Importando os Dados

```
#importar arquivo do drive

from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
#procurar o diretório

!ls '/content/drive/MyDrive/ADS/Inovação/racist_or_sexist_tweets.csv'
```

```
/content/drive/MyDrive/ADS/Inovação/racist_or_sexist_tweets.csv
```

Analisando os Dados

```
#Fazendo análise de dados usando as bibliotecas

import numpy as np
import pandas as pd

dados = pd.read_csv('/content/drive/MyDrive/ADS/Inovação/racist_or_sexist_tweets.csv', sep
```

```
dados.shape
```

```
(31962, 3)
```

```
dados.head()
```

	id	label	tweet
0	1	0	@user when a father is dysfunctional and is s...
1	2	0	@user @user thanks for #lyft credit i can't us...
2	3	0	bihday your majesty
3	4	0	#model i love u take with u all the time in ...

```
#Concluimos que a coluna id não influencia em nada no nosso dado, então retiramos.
dados=dados.drop(['id'],axis=1)
dados.head()
```

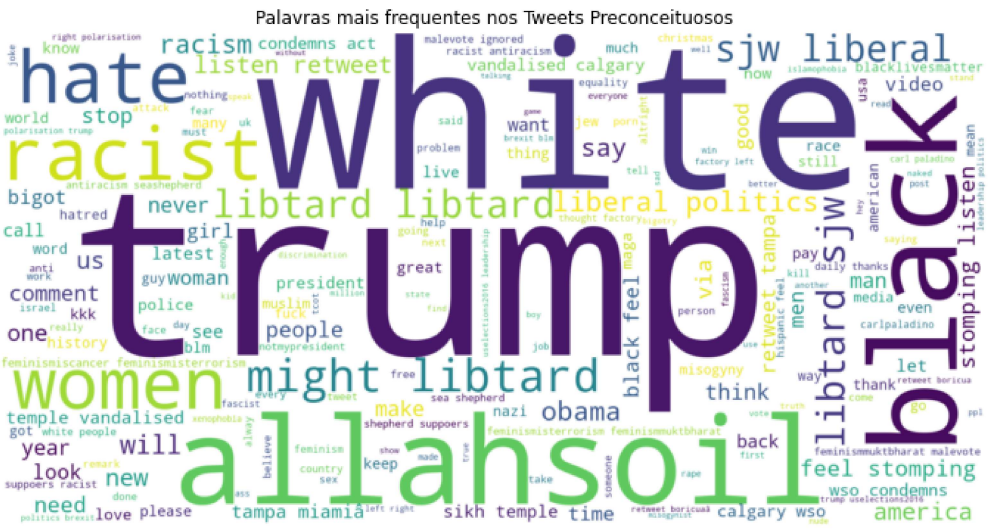
	label	tweet
0	0	@user when a father is dysfunctional and is s...
1	0	@user @user thanks for #lyft credit i can't us...
2	0	bihday your majesty
3	0	#model i love u take with u all the time in ...

```
dados.isna().sum() #verificando se o arquivo tem dados faltando
```

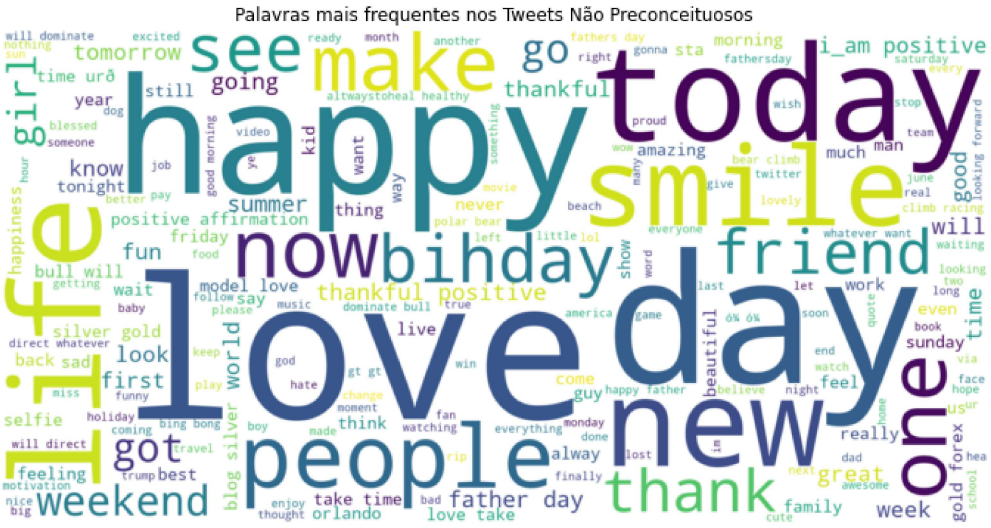
```
label    0
tweet    0
dtype: int64
```

```
#Fazendo Nuvem de tags
#1) Para as palavras mais frequentes nos tweets preconseituosos
preconceituosos = dados.loc[dados['label']==1] #
textoRac = preconceituosos.dropna(subset=['tweet'], axis=0)['tweet']
cloud1 = " ".join(s for s in textoRac)
```

```
import matplotlib.pyplot as plt
from wordcloud import WordCloud, STOPWORDS

stopwords = set(STOPWORDS)
stopwords.update(['the','it','a','as','for','and','user','amp'])
plt.figure(figsize=(12,10))
words = WordCloud(stopwords = stopwords, background_color="white", width=1600, height=800)
plt.imshow(words, interpolation="bilinear")
plt.axis('off')
plt.title("Palavras mais frequentes nos Tweets Preconceituosos");
```



```
#1) Para as palavras mais frequentes nos tweets preconseituosos

sadios = dados.loc[dados['label']==0] #Comando loc seleciona os dados dentro dos [].
#Os dados dentro do colchete são apenas os valores da coluna Label que tem valor 0, ou sej
textoNoRac = sadios.dropna(subset=['tweet'], axis=0)['tweet']  #Seleciona o a coluna de te
cloud2 = " ".join(s for s in textoNoRac)  #separa as palavras do texto em "".



import matplotlib.pyplot as plt  #Biblioteca para gráfico
from wordcloud import WordCloud, STOPWORDS   #Biblioteca para word clouds

stopwords = set(STOPWORDS)  #importando comando que retira palavras desnecessárias
stopwords.update(['the','it','a','as','for','and','user','amp'])  #algumas palavras desnec
plt.figure(figsize=(12,10)) #seleciona o tamanho da janela
words = WordCloud(stopwords = stopwords, background_color="white", width=1600, height=800)
plt.imshow(words, interpolation="bilinear") # importa o wordcloud na janela
plt.axis('off')
plt.title("Palavras mais frequentes nos Tweets Não Preconceituosos");
```



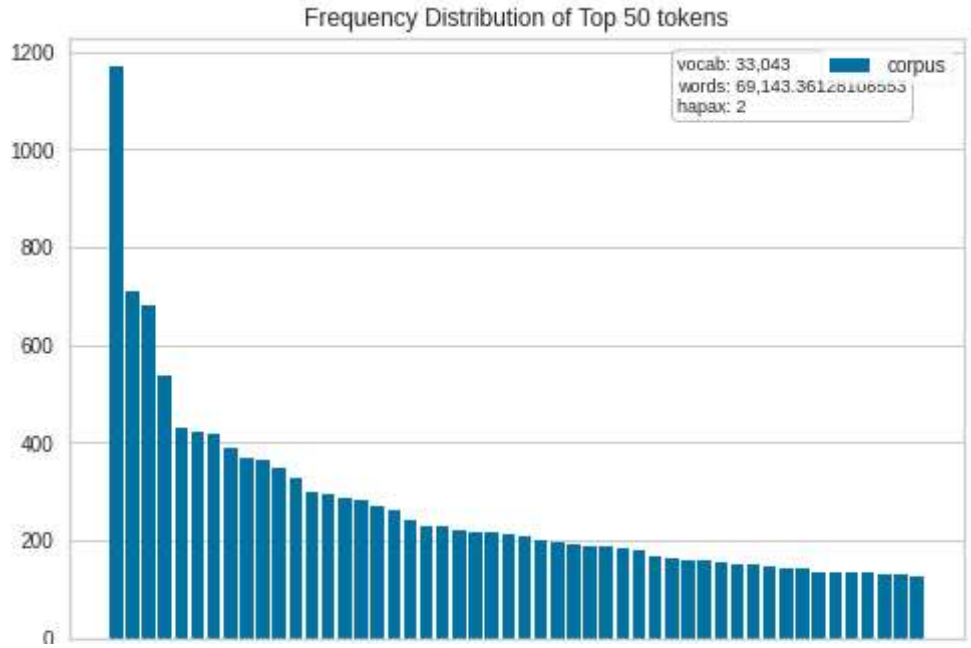
Processando os Dados

```
#Separando nosso arquivo em variável atributo (x) (as mensagens) e classe (y) ( os sentime

y = dados['label']
x = dados['tweet']

#Separando os dados em parte de treinamento e parte de teste
from sklearn.model_selection import train_test_split

x_treino, x_teste, y_treino, y_teste = train_test_split(x,y,test_size = 0.3)
```

```
#precisamos vetorizar os dados, quando trabalhamos com palavras devemos transformar em núm
from sklearn.feature_extraction.text import TfidfVectorizer

vetorizar = TfidfVectorizer(analyzer='word',ngram_range = (1,1)) # Esses comando vetoriza.

frasesVetorizadasTreino = vetorizar.fit_transform(x_treino)  #comando fit_transform faz a
frasesVetorizadasTeste = vetorizar.transform(x_teste)
```

```
#verificando as palavras mais comum nos dados
# Nessas linhas de comando, estamos verificando quasi palavras são mais frequentes em mess
from yellowbrick.text import FreqDistVisualizer

palavras = vetorizar.get_feature_names()
visualizar = FreqDistVisualizer(features= palavras, orient='v')

visualizar.fit(frasesVetorizadasTreino)
visualizar.show()
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.
  warnings.warn(msg, category=FutureWarning)
```




```
#Organizando as Classes para deixá-las de forma binárias também,elas são numéricas mas não
from sklearn.preprocessing import MultiLabelBinarizer
```

```
mlb = MultiLabelBinarizer()
rotuloTreino = mlb.fit_transform(map(str,y_treino))
rotuloTeste = mlb.fit_transform(map(str,y_teste))
```

Classificação

```
#1)
#Usando o Classificador ExtraTreesClassifier

from sklearn.metrics import classification_report
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.metrics import accuracy_score


#Criação de modelo

modelo1 = ExtraTreesClassifier()   #importando nosso classificador para variável modelo1;
modelo1.fit(frasesVetorizadasTreino,rotuloTreino)
previsoes1 = modelo1.predict(frasesVetorizadasTeste)

print(classification_report(rotuloTeste, previsoes1)) # mostra relatório
print('A acurácia é ',accuracy_score(previsoes1, rotuloTeste)) # exibe acurácia
```

```
                  precision    recall  f1-score   support

               0       0.96      1.00      0.98      8896
               1       0.90      0.51      0.65       693

       micro avg       0.96      0.96      0.96      9589
       macro avg       0.93      0.75      0.81      9589
    weighted avg       0.96      0.96      0.96      9589
     samples avg       0.96      0.96      0.96      9589

    A acurácia é  0.9601626864115131
    /usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1308: Undef
      _warn_prf(average, modifier, msg_start, len(result))
```

O classificador ExtraTreesClassifier classificou corretamente 96% dos dados. MUITO BOM!!

```
#Usando o Classificador KNeighborsClassifier
from sklearn.neighbors import KNeighborsClassifier

#Criação de modelo

modelo2 = KNeighborsClassifier()
modelo2.fit(frasesVetorizadasTreino,rotuloTreino)
previsoes2 = modelo2.predict(frasesVetorizadasTeste)
```

```
print(classification_report(rotuloTeste, previsoes2)) # mostra relatório
print('A acurácia é ',accuracy_score(previsoes2, rotuloTeste)) # exibe acurácia
```

```
                  precision    recall  f1-score   support

               0       0.95      1.00      0.97      8896
               1       0.92      0.27      0.42       693

        micro avg       0.95      0.95      0.95      9589
        macro avg       0.93      0.63      0.70      9589
     weighted avg       0.94      0.95      0.93      9589
      samples avg       0.95      0.95      0.95      9589

     A acurácia é  0.9456669100010429
```

O classificador KNeighborsClassifier classificou corretamente 94% dos dados

```
#Usando o Classificador DecisionTreeClassifier
from sklearn.tree import DecisionTreeClassifier

#Criação de modelo

modelo3 = DecisionTreeClassifier()
modelo3.fit(frasesVetorizadasTreino,rotuloTreino)
previsoes3 = modelo3.predict(frasesVetorizadasTeste)

print(classification_report(rotuloTeste, previsoes3)) # mostra relatório
print('A acurácia é ',accuracy_score(previsoes3, rotuloTeste)) # exibe acurácia
```

```
                  precision    recall  f1-score   support

               0       0.96      0.98      0.97      8896
               1       0.63      0.53      0.57       693

        micro avg       0.94      0.94      0.94      9589
        macro avg       0.80      0.75      0.77      9589
     weighted avg       0.94      0.94      0.94      9589
      samples avg       0.94      0.94      0.94      9589

     A acurácia é  0.943372614454062
```

```
#Usando o Classificador RandomForestClassifier
from sklearn.ensemble import RandomForestClassifier

#Criação de modelo

modelo4 = RandomForestClassifier()
modelo4.fit(frasesVetorizadasTreino,rotuloTreino)
previsoes4 = modelo4.predict(frasesVetorizadasTeste)

print(classification_report(rotuloTeste, previsoes4)) # mostra relatório
print('A acurácia é ',accuracy_score(previsoes4, rotuloTeste)) # exibe acurácia
```

```
                  precision    recall  f1-score   support

               0       0.96      1.00      0.98      8896
               1       0.94      0.40      0.57       693

        micro avg       0.96      0.96      0.96      9589
        macro avg       0.95      0.70      0.77      9589
     weighted avg       0.96      0.96      0.95      9589
      samples avg       0.96      0.96      0.96      9589

     A acurácia é  0.9551569506726457
     /usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1308: Undef
       _warn_prf(average, modifier, msg_start, len(result))
```

```
#Usando o Classificador RandomForestClassifier
from sklearn.multiclass import OneVsRestClassifier # algoritmos binários
from sklearn.svm import LinearSVC

#Criação de modelo

modelo5 =  OneVsRestClassifier(LinearSVC(), n_jobs=-1)
modelo5.fit(frasesVetorizadasTreino,rotuloTreino)
previsoes5 = modelo5.predict(frasesVetorizadasTeste)

print(classification_report(rotuloTeste, previsoes5)) # mostra relatório
print('A acurácia é ',accuracy_score(previsoes5, rotuloTeste)) # exibe acurácia
```

```
                  precision    recall  f1-score   support

               0       0.97      0.99      0.98      8896
               1       0.89      0.56      0.69       693

        micro avg       0.96      0.96      0.96      9589
        macro avg       0.93      0.78      0.83      9589
     weighted avg       0.96      0.96      0.96      9589
      samples avg       0.96      0.96      0.96      9589

     A acurácia é  0.9630826989258525
```

```
from sklearn.naive_bayes import MultinomialNB

#Criação de modelo

modelo6 =  OneVsRestClassifier(MultinomialNB())
modelo6.fit(frasesVetorizadasTreino,rotuloTreino)
previsoes6 = modelo6.predict(frasesVetorizadasTeste)

print(classification_report(rotuloTeste, previsoes6)) # mostra relatório
print('A acurácia é ',accuracy_score(previsoes6, rotuloTeste)) # exibe acurácia
```

```
                  precision    recall  f1-score   support

               0       0.93      1.00      0.97      8896
               1       1.00      0.09      0.17       693
```

```
micro avg       0.93      0.93      0.93      9589
macro avg       0.97      0.55      0.57      9589
weighted avg       0.94      0.93      0.91      9589
samples avg       0.93      0.93      0.93      9589

A acurácia é  0.9342997184273647
```

```
from sklearn.ensemble import GradientBoostingClassifier

#Criação de modelo

modelo7 =   OneVsRestClassifier(GradientBoostingClassifier())
modelo7.fit(frasesVetorizadasTreino,rotuloTreino)
previsoes7 = modelo7.predict(frasesVetorizadasTeste)

print(classification_report(rotuloTeste, previsoes7)) # mostra relatório
print('A acurácia é ',accuracy_score(previsoes7, rotuloTeste)) # exibe acurácia
```

```
                  precision    recall  f1-score   support

               0       0.95      1.00      0.97      8896
               1       0.90      0.26      0.41       693

   micro avg       0.94      0.94      0.94      9589
   macro avg       0.92      0.63      0.69      9589
weighted avg       0.94      0.94      0.93      9589
 samples avg       0.94      0.94      0.94      9589

A acurácia é  0.9439983314214204
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1308: Undef
  _warn_prf(average, modifier, msg_start, len(result))
```