Predicting Covid-19 Diagnosis from Clinical Data

Runsheng Wang, Jinqi Lu[1]

April. 29, 2021

Abstract

This study seeks to use machine learning to predict Covid-19 diagnosis from a multitude of clinical data. The data was collected, aggregated, de-identified, and published by Carbon Health, a U.S. primary and urgent care provider. The dataset contains a total of 46 variables, including epidemiological factors, comorbidities, vitals, clinical assessed and patient reported symptoms, as well as lab results and radiological findings. The dataset was first cleaned and encoded, before splitted into a training set and a testing set. Exploratory Data Analysis was performed on the training set to obtain basic understandings about the characteristics of input variables. Feature selection was then performed by ensembling the Chi-Squared and Mutual Information criteria. Six features, namely Cough, Fever, Headache, Loss of Smell, Loss of Taste, and Muscle Sore, were ultimately used for model construction. Due to the extreme imbalance (99:1 ratio) in the class distribution of the response variable, resampling methods were applied to the training set in order to reduce classification bias towards the majority class. Since the dataset consists solely of categorical features, SMOTE-N, a categorical variation of the Synthetic Minority Oversampling Technique (SMOTE), was adapted to the training set to achieve a negative-to-positive class ratio of 5:1. Undersampling was then performed using One Sided Selection and Neighborhood Cleaning Rule to further balance the class distributions of the response variable. To identify the best model for this particular dataset, six classifiers, namely kNN, Logisti, Decision Tree, Random Forest, Categorical Naive Bayes, XGBoost, were fitted and baseline performances were compared. To optimize the hyperparameter tuning process, Random Search CV was executed 2000 times on the Random Forest Classifier, with 4-fold CV repeated three times on each iteration. Best hyperparameters were recorded locally, and Grid Search CV was applied to further tune the hyperparameters by checking the immediate neighbors of each hyperparameter value. The model predicts the testing set quite well, and model performance is substantially better compared to the baseline. However, the model underperforms at predicting positive results, which is to be expected considering the severe class imbalance present in the original dataset. Our model could be applied to prioritize testing when testing resources are scarce or inaccessible. It could also aid the diagnosis of Covid-19 in ambiguous or conflicting cases.

1 Introduction

As Covid-19 continues to ravage countries across the globe, testing has become an indispensable tool for identifying and containing the spread of this highly contagious virus. Test results can lead to early prevention of further spread as well as early intervention and treatment for patients experiencing non-severe symptoms, resulting in less mortality and faster recovery. However, test results are often inaccurate, especially with molecular tests whose "false-negative rate was 20% when testing was performed five days after symptoms began, and much higher (up to 100%) earlier in infection" (Shmerling 2021). Given that the infection of Covid-19 comes with clinical symptoms, we hypothesized that certain symptoms are more indicative of Covid-19 test results than others (CDC 2021). We will attempt to identify features of symptoms that exert significant influences on test results, and construct a machine learning model that predicts Covid-19 diagnosis based on clinical data. Our model could be used to prioritize Covid-19 testing when testing resources/kits are limited, or when testing is inaccessible. We do not anticipate the model to replace testing, but it could be used in conjunction with test results when results are ambiguous, or when results disagree. Furthermore, by highlighting important symptoms, we hope to enable individuals to self-screen and determine whether they should consider getting tested or not.

[1] R.W., J.L. contributed equally to this work.
R.W., J.L. performed wrangling, modelling, prediction, and wrote the paper.

2 Materials and Methods

2.1 Data Collection

As a result of the Health Insurance Portability and Accountability Act of 1996 (HIPAA) Privacy Rule, we faced significant difficulties in obtaining our desired datasets. To protect patients from identity theft and fraud, HIPAA was established to hold the healthcare industry accountable for how they manage and protect the sensitive private information of their patients (HIPAA Journal 2017). Our desired dataset would contain not only patient reported symptoms on Covid-19, but also patient comorbidities and vitals, among other factors. These features are classified as highly sensitive private medical data, and are therefore rarely available on public domains, whether they have been de-identified or not. After some initial research, we found a large comprehensive patient clinical dataset actively maintained by the National Covid Cohort Collaborative (N3C, n.d.). We applied for access to this database, and our request was approved. However, accessing any actual datasets requires submitting another detailed project proposal for review. Since waiting for approval would further delay our progress, we opted to search for de-identified datasets elsewhere.

Our extensive search led us to two datasets with our desired information. The first dataset was sourced from the nationwide clinical data publicly reported by the Israeli Ministry of Health (Nshomron 2021). However, the original dataset has been removed from the official website, and the accessible data is a preprocessed version containing only eight input features. The second dataset was sourced from Carbon Health, a U.S. primary and urgent care provider (Carbon Health 2020). This dataset contains de-identified clinical data on 93995 patients who received Covid-19 testing at Carbon Health, aggregated over the course of six months. This dataset includes 46 features sorted into nine major categories (See Appendix A for Data Availability and Data Dictionary). Since the difference in feature space is large, merging the two datasets, or utilizing one as test data for the other, would result in significant loss of useful information. Therefore, only the Carbon Health dataset is considered for the purpose of this study. The data collection process highlights not only the difficulties in obtaining restricted data, but also the importance of maintaining data privacy and protecting anonymity. As described by Narayanan and Shmatikov, it is possible to de-anonymize personal data even when proper de-identification protocols are applied (Narayanan and Shmatikov 2008). In our future studies, we will make sure that we apply to restricted databases as early as possible, and that we strictly follow the data usage guidelines when working with sensitive dataset.

2.2 Data Concatenation and NA Processing

The raw dataset was given as 29 separate CSV files, each representing data for a particular week beginning from Apr. 12, 2020 and ending at Oct. 20, 2020. The files were first concatenated into a larger CSV and saved locally.

There are 46 features in the dataset, but several features contain large amounts of NA values. This is to be expected, partially as a direct result of the de-identification process which sets data items to null if the data item could potentially be used for re-identification. Features with over 40% of its values as NAs are excluded from model building, as it is simply impossible to impute such large amounts of missing values, and imputing would necessarily result in bias. Several imputation methods were considered for features with less NAs. However, none were eventually adopted, due to the categorical nature of this dataset. Categorical variables are often imputed with the most frequent class (mode-imputing) or through a classifier (e.g., kNN). Since all categorical features are binary in this dataset, mode-imputing would lead to significant bias towards the most frequent class. kNN imputation is also unsuitable; not only is the distance metric ambiguous for categorical features, but also due to the high dimensionality of the feature space, resulting in Curse of Dimensionality for all distance-based metrics.

Classifiers such as XGBoost have built-in methods to deal with NA values, where a default direction is assigned to each decision node and missing values are directed to the default direction. However, assigning the default direction to each node correctly is an extremely complex problem, and improved results are often not guaranteed. To avoid introducing any bias, we decided to discard rows with any NA values after we excluded features with over 40% NAs.

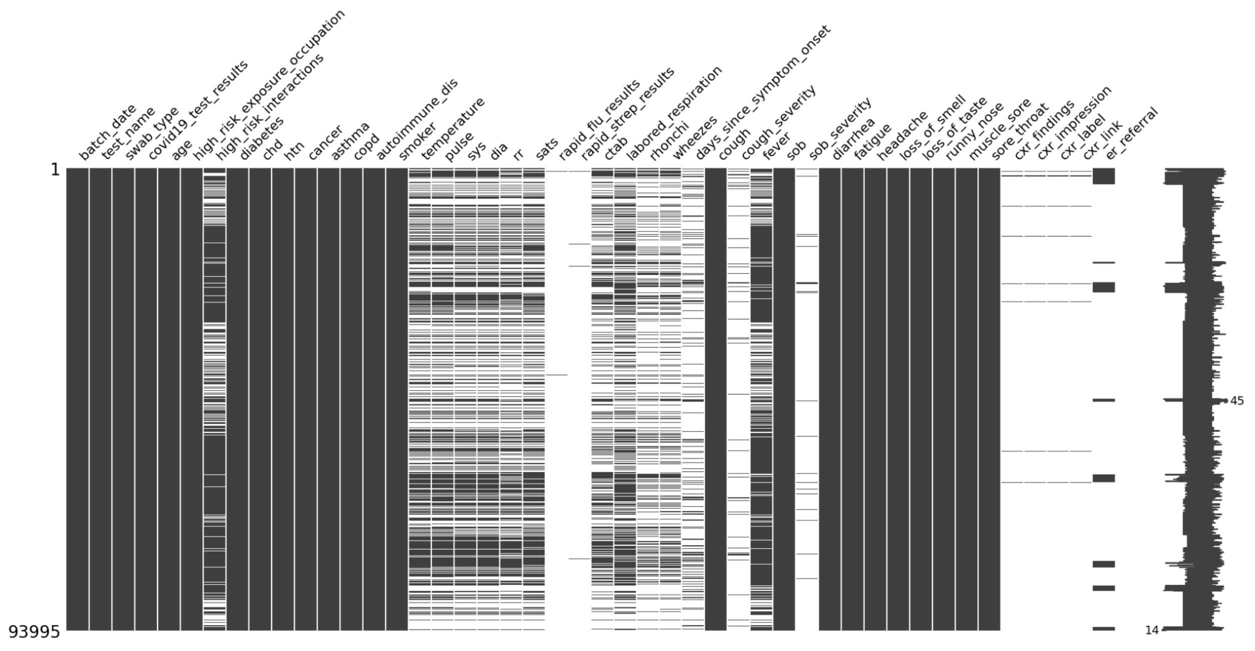


Figure 1. NA Distribution

In Figure 1, the distributions of NA values in each feature were plotted using the Missingno package. Each line white line represents a single missing entry in a particular column, and each black line represents a non-NA entry. After excluding features with over 40% NAs and other irrelevant features (e.g., batch_date, test_name, etc.), 22 features remain in the dataset. They are:

- Covid19_test_results, age, high_risk_exposure_occupation, diabetes, chd, htn, cancer, asthma,
- copd, autoimmune_dis, smoker, cough, fever, sob, diarrhea, fatigue, headache, loss_of_smell,
- loss_of_taste, runny_nose, muscle_sore, sore_throat

The original dataset has 93995 observations. After dropping all NAs row-wise, 71035 samples remain. The dataset is still considerably large, providing sufficient samples for machine learning algorithms to draw conclusions.

We considered splitting the data into training and testing prior to handling NAs, as unseen data could technically include NA values. We decided to split after dealing with NAs, primarily because we anticipate to apply a classifier that only works on complete datasets. Moreover, very few classifiers handle NA values as input without using some form of imputation. We have justified that imputation is not appropriate for this particular dataset. Therefore, we will be using a classifier without NA capabilities.

2.3 Encoding

The categorical features in the raw dataset are encoded as binary string variables or boolean True/False. Most machine learning algorithms only accept numeric types as input and output; model runtime and model performance can also vary based on how the input is encoded. Thus, encoding categoricals to numerics are necessary prior to any modeling.

Since the raw dataset contains missing values, the "read_csv" function from pandas uses type "object" to represent both booleans and string binaries in order to account for the existence of NAs. To ensure proper encoding, categorical features are first type-casted to strings and lower-cased. The Label Encoder and One Hot Encoder from sklearn.preprocessing were considered, but neither were eventually used due to their slow runtime. Encoding after cleaning NA values also helps to avoid errors, as most prebuilt encoders do not handle missing values. We designed a custom encoder using pandas' vectorized string replace function. Since all categoricals have binary classes, One-Hot-Encoding is not necessary. The problem of one class being further away or closer to another class in a multi-class (3 or more) classification is not present in this scenario. We proceeded to encode "False" as 0 and "True" as 1, while encoding "Negative" as 0 and "Positive" as 1.

The raw dataset contains one continuous variable "Age". Several classifiers (e.g., Naive Bayes, Random Forest) do not directly operate on continuous values. For instance, Naive Bayes turns continuous features into quartiles and encodes it from 1 to 4. Random Forest, on the other hand, determines a threshold for splitting the continuous feature on each node. Both methods are similar, if not equivalent, to binning continuous values into discrete categories. Numerous prior Covid-19 research have indicated that seniors are at greater risk of contracting Covid-19 and experiencing severe illness (CDC 2021). Using this domain knowledge and the knowledge about classifier designs, we decided to engineer "Age" into a new categorical feature with entry = 1 if the patient is above 60 years old and 0 otherwise.

The order in which encoding and train-test-split are performed is not of major concern because the test set would have to be encoded as well upon acquiring. In scenarios where both the validation and testing set are to be accessed later in the study, one can always apply the same preprocessing pipeline to the unseen sets. The sklearn.preprocessing package offers encoders that "memorize" the encoding schema, which can be fitted later. The vectorized replace can perform the same thing, with the added benefit of handling missing values when necessary.

2.4 Train-Test Split

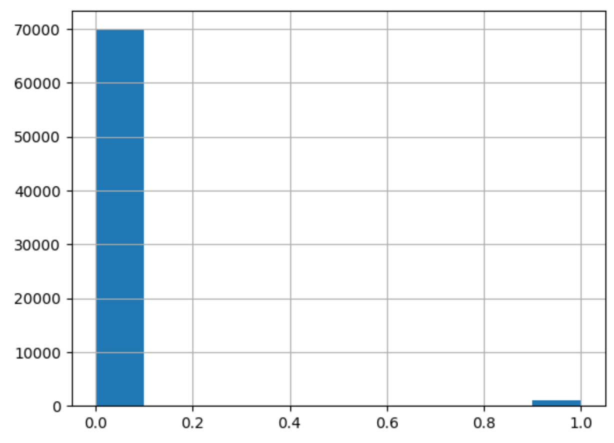


Figure 2. Class Distribution of Covid-19 Test Result in Processed Dataset

To enable testing on unseen data and prevent the model from overfitting, train-test split was applied to the processed dataset. Random sampling (shuffle-then-select) and stratified sampling were both considered. Stratified sampling was ultimately used due to the extremely severe class imbalance (99:1) of the target variable, as shown in Figure 2. Since the number of positive samples are limited, it is very likely for random sampling to distribute the already limited samples unevenly among the training and testing set. If the majority of the positive samples are assigned to the training set, a classifier would overfit the training data and there would not be enough positive samples in the testing set to validate performance. If the majority of the positive samples are assigned to the testing set, a classifier might fail to extrapolate patterns and relationships that separates a positive sample from a negative sample. Furthermore, classification algorithms are too easily biased by the over or under representation of classes; stratified sampling most closely resembles the class distribution of the raw data, allowing further resampling methods to be applied prior to model-building. Since the dataset is large, the train-test split ratio is less influential on model results. We used a 80:20 ratio, commonly used in data science research and partially justified by the Pareto Principle. We did not use a three-way split as we don't want to impact training performance by further reducing the scarce positive samples in the training set. Instead, we will be using cross validation to perform hyperparameter tuning.

2.5 Feature Selection

With 21 input features, the feature space of the processed dataset exhibits high dimensionality. Not only is high-dimensional feature space more prone to overfitting, one extra dimension requires an exponential increase in the volume of data to produce statistically meaningful results. All L2-norm based methods suffer as a direct result of the Curse of Dimensionality, in that object appears to be sparse and dissimilar in many ways in higher dimensional space, or that tiny pairwise differences in higher dimensions result in huge differences in similarity/dissimilarity. Including more features does not necessarily improve model performance, and doing so might introduce noise and hamper the predictive power of both distance and non-distance based models. Given that the health conditions of individuals vary greatly and certain symptoms could be more indicative of Covid-19 results than others, it might be useful to select features that offer greater predictive power than others. Although some classifiers have built-in methods that control feature selection, such as the "max_features" parameter that exists in multiple sklearn classifiers, eliminating less relevant features prior to modeling would allow the algorithm to fine tune the preselected set of features and further improve performance while avoiding overfitting.

In order to make a statistically justified feature selection and obtain the best model possible with optimized performance and lowest possible computational cost, an ensemble of two feature selection methods were applied to the processed dataset. We originally considered PCA as the dimensionality reduction method, but whether it is appropriate to apply PCA to binary data remains a statistically controversial topic with limited research available. One argument against PCA is that the variance minimization object breaks down with binary variables, and the concept of computing the mean of a categorical variable might not be easily interpretable. Some also argue that PCA performed on a binary dataset is the same as performing a Multiple Correspondence Analysis, but justifications for this statement remain unclear.

We did not bother with PCA as there are other methods suited for categorical features, namely the Chi-Squared statistics and the Mutual Information criterion.

$$\chi^2 = \sum_{i=1}^{R} \sum_{j=1}^{C} \frac{(o_{ij} - e_{ij})^2}{e_{ij}}$$

Figure 3. Chi-Squared Statistics

The Chi-Squared statistics can be used to test whether two categorical variables are correlated or not based on the contingency table. The equation in Figure 3 computes the Chi-Squared statistics. The term o_{ij} correspond to the observed cell count in the i^{th} row and j^{th} column of the contingency table, and the term e_{ij} correspond to the expected cell count in the i^{th} row and j^{th} column of the contingency table, which can be computed by

$$e_{ij} = \frac{\text{row } i \text{ total} * \text{col } j \text{ total}}{\text{grand total}}$$

Figure 4. Chi-Squared Statistics

The capital R and C in the Chi-Squared statistics correspond to the number of rows and number of columns of the contingency table, respectively. Each individual term in the double summation can be interpreted as: how much different is the observed frequency of a specific pair of factor levels different from the expected frequency of this pair, if there is no relationship at all between this pair. The double summation adds up this difference for each possible pair of factor levels between the two categorical variables of interest. If the observed frequency of a pair is exactly the same as its expected frequency, it would match the null hypothesis that the pair is uncorrelated. Mathematically, observed frequency being close to the expected frequency results in the $o_{ij} - e_{ij}$ term approaching 0, hence contributing a small amount to the overall summation. If all pairs have their observed frequency close to their expected frequency, then the total value of the summation would be small. Therefore, smaller Chi-Squared statistics is an indication of weak correlation between two variables, and feature selection can be performed by ranking the larger Chi-Squared values.

We first constructed a function that uses sklearn's SelectKBest to perform Chi-Squared Test of Independence on all input features in the training set with respect to Covid-19 test results. The function returns an array of scores, or values of the test statistics corresponding to each feature. This output array is used as the input for another function, which produced the plot below showing the Chi-Squared statistics of each feature

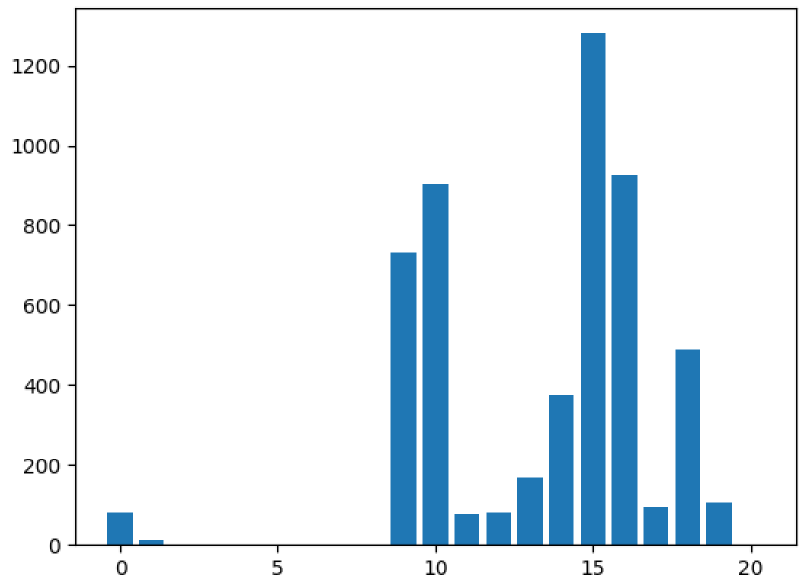


Figure 5. Chi-Squared Statistics for Input Features

As shown in Figure 5, features 9, 10, 14, 15, 16, and 18 have significantly higher Chi-Squared statistics compared to other features, and the difference reaches up to ten orders of magnitude. These features are

- Feature 9: Cough
- Feature 10: Fever
- Feature 14: Headache
- Feature 15: Loss of Smell
- Feature 16: Loss of Taste
- Feature 18: Muscle Sore

It is worth noting that SelectKBest does not use p-values when selecting features, according to its source code. However, high t-values intrinsically mean low p-values and greater confidence in rejecting the null hypothesis. Therefore, we used SelectKBest by specifying k to be all the features, and ranking the Chi-Squared statistics afterwards.

To further validate this finding, we decided to use a different selection criterion in conjunction with the Chi-Squared statistics.

In information theory, Mutual Information (MI) is a commonly used measure for feature independence. Specifically, it measures how much knowing one variable reduces the uncertainty about the other. In the case of two jointly discrete random variables, it can be computed as follows

$$\mathrm{I}(X;Y) = \sum_{y\in\mathcal{Y}}\sum_{x\in\mathcal{X}} p_{(X,Y)}(x,y)\log\left(\frac{p_{(X,Y)}(x,y)}{p_X(x)\,p_Y(y)}\right)$$

Figure 6. Mutual Information Criterion for Jointly Discrete R.V. X and Y

The term $p_{(X,Y)}$is the joint PMF of $X$and Y; $p_X$and $p_Y$are the marginal PMF of $X$and Y, respectively. From the Figure 6, it can be seen that MI is a measure of the inherent dependence expressed in the joint distribution of $X$and$Y$relative to the marginal distribution of X and Yunder the assumption that they are independent. When $X$and$Y$are independent, the term $p_{(X,Y)}(X,\ Y) = p_X \cdot p_Y$, which results in

$$\log\left(\frac{p_{(X,Y)}(x,y)}{p_X(x)\,p_Y(y)}\right) = \log 1 = 0$$

Figure 7. Mutual Information Criterion when X and Y are Independent

On the other end of the spectrum, if X determines the value of Y, or that by knowing Xwe automatically know Y, then all information of $Y$is contained in X, and the MI value would be high. Therefore, smaller MI values is an indication of weak correlation between two variables, and feature selection can be performed by ranking the larger MI values, similar to how we used Chi-Squared.

According to sklearn's documentation, the sklearn implementation of SelectKBest using MI relies on nonparametric methods based on entropy estimation from k-nearest neighbors distances. Therefore, it would produce slightly different results each time, and the ranking of the weakly correlated features can vary a lot at each iteration. Unfortunately, SelectBestK does not allow the use of random seed for MI feature selection. We have devised an alternative method to not only improve feature selection confidence but also allow for reproducibility of results.

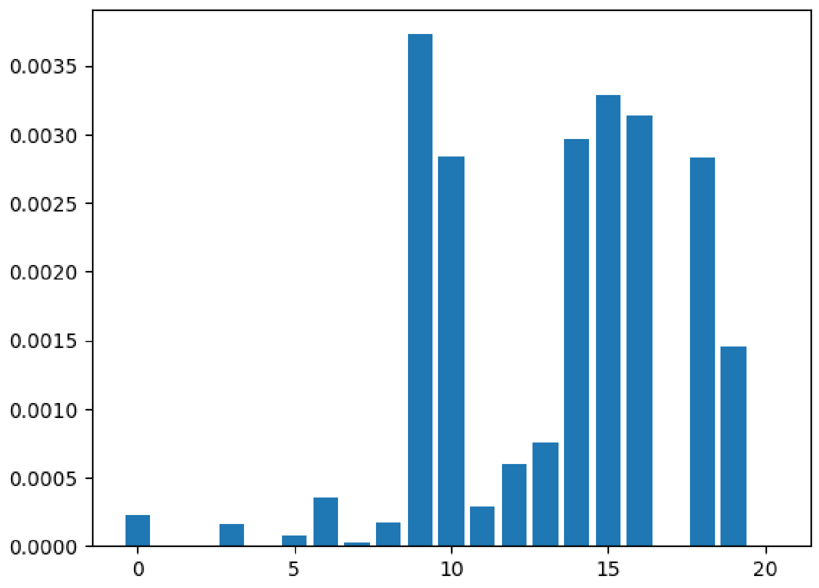


Figure 8. Mutual Information Criterion for Input Features

Figure 8 illustrates one random iteration of the MI feature selection process. Similar to how the Chi-Squared output was constructed, we first computed MI for each feature, and plotted Figure 8 via a separate function. The resultant MI values are in close agreement with the output from Chi-Squared. However, the MI method appears to be more sensitive to weakly correlated features. To combine information from both criteria, we developed an ensemble method that takes into account both the input from Chi-Squared and MI.

We first created a union of all features with a Chi-Squared statistics greater than 1 and a MI value greater than 0.0001 (call this set S). Due to the stochastic nature of the implementation of the MI method, we performed MI feature selection 10 additional times. On each iteration, we ranked all features with MI greater than 0.0001 (17 of them) by their MI value and selected only the top n features to be intersected with set S. The intersection set then replaces the old S set before the start of the next iteration. The value n is also decremented by 1 on each iteration, starting at 17 and ending at 8. This repeated feature selection method removes uncertainty associated with entropy estimation in the implementation of SelectKBest.

This method also confirms that features 9, 10, 14, 15, 16, and 18 are indeed the significant features that should be considered in modeling. There could be correlations between these selected features, as certain symptoms are often concurring. However, correlations between input features could actually be indications of particular patterns. For instance, a patient experiencing cough, fever, headache, loss of smell, and muscle sore all at once could be a strong indication for Covid-19 infection. On the other hand, if a patient only experiences headache without cough and fever, he/she might be less likely to be diagnosed as positive due to critical symptoms not present/concurring. For this reason, we are more concerned about obtained correlated features, rather than obtaining independent features.

To further confirm the validity of these findings, we cross referenced the common Covid-19 symptoms published on CDC's website as well as other research findings (CDC 2021). It appears that the features selected are top symptoms in multiple studies, which justifies the correctness of feature selection from the domain knowledge perspective. Since our repeated feature selection takes a while to run, we hardcoded the significant features to save time and avoid repeating previous work. It is also worth noting that feature selection helps with distance-based undersampling method, as dimensionality reduction makes these algorithms less prone to the Curse of Dimensionality.

2.6 Resampling

Before proceeding with model construction, a critical problem needs to be addressed. As shown in Figure 2, the target variable exhibits a severe class imbalance (99:1). Any attempt to directly classify this dataset would result in the minority class being swamped by the overwhelming number of samples in the majority class. Training a model on such an imbalanced dataset might yield a great accuracy score, but high accuracy is a direct result of the severe imbalance and fails to capture misclassifications in the minority class. For our specific dataset with a class ratio of 99:1 for the target variable, an algorithm that simply classifies all results as negative would have an accuracy of 99%. In other words, accuracy is a very inappropriate metric for this specific dataset. Moreover, the class imbalance in this particular set is so severe that any recognizable patterns in the positive samples might not be properly captured by the classifier. Therefore, a close inspection and adjustments are needed in order to achieve our desired outcome.

The best approach to solve this problem is to collect more data, but the significant difficulties in data collection have been discussed earlier in the report. Another great alternative is to apply resampling methods. Originally, a combination of random oversampling and random undersampling were considered, which consists of randomly duplicating samples from the minority class and randomly deleting samples from the majority class. However, random resampling introduces several serious problems. Specifically, random oversampling can result in model overfit, as duplicate samples increase the classifier's bias towards the input features that make up these samples. Random undersampling runs the risk of losing invaluable information that contributes to classification, especially on boundary conditions. We decided to apply more rigorous methods in order to improve resampling outcomes.

For oversampling, we decided to use SMOTE-N, a categorical variation of the Synthetic Minority Oversampling Technique (SMOTE). Traditional SMOTE is a synthetic data generation technique which works by connecting minority samples that are close in the feature space via a hyperplane and drawing a new sample at a random location on the hyperplane. However, SMOTE can only be applied to continuous data. Even though the one-hot-encoding of our dataset allows the notion of "distance" to be calculated to some extent, randomly picking a location in the hyperplane would result in non-discrete data entries, which violates the binary property of every input feature. This is where SMOTE-N shines.

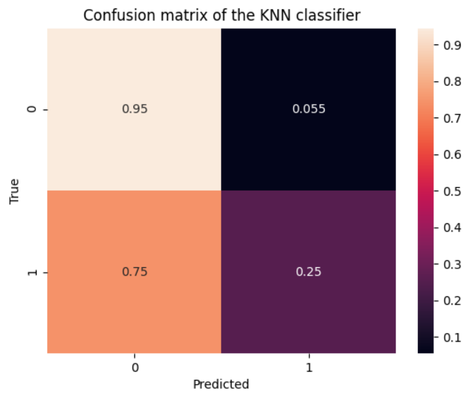
Using SMOTE-N, we achieved a negative to positive ratio of 5:1. We also applied undersampling to further reduce majority samples and clean decision boundaries. Specifically, we applied One-Sided-Selection, which uses a combination of Tomek Links and Condensed Nearest Neighbor rule. Neighborhood Cleaning Rule was then applied, which combines Condensed Nearest Neighbors with Edited Nearest Neighbors. All together, resampling resulted in a negative to positive class distribution slightly lower than 5:1

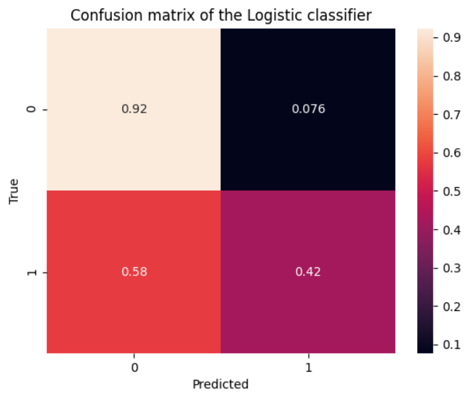
2.7 Model Selection

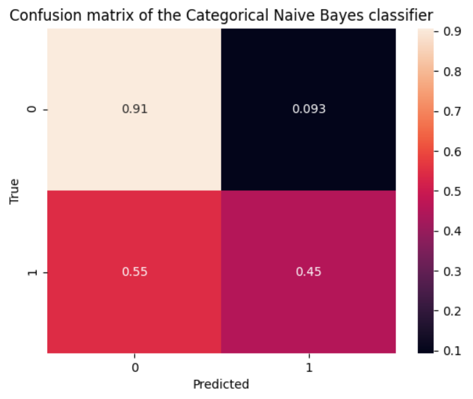
Base models using six different classifiers were spot-checked for their performance. These models are

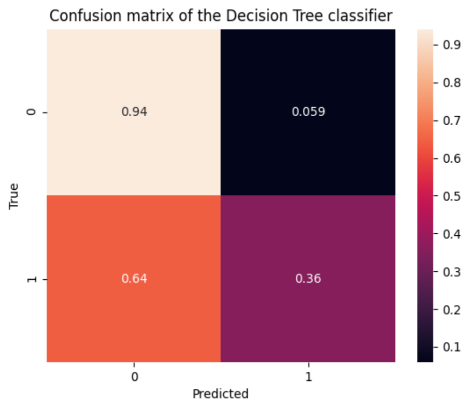
- kNN
- Logistics
- Categorical Naive Bayes
- Decision Tree
- Random Forest
- XGBoost

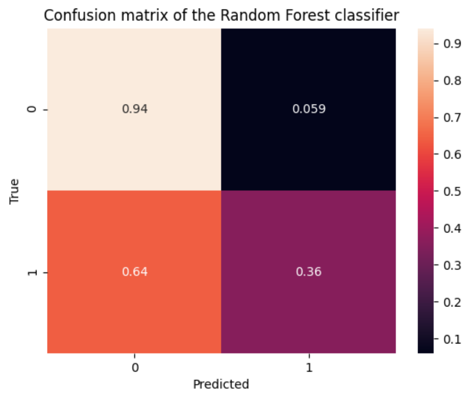
As there exists severe class imbalance, many evaluation metrics are not appropriate for this dataset. Confusion matrices are a straightforward way to visualize performance. The confusion matrix for all the base classifiers are shown below

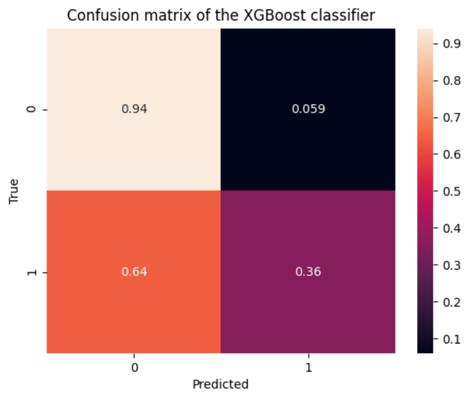












4 Appendix

A Data Availability

The Data can be found under our data folder of our project folder in the class repository, named "Predicting_COVID-19_Test_Result/data". The data folder contains all the original raw datasets.

B Code Availability and Reproduction Process

The code can also be found under our main project folder, tilted “process-predict.py”. The code can be directly run to produce all the data tables and outputs, which will be placed into the folder named “output”