DSC 441 – Final Report
Shreyas Ravi
2021962

Business Problem:

This pandemic has affected every single person in our world in some way or another. Some people had it much worse than others, especially considering countries with higher population of people above 60 years of age and countries with large populations in general. This pandemic not only reminded us of our ability to handle healthcare for everyone within the country, but also was a rude awakening to how our actions could affect our own health and the health of others.

Then, when the COVID-19 vaccine came out in record time, there was obviously an air of relief since the vaccine (initially) promised to stop transmission of the virus. As the doses were administered to people throughout the country, videos began surfacing on the internet of people developing horrible side effects after taking the vaccine. They were obviously disturbing considering these videos were quite graphic, displaying blood clots, seizures and sometimes on rare occasion – Bell's Palsy. This began raising questions on the safety of the vaccine, especially considering that a lot of this information was not being presented by mainstream media, but floated around on the internet.

Often, doctors would dismiss vaccine hesitant people's concerns as "anti-vaxxers", which ended up causing more problems, since a lot of people did not know what to expect when taking these vaccines.

Something that would definitely help people be less hesitant is if they had a structured way of knowing, based on their existing conditions, medical history and more, whether or not they're prone to adverse effects from the vaccine or not. A predictive model that takes anomalies into consideration before predicting whether or not somebody would be prone to adverse effects, would be something helpful to reduce vaccine hesitancy, or even give people a sense of security before they take the vaccines. This is the problem I attempt to solve with my project.

The point of this project is to predict if a particular person coming in for the vaccine will end up getting an adverse reaction from the covid-19 vaccine. For this, we would need to classify the person as either getting or not getting an adverse effect because of the COVID vaccine.

In this case, recall (as opposed to precision) is very important since any person who might have the slightest chance of getting an adverse effect from this vaccine should not be receiving it, even if that means that some people might end up not receiving the vaccine. Therefore, we are looking for a high recall value for this case.

In this case, we can either work with F-1 scores or with recall. F-1 score is the weighted average between precision and recall.

Data Understanding:

I found this dataset on Kaggle, which is a popular website among programmers and data scientists alike, which allows people to not only download datasets for practice but also provides people with an opportunity to solve problems and submit code for those problems. Below is the link where I found the dataset –

- https://www.kaggle.com/ayushggarg/covid19-vaccine-adverse-reactions?select=2021VAERSDATA.csv

My initial idea was to work on a combination of classification and clustering due to the existence of categorical and numerical variables, which is what I had initially submitted in my proposal, however, with some feedback, I have decided to work only with classification. This would make the data cleaning process a lengthy one, considering this dataset contains a lot of missing values and explanatory variables in general.

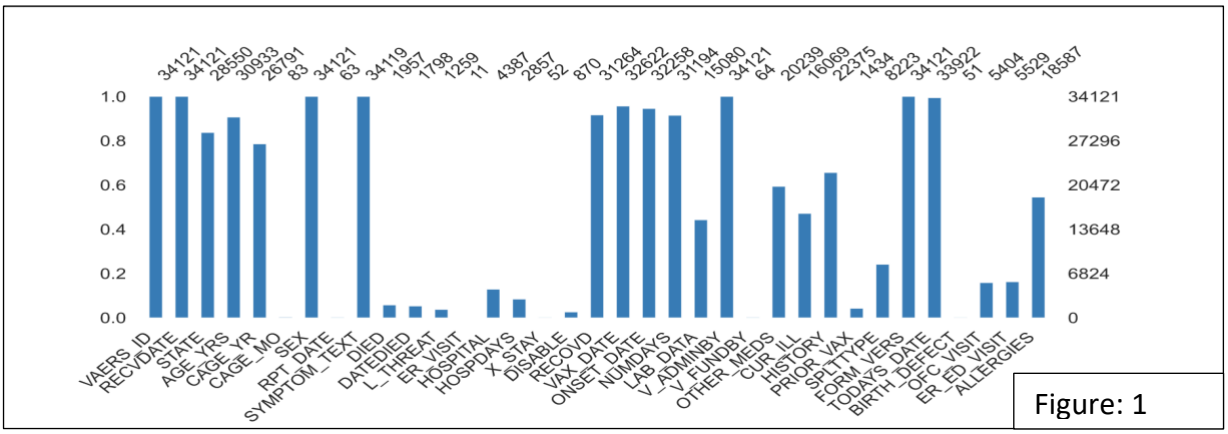

Figure: 1

The above shows the missing values in each column. Compared to my original predictions, this seems much higher. There are a total of 34,122 records, 35 columns of data and the approximate size of the file is 26.46MB. With 34 explanatory variables (potentially) and 1 dependent variables, it would be very important for me to clean this dataset properly and from the bar graph above, I can already see many variables/columns that will not make through the cleaning process due to such a large amount of missing values.
This would present a large array of issues, considering a lot of the columns have large amount of missing values, which would either mean I'd have to delete the entire column or find other methods of replacing the missing values. Considering some columns have a good chunk of information missing, it would be more prudent to delete the column rather than make the effort to fill the information in, especially considering the main variables I'd be looking at in this dataset would be the symptoms, death, age, sex, allergies, pre-existing conditions and a few more.

Data Preparation:

I already knew when I first started that this dataset would require a lot of cleaning. With my main column being a bunch of random text from patients and doctors with no discernable patterns I had to find a way to remove that column and create a way to account for the main issues highlighted within it, in a way that the model could understand and predict. Not only that, there were plenty of missing values in the dataset and almost too many irrelevant explanatory variables which I had to remove or convert into dummy variables so they could be properly recognized by the classifier. Considering I felt more comfortable cleaning the data on Python than on RStudio, I decided to choose to clean my data using pandas and numpy, which made the process quite easy considering most things could be done with single lines of code. I saw it more useful to run the classification models on RStudio since I had the help of previous lectures and Labs for this. Considering I have been coding on Python for a year and on RStudio only for 11 weeks, I saw it apt to finish it up on Python so the cleaning process would get over quicker rather than me having to dig through multiple StackOverflow pages to clean the data the way it needed to be cleaned.

To judge the data properly, I ran the following code on python to get an analysis of each variable:

```
from pandas_profiling import ProfileReport
import pandas as pd
df = pd.read_csv("2021VAERSDATA.csv", encoding = "ISO-8859-1")
profile = ProfileReport(df, title="VAERS profiling")
profile.to_file("VAERS Profile.html")
```

The above code provides an HTML page which shows an analysis of each variables and lets me know if the variables contain high correlation, cardinality or missing values. Furthermore, it also provides a list of common features within each column and a breakdown of uncommon features as well. The below screenshot provides an overview of the dataset including the number of variables, observations, missing cells, duplicate rows and a couple more metrics which help understand the data and where it requires cleaning.

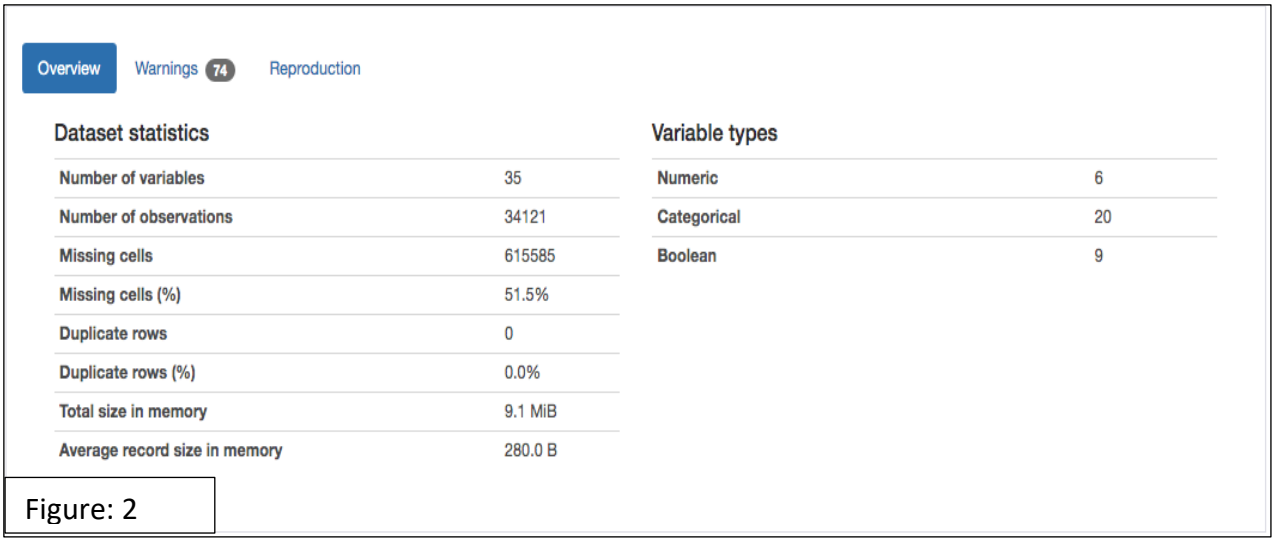

Figure: 2

Below is an example of what this code does for each variable. I have attached an example for AGE_YRS which is one of the independent variables in the dataset.

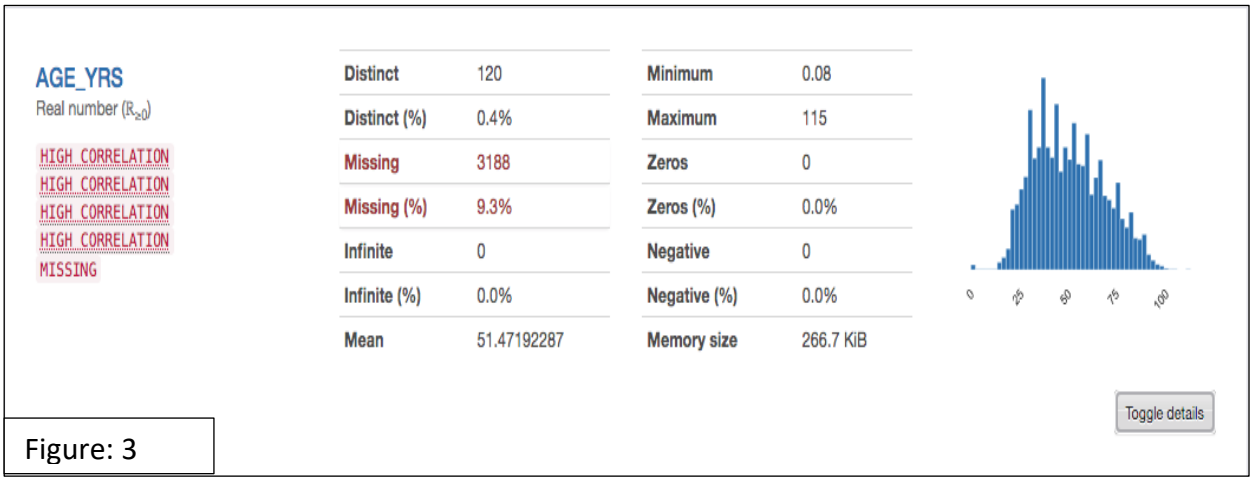

Figure: 3

Based on the correlation, cardinality and percent of missing values in a particular column, I either deleted the column or converted it into dummy variables if I thought it would be important. Below is a table included with the variables I deleted and kept.

Header	Type	Description of Contents	Keep/Remove	Reason for Removal
VAERS_ID	Num	VAERS ID Number	Remove	Index
EWCVDATE	Date	Date report received	Remove	Irrelevant date
STATE	Char	State	Keep	
CAGE_YR	Num	Age		
CAGE_MO	Num	Age in months	Remove	Derived variable/Same as CARE_YR
SEX	char	Gender/Sex	Keep	
RPT_DATE	Date	Date form completion	Remove	
SYMPTOM_TEXT	Char	Reported symptom text	Merged with col(DIED) to create DEP variable	
DIED	Char	Died (or not)	Merged with col(SYMPTOM_TEXT) to create DEP variable	
DATEDIED	Date	Date of Death	Remove	Irrelevant for identification of potential vaccinations

L_THREAT	Char	Life-Threatening Illness	Keep	
ER_VISIT	Char	Emergency Room or Doctor Visit	Keep	
HOSPITAL	Char	Hospitalized	Keep – Merge w/ HOSPDAYS	
HOSPDAYS	Num	Number of Days Hospitalized	Keep - (If hospitalized no days > 0)	
X_STAY	Char	Prolongation of Existing Hospitalization	Keep	
DISABLE	Char	Disability	Keep	
RECOVD	Char	Recovered	Remove	Unrequired for predicting if someone should or shouldn't take the vaxx
VAX_DATE	Date	Vaccination Date	Remove	Unimportant
ONSET_DATE	Date	Adverse Event Onset Date	Remove	Unrequired
NUMDAYS	Num	Number of Days (Onset date-Vax.Date)	Remove	Unrequired-Can't predict actual onset of effect before onset date
LAB_DATA	Char	Diagnostic Laboratory Data	Remove	Not Important
V_ADMINBY	Char	Type of Facility	Keep	
V_FUNDBY	Char	Type of Funds used to Purchase vaccines	Keep	
OTHER_MEDS	Char	Other Medications	Keep	
CUR_ILL	Char	Illnesses at time of vaccination	Keep	
HISTORY	Char	Chronic or long standing health conditions	Kept for creation of dummy variables	
PRIOR_VAX	Char	Prior Vaccination Event Information	Keep	
SPLTTYPE	Char	Manufacturer/Immunization Project Report Number	Remove	Vague data
FORM_VERS	Num	VAERS form version 1 or 2	Keep	
TODAYS_DATE	Date	Date Form Completed	Remove	Not Important

BIRTH_DEFECT	Char	Congenital anomaly or birth defect	Keep->Dummy Variables	
OFC_VISIT	Char	Doctor or other HC provider office or clinic	Keep-> Dummy Variables	
ER_ED_VISIT	Char	Emergency room/Department or urgent care	Keep -> Dummy Variables	
ALLERGIES	Char	Allergies to medications, food or products	Keep	

The cleaning of the dataset was done based on the above table. The idea was to convert the dataset into one that could be easy to work with. With so many columns that contain large amounts of text, different values and missing values, it was important to convert the columns into dummy variables to make it easier to enter into the model.

Based on the variables kept and deleted, I needed to find a way to identify adverse symptoms and feed the model with this information minus all the cluttered text. A simple way to do this was to create a list of "buzz" words and iterate through the column to find a match to the words in this list. I also went ahead and used Numpy to join the "Deaths" and "Symptoms" column into "AE" or Adverse Effects column in the form of dummy variables. In simple terms, if there is a word within the symptoms column that matches the bag of words, I add a "1" to the "AE" column. This means that this particular record/person has faced adverse effects due to the vaccine. This list of words was created after I scoured through the dataset to find the more severe side effects that were mentioned within the "SYMPTOMS_TEXT" column. I ignored the more common and expected side effects such as fevers, chills, shoulder soreness, nausea, dizziness. These are expected side effects of the vaccine that vaccine administrators warn people of before they are given their first and second dosages and therefore, I ignored them. The point of this project is to focus on the more life threatening side effects such as palpitations, tachycardia, anaphylaxis and blood clots, to name a few. After creating this column, I deleted the old ones ("SYMPTOMS_TEXT and "DEATHS") since I had combined both of them into one "AE" column. I also deleted the hospitalized column and converted the "HOSP_DAYS" to contain 0's for blank cells. Deleting the "HOSPITAL" (hospitalized (y/n)) column was important as it reduced the number of columns in the dataset while still retaining the information put forth by this column through the "HOSP_DAYS" column.

I then went ahead and created a new column "OM" (Other Meds). This column would also include the unknowns and NA values from "OTHER_MEDS" with the help of a "list of unknowns". Creating this list also involved scouring through the dataset to find the different ways an unknown or NA value was jotted into the dataset. This list of unknowns was used in multiple areas as a placeholder for blank cells. Then, to take comorbidities into consideration I created dummy variables for the most life threatening existing conditions such as Diabetes, Asthma, Hypertension and High Blood Pressure to name a few. The rest of the less life threatening comorbidities were included in a separate column "Other Comorbidities". Once I created this I was able to delete the "HISTORY" column as it was similar to the

"SYMPTOMS_TEXT" column – only contained a bunch of text rather than properly structured data. The dataset required this to be cleaned as the model would not be able to make inferences from string value. Therefore, the best way was to convert this categorical/qualitative information into numerical form so the model could understand the information. A way to combat the high cardinality between each variable in the column was to remove any cardinality so the model could understand the information, I converted the information into 1's and 0's. Columns such as "ALLERGIES", "BIRTH_DEFECT", "SEX", "PRIOR_VAX" and a few more, contained different inputs which would not be useful for the model. To standardize these values, I went ahead and made them all 1's and 0's.
Once all of the dataset was properly cleaned and easily understandable, all of these changes were dumped into a new CSV file called "2021VAERSDATA_CLEANED.csv". This file was ready to be used for the modeling processes.

Once the data was cleaned a new CSV was created it was easy to move ahead with the modeling process. I went ahead and loaded the data on RStudio and made a 0.8/0.2, train/test split and used N fold cross validation for my fitControl with 3 folds. This was a similar process done for both my models. The modelling processes I decided to use for this project are Naïve Bayes and Random Forests.

Data Modeling + Evaluation:

I chose to move ahead with the Naïve Bayes classifier as it was a classifier that was not only easy to implement but also worked well with categorical and numeric data. The initial thought process, before I had created the dummy variables, was to leave a lot of the columns in the final dataset with their original contents in. This however, didn't seem to work too well with the model, which required me to convert a lot of them into dummy variables. Considering Naïve Bayes is a classifier who's algorithm works with probability to predict outcomes, it was an important application for this project. The idea in my head was that Naïve Bayes would consider all the independent variables and their effects on the dependent variables ("Adverse Effects") and use probability to find out how likely a particular person is to have adverse effects if they take the vaccine. This was my thought process when I decided to choose the Naïve Bayes classifier. Since Naïve Bayes uses the Bayes' Theorem (a theorem that provides a description of the probability of an event occurring based on pre-existing information of the conditions associated with that event), which quite easily applied to the process I had highlighted in my mind, I decided to go ahead and use this as my first classifier.

As I mentioned earlier I imported the data into RStudio and used the "factor" method on the "Adverse Effects" column to change it from an "int". I then went ahead and split the data into a training set and testing set. Initially I was planning to do it at a 0.7/0.3 split, however, with some more research I found that most splits are done with 0.8+ (0.8/0.2 or 0.85/0.15) and therefore I decided to go for a 0.8/0.2 split. I ran the following code for the main model (the rest of the code is included in my RStudio file which has been uploaded as well) –

```
bayes <- train(AE ~ ., data = training, method = "nb", trControl=fitControl,
                tuneGrid=tuneControl)
```

```
Naive Bayes

27298 samples
   23 predictor
    2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (3 fold)
Summary of sample sizes: 18198, 18198, 18200
Resampling results:

  Accuracy   Kappa
  0.5965272  0.001135923

Tuning parameter 'fL' was held constant at a value of 1

Tuning parameter 'usekernel' was held constant at a value of
 TRUE
Tuning parameter 'adjust' was held constant at a value of 1
```

Figure: 4

```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 16269 10984
         1    23    22

               Accuracy : 0.5968
                 95% CI : (0.5909, 0.6026)
    No Information Rate : 0.5968
    P-Value [Acc > NIR] : 0.5075

                  Kappa : 7e-04

 Mcnemar's Test P-Value : <2e-16

              Precision : 0.5970
                 Recall : 0.9986
                     F1 : 0.7472
             Prevalence : 0.5968
         Detection Rate : 0.5960
   Detection Prevalence : 0.9984
      Balanced Accuracy : 0.5003

       'Positive' Class : 0
```

Figure: 5

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 4058 2739
         1   14   12

               Accuracy : 0.5965
                 95% CI : (0.5848, 0.6082)
    No Information Rate : 0.5968
    P-Value [Acc > NIR] : 0.5249

                  Kappa : 0.0011

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.996562
            Specificity : 0.004362
         Pos Pred Value : 0.597028
         Neg Pred Value : 0.461538
             Prevalence : 0.596805
         Detection Rate : 0.594753
   Detection Prevalence : 0.996189
      Balanced Accuracy : 0.500462

       'Positive' Class : 0
```

Figure:6

Above, Figure 4 displays the outcome of the Naïve Bayes model. As per usual we get the accuracy of the model along with the Kappa. Furthermore, we also have the two classes '0' and '1' for "AE" and this checks out since I had derived "AE" from "SYMPTOMS_TEXT" and "DEATHS" and used dummy variables to display the information.
In this case, we focus on the accuracy of the model, which comes out to 0.5965272, which is a relatively less desirable figure for the accuracy of the model. The accuracy of this model would also affect its predictive capabilities; in other words, this model would not be able to effectively predict values properly for the testing data.
Figure 5 and Figure 6 are the confusion matrices for the pred, training$AE and pred, testing$AE. Despite the confusion matrix for the latter being the more important one I wanted to display both to show any potential contrasts. As we saw before, we see the accuracy metric for predictive capability being relatively low at 0.5965, which is the same value we received from the summary of the Naïve Bayes model. This obviously means that the model is not effective enough to predict whether or not somebody is susceptible to adverse effects from the vaccine, with high accuracy. This would be an issue as we would need more true positive than false positive, therefore, more accuracy in the predictions than precision. This brings us to the recall score in the confusion matrix for the training set, at 0.9986, which is a very high score. I will be comparing my next model's Recall score to this Naïve Bayes model. Due to the lack of enough tuning, I was not able to get the recall score for the testing set, however, I predict it would be around the same. Below I have also included the variable importance for this model and dataset.

Figure: 7

	Importance
ER_ED_VISIT	100.0000
High.BP	92.5458
Obesity	91.4080
Coronary.Art.Dis	90.7219
Diabetes	78.0188
Asthma	76.2942
Hypertension	75.3246
AGE_YRS	73.9426
Other.Comoridities	64.5519
OM	56.0221
OFC_VISIT	33.5821
CUR_ILL	29.6044
HOSPDAYS	25.8429
L_THREAT	18.8971
ALLERGIES	11.8443
SEX	2.8347
DISABLE	2.7972
PRIOR_VAX	1.4108
FORM_VERS	1.3764
BIRTH_DEFECT	0.4868

Considering the accuracy rate of prediction for the Naïve Bayes model with my data was relatively low, I decided to move on to a commonly used model – Random Forest. Despite this being a relatively slower model, I found this better than a simple decision tree and decided to use it instead. Since Random Forest builds each decision tree with a different set of variables to reduce overfitting, I thought this would be apt since overfitting wouldn't be a desirable trait in

this case (or at all) and if it is something I can avoid, then all the better. Furthermore, despite doing my best to clean the entire dataset, there are areas where I simply filled in blank spaces with an assigned value. There is a possibility that some outliers might influence patterns and since Random Forest is said to be rather robust to noise and overfitting and also an excellent performance model, this works best for me. Like Naïve Bayes, Random Forests can work with categorical and numerical variables, which is also another reason I decided to pick it, since a lot of my data handles categorical variables in the form of dummy variables.
Considering this is a rather slow model to build, it took almost too much time to run on RStudio despite me loading all the required libraries and making sure my machine has a proper capacity to support a quick processing of the model. After multiple tries of running the model with no success, I decided to run the same model on Python. Since I am already relatively familiar with Pandas and Scikit Learn, I decided to apply whatever knowledge I had into creating the Random Forest model.
The main metric I wanted to evaluate and compare to the Naïve Bayes model was the accuracy. I was hoping for a better accuracy with the Random Forest model. After importing the "RandomForestClassifier" from "sklearn.ensemble", the "make_classification" from "sklearn.datasets" and "train_test_split" from "sklearn.model_selection", I split the data into a 0.8/0.2, train/test split, which was the exact same split I used for the Naïve Bayes model as well. To make sure both models are relatively comparable, I made sure that the training and testing splits had the same amounts of records.
Once I had imported the required libraries/packages for finding the accuracy of the model, I found the following -

```
In [12]: accuracy_score(train['AE'],clf.predict(train[X_var]))   #accuracy of training
data
Out[12]: 0.7952080890973037
```

Figure: 8

This accuracy value looks very promising compared to the Naïve Bayes model's accuracy. Below I have attached the accuracy of the model with the testing data.

```
In [13]: accuracy_score(test['AE'],clf.predict(test[X_var]))     # accuracy for testing
data (overfitting)
Out[13]: 0.5925274725274725
```

Figure: 9

This was the most disappointing moment throughout my entire project. I was very much hoping for a similar accuracy value for the testing data with the Random Forest model, however, as we can clearly see, the accuracy is much lower for the testing data than it is for the training data which only points to one thing. The model overfitted for the training data, which is why the accuracy of its predictions for the testing data fell. Below I have also attached the recall scores for the training and testing data.

```
In [15]: recall_score(train['AE'], clf.predict(train[X_var]))
Out[15]: 0.6034937676280593
```

Figure: 10

```
In [16]: recall_score(test['AE'], clf.predict(test[X_var]))
Out[16]: 0.3600867678958785
```

Figure: 11

Both Figure 11 displays the recall score of the testing data, which is too low considering a high recall score is more important in this case than a high precision score. This score is obviously lower than the recall score we saw earlier with the Naïve Bayes model.
This model has overfitted to the training set and therefore cannot be relied on for proper predictions. Despite Random Forest being a model that works well with minimal tuning, to fix this overfitting the model requires a certain amount of tuning. This was the reason why I did not work on any tuning earlier.

Conclusion + Discussion:

After spending an entire quarter studying classification methods, I was unfortunately unable to receive the results I wanted from the classification models I worked with in this project. The main idea was to properly predict whether or not a particular person would be highly likely to receive "Adverse Effects" as a result of the COVID-19 vaccine and my thought process revolved around cleaning the data in a way that would allow models to easily recognize the patterns and provide a good output. I mentioned my reasons for choosing Naïve Bayes and Random Forests and the main reason was the ability of both models to work with categorical variables. This is an issue I obviously had a lot of interest in as COVID has been around enough to cause a lot of problems for every single one of us. I chose this dataset because I wanted to model the patterns that came out of it rather than making guesses about the percentages of adverse events caused by the COVID vaccine. After all, the numbers would speak for themselves. However, despite all the desire to work with such a dataset, I think I might have picked a rather untidy one to work with especially considering this is one of my first machine learning/classification projects.

This dataset was untidy with too many missing values, high cardinality and correlation among certain variables and was just overall quite a pain to clean. It took me a while to arrive at the idea of creating simple dummy variables for the adverse effects and that in itself was not a time saving task. I had to scour through the entire 34000+ records of the dataset to find the more life threatening adverse effects and add them to a list of words which I would then use in my cleaning process (and to create the dummy columns). Furthermore, dealing with outliers and missing values was definitely a process that had me worried, considering there were a fair bit of missing values that I had substituted with the median of the column's data. There were some outliers I had to remove as well, however, this process was not as tedious as making the dataset a clean and interpretable one for any person to understand and for the model to understand as well.

An important change I'd make, should I work with such data again would be to tune hyperparameters. I felt like this was the one process I did not follow from the lectures and Labs and it may or may not have cost me a proper outcome from my models. Furthermore, I would also do more external research on examples of existing models for the classification methods I choose, so I can properly understand what kind of outcomes are good outcomes and how I can

work with my data and its requirements to choose the best model. I feel by only referring to the PowerPoints and not doing any external research on Naïve Bayes and Random Forests, I might have missed out on some key information which would have potentially caused me to even change the classification method I used. This is of course, by no means saying that the PowerPoints did not have enough information, as they certainly did; I just think a little external research on my own would have helped me with a better outcome.

Overall, I am quite glad I attempted this project regardless of the outcome. It gave a very good insight into what to do when working with predictive models and it definitely gave a good amount of data cleaning practice. In hindsight, it might have just been a good idea to choose a slightly less clean dataset so I could apply my knowledge towards cleaning it and learn in the process.