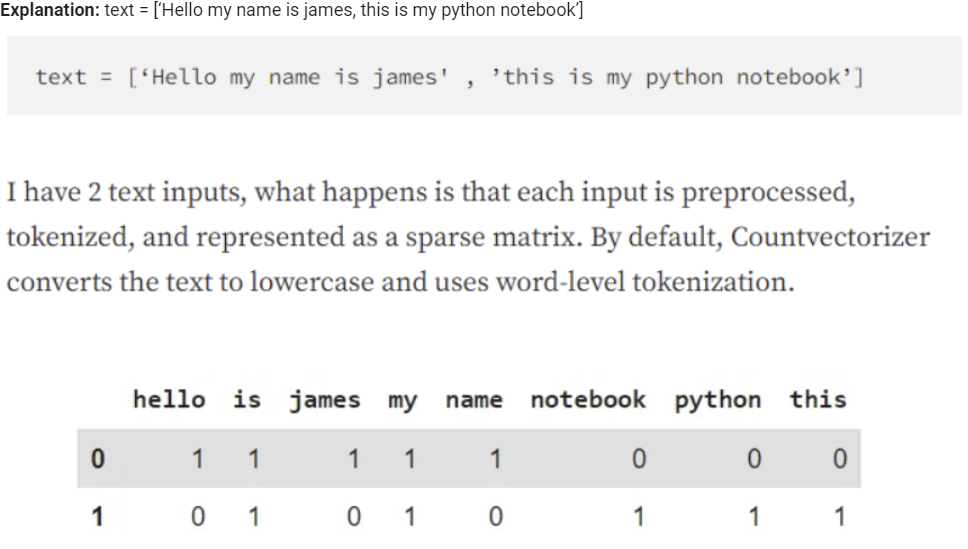
**Sentiment Analysis on Abusive Tweets**

1. Import necessary libraries
2. Import dataset (Tweet.csv)
3. Give name to columns respectively (Sentiment, Text)
4. Pre-processing dataset:
5. Defining function text\_prepoc() to do pre-processing on data i.e., removing non-ASCII characters, removing twitter handles(@), removing links, stopwords, special characters.etc
6. Creating/Defining new columns ‘clean\_text’ to store cleaned values from above function
7. Part of Speech(POS) Tagging:
8. Tokenize the ‘clean\_text’ column by using word\_tokenize function and store in variable ‘tokens’
9. Use function pos\_tag(tokens) and unpenn\_tagset(‘NN’) to do POS tagging and view the meaning of each tagset
10. Tokenization:
11. Tokenize text from ‘clean\_text’ column using df['clean\_text'].apply(lambda x: x.split())
12. Stemming:
13. Use PorterStemmer() to stem the text(tokens). By doing this holding becomes hold, committing becomes commit, etc.
14. Combine the words and join this to ‘clean\_text’ column
15. Exploratory Data Analysis:
16. Visualize the data using wordcloud. (larger words represent the most common words and the smaller words represent the less common words)
17. Define a function to extract #hashtags from ‘clean\_text’ column.
18. We have 4 types of Sentiment(01,2,3,4). We will extract #hashtags for each Sentiment type
19. Plotting frequency graph of #hashtag for each type of Sentiment
20. We got the following findings: We have #islam used more when Sentiment==0, #moham for Sentiment==1, etc;
21. We will create ‘hashes’ variable to store list of all hashtags and variable ‘sentiment\_class’ to store list of Sentiment labels (0,1,2,3,4) and empty list ‘top\_hashes’ to store top hashes for each Sentiment
22. Plot bar chart to visualize top Hashtags against Count with hue=Sentiment
23. We got following from bar chart: Where sentiment==0:  #islam, sentiment==1 : #Moham, sentiment==2 : #buildthewall, sentiment==3 : #jihad, sentiment==4 : #clinton
24. Splitting Input
25. Using CountVectorizer to generate bag of words i.e, it generates matrix of words along with its occurrence and store it in ‘bow’ variable
26. Explanation below:



1. Split the data into train and test as: bow = training & testing; df['Sentiment]=target (prediction); x\_train = 75% of bow; y\_train=75% of df['Sentiment]; x\_test= 75% of df['Sentiment']; y\_test=25% of df['Sentiment]
2. Model Training:
3. Using **Logistic Regression**(statistical method used for binary classification tasks), model.fit(x\_train, y\_train)
4. Testing data using average=macro i.e., pred = model.predict(x\_test) and calculate f1\_score(y\_test, pred, average='macro')
5. **1.The F1-score** : combines precision and recall. How well can model correctly identify TP & TN while minimizing FP & FN

**2.Accuracy:** number of correct predictions over all predictions

**3.Precision** (positive predictive value) i.e. measure of how many of the positive predictions made are correct (true positives)

**4.Recall** is a measure of how many of the positive cases the classifier correctly predicted, over all the positive cases in the data

**5. average=macro:** macro takes the average of each class’s F-1 score, it treats all classes as equal, independent of sample sizes

**6.** **average=weighted:** averaged by using the number of instances in a class as weights

1. To return the macro F1 score all we need to do is calculate the mean of the four class F1 scores
2. f1score is weighted by no. of sample/instances in each class
3. To predict per class scores of a multi-class classification problem: f1\_score(y\_test, pred, average=None) and accuracy\_score(y\_test,pred)
4. Using KNN Model. KNN predicts based on similarity, Logistic Regression on linear relationships. KNN is non-parametric, Logistic Regression is parametric. KNN's boundary is flexible, Logistic Regression's is linear. KNN stores data for prediction, Logistic Regression learns during training. KNN's prediction is slower as compared.
5. Importing import KNeighborsClassifier and seaborn as sns and plotting as sns.set\_style("darkgrid") and sns.pairplot(df, hue="clean\_text", height=3)
6. Again splitting data into x\_train, x\_test, y\_train, y\_test an training it usimg knn\_clf=KNeighborsClassifier(); knn\_clf.fit(x\_train,y\_train)
7. Predicting using ypred=knn\_clf.predict(x\_test)
8. Confusion matrix:
9. It compares the actual labels of a dataset with the predicted labels from the model, showing the number of true positives, true negatives, false positives, and false negatives
10. Conclusion: The conclusion of a sentiment analysis of abusive tweets using Multinomial Logistic Regression and KNN model depends on the specific dataset and analysis conducted. In our case, accuracy of Multinomial Logistic Regression is more as compared to KNN indicating that the data is more suited for linear modeling rather than clustering. Overall, the choice of the most suitable model for sentiment analysis of abusive tweets depends on various factors such as the characteristics of the dataset, the goals of the analysis, and the resources available.