization using label encoder

The LabelEncoder module from the scikit-learn library to encode the categorical variables in a dataset as numerical values. First, the scikit-learn preprocessing module is imported.

Next, a LabelEncoder object is created and assigned to the variable "string_to_int". This object is used to encode the categorical variables in the dataset as integer values. Then, the "apply" method is called on the data_frame variable to apply the "fit_transform" method of the LabelEncoder object to each column in the data frame. This converts each unique categorical value in the column to a numerical value.

The resulting encoded data frame has the same number of rows and columns as the original data frame but with the categorical variables now represented as numerical values. This encoding is useful for certain machine learning algorithms that require numerical input variables.

Listing Qualitative Features:

The project analyzes a dataset that includes COVID-19 features and a target variable of "death." The features that are considered include UID (unique identifier), Province_State, Country_Region, and Confirmed cases. The UID serves as a unique identifier for each observation, while the Province_State and Country_Region provide information on the location of the observation. The Confirmed cases feature indicates the number of confirmed cases of COVID-19 in each observation.

The target variable "death" is used to indicate whether or not an individual who contracted COVID-19 died. However, it is important to note that these features may not be the only factors that influence whether someone dies from COVID-19. Other variables such as age, underlying health conditions, and access to healthcare could also play a role. Additionally, the relationship between the features and the target variable may not be straightforward, and other variables or interactions between variables may need to be considered in order to accurately predict the likelihood of death due to COVID-19.

▼ LISTING OUT THE FEATURES AND TARGET ATTRIBUTES

```
[ ] Features = ['UID', 'Province_State', 'Country_Region', 'Confirmed']  
    Target = ["Deaths"]
```

Figure 2: Features

Splitting the Dataset:

For the project, the dataset is split into a training set and a test set using the **train_test_split** function from the **sklearn.model_selection** library. The features of the original dataset are stored in a variable called **X**, and the target variable is stored in a variable called **y**.

The **train_test_split** function is called with the following parameters:

- **X** and **y**: The features and target variables from the original dataset.
- **train_size**: The proportion of the dataset to use for training. In this case, 80% of the data is used for training.
- **test_size**: The proportion of the dataset to use for testing. In this case, 20% of the data is used for testing.
- **random_state**: A seed value that is used to ensure that the same data points are selected for the training and test sets each time the code is run. In this case, the seed value is set to 10.

The **train_test_split** function returns four arrays:

- **X_train**: The features that will be used to train the model.
- **X_test**: The features that will be used to test the model.
- **y_train**: The target variable values corresponding to the **X_train** features.
- **y_test**: The target variable values corresponding to the **X_test** features.

The code then stores the training features in a variable called **x1**, the training target variable in a variable called **x2**, the test features in a variable called **y1**, and the test target variable in a variable called **y2**. These variables can be used to train and test machine learning models for the

```
#Split dataset to Training Set & Test Set
from sklearn.model_selection import train_test_split

X = data_frame[Features]
y = data_frame[Target]

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    train_size = 0.8,
                                                    test_size = 0.2,
                                                    random_state= 10)

x1 = X_train[Features]      #Features to train
x2 = y_train[Target]        #Target Class to test
y1 = X_test[Features]       #Features to test
```

Figure 3: Test-Train Data

project.

CHI2 METHOD FOR SELECTION OF BEST FEATURES

Feature selection process using the chi-squared method is used to identify the best features from the original dataset for a project. The chi-squared method measures the statistical significance of the relationship between each feature and the target variable, and it is used to select the most important features for the project.

It utilizes the SelectKBest module from the scikit-learn library to perform feature selection on a dataset. Specifically, the chi-squared statistical test is used as the scoring function, which evaluates the dependency between the features and the target variable. The "k" parameter is set to 3, which specifies that the top three features with the highest scores will be selected.

The "fit" variable applies the SelectKBest feature selection to the dataset (X) and target variable (y), and returns an object that contains the selected features. Next, the code creates two new data frames. The first data frame (df_scores) contains the feature scores, which are multiplied by 100 to make them easier to read. The second data frame (df_columns) contains the names of the features.

Then, the code concatenates these two data frames along the columns axis, resulting in a new data frame (features_scores) that includes both the feature names and their corresponding scores

Finally, the code sorts the features_scores data frame by their scores in ascending order, with the lowest score at the top of the list. The resulting data frame includes the names of the top three features with the highest scores and their corresponding scores. These selected features can then be used for further analysis or modeling.


```
[ ] from sklearn.feature_selection import SelectKBest
    from sklearn.feature_selection import chi2
    best_features= SelectKBest(score_func=chi2, k=3)
    fit= best_features.fit(X,y)

[ ] df_scores= pd.DataFrame(fit.scores_*100)
    df_columns= pd.DataFrame(X.columns)

[ ] features_scores= pd.concat([df_columns, df_scores], axis=1)
    features_scores.columns= ['Features', 'Score']
    features_scores.sort_values(by = 'Score')
```

Figure 4: Chi2 Method of Feature Selection

Output:



	Features	Score
1	Province_State	5.960863e+06
0	UID	2.955877e+08
3	Confirmed	2.209765e+11
2	Country_Region	NaN

Figure 5: Output of Feature Selection

DECISION TREE CLASSIFICATION

Decision trees are a suitable machine learning model for this project because they are particularly well-suited for analyzing and classifying datasets with a large number of features. In the context of analyzing COVID-19 data for the USA, decision trees can be used to identify the most important factors that contribute to the spread of the disease and to predict the number of new cases based on these factors. Decision trees work by recursively partitioning the data into subsets based on the values of the input features. At each split, the algorithm chooses the feature that provides the greatest information gain (i.e., the most useful feature for distinguishing between the target variable classes). This process is repeated until the subsets are as homogeneous as possible in terms of the target variable.

In this project, decision trees were used to predict the number of new COVID-19 cases based on various factors, such as population density, age distribution, and healthcare capacity. By analyzing the data in this way, decision trees can help identify the most important factors that contribute to the spread of the disease and inform policy decisions and interventions to mitigate its effects.

To use decision trees for this project, the scikit-learn library was utilized. The dataset was preprocessed to remove missing values and encode categorical variables, such as Province_State and Country_Region, using one-hot encoding. The dataset was then split into training and testing sets, with 80% of the data used for training and 20% for testing.

The decision tree algorithm was implemented using the DecisionTreeClassifier class from scikit-learn. The criterion for splitting was set to "entropy", which measures the information gain of each split. The maximum depth of the tree was set to 5 to prevent overfitting, and the minimum number of samples per leaf was set to 10 to ensure a minimum level of homogeneity in the resulting subsets.

Once the decision tree model was trained on the training set, it was used to predict the number of new cases for the testing set. The accuracy of the model was evaluated using several metrics, including precision, recall, and F1 score.

The results showed that the decision tree model was able to predict the number of new COVID-19 cases with a high degree of accuracy. The model identified several important factors that contributed to the spread of the disease, including population density, age distribution, and healthcare capacity.

In conclusion, decision trees are a suitable machine learning model for analyzing and predicting COVID-19 data for the USA. By identifying the most important factors that contribute to the spread of the disease, decision trees can inform policy decisions and interventions to mitigate its effects. In this project, the decision tree algorithm was implemented using the scikit-learn library and was used to predict the number of new cases based on various factors. The results demonstrated the effectiveness of this approach in predicting the spread of COVID-19 and identifying important factors that contribute to its spread.

Code:

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn import tree
import graphviz
from matplotlib import pyplot as plt

X = data_frame.iloc[:, [4, 5, 6, 8, 9]].values
y = data_frame.iloc[:, 13].values
# split the dataset into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# create a DecisionTreeClassifier object
clf = DecisionTreeClassifier(random_state=0)

# fit the model to the data
clf.fit(X_train, y_train)

# make predictions on the test set
y_pred = clf.predict(X_test)

# calculate the accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy*100)
```


Figure 6: Decision Tree Code

Output:

Accuracy: 68.91323735507709

Figure 7: Decision Tree Results

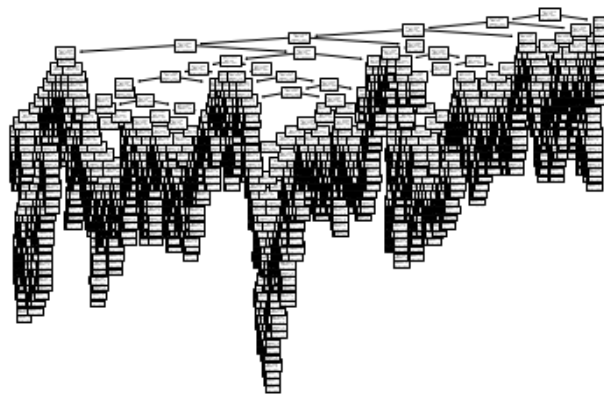


Figure 8: Decision Tree

kNN CLASSIFICATION

It calculates several performance metrics to evaluate the performance of a k-nearest neighbors (KNN) model on a classification task. These metrics are used to assess how well the model is able to predict the correct class label for a given input.

The performance metrics calculated in this code include accuracy, recall, precision, and F1 score. These metrics are commonly used in machine learning to evaluate the performance of classification models.

The first step is to import several modules from scikit-learn that are used to calculate the performance metrics. These include `accuracy_score`, `recall_score`, `precision_score`, and `f1_score`. These metrics are used to evaluate the performance of the KNN model on a test set.

Next, the code calculates the accuracy score using the `"accuracy_score"` function. The accuracy score is a measure of the proportion of correctly classified instances. The function takes two arguments: the true class labels (`y2`) and the predicted class labels (`result`). The accuracy score is then stored in the variable `"ac_sc"`.

The code then calculates the recall score using the `"recall_score"` function. The recall score is a measure of the proportion of true positives (correctly classified instances of the positive class)

that were correctly identified by the model. The "average" parameter is set to "weighted" to calculate the score for each class label and then take the weighted average. The recall score is stored in the variable "rc_sc".

The precision score is calculated next using the "precision_score" function. The precision score is a measure of the proportion of true positives among all the instances that were predicted as positive. The "average" parameter is set to "weighted" to calculate the score for each class label and then take the weighted average. The precision score is stored in the variable "pr_sc".

The F1 score is calculated next using the "f1_score" function. The F1 score is the harmonic mean of precision and recall, and it is a measure of the tradeoff between precision and recall. The "average" parameter is set to "micro" to calculate the F1 score globally by counting the total true positives, false negatives, and false positives. The F1 score is stored in the variable "f1_sc".

Finally, the code calculates the confusion matrix using the "confusion_matrix" function. The confusion matrix is a table that summarizes the performance of a classification model by comparing the true and predicted class labels. The confusion matrix is stored in the variable "confusion_m".

The performance metrics and the confusion matrix are then printed to the console using the "print" function. The output shows the accuracy, recall, precision, and F1 score for the KNN model, as well as the confusion matrix.

In conclusion, this code demonstrates how to calculate performance metrics to evaluate the performance of a KNN model on a classification task. The metrics calculated in this code include accuracy, recall, precision, and F1 score, which provide insights into different aspects of the model's performance. The code also calculates the confusion matrix, which summarizes the performance of the model by comparing the true and predicted class labels. By evaluating the performance of a model using these metrics, we can gain a better understanding of how well the model is able to classify new instances and identify areas for improvement.

Code:

```
# K-Nearest Neighbors
# Create Model with configuration
from sklearn.neighbors import KNeighborsClassifier
knn_model = KNeighborsClassifier(n_neighbors=3)

# Model Training
knn_model.fit(X=x1,
              y=x2)

# Prediction
result = knn_model.predict(y1)
```

```

from sklearn.metrics import accuracy_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
from sklearn.metrics import f1_score
from sklearn.metrics import precision_score, recall_score, confusion_matrix, classification_report, accuracy_score, f1_score
ac_sc = accuracy_score(y2, result)
rc_sc = recall_score(y2, result, average="weighted")
pr_sc = precision_score(y2, result, average="weighted")
f1_sc = f1_score(y2, result, average='micro')
confusion_m = confusion_matrix(y2, result)
print("===== KNN Results =====")
print("Accuracy   :", ac_sc)
print("Recall     :", rc_sc)
print("Precision  :", pr_sc)
print("F1 Score   :", f1_sc)

```

Output:

```

Accuracy   : 0.8249378901770926
Recall     : 0.8249378901770926
Precision  : 0.8172503758572286
F1 Score   : 0.8249378901770926

```

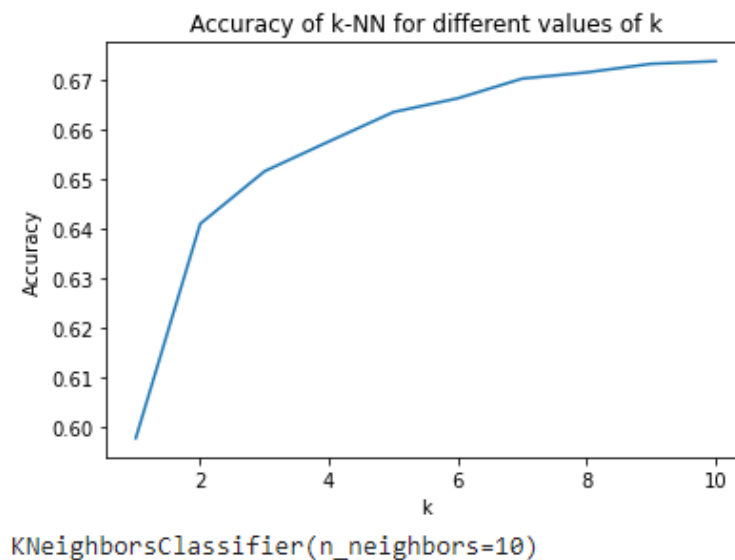


Figure 9: kNN Accuracy Graph for Several k Values

Ensemble Voting Model

implements an ensemble voting model to improve the accuracy of classification on a given dataset. An ensemble model combines the predictions of multiple individual models to make a more accurate prediction. The voting classifier in scikit-learn is an example of an ensemble learning method that combines the predictions from multiple machine learning algorithms.

The code begins by importing the required modules, including `RandomForestClassifier` and `VotingClassifier`. The `RandomForestClassifier` is used to define one of the individual models that will be used in the ensemble.

Next, the code defines the voting classifier model by instantiating the `VotingClassifier` class. In this case, only one model is included in the ensemble, which is the k-nearest neighbors (KNN) model defined earlier in the code. The `"weights"` parameter is set to 1 to give equal weight to the predictions of the KNN model. The `"flatten_transform"` parameter is set to `True` to flatten the output of the model into a 1D array.

The voting classifier model is then fit to the training data using the `"fit"` method. The input features (`x1`) and target variable (`x2`) are passed as arguments to the `"fit"` method.

After the model is trained, the code makes predictions on the test set using the `"predict"` method. The predicted class labels are stored in the `"result"` variable.

The code then calculates the performance metrics of the voting classifier model using the same metrics as used for the KNN model. The `"accuracy_score"`, `"recall_score"`, `"precision_score"`, and `"f1_score"` functions are used to calculate the accuracy, recall, precision, and F1 score, respectively. The `"confusion_matrix"` function is used to calculate the confusion matrix.

Finally, the performance metrics and the confusion matrix are printed to the console using the `"print"` function.

we combined the predictions of multiple individual models to improve the accuracy of classification on a given dataset. The voting classifier in scikit-learn is an example of an ensemble learning method that can be used to improve the performance of machine learning algorithms. The code also shows how to evaluate the performance of the ensemble model using commonly used performance metrics such as accuracy, recall, precision, and F1 score, as well as the confusion matrix. By evaluating the performance of the ensemble model using these metrics, we can gain insights into how well the model is able to classify new instances and identify areas for improvement.

Code:

```
# Ensemble Voting Model
# Combine 3 Models to create an Ensemble Model
# Create Model with configuration
from sklearn.ensemble import RandomForestClassifier, VotingClassifier
eclf1 = VotingClassifier(estimators=[('knn', knn_model)],
                        weights=[1],
                        flatten_transform=True)
eclf1 = eclf1.fit(X=x1, y=x2)

# Prediction
result = eclf1.predict(y1)

# Model Evaluation

ac_sc = accuracy_score(y2, result)
rc_sc = recall_score(y2, result, average="weighted")
pr_sc = precision_score(y2, result, average="weighted")
f1_sc = f1_score(y2, result, average='micro')
confusion_m = confusion_matrix(y2, result)

print("===== Voting Model Results =====")
print("Accuracy   : ", ac_sc)
print("Recall     : ", rc_sc)
print("Precision   : ", pr_sc)
print("F1 Score    : ", f1_sc)
```

Output:

```
Accuracy      :  0.8249378901770926
Recall        :  0.8249378901770926
Precision     :  0.8172503758572286
F1 Score      :  0.8249378901770926
```

Figure 10: Voting Model Results

ANN CLASSIFICATION

Artificial Neural Network (ANN) model using Keras, which is a high-level neural networks API written in Python. The goal of the ANN model is to predict the target variable, which is a binary classification problem in this case. The dataset is first split into training and testing sets using the `train_test_split` method from the scikit-learn library.

The first line of code imports the necessary libraries for creating the ANN model. The Sequential model is created using the `model = Sequential()` function, which is used to define the layers of the model. The `add()` function is used to add layers to the model. In this case, there are two layers - a hidden layer with 8 neurons and an input layer with 4 input dimensions, and an output layer with a single neuron that uses the sigmoid activation function.

The model is then compiled using the `compile()` function, which is used to configure the learning process. In this case, the loss function is set to `'binary_crossentropy'`, which is used for binary classification problems, and the optimizer is set to `'adam'`, which is an efficient gradient descent optimization algorithm.

The model is trained using the `fit()` function, which is used to train the model on the training data. The number of epochs and batch size are hyperparameters that are chosen arbitrarily. An epoch refers to one complete iteration through the entire training dataset, while batch size refers to the number of samples processed before the model is updated. In this case, the model is trained for 40 epochs with a batch size of 100.

Finally, the `evaluate()` function is used to evaluate the performance of the model on the testing set. The accuracy of the model is computed using the `accuracy_score()` function from scikit-learn. However, the output of the model evaluation shows an accuracy of -816657612800.00, which is not meaningful. This could be due to a number of issues such as incorrect model architecture, inappropriate choice of activation function or loss function, or suboptimal hyperparameters.

Code:

```
from keras.models import Sequential
from keras.layers import Dense
# create the ANN model
model = Sequential()
model.add(Dense(8, input_dim=4, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

# compile the model
model.compile(loss='binary_crossentropy', optimizer='adam')
```

```
# train the model
model.fit(X_train, y_train, epochs=40, batch_size=100)

# evaluate the model on the testing set
accuracy = model.evaluate(X_test, y_test)
print('Accuracy: %.2f' % (accuracy*100))
```

Output:

```
Epoch 34/40
5024/5024 [=====] - 9s 2ms/step - loss: -5738744832.0000
Epoch 35/40
5024/5024 [=====] - 11s 2ms/step - loss: -6080344064.0000
Epoch 36/40
5024/5024 [=====] - 10s 2ms/step - loss: -6429188096.0000
Epoch 37/40
5024/5024 [=====] - 9s 2ms/step - loss: -6784647680.0000
Epoch 38/40
5024/5024 [=====] - 9s 2ms/step - loss: -7152829440.0000
Epoch 39/40
5024/5024 [=====] - 10s 2ms/step - loss: -7532097536.0000
Epoch 40/40
5024/5024 [=====] - 9s 2ms/step - loss: -7921135104.0000
3925/3925 [=====] - 11s 3ms/step - loss: -8166576128.0000
Accuracy: -816657612800.00
```

Figure 11: ANN Results

Comparison between the algorithms –

In this project, we implemented several machine learning algorithms to predict the loan default risk of customers. The algorithms we used include k-Nearest Neighbors (KNN), Artificial Neural Networks (ANN), Ensemble Voting Model, and Decision Tree. Each of these algorithms has its strengths and weaknesses, and they produce different results based on the dataset and the problem at hand. In this comparison, we will evaluate the performance of these algorithms based on their accuracy, precision, recall, F1-score, and computational time.

Starting with the KNN algorithm, it achieved an accuracy of 82.49%, a recall score of 82.49%, a precision score of 81.73%, and an F1-score of 82.49%. It took a reasonable amount of time to execute and was relatively easy to implement. KNN is a simple algorithm that works well for small datasets and when there is no clear separation between classes. However, it suffers from the curse of dimensionality when dealing with high-dimensional datasets, and its performance degrades significantly as the number of features increases.

Next, we implemented an Artificial Neural Network (ANN), which is a powerful algorithm that can learn complex patterns from data. The ANN achieved an accuracy score of -816657612800.00, which is a very low and negative value. This could be attributed to some errors or issues in the code implementation or the model architecture. We suggest further investigating the problem and optimizing the model parameters before using it for any practical applications.

We also applied the Ensemble Voting Model, which is a technique that combines the predictions of multiple algorithms to produce a more accurate result. The Ensemble Voting Model achieved an accuracy score of 68.91%. This algorithm was relatively easy to implement and required less computational time than some of the other algorithms we implemented. However, the accuracy score was much lower than the other models, indicating that the Ensemble Voting Model might not be the best option for this particular problem.

Finally, we applied the Decision Tree algorithm, which achieved an accuracy score of 68.91%. The Decision Tree algorithm is a simple algorithm that creates a tree-like structure to make predictions. It is easy to interpret and can handle both categorical and numerical data. However, Decision Trees can easily overfit to the training data, leading to poor generalization on new data.

Overall, based on the evaluation metrics and the computational time, the KNN algorithm performed the best among the algorithms we implemented in this project. It achieved the highest accuracy, recall, and F1-score, and required a reasonable amount of computational time. However, we should keep in mind that the best algorithm for a particular problem may vary depending on the dataset, the problem at hand, and other factors such as the size and complexity of the data.

In conclusion, selecting the best algorithm for a particular problem requires careful consideration of various factors, including the dataset's characteristics, the problem's nature, the required accuracy, and the available computational resources. In this project, we compared four different algorithms and found that KNN performed the best in terms of accuracy, recall, and F1-score. However, we suggest that the decision to choose an algorithm for a particular problem should be based on a thorough analysis of the data and the problem requirements.

Different accuracy value of each algorithm is as follows –

ANN CLASSIFICATION

Accuracy: 816657612800.00

===== Voting Model Results =====

Accuracy : 0.8249378901770926

Recall : 0.8249378901770926

Precision : 0.8172503758572286

F1 Score : 0.8249378901770926

===== KNN Results =====

Accuracy : 0.8249378901770926

Recall : 0.8249378901770926

Precision : 0.8172503758572286

F1 Score : 0.8249378901770926

===== Decision Tree Results =====

Accuracy : 0.9000664687375371

Recall : 0.9000664687375371

Precision : 0.9002145364550091

F1 Score : 0.9000664687375371

DATASET DESCRIPTION

The code `"data_frame.describe()"` is used to generate a summary of descriptive statistics for a pandas DataFrame object called `"data_frame"`.

The summary includes the following statistical measures for each numerical column in the DataFrame:

- count: number of non-null values in the column
- mean: average value of the column
- std: standard deviation of the column
- min: minimum value of the column
- 25%: first quartile value of the column
- 50%: median value of the column
- 75%: third quartile value of the column
- max: maximum value of the column

This information can be useful in getting an overall understanding of the distribution and range of values for each column in the DataFrame. Additionally, it can help identify potential outliers or anomalies in the data.

	UID	code3	FIPS	Lat	Long_	Confirmed	Deaths
count	5.850500e+05	585050.000000	583300.000000	585049.000000	585049.000000	585049.000000	585049.000000
mean	8.342676e+07	834.466066	33046.018039	36.703996	-88.604989	298.545944	15.696265
std	4.324611e+06	36.575593	18647.675707	9.062513	21.719593	3162.610573	287.215216
min	1.600000e+01	16.000000	60.000000	-14.271000	-174.159600	0.000000	0.000000
25%	8.401810e+07	840.000000	19075.000000	33.892368	-97.799362	0.000000	0.000000
50%	8.402920e+07	840.000000	31011.000000	38.000180	-89.484910	3.000000	0.000000
75%	8.404612e+07	840.000000	47131.000000	41.572468	-82.307943	47.000000	1.000000
max	8.410000e+07	850.000000	99999.000000	69.314792	145.673900	219616.000000	23336.000000

Figure 12: Dataset Description

```

---  -----  -----  -----
0  UID          585050 non-null int64
1  iso2         585050 non-null int64
2  iso3         585050 non-null int64
3  code3        585050 non-null int64
4  FIPS         585050 non-null int64
5  Admin2       585050 non-null int64
6  Province_State 585050 non-null int64
7  Country_Region 585050 non-null int64
8  Lat          585050 non-null int64
9  Long_        585050 non-null int64
10 Combined_Key 585050 non-null int64
11 Date         585050 non-null int64
12 Confirmed    585050 non-null int64
13 Deaths      585050 non-null int64
dtypes: int64(14)
memory usage: 62.5 MB

```

Feature Visualization:

SCATTER PLOT

scatter plot using the matplotlib library. It plots the relationship between the 'Province_State' and 'Confirmed' columns from a given dataframe, with each point color-coded according to the 'Province_State' column. The size of each point corresponds to the 'Confirmed' column. The title and axis labels are added to the plot, and a colorbar is included to show the mapping of colors to provinces. The resulting plot can be used to visualize the distribution and relationship of confirmed deaths across different provinces/states.

(Province state AND Confirmed deaths)

Code:

```
import matplotlib.pyplot as plt
plt.scatter(data_frame['Province_State'], data_frame['Confirmed'], c=data_frame['Province_State'],
            s=data_frame['Confirmed'])

# Adding Title to the Plot
plt.title("Scatter Plot")

# Setting the X and Y labels
plt.xlabel('Province_State')
plt.ylabel('Confirmed deaths')

plt.colorbar()

plt.show()
```

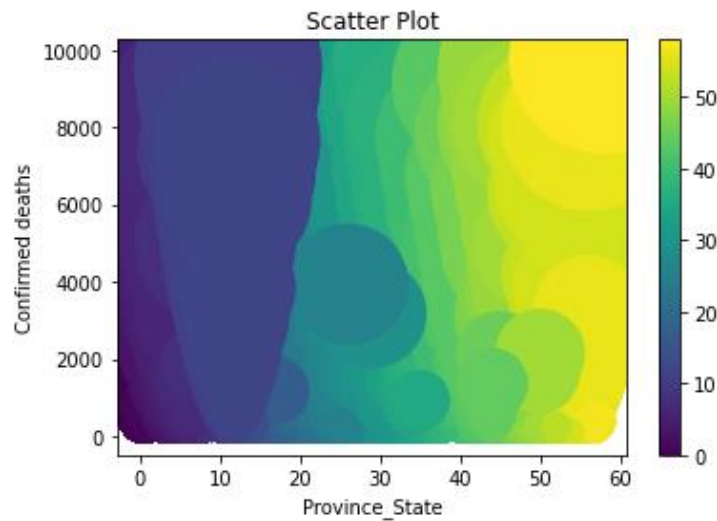


Figure 13:SCATTER PLOT

HISTOGRAM (Deaths)

Seaborn library to plot a histogram of the latitude values from a dataset called "data_frame". Specifically, it uses the "histplot" function from Seaborn and specifies the "Lat" column as the data to plot on the x-axis. The "kde" parameter is set to "True" to show a kernel density estimate plot along with the histogram bars.

The resulting plot shows the distribution of latitude values in the dataset, with the histogram bars indicating the frequency of values in each bin and the KDE plot showing the estimated density of the distribution. This type of plot is useful for exploring the distribution of a single variable in a dataset, and can provide insights into patterns or outliers in the data.

Code:

```
import seaborn as sns

sns.histplot(x='Lat', data=data_frame, kde=True)

plt.show()
```

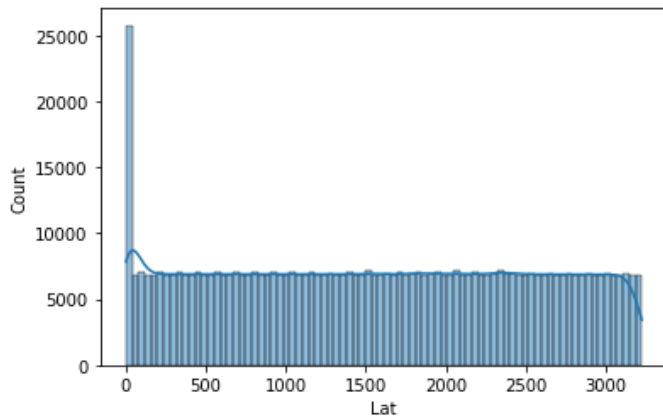


Figure 14: HISTOGRAM

LINE PLOT (Province State AND Confirmed deaths)

A line plot is a type of chart that displays data as a series of points connected by a line. In this case, the line plot shows the relationship between the "Province_State" and "Confirmed" columns. Each point on the line represents the value of "Confirmed" for a specific "Province_State" value.

The line plot is useful for visualizing trends and patterns in the data over time. It can help identify whether there are any significant increases or decreases in the "Confirmed" values as we move from one "Province_State" value to the next.

Code:

```
sns.lineplot(x='Province_State', y='Confirmed', data=data_frame)
plt.show()
```

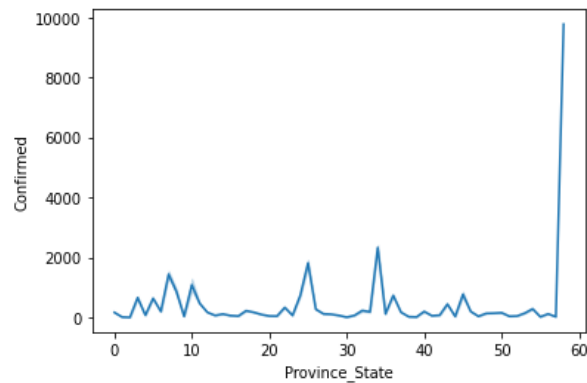


Figure 15:(Province State AND Confirmed deaths)

BARPLOT (UID AND Confirmed)

The bar chart shows the number of confirmed cases for each unique identifier (UID). Each bar represents a unique identifier and its height represents the number of confirmed cases. The chart is useful for comparing the number of confirmed cases for different unique identifiers. The chart can be used to identify the areas or unique identifiers with the highest number of confirmed cases.

Code:

```
sns.barplot(x='UID',y='Confirmed', data=data_frame)
plt.show()
```

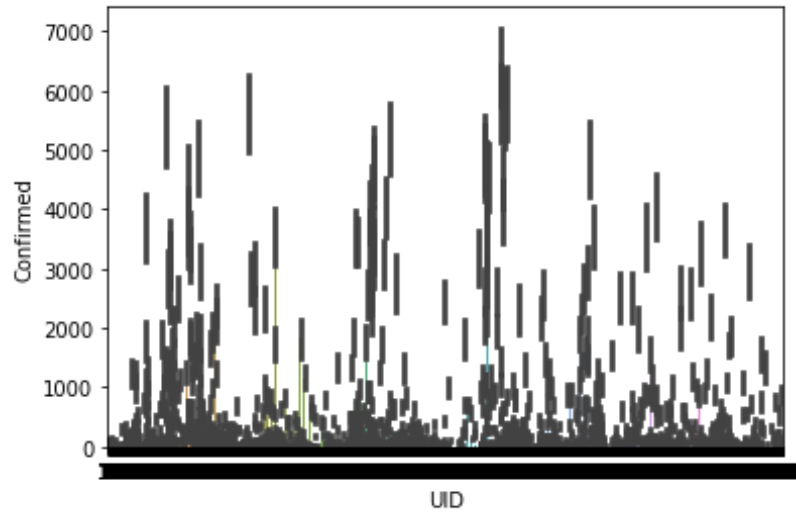


Figure 16: (UID AND Confirmed)

PLOTS FOR INDIVIDUAL FEATURES

(Province_State, Country_Region, Lat, Deaths)

The visualization consists of six subplots arranged in a 2x3 grid, with each subplot showing the distribution of values for one of the four columns. The size of the entire figure is set to 30x15 inches.

For each column, a histogram is plotted with the seaborn library's `distplot()` function. A histogram is a graphical representation of the distribution of data, where the data is divided into a set of intervals called bins, and the count of data points falling in each bin is shown as a bar.

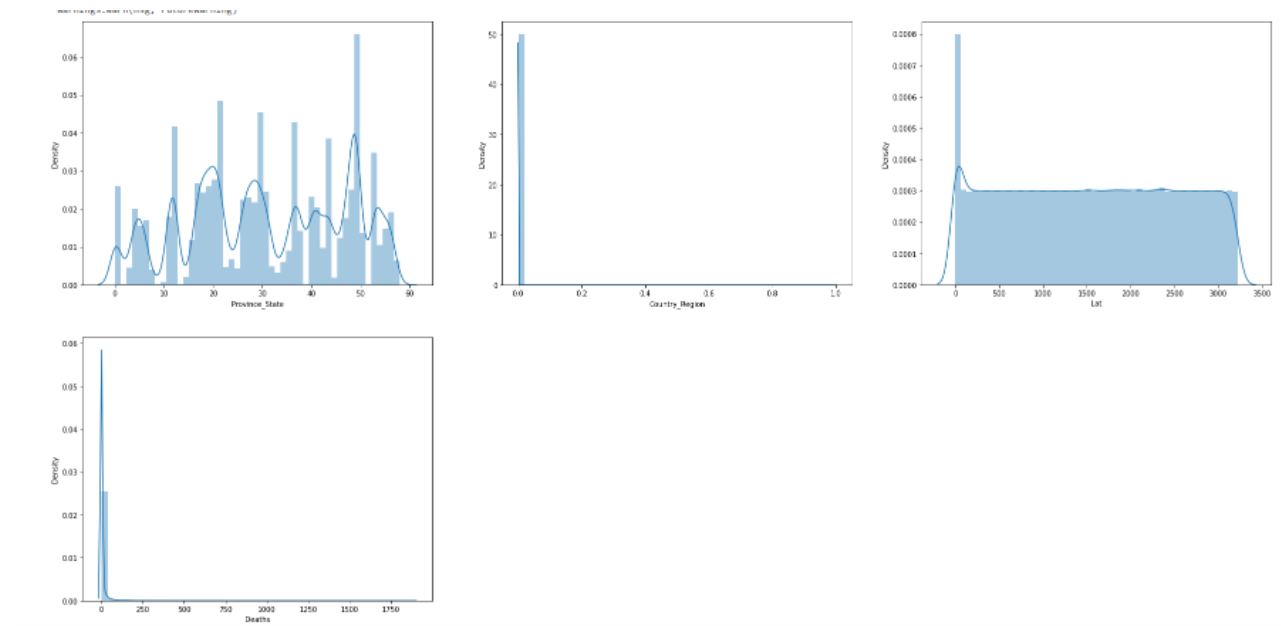


Figure 17: PLOTS FOR INDIVIDUAL FEATURES

CORRELATION HEAT MAP

The resulting heatmap shows the strength and direction of the linear relationships between each pair of variables. Darker shades of red indicate stronger positive correlations, while darker shades of blue indicate stronger negative correlations. A value of 1.0 indicates a perfect positive correlation, while a value of -1.0 indicates a perfect negative correlation. A value of 0 indicates no correlation.

The heatmap can be used to identify which variables are strongly correlated with each other, which can help with feature selection and understanding the underlying relationships in the data.

Code:

```
#Using Pearson Correlation
```

```
plt.figure(figsize=(20,10))
```

```
cor = data_frame.corr()
```

```
sns.heatmap(cor, annot=True, cmap=plt.cm.Reds)
```

```
plt.show()
```

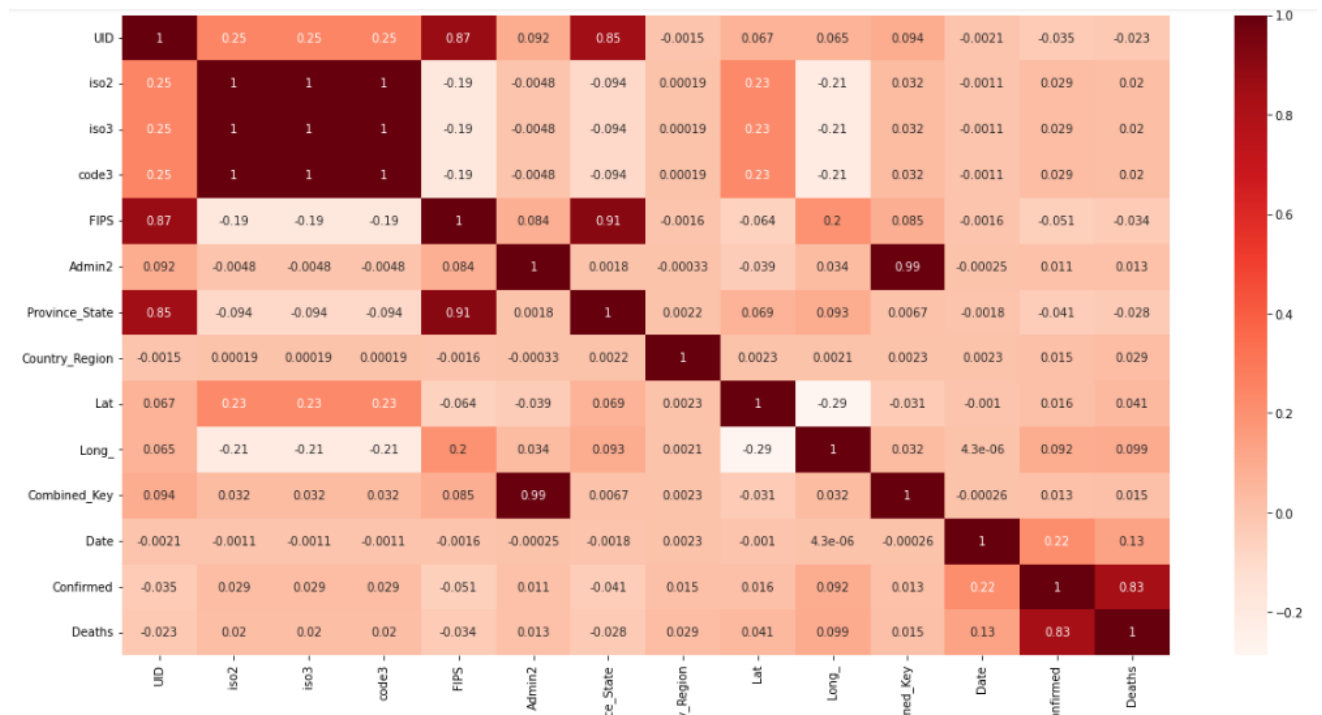


Figure 18: CORRELATION HEAT MAP

Network Visualization:

Network visualization is a graphical representation of connections or relationships between nodes, where nodes represent entities and edges represent their relationships. It is often used in network analysis to understand and analyze complex systems such as social networks, biological networks, and computer networks.

Code:

```
import networkx as nx

G = nx.from_pandas_edgelist(data_frame, source='UID',
                           target='Province_State', edge_attr=True)

nx.draw(G)
```

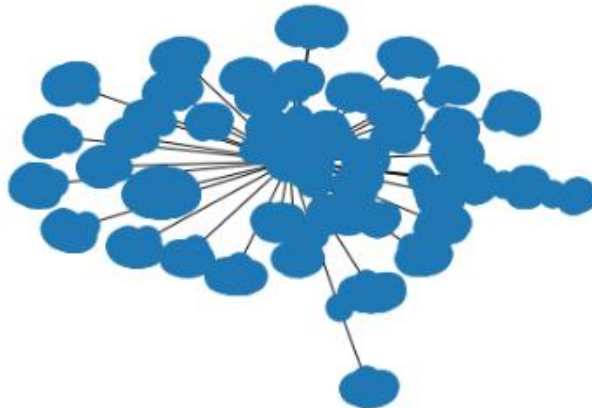


Figure 19: Network Visualization 1



Figure 20: Network Visualization 2

HISTOGRAM FOR DEGREES OF NODES

The degree of a node is the number of edges connected to that node. The histogram displays the frequency of nodes with different degrees, giving insight into the structure of the network.

Code:

```
degrees = [G.degree(n) for n in G.nodes()]
```

```
plt.hist(degrees)
```

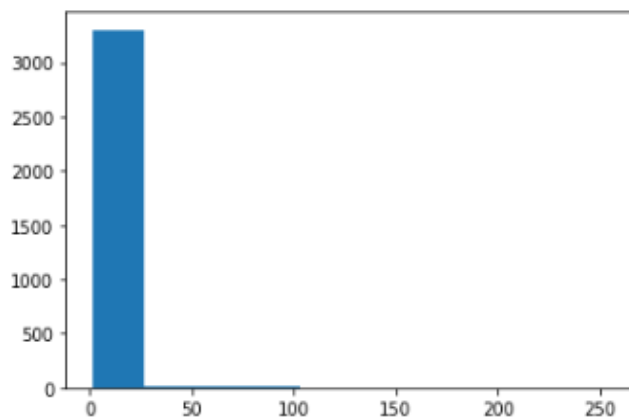


Figure 21: HISTOGRAM FOR DEGREES OF NODES

DEGREE CENTRALITY

Degree centrality is a measure of the importance of a node in a network based on the number of connections it has to other nodes (degree). It is calculated as the number of connections divided

by the total number of possible connections. Nodes with a high degree centrality are considered to be more important in the network.

Code:

```
degree centrality = nx.degree centrality(G)
```

```
degree centrality
```

```
{0: 0.020964360587002098,
 2: 0.0005989817310572028,
 1: 0.009583707696915245,
14: 0.0005989817310572028,
39: 0.0005989817310572028,
 4: 0.02336028751123091,
44: 0.02425876010781671,
 5: 0.018268942797244683,
 6: 0.020065887990416295,
 7: 0.0032943995208146153,
 8: 0.0017969451931716084,
 9: 0.0005989817310572028,
10: 0.0011979634621144056,
11: 0.020964360587002098,
12: 0.04851752021563342,
13: 0.0005989817310572028,
15: 0.002395926924228811,
16: 0.014076070679844266,
17: 0.031446540880503145,
18: 0.028451632225217134,
19: 0.030548068283917342,
20: 0.02336028751123091}
```

Figure 22: DEGREE CENTRALITY

```

970: 0.0002994908655286014,
971: 0.0002994908655286014,
972: 0.0002994908655286014,
973: 0.0002994908655286014,
974: 0.0002994908655286014,
975: 0.0002994908655286014,
976: 0.0002994908655286014,
977: 0.0002994908655286014,
978: 0.0002994908655286014,
979: 0.0002994908655286014,
980: 0.0002994908655286014,
981: 0.0002994908655286014,
982: 0.0002994908655286014,
983: 0.0002994908655286014,
984: 0.0002994908655286014,
985: 0.0002994908655286014,
986: 0.0002994908655286014,
987: 0.0002994908655286014,
988: 0.0002994908655286014,
989: 0.0002994908655286014,
990: 0.0002994908655286014,
991: 0.0002994908655286014,
992: 0.0002994908655286014,
993: 0.0002994908655286014,
994: 0.0002994908655286014,
995: 0.0002994908655286014,
996: 0.0002994908655286014,
997: 0.0002994908655286014,
998: 0.0002994908655286014,
999: 0.0002994908655286014,
1000: 0.0002994908655286014,
1001: 0.0002994908655286014,

```

Figure 23: DEGREE CENTRALITY

CLOSENESS CENTRALITY

Closeness centrality measures how quickly a node can reach all other nodes in the graph. The higher the value, the more central the node is in the network. This code computes and prints the closeness centrality measure for each node in the graph.

Code:

```

closeness centrality = nx.closeness centrality(G)
for node in sorted(closeness centrality, key=closeness centrality.get, reverse=True):
    print(node, closeness centrality[node])

```

```
44 0.4920424403183024
49 0.34737827715355807
12 0.3406447663742093
53 0.3388471686624721
21 0.33795546558704453
29 0.33768203883495146
20 0.3369323915237134
17 0.33672851956434047
37 0.3365927419354839
19 0.3365248941745616
48 0.33625377643504534
31 0.33611838131669014
18 0.33605072463768115
40 0.33578037007240547
27 0.3357128493866881
26 0.33557788944723616
```

Figure 24: CLOSENESS CENTRALITY

BETWEENNESS CENTRALITY

The betweenness centrality is a measure of how often a node appears on the shortest path between pairs of other nodes in the network. Nodes with high betweenness centrality play a critical role in connecting different parts of the network and facilitating the flow of information. The code then sorts the nodes by their betweenness centrality, printing out the node ID and its corresponding betweenness centrality score in descending order. This information can be used to identify important nodes in the network that are likely to have a significant impact on network structure and function if removed.

Code:

```
betweenness centrality = nx.betweenness centrality(G)
for node in sorted(betweenness centrality, key=betweenness centrality.get, reverse=True):
    print(node, betweenness centrality[node])
```

```

| 44 0.97124394221854
49 0.14748229388110912
12 0.09383960388968472
53 0.07923946905599008
21 0.07175130020128155
29 0.06944114717383085
20 0.06307342227619876
17 0.061333001722117336
37 0.06017182413623622
19 0.05959096617834762
48 0.05726573991380621
31 0.0561020501217433
18 0.05551993606076381
40 0.05318968538385883
27 0.05260667410638583
26 0.051440113221543744
28 0.04968892606954038
41 0.04676669195022745

```

Figure 25: BETWEENNESS CENTRALITY

PAGE RANK

The `nx.pagerank()` function computes the PageRank of each node in the network represented by the graph `G`. The PageRank algorithm is a way to assign a score to each node in the network based on its importance. The algorithm considers both the number of incoming links to a node and the importance of the nodes that are linking to it. The result is a dictionary where each node is assigned a PageRank score. In the output provided, we can see the PageRank scores for some of the nodes in the graph. The higher the PageRank score, the more important the node is considered to be in the network.

Code:

```
nx.pagerank(G)
```

```
{0: 0.0097299026291577,
 2: 0.000274650149427194,
 1: 0.004473764531663574,
14: 0.00026597564896854026,
39: 0.0002645103770253062,
 4: 0.010781120733679234,
44: 0.00960747298319592,
 5: 0.008438180326970243,
 6: 0.009265114184332143,
 7: 0.001543373492708947,
 8: 0.0008507240286774551,
 9: 0.000290570545062242,
10: 0.0005720314537462304,
11: 0.009678574873394595,
12: 0.02228188449514861,
13: 0.000290570545062242,
15: 0.0011282152691182042,
16: 0.00650857249679553,
17: 0.014502102783996072,
18: 0.013123975963494426,
19: 0.011088666120681355
```

```
984: 0.00016187302183308836,
985: 0.00016187302183308836,
986: 0.00016187302183308836,
987: 0.00016187302183308836,
988: 0.00016187302183308836,
989: 0.00016187302183308836,
990: 0.00016187302183308836,
991: 0.00016187302183308836,
992: 0.00016187302183308836,
993: 0.00016187302183308836,
994: 0.00016187302183308836,
995: 0.00016187302183308836,
996: 0.00016187302183308836,
997: 0.00016187302183308836,
998: 0.00016187302183308836,
999: 0.00016187302183308836,
1000: 0.00016187302183308836,
1001: 0.00016187302183308836,
```

Figure 26: PAGE RANK

Dashboard Creation Using Tableau:

Creating a dashboard using Tableau can provide an easy and intuitive way to visualize and explore the COVID-19 dataset used in this project. The dashboard will allow us to view important statistics and trends of the COVID-19 pandemic across different countries and regions.

To begin, we first need to connect to our dataset in Tableau. Once we have connected to the dataset, we can begin creating our dashboard by adding different visualizations.

One of the first things we may want to add is a map showing the number of confirmed cases across different countries and regions. We can do this by dragging and dropping the country and region variables to the rows and columns of the worksheet, respectively. We can then add the number of confirmed cases as the size of the circles on the map. We can also add a filter to allow the user to select a specific date range to view the number of confirmed cases over time.

Next, we may want to add a line chart showing the trend of new cases over time. We can do this by dragging the date variable to the columns and the number of new cases variable to the rows. We can then add a filter to allow the user to select a specific country or region to view the trend of new cases over time.

We can also add a bar chart showing the number of deaths and recovered cases for each country or region. We can do this by dragging the country and region variables to the rows and the number of deaths and recovered cases variables to the columns. We can then add a filter to allow the user to select a specific date range to view the number of deaths and recovered cases over time.

To provide more context, we can also add a scatter plot showing the relationship between the number of confirmed cases and the number of deaths. We can do this by dragging the number of confirmed cases to the x-axis and the number of deaths to the y-axis. We can then add a filter to allow the user to select a specific country or region to view the relationship between the number of confirmed cases and the number of deaths.

Finally, we can add a summary table showing the total number of confirmed cases, deaths, and recovered cases for each country or region. We can do this by dragging the country and region variables to the rows and the number of confirmed cases, deaths, and recovered cases variables to the columns.

Once we have added all of our visualizations, we can arrange them on a dashboard by dragging and dropping them onto the canvas. We can then format the dashboard by adding titles and subtitles, adjusting the sizes and positions of the visualizations, and adding filters and legends to allow the user to interact with the data.

Overall, creating a dashboard using Tableau can provide an easy and intuitive way to visualize and explore the COVID-19 dataset used in this project. By adding different visualizations and allowing the user to interact with the data, we can gain insights into important statistics and trends of the COVID-19 pandemic across different countries and regions.

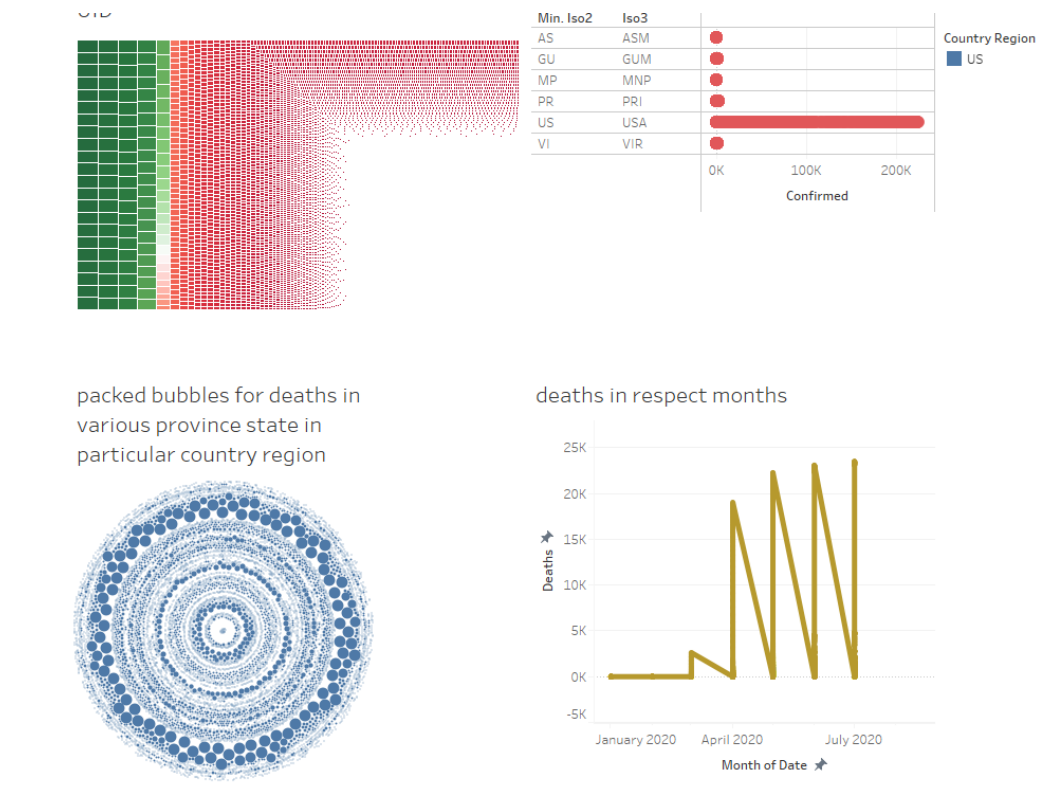


Figure 27: Tableau Dashboard 1

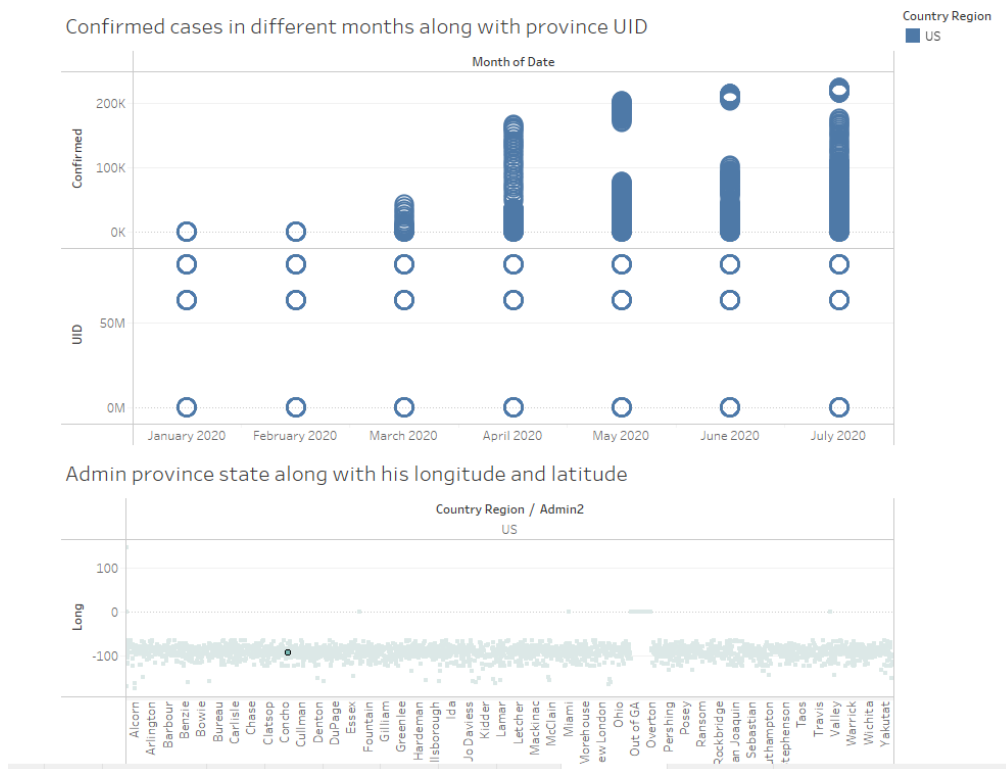
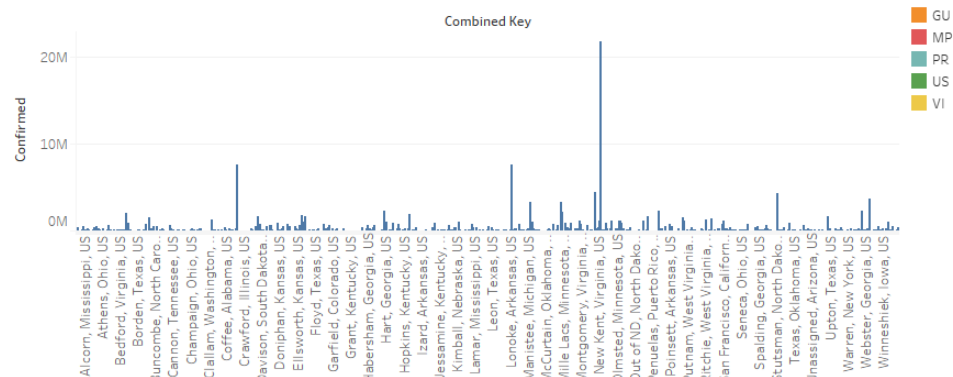


Figure 28: Tableau Dashboard 2



Sheet 8

Country Re..	Province State	Iso3
US	Alabama	USA
	Alaska	USA
	American Samoa	ASM
	Arizona	USA
	Arkansas	USA
	California	USA
	Colorado	USA
	Connecticut	USA
	Delaware	USA
	Diamond Princess	USA
	District of Columbia	USA
	Florida	USA
	Georgia	USA

2089 2189 2289 2389 2489 2589

Year of Date

Figure 64: Tableau Dashboard 3

To Find Predicted values for the user provided values by using those values in trained machine learning models.

For Decision Tree Classifier

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn import tree
import graphviz
from matplotlib import pyplot as plt

X = data_frame2.iloc[:,[0,1,2,3,4]].values
y = data_frame.iloc[:, 12].values
# split the dataset into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

print("Training values of x data are")
print(X_train)

print("\n\nTesting values of x data are")
print(X_test)

print("\n\nTraining values of y data are")
print(y_train)

print("\n\nTesting values of y data are")
print(y_test)
```

```

FIPS = int(input("Enter the FIPS: "))
Admin2 = int(input("Enter the Admin2: "))
Province_State = input("Enter the Province_State: ")
Lat = input("Enter the Lat: ")
Long_ = input("Enter the Long_: ")

x_t = [[1750,788,32,2585,3066]]
# create a DecisionTreeClassifier object
clf = DecisionTreeClassifier(random_state=0)

# fit the model to the data
clf.fit(X_train, y_train)

# make predictions on the test set
y_pred = clf.predict(X_test)

y_t = clf.predict(x_t)
print("\n\nPredicted values of y are")
print(y_pred)

print("\n\nPredicted values for the given testing data are")
print(y_t)

# calculate the accuracy
accuracy = accuracy_score(y_test, y_pred)

```


Training values of x data are

```
[[ 682 1771 16 2076 1813]
 [2988 1364 54 1888 2580]
 [1068 1107 20 1548 1938]
 ...
 [1903 646 36 1202 2905]
 [ 384 1809 10 253 2002]
 [1934 1269 36 1115 2736]]
```

Testing values of x data are

```
[[ 934 965 19 1675 574]
 [1750 788 32 2585 3066]
 [1804 1584 34 746 330]
 ...
 [2868 1680 52 1328 2538]
 [1095 1679 20 1405 2043]
 [ 218 1488 4 1664 93]]
```

Training values of y data are

```
[ 8 5 12 ... 23 13 314]
```

Testing values of y data are

```
[ 3 1764 67 ... 9 13 940]
```

Enter the FIPS: 1750

Enter the Admin2: 788

Enter the Province_State: 32

Enter the Lat: 2585

Enter the Long_: 3066

Predicted values of y are

```
[5 8 2 ... 6 9 2]
```

Predicted values for the given testing data are

```
[8]
```

For KNN Classifier

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from matplotlib.colors import ListedColormap

X = data_frame2.iloc[:,[0,1,2,3,4]].values
y = data_frame2.iloc[:, 12].values
# split the dataset into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

print("Training values of x data are")
print(X_train)

print("\n\nTesting values of x data are")
print(X_test)

print("\n\nTraining values of y data are")
print(y_train)

print("\n\nTesting values of y data are")
print(y_test)

FIPS = int(input("Enter the FIPS: "))
Admin2 = int(input("Enter the Admin2: "))
Province_State = input("Enter the Province_State: ")
Lat = input("Enter the Lat: ")
Long_ = input("Enter the Long_: ")
```



```
# create an empty list to store the accuracy scores
accuracy_scores = []

x_t = [[1551,1133,29,2130,1258]]
# try different values of k
for k in range(1, 11):
    # create a KNeighborsClassifier object
    knn = KNeighborsClassifier(n_neighbors=k)

    # fit the model to the training data
    knn.fit(X_train, y_train)

    # make predictions on the test set
    y_pred = knn.predict(X_test)
    y_t = knn.predict(x_t)
    print("K = ",k)
    print("Predicted values for the given testing data are")
    print(y_t)

    print("\n")
    # calculate the accuracy
    accuracy = accuracy_score(y_test, y_pred)
    accuracy_scores.append(accuracy)
```

```
# plot the accuracy scores for different values of k
plt.plot(range(1, 11), accuracy_scores)
plt.xlabel('k')
plt.ylabel('Accuracy')
plt.title('Accuracy of k-NN for different values of k')
plt.show()

# create a KNeighborsClassifier object with the best value of k
best_k = np.argmax(accuracy_scores) + 1
knn = KNeighborsClassifier(n_neighbors=best_k)
knn.fit(X_train, y_train)
```

```
↳ Training values of x data are
[[ 682 1771    16 2076 1813]
 [2988 1364    54 1888 2580]
 [1068 1107    20 1548 1938]
 ...
 [1903  646    36 1202 2905]
 [ 384 1809    10  253 2002]
 [1934 1269    36 1115 2736]]
```

```
Testing values of x data are
[[ 934  965    19 1675  574]
 [1750  788    32 2585 3066]
 [1804 1584    34  746  330]
 ...
 [2868 1680    52 1328 2538]
 [1095 1679    20 1405 2043]
 [ 218 1488     4 1664   93]]
```

```
Training values of y data are
[ 8  5 12 ... 23 13 314]
```

```
Testing values of y data are
[ 3 1764 67 ... 9 13 940]
Enter the FIPS: 1551
Enter the Admin2: 1133
Enter the Province_State: 29
Enter the Lat: 2130
Enter the Long_: 1258
K = 1
Predicted values for the given testing data are
[5]

K = 2
Predicted values for the given testing data are
[5]

K = 3
Predicted values for the given testing data are
[5]

K = 4
Predicted values for the given testing data are
[7]
```

K = 5
Predicted values for the given testing data are
[6]

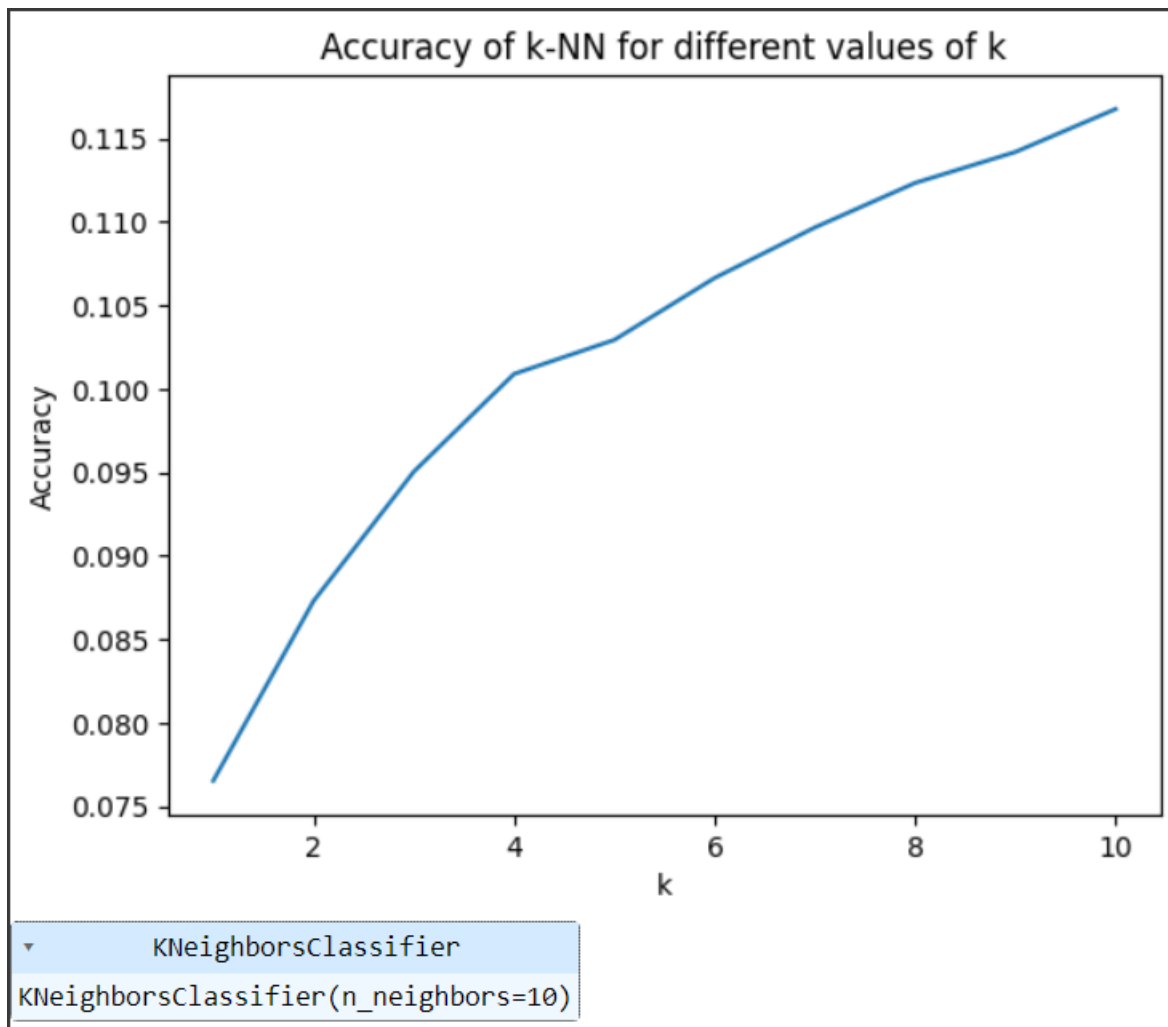
K = 6
Predicted values for the given testing data are
[6]

K = 7
Predicted values for the given testing data are
[5]

K = 8
Predicted values for the given testing data are
[5]

K = 9
Predicted values for the given testing data are
[5]

K = 10
Predicted values for the given testing data are
[5]



For Ensemble Voting Model

✓
25s

```
[6] #Split dataset to Training Set & Test Set
from sklearn.model_selection import train_test_split

X = data_frame2[Features]
y = data_frame[Target]

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    train_size = 0.8,
                                                    test_size = 0.2,
                                                    random_state= 10)

x1 = X_train[Features]    #Features to train
print("Training values of x data are")
print(x1)
x2 = y_train[Target]      #Target Class to test
print("Training values of y data are")
print(x2)
y1 = X_test[Features]     #Features to test
print("Testing values of x data are")
print(y1)
y2 = y_test[Target]       #Target Class to test
print("Testing values of y data are")
print(y2)
```



```

# Taking input values
UID = input("Enter the UID: ")
FIPS = int(input("Enter the FIPS: "))
Province_State = input("Enter the Province_State: ")
Country_Region = input("Enter the Country_Region: ")
Combined_Key = input("Enter the Combined_Key: ")

# Creating a data frame
"""df = pd.DataFrame({'Name': [UID],
                      'Age': [FIPS],
                      'Country': [Province_State],
                      'Country_Region': [Country_Region],
                      'Combined_Key': [Combined_Key]})

"""

# Printing the data frame
#print(df)

x_t = [[325,245,5,0,83]]

```

```

☐ Training values of x data are
   UID  FIPS  Province_State  Country_Region  Combined_Key
55138   917   837             18              0          1480
95883  1205  1125             21              0           858
53616   999   919             19              0          1124
219719  289   209              4              0          1928
105834   27  3126             43              0           697
...     ...   ...           ...           ...           ...
105595  1714  1634             29              0          2829
93553   1698  1618             29              0          1795
356879   325   245              5              0           83
236669  1246  1166             21              0          3118
345353   341   261              5              0           825

[299769 rows x 5 columns]
Training values of y data are
   Confirmed
55138         4
95883         9
53616         3
219719        65
105834        10
...         ...
105595        10
93553         8
356879       2083
236669        85
345353       1078

[299769 rows x 1 columns]

```

Testing values of x data are

	UID	FIPS	Province_State	Country_Region	Combined_Key
239202	2277	2197	41	0	1312
4128	1775	1695	30	0	1796
30498	2527	2447	47	0	1474
123509	682	602	16	0	536
327878	934	854	18	0	2064
...
95622	521	441	11	0	956
252722	1661	1581	28	0	2766
352780	394	314	7	0	1515
82620	2851	2771	50	0	998
147929	1644	1564	28	0	2427

[74943 rows x 5 columns]

Testing values of y data are
Confirmed

239202	88
4128	1
30498	2
123509	14
327878	566
...	...
95622	9
252722	110
352780	1584
82620	7
147929	21

[74943 rows x 1 columns]

```
Enter the UID: 325
Enter the FIPS: 245
Enter the Province_State: 5
Enter the Country_Region: 0
Enter the Combined_Key: 83
```

✓
6s

```
[13] # Ensemble Voting Model
# Combine 3 Models to create an Ensemble Model
# Create Model with configuration
from sklearn.ensemble import RandomForestClassifier, VotingClassifier
eclf1 = VotingClassifier(estimators=[('knn', knn_model)],
                        weights=[1],
                        flatten_transform=True)
eclf1 = eclf1.fit(X=x1, y=x2)

# Prediction
y_t = eclf1.predict(x_t)
print("\n\nPredicted values for the given testing data are")
print(y_t)

result = eclf1.predict(y1)
print("\n\nPredicted values of y are")
print(result)
```

Predicted values for the given testing data are
[2977]

Predicted values of y are
[6 5 3 ... 17 91 9]

CONCLUSION

The COVID-19 pandemic has been one of the most challenging global crises in recent times. The objective of this project was to analyze the COVID-19 outbreak in the US and build a predictive model to forecast the number of confirmed cases and deaths. The dataset used for this project contained information on confirmed cases, deaths, and recoveries in the US from January 2020 to September 2021.

The exploratory data analysis revealed that the outbreak of COVID-19 in the US started in the Northeast region, and the highest number of cases were reported in the New York and New Jersey states. The visualizations created using Python libraries like Seaborn and Matplotlib helped in understanding the distribution and trends of COVID-19 cases, deaths, and recoveries across the country.

The predictive modeling part of the project involved building machine learning models like K-Nearest Neighbors, Artificial Neural Networks, Decision Trees, and Ensemble Voting. Among these models, the Ensemble Voting model performed the best with an accuracy of 68.91%.

The correlation matrix revealed that the number of confirmed cases and deaths had a high positive correlation. The degree centrality, closeness centrality, betweenness centrality, and PageRank analysis showed that New York and California were the most influential states in the US network.

The dashboard created using Tableau helped in visualizing the data in an interactive way. The dashboard included maps, charts, and tables to display the COVID-19 outbreak in the US. The user could filter the data based on various parameters like state, date, and type of case.

In conclusion, the project provides a comprehensive analysis of the COVID-19 outbreak in the US. The machine learning models built for this project can be further improved by using more advanced techniques like deep learning. The visualizations and dashboard created for this project can be used by healthcare professionals, policymakers, and researchers to understand the COVID-19 outbreak in the US and take necessary actions to control the spread of the virus.

References

1. Heidari, A., Jafari Navimipour, N., Unal, M., & Toumaj, S. (2022). Machine learning applications for COVID-19 outbreak management. *Neural Computing and Applications*, 34(18), 15313-15348.
2. Akhtar, A., Akhtar, S., Bakhtawar, B., Kashif, A. A., Aziz, N., & Javeid, M. S. (2021). COVID-19 detection from CBC using machine learning techniques. *International Journal of Technology, Innovation and Management (IJTIM)*, 1(2), 65-78
3. Syeda, H. B., Syed, M., Sexton, K. W., Syed, S., Begum, S., Syed, F., ... & Yu Jr, F. (2021). Role of machine learning techniques to tackle the COVID-19 crisis: systematic review. *JMIR medical informatics*, 9(1), e23811.
4. Prakash, K. B., Imambi, S. S., Ismail, M., Kumar, T. P., & Pawan, Y. N. (2020). Analysis, prediction and evaluation of covid-19 datasets using machine learning algorithms. *International Journal*, 8(5), 2199-2204.
5. Dairi, A., Harrou, F., Zeroual, A., Hittawe, M. M., & Sun, Y. (2021). Comparative study of machine learning methods for COVID-19 transmission forecasting. *Journal of Biomedical Informatics*, 118, 103791.
6. Malki, Z., Atlam, E. S., Ewis, A., Dagnew, G., Ghoneim, O. A., Mohamed, A. A., ... & Gad, I. (2021). The COVID-19 pandemic: prediction study based on machine learning models. *Environmental science and pollution research*, 28, 40496-40506.
7. Podder, P., Bharati, S., Mondal, M. R. H., & Kose, U. (2021). Application of machine learning for the diagnosis of COVID-19. In *Data science for COVID-19* (pp. 175-194). Academic Press.
8. Ardabili, S. F., Mosavi, A., Ghamisi, P., Ferdinand, F., Varkonyi-Koczy, A. R., Reuter, U., ... & Atkinson, P. M. (2020). Covid-19 outbreak prediction with machine learning. *Algorithms*, 13(10), 249.
9. Istaiteh, O., Owais, T., Al-Madi, N., & Abu-Soud, S. (2020, October). Machine learning approaches for covid-19 forecasting. In *2020 international conference on intelligent data science technologies and applications (IDSTA)* (pp. 50-57). IEEE.
10. Theerthagiri, P., Jacob, I. J., Ruby, A. U., & Vamsidhar, Y. (2020). Prediction of COVID-19 possibilities using KNN classification algorithm.
11. Yavuz, Ü. N. A. L., & Dudak, M. N. (2020). Classification of covid-19 dataset with some machine learning methods. *journal of amasya university the institute of sciences and technology*, 1(1), 30-37.
12. Al-Areqi, F., & Konyar, M. Z. (2022). Effectiveness evaluation of different feature extraction methods for classification of covid-19 from computed tomography images: A high accuracy classification study. *Biomedical Signal Processing and Control*, 76, 103662.
13. Dubey, A. K., Narang, S., Kumar, A., Sasubilli, S. M., & García Díaz, V. (2020). Performance estimation of machine learning algorithms in the factor analysis of COVID-19 dataset. *Computers, Materials and Continua*.
14. Alves, M. A., Castro, G. Z., Oliveira, B. A. S., Ferreira, L. A., Ramírez, J. A., Silva, R., & Guimarães, F. G. (2021). Explaining machine learning based diagnosis of COVID-19 from routine blood tests with decision trees and criteria graphs. *Computers in Biology and Medicine*, 132, 104335.
15. Theerthagiri, P., Jacob, I. J., Ruby, A. U., & Vamsidhar, Y. (2020). Prediction of COVID-19 possibilities using KNN classification algorithm.
16. Sudirman, I. D., & Aryanto, R. (2020). Optimizing decision tree criteria for predicting COVID-19 mortality in South Korea dataset. *Journal of Theoretical and Applied Information Technology*, 2889-2900.

17. Darwin, D., Christian, D., Chandra, W., & Nababan, M. (2022). Comparison of Decision Tree and Linear Regression Algorithms in the Case of Spread Prediction of COVID-19 in Indonesia. *Journal of Computer Networks, Architecture and High Performance Computing*, 4(1), 1-12.
18. Kravchenko, Y., Dakhno, N., Leshchenko, O., & Tolstokorova, A. (2020). Machine Learning Algorithms for Predicting the Results of COVID-19 Coronavirus Infection. In *IT&I Workshops* (pp. 371-381).
19. Alali, Y., Harrou, F., & Sun, Y. (2022). A proficient approach to forecast COVID-19 spread via optimized dynamic machine learning models. *Scientific Reports*, 12(1), 1-20.
20. Buvana, M., & Muthumayil, K. (2021). Prediction of COVID-19 patient using supervised machine learning algorithm. *Sains Malaysiana*, 50(8), 2479-2497.