**Final Project Summary**

**Overview of Business Case / Question** →

* Our question is, given certain characteristics about an individual, can the machine predict if they would die due to COVID-19? Our group plans to filter the CDC data by first eliminating columns we do not want and then eliminating all rows that do not have complete information. We plan to keep the columns on age-group, sex, race, ethnicity, symptom, if they were hospitalized or not, and if they died or not. Then using this parsed data we trained the machine to make an accurate prediction as to if a certain person with COVID-19 will die. This algorithm will be especially important to COVID units in hospitals because it may help decide on courses of treatment and medical procedures pertaining to health problems prior to and because of the virus.

**Data Source and Description** →

* The data source is from the CDC and includes over 27 million covid cases from the United States dating to January of last year to this month along with specific details about each case. The variables include: case month/year, state of residence, state FIPS code, county of residence, county FIPS code, age group, sex, race, ethnicity, survived or not, exposure, symptoms, and more. But as I said, we have narrowed it down to a certain few.

**Data Cleaning (Benjamin)→** parsing data, SRS

The initial data was too large to import into colab and had lots of holes in the data. In order to filter out good information, parsed data was needed. Instead of importing directly into pandas - which would load all the rows into memory - it was instead read line by line. Each line was evaluated then placed in a new file. Only 1 row would be in memory at a time.

**EDA (Charles) →** bar plots (refer to notebook)

**Feature Engineering →** turning categorical data into integers, one hot encoding with a bunch of binary categories, dummy variables for null values in categorical data

* So, we used the one hot encoding method in a lot of the variables that the dataset included. Since a lot of the variables were binary such as death yes or no, hospitalized yes or no, male or female, etc. → we turned the values into 0s and 1s. Furthermore, after running the model we decided that for some of the categorical variables we needed to create dummy variables. Initially, what we did was that we took the race variable for example and labeled each race with a number. White would be 1, Black 2, so on so forth. Yet, this does not really establish how the differences in each race could affect the death rate. So, we decided to create dummy variables that essentially were new columns that had 0 or 1 values to represent if the person was black or not, or white or not, etc.

**Modeling →** Random Forest Classifier, grid search, threshold adjustment, decile chart

* We picked a Random Forest Classifier because we did not have any continuous variables and had many categorical variables. Initially, since we had a binary outcome, we were thinking of implementing a logistic regression model but since we do not have continuous variables it did not make sense. Once we did a train and test split of 70 and 30, we performed grid search on the random forest. We optimized the depth and the min\_sample\_leaf. When we ran the model on the test data, we saw that our sensitivity was horrendous and was around 20 percent. We then realized that our dataset was really imbalanced and that most of the data in the dataset did not actually result in death. Only about 2% of the total cases actually ended up in death, and so we adjusted the threshold to a number around there and our sensitivity shot up to 96 percent. Furthermore, our AUC also skyrocketed. Although it is a good thing, it is too good, and we tried to investigate why our model was doing so well. We tried a decile chart and saw that it wasn’t randomly guessing and again it told us that we were doing really well in predicting the patients with the highest risk for covid. Then we decided to make dummy variables for categorical data such as race, but that did not change the performance of the model as well.

**Results Overview →**

* To understand the result of our model better, let's pretend we have a hospital with 15000 covid patients, and in the future 300 patients will die (around 2% death rate) if treated the same as other patients.
* Given some simple demographic information about the patients, the model will predict around 16% or 2400 patients to have (increased risk of dying). Out of these 2400 patients, around 12% or 300 patients will die, and almost none of the people the model predicts as not die will die.
* The high rate of the model identifying a positive case(recall) comes with its drawback. Because by changing the threshold we can only make a tradeoff between precision and recall, which means the high recall comes with the drawback of low precision. In this case, this means out of the 2400 patients predicted, only 12% actually will die. However, this precision is acceptable in real life applications.
* So, after applying the model, we can allocate resources properly to those in need, thus decreasing the covid death rate. We can group patients with high risks in one area where resources are largely concentrated.
* Let’s go over the metrics we used to assess our model.
* The decile chart is used to evaluate the model’s ability to predict the outcome. An ideal decile chart would have the staircase effect, which is basically a decreasing height for the bins. The x bins in the chart represent the deciles, which is to say which interval of probability it lies on. For example, the x bin labeled as 1 represents the top 10 percent of predicted probability of dying. As the chart shows, the y-axis shows the percent of people that actually die in that decile. Our decile chart is ideal.
* The AUC ROC curve (area under receiver operating characteristic curve) score is a high 0.96. As a recap, the AUC ROC basically measures the ability of the model to distinguish between the positive and negative class or in our case whether the person dies or not.
* The confusion matrix helps us visualize the relationship between predictions and true/negative values. Specifically, it lists True False Positive Negatives. As you can see, this corresponds to the precision and recall we talked about before.

**Next Steps →** How can the model be improved, how will it be used

* There are a variety of improvements we could make. For example, some of the missing data were encoded as -1 instead of using dummy variables. However, turning missing data into a numerical value would introduce order that wasn’t supposed to be in the variable. For example, if we take into consideration the input category ‘sex’, the list of [‘missing’, Female, Male] would be encoded as [-1,0,1]. You can see how the order of -1,0,1 is not present in the previous list. However, because we tested the fact that even after using dummy variables, the model did not shift much as a result. Therefore, we kept some of the ‘missing values’ encoded as -1 to make the bar charts more readable.
* Later developments of the model can seek to include more variables to train with. This improvement has the ability to increase the precision greatly. In fact, a study that basically had the same purpose as ours trained their model with many more variables such as Diabetes, Strokes, etc.
* The model should also be verified with other datasets.
* To deploy the model, we would have to integrate this machine learning program with systems at hospitals. We would have to design the UI for the model, in which the user can input the demographics of the patient and get an immediate true or false result.