SCC0244 - Mineração a partir de Grandes Bases de Dados

Projeto 1

Profa. Dra. Agma J. M. Traina

Prof. Dr. Caetano T. Jr.

Christian C. Bones, João V. O. Novaes

Alunes:

Luiz Gustavo Ribeiro - 5967710

lgribeiro.info@gmail.com

Marcel Otoboni de lima - 9791069

marcel.lima@usp.br

Janeiro 2021

Conteúdo

1 Proposta

Será utilizada uma base de dados sobre os registros de pacientes relacionados ao COVID-19 disponibilizado pela FAPESP em: https://repositoriodatasharingfapesp.uspdigital.usp.br/ A Figura 1 apresenta as tabelas que serão utilizadas neste trabalho. A tabela paciente contém todos os pacientes que receberam atendimento no hospital com suspeita de Covid-19. A tabela exame contém todos os exames requisitados para um determinado paciente e os respectivos resultados. E a tabela desfecho contém quais foram as conclusões obtidas para o paciente

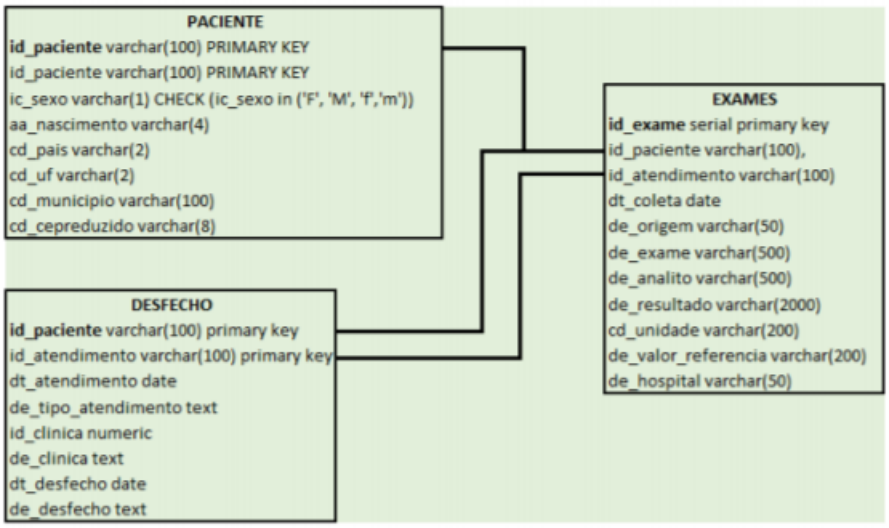

Figura 1: Diagrama das tabelas do banco

Nas seções abaixo seram apresentadas algumas questões para recuperar informações sobre os dados presentes na base de dados, úteis para cada um tomar um conhecimento geral dos dados disponibilizados.

2 Questões

Questões resolvidas através dos conhecimentos obtidos pela disciplina e utilizando os materiais fornecidos como os slides[1].

2.1 Questão 1

a) Qual a quantidade de pacientes presente na base de dados?

b) Quantos são homens e quantos são mulheres?

Resposta:

Para a questão "a"foi feito um simples count(*) da tabela de paciente, e para a questâo

”b”um group by do sexo dos pacientes, resultado do sql abaixo:

```sql
SELECT COUNT(*) FROM PACIENTE;

Select ic_sexo, count(ic_sexo)
    from paciente
    group by paciente.ic_sexo
    having UPPER(paciente.IC_SEXO) = 'M'
        or UPPER(paciente.IC_SEXO) = 'F';
```

Figura 2: Sql da questão 1

a)

	count bigint
1	332117

Figura 3: Resultado da execução 1a

b)

Data Output | Explain | Notifications | Me

	ic_sexo character varying (1)	count bigint
1	F	179864
2	M	152251

Figura 4: Resultado da execução 1b

2.2 Questão 2

Baseado na questão anterior:

a) Qual é faixa etária dos pacientes homens e mulheres?

b) Qual a distribuição dos quartis dentro de cada faixa?

c) Qual a distribuição em cada gênero por década de vida?

Resposta:

a) Para solução da questão ”a”foi feito uma contagem filtrada para homens e outra para mulheres, além disso, para a terceira coluna calculou-se a faixa etária dos pacientes, na

linha 6 garantiu-se que o ano de nascimento é um número pois há linhas com o valor "AAAA"nesta coluna.

```
1  SELECT
2      COUNT(P.IC_SEXO) FILTER (where UPPER(P.IC_SEXO) = 'F') as FEMALE,
3      COUNT(P.IC_SEXO) FILTER (where UPPER(P.IC_SEXO) = 'M') as MALE,
4      FLOOR((date_part('year', CURRENT_DATE)-P.AA_NASCIMENTO::INTEGER)/10) || '0s' as DECADA
5  FROM PACIENTE P
6  WHERE P.AA_NASCIMENTO SIMILAR TO '%(0|9)%'
7  GROUP BY DECADA;
```

Figura 5: Sql da questão 2.a

	female bigint	male bigint	decada text
1	3467	3902	00s
2	4993	4924	10s
3	26758	23150	20s
4	52269	41919	30s
5	41897	35183	40s
6	23174	21942	50s
7	14175	11739	60s
8	7258	6349	70s
9	3852	2436	80s
10	296	103	90s

Figura 6: Resultado da execução 2a

c)

```sql
with ages as (
    select min(date_part('year', CURRENT_DATE)- P.AA_NASCIMENTO::int)::int as min_age,
           max(date_part('year', CURRENT_DATE)- P.AA_NASCIMENTO::int)::int as max_age
    from PACIENTE P
    WHERE P.AA_NASCIMENTO SIMILAR TO '%(0|9)%'
),
     histogram as (
   select width_bucket((date_part('year', CURRENT_DATE)- P.AA_NASCIMENTO::int),
                       min_age, max_age, 9) as bucket,
          int4range(min(date_part('year', CURRENT_DATE)- P.AA_NASCIMENTO::int)::int,
                    max(date_part('year', CURRENT_DATE)- P.AA_NASCIMENTO::int)::int, '[]') as range,
        COUNT(P.IC_SEXO) FILTER (where UPPER(P.IC_SEXO) = 'F') as FEMALE,
        COUNT(P.IC_SEXO) FILTER (where UPPER(P.IC_SEXO) = 'M') as MALE
     from paciente P, ages
         WHERE P.AA_NASCIMENTO SIMILAR TO '%(0|9)%'
 group by bucket
 order by bucket
)
 select bucket, range, FEMALE, MALE
   from histogram;
```

Explain Notifications Messages

Figura 7: Sql da questão 2.c

	bucket integer	range int4range	female bigint	male bigint
1	1	[1,11)	3915	4316
2	2	[11,21)	5505	5428
3	3	[21,31)	30366	25824
4	4	[31,41)	53306	42775
5	5	[41,51)	39142	33398
6	6	[51,61)	22043	20798
7	7	[61,71)	13475	11151
8	8	[71,81)	6667	5793
9	9	[81,90)	3424	2061
10	10	[90,91)	296	103

Figura 8: Resultado da execução 2c

2.3 Questão 3

Qual a maior quantidade de exames solicitados para um único paciente ?

Resposta:

Para esta questão foi feito um group by dos pacientes distintos da tabela de exames, e ordenados descendentemente com limite de uma linha:

```sql
SELECT id_paciente, count(distinct id_exame)
FROM exames
group by id_paciente
ORDER BY 2 DESC LIMIT 1
```

Figura 9: Sql da questão 3

	id_paciente character varying (100)	count bigint
1	d5aef7e019e21e760b0a88f6...	10271

Figura 10: Resultado da execução 3

2.4 Questão 4

Qual é a média de exames pedidos para homens e para mulheres?

Resposta:

```sql
with totalPacientes as (
    Select ic_sexo as sexo, count(ic_sexo) as total
        from paciente
        group by paciente.ic_sexo
        having UPPER(paciente.IC_SEXO) = 'M'
            or UPPER(paciente.IC_SEXO) = 'F'
),
    totalExames as(
    Select ic_sexo as sexo, count(id_exame) as total_exame
    from (exames inner join paciente on exames.id_paciente = paciente.id_paciente)
        group by paciente.ic_sexo
        having UPPER(paciente.IC_SEXO) = 'M' or UPPER(paciente.IC_SEXO) = 'F'
)
 select tp.sexo, te.total_exame/tp.total as media_exames
   from totalPacientes as tp, totalExames as te where tp.sexo = te.sexo ;
```

Figura 11: Sql da questão 4

	sexo character varying (1)	media_exames bigint
1	F	30
2	M	29

Figura 12: Resultado da execução 4

2.5 Questão 5

Responda:

a) Quantos exames de Coronavírus (2019-nCoV) foram solicitados?

b) Quantos deles apresentam resultado positivo?

Resposta:

Para a alternativa 'a' foi feito um simples count onde o exame é o de nosso interesse. Para a alternativa 'b' foi adicionado um group by de resultado, contando assim a ocorrência de cada resultado, na figura 15 vê-se que as linhas 5 e 6 correspondem a resultados positivos, somando 19384 casos detectados. Como vantagem de resolver esta alternativa com group by foi possível ver a maior ocorrência de valores para esta coluna, no caso um valor não semântico "raspado de orofaringe e nasofa...", nota-se que será necessário processamento dos dados para organizar os resultados.

```
1  SELECT COUNT(*)
2  FROM EXAMES E
3  WHERE E.DE_EXAME = 'NOVO CORONAVÍRUS 2019 (SARS-CoV-2), DETECÇÃO POR PCR';
4
5  SELECT COUNT(*), E.DE_RESULTADO
6  FROM EXAMES E
7  WHERE E.DE_EXAME = 'NOVO CORONAVÍRUS 2019 (SARS-CoV-2), DETECÇÃO POR PCR'
8  GROUP BY E.DE_RESULTADO;
```

Figura 13: Sql da questão 5

a)

	count bigint
1	191064

Figura 14: Resultado da execução 5a

b)

	count bigint	de_resultado character varying (2000)
1	69001	raspado de orofaringe e nasofa...
2	56551	NÃO DETECTADO (NEGATIVO)
3	19299	NÃO DETECTADO
4	16715	swab de nasofaringe
5	11217	DETECTADO (POSITIVO)
6	8167	DETECTADO
7	6143	raspado de nasofaringe
8	2722	raspado de material do trato re...
9	520	raspado de material do trato re...
10	267	aspirado de nasofaringe

Figura 15: Resultado da execução 5b

2.6 Questão 6

Para cada idade, mostre os resultados dos exames de Coronavírus (2019-nCoV).

Resposta:

Para esta questão foi realizado um join entre as tabelas pacientes e exames e um matching de string para garantir que o exame em questão é de covid, na linha 4 da figura 16 é garantida que o ano de nascimento tem um número, pois alguns pacientes possuem este campo como "AAAA", nas linhas 5 e 6 garante-se que há um resultado válido("detectado"ou "não detectado") pois vários valores nesta coluna não são semanticamente interessantes para o trabalho.

```
1 SELECT COUNT(*), E.DE_RESULTADO, date_part('year', CURRENT_DATE)-P.AA_NASCIMENTO::INTEGER as IDADE
2 FROM PACIENTE P INNER JOIN EXAMES E ON P.ID_PACIENTE = E.ID_PACIENTE
3 WHERE E.DE_EXAME = 'NOVO CORONAVÍRUS 2019 (SARS-CoV-2), DETECÇÃO POR PCR'
4 AND P.AA_NASCIMENTO SIMILAR TO '___(0|1|2|3|4|5|6|7|8|9)'
5 AND (E.DE_RESULTADO ILIKE 'NÃO%'
6 OR E.DE_RESULTADO ILIKE 'DETECTADO%')
7 GROUP BY IDADE,E.DE_RESULTADO;
```

Figura 16: Sql da questão 6

	count bigint	de_resultado character varying (2000)	idade double precision
95	187	NÃO DETECTADO	24
96	683	NÃO DETECTADO (NEGATIVO)	24
97	80	DETECTADO	25
98	192	DETECTADO (POSITIVO)	25
99	208	NÃO DETECTADO	25
100	754	NÃO DETECTADO (NEGATIVO)	25
101	117	DETECTADO	26
102	179	DETECTADO (POSITIVO)	26
103	252	NÃO DETECTADO	26
104	863	NÃO DETECTADO (NEGATIVO)	26

Figura 17: Resultado da execução 6

2.7 Questão 7

Qual é o desfecho para a maioria dos casos registrados?

Resposta:

Analisamos os desfechos de casos relacionados ao covid, utilizando o operador LIKE para encontrar palavras chave, como COVID e SARS, na figura 19 é possível ver que há poucos desfechos registrados dado a quantidade de exames de covid, logo não parece muito vantajoso utilizar esta tabela para a árvore de decisão.

```
1  SELECT COUNT(*), D.DE_DESFECHO
2  FROM EXAMES E INNER JOIN DESFECHO D ON E.ID_ATENDIMENTO = D.ID_ATENDIMENTO
3  WHERE UPPER(E.DE_EXAME) LIKE '%COVID%'
4  OR UPPER(E.DE_EXAME) LIKE '%SARS-COV-2%'
5  GROUP BY D.DE_DESFECHO;
```

Figura 18: Sql da questão 7

	count bigint	de_desfecho text
1	5947	Alta Administrativa
2	182	Alta a pedido
3	51	Alta médica curado
4	67	Alta médica Inalterado
5	2591	Alta médica melhorado
6	6	Alta por abandono
7	8	Assistência Domiciliar
8	86	Óbito após 48hs de internação sem...
9	10	Transferência Inter-Hospitalar Exter...
10	258	[null]

Figura 19: Resultado da execução 7

2.8 Questão 8

Considerando as tabelas e as consultas solicitadas anteriormente, escreva/projete uma consulta para extrair algum conhecimento da base de dados que não foi descoberto pelas consultas anteriores. Apresente uma breve justificativa do objetivo da consulta e, por que esse objetivo é relevante.

Resposta: Para análise e computação da seção 3 do trabalho será necessário pegar todos os exames de pessoas que testaram para o covid:

```
SELECT P.ID_PACIENTE, P.IC_SEXO, P.AA_NASCIMENTO, E.DE_EXAME, E.DE_RESULTADO, E.DE_VALOR_REFERENCIA, E.DE_ANALITO
FROM PACIENTE P INNER JOIN EXAMES E ON P.ID_PACIENTE = E.ID_PACIENTE
WHERE P.ID_PACIENTE IN (SELECT E.ID_PACIENTE
FROM EXAMES E
WHERE E.DE_EXAME = 'NOVO CORONAVÍRUS 2019 (SARS-CoV-2), DETECÇÃO POR PCR')
AND DATE_PART('year',  E.DT_COLETA) = 2020;
```

Figura 20: Sql da questão 8

	id_paciente character varying (100)	ic_sexo character varying (1)	aa_nascimento character varying (4)	de_exame character varying (500)	de_resultado character varying (2000)	de_valor_referencia character varying (200)
1	15769D85DDDAF296289DE4...	F	1971	NOVO CORONAVÍRUS 2019 (S...	DETECTADO (POSITIVO)	[null]
2	15769D85DDDAF296289DE4...	F	1971	NOVO CORONAVÍRUS 2019 (S...	raspado de orofaringe e nasofa...	[null]
3	15769D85DDDAF296289DE4...	F	1971	COVID19, ANTICORPOS IgG, s...	4,9	inferior a 0,8
4	15769D85DDDAF296289DE4...	F	1971	COVID19, ANTICORPOS IgG, s...	REAGENTE	Não reagente
5	15769D85DDDAF296289DE4...	F	1971	COVID19, ANTICORPOS IgM, ...	1,1	inferior a 0,8
6	15769D85DDDAF296289DE4...	F	1971	COVID19, ANTICORPOS IgM, ...	REAGENTE	Não reagente
7	15CAFCF6A89A5B1158AA61...	F	1975	SODIO, soro	139	136 a 145
8	9A065FC0BCA852472718C4...	F	1973	URINA TIPO I - JATO MEDIO	amarela	[null]
9	AF38FF33C6F9A1DAAC2AF5...	M	1989	ZINCO, plasma	1,09	0,50 a 1,10
10	283A48AA24494523F307120...	M	1986	HEMOGRAMA, sangue total	16,1	13,5 a 17,5
11	B0152783E77C33BDA6F9E49...	F	1979	HEMOGRAMA, sangue total	12,1	11,9 a 15,5
12	B0152783E77C33BDA6F9E49...	F	1979	CA 19-9, soro	7,9	Até 37
13	85ABB010B4A9027D9840052...	F	1963	POTASSIO, plasma	4,0	3,5 a 4,5
14	846D64573825521D4DB7417...	F	1981	GLICOSE, plasma	93	75 a 99
15	846D64573825521D4DB7417...	F	1981	HIV1/HIV2, ANTICORPOS E A...	0,06	[null]

Figura 21: Resultado da execução 8

3 Trabalho

3.1 Criar nova base de dados Covid19

Para a criação da nova base dados foi decidido realizar uma transformação da tabela de exames e pacientes. Foi utilizado a tabela resultante do sql da seção 2.8 - Questão 8, com todos os exames de pacientes que realizaram o teste de covid, sob esta tabela foi realizada operações de tal forma que cada paciente único só ocupe uma linha com as seguintes colunas colunas:

- Id do paciente
- Gênero
- Ano de nascimento
- N colunas de exames realizados

Se um paciente fez um exame que nenhum outro paciente fez, uma nova coluna será adicionada com o valor no campo deste paciente sendo, 0 se negativo ou 1 se positivo, todos os outros pacientes terão esta coluna marcada como nulo no momento da criação da coluna. Ou seja, a matrix tem uma quantidade significativa de nulos.

Para computação dos resultados da tabela de exames foi utilizado python e regex[2] para identificação e matching de strings da coluna DE_RESULTADO, abaixo temos a conversão dos exames de covid para binário(0 - negativo, vírus não encontrado; 1 - positivo, vírus encontrado):

In [10]:
```
'''
Este é um caso especial e nao foi observado para outros exames.
result_binary = 0, se não foi detectado covid-19 no paciente
result_binary = 1, cc
'''
covid_PCR_df = pd.DataFrame(data.loc[data['exame'] == 'NOVO CORONAVÍRUS 2019 (SARS-CoV-2), DETECÇÃO POR PCR'])
```

In [11]:
```
'''
Importante notar que o número de conversões foi apenas aproximadamente a metade,
isto porque na base de dados para cada exame de covid realizado de um paciente
este teste está registrado duas vezes,
uma linha contém dados analíticos e outra contém o método do teste(?)
'''
counter_covid_PCR = 0

for index, row in covid_PCR_df.iterrows():
    if(re.search('NÃO DETECTADO', row['resultado'], re.IGNORECASE)):
        result_binary[index] = 0
        counter_covid_PCR += 1
    elif(re.search('DETECTADO', row['resultado'], re.IGNORECASE)):
        result_binary[index] = 1
        counter_covid_PCR += 1
print("Numero de resultados de exames 'NOVO CORONAVÍRUS 2019 (SARS-CoV-2), DETECÇÃO POR PCR' convertidos para binario: "
      + str(counter_covid_PCR)+ " de um total de: " + str(len(covid_PCR_df)))
```

```
Numero de resultados de exames 'NOVO CORONAVÍRUS 2019 (SARS-CoV-2), DETECÇÃO POR PCR' convertidos para binario: 95234 de um tota
064
```

Figura 22: Conversão de exames de covid

Outros casos exigiram também a interpretação conjunta da coluna de de_resultado e da coluna de_valor_referencia, abaixo temos o código que reconhece exames do tipo menor que algum valor(<x), ou seja, se de_resultado ter um valor maior que isso será convertido para - 1, caso contrário - 0:

```
counter_lesst = 0

for index, row in lesst_df.iterrows():
    if(row['resultado'].replace(".","").replace(",",".").isdigit() and row['referencia'] is not None):
        if(float(row['resultado'].replace(".","").replace(",",".")) <= float(row['referencia'].split()[-1].replace(".","").replace
(",",".")):
            result_binary[index] = 0
        elif(float(row['resultado'].replace(".","").replace(",",".")) >= float(row['referencia'].split()[-1].replace(".","").replace
(",",".")):
            result_binary[index] = 1
        counter_lesst += 1
print("Numero de resultados de exames do tipo 'menor que' convertidos para binario: "
      + str(counter_lesst)+ " de um total de: " + str(len(lesst_df)))
```

```
Numero de resultados de exames do tipo 'menor que' convertidos para binario: 113486 de um total de: 121278
```

Figura 23: Conversão de exames com referência do tipo '<x'

Após computar de 89.144 pacientes, 894.788 exames foram convertidos para 0 ou 1 de um total de 1.571.682, ou seja, 676.894 não foram computados. Como próximo passo, esta tabela resultante foi salva para uma base de dados:

Query Editor Query History

```
1  CREATE TABLE public.exames (
2      id_paciente text,
3      genero text,
4      ano_nascimento text,
5      "UREIA, soro" integer,
6      "FOSFORO INORGANICO, soro" integer,
7      "GLICOSE, plasma" integer,
8      "HEMOGRAMA, sangue total" integer,
9      "SODIO, soro" integer,
10     "GLICOSE, soro" integer,
11     "CORTISOL, soro" integer,
12     "CALCIO, soro" integer,
13     "FERRITINA, soro" integer,
14     "FERRO, soro" integer,
15     "CLORO, soro" integer,
16     "CREATININA, soro" integer,
17     "COLESTEROL TOTAL, soro" integer,
18     "TRIGLICERIDES, soro" integer,
19     "ZINCO, plasma" integer,
20     "NOVO CORONAVÍRUS 2019 (SARS-CoV-2), DETECÇÃO POR PCR" integer,
21     "COVID19, ANTICORPOS IgG, soro" integer,
22     "COVID19, ANTICORPOS IgA, soro" integer,
23     "LDL-COLESTEROL, soro" integer,
24     "HDL-COLESTEROL, soro" integer,
25     "VLDL-COLESTEROL, soro" integer,
26     "NÃO-HDL-COLESTEROL, soro" integer,
```

Figura 24: Criação da tabela resultante

Para a população dos dados foi usado o comando copy no terminal psql[3]:

```
aluno  (e) base  ~  sudo -u postgres -i
[sudo] password for aluno:
postgres@aluno-VirtualBox:~$ psql -U covid19 covid19
psql (12.4 (Ubuntu 12.4-0ubuntu0.20.04.1))
Type "help" for help.

covid19=# \c Modelo_covid
You are now connected to database "Modelo_covid" as user "covid19".
Modelo_covid=# \copy exames FROM '/home/aluno/Downloads/out.csv' DELIMITER ',' CSV
COPY 89144
Modelo_covid=#
```

Figura 25: Populando a nova tabela exames

	id_paciente text	genero text	ano_nascimento text	UREIA, soro integer	FOSFORO INORGANICO, soro integer	GLICOSE, plasma integer	HEMOGRAMA, sangue total integer	SOI inte
525	01657E2C1E89...	M	1977	[null]	[null]	[null]	[null]	
526	0165BADB9B81...	M	2004	[null]	[null]	[null]	[null]	
527	0165D57FDC59...	M	1977	[null]	[null]	0	0	
528	0166665FFD25...	F	1976	[null]	[null]	[null]	1	
529	0167F1188991...	F	1987	[null]	[null]	[null]	[null]	
530	016859E0C340...	M	1943	[null]	[null]	[null]	[null]	
531	016904F168025...	M	1976	[null]	[null]	[null]	0	
532	0169A7ADE9CB...	M	1980	[null]	[null]	[null]	[null]	
533	016A284934CC...	M	1959	0	0	1	0	
534	016BFEC306BE...	F	1977	[null]	[null]	[null]	[null]	
535	016C7631EC23...	F	1984	[null]	[null]	[null]	[null]	
536	016CC875F6F0...	F	1934	[null]	0	1	0	
537	016CF6A4CB24...	F	1965	[null]	[null]	[null]	[null]	
538	016D9E784F97...	M	1977	[null]	[null]	[null]	[null]	
539	016DF7CC5323...	M	1975	[null]	[null]	[null]	[null]	
540	016DFB556AFD...	F	1985	[null]	[null]	[null]	[null]	
541	016E3E7813B2...	F	AAAA	[null]	[null]	[null]	[null]	

Figura 26: Resultado de um simples select * da tabela criada e populada

3.2 Classificação

O ponto mais importante dessa questão é entender cada atributo (features) da base de dados. Nessa linha de raciocínio foi feito uma planilha com um estudo de todos os atributos da base relacionando a sua influência, caso tenha, com a Covid19. Com a informação da planilha, o estudo de entropia de cada atributo e alguns métodos da biblioteca scikitlearn [4] foram selecionados os atributos mais relevantes que otimizaram a classificação da classe alvo, "NOVO CORONAVÍRUS 2019 (SARS-CoV-2),DETECÇAO POR PCR". Para a classificação da classe alvo foi implementado também em python o modelo de machine learning, Árvore de Decisão [5]. Todos os códigos estão no arquivo '.zip' e também no repositório do Github [6].

Toda implementação dessa questão foi feita em python, usando o jupyter-notebook.

3.2.1 Disponibilizando os dados

Os dados usados nessa questão foram disponibilizados da base de dados criada na seção anterior. Abaixo encontra se o código de acesso ao banco e o comando SQL para o retorno dos dados.

Lendo os dados banco postgrees

```
import psycopg2
import pandas as pd

conn = psycopg2.connect(host="localhost", database="Modelo_covid",
                        user="covid19", password="covid19")

sql = "select * from exames;"
df = pd.read_sql_query(sql, conn)

conn = None
df.info
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 89144 entries, 0 to 89143
Data columns (total 74 columns):
 #   Column
---  ------
 0   id_paciente
 1   genero
 2   ano_nascimento
```

Figura 27: Retornando tuplas da nova base de dados

3.2.2 Exploração e tratamento dos dados

Esta etapa nos permite conhecer os dados e fazer o tratamento necessário para que o treinamento, avalição e classificação tenha uma bom desempenho e uma boa acurácia. Assim verificou se que a base possui muitos valores Nan (not a number), atributos com valores em string, numérico e categóricos e desbalanceamento da classe alvo a ser classificada (bem como outros atributos).

```python
# verificando a % de valores nulos/ausentes nos atributos do dataset

total = df.isnull().sum().sort_values(ascending=False)
percent_1 = df.isnull().sum()/df.isnull().count()*100
percent_2 = (round(percent_1, 1)).sort_values(ascending=False)
missing_data = pd.concat([total, percent_2], axis=1, keys=['Total', '%'])
missing_data.head(100)
```

	Total	%
ALFA-FETOPROTEINA, soro	89121	100.0
SARS-CoV-2, ANTICORPOS IgM E IgG, TESTE RÁPIDO	89067	99.9
CONTAGEM DE RETICULOCITOS, sangue total	88800	99.6
ANTICORPOS ANTI-PEROXIDASE TIROIDIANA, soro	88648	99.4
IgE, IMUNOGLOBULINA, soro	88423	99.2
...	...	...
COVID19, ANTICORPOS IgG, soro	72133	80.9
NOVO CORONAVÍRUS 2019 (SARS-CoV-2), DETECÇÃO POR PCR	202	0.2
ano_nascimento	0	0.0
genero	0	0.0
id_paciente	0	0.0

74 rows × 2 columns

Figura 28: Verificando valores Nulos

```python
# Comparando a variavel de interesse com a as variaveis categoricas

print(pd.pivot_table(df, index = 'NOVO CORONAVÍRUS 2019 (SARS-CoV-2), DETECÇÃO POR PCR',
                     columns = 'genero', values = 'ano_nascimento' ,aggfunc ='count'))

```

```
genero                                                F      M
NOVO CORONAVÍRUS 2019 (SARS-CoV-2), DETECÇÃO PO...
0.0                                                 40008  31467
1.0                                                  9556   7911
```

```python
df_remove = df.loc[(df['ano_nascimento'] == 'AAAA')]
df = df.drop(df_remove.index)

```

```python
df['ano_nascimento'] = 2020 -  df['ano_nascimento'].astype(int)
df.info()

```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 87919 entries, 0 to 89143
Data columns (total 74 columns):
 #   Column                                                Non-Null Count  Dtype
---  ------                                                --------------  -----
 0   id_paciente                                           87919 non-null  object
 1   genero                                                87919 non-null  object
 2   ano_nascimento                                        87919 non-null  int32
 3   UREIA, soro                                           11310 non-null  float64
```

Figura 29: Tratando os dados

Toda base foi convertida para atributos categóricos mas representados em números. Assim alguns atributos foram transformados em novas colunas, por exemplo o atributo 'nascimento' foi transformado em faixas de idade.

```python
# Convertendo atributo categorico
df['genero'] = df['genero'].map( {'F': 0, 'M': 1} ).astype('category')

```

```python
df['AgeBand'] = pd.cut(df['ano_nascimento'], 9)
df[['AgeBand', 'NOVO CORONAVÍRUS 2019 (SARS-CoV-2), DETECÇÃO POR PCR']].groupby(['AgeBand'],
                            as_index=False).mean().sort_values(by='AgeBand', ascending=True)
```

	AgeBand	NOVO CORONAVÍRUS 2019 (SARS-CoV-2), DETECÇÃO POR PCR
0	(-0.089, 9.889]	0.134434
1	(9.889, 19.778]	0.139848
2	(19.778, 29.667]	0.196376
3	(29.667, 39.556]	0.208792
4	(39.556, 49.444]	0.198476
5	(49.444, 59.333]	0.201643
6	(59.333, 69.222]	0.199639
7	(69.222, 79.111]	0.183182
8	(79.111, 89.0]	0.179707

Figura 30: Convertendo em categóricos

```
# tranformando em faixas de idades e atribuindo um valor numerico

df['idade'] = df['ano_nascimento'].astype(int)
df.loc[ df['idade'] <= 10, 'idade'] = 0
df.loc[(df['idade'] > 10) & (df['idade'] <= 19), 'idade'] = 1
df.loc[(df['idade'] > 19) & (df['idade'] <= 29), 'idade'] = 2
df.loc[(df['idade'] > 29) & (df['idade'] <= 39), 'idade'] = 3
df.loc[(df['idade'] > 39) & (df['idade'] <= 49), 'idade'] = 4
df.loc[(df['idade'] > 49) & (df['idade'] <= 59), 'idade'] = 5
df.loc[(df['idade'] > 59) & (df['idade'] <= 69), 'idade'] = 6
df.loc[(df['idade'] > 69) & (df['idade'] <= 79), 'idade'] = 7
df.loc[ df['idade'] > 79, 'idade'] = 8

df['ano_nascimento'] = df['idade'].astype('category')
df = df.drop(['AgeBand','idade'], axis=1)

```

Figura 31: Faixa de idade

3.2.3 Seleção de atributos e treinamento com Árvore de Decisão

Para criar um baseline, foi executado o modelo de Árvores de Decisão [7] usando o método de validação cruzada aninhada [8] com StratifiedKFold [9] , que separa a base em treinamento e teste com finalidade de diminuir o problema de overfiting [10] junto com o algoritmo SearchGrid [11] que retorna os melhores hiperparâmentros para o modelo.

O objetivo principal é ajustar o modelo com os melhores hiperparâmentros, bem como resolver o problemas do desbalanceamento e diminuir a dimensionalidade da base para que nos próximos treinamentos as métricas de validação e performance do algoritmo sejam melhores que a do baseline.

```
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import classification_report, precision_score, recall_score, roc_

hp_grid = {
    'criterion': 'entropy',
    'max_depth': 7,
    'max_features': 'sqrt',
    'max_leaf_nodes': 15,
    'min_samples_split': 10
}

# usando o metodo estratificado
def dt_metricas(X, y, hp_grid):

    model = DecisionTreeClassifier(criterion='entropy',
                                   max_depth=hp_grid['max_depth'],
                                   max_features=hp_grid['max_features'],
                                   max_leaf_nodes=hp_grid['max_leaf_nodes'],
                                   min_samples_split = hp_grid['min_samples_split'])

    cv = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

    nrauc = list()
    nrprecision = list()
    nrrecall = list()
    nraccuracy = list()

    for train_index, test_index in cv.split(X, y):
        X_train, X_test = X.iloc[train_index], X.iloc[test_index]
        y_train, y_test = y.iloc[train_index], y.iloc[test_index]
```

Figura 32: StratifiedKFold e SearchGrid

A seleção de atributos é uma etapa muito importante não somente para reduzir a dimensionalidade da base mas também para:

- Deixar o treinamento mais rápido, diminuindo o tempo de treinamento;
- Reduzir a complexidade de um modelo e torna-o mais simples de interpretar;
- Facilitar o entendimento da relação dos atributos com a saída;
- Melhorar a precisão do modelo se o subconjunto "ótimo"for escolhido; e
- Reduzir a chance de overfitting e melhora a generalização.

Deste modo, para entender a quantidade de atributos mais relevantes foi usado o algoritmo de eliminação recursiva de atributos (recursive feature elimination, RFE). RFE [12] [13] é um algoritmo cujo objetivo é encontrar o melhor subconjunto de atributos. Repetidamente, o algoritmo cria modelos e mantém o melhor ou o pior atributo a cada iteração. Então, um novo modelo é treinado com os melhores atributos até que os atributos se acabem. Finalmente, o algoritmo faz o ranking dos atributos baseado na ordem das suas eliminações. A implementação da RFE retorna o valor da curva roc AUC e a precisão para cada conjunto de atributos.

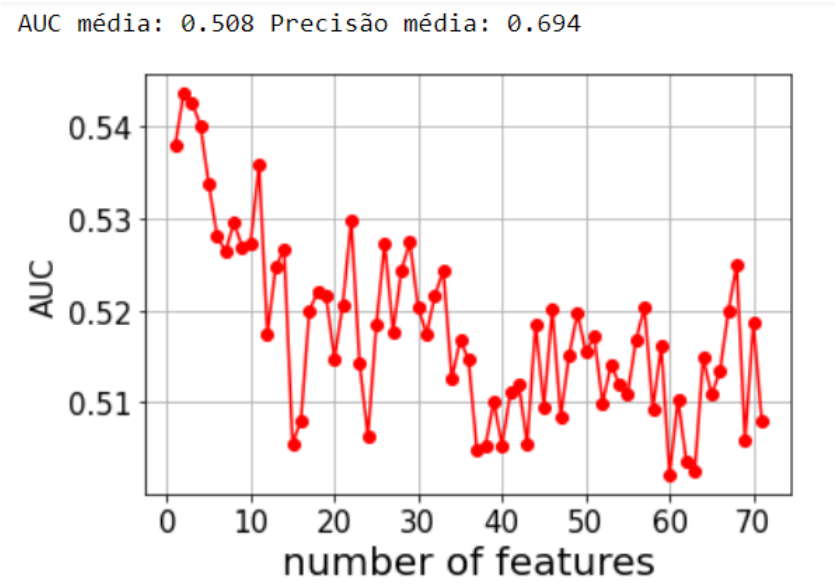

Figura 33: Avaliando o numero de atributos

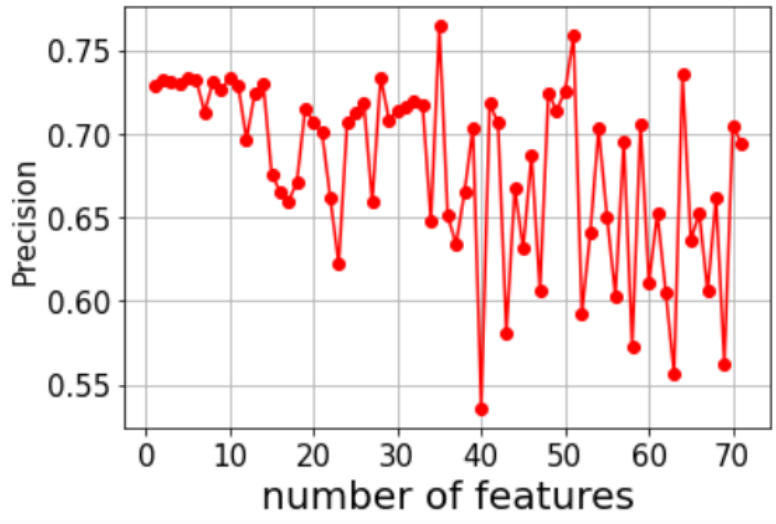

Figura 34: Avaliando o numero de atributo

Esse valor é uma base de quantos atributos pode se excluir da base sem que haja perda na qualidade da classificação e treinamento.

Para conhecer os atributos de mais importância na base de dados foi usado o Random Forest [14] (floresta aleatória) que pode lidar com dados em grandes volumes e com muitas dimensões. Assim ele processa milhares de variáveis de entrada e identifica quais as variáveis mais significativas, feature importance [15], por isso é considerado um bom método de redução de dimensões.

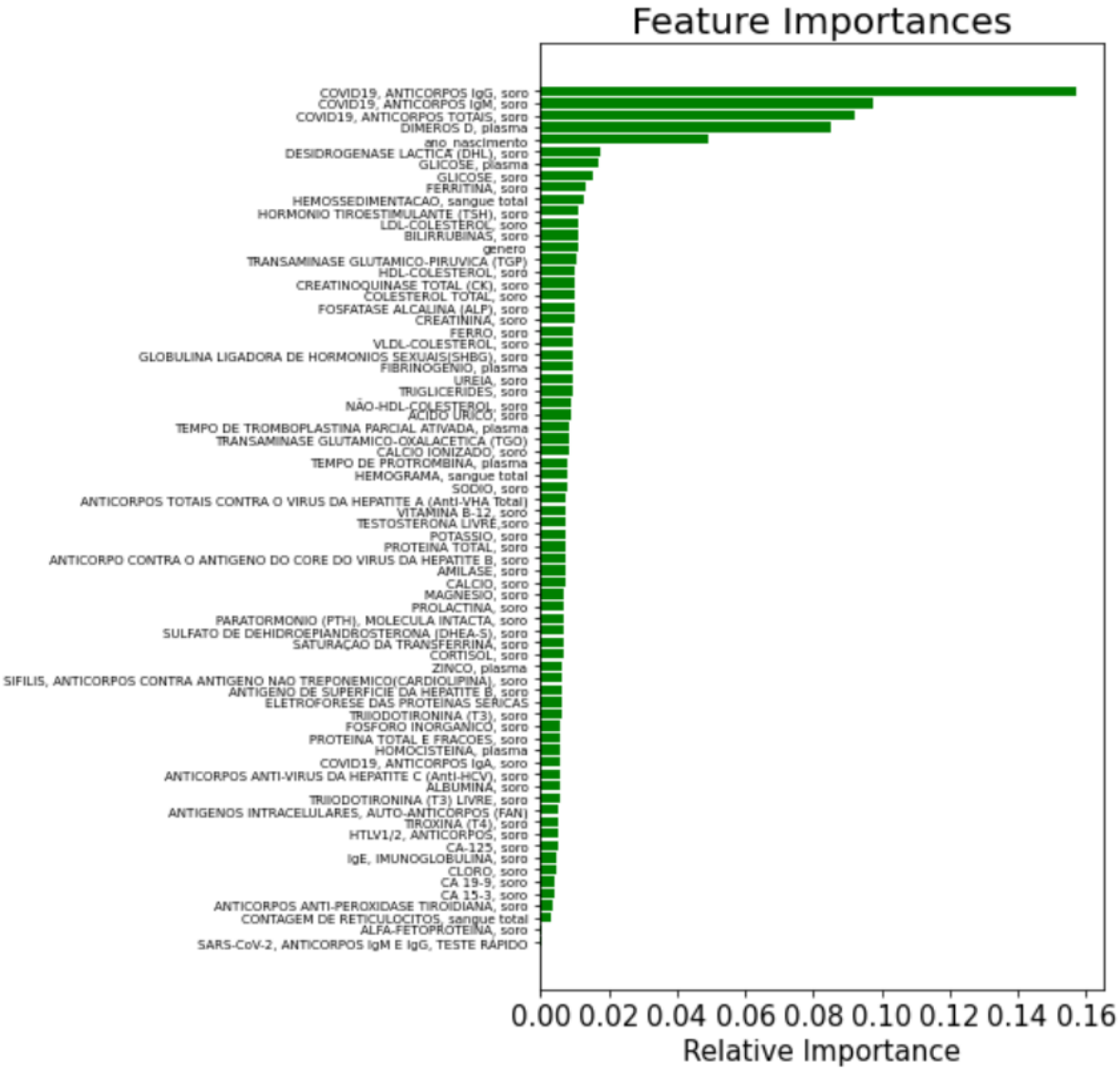

Figura 35: Random Forest - Grau de Importância dos Atributos

Com todas essas informações, a base de dados foi treinada e classificada removendo primeiramente 23 e depois 36 atributos.

Devido ao desbalanceamento [16] dos dados o algoritmo pode enviesar o resultado, então foi usado alguns métodos de subamostragem. O NearMiss, OneSidedSelection e o ALLKNN da biblioteca imblearn [17] são algoritmos que usam técnicas de subamostragem que torna a classe maioritária próxima ou igual à classe minoritária.

As técnicas de subamostragem foram executadas com a base reduzida em 38 atributos, aproximandamente a metade do que era inicialmente.

NearMiss [18]

NearMiss-3 é um algoritmo de 2 etapas. Primeiro, para cada amostra negativa, seus M vizinhos mais próximos serão mantidos. Então, as amostras positivas selecionadas são aquelas para as quais a distância média para os N vizinhos mais próximos é a maior.

- Usando NearMiss com os parâmetros version=3 e n_neighbors_ver3=10:

A classe majoritária foi reduzida de 70463 para 11658 (menor que a classe minoritária). Houve uma melhora significativa em todas as métricas.

```
X, y = data_base(df_ft)
yb = y
sm = NearMiss(version=3, n_neighbors_ver3=10)
X, y = sm.fit_sample(X, y)
print('Antes\nQuantidade positvo: {} e negativo: {}'
      .format(np.count_nonzero(yb == 1), np.count_nonzero(yb == 0)))
print('\nDepois')
print('Quantidade positvo: {} e negativo: {}'
      .format(np.count_nonzero(y == 1), np.count_nonzero(y == 0)))
ax = sns.countplot(x=y).set_title(
    'NearMiss Balanceamento - Quantidade de pessoas testadas')
```

```
Antes
Quantidade positvo: 17255 e negativo: 70463

Depois
Quantidade positvo: 17255 e negativo: 11658
```

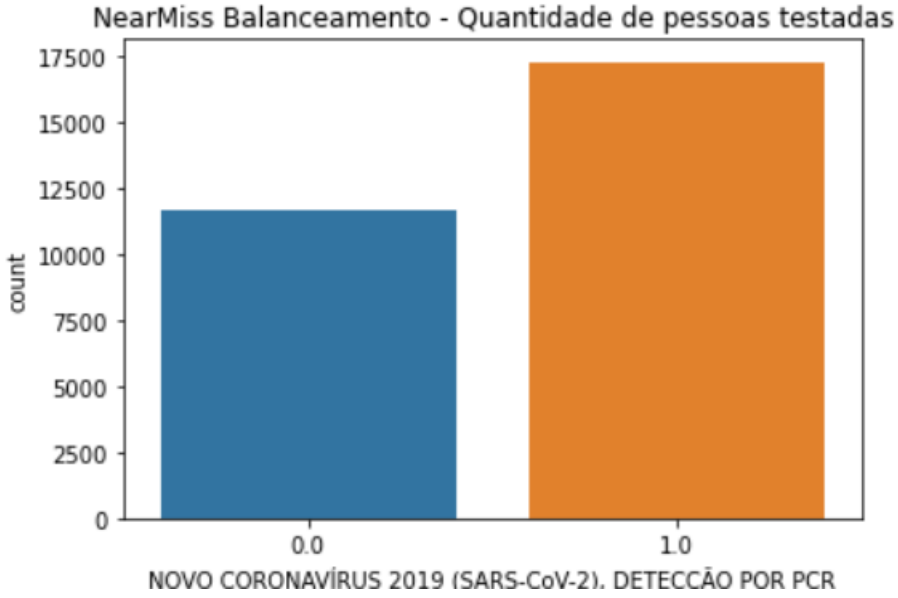

Figura 36: Balanceamento com NearMiss

```
Result StratifiedKFold
av_accuracy 0.8124717908824689
std_accuracy 0.0089015067290793O2
av_auc 0.8079653550512935
std_auc 0.0136022293827106O5
av_prec 0.8055816343281968
std_prec 0.009039148846150452
av_vrecall 0.8079653550512937
std_vrecall 0.013602229382710551

Result train_test_split
av_accuracy 0.8298461006398063
av_auc 0.8288031209427433
av_prec 0.8226647988227218
av_vrecall 0.8288031209427433
```

Figura 37: Avaliando com o balanceamento NearMiss

- Usando NearMiss com os parâmetros version=3 e n_neighbors_ver3=50:

A classe majoritária foi reduzida de 70463 para 17255 (igual a classe minoritária).Houve uma melhora significativa em todas as métricas.

```
X, y = data_base(df_ft)
yb = y
sm = NearMiss(version=3, n_neighbors_ver3=50)
X, y = sm.fit_sample(X, y)
print('Antes')
print('Quantidade positvo: {} e negativo: {}'
      .format(np.count_nonzero(yb == 1), np.count_nonzero(yb == 0)))
print('\nDepois')
print('Quantidade positvo: {} e negativo: {}'
      .format(np.count_nonzero(y == 1), np.count_nonzero(y == 0)))
ax = sns.countplot(x=y).set_title(
    'NearMiss Balanceamento - Quantidade de pessoas testadas')
```

```
Antes
Quantidade positvo: 17255 e negativo: 70463

Depois
Quantidade positvo: 17255 e negativo: 17255
```

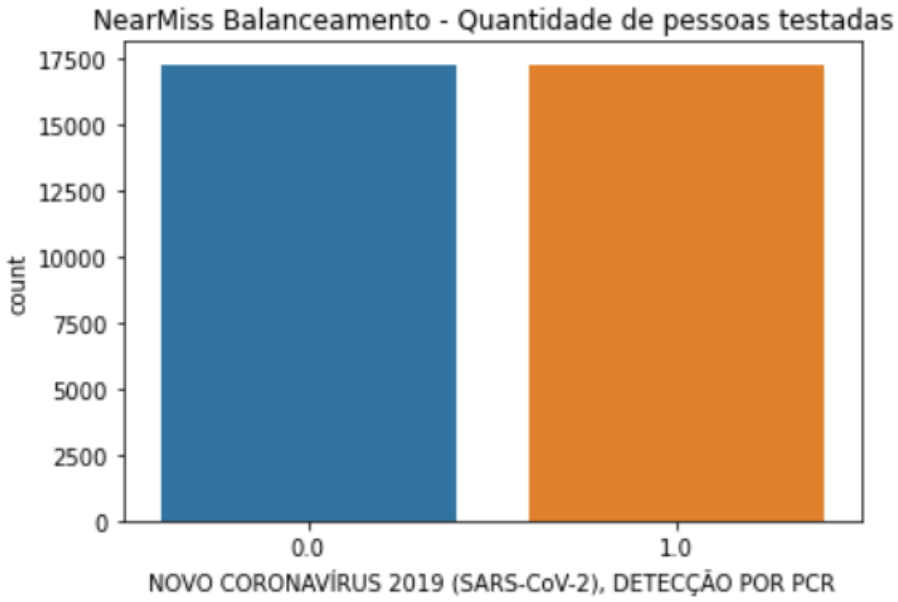

Figura 38: Balanceamento com NearMiss

```
Result StratifiedKFold
av_accuracy 0.7997392060272384
std_accuracy 0.018222638660580867
av_auc 0.7997356709825851
std_auc 0.018226207414219518
av_prec 0.8032628271027139
std_prec 0.019167775351932752
av_vrecall 0.7997356709825851
std_vrecall 0.01822620741421952

Result train_test_split
av_accuracy 0.7806432917994784
av_auc 0.7808320412590491
av_prec 0.7825814671337732
av_vrecall 0.7808320412590493
```

Figura 39: Avaliando com o balanceamento NearMiss

OneSidedSelection [19]

Irá usar TomekLinks [20] para remover amostras ruidosas.

- Usando OneSidedSelection com os parâmetros n_neighbors=1 e n_seeds_S=200:
  Reduziu muito pouco a classe majoritária e os resultados não foram satisfatórios

```
Antes
Quantidade positvo: 17255 e negativo: 70463

Depois
Quantidade positvo: 17255 e negativo: 69753
```

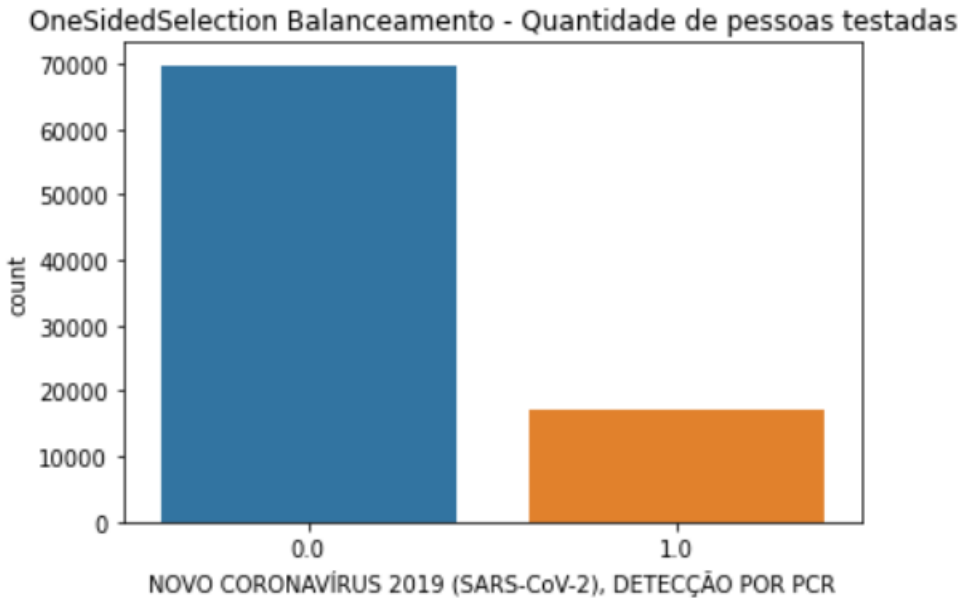

Figura 40: Random Forest - Grau de Importância dos Atributos

```
Result StratifiedKFold
av_accuracy 0.8058110099645039
std_accuracy 0.0029894574197052676
av_auc 0.5253868654892357
std_auc 0.011227872549051532
av_prec 0.6985597415345441
std_prec 0.041513411519856186
av_vrecall 0.5253868654892356
std_vrecall 0.011227872549051538

Result train_test_split
av_accuracy 0.8056545224686817
av_auc 0.5039318276702625
av_prec 0.6994551618715787
av_vrecall 0.5039318276702625
```

Figura 41: Avaliando com o balanceamento OneSidedSelection

ALLKNN [21]

Aplica um algoritmo de vizinhos mais próximos e "edita"o conjunto de dados removendo amostras que não concordam "o suficiente"com sua vizinhança.

- Usando OneSidedSelection com os parâmetros sampling_strategy='majority'e n_neighbors=20

A classe majoritária foi reduzida de 70463 para 17255 (igual a classe minoritária)

```
Antes
Quantidade positvo: 17255 e negativo: 70463

Depois
Quantidade positvo: 17255 e negativo: 17069
```

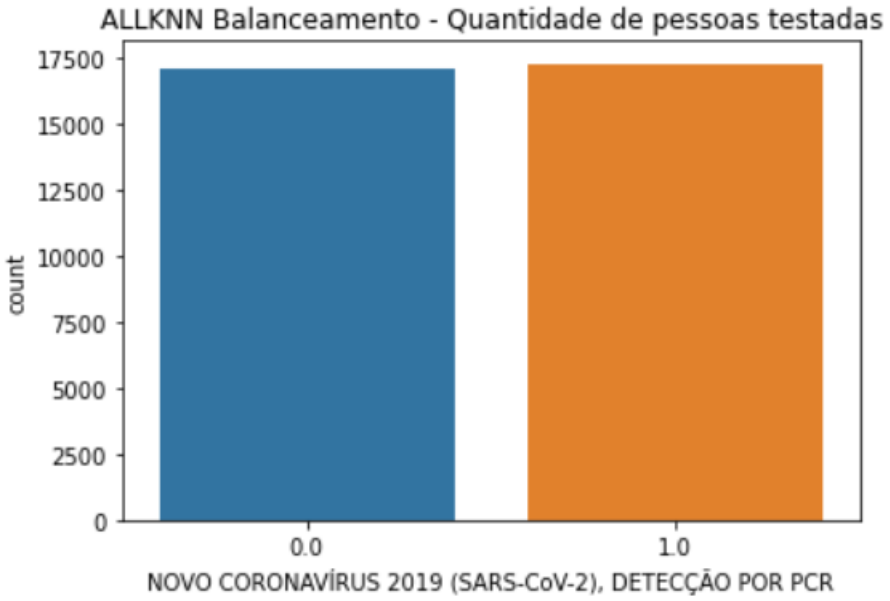

Figura 42: Balanceamento com ALLKNN

```
Result StratifiedKFold
av_accuracy 0.7582441044245589
std_accuracy 0.04039913529955311
av_auc 0.7587699987311813
std_auc 0.040594311095903265
av_prec 0.7828887030751839
std_prec 0.04175063744763199
av_vrecall 0.7587699987311813
std_vrecall 0.04059431109590328

Result train_test_split
av_accuracy 0.6767662053896577
av_auc 0.6763453065899933
av_prec 0.6883458369208739
av_vrecall 0.6763453065899933
```

Figura 43: Avaliando com o balanceamento ALLKNN

3.2.4 Resultados

A estratégia de seleção de atributos não melhorou significativamente a acurácia, precisão, recall e area da curva roc AUC do modelo [22]. Em contrapartida, quando criou se estratégias de subamostragem houve uma expressiva melhoria nas métricas citadas acima. Os resultados abaixo apontam que o balanceamento usando o algoritmo NearMiss obteve as melhores pontuações em todas as métricas.

Tabela com as métricas de validação da classificação usando o método StratifiedKFold

Ordenado pelo melhor Recall

Modelo	ACC	ACC_Std	AUC	AUC_std	Precision	Precision_Std	Recall	Recall_std
DTreeFeatImport_38_NearMiss10	0.812472	0.008902	0.807965	0.013602	0.805582	0.009039	0.807965	0.013602
DTreeFeatImport_38_NearMiss50	0.799739	0.018223	0.799736	0.018226	0.803263	0.019168	0.799736	0.018226
DTreeFeatImport_38_ALLKNN	0.748105	0.045241	0.748212	0.045780	0.761713	0.048637	0.748212	0.045780
DTreeFeatImport_38	0.807394	0.003066	0.522399	0.015395	0.674533	0.097919	0.522399	0.015395
DTreeFeatImport_51	0.806095	0.002909	0.513912	0.013220	0.656011	0.129664	0.513912	0.013220
DTreeFeatImport_38_OneSelection	0.782587	0.003678	0.512301	0.012462	0.641144	0.091194	0.512301	0.012462
DTreeBaseLine	0.803780	0.000651	0.502909	0.002573	0.657451	0.139542	0.502909	0.002573

Figura 44: Tabela dos resultados da Arvore de Decisão

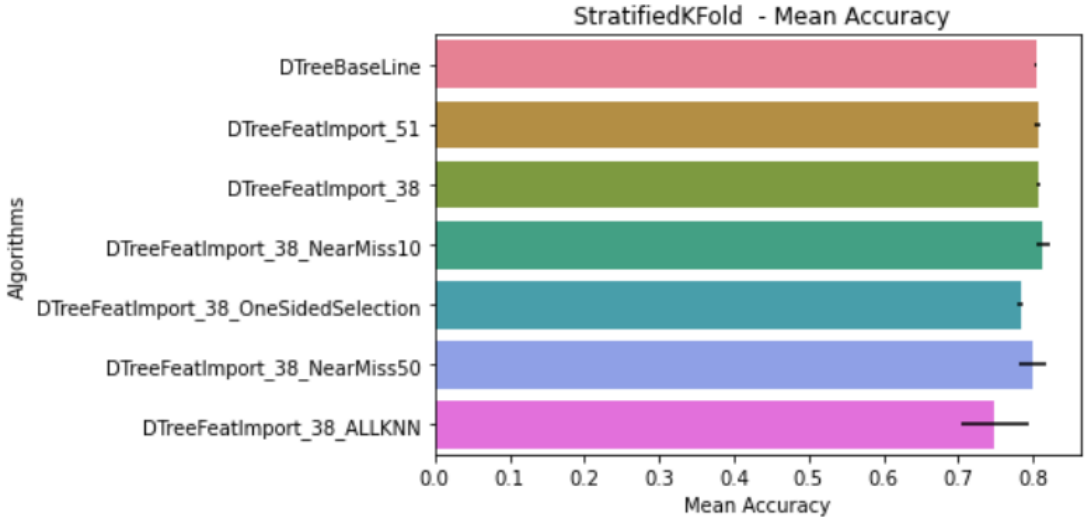

Figura 45: Gráfico da acurácia média das estratégias usadas

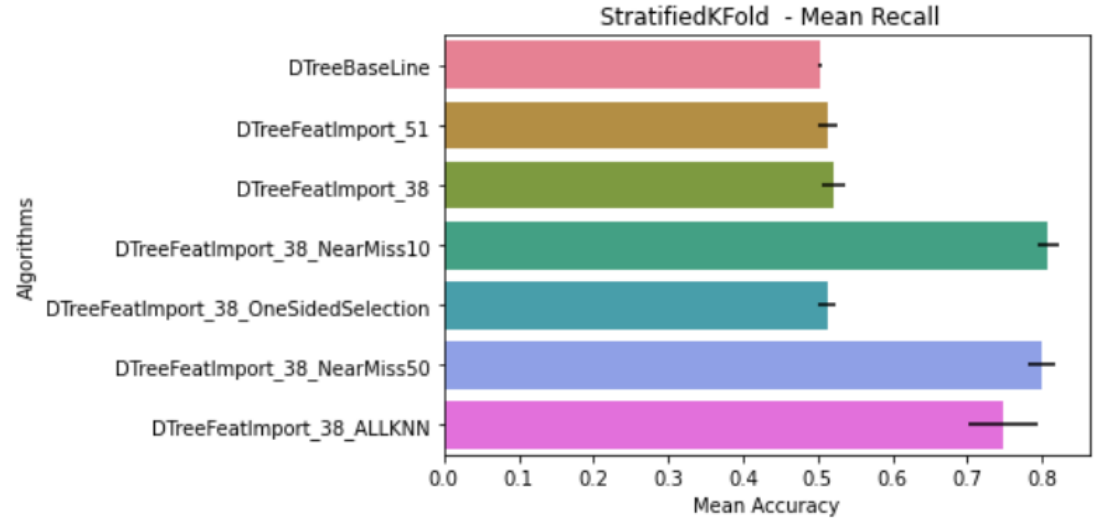

Figura 46: Gráfico do recall médio das estratégias usadas

A estratégia de reduzir a dimensionalidade da base de dados diminuiu drasticamente no número de atributos, trazendo ganho de perfomance relevante para o treinamento e classificação da base de dados, que estava sendo muito custoso. O balancemaneto da base de dados melhorou muito a métrica de recall, que é a mais importante quando se trata de exames médicos, pois não se pode classificar uma pessoa doente como não doente.

Referências

[1] Caetano Traina Júnior. *Slides do professor*.

[2] *Regular expression operations library python*. URL: https://docs.python.org/3/library/re.html.

[3] The PostgreSQL Global Development Group. *PostgreSQL interactive terminal(psql*. URL: https://www.postgresql.org/docs/9.2/app-psql.html.

[4] scikit-learn. *Machine Learning in Python*. URL: https://scikit-learn.org/stable/index.html. Acesso em: 01 dez. 2020.

[5] scikit-learn. $sklearn.model_selection.DecisionTreeClassifier$. URL: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html. Acesso em: 01 dez. 2020.

[6] Luiz Gustavo Ribeiro. *Mineração de dados*. URL: https://github.com/lgribeiro/mineracao_dados. Acesso em: 1 nov. 2020.

[7] Vooo – Insights. *Um tutorial completo sobre modelagem baseada em árvores de decisão (códigos R e Python)*. URL: https://www.vooo.pro/insights/um-tutorial-completo-sobre-a-modelagem-baseada-em-tree-arvore-do-zero-em-r-python/. Acesso em: 22 dez. 2020.

[8] Jason Brownlee. *Nested Cross-Validation for Machine Learning with Python*. URL: https://machinelearningmastery.com/nested-cross-validation-for-machine-learning-with-python/. Acesso em: 22 dez. 2020.

[9] scikit-learn. $sklearn.model_selection.StratifiedKFold$. URL: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html. Acesso em: 01 dez. 2020.

[10] 3dimensoes - Michel. *Overfitting e Underfitting*. URL: https://www.3dimensoes.com.br/post/overfitting-e-underfitting. Acesso em: 02 dez. 2020.

[11] scikit-learn. $sklearn.model_selection.GridSearchCV$. URL: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html. Acesso em: 01 dez. 2020.

[12] Jason Brownlee. *Recursive Feature Elimination (RFE) for Feature Selection in Python*. URL: https://machinelearningmastery.com/rfe-feature-selection-in-python/. Acesso em: 27 dez. 2020.

[13] scikit-learn. *Feature selection*. URL: https://scikit-learn.org/stable/modules/feature_selection.html. Acesso em: 27 dez. 2020.

[14] scikit-learn. *sklearn.ensemble.RandomForestClassifier*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier. Acesso em: 27 abr. 2020.

[15] Machine learning mastery - Jason Brownlee. *How to Calculate Feature Importance With Python*. URL: https://machinelearningmastery.com/calculate-feature-importance-with-python/. Acesso em: 15 dez. 2020.

[16] Machine learning mastery - Jason Brownlee. *Undersampling Algorithms for Imbalanced Classification*. URL: https://machinelearningmastery.com/undersampling-algorithms-for-imbalanced-classification/. Acesso em: 25 dez. 2020.

[17] imbalanced-learn. *imbalanced-learn API*. URL: https://imbalanced-learn.org/stable/api.html#module-imblearn.under_sampling. Acesso em: 30 dez. 2020.

[18] imbalanced-learn. *imblearn.under$_s$ampling.NearMiss*. URL: https://imbalanced-learn.org/stable/generated/imblearn.under_sampling.NearMiss.html. Acesso em: 04 jan. 2021.

[19] imbalanced-learn. *imblearn.under$_s$ampling.OneSidedSelection*. URL: https://imbalanced-learn.org/stable/generated/imblearn.under_sampling.OneSidedSelection.html. Acesso em: 04 jan. 2021.

[20] imbalanced-learn. *Tomek's links*. URL: https://imbalanced-learn.org/stable/under_sampling.html#tomek-links. Acesso em: 05 dez. 2021.

[21] imbalanced-learn. *imblearn.under$_s$ampling.ALLKNN*. URL: https://imbalanced-learn.org/stable/generated/imblearn.under_sampling.AllKNN.html. Acesso em: 04 jan. 2021.

[22] Medium - Vitor Rodrigues. *Métricas de Avaliação: acurácia, precisão, recall... quais as diferenças?* URL: https://medium.com/@vitorborbarodrigues/. Acesso em: 6 dez. 2020.