Covid-19 classifier from Brazil's SRAG data

Machine Leaning Engineering Capstone Project Report - Pedro Tavares de Carvalho

ℹ️ *All of the datasets and info for this construction are in Brazilian Portuguese.*

I - Definition

COVID-19 information is spreading quickly, from multiple sources, fact, and fiction alike. From amazing data tracking by Worldometers to misleading information regarding drugs and vaccines, the range of reliability in data is very wide.

Brazil is not out of the woods yet with cases of COVID- 19 infections still rising. Unfortunately, Brazil is also in a way the epicenter of misinformation and data occlusion.

Currently going through difficult times with multiple crises springing up in the political, health, and economic sectors, there are two major factors that have fueled numerous crises in Brazil. The first, insufficient availability of valid, credible data from reliable sources and the second, overwhelming availability of misleading information and data.

Despite the grim state of information circulation in Brazil, there is light at the end of the tunnel with the OpenDataSUS SRAG (SARS for the non-Portuguese speakers) datasets. OpenDataSUS is a governmental effort to provide easy access to data regarding the Unified Health System (Sistema Único de Saúde in Portuguese) researches and aggregated information.

These datasets contain information on patients hospitalized due to SRAG symptoms from every year since 2013, they also provide information on patients who were diagnosed with COVID-19, or other diseases. Information relating to patients with inconclusive diagnostic results are also presented. An inquirer may question whether or not a patient was correctly diagnosed from available information.

These datasets were actually very well documented and organized, containing a description for what each column meant and the actual name of the column, all well described in a PDF file. (all the information is in Portuguese, but it's pretty easily translated via Google).

Based on the data, I wanted to try and build a classifier for the cases (since almost half of the cases are actually unidentified). The classifier will be evaluated via four metrics, the classic scoring, F1, accuracy, precision and recall, and will be evaluated in two datasets, which are defined in this section.

For that, I went through a few steps, the first one of which was **The Exploration.**

> ℹ️ All code is contained in this GitHub repository.

II - Analysis

I split the data into two types of data:

- Pre-COVID (which contained data from every year before 2020)
- Post-COVID (which contained data from 2020 onwards)

The first official case in Brazil was registered in February 2020, funny thing is that the first SRAG official case diagnosed as COVID-19 was actually reported in early 2019, but that's probably a reporting mistake, or a data processing one.

To solve that, I downloaded the data with a little bash magic

```
#!/bin/bash

BLUE="\e[1;34m"
WHITE="\e[1;37m"
RED="\e[1;31m"
GREEN="\e[1;32m"

MESSAGE="${BLUE} -> "
ERROR="${RED}ERROR: ${WHITE}"
SUCCESS="${GREEN}"

# This scripts gets all the data from the SRAG databases from Brazil's OpenD
echo
echo -e "${MESSAGE}Creating data directory..."
echo
mkdir -p data

sums=(`cat sums | awk '{print $1}'`)

# The 2020 data constantly moves between adresses, since it's updated every
# we use some grep magic
link2020=`curl -kL https://opendatasus.saude.gov.br/dataset/bd-srag-2020 | g

links=("https://opendatasus.saude.gov.br/dataset/18254c56-0859-4073-a6ea-97

YEAR=("201"{4..9})

for i in "${!links[@]}"; do
    echo -e "${MESSAGE}Downloading the ${WHITE}${YEAR[$i]} ${BLUE}data...${

    curl -Lko "data/${YEAR[$i]}.csv" "${links[$i]}"

    echo
    echo -e "${MESSAGE}Checking sha256 sum for ${WHITE}${YEAR[$i]} ${BLUE}fi

    current_sum=`sha256sum "data/${YEAR[$i]}.csv" | awk '{print $1}'`

    if [ $current_sum != "${sums[$i]}" ]; then
        echo -e "${ERROR}Checksum failed! Try downloading again..."
        exit 1
    else
        echo
        echo -e "${SUCCESS}Successfully downloaded files for year ${WHITE}$
        echo
    fi
done
```

```
48
49  echo -e "${MESSAGE}Downloading the ${WHITE}2020 ${BLUE}data..."
50  curl -Lo data/2020.csv $link2020
51
52  echo -e "${SUCCESS}Successfully downloaded all files!"
```

GET_DATA.sh hosted with ❤ by **GitHub** **view raw**

Getting data from the OpenDataSUS sources.

This script has to handle a little weird thing because the link for the 2020 data is constantly updated as new data arrives, so I handled that with **grep** and **awk**.

After downloading everything, it was time to do some data exploration and processing.

The first step was to consume the data into the data frames, and this was done through pandas and CSV reading.

```
1   # latin2 encoding because of the Brazilian Portuguese characters (which prol
2   # due to some SQL configuration)
3   df_2020 = pd.read_csv('data/2020.csv', encoding='latin2', sep=';', low_memor
4
5   # the other dataframes have to be concatenated together, from 2014 to 2019
6
7   # reading all dataframes
8   dfs = []
9   for year in range(2014, 2020, 1):
10      dfs += [pd.read_csv('data/{}.csv'.format(year), encoding='latin2', sep=
11
12  # and concatenating them together
13  df_control = pd.concat(dfs, sort=True)
```

consume_dfs.py hosted with ❤ by **GitHub** **view raw**

Data Exploration

The first thing I did was validate the conjecture that a lot of the data was actually classified as unknown causes, and these graphics strongly point to it:

The massive amount of COVID after 2020 is not a surprise here, but the massive amount of unidentified shows pretty clearly that some numbers are misrepresented in the current data.

As you can see, the cases in the first half of 2020 are comparable to the total number of cases from 2014 to 2019, which shows an increase that doesn't seem likely to occur naturally.

This increase in unknown causes can also be seen in a time plot of the total number in 2020.

This graph clearly shows an immense increase in cases considered unknown in a similar fashion to the ones considered COVID.

Column Checking

The next step in the exploration phase is to try and identify utility midst chaos in the provided SARS data. Most of the columns describe either useless data regarding the subject of the registration (like the name of the patient, the identification number of the registration, or details regarding the diagnosis).

The columns that I considered useful are as follow:

- SG_UF_NOT -> This column identifies the Federal Unity (or state) where the patient was registered. This is useful mostly because the social background of each Brazilian state can differ hugely, and this can affect the treatment and exposure of the patient both to COVID-19 and to better health care.
- CS_SEXO -> This is the sex of the patient. Although COVID isn't the most misogynistic virus, it still has some biases.
- TP_IDADE and NU_IDADE_N -> These columns state the age of the patient.
- CS_RACA -> This describes the ethnicity of the subject.
- SURTO_SG, NOSOCOMIAL, FEBRE, TOSSE, GARGANTA, DISPNEIA, DESC_RESP, SATURACAO, DIARREIA, VOMITO, OUTRO_SIN -> These columns describe some basic diagnostic factors (symptoms like cough and fever, and environmental hazards, like the kind of work).
- PUERPERA, CARDIOPATI, HEMATOLOGI, SIND_DOWN, HEPATICA, ASMA, DIABETES, NEUROLOGIC, PNEUMOPATI, RENAL, OBESIDADE -> These define risk factors.
- VACINA -> This states if the patient was vaccinated for the flu.
- ANTIVIRAL, TP_ANTIVIR -> This states if the patient used any antiviral medicine.
- HOSPITAL -> This describes if the patient was sent to a hospital.
- UTI -> This states if the patient was held in an ICU.
- SUPORT_VEN -> This states if the patient used respiratory support.
- RAIOX_RES -> This states the result of the chest X-ray exam.
- AMOSTRA, TP_AMOSTRA -> This states if the patient collected samples for the diagnosis, and what kind of sample was collected.
- EVOLUCAO -> This states what was the evolution of the case if the patient died or was cured.
- ID_MN_RESI or CO_MUN_RES -> this identifies the city where the patient resides

Most of the columns already had numerical values, which saved a lot of time for me, but most of them also had a very large number of `nan` values. This was fixed by just adding the 'skipped' flag to these `nan` values, which was a valid flag in most of the column descriptions.

Some columns, though, had to have more processing time, to get rid of imperfections and to account for conditional data. This is where we get to **The Processing**.

The Processing

To process the data, there were some easy steps, and some steps that asked for some deeper thinking.

All the processing was done in this function below:

```
def pre_process(df_in):
    # feature_list is the list of selected features
    df_out_X = df_in[feature_list]
    df_out_Y = df_in['CLASSI_FIN']


    # converting categorical into numeric
    df_out_X.loc[:,'SG_UF_NOT'] = df_out_X['SG_UF_NOT'].astype('category').c

    df_out_X.loc[:,'CS_SEXO'] = df_out_X['CS_SEXO'].astype('category').cat.c

    # the age in older dataframes has two formats
    # one where the value of 'NU_IDADE_N' is the category of if
    # as in years, months or days, and the other where this definition
    # is in another column, 'TP_IDADE'

    # to tackle that, I did it like this
    for i in range(len(df_out_X.index)):
        age = df_out_X.loc[i, 'NU_IDADE_N']
        if age > 1000:
            category = age // 1000
        else:
            category = df_out_X.loc[i, 'TP_IDADE']

        if category == 4:
            df_out_X.loc[i, 'NU_IDADE_N'] = age
        else:
            df_out_X.loc[i, 'NU_IDADE_N'] = 0


    # this was filling the null values with 9, the 'skip' token
    df_out_X = df_out_X.fillna(9)

    df_out_Y[df_out_Y['CLASSI_FIN'] != 5] = 0
    df_out_Y[df_out_Y['CLASSI_FIN'] != 0] = 1
    return df_out_X.values, df_out_Y.values
```

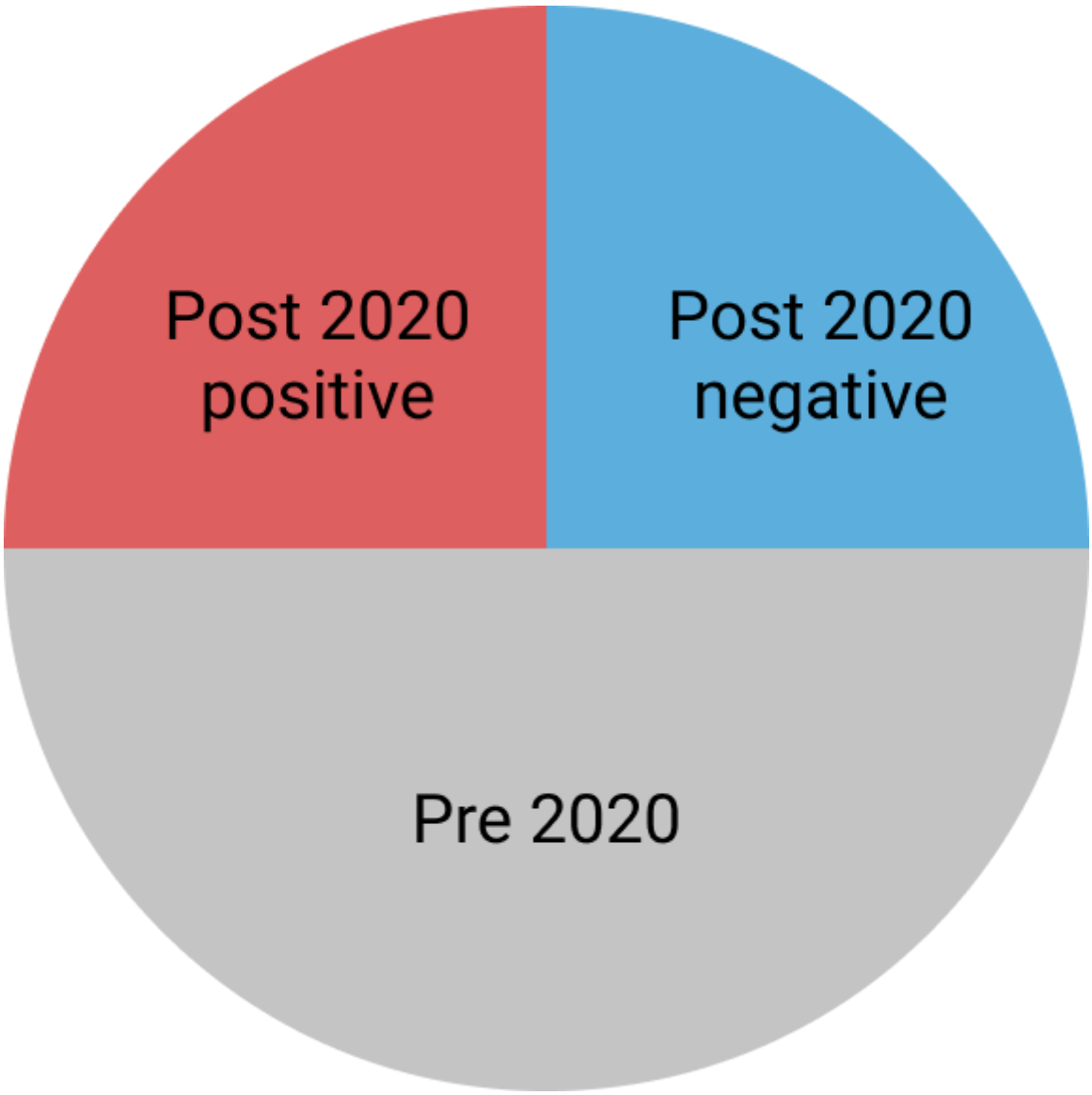
process_data.py hosted with ❤ by **GitHub** **view raw**

The next step was normalization and then we'd go straight for model construction.

The Model

To build the model, I needed to clarify what my training and testing datasets would be. You see, this specific data is weird because the labeling is not consistent through time.

The data is divided into three main categories.



For the Pre 2020 category, all data is definitely not COVID. Even the unknown causes data. As for the 2020 data, all positive data is classified clinically, so we can be sure that it's positive, in the exam sense. Meanwhile, the Post 2020 negative data is, as we've demonstrated, a bit iffy. The labeling is not trustworthy and it probably wouldn't help the model in any way.

Considering that, I made a choice to divide the training code in these two labels:

- Pre 2020, as the negative
- Post 2020 positive, as the positive

and train a binomial classifier based on these labels.

This choice can cause some biases if the distribution of this data has changed in ways other than the symptomatic, but, given the macrostructure of the data, it's the best solution I could think of.

Moving on, we get to the model

XGBoost

The current state of the art for tabular classification ranges in a few algorithms, and XGBoost is one that is close to the state of the art. There are some others, like the FastAI tabular algorithm, random forests, and others, but XGBoost is among the best, and its implementation is simple and reliable, being contained in the SageMaker default library.

Considering the data had already been preprocessed, the training was made quite simple. Just upload the data to S3 and use the built-in classifier for SageMaker.

This model gave out some interesting results. In the training process, the model achieved incredibly good metrics, with accuracy near perfect.

```
F1:  0.9992641648270787
acc:  0.9990455358918274
prec:  0.9986517955631817
recall:  0.99987728555651
```

These metrics have unusually high values, with accuracy that made me uncomfortable, but these values were from the testing dataset, which contained never seen before data, therefore the fear of overfitting was not that great.

Anyway, these metrics were from the testing dataset, which gave me confidence that, at least when the labels were reliable, the model was also reliable.

From this point, I had built a COVID-19 classifier from SRAG data. But I wanted to test this classifier against the cases from 2020 which were classified as Unknown Causes, to see if the model was compatible with studies from earlier this year that state how many cases of COVID-19 were hidden in the unknown causes.

To do that, I first made the same processing from before in the Post 2020 positive cases, then ran the same metrics, but considering the whole data as positive, to get a grasp for how many cases my model would classify as COVID, from the unknown causes.

```
F1:  0.45086413051108415
acc:  0.29104234134080914
prec:  1.0
recall:  0.29104234134080914
```

From this test, I got about 30% accuracy, which was much less than the statistically expected, but still, meant that my model considered that there were about 40k more COVID cases in the dataset that the official classification.

Conclusion

Statistically, the model was incorrect, but this effort doesn't seem meaningless. There is some information to be extracted from the data, and the classifier seems to do a good job in the labeled data, and from the unlabeled, it can at least split apart some of it.

There are a bunch of other datasets regarding COVID-19 in Brazil, and they're mostly unexplored in a machine learning environment. This project gave me the opportunity to see for myself what can be done, and I intend to keep digging and improving the models.