Healthcare Final Project: A Case Study Using Predictive Analytics for Risk Profiling of COVID-19 Patients

Healthcare Data Analytics and Data Mining

Group 1

Heyuan Gao
Zijing Hu
Jinman Rong
Ziyu Tang (Audit)
Pengcheng Xu
Haina Yan

Summary

In this report, we did a predictive analysis to identify the at-risk population and come up with economy re-opening solutions using the clinical dataset which includes about 400,000 rows, in which 250 variables including patient demographics, past diagnostic histories and mortality status are reported.

Main topics are discussed as follows:

- Background of problem
- Research objective
- Descriptive analysis
- Modeling Process and Result Interpretation
- Model comparison
- Insights and suggestions

Through this report, you will get to know the age, gender and disease distribution regarding death and non-death population. Five types of machine learning methods are provided to reinforce each other, delivering high predictive performance. In the end, we checked the literature review to cross-validate our results.

Introduction

Background of Problem

Covid-19 has hit the U.S. particularly hard. Since the first case was reported in late January, more than 1.3 million American have been infected and a total of more than 70 thousand deaths have been recorded.[1] Moreover, recent autopsies reports in California pushed first Covid-19 death back to early February, though it was initially confirmed on February 26th in Washington State.[2] As a result, Governor Gavin Newsom ordered autopsies back to December to find out how long the disease has ravaged the golden state. In the meanwhile, the mayor of Belleville, New Jersey, Michael Melham, claimed that he had coronavirus in November 2019, at which time he underwent a high fever, chills, hallucinations and a sore throat that ended up lasting for three weeks. Back then he was diagnosed with a bad flu, however, recently he went to test his blood for Covid-19 antibodies and got a positive.[3] Putting these information together, it is safe to assume that Covid-19 has existed for a much longer time in the U.S. than people originally thought, and the situation may be worse than current models suggest.

Adding to the terrible situation is an incompetent administration and a polarized political climate in Washington, which only before-Civil-War Parisian divergence came close. The Trump administration has delivered controversial if not stupid talks on several occasions. The president downplayed the pandemic in the beginning and the administration was significantly lagging behind in terms of preparing sufficient medical supplies for an imminent pandemic. [4] Moreover, making falseful and dangerous claims such as injecting disinfectant into bodies[5] or using of antimalarial drugs which never clinically proved to be effective[6] to treat Covid-19 confused the public and exacerbated the situation.

---

[1] https://coronavirus.1point3acres.com/

[2] https://www.washingtonpost.com/nation/2020/04/22/death-coronavirus-first-california/

[3] https://news.cgtn.com/news/2020-05-05/U-S-Belleville-mayor-claims-that-he-had-coronavirus-in-November-2019-Qfq40LrHlC/index.html

[4] https://theintercept.com/2020/03/25/coronavirus-flu-comparison-trump/

[5] https://www.nytimes.com/2020/04/24/us/politics/trump-inject-disinfectant-bleach-coronavirus.html

[6] https://www.wsj.com/articles/antimalaria-drug-doesnt-help-treat-covid-19-large-but-inconclusive-study-finds-11588890515

In order to curb the effect brought by the super contagious virus, social distancing has been imposed in most states in the U.S., and non-essential economic activities have been called to halt. Despite all efforts devoted to slowing the spread of Covid-19, the number of new cases in the U.S. is increasing by over 20,000 constantly, and so far, there is no sign of it going down soon.

On the other hand, owing to the halt of economic activities, all indices tracing American economy (except the equity market indices) have slipped to all time low since the Great Depression in the 1930s[7]. The labor market is hit hard as well. Another 3.2 million people filed for initial jobless claims last week, bringing the total number of claims over 33 million. Initial claims are considered a proxy for layoffs or furloughs, and that level represents about 21% of the March labor force, which has been the worst since the 1930s.[8] Projections into future weeks even suggest that the unemployment rate will surpass that in the trough of the Great Depression.

Now individuals and the government are facing a critical trade-off: should they continue to shut down the economy, more businesses will be forced to go bankrupt and more people will lose jobs[9]; on the other hand, if the re-openings progress at too fast a pace, more people are at risk of exposure to the virus, and an explosively increasing infection cases will drain the limited medical resources. As a result, the re-opening effort will be supported, only when there are enough test kits to limit the scope of infection.

However, the number of tests still falls short of what is needed. Despite President Trump boasted about testing capabilities in the U.S. on several occasions, data shows that testing per capita in the U.S. ranks roughly 40th in the world and trails countries such as South Korea, which uses massive testing to keep down the number of infections. The testing capabilities in the U.S. simply cannot support a full re-opening.[10]

[7] https://www.nytimes.com/2020/04/29/business/economy/us-gdp.html

[8] https://www.cnn.com/2020/05/07/economy/unemployment-benefits-coronavirus/index.html

[9] https://www.marketwatch.com/story/neiman-marcus-and-jcrew-could-survive-bankruptcy-filings-experts-say-2020-05-08

[10] https://www.theatlantic.com/science/archive/2020/05/theres-only-one-way-out-of-this-mess/611431/

Research Objective

The dire economic situation in the U.S., on the other hand, does not allow people to wait until the testing capability reaches its full potential. Thus, state by state, partial re-openings are being allowed to progress. Partial re-openings without massive testing will require people who are at high risk of succumbing to the virus to stay at home, and the limited test kits should be allocated to meet their needs first. The rest of the population who exhibit little or no symptoms or recover from the infection will be spared from testing in order not to cause a shortage in test kits. As a result, accurately predicting and identifying those who are at high risk becomes an essential task in order to make the partial re-openings a smooth progress and to avoid future shutdowns again.

We are targeting such an objective in this report. Acquiring a dataset consisting of about 400,000 rows Covid-19 patients' data, in which patient demographics, past diagnostic histories and mortality status are reported, we are set to predict whether a patient will succumb to the disease using demographic and diagnostic information. We will be training several machine learning models such as bagged logistic regressions, random forest and neural networks to achieve this target.

Descriptive analysis

Data Description and Preparation

In order to get from the COVID19 source file to analytic file, we mapped the age and sex variable to six age groups (from 18 to over 75) and dummy gender variable respectively. Note that we treat unknown gender as female (GenderMale=0) and the unknown age as 0 which belongs to none of the age groups. Also, we converted the 6 age groups to dummy variables. In terms of ICD10 DX codes, we mapped them to their upper categories (DGL_3_Extend) and made them dummies as well. For each patient, if he/she was diagnosed by at least one ICD10 DX code of a specific DGL_3_Extend code, we will treat the code dummy variable as 1. For example, if one patient was diagnosed by ICD10 code of F4320 which belongs to the dummy variable BEHAV_AdjustmentStress in our analytics data, then this variable would be 1.

As a result, we got an analytics dataset full of dummy variables sized in 370633 records and 229 features.

Gender Bias

Exploring our dataset, we generated a cross table of gender and COVID19 mortality (Table 1 Cross Table of Gender and Mortality). Accordingly, we applied the Fisher Exact Test to this table and gained odds ratio of 1.62 and p-value near 0. Therefore, we conclude that the gender bias on COVID19 mortality is statistically significant, to be specific, males might be more sensitive to COVID19 so that they have a higher death rate than females.

Table 1 Cross Table of Gender and Mortality

	Female (or unknown)	Male
Non-Death	193034	156329
Death	9197	12073

Age and Mortality

For each mortality group (0 and 1), We plotted a bar chart on the number counts of different age groups. From the plots below, it is obvious that younger people are more tolerant on COVID19, in other words, most deaths happened on elder group, especially age over 75. Among all patient records older than 75, the death rate incredibly reached 19.5%, compared with overall death rate of 5.7%.

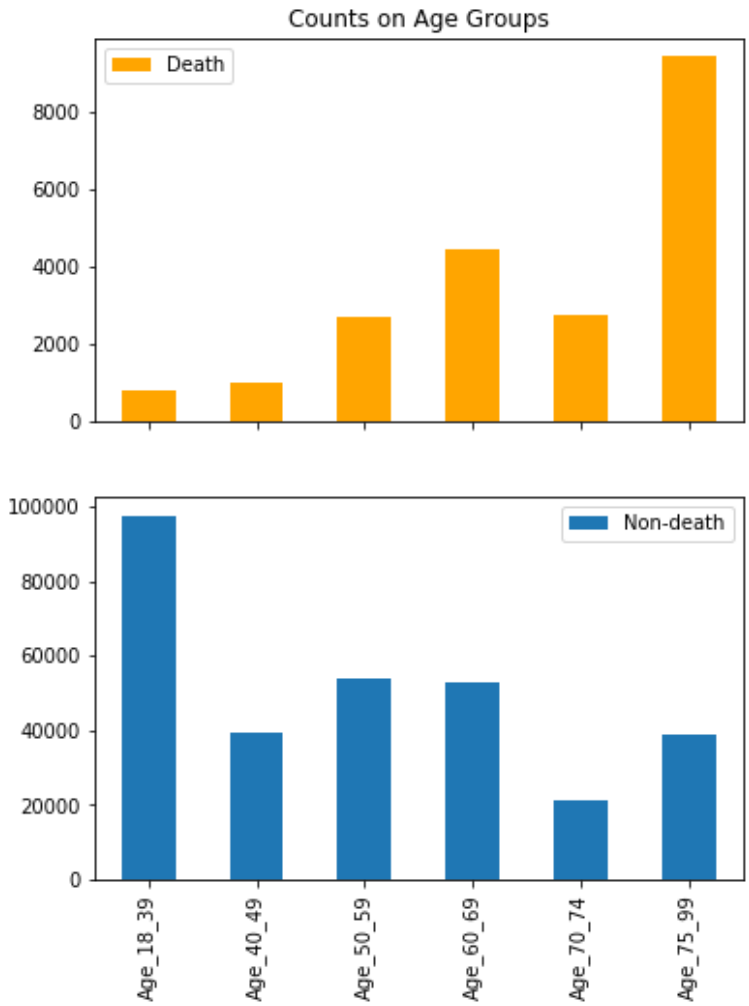

Figure 1 Age Distribution regarding Death and Non-death

DX Group and Mortality

Additionally, we checked the DX Groups and plotted the counts on death and non-death patients. The top DX groups on death patients are CVAS (cerebrovascular accidents), General Unspecified group, and Substance Abuse (e.g. alcohol abuse), etc. For non-death patients, top DX groups are General Unspecified, GSTIN_Other (gastrointestinal related), and Substance Abuse, etc. It is hard to directly discriminate between death and non-death as there are some overlaps of DX groups. However, we could still see groups like ENDOC_MET (diabetes and metabolic disease) occur in top 10 DXs of death and groups like ENTD_Other only show in top10 Dx groups of non-death patients.

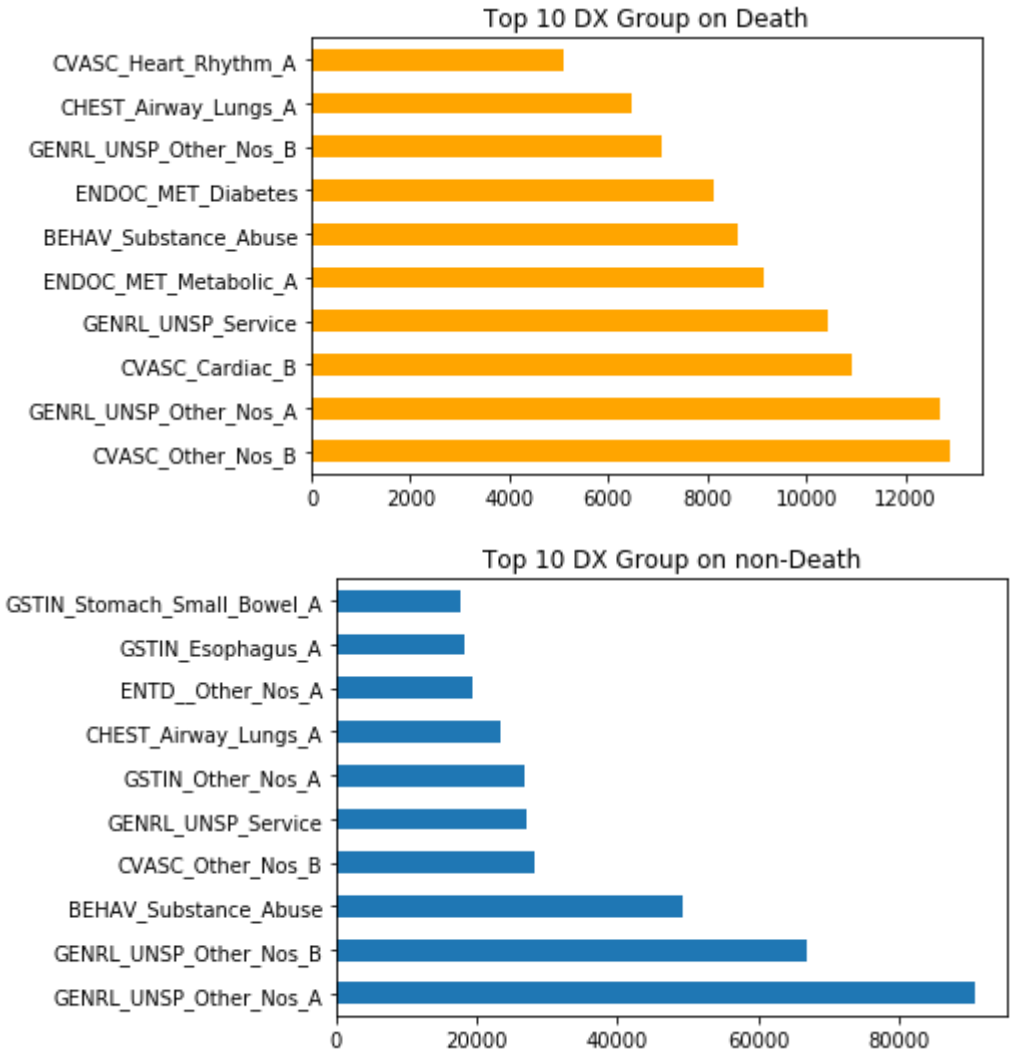

Figure 2 DX Group Distribution on Death and Non-death

Modeling Process and Result Interpretation

Data Partition

First of all, we split the original data into two parts: train data and test data. Train data was for training the model and test data was to evaluate the model to see its prediction performance about unseen data. We used a threshold of 75/25 for splitting train and test data. Additionally, we set a fixed random seed for reproducibility.

Data Balance

We found that we came across imbalanced class distribution. The number of observations belonging to death was significantly lower than those belonging to non-death – we saw that the death rate of the original data set was about 5.7%. In this situation, the predictive model developed using conventional machine learning algorithms could be biased and inaccurate.

Imbalanced data also leads to bias in evaluating model performance. For example, a dummy classifier can easily reach an accuracy of 94% only by guessing the majority label under any circumstances. This requires us to carefully choose metrics that reveal the predicting ability of our model. So, apart from accuracy, we use precision, recall rate and f1 score to provide more perspectives of the model performance.

There mainly to be ways to deal with imbalanced data, under-sampling and oversampling. Under-sampling balances the dataset by reducing the size of the abundant class. Oversampling is used when the quantity of data that we care about (e.g. death cases) is insufficient. It tries to balance the dataset by increasing the size of rare samples. Considering the need to reduce information loss, we decided to use a bootstrapping approach to rebalance the train data when running *Logistics Regression, ID3 Tree, Boosted Tree and Neural network*. For Naïve Bayes, it performs well in regard to imbalanced data.

Logistic regression

Model Introduction

Logistic regression model is a statistical method that statisticians and data scientists use to classify people, products, entities, etc. It is mostly used to analyze data that produces a binary classification based on one or many independent variables. This means, it produces two clear classifications (Yes or No, 1 or 0, etc).

Logistic regression model is widely used in the healthcare field. For example, healthcare providers use logistic regression to accurately target at risk individuals who should receive a more tailored and detailed behavioral health plan to help improve their daily health habits. This in turn opens the opportunity for better health for patients and lower costs for hospitals. For insurance companies, logistic regression helps them detect where there is potential fraud of insurance plans by measuring certain characteristics of their insurance payers. This will help companies reduce huge financial risks.

Since logistic regression is often used in binary classification, in this case, it is quite appropriate to predict mortality after learning features such as age, gender and different types of diseases.

Model Interpretation

After running the logistic regression model 200 times, we mainly get predictor coefficients, confusion matrix and feature importance to evaluate our model's predictive performance.

(1) Top 10 Predictor Coefficients

There are 226 predictors and we ranked them according to their corresponding coefficients. Below is table of the Top 10 predictors, which all have positive signs of their coefficients. Therefore, for people with those ten diseases are more likely to die due to covid-19, holding other features constant.

We can see that 5 of them are related to cardiac, 2 of them are related to chest. The result is aligned with the news that high-risk people have complications such as Heart disease, Lung disease and Respiratory disease. To be specific, the top 5 most impactive predictor are CVASC_Cardiac_B (related to heart disease), DERMA_Whole_Body_Nos_B (related to body burn), ENDOC_MET_Diabetes (related to diabetes), CVASC_Heart_Rhythm_A (related to heart disease) and CHEST_Status_A (related to chest disease).

One thing needs to mention is that 14 of them have the coefficients of zero and we discover that these ICD-10 codes represent diseases that are not correlated with covid-19, such as adjusting artificial arm, Corrosion of mouth and pharynx and Injury of optic chiasm.

Table 2 Top 10 Predictors and Corresponding Coefficients of Logistics regression

Predictor	Coefficient
CVASC_Cardiac_B	3.34
DERMA_Whole_Body_Nos_B	2.77
ENDOC_MET_Diabetes	2.16
CVASC_Heart_Rhythm_A	2.04
CHEST_Status_A	2.01
CHEST_Airway_Lungs_B	1.95
CVASC_Venous_B	1.93
CVASC_Heart_Rhythm_B	1.89
FGENT_Screening	1.79
CVASC_Other_Nos_B	1.72

(2) Confusion Matrix

As we can see from the confusion matrix, the overall accuracy is 0.94, which represents all the right predictions (real no death, predictive no death & real death, predictive death) divided by all the predictions. The high accuracy means our model made few wrong predictions.

To be specific, the recall value for label 0 is 0.94, which means among total actual no death (82436+4954), 94% can be detected as no death (82436). The precision value for label 0 is 1, which means among total predicted no death people (82436+352), nearly 100% could be detected as no death.

The recall value for label 1 is 0.93, which means among total actual death (352+4917), 93% of them can be detected as death. As for precision, it measures the percentage of predicted death among total actual death. In this case, our model may have a half-half probability to mistake no death to death.

Table 3 Confusion Matrix of Logistics Regression

	Negative (No Death)	**Positive (Death)**
False (No Death)	82436	4954
True (Death)	352	4917

(3) Classification Report

The classification report, which includes indicators such as precision, recall, f1-score, support for each label, and accuracy, macro avg and weight avg. The report is used for model comparison.

Table 4 Classification Report of Logistic Regression

	precision	**recall**	**f1-score**	**support**
0	1.00	0.94	0.97	87390
1	0.50	0.93	0.65	5269
accuracy			0.94	92659
macro avg	0.75	0.94	0.81	92659
weighted avg	0.97	0.94	0.95	92659

(4) Feature Importance

Below is the feature importance graph, which is according to the rank of coefficients. We can see that CVASC_Cardiac_B and DERMA_Whole_Body_Nos_B are the most important predictors among top 10 predictors, which are more than 80%. The result is sensible since people with heart diseases are listed as high-risk people while patients with body burn are very sensitive to viruses and they have weak resistance towards a disease that can be spread by air.

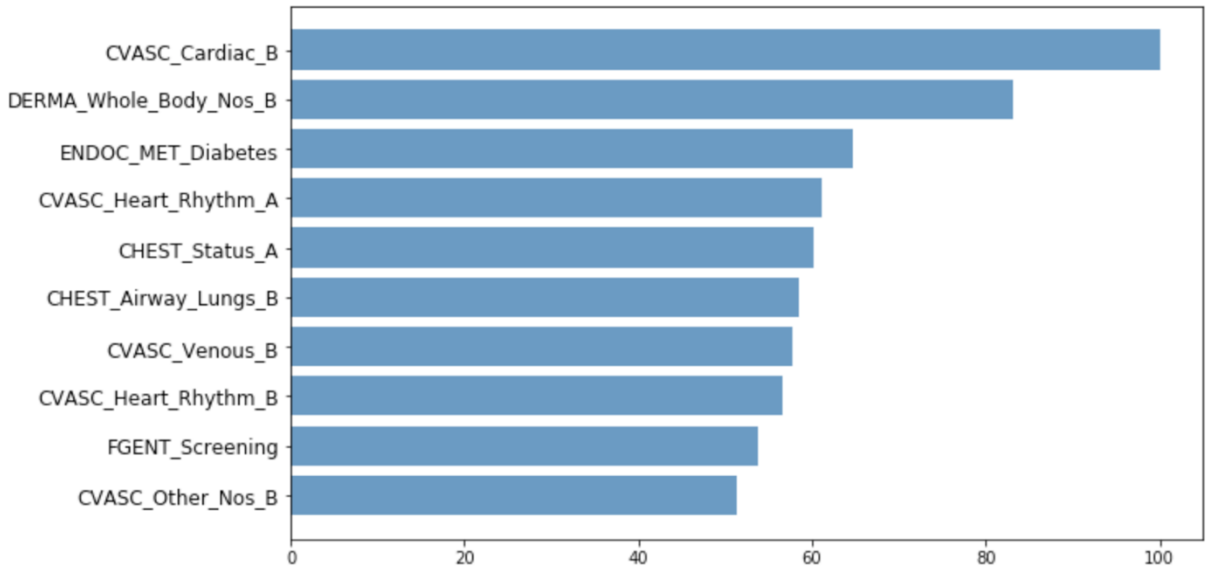

Figure 3 Feature Importance of Logistics Regression

Naïve Bayes

Model Introduction

All features in our dataset are dummy variables which form a sparse matrix. It is a classic scenario to apply a Naive Bayes model. Also, although our data is highly unbalanced, Naive Bayes is not sensitive to unbalanced data and could get use of all the information from our dataset. In this case, we used the Bernoulli Naive Bayes model which usually applied in document classification tasks and natural language process. It treats binary variables as the occurrence of terms which is perfectly suitable for our dataset. Also, it is easy to deploy and train Naive Bayes models since we need to neither set hyper parameters nor over or under sample our dataset. Under such advantages, we decided to make it as our baseline model. Naive Bayes model will generate priors based on the ratio of death/non-death counts to total counts.

Model Interpretation

Since Naive Bayes is a generative model, it will not return coefficients or feature importance. Tables below show the confusion matrix and evaluation scores on the test set. Focusing on the death class, the model has a f1 score of 0.69 with a high recall rate and a low precision, which means that it could detect 87% of death records, although among which some of the predictions are false positive.

Table 5 Confusion Matrix of Naive Bayes

	Predicted non-death	**Predicted death**
True non-death	83878	3512
True death	688	4581

Table 6 Classification Report of Naive Bayes

	precision	**recall**	**f1-score**	**support**
0	0.99	0.96	0.98	87390
1	0.57	0.87	0.69	5269
accuracy			0.96	92659

Bagging ID3 Tree

Model Introduction

Considering that all features we have in this dataset is binary, tree-based models might be a good choice under this scenario. We use ID3 (Iterative Dichotomiser 3) Tree here to help us conduct classification. It recursively searches the local best attribute with information gain and splits data into two subsets (since every feature is binary), and finally makes classification based on the distribution of classes on each leave. ID3 Tree is very easy to overfit the data if we do not restrict the growth of the tree, which leads to another problem that it is hard to find the best complexity that strikes a balance between underfitting and overfitting. So, instead of finding the optimal tree, we rigorously control the complexity even though the tree is too weak. Then we use the technique of bagging and bootstrap to train many ID3 Tree classifiers and then aggregate them into a powerful classifier.

Bagging and bootstrap also help us solve the issue of imbalance data. Stratified bootstrap that was firstly discussed in Pons (2007)[11] inspires us to use bootstrap separately in different classes to rebalance the data. We use the number of observations in the minority class as sample size and sample with replacement in each class. So, every time we can have a balance dataset to train our model. We repeatedly sample many times and build several trees and use the average of their classification results as our final output.

In practice, we have 16001 observations of positive class and 261973 of negative class in training data. We sample 200 times such that every observation has at least a probability of 99.99% to be selected at least once, and thus build 200 classifiers. Then we use these classifiers to make classification and average the result as the final output.

[11] Pons, O. (2007). Bootstrap of means under stratified sampling. Electronic Journal of Statistics, 1, 381-391.

Table 7 Confusion Matrix of ID3 Tree

	Predicted non-death	**Predicted death**
True non-death	83659	3731
True death	518	4751

Table 8 Classification Report of ID3 Tree

	precision	**recall**	**f1-score**	**support**
0	0.99	0.96	0.98	87390
1	0.56	0.90	0.69	5269
accuracy			0.95	92659
macro avg	0.78	0.93	0.83	92659
weighted avg	0.97	0.95	0.96	92659

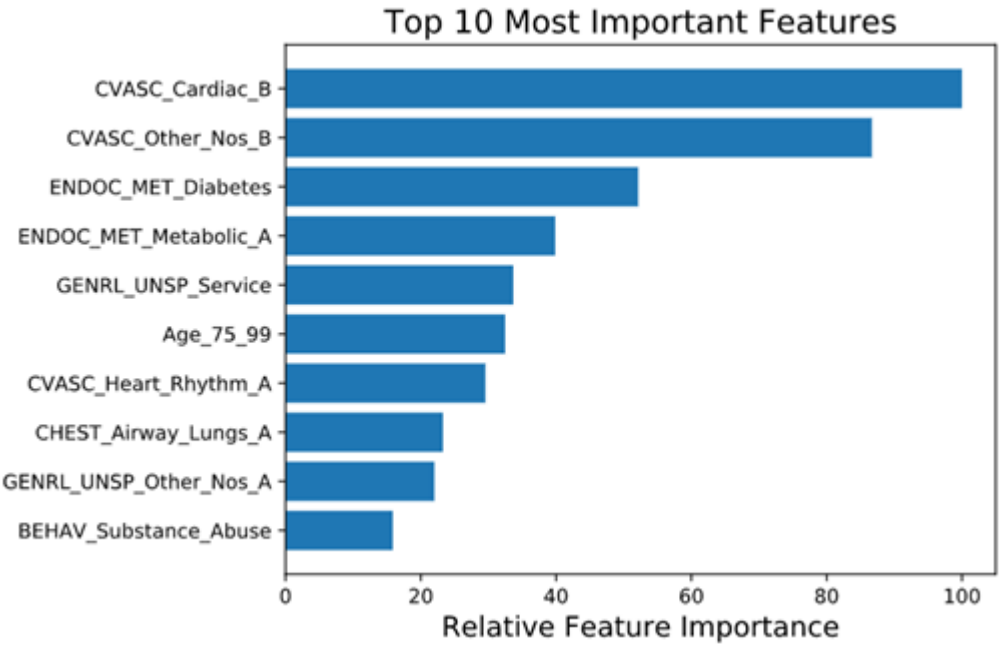

Figure 4 Feature Importance of ID3 Tree

We calculate feature importance in each model based on information gain and then average them to get the final feature importance. According to the plot below, we see that the top 10 most important features are

CVASC_Cardiac_B, CVASC_Other_Nos_B, ENDOC_MET_Diabetes, ENDOC_MET_Metabolic_A, GENRL_UNSP_Service, Age_75_99, CVASC_Heart_Rhythm_A, CHEST_Airway_Lungs_A, GENRL_UNSP_Other_Nos_A, BEHAV_Substance_Abuse. We can see that Top 2 features are both related to heart disease, which we will talk in detail in the insights part.

Random Forest

Model Introduction

Random forest is a bagging ensemble method, which trains a bunch of trees in a parallel way on random subsets and uses averaging to improve the predictive accuracy and control over-fitting. Results generated by random forest models are usually high in variance but low in bias, indicating that they usually run into overfitting problems and are sensitive to the unbalanced shape of a dataset. In order to mitigate the effect, the unbalanced shape brings to the random forest models, a class weight variable is introduced in the modelling process to make the models aware of the unbalanced weight between classes.

Model Interpretation

In this case, unbalanced dataset shape poses challenges to the random forest model, and a class weight of {1: 0.95, 0:0.05}, which is close to the real weight distribution, is set a priori in order for the model to capture the unbalanced shape. Before that, we tried the under-sampling method so that a balanced dataset was used to train the model, however, the results generated by models training on an under-sampled data did not work well on the real unbalanced data. Moreover, though in the beginning we set to use grid search to find the optimal parameters for the model, the process consumed too much time and even one night was not enough to determine the optimal parameters for the model on the branding new macbook:(. So we gave up grid search, and instead, set parameters for the model based on my understanding of the dataset. We set max_features = 180, max_depth = 35 and n_estimators = 100.

Table 9 Confusion Matrix of Random Forest

	Predicted non-death	**Predicted death**
True non-death	86819	571
True death	1879	3390

Table 10 F1 Scores of Random Forest

	precision	recall	f1-score	support
0	0.98	0.99	0.99	87390
1	0.86	0.64	0.73	5269
accuracy			0.97	92659
macro avg	0.92	0.82	0.86	92659
weighted avg	0.97	0.97	0.97	92659

Increasing the number of features or depth did not improve the model significantly anymore, and to avoid the problem of overfitting, we stopped at max_features = 180, max_depth = 35 and n_estimators = 100.

Features of importance are plotted on the graph below. As confirmed in other modelling, CVASC_Arterial_B pops out as the most significant variable to determine the mortality of a COVID-19 patient.

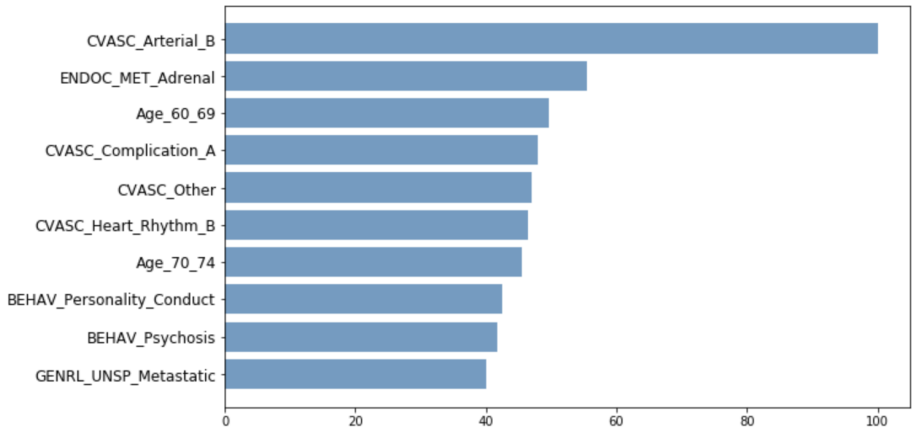

Figure 5 Feature Importance of Random Forest

Boosted Tree

We conducted a gradient boosted tree model to build a prediction model. The model introduction, modeling process, and results interpretation are shown as below.

Model Introduction

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. Gradient boosted tree fits a sequence of trees so that each tree concentrates on misclassified records from the previous tree and gets improvements each time. In other words, the gradient boosted tree is built in a stage-wise fashion and it can generalize them by allowing optimization of loss function.

Gradient boosting tree is especially useful when the labels of records are imbalanced. For example, for death prediction, most labels are non-death, but we are interested in the labels of death. Boosted trees can increase the accuracy of rare classes of special interest. Basic classifiers are tempted to classify cases as belonging to the domain class. And misclassifications with the single best-pruned tree can be those rare classes of special interest, which is what boosting algorithm concentrates on. Although we will rebalance the data first to make two classes balanced and don't have a severe imbalanced problem, it is still good to try a gradient boosted tree model and see whether it can improve the prediction performance.

For the model tuning, we used the Grid Search function to help us find the optimized parameters of the gradient boosted tree model, and finally used n_estimators=75, max_depth=4, learning_rate=0.5.

Finally, we used test data (92,659 observations) to evaluate the model performance, including confusion matrix, F1 score, and top 10 important features. The results were shown and interpreted in the following part.

Results Interpretation

(1) Confusion matrix

According to the confusion matrix of gradient boosted tree model, we concluded that the overall accuracy ($\frac{TN+TP}{TN+TP+FN+FP}$) was 94%.

The precision ($\frac{TP}{TP+FP}$) was 51%, which meant that for the cases we predicted to be dead, 51% of them were true death cases.

The recall ($\frac{TP}{TP+FN}$) was 93%, which meant that for the cases that were dead, our model can predict them correctly with accuracy of 93%.

Since it was bad to make mistakes, we further analyzed the mistakes and costs of the gradient boosted tree model. There are two different kinds of mistakes: false positive and false negative. False positive means that we regard a health person as an at-risk case. This can lead to unnecessary procedures such as quarantine and bring stress to people who are predicted as high risk but actually not. Another problem is false negative, which can be more serious because it means that the model can't predict the high-risk cases and regard them as safe. In such conditions, people with high risk don't get treatment, infect others and eventually die.

According to the confusion matrix, we concluded that the false positive ($\frac{FP}{FP+TN}$) was 49%, and the false negative ($\frac{FN}{TP+FN}$) was 7%. Gradient boosting tree model may make a bigger mistake on predicting a healthy person as a high-risk one, but it was not a big deal because unnecessary quarantine didn't cost too much. Fortunately, our false negative was very low which meant that we can predict the at-risk population with slight mistakes. This is much more important because we can not only save lives by providing early treatments on the high-risk population, but also ensure the safety of the healthy population by getting rid of being infected.

Table 11 Confusion matrix of boosted tree

	Predicted non-death	**Predicted death**
True non-death	82624	4766
True death	374	4895

(2) F1-score

Furthermore, we analyzed the F1 score of each class, especially class 1 which represented death. The results were shown as below.

F1 score ($2 \times \frac{Precison \times Recall}{Precision+Rcall}$) is the harmonic mean of Precision and Recall, and it gives a better measure of the incorrectly classified cases than the accuracy metric. We used the harmonic mean since it penalized the extreme values. Accuracy can be used when the class distribution is similar while F1-score is a better metric when there are imbalanced classes. Thus, in this case, it will be more reasonable to look at F1-score.

According to the results, we concluded that the F1-score of class 1 (death) was 0.66, which was slightly lower than that of our benchmark model - Naive Bayes.

Table 12 Comparison Report of Boosted Tree

	precision	recall	f1-score	support
0	1.00	0.95	0.97	87390
1	0.51	0.93	0.66	5269
accuracy			0.94	92659
macro avg	0.75	0.94	0.81	92659
weighted avg	0.97	0.94	0.95	92659

(3) Feature importance

Finally, we cared about the important features of the model that influenced the most. The importance of feature measured how important a feature was in the context of a model. That is, it measured how much (based on the training data) the accuracy of a model would decrease if that feature were removed.

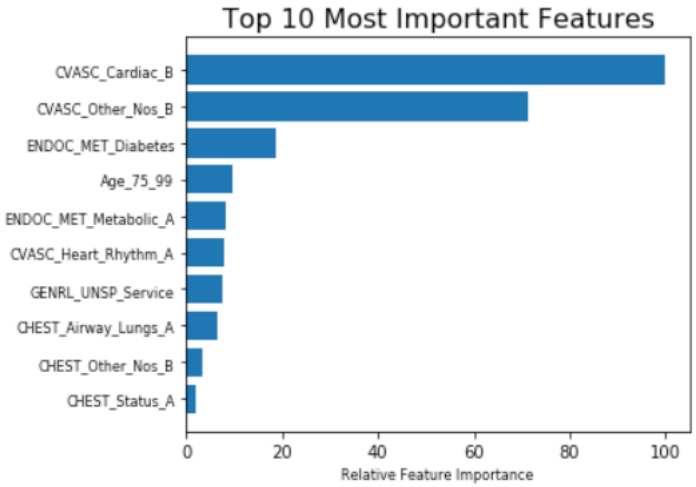

Figure 6 Feature Importance of Boosted Tree

According to the plot below, we can see that the top10 most important features are 'CHEST_Status_A', 'CHEST_Other_Nos_B', 'CHEST_Airway_Lungs_A', 'GENRL_UNSP_Service', 'CVASC_Heart_Rhythm_A', 'ENDOC_MET_Metabolic_A', 'Age_75_99', 'ENDOC_MET_Diabetes', 'CVASC_Other_Nos_B', and 'CVASC_Cardiac_B'.

We made a simple conclusion on these important features: a person aging from 75 to 99 with some diseases such as cardiac and heart problems, lungs problems, and diabetes may have a higher risk to die from COVID-19. We will further discuss this and provide deeper insights in the Section Insights and Suggestion.

Neutral Network

We created neural network to predict the death rate of COVID-19. And the model introduction, modeling process and results interpretation are discussed below.

Model Introduction

Neural networks are a set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns. They interpret sensory data through a kind of machine perception, labeling or clustering raw input. The patterns they recognize are numerical, contained in vectors, into which all real-world data, be it images, sound, text or time series, must be translated.

Neutral Networks have some key advantages that make them most suitable for coronavirus death prediction problems and situations. Neural networks have the ability to learn and model non- linear and complex relationships. In real life, many relationships between the predictor and target variable are nonlinear and complex. For instance, our target variable is the binary variable, death. However, we have a lot of predictors, different diseases such as heart disease, lung disease, and patient information such as age group, gender. The relationship between them may be intricate. Thus, neutral networks can help us to clarify their relationship and predict in a high accuracy.

Model Interpretation

(1) Confusion Matrix

Precision talks about how precise a model is out of those predicted positives, how many of them are actual positives. In this case, our precision is 0.62.

Recall calculates how many of the actual positives our model captures through labeling it as Positive. Recall shall be the model metric we use to select our best model when there is a high cost associated with False Negative. In this case, our recall is 92%.

Table 13 Confusion Matrix of Neutral Network

	Predicted non-death	**Predicted death**
True non-death	84377	3013
True death	433	4836

(2) F1-score

We analyzed the F1 score of each class, especially class 1 which represented death. The results were shown as below.

F1 Score is needed when we want to seek a balance between Precision and Recall. Accuracy can be largely contributed by a large number of True Negatives. F1 Score might be a better measure to use if we need to seek a balance between Precision and Recall. And there is an uneven class distribution (large number of actual negatives). Thus, in this case, F1 score will be better for our death prediction.

According to the results, we concluded that the F1-score of class 1 (death) was 0.74.

Table 14 Comparison Report of Neural Network

	precision	**recall**	**f1-score**	**support**
0	0.99	0.97	0.98	87390
1	0.62	0.92	0.74	5269
Accuracy			0.96	92659
Macro avg	0.81	0.94	0.86	92659
Weighted avg	0.97	0.96	0.97	92659

Model Comparison

Here we compare the performances among all models discussed above. We evaluate a model from four metrics on the minor class: precision, recall, f1-score and accuracy. As shown below, Neural Network achieved the best performance with the highest f1-score, which means this model strikes a balance in classifying both classes. We also find some interesting things in other models. Random Forest, for example, gained a highest precision of 86%. This indicates that Random Forest has the best performance in making classification in the minority class. However, it only has a recall rate of 64%, much lower than other models. Thus, Random Forest is more likely to misclassify the majority class. Other models such Logistic Regression and Boosting Tree also perform well in recall rate but sacrifice their precision rate.

Table 15 Model Performance Comparison

	precision	**recall**	**f1-score**	**accuracy**
Logistic Regression	0.5	0.93	0.65	0.94
Naïve Bayes	0.57	0.87	0.69	0.96
Bagging ID3 Tree	0.56	0.9	0.69	0.95
Random Forest	0.86	0.64	0.73	0.97
Boosting Tree	0.51	0.93	0.66	0.94
Neural Networks	0.62	0.92	0.74	0.96

Though we already demonstrate that our models can perform well in classification. These machine learning models still remain meaningless black boxes if we cannot interpret the logic behind them. One nice feature that Tree-based models have is that we can learn how they split the data and thus explain how they make predictions. So, here we export feature importances from our tree-based models. In addition, we select the top 10 features with largest absolute coefficient and put them into the comparison of feature importance. The table below shows the ranking of feature importance according to different models.

Table 16 Top 10 Feature Importance According to Different Models

	Boosting Tree	**Random Forest**	**Bagging ID3 Tree**	**Logistic Regression**
Age_60_69		3		
Age_70_74		7		
Age_75_99	**7**		**6**	
BEHAV_Personality_Conduct		8		
BEHAV_Psychosis		9		
BEHAV_Substance_Abuse			10	
CHEST_Airway_Lungs_A	**3**		**8**	
CHEST_Airway_Lungs_B				6
CHEST_Other_Nos_B	**2**		**2**	
CHEST_Status_A	**1**		**1**	**5**
CVASC_Arterial_B		1		
CVASC_Cardiac_B	**10**			**1**
CVASC_Complication_A		4		
CVASC_Heart_Rhythm_A	**5**		**7**	**4**
CVASC_Heart_Rhythm_B		**6**		**8**
CVASC_Other		5		
CVASC_Other_Nos_B				10
CVASC_Other_Nos_B	9			
CVASC_Venous_B				7
DERMA_Whole_Body_Nos_B				2
ENDOC_MET_Adrenal		2		
ENDOC_MET_Diabetes	**8**		**3**	**3**
ENDOC_MET_Metabolic_A	**6**		**4**	
FGENT_Screening				9
GENERL_UNSP_Metastatic		10		

GENRL_UNSP_Other_Nos_A		9
GENRL_UNSP_Service	**4**	**5**

We highlight features that are indicated to be important in more than one model and then summarized in 10 most important factors that make COVID-19 more deadly. We will cross validate the result of this part with text analysis and literature review and give a deeper analysis in the next few sections.

Text Analysis

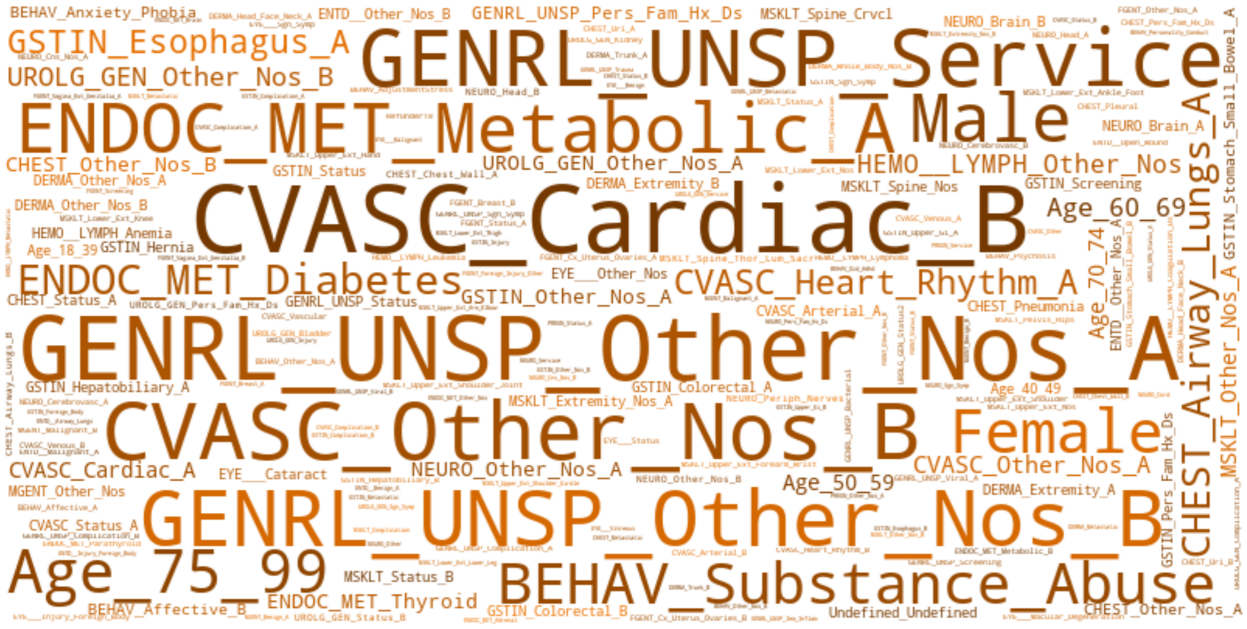

Figure 7 Word Cloud

A word cloud is a visual representation of word frequency. The more commonly the term appears within the text being analyzed, the larger the word appears in the image generated. In this project, word clouds have been applied to analyze the Predictive Analytics for Risk Profiling of Covid-19 Patients to see how each feature or illness is affecting death rate. We filter all of the patient information from the death group and create the word cloud.

CVASC_Cardiac_B is the first thing that catches people's attention, which stands for heart disease. More specifically, CVASC_Cardiac_B includes congenital heart, myocarditis, endocarditis, heart failure, heart

injury including hemopericardium, heart neoplasm and heart transplant cad. The second biggest risk of death is GENRL_UNSP_OTHER_Nos_A, which means the abnormal findings lab or exam, blood results. In addition, CVASC_Other_Nos_B, GENRL_UNSP_Other_Nos_B and GENRL_UNSP_Service ENDOC_Met_Metaboliv_A are also the key factors of death.

From above, it comes to conclude that people with some heart disease, especially classified in Cardiac_B and Other_Nos_B, have high risk of death. Libby and Ridker, who are practicing cardiologists at Brigham and Women, say COVID-19-related heart injury could happen in any several aspects. people with preexisting heart disease are at a greater risk for severe cardiovascular and respiratory complications from COVID-19.According to research, heart attacks can actually be brought respiratory infections such as the flu. Secondly, some people may experience heart damage that mimics heart attack injury. COVID-19 may be triggered by a mismatch between oxygen supply and oxygen demand. Fever and inflammation accelerate heart rate and increase metabolic demands on many organs. That stress is compounded when lungs are infected and incapable of exchanging oxygen and carbon dioxide. Apart from heart disease, ENDOC_MET_Metabolic is also a large category leading to death.

Metabolic is a vital part of keeping people healthy. When metabolic inflammation happens, it could compromise the immune system. In this way, the body will have lower ability to defend the infection, so the healing process will be impaired, and recovery will be much longer than others.
At the lower left corner, Age 75-99 is the majority of death from COVID-19. As people get older, lungs are not as elastic or as resilient as when people are younger. Besides, elderly people might couple other chronic diseases, which may cause this loss of airway function and respiratory function.

In conclusion, a person aging from 75 to 99 with heart problems, metabolic and other abnormal lab results may have a higher risk to die from COVID-19.

Insights and suggestions

Comparing all predictive models and important features, we concluded top 10 common important features that had the highest predictive influence on whether a person was at a high risk of death from COVID-19. The features and their descriptions were listed as below.

Table 17 Top10 Common Features of Importance

DGL_3_Extend	Description
CVASC_Cardiac_B	Cardiac diseases such as heart failure
CVASC_Other_Nos_b	Mixed diseases include lung, heart, blood pressure, blood vessel, etc.
ENDOC_MET_Diabetes	Diabetes and concomitant disease like diabetic chronic kidney disease
ENDOC_MET_Metabolic_A	Metabolic abnormality such as obesity
GENRL_UNSP_Service	Aftercare, counseling, and medical exam
Age_75_99	Elder population
CVASC_Heart_Rhythm_A	Irregular heartbeats such as tachycardia
CHEST_Airway_Lungs_A	Lung and respiratory abnormalities
GENRL_UNSP_Other_Nos_A	Mixed diseases include infection of liver, immunodeficiency, etc.
CHEST_Status_A	Lung and respiratory abnormalities

We drew a patient persona that have a higher risk of dying from COVID-19:

S/he is an elder person aging from 75 to 99 or people of any age with existing health problems. Those health problems are cardiac and heart rhythm diseases, diabetes and the chronic disease brought by it such as diabetic chronic kidney disease, metabolic abnormality such as obesity, lung and respiratory abnormalities, blood pressure and vessel problems, liver failure, and diseases of the immune system. S/he may also rely on aftercare and medical exams.

Our model results were completely consistent with those of CDC and other scientific sources and can be cross verified with each other.

According to the report of centers for disease control and prevention (CDC)[12], older adults and people of any age who have serious underlying medical conditions might be at higher risk for severe illness from COVID-19.

[12] https://www.cdc.gov/coronavirus/2019-ncov/need-extra-precautions/people-at-higher-risk.html

To be specific, those at high-risk for severe illness from COVID-19 are:

- People 65 years and older
- People who live in a nursing home or long-term care facility

People of all ages with underlying medical conditions, particularly if not well controlled, including:

- People with chronic lung disease or moderate to severe asthma
- People who have serious heart conditions
- People who are immunocompromised, for example cancer treatment, smoking, bone marrow or organ transplantation, immune deficiencies, poorly controlled HIV or AIDS, and prolonged use of corticosteroids and other immune weakening medications
- People with severe obesity (body mass index [BMI] of 40 or higher)
- People with diabetes
- People with chronic kidney disease undergoing dialysis
- People with liver disease

Apart from the key features above, our other important features were also verified, such as hypertension (high blood pressure), which was also thought as an underlying medical condition that drove people to be at risk for severe illness from COVID-19 according to the government of Canada[13].

Harvard Medical School claimed that about 10% of patients with pre-existing cardiovascular disease (CVD) who contract COVID-19 will die, compared with only 1% of patients who are otherwise healthy. Increased risk has also been seen in people with high blood pressure (hypertension) and coronary artery disease (CAD)[14]. It also proved that cardiac disease was the most important feature in almost all predictive models we generated.

Figuring out people who are at higher risk for severe illness from COVID-19, we provided three suggestions to protect these people.

[13] https://www.canada.ca/en/public-health/services/publications/diseases-conditions/people-high-risk-for-severe-illness-covid-19.html

[14] https://www.health.harvard.edu/blog/how-does-cardiovascular-disease-increase-the-risk-of-severe-illness-and-death-from-covid-19-2020040219401

First of all, we should make sure people with high-risk stay at home even when we are making reopening efforts. We should be careful and cautious to allow those people to go back to the workplace.

Moreover, the government may consider employing them and pick up their salary until a vaccine is made available to them. According to the research, personal financial condition also affected the probability of disease[15]. It is important to ensure their financial power to enhance their ability to resist risks.

Finally, if a vaccine is available, we should prioritize the at-risk population for early reception of the vaccine. Then we can consider making it available for the population with low risk.

[15] https://www.kff.org/disparities-policy/issue-brief/low-income-and-communities-of-color-at-higher-risk-of-serious-illness-if-infected-with-coronavirus/

Conclusion

In this report, we try to find those factors that have significant influence on morality based demographic and diagnostic characteristics of about 400,000 Covid-19 patients. We combine the advantages of several methods such as literature review, text analysis and machine learning. In text analysis, we use word cloud to visualize patient information from the death group with relatively higher frequency, which gives us a descriptive perspective of factors that have the closest relationship with mortality rate. Then, we conduct a predictive analysis with machine learning to gain a deeper understanding of the relationship between mortality rate and patient information. We carefully select models suitable for the predictive scenario that we are facing, and conclude in Logistic Regression, Naive Bayes, ID3 Tree, Random Forest, Boosted Tree and Neural Networks. As a result, we get the best performance from Neural Networks. However, tree-based models give us a detailed interpretation about how they relate the features of patients with mortality rate.

We finally combine results drawn from data with literature research and find they are highly consistent. In summary, we can give a persona of those who are much more vulnerable to the COVID-19 from two perspectives. From the aspect of demographic information, those who are 65 years and older have a higher mortality rate. On the other hand, our analyses show that chronic lung disease, serious heart conditions, immunocompromised, severe obesity, diabetes, chronic kidney disease or liver disease also lead to high mortality.

Considering many states start to generally reopen their economies, our findings can help state governments and organizations find those who are more vulnerable to COVID-19. Therefore, those people can be carefully protected, and others can return to work to maintain their livelihood. It is of great importance to strike a balance between economy and safety.

Appendix

Data preparation

```
# In[110]:
# load important stuff
get_ipython().run_line_magic('matplotlib', 'inline')
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.preprocessing import StandardScaler
# These are some new scikit learn toys that will make your life easier (see
below)
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import ShuffleSplit
from sklearn.pipeline import make_pipeline
# ### Read data
# In[67]:
import os
os.chdir ('/Users/tangziyu/Desktop/AI Opening')
# In[6]:
covid = pd.read_csv('Covid19_SourceFile.csv')
sample = pd.read_csv('1000AnalyticFileSample.csv')
dx = pd.read_csv('ICD10_Dx_TaxonomyTree.csv')
age = pd.read_csv('age.csv')
# In[7]:
age.head()
# In[8]:
sample.head()
# In[9]:
dx.head()
# In[10]:
covid.head()
# ### Data Preparation
# #### convert gender to be 0-1 dummy
# In[11]:
covid = covid.replace({'SexCode': 2.0}, 0)
covid['SexCode'] = covid['SexCode'].fillna(0)
convert_dict = {'SexCode': int
                }
covid = covid.astype(convert_dict)
covid = covid.rename(columns={'SexCode': 'Gender_Male'})
# #### map age
# In[12]:
covid_age = pd.merge(covid,
                     age[['AgeCode', 'Map to ']],
                     on='AgeCode')
map_age = pd.get_dummies(covid_age['Map to '])
map_age = map_age[map_age.columns[0:6]]
covid_age = pd.concat([covid_age, map_age], axis=1)
covid_age = covid_age.drop(['AgeCode', 'Map to '],axis=1)
covid = covid_age
covid = covid.sort_values(by=['Patient ID'])
# ##### check column with all nulls
# In[13]:
print(covid.isnull().all())
# #### convert code of dx to class name
# In[14]:
names = ['DX1',
 'DX2',
 'DX3',
 'DX4',
 'DX5',
 'DX6',
 'DX7',
 'DX8',
 'DX9',
 'DX10',
 'DX11',
 'DX12',
 'DX13',
 'DX14',
 'DX15',
 'DX16',
 'DX17',
 'DX18',
 'DX19',
 'DX20',]
for name in names:
    covid[name] = covid[name] .map(dx.set_index('ICD10 Dx
Code')['DGL_3_Extend'])
# ##### map dxs to be dummies according to requirements
# In[15]:
covid_dx = covid
```

```
covid_dx = covid_dx.melt('Patient ID').groupby(['Patient ID',
'value']).size().unstack(fill_value=0)
# In[16]:
covid_dx = covid_dx.reset_index()
covid_final = pd.merge(
                    covid[['Patient ID','Mortality (1=
death)','Age_18_39','Age_40_49','Age_50_59',

'Age_60_69','Age_70_74','Age_75_99','Gender_Male']],covid_dx,
                    on='Patient ID')
# In[17]:
covid_final.columns.tolist()
# ##### convert 0-1 dummies
# In[18]:
covid_final = covid_final.drop([0,1],axis=1)
names = covid_final.columns.tolist()[9:]
for column in names:
    covid_final.loc[covid_final[column]!=0, column] = 1
# In[19]:
# check the dummy variable - only 0 and 1
for column in names:
    print(covid_final[column].max())
# #### check the results with sample data
# In[20]:
covid_final.iloc[0]
# In[21]:
sample.columns.difference(covid_final.columns)
# In[22]:
len(dx['DGL_3_Extend'].unique())
# Besides, some of the dx don't exist in the cases - that's why we just have
220 dx instead of 229.
# ### Export data
# In[434]:
covid_final.to_csv('final result.csv')

In[2]:

clean = pd.read_csv('clean_data.csv', index_col=0)

# In[3]:

raw_data = pd.read_csv('Covid19_SourceFile.csv')
category = pd.read_csv('ICD10_Dx_TaxonomyTree.csv')

# In[4]:

len(category.DGL_3_Extend.unique())

# In[5]:

clean['Mortality (1= death)'].sum()/len(clean)

# In[6]:

clean['Mortality (1= death)'].sum()

# In[7]:

len(clean.columns)

# In[8]:

clean.shape

# In[ ]:


# ## Data description
# In[19]:

# Gender bias
cross_tab = pd.crosstab(clean['Mortality (1= death)'], clean['Gender_Male'])

# In[21]:

cross_tab

# In[20]:

scipy.stats.fisher_exact(cross_tab)

# In[9]:

clean.head()

# In[11]:

clean.loc[clean['Mortality (1= death)']==1, 'Age_75_99'].sum()

# In[14]:

clean['Age_75_99'].sum()
```

In[24]:

```
clean_death = clean[clean['Mortality (1= death)'] == 1]
clean_health = clean[clean['Mortality (1= death)'] == 0]
```

In[25]:

```
clean_death_sum = clean_death.sum()
clean_health_sum = clean_health.sum()
```

Age group

In[66]:

```
fig, ax = plt.subplots(2, 1, sharex=True)
clean_death_sum[clean.columns[2:8]].plot(kind='bar', use_index=True,
color='orange', ax=ax[0], figsize=(6,8), label='Death')
clean_health_sum[clean.columns[2:8]].plot(kind='bar', use_index=True,
ax=ax[1], label='Non-death')
ax[0].legend()
ax[1].legend()
ax[0].set_title('Counts on Age Groups')
plt.show()
```

Gender

In[ ]:

DX Group

Top10 DX

In[78]:

```
clean_death_sum[clean.columns[9:]].sort_values(ascending=False).head(10
).plot.barh(color='orange')
plt.title('Top 10 DX Group on Death')
plt.show()
```

In[73]:

```
top10_death =
clean_death_sum[clean.columns[9:]].sort_values(ascending=False).head(10
).index
category[category.DGL_3_Extend.isin(top10_death)]
```

In[74]:

```
top10_death
```

In[79]:

```
clean_health_sum[clean.columns[9:]].sort_values(ascending=False).head(1
0).plot.barh()
plt.title('Top 10 DX Group on non-Death')
plt.show()
```

naive bayes without oversampling

In[26]:

```
model = BernoulliNB()
param_grid={'alpha': [2.25]}
shuffle_split = ShuffleSplit(test_size=0.25, n_splits=50)
grid_search=GridSearchCV(model, param_grid, cv=shuffle_split,
scoring='roc_auc', return_train_score=True)
grid_search.fit(X,y)
results = pd.DataFrame(grid_search.cv_results_)
print(results[['rank_test_score','mean_test_score','param_alpha']])
```

In[39]:

```
model = BernoulliNB(alpha=2.25)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=0)
model.fit(X_train, y_train)
```

In[40]:

```
cm = confusion_matrix(y_test, model.predict(X_test))
```

In[41]:

```
pd.DataFrame(cm)
```

In[42]:

```
pd.DataFrame(precision_recall_fscore_support(y_test,
model.predict(X_test)), index=['precision', 'recall', 'f1_score', 'support'])
```

With over-sample SVM classifier

In[38]:

```
X.shape
```

In[ ]:

```
pd.read_csv('')
```

Logistic model

```
#!/usr/bin/env python
# coding: utf-8
# ## bagging+bootstrap logistic model
# In[2]:
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
from functools import reduce
from sklearn.metrics import confusion_matrix, accuracy_score, f1_score,
classification_report
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
# In[3]:
df = pd.read_csv('/Users/jemma/Downloads/final result.csv')
df = df[df.columns[2:-1].to_list()]
X = df[df.columns[1:]].values
y = df['Mortality (1= death)'].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=0)
# In[4]:
class LogisticRegressionClassifier:
    def __init__(self, n=100):
        self.n = n
    def fit(self, X, y, random_state=42, criterion='entropy'):
        X_train_p = X[y==1]
        X_train_n = X[y==0]
        y_train_p = y[y==1]
        y_train_n = y[y==0]
        l = sum(y_train_p)
        clf_list = []
        np.random.seed(seed=random_state)
        for i in range(self.n):
            idx_n = np.random.choice(range(X_train_n.shape[0]), l)
            idx_p = np.random.choice(range(X_train_p.shape[0]), l)
            X_train_f = np.concatenate([X_train_n[idx_n, :],
X_train_p[idx_p]])
            y_train_f = np.concatenate([y_train_n[idx_n],
y_train_p[idx_p]])
            clf = LogisticRegression()
            clf.fit(X_train_f, y_train_f)
            clf_list.append(clf)
            print(f'\rClassifier {i+1} completed.', end='')
        self.clfs = clf_list

    def coefficients(self):
        feature_importance = abs(self.clfs[0].coef_)
        for tree in self.clfs[1:]:
            feature_importance += tree.coef_
        return feature_importance/ self.n

    def predict(self, X):
        result_list = list(map(lambda clf: clf.predict_proba(X)[:,1],
self.clfs))
        score = reduce(lambda a, b: a+b, result_list) / len(self.clfs)
        return score

    def report(self, X, y):
        y_hat = self.predict(X)
        print(classification_report(y_test, 1.*(y_hat>0.5)))
# In[5]:
btc = LogisticRegressionClassifier(1)
btc.fit(X_train, y_train)
y_hat = 1.0 * (btc.predict(X_test)>0.5)
# In[6]:
btc.coefficients()
# In[8]:
reshaped_cof = pd.DataFrame(btc.coefficients().reshape(-1,1))
predictors = pd.DataFrame(df.columns)
predictors_final = predictors.drop(predictors.index[0], axis=0)
predictors_final.index = np.arange(1,len(predictors_final)+1)
reshaped_cof.index =    np.arange(1,len(reshaped_cof)+1)
# In[9]:
coefficient_tb1= pd.concat([predictors_final,reshaped_cof], axis=1,
ignore_index=True)
coefficient_tb2 = coefficient_tb1.rename(columns={0: "Predictor", 1:
"Coefficient"})
coefficient_tb_final =
coefficient_tb2.sort_values(by=["Coefficient"],ascending=False)
coefficient_tb_final.head(10)
coefficient_tb_final.to_csv('coefficient_tb_final.csv')
# In[22]:
print(classification_report(y_true=y_test, y_pred=y_hat))
# In[11]:
f1_score(y_true=y_test, y_pred=y_hat)
# In[12]:
accuracy_score(y_true=y_test, y_pred=y_hat)
```

```
# In[13]:
confusion_matrix(y_true=y_test, y_pred=y_hat)
# In[123]:
import matplotlib as mpl
import matplotlib.pyplot as plt
feature_importance =
abs(coefficient_tb_final[coefficient_tb_final.columns[1]])
feature_importance = 100.0 * (feature_importance /
feature_importance.max())
score = feature_importance.head(10)
predictors_10 =
coefficient_tb_final[coefficient_tb_final.columns[0]].head(10)
feature_importance_table = pd.concat([predictors_10,score], axis=1)
feature_importance_table
feature_importance_table_desc =
feature_importance_table.sort_values(by=["Coefficient"],ascending=True)
feature_importance_table_desc
# In[124]:
fig = plt.figure(figsize=(10,6))
share =
feature_importance_table_desc[feature_importance_table_desc.columns[1]]
plt.barh(range(10), share, align = 'center',color='steelblue', alpha = 0.8)
plt.yticks(range(10),feature_importance_table_desc[feature_importance_tab
le_desc.columns[0]], fontsize=12)
ax.set_xlabel('Relative Feature Importance (%)')
#plt.tight_layout()
plt.show()
# ID3 Tree
#!/usr/bin/env python
# coding: utf-8
# ## ID3 Tree
# ##### Jimmy Hu
# In[6]:
import numpy as np
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
# In[2]:
df = pd.read_csv('final result.csv')
df = df[df.columns[2:-1].to_list()]
X = df[df.columns[1:]].values
y = df['Mortality (1= death)'].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,
random_state=42)
# In[12]:
dtc = DecisionTreeClassifier(criterion='entropy', ccp_alpha=0.001)
dtc.fit(X_train, y_train)
# In[50]:
feature = dtc.feature_importances_ > 0
d = dict(zip(list(df.columns[1:].values[feature]),
                list(dtc.feature_importances_[feature].round(3))))
ranking = dict(sorted(d.items(), key=lambda x: x[1], reverse=True))
# In[13]:
dtc.get_depth()
# In[14]:
dtc.score(X_test, y_test)
# In[49]:
print('Feature Importance:')
for i in ranking.keys():
    print(ranking[i], i)
```

Random Forest

```
#!/usr/bin/env python
# coding: utf-8
# In[3]:
# load important stuff
get_ipython().run_line_magic('matplotlib', 'inline')
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.preprocessing import StandardScaler
# These are some new scikit learn toys that will make your life easier (see
below)
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import ShuffleSplit
from sklearn.pipeline import make_pipeline
# In[1]:
import os
os.chdir ('/Users/xupech/Desktop/brandeis graduate school/Academics/2020
SPRING/HS 339 ADVANCED HEALTHCARE ANALYTICS/Case
4/Data')
# In[4]:
covid = pd.read_csv('final result.csv')
covid.dtypes
# In[51]:
covid.head()
# In[5]:
y = covid['Mortality (1= death)']
```

```
X = covid.iloc[:,3:-1]
print(np.mean(y))
print(len(y))
X.head()
# In[6]:
X_train_set, X_test_set, y_train_set, y_test = train_test_split(X, y,
train_size=0.75, test_size=0.25, random_state = 0)
#scaler = StandardScaler().fit(X_train_set)
#X_train_set = scaler.transform(X_train_set)
#X_test_set = scaler.transform(X_test_set)
len(X_test_set)
# In[ ]:
#X_train_set.shape
#y_train_set = np.array(y_train_set)
#y_train_set=np.reshape(y_train_set, (-1,1))
#y_train_set.T
covidt = np.concatenate((X_train_set, y_train_set), axis=1)
covidt
covidt[:, -1]
# generate special classes for types
aclass0 = covidt[covidt[:,-1]==0]
aclass1 = covidt[covidt[:,-1]==1]
count0 = len(aclass0)
count1 = len(aclass1)
print(count0)
print(count1)
# under sampling
# Generate sample of class 0 types matching number of class 1 types
n_samples= count1
inds = np.random.randint(0,count0, size=n_samples)
under0 = aclass0[inds]
covidus = np.concatenate((under0, aclass1), axis=0)
# set up data, and check if balanced
ytrain = covidus[:,-1]
Xtrain = covidus[:,:-1]
print(np.mean(ytrain))
# In[ ]:
# Try for random forest and grid search
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
nmc = 100
#    Note:    no need for scaling, back to simple parameters
# set up dictionary for grid search
param_grid={'max_features':[50,100,150],'max_depth':[10,20,35],'n_estima
tors':[100]}
# set up cross-validation shuffles
cvf = ShuffleSplit(test_size=0.25,n_splits=nmc)
# set up search
grid_search=GridSearchCV(RandomForestClassifier(class_weight =
{0:0.95, 1:0.05}),param_grid,cv=cvf,scoring = 'f1',
return_train_score=True)
# implement search
grid_search.fit(X_train_set,y_train_set)
# move results into DataFrame
results = pd.DataFrame(grid_search.cv_results_)
print(results[['rank_test_score','mean_test_score','param_max_features','par
am_max_depth']])
# In[14]:
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(max_features=150, max_depth=35,
n_estimators=100, class_weight = {0:0.95, 1:0.05})
model.fit(X_train_set, y_train_set)
# In[15]:
y_pred = model.predict(X_test_set)
y_pred
# In[16]:
from sklearn.metrics import confusion_matrix, accuracy_score, f1_score,
classification_report, roc_auc_score, confusion_matrix
print(classification_report(y_test, y_pred))
# In[17]:
print(confusion_matrix(y_true=y_test, y_pred=y_pred))
# In[18]:
model.feature_importances_
# In[20]:
reshaped_cof = pd.DataFrame(model.feature_importances_.reshape(-1,1))
predictors = pd.DataFrame(covid.columns)
predictors_final = predictors.drop(predictors.index[0], axis=0)
predictors_final.index = np.arange(1,len(predictors_final)+1)
reshaped_cof.index =    np.arange(1,len(reshaped_cof)+1)
# In[21]:
coefficient_tb1= pd.concat([predictors_final,reshaped_cof], axis=1,
ignore_index=True)
coefficient_tb2 = coefficient_tb1.rename(columns={0: "Predictor", 1:
"Coefficient"})
coefficient_tb_final =
coefficient_tb2.sort_values(by=["Coefficient"],ascending=False)
coefficient_tb_final.head(10)
coefficient_tb_final.to_csv('coefficient_tb_final.csv')
# In[22]:
import matplotlib as mpl
import matplotlib.pyplot as plt
```

```
feature_importance =
abs(coefficient_tb_final[coefficient_tb_final.columns[1]])
feature_importance = 100.0 * (feature_importance /
feature_importance.max())
score = feature_importance.head(10)
predictors_10 =
coefficient_tb_final[coefficient_tb_final.columns[0]].head(10)
feature_importance_table = pd.concat([predictors_10,score], axis=1)
feature_importance_table
feature_importance_table_desc =
feature_importance_table.sort_values(by=["Coefficient"],ascending=True)
feature_importance_table_desc
# In[24]:
fig = plt.figure(figsize=(10,6))
share =
feature_importance_table_desc[feature_importance_table_desc.columns[1]]
plt.barh(range(10), share, align = 'center',color='steelblue', alpha = 0.8)
plt.yticks(range(10),feature_importance_table_desc[feature_importance_tab
le_desc.columns[0]], fontsize=12)
#ax.set_xlabel('Relative Feature Importance (%)')
#plt.tight_layout()
plt.show()
```

Neural Network

```
#!/usr/bin/env python
# coding: utf-8
# ## Neural network
# In[1]:
import numpy as np
import pandas as pd
from functools import reduce
from sklearn.metrics import confusion_matrix, accuracy_score, f1_score,
classification_report
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
# In[2]:
import os
os.chdir ('/Users/hainayan/Downloads')
df = pd.read_csv('final result.csv')
df = df[df.columns[2:-1].to_list()]
X = df[df.columns[1:]].values
y = df['Mortality (1= death)'].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=0)
# In[3]:
class BaggingTreeClassifier:
    def __init__(self, n=100):
        self.n = n
    def fit(self, X, y, random_state=42, criterion='entropy'):
        X_train_p = X[y==1]
        X_train_n = X[y==0]
        y_train_p = y[y==1]
        y_train_n = y[y==0]
        l = sum(y_train_p)
        clf_list = []
        np.random.seed(seed=random_state)
        for i in range(self.n):
            idx_n = np.random.choice(range(X_train_n.shape[0]), l)
            idx_p = np.random.choice(range(X_train_p.shape[0]), l)
            X_train_f = np.concatenate([X_train_n[idx_n, :],
X_train_p[idx_p]])
            y_train_f = np.concatenate([y_train_n[idx_n],
y_train_p[idx_p]])
            clf = MLPClassifier(solver = 'lbfgs',random_state=0)
            clf.fit(X_train_f, y_train_f)
            clf_list.append(clf)
            print(f'\rClassifier {i+1} completed.', end='')
        self.clfs = clf_list

    def predict(self, X):
        result_list = list(map(lambda clf: clf.predict_proba(X)[:,1],
self.clfs))
        score = reduce(lambda a, b: a+b, result_list) / len(self.clfs)
        return score

    def report(self, X, y):
        y_hat = self.predict(X)
        print(classification_report(y_test, 1.*(y_hat>0.5)))
# In[25]:
btc = BaggingTreeClassifier(500)
btc.fit(X_train, y_train)
y_hat = 1.0 * (btc.predict(X_test)>0.5)
# In[27]:
print(classification_report(y_true=y_test, y_pred=y_hat))
# In[28]:
f1_score(y_true=y_test, y_pred=y_hat)
# In[31]:
accuracy_score(y_true=y_test, y_pred=y_hat)
# In[32]:
confusion_matrix(y_true=y_test, y_pred=y_hat)
class BaggingTreeClassifier:
    def __init__(self, n=100):
```

```
        self.n = n
    def fit(self, X, y, random_state=42, criterion='entropy'):
        X_train_p = X[y==1]
        X_train_n = X[y==0]
        y_train_p = y[y==1]
        y_train_n = y[y==0]
        l = sum(y_train_p)
        clf_list = []
        np.random.seed(seed=random_state)
        for i in range(self.n):
            idx_n = np.random.choice(range(X_train_n.shape[0]), l)
            idx_p = np.random.choice(range(X_train_p.shape[0]), l)
            X_train_f = np.concatenate([X_train_n[idx_n, :],
X_train_p[idx_p]])
            y_train_f = np.concatenate([y_train_n[idx_n],
y_train_p[idx_p]])
            clf =
GradientBoostingClassifier(n_estimators=75,max_depth=4,learning_rate=0
.5,random_state=0)
            clf.fit(X_train_f, y_train_f)
            clf_list.append(clf)
            print(f'\rClassifier {i+1} completed.', end='')
        self.clfs = clf_list

    def predict(self, X):
        result_list = list(map(lambda clf: clf.predict_proba(X)[:,1],
self.clfs))
        score = reduce(lambda a, b: a+b, result_list) / len(self.clfs)
        return score

    def report(self, X, y):
        y_hat = self.predict(X)
        print(classification_report(y_test, 1.*(y_hat>0.5)))

# In[11]:

btc = BaggingTreeClassifier(200)
btc.fit(X_train, y_train)
y_hat = 1.0 * (btc.predict(X_test)>0.5)

# In[12]:

print(classification_report(y_true=y_test, y_pred=y_hat))

# In[13]:

f1_score(y_true=y_test, y_pred=y_hat)

# In[14]:

accuracy_score(y_true=y_test, y_pred=y_hat)

# In[15]:

confusion_matrix(y_true=y_test, y_pred=y_hat)
```