Data Understanding and Processing

- **Source:** The dataset COVID19.csv contains patient medical records from Hocus Pocus town.
- **Target Variable:** The Result column consit of (POSITIVE, NEGATIVE, PENDING, INDETERMINATE).
- **Features:** A mix of demographic data (Sex, Birth Year) and a wide array of clinical symptoms (Cough, Fever, Difficulty breathing, etc.) many symptoms are recorded as either 'YES' or 'NO' with more of missing or empty values.
- **Size:** The dataset accurately has 109927 rows and 47 columns before data cleaning is done on the data.
- **Data Types:** Primarily categorical (strings: 'YES', 'NO') and one integer column (Birth Year). The target variable is multi-class but will be simplified to a binary classification problem (POSITIVE vs. NEGATIVE) based on the use case.

Preprocessing Steps:

1. **Simplify Target Variable:** the Combination of 'NEGATIVE', 'PENDING', and 'INDETERMINATE' into a single class (e.g., 0 for "NEGATIVE") will be discarded because the proportion of NEGATIVE class compare to POSITIVE class is higher which thus has created an imbalanced class between the two classes ("NEGATIVE = 0, POSITIVE = 1"). Doing this turns it into a binary classification problem, which is more straightforward and aligns with the goal of predicting positive cases.
2. **Handle Missing Values:**
   a. For the EDA part it shows that the missing values don't just occurs randomly rather it has a pattern. That means the missingness itself carries predictive power: people with missing symptom entries are mostly PENDING or NEGATIVE. Dropping the missing values will lead to lost of important signals in the EDA section

   b. For the model building part, i need to drop some features that the missing values are more than 70% , and also impute the categorical variable using the most frequent strategy and for the numerical variables using the median as the strategy.
3. **Encode Categorical Features:** Convert all 'YES'/'NO' features into binary integers (1/0). Convert the Sex column into a binary integer (1 for MALE, 0 for FEMALE).

4. **Create New Feature:** use symptoms to engineer Gastrointestinal (GI) symptom and Neurological symptom columns

Data Cleaning

1. **Drop Irrelevant Features:** Columns like Specify other complications are mostly empty and will be dropped. Other clinical symptoms and Other complications are too vague for reliable feature engineering and will also be dropped.
2. **Drop Unknown rows:** rows having unknown are dropped in order to increased the predictive power of the model and furthermore, building a model for this particular case study cannot be trained with an uncertainty since it's specify.
3. **Drop Redundant Features:** The **"**Sore throat or pharyngitis" and "Thorax (sore throat)" features are the same feature so it serve as noise while training the model.

Feature Engineering

1. **Symptom per patient:** the total count of symptoms per each patient is carried out by summing up the number of symptoms marked "YES".
2. **Gastrointestinal group:** categorize the symptoms of gastrointestinal by combining ("Diarrhea", "Vomiting", and "Nausea") symptoms as one. And created these columns: "GI_Symptom_Count" a number of GI symptoms a patient reports (0–3). "GI_Symptom_Flag" it is just a quick yes/no if any GI symptom is present.
3. **Neurological group:** categorize the symptoms of neurological disorders by combining ("Headache", "Loss of Smell", "Loss of Taste") symptoms as one. And created these columns: "`Neuro_Symptom_Count`" a number of neurological symptoms a patient reports (0–3) are "YES". "`Neuro_Symptom_Flag`" it is just a quick yes/no if any neurological symptom is present.

Model Development

1. **Define:** Define the predictors (X) and target variable (y).
2. **Splitting:** split the data into X_train, y_train, X_test, y_test.
3. **Data Processing:** cleaning and transformation of the data. With the train data separated from the test data in order to avoid data leakage.
4. **Class Imbalanced:** Handle class imbalance with SMOTE

Choosing Algorithm

Logistic Regression: A good baseline model, interpretable and has class weight properties to balance classes.

Random Forest: Powerful, robust to outliers, it also has class weight properties to balance classes and handles interactions well.

XGBoost (Gradient Boosting): Often provides state-of-the-art results on structured data, handles imbalanced data well.

Model Interpretation and Deployment

1. **Model Interpretation:** Logistic regression model outperformed all other model so in this case it is consider as the best model because it has:
   - Highest Recall (7.69%) - Most important for catching positive cases
   - Highest F1 Score (11.90%) - Best balance between precision and recall
   - Highest AUC-ROC (52.79%) - Best overall ranking performance
   - Good Precision (26.25%) - Reasonable precision given the recall

Recommendation: all of the models could not properly predict the minority class due to the degree of the data quality, so it will be better to collect more data for the minority class.

2. **Deployment:** The Deployment is done using the Flask API, HTML, CSS and hosting it on the PythonAnywhere server.

Ethical Considerations

1. **Medical Disclaimer:** The model is a **decision support tool**, not a diagnostic tool. It must never replace professional medical advice, diagnosis, or treatment. The output should be clearly labeled as a prediction, not a certainty.
2. **Bias and Fairness:** The model is trained on data from "Hocus Pocus." It may not generalize well to other populations with different demographics, genetics, or testing protocols. It could perpetuate biases present in the training data (e.g., if certain groups were under-tested).
3. **Data Privacy:** Patient data used for training must be anonymized and handled in compliance with regulations like HIPAA or GDPR. The API should not store the input data from users.
4. **Transparency:** The report and feature importance plot provide some transparency into the model's logic ("interpretability"), though complex models like XGBoost are often seen as "black boxes."
5. **Consequences of Error:**
    a. **False Negative:** Telling an infected person they are not positive could lead to them not isolating, potentially spreading the virus.
    b. **False Positive:** Causing unnecessary stress, quarantine, and further medical tests for a healthy person.
    The cost of a False Negative is likely higher in a pandemic context. The model's decision threshold can be tuned to prioritize **Recall** (minimizing False Negatives) over Precision, depending on the use case.