**NATURAL LANGUAGE PROCESSING**

Task1:- **Sentiment Analysis:** Perform Sentiment Analysis on tweets that you can fetch from Twitter API or use any dataset for the same.

From my research in overview NLP can be done in 2 ways . One is using supervised learning and other methods on cleaned data. There is another whole subset of NLP which is DLNP.

Before using any machine learning model, The first step is to clean the data.

Data cleaning:-

1)Tokenisation:-Breaking down of larger components into smaller components, like paragraphs to sentences, sentences to word .

2)After breaking down into words there exists a list of non-essential words(that do not add any meaning to the sentiment or any other useful thing (generally pronouns, articles etc.) called stopwords. From the list of words we obtained after tokenisation we remove the stop words. Then we rejoin all the words with spaces in the middle and get back reduced sentences .

3)Stemming and lemmatization:-Reducing words to the crux or the word the other word was derived from is stemming.

Lemmatization is reducing the word to the most grammatically correct word ,thus leaving us with the least variety of words but then we also have all bases cleared as well .

After Cleaning of data we have to convert our cleaned data into a format where the machine can read it and then using normal classification algorithms like bayes thm. ,we can classify the sentiments.

1)BAG OF WORDS

In this technique count of words in every sentence is noted and a table kind of thing is made where sentences are rows and all words used in sentences are columns . If the word is present in a sentence It gets a one else it gets a zero in that cell .If we want we can also use the actual frequency of the word in the sentence.

2)TFIDF

As only the frequency of the word does not tell us how important a word is we use TFIDF in those cases . TFIDF gives us a specified quantified value which is the product of [TF(term frequency)=number of times a word is in a sentence/total words in a sentence] and[IDF(Inverse document frequency)=

log(number of sentences/ number of times a word is in a sentence)]

This gives us a values which also give us an idea of how important a word is in a sentence.

After learning about all this I made a spam classifier for text messages using naïve Baye’s as the classification technique on previously cleaned data.

3)HASHING Vectorization also was an option but I didn’t understand that clearly.

Both bag of words and TFIDF do not give any semantic information that is order of words is never really taken into consideration ,only the occurrence is .

To solve the above problem another form of vectorization we use word embedding implemented using deep learning.

Word representation

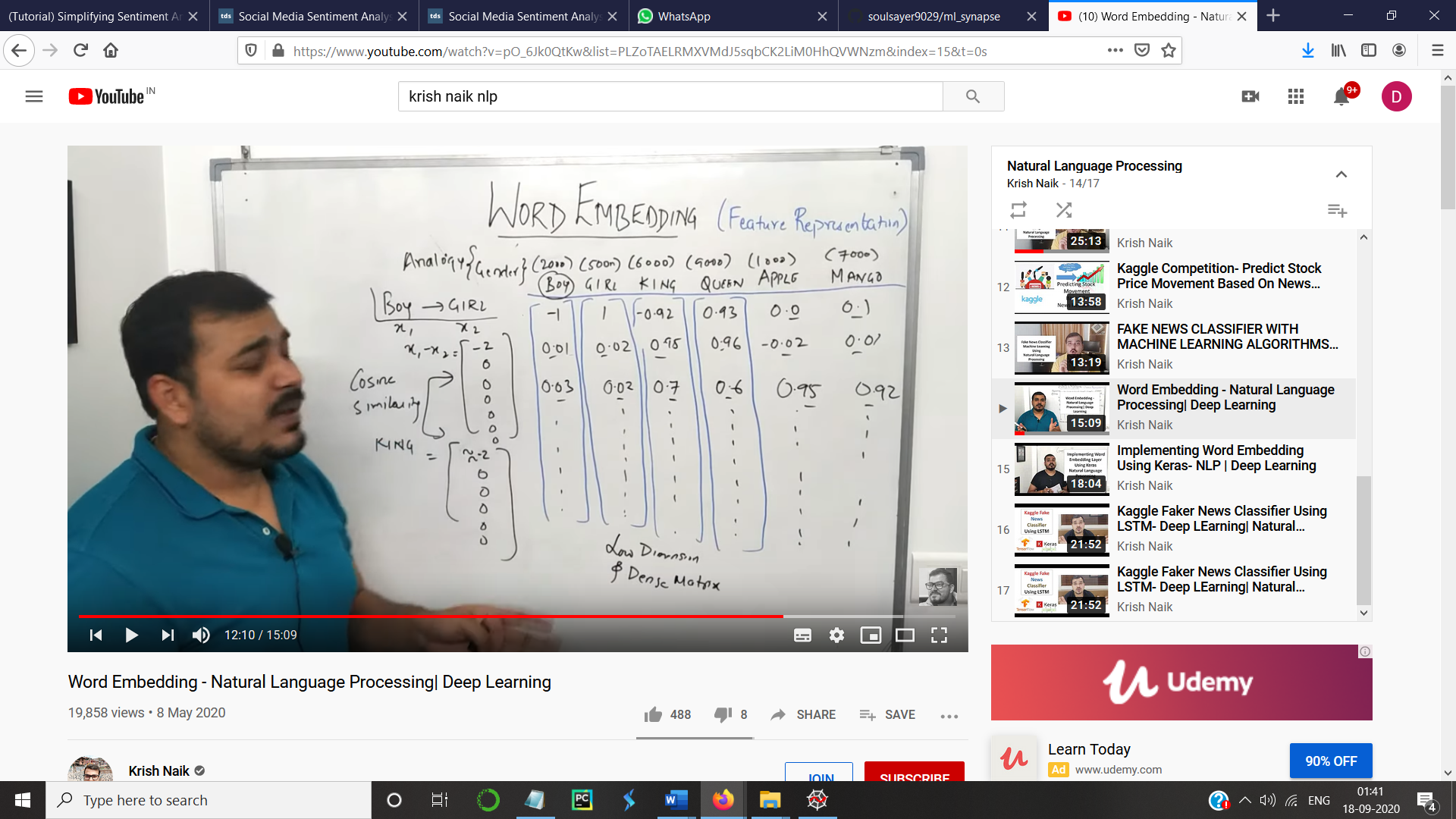
One hot representation

Consider a dictionary which contains word in alphabetical order say we have a word man. In a one hot representation of man there will be a vector containing all zeroes except one place that will be the index of man in that dictionary . Not much useful as not a lot of semantic is transferred here as well and also comparison of vectors like this very difficult. Also size will be very big as length will be size of the dictionary containing all the words,

Feature representation

All words will be given a number of features and based on the meaning of the word the feature will be rated ,thus words having similar meaning will have similar values for the features and thus will be easy to compare the two words.

Example :- analogy also explained



Gender

Royalty

Food

Also length is say if we 300 features and when well plot it converting the 300 dimensions to 2d then we’ll have all similar things like king and queen ,boy and girl, apple and mango close to each other.

Implementation

Step0:- we set a vocabulary size

Step1:-We have a few sentences. For each word we create a one hot representation for a sentence. So now we have a indexes of all the words in a sentence.

Step2:-We then pass the one hot representation through keras embedding layer, and define the number of features

1)Word2Vec

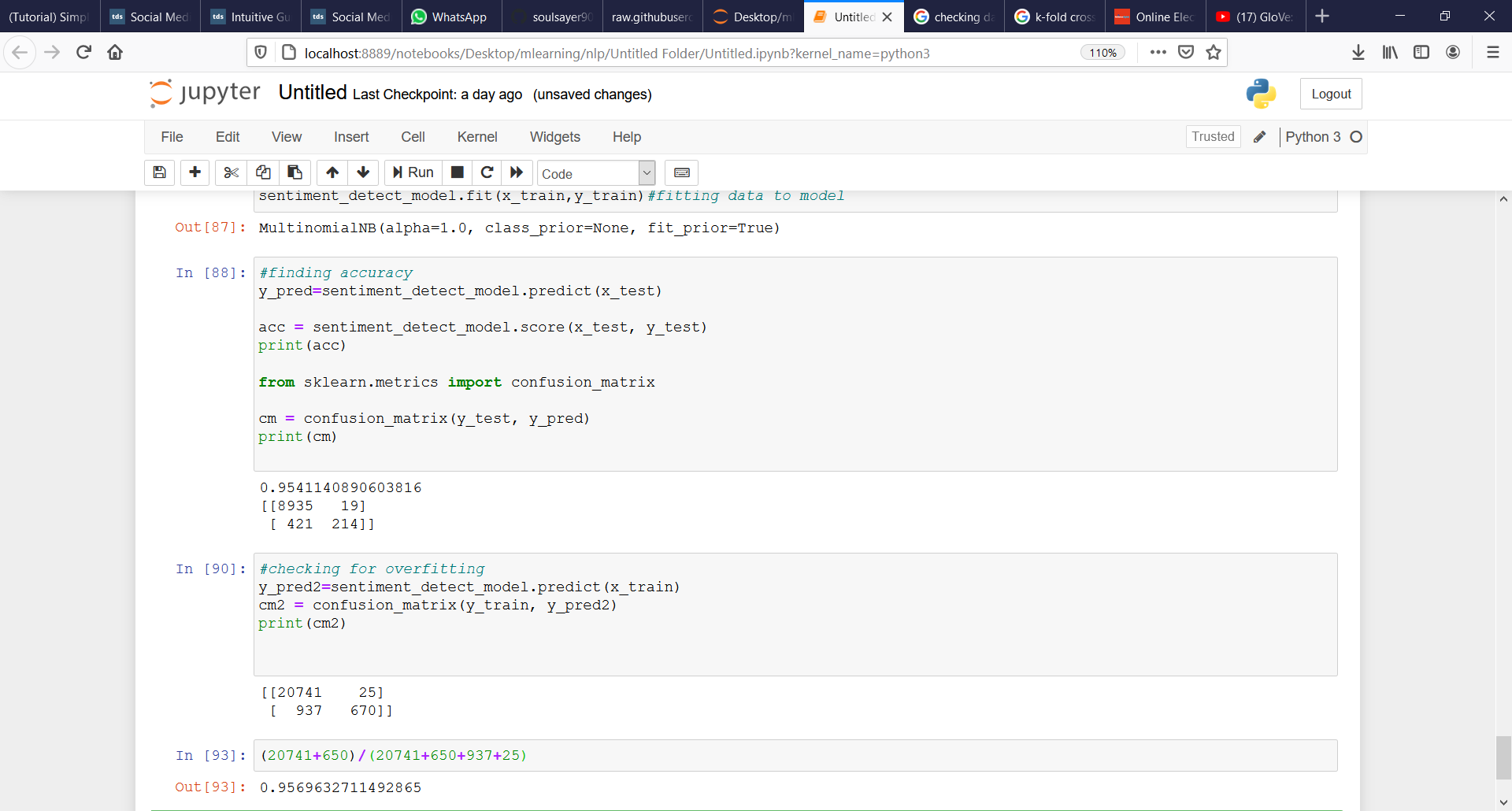
In this form of vectorization every word is converted into a vector of more than 32 dimensions(100 dimensions in python).Words that may have similar meaning like man, king will be close to each other while others will be far away.

Another method to do the same thing is the GloVe technique.

After the vectorization step we are then ready to implement the model by dividing the data into test and training set and fit the train set to our model

I used the tfidf vectorizer and the naïve bayes classifier’s multinomial naïve bayes model in my code. Word embedding techniques like word2vec are used along with algorithms involving neural networks as they are passed as the embedding layer in the neural networks .Most frequently used algorithm using neural networks is LSTM.

On using the above model that is naïve bayes I got a 95.4% accuracy on the dataset I used.



Then to check for overfitting I tried to calculate the accuracy on the training set itself and , as I got a very similar value to my accuracy thus there is no overfitting.

Confusion matrix

For analysing a classification model we use confusing matrix which gives us true positive false Positive

False negative true negative

true positive:- 1’s predicted correctly(in our case 0)

true negative;-0’s predicted correctly(in our case1)

false positive:-predicted 1’s and actual 0

false negatives:-predicted 0’s and actual 1

to analyze a confusion matrix we have recall and precision

recall=t.p/(t.p+f.n) precision=t.p/(t.p+f.p)

but comparing these two values is difficult ,so we use a new variable called

f-value (f-1 score) which is the harmonic mean of recall and precision .

this is something that we use to compare the accuracies of different models.

Higher the f-1 score better will be the model. This is used when we use multiple models together and to check all models are up to the mark .