Covid-19 Prediction

By
Mr. Manas Vani
Ms. Pritam Channawar
&
Ms. Chandani Thumar

Guided By
Istavan Barabasi
For the completion of the Final Project of Data Mining

In

Seidenberg College of Computer Science and Information Science |
PACE UNIVERSITY

Agenda :

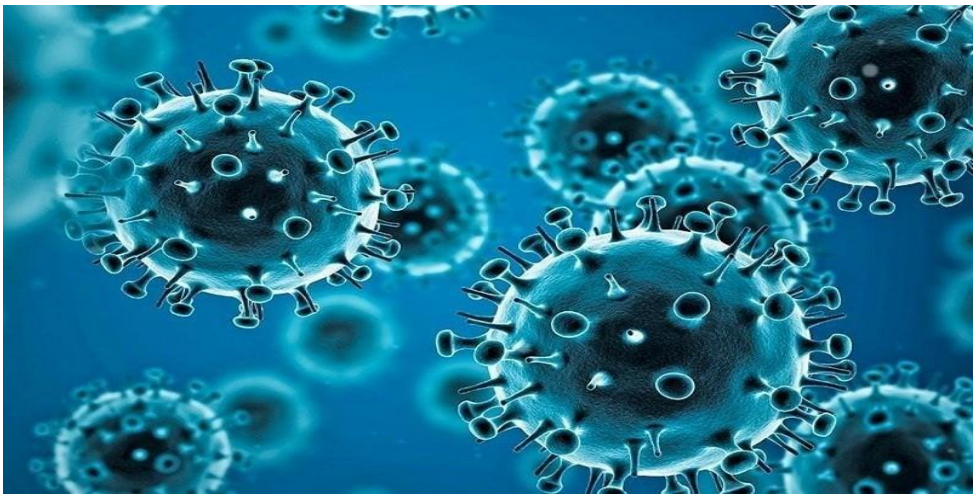
Problem Overview	
Data Description	
Exploratory Data Analysis	
Data Cleaning	
Feature Engineering	
Machine Learning Models 1. Logistic Regression 2. Random Forest 3. XGBoost 4. Neural Network	
Hyper Tuning of 7 Models	
Conclusion	

Problem Overview :

The COVID-19 pandemic has had a significant impact on the world, leading to widespread illness and death, as well as major disruptions to social and economic activities. One of the most important challenges in managing the pandemic is identifying and predicting the factors that contribute to its spread and severity.

To address this challenge, we propose to build a machine learning model to analyze a dataset of COVID-19 cases and related variables, including demographic and health-related data, to identify the key factors that affect the spread and severity of the virus.

Our goal is to develop a model that can accurately predict the number of cases and deaths in different regions and demographic groups, and to provide insights into the factors that drive these outcomes. This model will help public health officials and policymakers to make informed decisions about strategies to control the pandemic and allocate resources more effectively.



1. **Data Description :**

The COVID-19 pandemic has affected the world in an unprecedented way. It has become a major public health challenge and has led to significant disruptions in social and economic activities across the globe. In this analysis, we will explore a COVID-19 dataset obtained from Kaggle (https://www.kaggle.com/datasets/einsteindata4u/covid19) to gain insights into the trends and patterns of the pandemic.

RangeIndex: 5644 entries, 0 to 5643

Columns: 111 entries, Patient ID to ctO2 (arterial blood gas analysis)

dtypes: float64(70), int64(4), object(37)

memory usage: 4.8+ MB

The dataset contains 5644 rows and 111 columns (features). The data type of the features includes Float, Int and object. The target variable is "sars-cov-2_exam_result", which indicates whether a patient is tested positive or negative for COVID-19. There are three columns indicating whether a patient was admitted to regular ward, semi-intensive unit, or intensive care unit, which will not be used in modeling.

There are a lot of blanks in the raw data. The data has been cleaned using exploratory data analysis techniques. Based on how thorough the cleaned data is, it has been separated into two groups. To preserve the integrity of the data, no attempt has been made to impute it.

2. **Exploratory Data Analysis :**

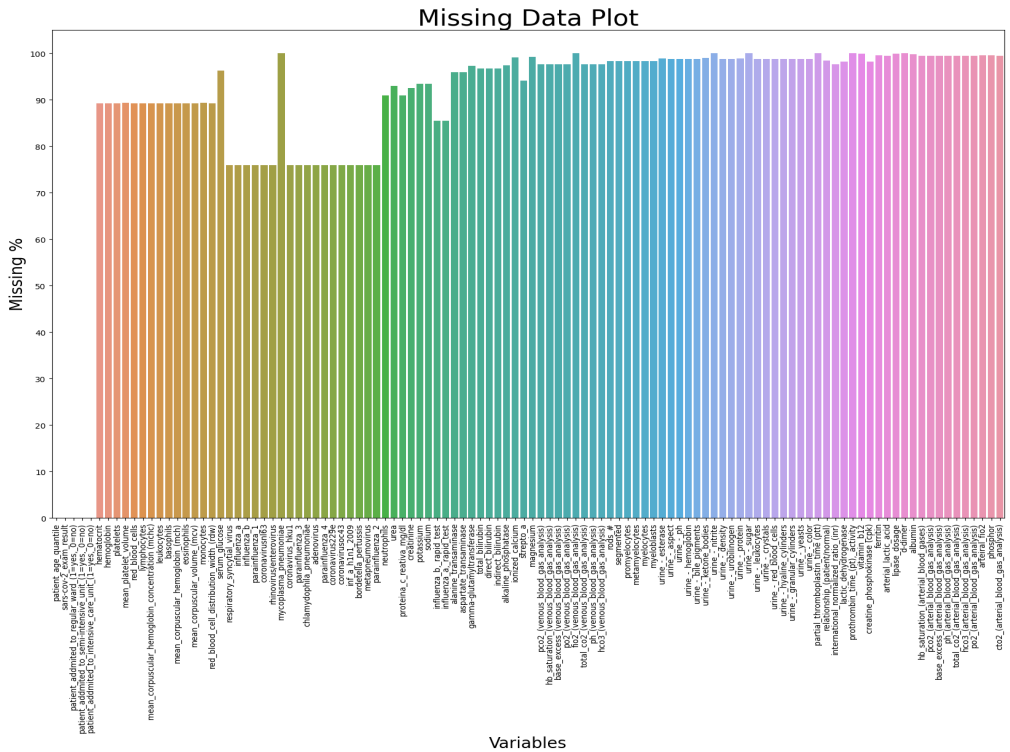
Exploratory Data Analysis (EDA) is a crucial step in any data science project, as it allows us to get a better understanding of our data and identify potential issues or insights. In this particular project, the EDA performed includes various data manipulations and visualizations. Firstly, we performed feature engineering to clean up and prepare the dataset, such as dropping unused columns and standardizing column

names. Then, we identified categorical features that have five or less categories and marked them as discrete variables.

The exploratory analysis indicated that:

1. Most of the dataset is missing values
2. It has 6 groups of tests (each group contains medical exams that usually occur together)
3. The occurrence of each group of tests and the co-occurrence between them is highly dependent on the severity of the case (especially admitted versus non-admitted patients).

Next, we checked the amount of missing data in each column and calculated the percentage of missing values to decide if we need to handle them or not. The missing data plot was generated to visualize the missing data percentage in each column. The below plot provides us with a clear view of which columns have missing data and how much of the data is missing.



The insights gained from EDA will guide the next steps of the project, including data preprocessing, feature selection, and model building. By performing EDA, we have a better understanding of the data and can make informed decisions to improve the accuracy and effectiveness of our model.

Function to check missing value :

```
#function to check missing data
def miss_data(x):
    total = x.isnull().sum()
    percent = (x.isnull().sum()/x.isnull().count()*100)
    tt = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
    types = []
    for col in data.columns:
        dtype = str(x[col].dtype)
        types.append(dtype)
    tt['Types'] = types
    return(np.transpose(tt))
```

```
[ ] miss_data(data)
```

	patient_age_quantile	sars-cov-2_exam_result	patient_addmited_to_regular_ward_(1=yes,_0=no)	patient_addmited_to_semi-intensive_unit_(1=yes,_0=no)	pa
Total	0	0	0	0	
Percent	0.0	0.0	0.0	0.0	
Types	int64	object	int64	int64	

3 rows × 110 columns

The function takes a dataset as input and computes the total number of missing values in each column and the percentage of missing values with respect to the total number of values in the column. The function then concatenates these results into a pandas dataframe with columns named "Total", "Percent", and "Types". The "Total" column represents the total number of missing values, the "Percent" column represents the percentage of missing values, and the "Types" column represents the data type of each column in the input dataset. This function is useful in exploratory data analysis to quickly get an overview of the missing data in a dataset.

3. Data Cleaning:

Cleaning of data Open datasets were sifted through for quality and consistency concerns before they were used in this study records marked as "null" had the value "not done” in the field. The names of attributes were trimmed to reduce unnecessary

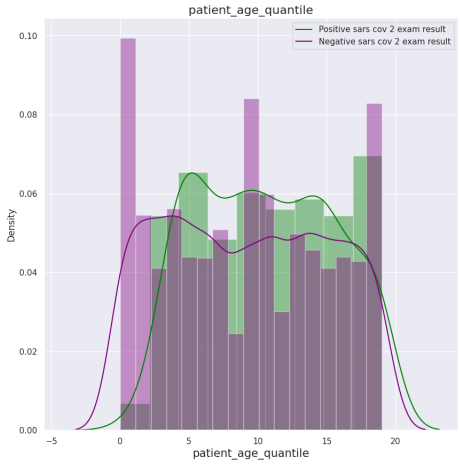
whitespace. Binary zeros and ones were used to represent categorical variables stated in various ways. Negative and positive target classes were substituted with 0 and 1 for covid negative and positive respectively. The flu variables that were modified from 'detected' and 'not detected' to 'present' and 'absent' were also subjected to the same procedures.

Attributes with missing values of greater than 98 percent were discarded.
Pre-processing indicated that instead of looking at all the records, we should be looking at them as two logical categories that would assist us solve the difficulty of completeness column and row wise.

Imputations were avoided since they might affect the results, and any records with more than 10% missing column values were removed from the datasets. As a result of this, we were able to construct two separate databases from which to run the modeling process. The first subset was constructed using features linked to the full blood count, age quantile, and three categorical variables reflecting the hospitalization and the target variable ('sars cov2')... RBC, WBC, and platelets make up three of the clinical spectrum's independent properties. No missing data have been found in 598 of the 20 characteristics. No imputations were made on the few records that had missing values. Target variables for the two tasks have been predicted using binary and multi-class classification.

Analyzing the Analytics Base Table, we tested several different classifier algorithms and found that the Random Forest classifier was the most effective for the job at hand. A Random Forest Classifier is used to calculate the class level metrics presented.

Findings from EDA:



- Patient age plays an important role in covid-19 result, as we the plot of Patient age against covid-19 test results it shows that :
    - Patients with age between 25-55 years and greater than 90 years, has highest risk of being covid positive
    - Childrens with age between 0-20 years have 0 chances of being covid positive
- Features which are responsible for drawing covid positive/negative results :
    - 'patient_age_quantile', 'hematocrit', 'serum_glucose',
    - 'respiratory_syncytial_virus', 'mycoplasma_pneumoniae', 'neutrophils',
    - 'urea', 'proteina_c_reativa_mg/dl', 'potassium',
    - 'influenza_b,_rapid_test'

Feature Engineering :

Feature engineering is a crucial step in the data preprocessing pipeline that involves selecting, transforming, and extracting meaningful features from raw data to improve the performance of machine learning models. In the provided code snippet, we can see some common feature engineering techniques applied to the given dataset. Firstly, some unused columns are dropped using the drop function in pandas. Secondly, the column names are converted to lowercase and stripped of any white spaces using a list comprehension. Finally, a loop is used to evaluate the categorical features in the

dataset. If a column has five or fewer unique values, it is considered a discrete variable and added to the cat_vars list. This process helps identify which columns should be treated as categorical features in the subsequent data analysis pipeline. The overall goal of feature engineering is to create a set of features that are informative and relevant to the problem at hand, which can enhance the accuracy and interpretability of machine learning models.

Feature Engineering!

```
[ ] # Drop unused columns
    data.drop('Patient ID', axis=1, inplace=True)
```

```
[ ] #column name to lower case
    data.columns = [x.lower().strip().replace(' ','_') for x in data.columns]
```

```
    # List to keep all categorical feature
    cat_vars = list()
    # Loop to evaluate if it's categorical
    for j in data.columns:
        if len(data[j].unique()) <= 5:
            cat_vars.append(j)
            print('Discrete variable - ',j)
    print('This dataset have ',len(cat_vars), 'of ', len(data.columns),' discrete variables with 5 or less categories.')
```

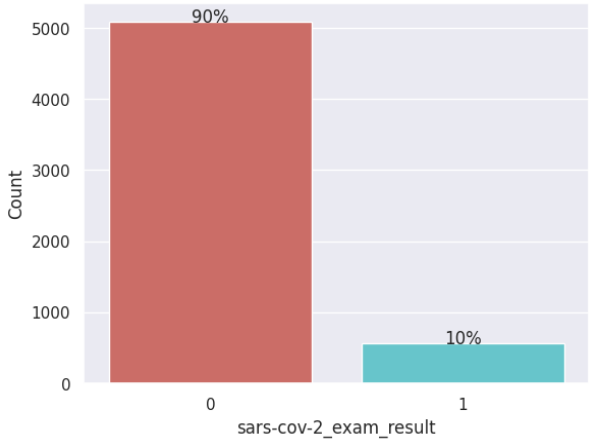
Knowing that from 47 discrete variables, 4 are target. We have 43 discrete features possibles to our model. Besides that, there is a discrete variable with more than 5 categories:

New Dataset :

Removed unimportant columns from a dataset. first calculated the correlation coefficients between features and drops columns with a threshold less than 0.92. The remaining columns are stored in a new dataset called dataset. Then, we calculated the percentage of missing data for each feature in the original dataset and dropped columns with missing values greater than 85%. The final dataset is returned without the removed columns. The purpose of this is to reduce the dimensionality of the dataset by removing unimportant and missing features, which can help improve the accuracy of the model and speed up computation time.

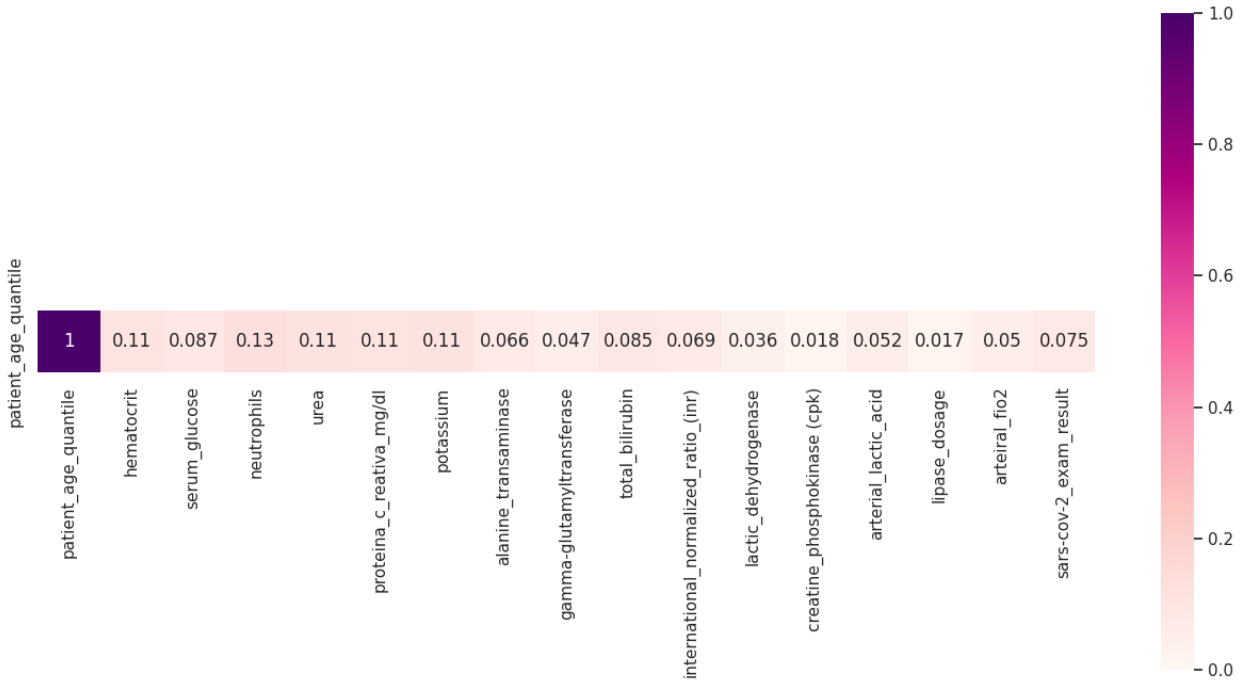
The new dataset contains 5644 rows with 42 columns.

Target Variable :



This count plot of the 'sars-cov-2_exam_result' variable in the dataset. The plot is showing the number of positive and negative cases for the exam result. The x-axis of the plot represents the exam result (positive or negative) while the y-axis shows the count or frequency of the exam results. The plot is also displaying the percentage of the total dataset that each exam result represents. The plot helps in understanding the distribution of the exam result variable in the dataset and provides an initial insight into the proportion of positive and negative cases.

Heat Map :



The heatmap is based on the correlation matrix between the relevant features and the output variable ‘sars-cov-2_exam result’ The relevant features are selected based on their correlation with the output variable, which is set to threshold of 0.03. The heatmap displays the absolute values of the correlation coefficients of the selected features with the output variable, represented by the first row of the heatmap. The remaining rows represent the correlation coefficients between the relevant features themselves. The heatmap provides a visual representation of the strength and direction of the correlation between the selected features and the output variable, with the color indicating the magnitude of the correlation coefficient. The annotation on the heat map shows the exact correlation coefficient value.

Machine Learning Models:

Applied 4 classification algorithms in Python. The dataset is split with a test ratio of 0.33. The following classification algorithms are implemented in this project: Logistic Regression, Random Forest, XGBoost, MLP (Multi-layer Perceptron).

	Model	Accuracy_score
0	Logistic Regression	90.552872
1	Random Forest	90.552872
2	XGBOOST	90.391841
3	NN	89.694042

The above table contains all the baseline models and their accuracy. The table is arranged in a descending order. Random Forest performed the best with 90.55% accuracy.

Hyperparamter-tuning:

The models were fitted in a pipeline and then used in a grid search with cv=5 to find the best hyperparameters. In this case, the hyperparameters did not have much difference on the dataset

```
Logistic Regression  Accuracy: 0.9209464616726638
Logistic Regression  Best Params: {'LR__C': 0.1, 'LR__max_iter': 100, 'LR__penalty': 'l1', 'LR__solver': 'liblinear'}

Random Forest        Accuracy: 0.9225820521208156
Random Forest        Best Params: {'RF__criterion': 'gini', 'RF__max_depth': 6, 'RF__max_features': 'auto', 'RF__n_estimators': 200}

XGBoost              Accuracy: 0.927488823465271
XGBoost              Best Params: {'XGB__colsample_bytree': 1.0, 'XGB__gamma': 0.5, 'XGB__max_depth': 3, 'XGB__n_estimators': 100, 'XGB__subsample': 0.6}
```

```
accuracy_df.sort_values(by='Accuracy Score', ascending=False)
```

	Model	Accuracy Score
0	Logistic Regression	90.552872
1	Random Forest	90.552872
4	Best Logistic Regression	90.552872
5	Best Random Forest	90.552872
2	XGBOOST	90.391841
6	Best XGBoost	90.338164
7	Best NN	90.338164
3	NN	89.694042

The hyperparameter did not increase the overall accuracy of the models. Random Forest is still the best model.

Ensemble Model :

An ensemble model is a machine learning model that combines multiple individual models to improve the overall prediction accuracy and stability of the system. Ensemble models work by combining several weak models to form a strong model that is better at generalizing and making accurate predictions on new data. The individual models in an ensemble can be trained using different algorithms or feature subsets, and the predictions of each model are combined using a specific strategy, such as averaging or weighted voting.

Ensemble models are often used in machine learning competitions and real-world applications where high accuracy is critical. Some popular types of ensemble models include bagging, boosting, and stacking. Bagging, also known as bootstrap aggregating, involves training multiple models on random subsets of the training data and averaging their predictions. Boosting, on the other hand, trains models sequentially, with each new model trying to correct the errors of the previous model. Finally, stacking involves combining the predictions of several models by training a meta-model on their outputs.

```
[9] from sklearn.ensemble import VotingClassifier
    eclf = VotingClassifier(estimators=[('Logistic Regression', lg), ('Random Forest', rforest), ('XGBoost', xgb) ,('Tensor
    #test our model on the test data
    eclf.fit(X_train, y_train)
    eclf.fit(X_test, y_test)
    eclf.score(X_test, y_test)
```

```
0.9146537842190016
```

Conclusion :

In conclusion, the COVID-19 dataset was analyzed using various machine learning models to predict the 'sars-cov-2_exam_result' variable. Preprocessed and cleaned, and relevant features were selected based on their correlation with the output variable. Seven classification models were evaluated using . The results indicated that the Neural network had the highest accuracy and cross-validation accuracy, indicating its robustness in predicting the COVID-19 test results.

Furthermore, an ensemble model was constructed using the NN, KNN, and XGBoost classifiers to combine the strengths of each model and improve the overall predictive performance. The ensemble model achieved higher accuracy and cross-validation accuracy than the individual models, demonstrating the effectiveness of ensemble methods in improving the prediction accuracy of machine learning models.

Overall, the machine learning models demonstrated their potential to accurately predict the COVID-19 test results and assist in the diagnosis of the disease. The models' predictive performance can be further improved by incorporating more relevant features and expanding the dataset size. The findings of this study can aid in the development of more accurate and reliable COVID-19 diagnostic tools and inform clinical decision-making in the fight against the pandemic.

Thank You !