**TECHNICAL REPORT**

| Aluno: Paulo Henrique Santos Marques | | |
| --- | --- | --- |

1. **Introdução**

**Predict Diabetes**

*O dataset em questão é voltado para a análise de fatores de risco e características associadas ao desenvolvimento de diabetes em indivíduos. Ele contém informações detalhadas sobre várias métricas de saúde e demográficas, incluindo o número de gestações (Pregnancies), níveis de glicose no sangue (Glucose), pressão arterial (BloodPressure), espessura da pele (SkinThickness), níveis de insulina (Insulin), índice de massa corporal (BMI), função de pedigree de diabetes (DiabetesPedigreeFunction), idade (Age) e o resultado final (Outcome), onde o valor 1 indica a presença de diabetes e 0 indica a ausência.*

*O dataset apresenta algumas colunas que se trata de identificadores únicos de registros, o que não interessa para a gente. Vamos considerar todas as características que se trata de features relacionadas a doença importantes para prosseguirmos com as análises.*

**CyberBullying Detection**

*O segundo dataset é voltado para a análise e detecção de cyberbullying em comentários de redes sociais, com foco na identificação de discursos de ódio, conteúdo ofensivo e comentários normais. O dataset é cuidadosamente anotado por vários revisores, garantindo a precisão e a confiabilidade das classificações.*

*Ele contém duas partes principais: o arquivo hateXplain.csv e o final\_hateXplain.csv. No hateXplain.csv, cada post é identificado por um post\_id e é acompanhado por anotações feitas por diferentes revisores (annotators). Essas anotações incluem o rótulo atribuído ao post, que pode ser classificado como discurso de ódio (Hatespeech), ofensivo (Offensive), ou normal (Normal). Além disso, as anotações incluem a comunidade alvo identificada no post e as "rationales", que são trechos do texto que justificam a classificação dada pelos revisores.*

*O arquivo final\_hateXplain.csv categoriza os rótulos de acordo com diferentes aspectos, como raça (ex.: african, asian, caucasian), religião (ex.: christian, islam), gênero (ex.: men, women), orientação sexual (ex.: bisexual, homosexual), e outras categorias diversas (ex.: disability, refugee).*

*O objetivo ao analisar este dataset é aplicar classificadores e análises que nos levem a obter insights importantes sobre o dataset.*

1. **Observações**

*Nem sempre os dados seguem uma estrutura ideal para algumas das análises requeridas nas questões. Isso faz com que alguns erros ocorram, ou uma acurácia que se não tiver bem processada pode chegar a ser baixa.*

1. **Resultados e discussão**

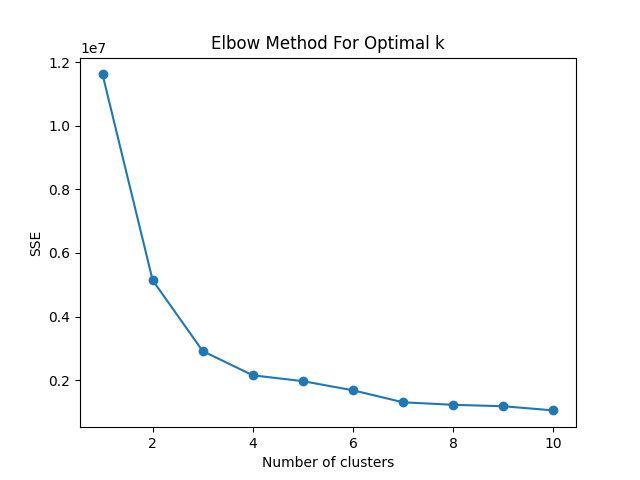
***Questão 1 -***

O código em questão apresenta a seguinte lógica:  
O código implementa uma solução para o algoritmo K-means que atende à questão exigida, encapsulando a lógica dentro de uma classe Kmeans. Ele inicializa centroids aleatoriamente, atribui rótulos aos dados com base na proximidade aos centroids, recalcula os centroids como a média dos pontos atribuídos, e repete esse processo até que a convergência seja alcançada. Para determinar o número ideal de clusters, o código utiliza o método do cotovelo, garantindo que a quantidade de clusters seja escolhida sem depender de um rótulo de target específico, atendendo assim a todas as exigências da questão.

**Resultados:**

Após a Classe atender aos requisitos de generalização, inicializar com número de centroid e aplicar a lógica do Kmeans, um dos métodos escolhidos para retornar a quantidade de Cluster ideal, foi o método do cotovelo, como mostra a figura 1.

*Figure 1 -*



Com base na análise do gráfico é possível ver que a quantidade de Cluster ideal para nosso modelo é 3, visto que é o ponto em que a discrepância de decrescimento mais tem uma diminuição. Após esse resultado foi feito a predição e os resultados dos clusters foram armazenados na coluna label.

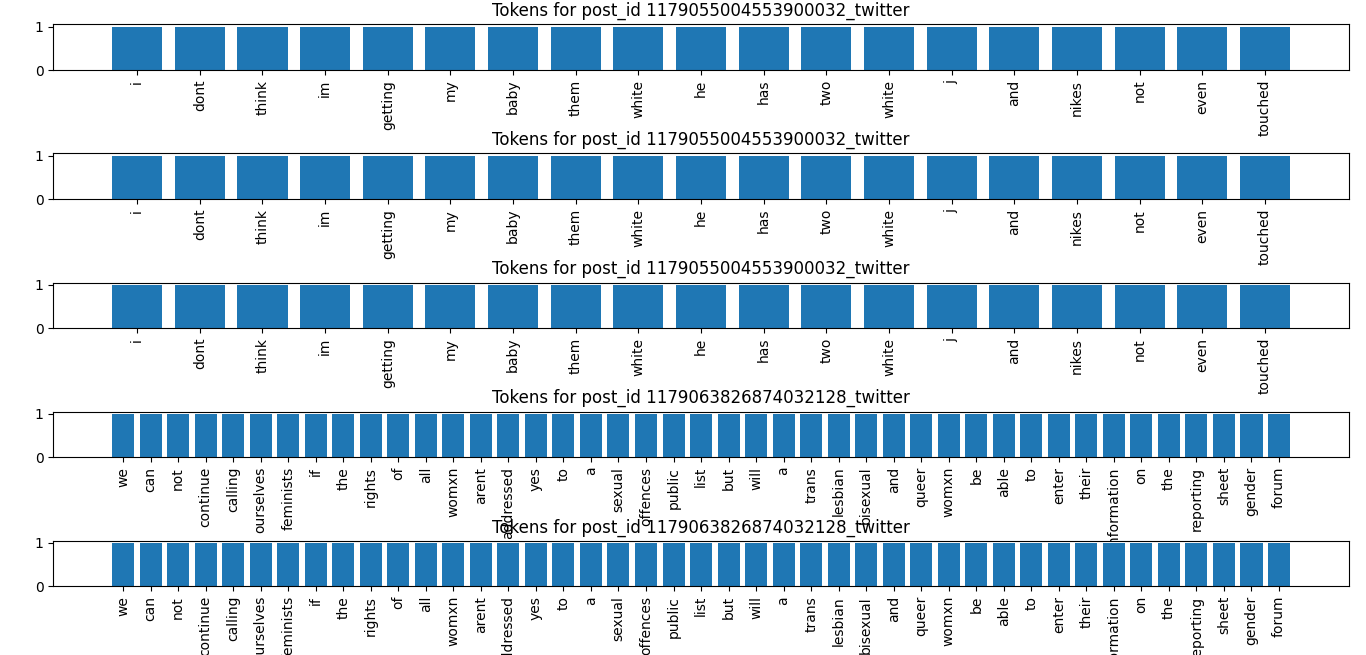
**Questão 2 -**

Para responder a segunda questão foi utilizado técnicas de mlp para realizar o pré processamento e a tokenização do texto em colunas consideradas importantes. O código atende à questão de forma completa ao realizar o pré-processamento necessário (remoção de caracteres desnecessários e tokenização) e ao fornecer uma função reutilizável que pode ser aplicada em outras análises. Além disso, a visualização dos cinco primeiros tokens em subplots permite uma análise comparativa clara e organizada, facilitando a interpretação das características textuais dos dados processados.

**Resultados:**

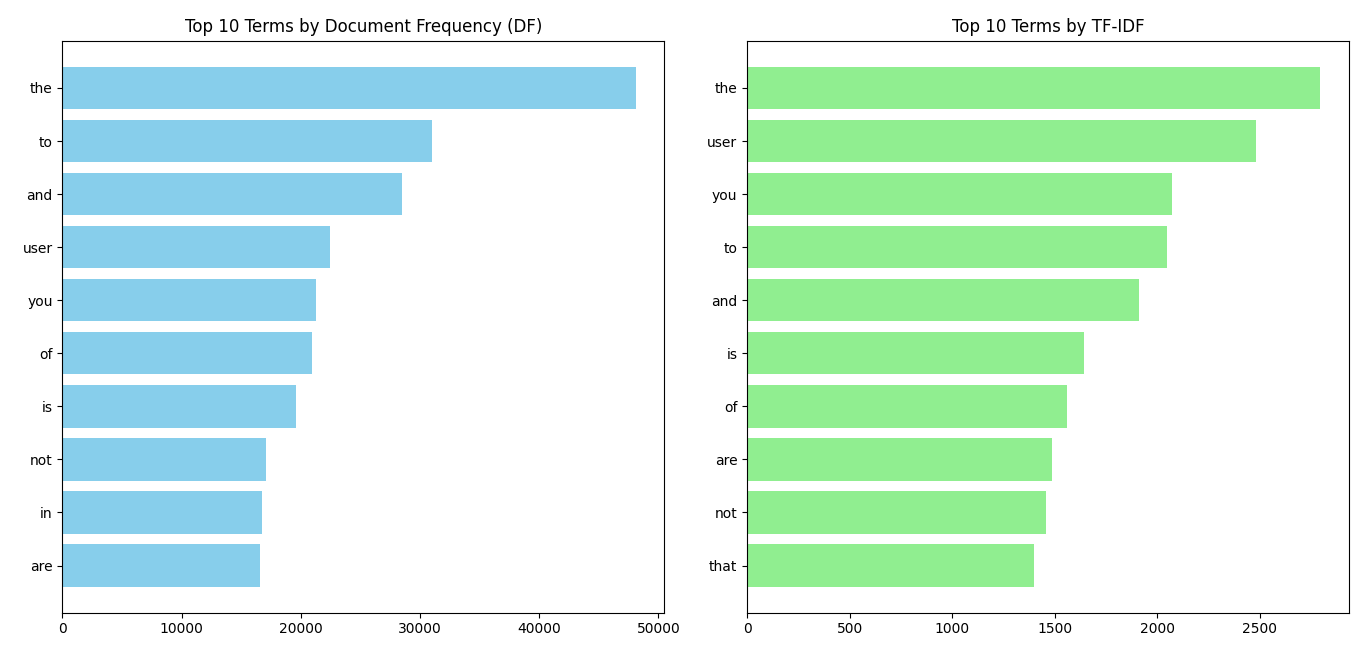
Como resultado é possível ver a distribuição das 5 primeiras listas de tokens gerados. A estratégia abordada para gerar os gráficos foi a mesma para todos, enfatizando que todos os gráficos estão agrupados em uma só figura através de subplots. Como mostra a figura 2.

**Figure 2 -**



**Questão 3 -**

A terceira questão é um pouco mais simples, dado que precisamos apenas importar a função utilizada na questão anterior e gerar dois atributos baseado no DF e o outro baseado no TFIDF e plotar e analisar os 10 maiores termos de ambos. Feito a importação dos métodos da questão anterior e a criação dos atributos numéricos, o que resta como resultado é a plotagem da figura 3.  
**Figure 3 -**



*Pontos Importantes:*

**Document Frequency (DF)**: No gráfico à esquerda, os termos estão classificados de acordo com a sua frequência de aparição em documentos (DF). Isso significa que os termos mais à esquerda (como "the", "to", "and") aparecem em mais documentos dentro do corpus analisado, independentemente de sua importância contextual. Esses termos são geralmente comuns e não necessariamente informativos.

**TF-IDF (Term Frequency-Inverse Document Frequency)**: No gráfico à direita, os termos estão classificados com base em seus valores de TF-IDF. TF-IDF é uma métrica que considera não apenas a frequência de um termo em um documento específico, mas também a frequência inversa desse termo em todo o corpus. Isso significa que termos comuns em muitos documentos terão um peso menor, enquanto termos que são frequentes em poucos documentos, mas raros no corpus geral, terão um peso maior. No gráfico, termos como "the", "user", "you" têm valores altos de TF-IDF, indicando que esses termos são significativos em determinados contextos, mesmo que possam ser comuns.

**Questão 4 -**

A questão solicita a aplicação de um algoritmo de classificação utilizando as funções implementadas nas questões anteriores para extrair atributos baseados em Document Frequency (DF) e TF-IDF, e comparar o desempenho entre essas duas abordagens.

### Passos Implementados no Código:

1. **Importação e Pré-processamento dos Dados:**

O código carrega o dataset hateXplain.csv e aplica a função preprocess\_data (importada da questão 2) para realizar o pré-processamento e a tokenização dos textos.

As labels dos dados são codificadas utilizando LabelEncoder para que possam ser usadas na classificação.

1. **Divisão do Conjunto de Dados:**

O dataset é dividido em conjuntos de treino e teste usando train\_test\_split, garantindo que 20% dos dados sejam reservados para teste, enquanto os 80% restantes são utilizados para treinar o modelo.

1. **Função para Treinar e Avaliar o Modelo:**

A função train\_and\_evaluate\_model recebe os dados de treino e teste, além de um vetor de transformação (CountVectorizer para DF ou TfidfVectorizer para TF-IDF), e então:

Cria um pipeline que primeiro vetoriza o texto usando o vetorizador apropriado e, em seguida, aplica o algoritmo de classificação Naive Bayes Multinomial (MultinomialNB).

O modelo é treinado com os dados de treino e, em seguida, usado para fazer previsões no conjunto de teste.

O desempenho é avaliado usando o classification\_report do scikit-learn, que fornece métricas como precisão, recall e f1-score para cada classe.

1. **Comparação dos Desempenhos:**

O código avalia o desempenho do modelo utilizando CountVectorizer (para DF) e TfidfVectorizer (para TF-IDF) e imprime os relatórios de classificação para ambas as abordagens, permitindo uma comparação direta entre as duas técnicas de extração de atributos.

**Resultados:**

*Desempenho usando DF (CountVectorizer):*

***Table 1 -***

|  | *precision* | *recall* | *f1-score* | *support* |
| --- | --- | --- | --- | --- |
| *hate speech* | *0.66* | *0.70* | *0.68* | *3620* |
| *normal* | *0.70* | *0.73* | *0.71* | *4930* |
| *offensive* | *0.51* | *0.45* | *0.48* | *3539* |
|  |  |  |  |  |
| *accuracy* |  |  | *0.64* | *12089* |
| *macro avg* | *0.62* | *0.63* | *0.62* | *12089* |
| *weighted avg* | *0.63* | *0.64* | *0.64* | *12089* |

*Desempenho usando TF-IDF (TfidfVectorizer):*

***Table 2 -***

|  | *precision* | *recall* | *f1-score* | *support* |
| --- | --- | --- | --- | --- |
| *hate speech* | *0.69* | *0.63* | *0.66* | *3620* |
| *normal* | *0.61* | *0.86* | *0.71* | *4930* |
| *offensive* | *0.56* | *0.29* | *0.38* | *3539* |
|  |  |  |  |  |
| *accuracy* |  |  | *0.62* | *12089* |
| *macro avg* | *0.62* | *0.59* | *0.58* | *12089* |
| *weighted avg* | *0.6* | *0.62* | *0.60* | *12089* |

Os resultados da classificação utilizando Document Frequency (DF) e TF-IDF mostram diferenças significativas no desempenho do modelo Naive Bayes para as diferentes classes (hatespeech, normal e offensive).

#### Desempenho usando DF (CountVectorizer) : (Table 1)

**Precisão:** O modelo obteve uma precisão geral de 0.64, com a classe "normal" alcançando a maior precisão (0.70) e a classe "offensive" a menor (0.51).

**Recall:** A classe "normal" também teve o maior recall (0.73), enquanto "offensive" teve o menor (0.45). Isso indica que o modelo foi melhor em identificar corretamente os exemplos da classe "normal".

**F1-Score:** O F1-score, que equilibra precisão e recall, foi mais alto para "normal" (0.71) e mais baixo para "offensive" (0.48), sugerindo que o modelo teve dificuldades em classificar corretamente textos ofensivos.

#### Desempenho usando TF-IDF (TfidfVectorizer):(Table 2)

**Precisão:** O uso de TF-IDF resultou em uma ligeira diminuição na precisão geral (0.62). A precisão para "hatespeech" aumentou para 0.69, enquanto a precisão para "offensive" melhorou um pouco para 0.56.

**Recall:** O recall da classe "normal" aumentou significativamente para 0.86, indicando que o modelo com TF-IDF identificou quase todos os exemplos dessa classe, mas o recall para "offensive" caiu drasticamente para 0.29.

**F1-Score:** O F1-score para "normal" permaneceu relativamente alto (0.71), mas caiu para 0.38 para "offensive", mostrando que, embora o modelo possa estar capturando mais exemplos da classe "normal", ele sacrifica a performance em capturar textos ofensivos.

1. **Conclusões**

*As questões foram respondidas com os resultados que devia. Avaliando o modelo nos deparamos com nossa melhor taxa de sucesso em 70 % para uma das classes, entre tanto ainda sim não é um modelo que poderia entrar em produção, data sua taxa de acerto não ser consideravelmente boa.*

1. **Próximos passos**

*Verificar o que falta para que o modelo alcance uma acurácia geral acima de 90 %, para que possa realmente ser desenvolvidos trabalhos com ele.*