Fatec São José dos Campos
Prof. Jessen Vidal

CENTRO PAULA SOUZA

GOVERNO DO ESTADO SÃO PAULO

Trabalho Prático - EpidemiWeb

Arthur Barbero;
Felippe Alves;
Gabriel Landim;
José Vinicius Santana;
Thyago Odorico.

Inteligência Artificial - Análise e Desenvolvimento de Sistemas

Prof. Me. José Walmir G. Duque. – 6º semestre – 2021

Sumário

1. Introdução 3
2. Business Understanding 4
2.1. Objetivo 4
2.2. Funcionalidades e Requisitos do projeto 5
2.3. Fases do projeto 6
3. Data Understanding 7
4. Data Preparation 9
4.1. Tarefas de Machine Learning empregadas 9
4.2. Preparação dos dados 12
4.2.1. Importação de Dados 12
4.2.2. Leitura e limpeza de dados 12
4.2.3. Enriquecimento dos dados 13
5. Modeling e Evaluation 14
5.1. Modeling 14
5.2. Evaluation 17
5.2.1. Avaliação de aprendizados não-supervisionados 17
5.2.2. Resultados obtidos 18
5.2.3. Conclusões acerca das avaliações 19
6. Glossário 21
7. Referências Bibliográficas 21

1. Introdução

2019 foi um ano que irá ficar nos arautos da história, principalmente pela grande pandemia de uma nova doença com origem nos países asiáticos, uma evolução da SARS, denominada COVID-19.

A ideia de passar novamente por uma epidemia assusta qualquer pessoa nos dias atuais, e olhando as ações que foram tomadas, a única forma de minimizar os problemas e causas é a rápida identificação e comunicação, ou seja, informação.

Considerada como o novo petróleo, a informação é tida como o grande recurso dos dias atuais, e não são poucos os tipos de softwares que tentam acessar e coletar nossos dados dia após dia na esperança de vincular nossos dados ao consumo ou estilo de vida, para que as empresas possam oferecer seus serviços ou despontar à frente de seus concorrentes. Com essa visão, a informação também pode ser usada para o bem das sociedades, basta que consigamos reter informações dos usuários de forma inteligente, concisa e transparente.

Desta maneira, utilizaremos formas de capturar informações, retê-las e utilizá-las de forma a identificar novas doenças, padrões de relacionamento entre doenças-pacientes e doenças-sintomas, padrões de localidade e gerar insights de possíveis novas epidemias.

A Inteligência Artificial entra neste cenário auxiliando o projeto como um todo e se beneficiando de seus dados que serão coletados ao longo de seu uso visando a implementação de modelos treinados à uma aplicação WEB.

2. Business Understanding

2.1. Objetivo

É de conhecimento geral que o setor da saúde, na maioria dos países, é majoritariamente privado e de acesso a poucos. No Brasil e em outros poucos países como Reino Unido, Canadá, Dinamarca, Suécia, Espanha entre outros, o acesso a saúde, de forma unificada e universal, é um direito do cidadão. Restringindo o escopo para o território brasileiro, podemos identificar que o SUS (Sistema Único de Saúde) é o grande catalizador de informações relacionadas a ocorrência de doenças e mantêm histórico que pode ser acessado em qualquer Estado brasileiro. Segundo o IBGE em 2019, 71,5% da população brasileira dependem exclusivamente do SUS para diversos tratamentos.

Pensando no grande fluxo de dados públicos que poderíamos utilizar e na transparência de dados disponibilizados ao público, seria de fácil acesso para que pudéssemos traçar ligações entre doenças e sintomas aos locais de ocorrência, ou até com informações socioeconômicas, porém, infelizmente os dados que podemos acessar são escassos, e por mais que tenhamos acesso as estas informações do Portal da Saúde, só podemos visualizar dados de estatísticas vitais ou de mortalidade, algumas doenças epidemiológicas são citadas mas não existe a possibilidade de verificar novas doenças ou possíveis epidemias pela falta de padrão e informações mais qualificadas como localização, datas exatas, gênero, idade e etc ...

Com este objetivo, coletando as informações de forma padronizada e com máximo de aproveitamento, utilizaremos a Inteligência Artificial para criar modelos que realizam diagnósticos preliminares com base nos sintomas informados pelo usuário.

2.2. Funcionalidades e Requisitos do projeto

Conforme o proposto, os principais atores do projeto são:

Papel	Descrição	Nível de acesso
Agente da saúde	Maior responsável ao acesso e inserção de informações pertinentes aos cidadãos que são atendidos no estabelecimento da saúde como também pelas informações de incidência da doença e sintomas descritos pelos mesmos.	Tático
Cidadão	Usuário que deseja o acesso as informações coletadas e disponibilizadas com transparência, dentro das normas previstas em lei, podendo consultar os dados e fazer uso dos modelos treinados para identificar possíveis epidemias em sua região como a realização de diagnósticos preliminares de doenças a partir dos sintomas informados	Operacional

Dado os principais atores do projeto, para que seja possível a conclusão do projeto, o mesmo deverá atender os seguintes requisitos:

Ator	Tarefa	Regra de negócio
Agente da saúde	Cadastrar Doença	R1 – A doença deverá conter os campos “Nome” e “Data de criação”;
Agente da saúde	Cadastrar Sintoma	R1 – O sintoma deverá conter os campos “Nome”, “Descrição”, “Severidade” de 1 a 5 e “Data de criação”; R2 – Ao criar um sintoma, o mesmo deve ser relacionado ao menos a uma doença já cadastrada;
Agente da saúde	Atribuir Sintoma	R1 – A partir dos sintomas existentes, o agente poderá relacionar um sintoma a outra doença existente;
Agente da saúde	Cadastrar Incidência	R1 – No atendimento de um paciente, sua incidência deve ser criada para registro do acontecimento de uma nova doença; R2 – Para a criação da incidência, deverá conter relação com uma doença já existente e a um usuário já existente; R3 – Também serão necessários o preenchimento dos campos “Data da incidência” e “Data de criação”;

Agente da saúde	Cadastrar Usuário	R1 – Será necessário o preenchimento dos campos “Nome completo”, “E-mail”, “Senha”, “Endereço”, “Número do endereço”, “Bairro”, “Cidade”, “Estado”, “País”, “Data de criação”; R2 – Usuários com permissão de “Agente da saúde” poderão escolher também qual o perfil do usuário que está sendo criado.
Qualquer usuário	Diagnóstico Preliminar	R1 – Com a escolha dos sintomas já existentes e sua respectiva severidade, de 1 a 5, demonstrar a doença que preliminarmente se aproxima do caso apontado. R2 – Demonstrar também a porcentagem de proximidade de cada sintoma escolhido com a ocorrência da doença.

O projeto também deverá ser de fácil implementação e utilização, podendo ser acessado via WEB e também via integração nos padrões de uma aplicação REST.

2.3. Fases do projeto

- **Entendimento dos dados**
  Após o entendimento do negócio, necessitamos entender a disposição dos dados que iremos coletar e realizar as relações necessárias entre doença, incidências, sintomas e usuários, para que assim possamos construir um banco de dados na qual o projeto possa acessar para criar seus modelos e treina-los;

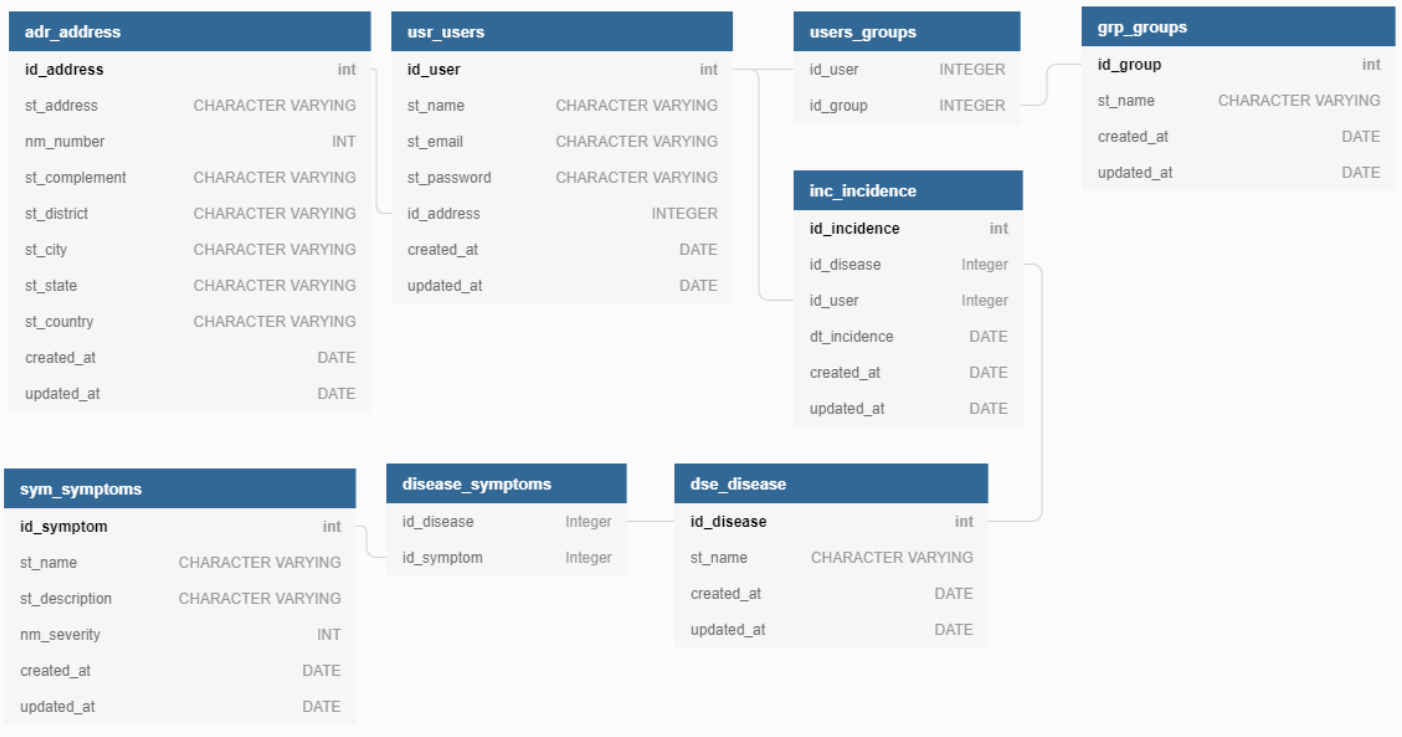
- **Confecção de Bancos de Dados**
  Com base nos requisitos e atores, criar as entidades e relacionamentos entre tabelas para que o projeto seja suportado por uma padronização e concentração de dados;

- **Criação dos modelos e rotas**
  Realizar a criação dos métodos que irão realizar o treinamento e iniciação de modelos como também as rotas que irão ser consumidas para a entrega dos resultados via API.

3. Data Understanding

Para atingirmos os objetivos do projeto, faz-se necessário um entendimento mais aprofundado dos dados, as disposições de seus relacionamentos e a identificação entre subgrupos, qualidade, relevância entre outros.

A criação de um banco de dados para o recebimento dos dados coletados do uso da ferramenta foi concebido no seguinte diagrama de entidade relacionamento:



Armazenamos os dados dos usuários entre 3 tabelas, sendo elas “usr_users” que possui a informação de nome, e-mail e senha, utilizados para entrar na plataforma, uma relação um para um com a tabela “adr_address” que armazena todas as informações de endereço do usuário e a tabela “grp_groups” que armazena as permissões do usuário.

Já os dados da regra de negócio são armazenadas entre as tabelas “dse_disease” que possuí o nome da doença, com relação muitos para muitos com a tabela “sym_symptoms”, que armazena os sintomas e sua severidade sobre a doença relacionada, também possuímos a tabela “inc_incidence” que armazena as incidências das doenças, relacionando uma doença à um usuário.

Com o entendimento das relações entre as informações, necessitamos agora entender como o projeto irá se utilizar destes para conseguir realizar os diagnósticos preliminares. Entre os dados possuímos as seguintes variáveis:

- Symptom (Sintoma):
  O sintoma é um atributo nominal, que representa um sintoma do paciente;

- Severity (Severidade):
  Atributo numérico que representa a severidade de um sintoma sobre uma doença em específico;

- Disease (Doença):
  De tipo nominal, o atributo da doença representa o diagnóstico na qual o paciente pode estar sendo acometido.

Tendo em vista que o especialista no domínio é a pessoa mais qualificada para realizar uma avaliação de peso entre o sintoma e a doença, seu input decorrente das ocorrências ao longo da vida do projeto irão alimentar o "dataset" com mais informações a cada incidência, conseguindo assim fazer predições preliminares das doenças a partir dos sintomas e pesos listados.

Exemplo da disposição dos dados:

Symptom_id	Disease_id	Severity	Symptom_desc	Disease_desc
1	163	2	Dor abdominal superior	Inflamação da colicistite da vesícula biliar
4	20	1	Abuso de álcool	Intoxicação por álcool etanol
5	732	1	Ansiedade (nervosismo)	Estresse
6	729	2	Dor ou aflição no braço	Tensão muscular puxando músculo

Utilizando uma abordagem de Machine Learning do tipo "Não-Supervisionado", teremos como variável de objetivo o atributo "Disease_desc" informando ao paciente o possível diagnóstico.

4. Data Preparation

Dado as informações sobre o “dataset”, necessitamos preparar os dados utilizando técnicas de “Data Mining” que mais se adequam ao problema principal do projeto, o diagnóstico preliminar de doenças dada seus sintomas.

4.1. Tarefas de Machine Learning empregadas

As possíveis tarefas a serem utilizadas nesse processo são:

4.1.1. Associação

A Associação visa identificar as relações dentre os atributos dos dados e geralmente apresentam a forma “SE *atributo* X ENTÃO *atributo* Y”. é uma das tarefas mais conhecidas no mercado devido aos ótimos resultados no varejo, onde é sugerido itens compatíveis com uma lista de compras ou carrinho de compras que o usuário já está inclinado a realizar a compra.

Em nosso projeto utilizamos a associação em cima de uma regra estatística denominada **Okapi BM25 (Best Match 25)** similar a TFIDF (Term Frequency, Inverse Document Frequency), ela determina a frequência em que um atributo ou termo aparece dentro de um documento ou coleção de dados. Esta associação servirá como peso para nossa segunda tarefa de “Data Mining”.

$$\mathrm{BM25}(D,Q) = \sum_{i=1}^{n} IDF(q_i, D) \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i) + k_1 \cdot (1 - b + b \cdot |D|/d_{avg}))}$$

A fórmula acima demonstra a forma algébrica de como o Okapi BM25 é capaz de dar valor a uma sentença, termo ou variável de acordo com o documento ou matriz de informações.

Resumidamente, a base do BM25 é a somatória de um valor de relevância entre o termo pesquisado e o todo do documento ou matriz de informações, levando em consideração a saturação desse termo e o limite de aparições dentre o tamanho médio documento ou matriz de informações.

Esta fórmula pode ser lida da seguinte maneira, “O valor de um termo **Q** contido em um documento **D**, é a somatória da **Frequência Inversa no Documento** (IDF – Inverse Document Frequency) de cada parte do termo **q_i** dentro do documento **D** pela divisão da **frequência do termo no documento *$f(q_i, D)$*** multiplicado pelo índice de saturação geral de termos **k1**, pela **frequência do termo *$f(q_i)$*** somado pelo índice de saturação geral de termos **k1** multiplicado pelo resultado do limite **b** multiplicado pelo tamanho médio do documento **d_{avg}**”

Onde:

- **D** = Documento como um todo;
- **Q** = Variável/Termo a ser pesada diante do todo;

- **IDF** = Inverse Document Frequency (Frequência inversa à aparição no documento);
- **k_1** = Porcentagem de saturação do termo dentro do texto;
- **b** = Bounds (Limites de relevância da aparição de um termo);
- **d_{avg}** = Tamanho médio do Documento.

O **BM25** é famoso por sua utilização dentro de textos, um exemplo de sua utilização dado um texto seria:

```
corpus = [
    'Human machine interface for lab abc computer applications',
    'A survey of user opinion of computer system response time',
    'The EPS user interface management system',
    'System and human system engineering testing of EPS',
    'Relation of user perceived response time to error measurement',
    'The generation of random binary unordered trees',
    'The intersection graph of paths in trees',
    'Graph minors IV Widths of trees and well quasi ordering',
    'Graph minors A survey'
]
```

Sua valoração, dado o termo "*The intersection of graph survey and trees"*, seria:

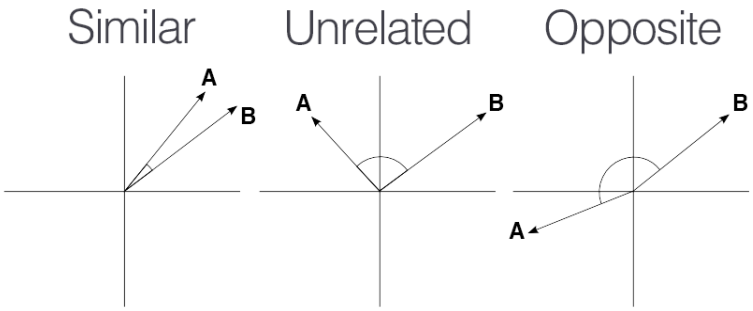
```
0.0     Human machine interface for lab abc computer applications
1.025   A survey of user opinion of computer system response time
0.0     The EPS user interface management system
0.0     System and human system engineering testing of EPS
0.0     Relation of user perceived response time to error measurement
1.462   The generation of random binary unordered trees
2.485   The intersection graph of paths in trees
2.161   Graph minors IV Widths of trees and well quasi ordering
2.507   Graph minors A survey
```

4.1.2. Agrupamento (Clustering)

O agrupamento é uma tarefa de "Data Mining" que visa identificar e aproximar os registros similares entre si. Dentre as várias técnicas utilizadas para avaliar as similaridades entre os atributos, para este projeto utilizaremos a "Cosine Similarity" (Similaridade por cosseno), onde dentre um intervalo fechado [-1,1], é medida à similaridade entre dois vetores em uma matriz de dados.

A "Cosine Similarity" é um método da biblioteca "scikit-learn" que dada duas matrizes de dados, realiza a similaridade entre suas linhas e colunas baseado em seus valores, e, dentro de uma lógica trigonométrica, retorna uma matriz única de relacionamento entre as partes.

O modelo trigonométrico pode ser representado da seguinte forma:



Um exemplo da similaridade pode ser demonstrada a seguir:

Dado duas matrizes, todas as doenças e todos os sintomas:

```
all_diseases.head()
```

	Disease_id	Disease_desc
0	1	Abdominal aortic aneurysm (enlarged major bloo...
1	2	Abdominal swelling
2	3	Abdominal trauma
3	4	Abrasions (scrapes)
4	5	ACE inhibitor induced cough blood pressure med...

```
all_symptoms.head()
```

	Symptom_id	Symptom_desc
0	1	Upper abdominal pain
1	2	Lower abdominal pain
2	3	Abscess (Collection of pus)
3	4	Alcohol abuse
4	5	Anxiety (Nervousness)

A similaridade entre eles é representada entre doenças (linha) e sintomas (colunas):

```
Disease_df = pd.DataFrame(cosine_similarity(Ur,VTr.T), columns=symptom_count.index, index=list(sum_disease.index))
Disease_df.head()
```

symptom	1	2	3	4	5	6	7	8	9	10	...	290	291	292	
1	0.205484	0.272114	0.043167	-0.047387	-0.021459	0.015953	0.633309	-0.019115	-0.073327	-0.012857	...	-0.029264	-0.128676	-0.001867	-0.0
2	-0.046659	-0.078299	-0.071982	0.063161	0.045799	-0.004549	-0.168562	0.049388	0.049649	-0.014734	...	0.074478	0.079732	0.024173	0.0
3	0.114272	0.381978	0.005659	0.049680	-0.043052	-0.014820	0.091332	-0.013405	-0.059609	0.009341	...	-0.033005	-0.009946	-0.020345	0.0
4	0.017050	0.022067	-0.036084	0.040988	0.001289	0.038791	-0.050136	-0.021242	-0.006880	-0.012500	...	0.009720	-0.006165	-0.050009	-0.0
5	-0.034572	-0.030762	-0.051101	-0.086641	-0.025164	0.005332	-0.028281	0.025686	-0.041621	-0.038750	...	-0.017533	0.007516	0.002488	-0.0

Em decorrência do “dataset” utilizado, optamos por estas duas tarefas de “Data Mining” por se tratar de aprendizados não-supervisionados e também pela utilização dos pesos e da incidência dos sintomas e das doenças.

Conforme mais incidências ocorrerem, a atribuição de pesos pelo algoritmo BM25 irá se atualizando e sua similaridade, atribuída pelo algoritmo de “Cosine Similarity”, identificará de forma mais assertiva quais doenças estão mais próximas aos sintomas informados.

4.2. Preparação dos dados

4.2.1. Importação de Dados

Para a conclusão do projeto trabalharemos com Python na versão 3.6.9, com as seguintes bibliotecas:

- **Pandas** – Biblioteca que oferece estruturas e operações para manipular tabelas numéricas e séries temporais;
- **Numpy** – Biblioteca que oferece funções matemáticas no domínio na álgebra linear;
- **Scipy** – Biblioteca com uma coleção de algoritmos matemáticos e funções convenientes criadas como extensões ao Numpy;
- **SkLearn** ou **Scikit-Learn** – Biblioteca com ferramentas eficientes para análise de dados preditivos contendo vários métodos para Machine Learning.

```
import numpy as np
import pandas as pd
from scipy.sparse import coo_matrix
from scipy.sparse.linalg import svds
from sklearn.metrics.pairwise import cosine_similarity
```

4.2.2. Leitura e limpeza de dados

Após as importações, realizamos a leitura dos dados via Pandas e já realizamos uma prévia exclusão de colunas que não precisamos utilizar nos próximos passos.

Utilizamos somente os dados dos Id's de cada coluna, além de substituir os nomes das colunas para "Symptom", "Disease" e "Severities" e eliminar a primeira linha que simboliza o cabeçalho.

```
data = pd.read_csv(filename,
                   separator,
                   usecols=[0,1,2],
                   names=['symptom', 'disease','severities'],
                   skiprows=1)

data=data.dropna()
```

Além de realizar a prévia exclusão de alguns dados, também forçamos a limpeza de qualquer dado que venha a ser classificado como "NaN" (Not a Number).

4.2.3. Enriquecimento dos dados

Para que os dados sejam utilizados da melhor forma, realizamos primeiramente a mudança dos tipos das colunas “Symptom” e “Disease”, originalmente como inteiros, para o tipo “Category” que cuidará de manter os dados como únicos, salvando memória, permitindo usar mais funções estatísticas e de ordenação.

```
data['symptom'] = data['symptom'].astype("category")
data['disease'] = data['disease'].astype("category")
```

Também será necessário a conversão desta matriz em uma matriz esparsa. Matrizes esparsas são matrizes que transformam vetores em dados pontuais, podendo fazer seu uso de acordo com sua posição dentro da matriz, salvando tempo e processamento da máquina ao evitar de realizar filtros e loopings para requisitar dados.

Matriz comum:

```
matriz_comum = pd.DataFrame({
    "filtro_1": list("abcde"),
    "filtro_2": ["azul", "amarelo", "verde", "vermelho", "laranja"],
    "valor": [10,20,30,40,50]})
matriz_comum
```

	filtro_1	filtro_2	valor
0	a	azul	10
1	b	amarelo	20
2	c	verde	30
3	d	vermelho	40
4	e	laranja	50

Matriz esparsa:

```
matriz_comum["filtro_1"] = matriz_comum["filtro_1"].astype("category")
matriz_comum["filtro_2"] = matriz_comum["filtro_2"].astype("category")

matriz_esparsa = coo_matrix((matriz_comum["valor"].astype(float),
                             (matriz_comum["filtro_1"].cat.codes.copy(),
                              matriz_comum["filtro_2"].cat.codes.copy())))
matriz_esparsa.toarray()
```

```
array([[ 0., 10.,  0.,  0.,  0.],
       [20.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0., 30.,  0.],
       [ 0.,  0.,  0.,  0., 40.],
       [ 0.,  0., 50.,  0.,  0.]])
```

Com estes dados preparados, estamos prontos para a utilização dos algoritmos de Associação, **BM25**, e de Agrupamento, **Cosine_similarity**, :

- "**Data**":

	symptom	disease	sev
0	1	163	2
1	1	164	2
2	1	165	1
3	1	187	2
4	1	306	2

- "**Matrix esparsa**":

```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 1., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

5. Modeling e Evaluation

5.1. Modeling

Com todos os dados e métodos definidos, esta seção apresentará a aplicabilidade dos algoritmos salientados nos capítulos anteriores para a construção de um modelo útil para a predição de possíveis doenças através dos sintomas informados pelo usuário.

Na construção de um modelo de dados não-supervisionados, conforme a professora e coordenadora da PUC-Rio, Cientista de dados Tatiana Escovedo:

> *"[...] No aprendizado não-supervisionado, não existe a informação dos rótulos históricos, ou seja, não temos as saídas desejadas a serem estimadas e, por este motivo, dizemos que nossos dados são não-rotulados. Assim, o algoritmo não recebe durante o treinamento os resultados esperados, devendo descobrir por si só, por meio da exploração dos dados, os possíveis relacionamentos entre eles. Neste caso, o processo de aprendizado busca identificar regularidades entre os dados a fim de agrupá-los ou organizá-los em função das similaridades que apresentam entre si. Como não temos dados rotulados, não há necessidade de realizar particionamento em conjuntos de treino e teste."*

Sendo assim, nosso modelo passa por duas tarefas importantes que são a Associação e o Agrupamento, em cada uma delas utilizamos algoritmos distintos, ranqueamento pelo Okapi BM25 e Similaridade de Cossenos respectivamente.

Para o BM25, o ranqueamento acontece, resumidamente, através da análise de um todo, que podemos chamar de “Documento”, e a partir de cada parte intrínseca de si mesmo é atribuído pesos sobre o “Documento” como um todo. Porém, ao contrário do TFIDF, existem dois parâmetros que são levados em consideração neste ranqueamento:

Para determinarmos os valores de “**k**” e “**b**”, utilizamos algumas referências no assunto retiradas de um artigo da empresa Elastic, responsável pelo mecanismo de busca e análise de dados distribuídos, Elasticsearch.

- **Limite de relevância da aparição de um termo = b :**

  Estudos indicam que precisamos levar a seguinte questão em mente, “*Quando pensamos que um ‘Documento’ provavelmente será muito longo e quando isso deve prejudicar sua relevância para um termo?*”. Quando um documento aparenta ser mais específico, como uma especificação de engenharia, em detrimento do tamanho e relevância, “b” tende a ser mais apropriado se mais próximo de 0. Já no caso contrário, quando documentos tendem a abordar vários tópicos como um ranqueamento de palavras dentro de artigos de jornal, faz mais sentido trazer o valor de “b” para mais próximo de 1.

  A aplicação de um limite faz mais sentido quando estamos utilizando o algoritmo para o ranqueamento de textos, mas no nosso caso, sua aplicação será a combinação de números que simbolizam respectivamente uma doença, sintoma e grau de relevância. Sendo assim, realizamos o uso mais apropriado conforme estudos da Elastic, predefinindo o valor de “b” para 0.3.

- **Porcentagem de saturação do termo dentro do texto = k :**

  Para definição da porcentagem de saturação de um termo, precisamos fazer o seguinte questionamento, “Quando achamos que um termo provavelmente ficará saturado?”, da mesma forma que a variável “b”, para documentos muito extensos como livros, especialmente livros de ficção ou que abordam muitos tópicos, é provável que tenhamos muitos termos diferentes sendo utilizados varias vezes no “Documento” como um todo, para estes casos, é sugerido valores mais elevados para “k”, e proporcionalmente, caso o “Documento” seja algo de escala menor, é sugerido valores menores para “k”.

Segundo o estudo da empresa, “k” é geralmente avaliado entre um range de 0 a 3, sendo assim, utilizamos um nível de saturação com 1 de valor como padrão, principalmente por utilizarmos do fator “Repetição” da combinação entre doenças e sintomas como um importante atributo para ranqueamento, assim não desejamos que uma combinação seja desconsiderada em seu ranqueamento.

BM25(data, K=1, B=0.3)

Já para o algoritmo de Similaridade por Cosseno (Cosine Similarity), este possuí menos parâmetros de configuração, sendo necessários somente os dados a serem analisados e um parâmetro indicado se a saída será uma matriz densa ou esparsa, por padrão a matriz densa é a escolhida.

cosine_similarity(X, Y, dense_output=True)

O resultado da combinação destes algoritmos é um modelo que pode ser utilizado para trazer, diante dos padrões de ocorrência dos sintomas e doenças, uma lista de possíveis doenças relacionadas ao sintoma escolhido

```
symptoms_matrix = pd.read_csv('archives/all_symptoms.csv', ';')
disease_matrix = pd.read_csv('archives/all_disease.csv', ';')

modelo = Diagnostico_preliminar(b=0.3, k=1, symptoms = symptoms_matrix, disease = disease_matrix)
modelo.fit('archives/dataset.csv', ';')

modelo.get_diseases_by_symptoms(35)
```

```
Sintoma  Foot pain
Top 10 prováveis doênças relacionadas
--------------------------------------
621.0 ['Puncture wound']
1097.0 ['Flat feet pes planus']
582.0 ['Plantar wart human papilloma virus infection, foot warts']
1098.0 ['Heel spur']
947.0 ['Foot fracture broken foot']
1287.0 ['Foot sprain']
963.0 ["Sever's disease calcaneal apophysitis, common cause of heel pain"]
581.0 ['Plantar fasciitis inflammation of tissue at the bottom of the foot']
124.0 ['Calluses and corns']
323.0 ['Bunion of big toe hallux valgus']
```

5.2. Evaluation

5.2.1. Avaliação de aprendizados não-supervisionados

Após a criação e exemplificação dos modelos de Inteligência Artificial, devemos verificar a precisão sistemática destes modelos, para assegurar que estão coerentes com seus objetivos.

Para nosso projeto, o objetivo da Inteligência Artificial é encontrar padrões entre os sintomas e doenças, e com base na severidade realizar o ranqueamento e similaridades para buscar, de acordo com o sintoma escolhido, a doença que mais se destaca nos parâmetros acima.

Tendo o objetivo em mente, realizamos algumas avaliações em cima dos dados que possuímos:

- **LogisticRegression:**

A regressão logística é uma técnica estatística que tem como objetivo produzir, a partir de um conjunto de observações, um modelo que permita a predição de valores tomados por uma variável categórica, frequentemente binária, a partir de uma série de variáveis explicativas contínuas e/ou binárias. LogisticRegression é um método da biblioteca Scikit-learn.

- **XGBClassifier:**

O XGBClassifier é um método da biblioteca XGBoost que implementa vários modelos e entre eles um algoritmo de classificação e medição que utilizamos para produzir nossas análises.

5.2.2. Resultados obtidos

Para a execução dos algoritmos de avaliação, realizamos os seguintes passos:

Primeiramente realizamos as importações necessárias para a utilização destes algoritmos em uma função que realizará a validação dos valores do modelo.

```
from sklearn.linear_model import LogisticRegression
from xgboost import XGBClassifier

from sklearn.model_selection import cross_val_predict
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import log_loss, accuracy_score

import scikitplot.plotters as skplt
```

```
def evaluate(X, y, clf=None):
    probas = cross_val_predict(clf, X, y, cv=StratifiedKFold(n_splits=5, random_state=8, shuffle=True),
                               n_jobs=-1, method='predict_proba', verbose=2)

    pred_indices = np.argmax(probas, axis=1)
    classes = np.unique(y)
    preds = classes[pred_indices]
    print('Log loss: {}'.format(log_loss(y, probas)))
    print('Accuracy: {}'.format(accuracy_score(y, preds)))
    skplt.plot_confusion_matrix(y, preds)
```

A função “evaluate” receberá três variáveis como parâmetro, sendo eles, “X” o “DataFrame” resultado da aplicação do modelo, ranqueado e assimilado, “y” sendo uma lista de valores considerados corretos para a realização da avaliação e “clf” sendo o algoritmo de avaliação a ser utilizado.

Para conseguirmos prosseguir com a avaliação preparamos os parâmetros conforme supracitado.

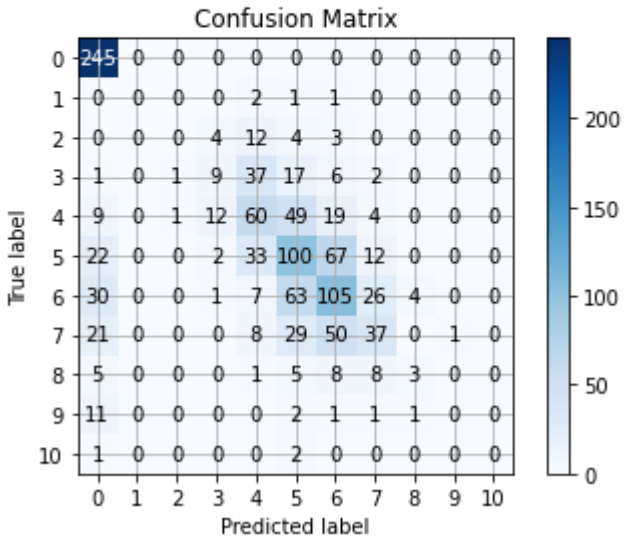
```
modelo = preliminary_diagnosis(k=1, b=0.3, symptoms = all_symptoms, diseases = all_diseases)
X = modelo.fit('archives/dataset.csv', ';')

weight = bm25_weight(matrix)
weight_df = pd.DataFrame(weight.toarray())
y = weight_df.max(axis=1).astype('int').values
```

Assim estamos prontos para realizar avaliações de cada algoritmo.

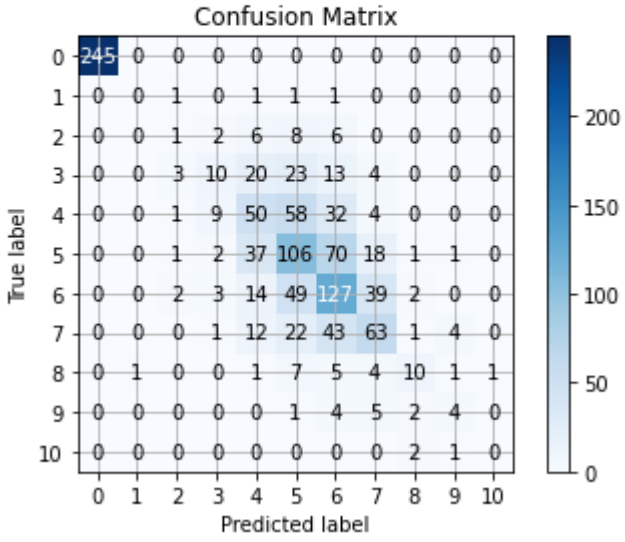
```
evaluate(X, y, clf=LogisticRegression())
```

```
Log loss: 1.6032106051839263
Accuracy: 0.4794168096054888
```




```
evaluate(X, y, clf=XGBClassifier())
```

```
Log loss: 1.5938048736609824
Accuracy: 0.5283018867924528
```



5.2.3. Conclusões acerca das avaliações

Podemos perceber que nas avaliações realizadas, temos uma acurácia de cerca de 50% dos ranqueamentos e similaridades ocorridos em cima dos nossos dados de sintomas e doenças.

Porém, identificamos que para aprendizados não-supervisionados, na maioria dos casos estudados, após a utilização dos algoritmos de ranqueamento e similaridade, o resultado encontrado pode ou não apresentar resultados favoráveis para seu objetivo, simplesmente por se tratar de aprendizagem não-supervisionada, ou seja, uma vez que não existem rótulos que indiquem a exata classificação de um item ou posição em um ranqueamento, sendo este trabalho dado pelas função algorítmicas, sua avaliação depende muito de pessoas que entendam sobre o objetivo final ou outros conjuntos de dados já previamente rotulados que sirvam como base para verificarmos a acuracidade do modelo.

Como em nosso projeto utilizamos dados muito específicos, conjuntos de sintomas-doenças relacionadas a um nível de severidade, não podemos afirmar com certeza das análises realizadas acima.

Os dados que contribuiriam como validadores, registrados pela variável “y” na função “evaluate”, são basicamente as possíveis classificações realizadas pelo algoritmo BM25 que vão de 0 a 10, conforme pode ser visto nos gráficos de confusão.

Porém, em testes empíricos é possível analisar que existe um ranqueamento entre os dados imputados conforme suas ocorrências. No âmbito do projeto, conforme os dados são coletados por agentes da saúde, muitos sintomas e doenças são registrados no DataFrame e conforme os algoritmos de ranqueamento e similaridade, podemos concluir que, empiricamente, o objetivo do modelo está sendo realizado.

Um exemplo pode ser obtido quando realizamos a execução do modelo com os dados atuais:

```
all_symptoms = pd.read_csv('archives/all_symptoms.csv', ';')
all_diseases = pd.read_csv('archives/all_disease.csv', ';')

modelo = preliminary_diagnosis(k=1, b=0.3, symptoms = all_symptoms, diseases = all_diseases)
modelo.fit('archives/dataset.csv', ';')
modelo.get_diseases_by_symptoms(35)
```

```
Sintoma  Foot pain
Top 10 prováveis doênças relacionadas
--------------------------------------
621 ['Puncture wound']
1097 ['Flat feet pes planus']
582 ['Plantar wart human papilloma virus infection, foot warts']
1098 ['Heel spur']
947 ['Foot fracture broken foot']
1287 ['Foot sprain']
963 ["Sever's disease calcaneal apophysitis, common cause of heel pain"]
581 ['Plantar fasciitis inflammation of tissue at the bottom of the foot']
124 ['Calluses and corns']
323 ['Bunion of big toe hallux valgus']
```

Podemos notar que ao incluir mais agumas incidências que relacionam o sintoma escolhido “Foot pain” com a doença “Heel spur”, o mesmo sobe no ranqueamento:

```
all_symptoms = pd.read_csv('archives/all_symptoms.csv', ';')
all_diseases = pd.read_csv('archives/all_disease.csv', ';')

modelo = preliminary_diagnosis(k=1, b=0.3, symptoms = all_symptoms, diseases = all_diseases)
modelo.fit('archives/Incidencias.csv', ';')
modelo.get_diseases_by_symptoms(35)
```

```
Sintoma  Foot pain
Top 10 prováveis doênças relacionadas
--------------------------------------
1098 ['Heel spur']
582 ['Plantar wart human papilloma virus infection, foot warts']
1097 ['Flat feet pes planus']
621 ['Puncture wound']
963 ["Sever's disease calcaneal apophysitis, common cause of heel pain"]
581 ['Plantar fasciitis inflammation of tissue at the bottom of the foot']
947 ['Foot fracture broken foot']
1287 ['Foot sprain']
324 ['Hammer toes toe deformity']
323 ['Bunion of big toe hallux valgus']
```

6. Glossário

- Dataset – Coleção de dados;

- Machine Learning – Aprendizado de máquina;

- Data Mining – Mineração de dados;

- DataFrame – Conjunto de dados tabulados em duas ou mais dimensões.

7. Referências Bibliográficas

"Symptom Disease sorting" Dataset utilizado, Kaggle (https://www.kaggle.com/plarmuseau/sdsort) último acesso em 18/04/2021;

"Mineração de Dados Utilizando Aprendizado Não-Supervisionado: Um estudo de caso para bancos de dados da saúde" Campos Serra Domingues, Miriam Lúcia (https://www.lume.ufrgs.br/bitstream/handle/10183/2702/000375416.pdf?sequence=1) último acesso em 18/04/2021;

"Okapi BM25: a non-binary model", 2008 Cambridge University Press (https://nlp.stanford.edu/IR-book/html/htmledition/okapi-bm25-a-non-binary-model-1.html) último acesso em 18/04/2021;

"Cosine Similarity – Understanding the math and how it works (with python codes)", Selva Prabhakaran (https://www.machinelearningplus.com/nlp/cosine-similarity/) último acesso em 18/04/2021;

"Machine Learning: Conceitos e Modelos – Parte II: Aprendizado Não-Supervisionado", Tatiana Escovedo (https://tatianaesc.medium.com/machine-learning-conceitos-e-modelos-parte-ii-aprendizado-n%C3%A3o-supervisionado-fb6d83e4a520) último acesso em 15/05/2021;

"Practical BM25 - Part 3: Considerations for Picking b and k1 in Elasticsearch" Shane Connelly – Elastic.co (https://www.elastic.co/pt/blog/practical-bm25-part-3-considerations-for-picking-b-and-k1-in-elasticsearch) último acesso em 15/05/2021.