Mining Novel Corona Virus 2019 Dataset

Elias Bestard Lorigados

Introduction

Corona Virus 2019 (Covid-19) has changed our lives this past year. Many studies and experiments have been focused on this topic recently, and, in most of them, there are Data Mining techniques present. Data mining applied to Covid-19 data is very useful and necessary to discover patterns of the virus spread, differences in symptoms and understanding what populations are more at risk.

In this study, the Novel Corona Virus 2019 Dataset [1] was used to obtain interesting patterns of the virus distribution among the different populations, which would help with ongoing and future studies related to this virus. The dataset used in this project includes data from 146 different countries and a total of 2676311 people, where each of them has tested positive for Covid-19. This real-world dataset was very comprehensive, up to date, and from reliable sources. However, it was also disorganized since it was rough data without any analysis, which presented an interesting challenge for this study.

This report aims to conduct an exploratory analysis of the latest version of Novel Corona Virus 2019 Dataset [2] corresponding to 2020-12-15, mine interesting patters (association rules), and learn to predict the outcome of positive tested Covid-19 patients. We will apply the main stages of Data Mining over the dataset to gain valuable information from the rough data, using preprocessing, clustering, classification, and association rules techniques to get results. Then we will analyze the obtained results and propose possible future approaches to continue our work.

The main objectives of this project are to mine association rules of interest and learn to predict the outcome of patients that have tested positive for Covid-19. Both objectives can support future research and help to prevent the worst outcome possible for this virus. This dataset contains interesting problems to be solved before we can proceed to extract association rules. First, we will organize and structure the data; then we will find missing values and fill them with valuable and correct information. Finally, we will solve the problem of obtaining the association rules and patterns for Covid-19 through data mining techniques.

Organization of the report

This report will contain the following structure:

- Introduction including objectives of the study and problems to be solved
- Related work with previous solutions and explanation of state of the art on this dataset about DM
- Our approaches explaining in detail each stage taken
- Results including a description of the experiments and discussion of the results
- Conclusions that summarize the major findings of this study as well as the limitations
- References
- Appendices

Related work

The study of Covid-19 has been a priority nowadays, and the application of Data mining techniques has been used on a great variety of Covid-19 datasets. However, most of the previous work using this project's dataset focused on Data analytics and visualization. Studies such as the one published online by Ye, A., 2020 [3] used compression of the original dataset and did not generate any other result but visualizing and analyzing the statistics on it. Therefore, the solutions found in previous projects were focused on data visualization and analyzing the data in statistically significant ways, disregarding data mining. Their objective was not to find and explore interesting patterns but to provide an in-depth statistical analysis of the dataset. However, in this current project, we aim to use data mining to examine this dataset further and offer insight into new patterns of this virus that previous studies did not consider.

Approach

The current work focused on using the principal Data Mining techniques to extract interesting patterns in the data. Firstly, we used tools on python to visualize the data and choose which columns will be better to keep in the next stages, as part of the exploratory analysis of the dataset. After removing unnecessary data, we proceeded to standardize the current dataset and set each of the columns in the accurate type that made it feasible to apply other preprocessing techniques to handle missing values. Then, after organizing and adding valuable information, we mined association rules from the dataset (Figure 1).

The next sections will focus on explaining each of these stages in detail.

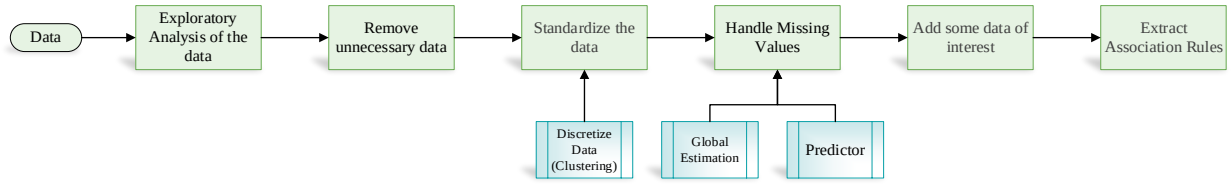


Figure 1. Flow chart of the program. This chart describes how the data is processed through the program showing all the main stages before extracting association rules and patterns.

Description

The first stage, Exploratory analysis of the data, focused on using libraries such as seaborn, and pandas, to visualize the content of the data and see how it is stored.

The original dataset contained 32 different columns and 2 676 311 entries, as shown in Appendix 1, where each of the entries represented a patient that tested positive for Covid-19. A heatmap chart of the missing values of the original dataset was shown in Figure 2, and Figure 3 showed the percentage with respect to the total amount of entries of actual values on each attribute. These two charts obtained from phyton explain how many columns and existing rows we should remove to achieve good performance on our objectives.

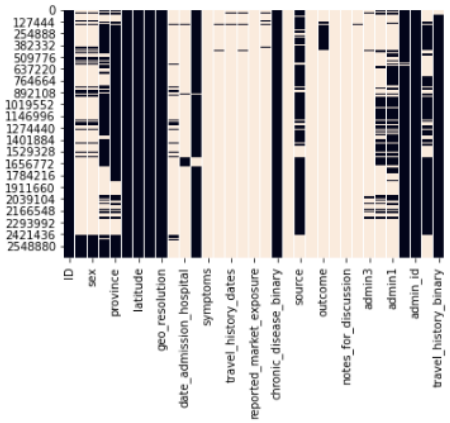


Figure 2. Heatmap of missing values.

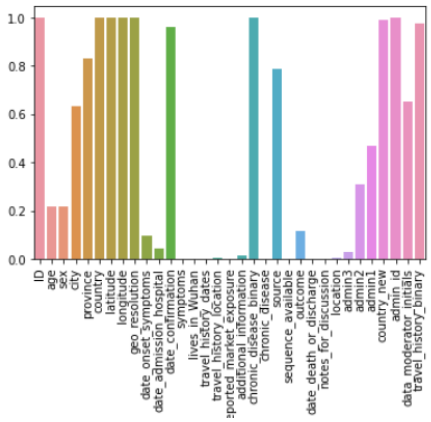


Figure 3. Percentage of missing values per attribute.

In the next stage of the process, we proceed to remove unnecessary and duplicated data. Columns like 'data_moderator', 'admin1', admin2', 'admin3', 'notes_for_discussion', 'admin_id', and 'additional_information' do not contain valuable information but administrative information; thus, we removed them. Other columns had geographical information of the patients, and we kept only one, 'country', as it is the most representative and useful one for our project. Finally, other columns that were considered empty were removed, as shown in Figure 4. We observed that we still must deal with missing values and apply different techniques to fill them.

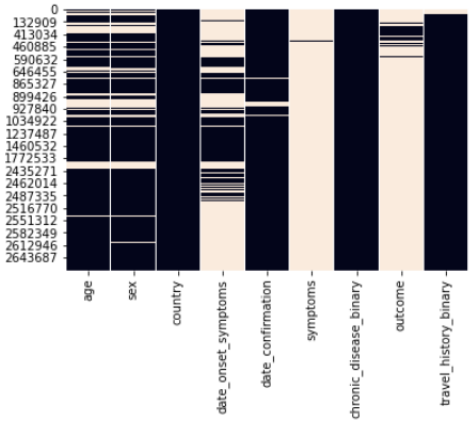


Figure 4. Heatmap of missing values after data selection.

After completing these first steps, we started to work on each column independently to standardize the data. The first attribute to work on was the 'outcome' attribute. This column originally contained 35 different categories, and some of them were repetitive, as we can see in Figure 5.

We can clearly observe that several categories such as 'critical condition, intubated as of 14.02.2020', 'severe', 'Critical condition', 'severe illness' and others represent the same outcome, 'critical'. Then, we discretized this attribute into six categories: 'dead', 'critical', 'discharged', 'recovered', 'stable', and 'hospitalized', with the numbers from 0 to 5 respectively as shown in Figure 6.

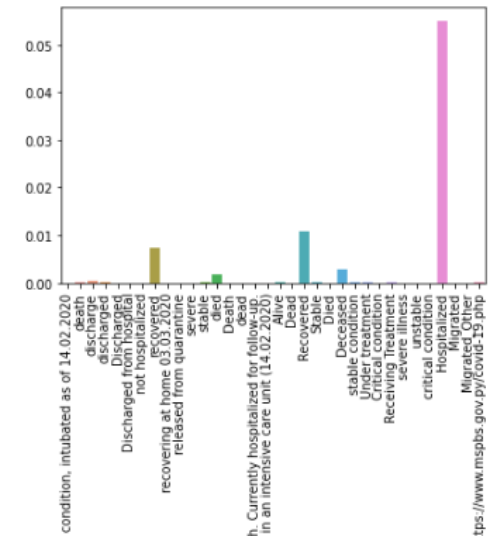


Figure 5. Outcome-column before data gets discretized.

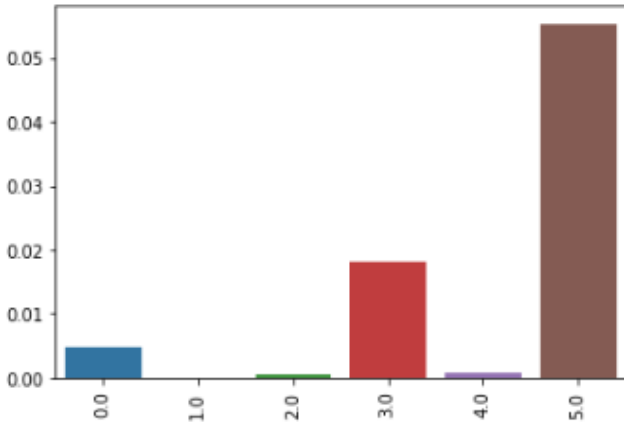


Figure 6. Outcome-column after discretizing the data.

Similarly, the 'symptoms' attributes also had repetition of attributes, but on a larger scale. This column contains 449 unique values, and each of them had more individual symptoms, where many had the same meaning expressed in different ways. In this case, we proceeded to group them using clustering. We used the DB-Scan algorithm with the Levenshtein metric to group each symptom with the ones that have similar lexicography. We reduced the number of symptoms, but still this work can be further improved in other research projects by applying deep techniques of text analysis and word embedding to obtain better results.

By standardizing the column 'age', we converted this column into a numerical column and plotted a box chart to analyze the outliers above the 75% quartile, as shown in Figure 7. After dealing with the outliers, we analyzed the descriptive statistics (Appendix 2) on this attribute and filled the missing values using a normal distribution with the actual mean and standard deviation of the data. Therefore, we changed the missing values using a global estimation. Instead of using the same value (mean), we placed different ones keeping the same distribution of the data.

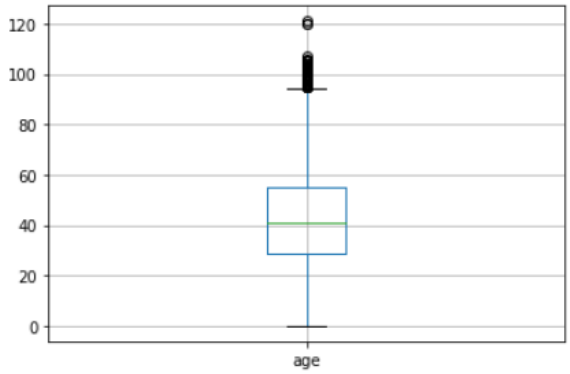


Figure 7. Age boxplot before preprocessing.

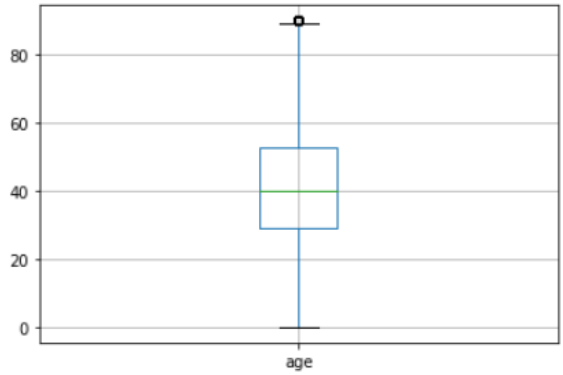


Figure 8. Age boxplot after preprocessing.

The ‘date_type’ attributes were converted into an integer, which is the difference in days between the actual value and the first date that appears in the data (2019-12-01). Then, looking for a more representative value of the dates, we added a new column that contains the values of the difference in days of the date in which the symptoms appeared and the day in which it was confirmed that the patient had Covid-19. With this new column, we aimed to have a value that can agglomerate more patients and see if it would appear on any association rules at the end of the experiments. With this value-added, we analyzed the descriptive statistics and realized that the mean difference in days between these two dates was 6 and used this to fill the values on each of the columns ‘date_onset_symptoms’ or ‘date_confirmation’ when one of this value did not appear. Lastly, the rest of the columns were encoded, keeping the values after converting them into their most accurate type.

In order to finish handling missing values, we implemented a predictor using the original values of the dataset, not the ones already added. This approach allowed us to learn to predict the outcome of a patient that tested positive for Covid-19 over the 6 categories that we already had the outcome set. For our experiments, we used 3 main algorithms, K-NN, Decision Tree, and MLP, and kept the best results to fill the ‘outcome’ attribute. We will show the results in the next section of the document.

After completing all the preprocessing stages, we had a better-structured dataset that contains fewer missing values that we can then proceed to extract association rules using FP-growth.

Experimental results

In this section, we will show the results of the implemented predictors as well as the association rules that we have obtained.

For the implementation of the predictors, we used mainly three algorithms, KNN, MLP, and Decision Tree, from sklearn using python. We also used the entries of the dataset that already contained an outcome for the supervised learning stage. In total, we worked with 40 044 entries and 7 different rows, as shown in Appendix 3, and split it into a test set and a training set. For measurement of the error, we compare those algorithms with accuracy, precision, recall, and f1-score.

Firstly, before going into deep preprocessing stages, we obtained the accuracy of these classifiers up to 68%. After standardizing the data, selecting the attributes and adding the new column, we were able to bring the accuracy factor up to 80% using KNN and MLP. Even though choosing K for KNN could be problematic, we plotted the accuracy value from 0 to 200 as shown in Figure 9. We found that K=26 had the best results using accuracy as the measurement metric, as shown in Appendix 5. For the multilayer perceptron, we set the maximum number of iterations as 300 and used the default set of the number of layers that sklearn had, 100, while the decision tree classifier was set using entropy as the criterion of the split for this experiment.

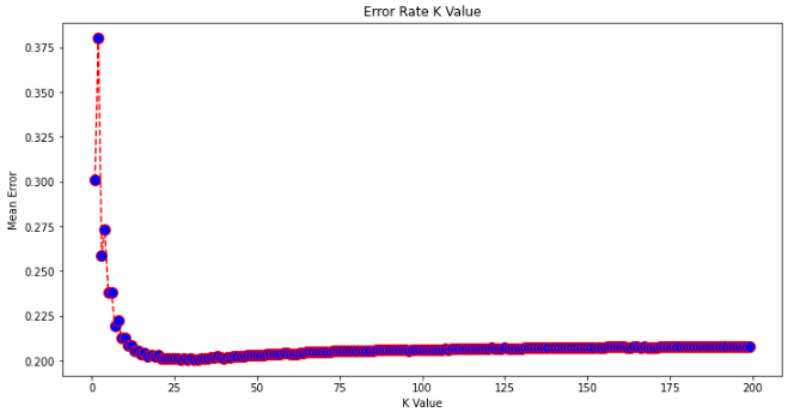


Figure 9. Mean error of KNN with different K values.

We used the 40 044 entries that originally contained an outcome to split into a train and test set. After training these algorithms on the training set, we found that KNN and MLP performance were very similar, while the Decision Tree obtained a better precision with 0.81 compared to 0.80 of the first two on the test set. KNN and MLP algorithms obtained the highest scores with 1 of recall, 0.8 accuracies, and 0.89 f1-score, as shown in Figure 10. On the other hand, when we compared the results using 10 folds cross-validation, we found that MLP had the best overall performance, as shown in Figure 11.

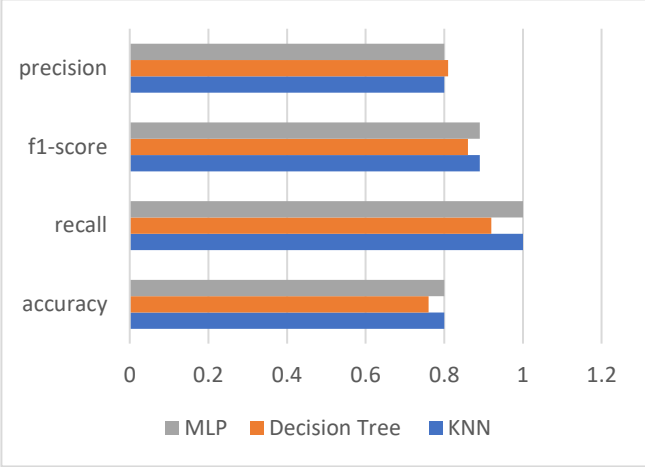


Figure 10. Measurements of the error of each predictor, KNN, MLP, and Decision Tree.

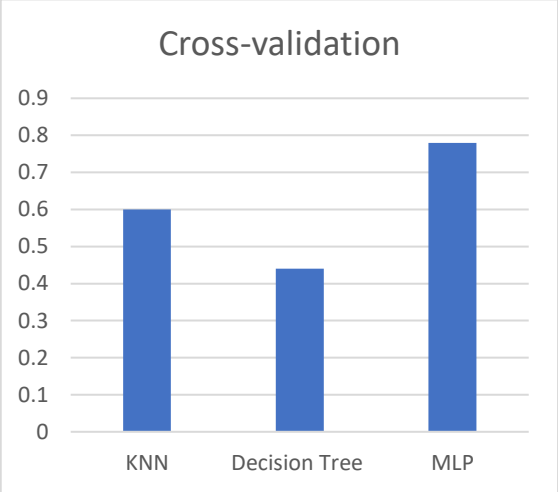


Figure 11. 10 folds cross-validation.

As a result of mining association rules using FP-growth with minimum support of 0.3 over the actual dataset, we found 27 item sets, and a total of 65 association rules, where 10 of those 65 were misleading having a correlation less than or equal to 1, as shown in Figure 10, and we found misleading association rules shown in **Appendix *6*. FP-growth output with lift lower or equal to 1 (misleading).**Appendix 6**.** We can observe all the 55 association rules that have lift greater than 1 in Appendix **7**, and we will briefly discuss some of the most interesting ones in the following paragraphs.

Firstly, it is good to point out that the column we added while preprocessing appeared with high frequency on the association rules, meaning that these values had a good correlation with the outcome and other attributes. This process allowed us to analyze more association rules with lift greater than 1 that contained the outcome as a consequence as well as different association rules with high confidence.

One of the first association rules found was that if a person did not travel recently, did not have a chronic disease, and was confirmed positive for Covid-19 within 6 days of symptom appearance, it implies that the outcome of the patient will be recovered, with a confidence of 0.56 and a lift of 1.43. Another one is that a patient with no chronic disease with 6 days between the appearance of symptoms and the Covid-19 confirmation day also implies recovered as consequence with a confidence of 0.43. Another association rule found in the study was that if a patient has an outcome of 'recovered' and tested positive for Covid-19 with 6 days from the appearance of the symptoms, it implies that the patient does not have a chronic disease with a confidence equal to 1. All these results were obtained from the association rules mining, and to analyze and see all of them in detail refer to Appendix **7.**

Conclusions

While Covid-19 continues affecting our daily lives, studies are narrowing the gap of knowledge between this virus and the outcomes on society. We proposed as objectives of this project to mine interesting patterns on the Novel Coronavirus 2019 Dataset and learn to predict the outcomes on patients who have been tested positive for Covid-19.

Firstly, we studied the dataset, and after organizing and standardizing it, we were able to get a well-structured version of this dataset; thus, stressing the importance of the preprocessing stages that allowed us to learn how to predict the outcome of patients with reasonable accuracy of 80%. We compared three multiclass classifiers and chose the highest accuracy one to fill the missing values on the outcome column. Finally, with the best version of the dataset, we obtained approximately 65 association rules where most of them were correlated, while 10 were misleading.

The findings of this study showed us the relationship of the outcome, the day of appearance of symptoms, the day of confirmation and whether a patient has chronic diseases over the association rules. These results stress the relevance of exploring patterns and obtaining good predictors to help improve health strategies on how to act depending on the patients and the outcomes possible.

Limitations and possible extensions

A limitation of this study is the large number of missing values. Those values were filled in our study, which could introduce biasing factors that could affect the results. However, we used proper techniques and high accuracy predictors to complete the values so there was no missing data. A better-structured original version of this dataset could help improve the results. Another limitation is that since this is real-world data, many other confounding factors are not included and could be affecting the Covid-19 patterns that we are reporting in this project. For example, the dataset includes if the patient has chronic diseases but does not mention which diseases were considered in this category. This could mean that other health risk factors that are not considered chronic diseases such as overweight or obesity, especially if the patient has not been obese for a long time, are not included in the dataset. Those confounding factors could still affect the outcome of the patient but are not considered in our predictions since the dataset was missing other health history factors. Therefore, a possible extension to this study is to include an up-to-date version of the dataset with the patient's health history of risk factors.

References

[1] "Kaggle Novel Corona Virus 2019 Dataset," 21 12 2020. [Online]. Available: https://www.kaggle.com/sudalairajkumar/novel-corona-virus-2019-dataset?select=COVID19_open_line_list.csv.

[2] O. C.-1. D. W. Group, "GitHub," 2020. [Online]. Available: https://github.com/beoutbreakprepared/nCoV2019/blob/master/latest_data/latestdata.tar.gz. [Accessed 15 12 2020].

[3] A. Ye, "towards data science," Medium, 10 March 2020. [Online]. Available: https://towardsdatascience.com/see-the-coronavirus-for-yourself-88ce06b88f5e. [Accessed 21 12 2020].

Appendices

Our program has been implemented using google Collab on a Jupiter notebook. To run it, we need to load it on google Collab, or any other Jupiter notebook editor, and change the dataset location on the first section while it is reading it. After this stage, we can run each section and see the results step by step.

```
RangeIndex: 2676311 entries, 0 to 2676310
Data columns (total 33 columns):
 #   Column                    Dtype
---  ------                    -----
 0   ID                        object
 1   age                       object
 2   sex                       object
 3   city                      object
 4   province                  object
 5   country                   object
 6   latitude                  float64
 7   longitude                 float64
 8   geo_resolution            object
 9   date_onset_symptoms       object
 10  date_admission_hospital   object
 11  date_confirmation         object
 12  symptoms                  object
 13  lives_in_Wuhan            object
 14  travel_history_dates      object
 15  travel_history_location   object
 16  reported_market_exposure  object
 17  additional_information    object
 18  chronic_disease_binary    bool
 19  chronic_disease           object
 20  source                    object
 21  sequence_available        object
 22  outcome                   object
 23  date_death_or_discharge   object
 24  notes_for_discussion      object
 25  location                  object
 26  admin3                    object
 27  admin2                    object
 28  admin1                    object
 29  country_new               object
 30  admin_id                  float64
 31  data_moderator_initials   object
 32  travel_history_binary     object
dtypes: bool(1), float64(3), object(29)
memory usage: 655.9+ MB
```

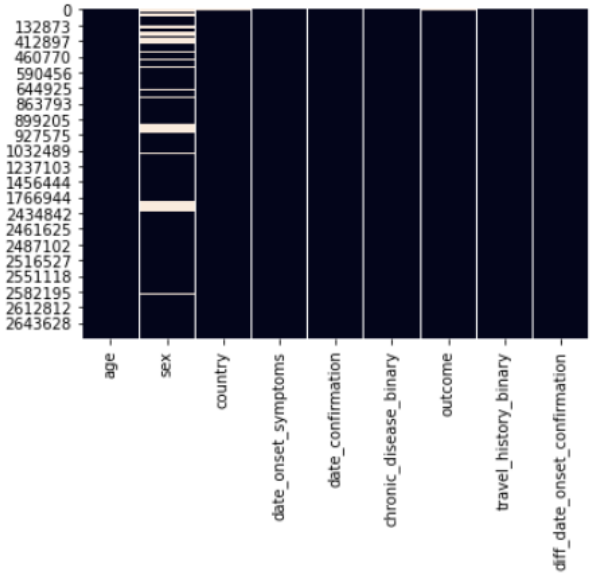
Appendix 1. Description of each type of a column of the original dataset before preprocessing.

```
count    274762
mean         42
std          18
min           0
25%          29
50%          41
75%          55
max         121
Name: age, dtype: int64
```

Appendix 2. Statistic of 'age'.

```
Int64Index: 40044 entries, 0 to 673607
Data columns (total 7 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   age                           40044 non-null  float64
 1   country                       40044 non-null  float64
 2   date_onset_symptoms           40044 non-null  float64
 3   date_confirmation             40044 non-null  float64
 4   chronic_disease_binary        40044 non-null  bool
 5   travel_history_binary         40044 non-null  bool
 6   diff_date_onset_confirmation  40044 non-null  float64
dtypes: bool(2), float64(5)
memory usage: 1.9 MB
```

Appendix 3. Compacted dataset to train the predictors of 'outcome'.



Appendix 4. Dataset after completing all the preprocessing stages.

```
[[  128     0     5   266    12   607]
 [    0     0     1     0     1     2]
 [    6     0   102     0    19     0]
 [   51     0     3  1411     4  2159]
 [    4     0     1    12   117    19]
 [    8     0     0    38     0 11046]]
              precision    recall  f1-score   support

         0.0       0.65      0.13      0.21      1018
         1.0       0.00      0.00      0.00         4
         2.0       0.91      0.80      0.85       127
         3.0       0.82      0.39      0.53      3628
         4.0       0.76      0.76      0.76       153
         5.0       0.80      1.00      0.89     11092

    accuracy                           0.80     16022
   macro avg       0.66      0.51      0.54     16022
weighted avg       0.79      0.80      0.76     16022
```

Appendix 5. Classification report of KNN with K=26. The confusion matrix is shown at the top and the scores bellow.

antecedents	consequents	support	confidence	lift
('6.0diff_date')	('False_chronic_disease')	0.600194	0.9995623	0.999931
('False_chronic_disease')	('6.0diff_date')	0.600194	0.6004155	0.999931
('6.0diff_date')	('False_travel')	0.565943	0.9425195	0.993968
('False_travel')	('6.0diff_date')	0.565943	0.5968354	0.993968
('6.0diff_date', 'False_chronic_disease')	('False_travel')	0.565873	0.9428162	0.994281
('False_travel', 'False_chronic_disease')	('6.0diff_date')	0.565873	0.596812	0.993929
('6.0diff_date')	('False_travel', 'False_chronic_disease')	0.565873	0.9424035	0.993929
('False_travel')	('6.0diff_date', 'False_chronic_disease')	0.565873	0.5967619	0.994281
('1.0_sex')	('False_chronic_disease')	0.449904	0.9994603	0.999829
('False_chronic_disease')	('1.0_sex')	0.449904	0.4500696	0.999829

Appendix 6. FP-growth output with lift lower or equal to 1 (misleading).

antecedents	consequents	support	confidence	lift
('6.0diff_date', 'False_travel')	('False_chronic_disease')	0.56587295	0.99987685	1.000245
('False_chronic_disease')	('6.0diff_date', 'False_travel')	0.56587295	0.56608148	1.000245
('1.0_sex')	('False_travel')	0.43165988	0.95893144	1.011276
('False_travel')	('1.0_sex')	0.43165988	0.4552226	1.011276
('1.0_sex', 'False_chronic_disease')	('False_travel')	0.43160611	0.95932973	1.011696
('1.0_sex', 'False_travel')	('False_chronic_disease')	0.43160611	0.99987545	1.000244
('False_travel', 'False_chronic_disease')	('1.0_sex')	0.43160611	0.45520413	1.011235
('1.0_sex')	('False_travel', 'False_chronic_disease')	0.43160611	0.958812	1.011235
('False_chronic_disease')	('1.0_sex', 'False_travel')	0.43160611	0.43176517	1.000244
('False_travel')	('1.0_sex', 'False_chronic_disease')	0.43160611	0.4551659	1.011696
('0.0_sex')	('False_chronic_disease')	0.3903611	0.99970932	1.000078
('0.0_sex')	('False_travel')	0.37644815	0.96407845	1.016704
('False_travel', '0.0_sex')	('False_chronic_disease')	0.37642823	0.9999471	1.000316
('False_chronic_disease', '0.0_sex')	('False_travel')	0.37642823	0.96430776	1.016946
('0.0_sex')	('False_travel', 'False_chronic_disease')	0.37642823	0.96402746	1.016736
('3.0_outcome')	('False_chronic_disease')	0.37018164	0.99989781	1.000266
('6.0diff_date')	('3.0_outcome')	0.33276981	0.55419405	1.496934
('3.0_outcome')	('6.0diff_date')	0.33276981	0.89884468	1.496934
('3.0_outcome')	('False_travel')	0.35930537	0.9705199	1.023497
('6.0diff_date', 'False_chronic_disease')	('3.0_outcome')	0.33275985	0.55442017	1.497545
('False_chronic_disease', '3.0_outcome')	('6.0diff_date')	0.33275985	0.89890965	1.497042
('6.0diff_date', '3.0_outcome')	('False_chronic_disease')	0.33275985	0.99997008	1.000339
('6.0diff_date')	('False_chronic_disease', '3.0_outcome')	0.33275985	0.55417747	1.497042
('3.0_outcome')	('6.0diff_date', 'False_chronic_disease')	0.33275985	0.89881778	1.497545
('6.0diff_date', 'False_travel')	('3.0_outcome')	0.33046791	0.58392473	1.577239
('6.0diff_date', '3.0_outcome')	('False_travel')	0.33046791	0.99308261	1.047291
('False_travel', '3.0_outcome')	('6.0diff_date')	0.33046791	0.91974108	1.531735
('6.0diff_date')	('False_travel', '3.0_outcome')	0.33046791	0.55036048	1.531735
('3.0_outcome')	('6.0diff_date', 'False_travel')	0.33046791	0.89262702	1.577239
('6.0diff_date', 'False_travel', 'False_chronic_disease')	('3.0_outcome')	0.33046592	0.58399313	1.577424
('6.0diff_date', 'False_chronic_disease', '3.0_outcome')	('False_travel')	0.33046592	0.99310634	1.047316
('False_travel', 'False_chronic_disease', '3.0_outcome')	('6.0diff_date')	0.33046592	0.91974573	1.531742
('6.0diff_date', 'False_travel', '3.0_outcome')	('False_chronic_disease')	0.33046592	0.99999397	1.000362
('6.0diff_date', 'False_chronic_disease')	('False_travel', '3.0_outcome')	0.33046592	0.55059818	1.532396
('False_chronic_disease', '3.0_outcome')	('6.0diff_date', 'False_travel')	0.33046592	0.89271287	1.577391
('6.0diff_date', 'False_travel')	('False_chronic_disease', '3.0_outcome')	0.33046592	0.58392121	1.577391
('6.0diff_date', '3.0_outcome')	('False_travel', 'False_chronic_disease')	0.33046592	0.99307662	1.047373
('False_travel', '3.0_outcome')	('6.0diff_date', 'False_chronic_disease')	0.33046592	0.91973554	1.532396
('6.0diff_date')	('False_travel', 'False_chronic_disease', '3.0_outcome')	0.33046592	0.55035716	1.531742
('3.0_outcome')	('6.0diff_date', 'False_travel', 'False_chronic_disease')	0.33046592	0.89262164	1.577424
('False_chronic_disease', '3.0_outcome')	('False_travel')	0.35930139	0.97060833	1.02359
('False_travel', '3.0_outcome')	('False_chronic_disease')	0.35930139	0.99998892	1.000357
('3.0_outcome')	('False_travel', 'False_chronic_disease')	0.35930139	0.97050914	1.023572
('False_chronic_disease')	('5.0_outcome')	0.4150866	0.41523957	1.000349
('5.0_outcome')	('False_chronic_disease')	0.4150866	0.99998081	1.000349
('False_travel')	('5.0_outcome')	0.40643058	0.42861613	1.032575
('5.0_outcome')	('False_travel')	0.40643058	0.97912769	1.032575
('False_travel', 'False_chronic_disease')	('5.0_outcome')	0.4064266	0.42864793	1.032651
('False_travel', '5.0_outcome')	('False_chronic_disease')	0.4064266	0.9999902	1.000359
('False_chronic_disease', '5.0_outcome')	('False_travel')	0.4064266	0.97913688	1.032584
('False_travel')	('False_chronic_disease', '5.0_outcome')	0.4064266	0.42861193	1.032584
('False_chronic_disease')	('False_travel', '5.0_outcome')	0.4064266	0.40657638	1.000359
('5.0_outcome')	('False_travel', 'False_chronic_disease')	0.4064266	0.9791181	1.032651
('False_travel')	('False_chronic_disease')	0.94815948	0.999916	1.000284
('False_chronic_disease')	('False_travel')	0.94815948	0.94850889	1.000284

Appendix 7.FP-growth output with lift greater than 1.