# 

# **FAKE JOB DESCRIPTION PREDICTION (Real or Fake)**

**Team #:** 7

**Members in Team:**

Haritha B – 8

Satya Anusha- 18

Yamini Dontireddy – 28

Vamshi Jakkula – 26

**PPT Link:** <https://drive.google.com/file/d/1McNiWbEJtyYs2yQ8RkHkbExy6QHl4iXl/view>

**Video Link:** [**https://www.loom.com/share/2a8047d28a05451ebbae143b3b6a85e3**](https://www.loom.com/share/2a8047d28a05451ebbae143b3b6a85e3)

**Project Goals**

**Objective**

The main objective behind this project is to understand the problem of Employment Scam to research and academic community.

**Motivation**

Now a days fake news has evolved into a sizeable industry all over the world and 35% of news read online are fake. By this, we can prevent students and gullible job seekers from sharing their confidential information and money and also sizeable economic loss to the human society.

**Description of the problem**

According to the statistics almost 35% of the news which we get daily are almost fake. This fake news creates many problems for many naïve people. These are even into the category of job search. Many fake job descriptions are even being on the social media in order to reach out better. Job and employment is such a delicate topic where fake advertisements are misleading naive students and innocent job seekers to share their confidential information.

**Scope**

The main scope of this project is to understand few real time scenarios and predict the job as real or fake by applying few algorithms to the dataset. We have implemented 3 algorithms which include

* + Count Vectorization
  + Glove technique
  + BERT technique.

**Dataset description**

The University of the Aegean published the Employment Scam Aegean Data set. The data contains about 18K real-life job advertisements. The dataset consists of both of real and fake advertisements. This data consists of both textual and meta-information about the jobs such as company profile, logo, industry etc.

**MODELS USED**

**1.BERT(Bidirectional Encoder Representations from Transformers)**

**2.Glove Technique**

**3.Countvec technique**

**EXECUTION INSTRUCTIONS**

• We used Google Colab to run our code

• We imported all required libraries and downloaded nltk.download('stopwords'), !pip install -q transformers,!pip install -q pytorch\_pretrained\_bert and nltk.download('punkt') in order to make use of some of the functionalities.

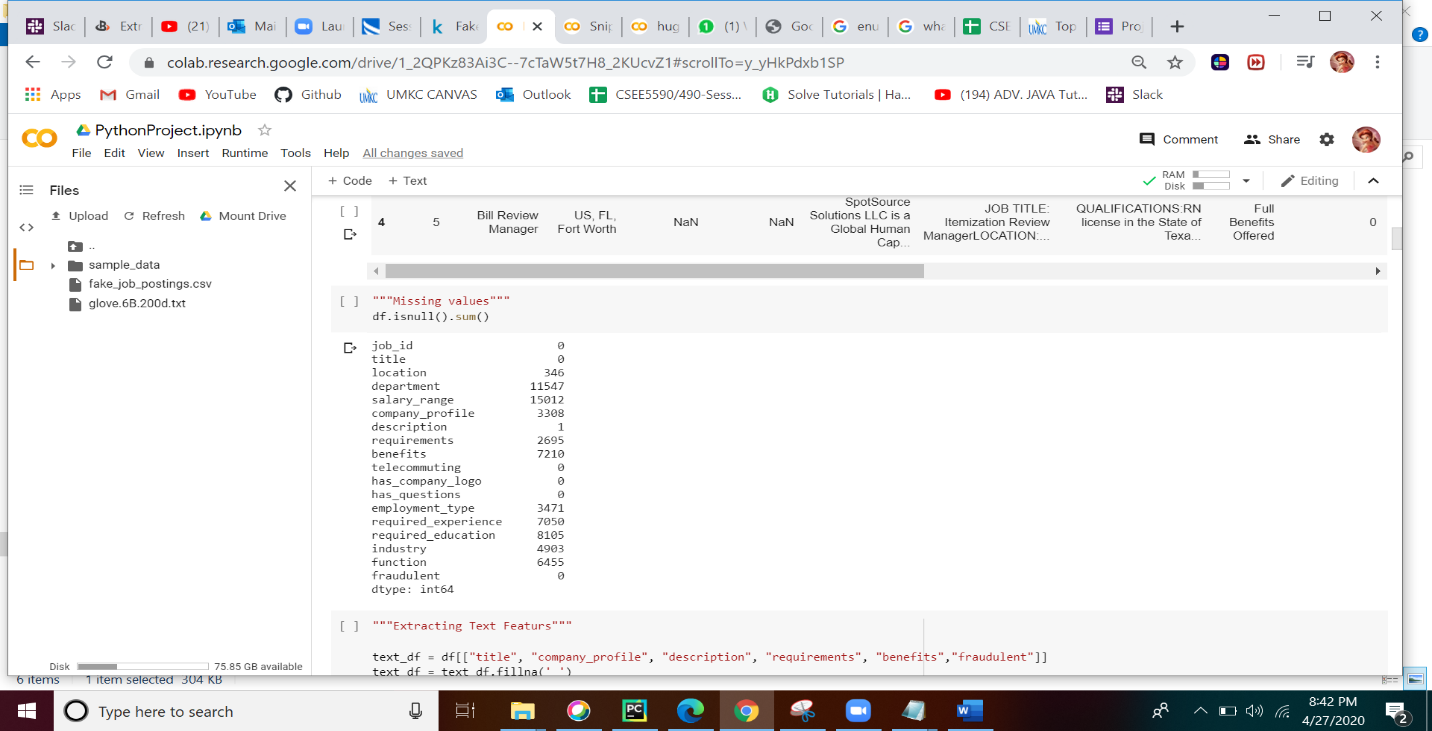
**Approach to solve the problem**

These are the three steps to solve the problem.

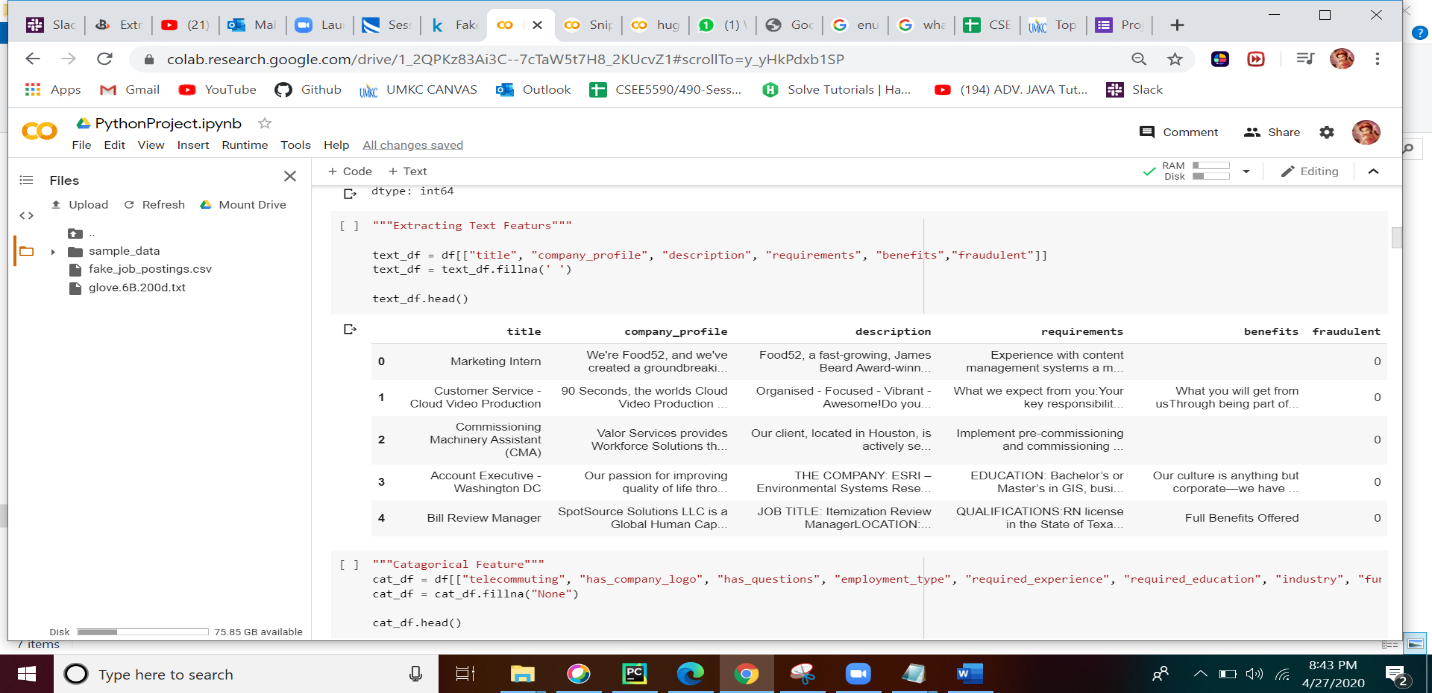
* Performing exploratory data analysis
* Data preprocessing
* Applying the dataset to various models.

**Performing Exploratory data analysis:**

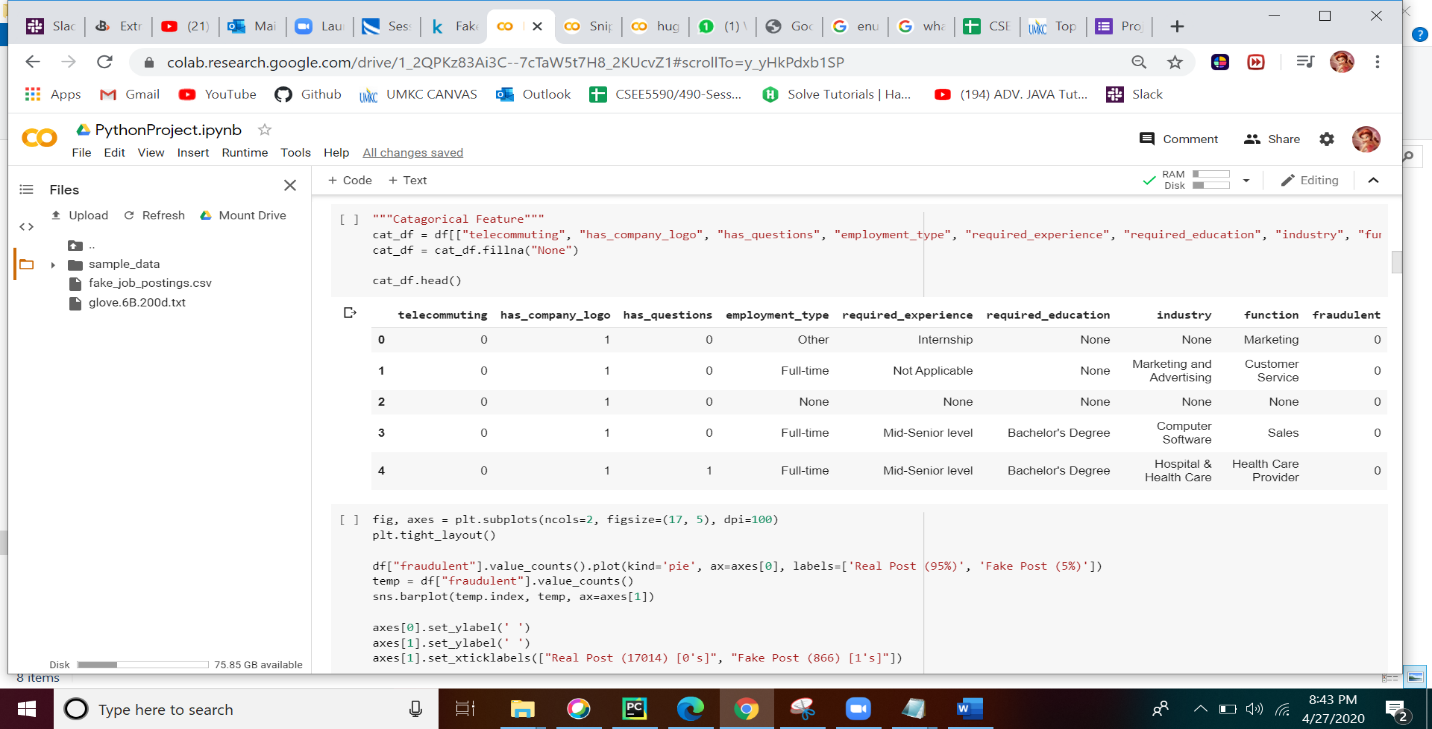
The initial step is to check for missing values.



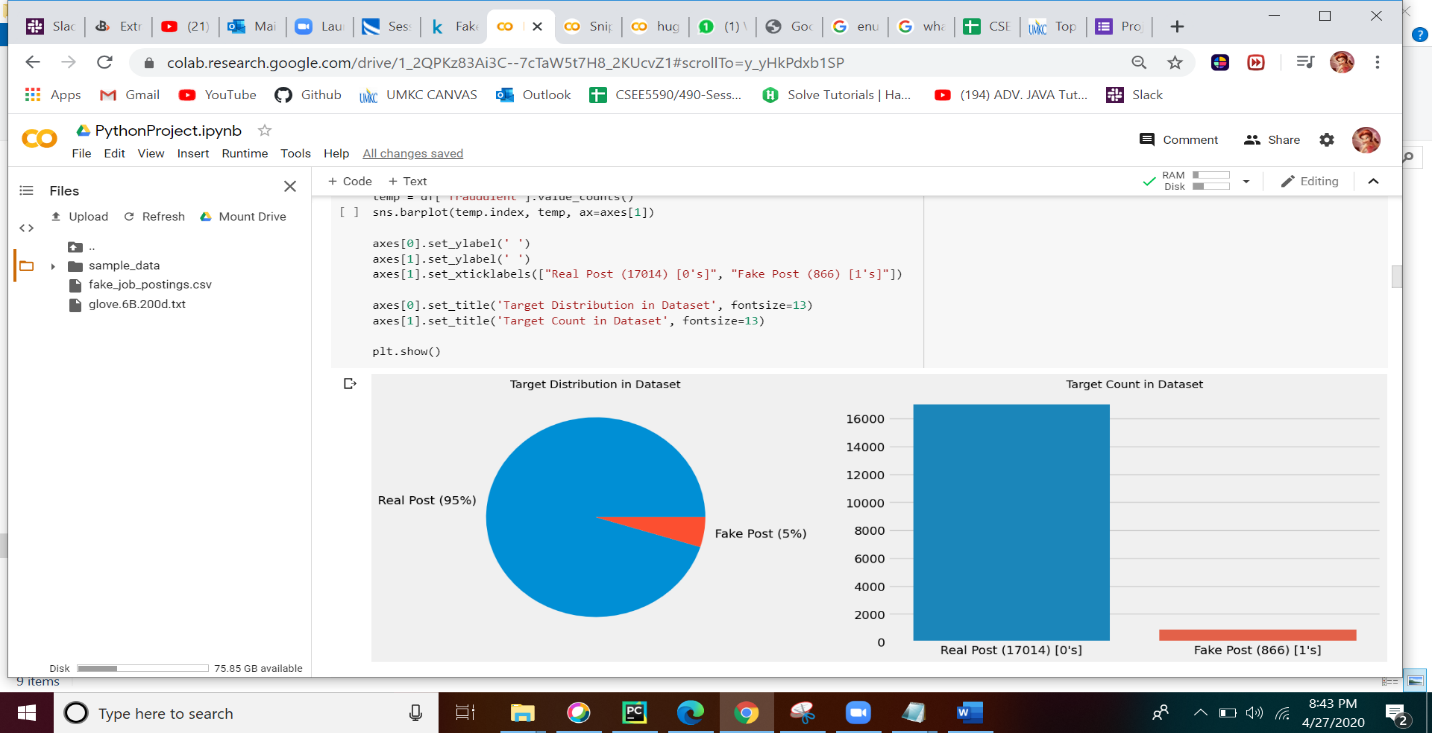
The second step is to extract text features.



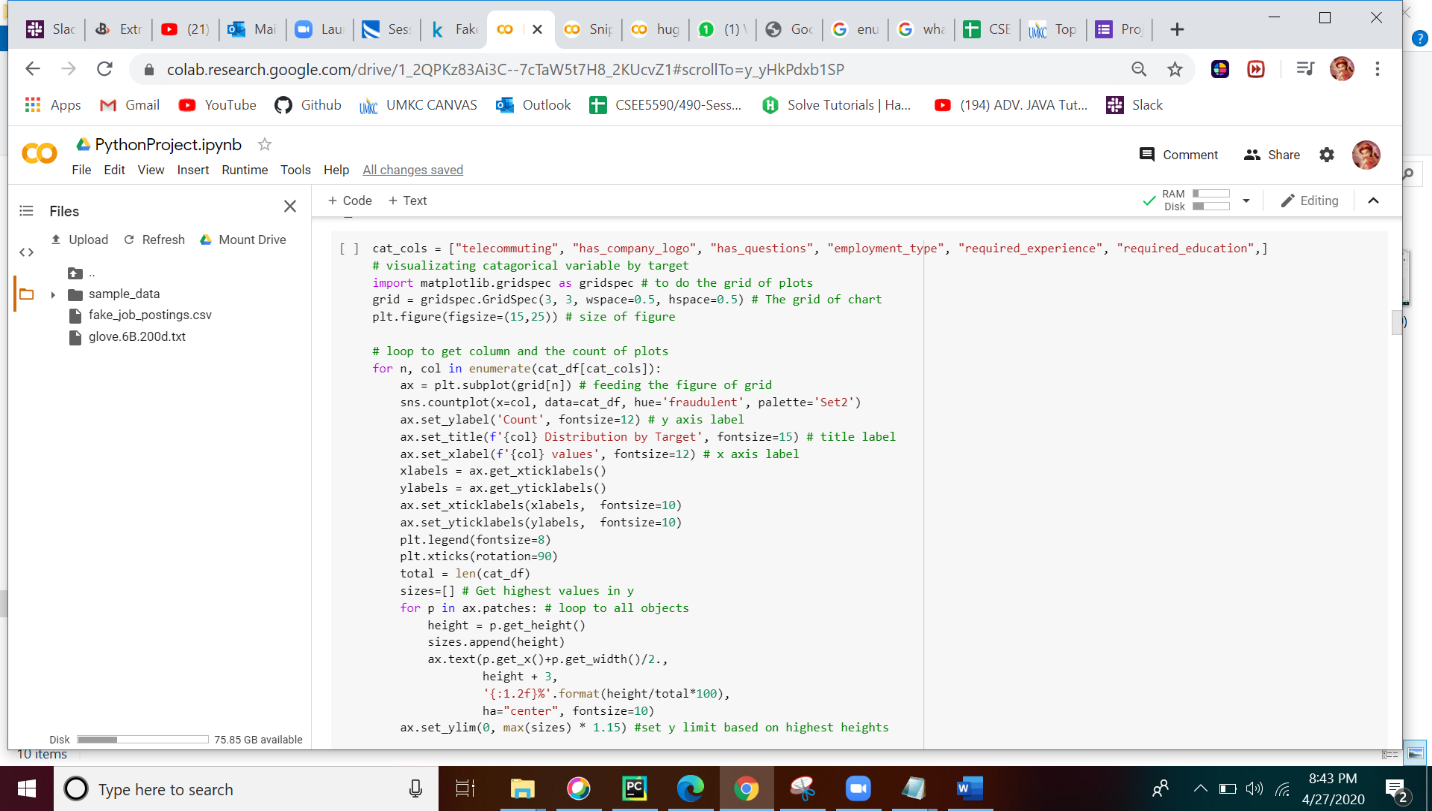
Next is to extract categorical features.



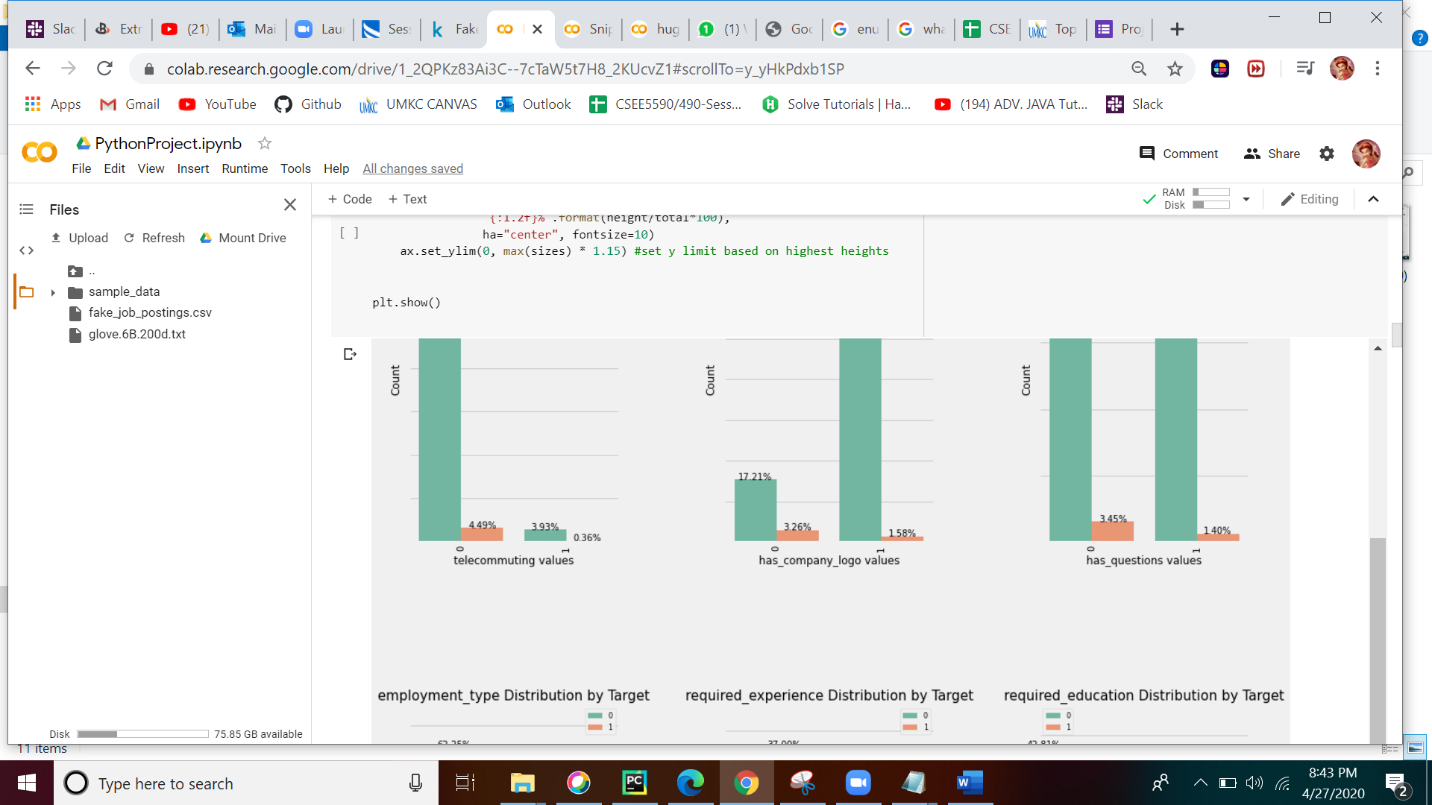
Visualize the categorical variable by target.



Considering several parameters compare number of characters in fake and real post. The code responsible for this part is as follows:



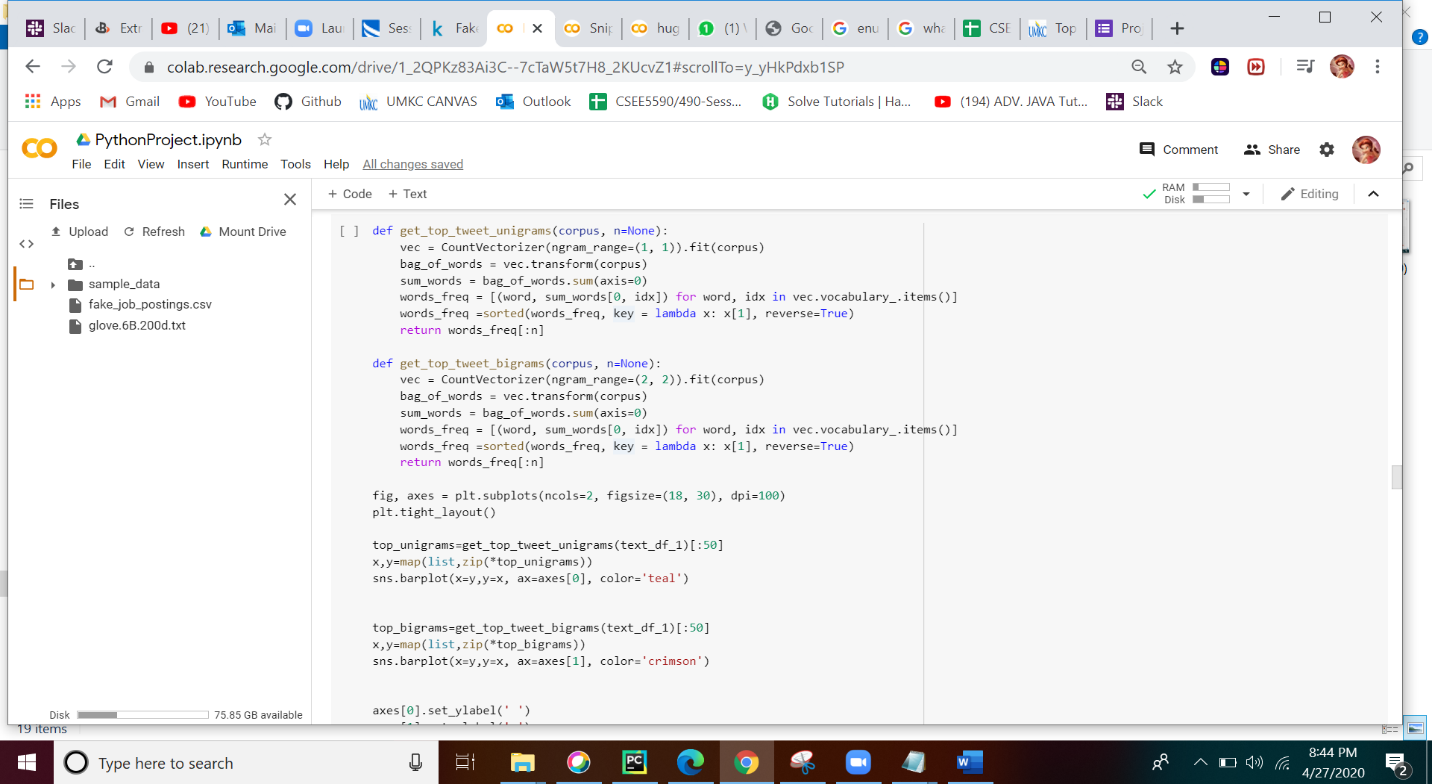
This results in the following output.

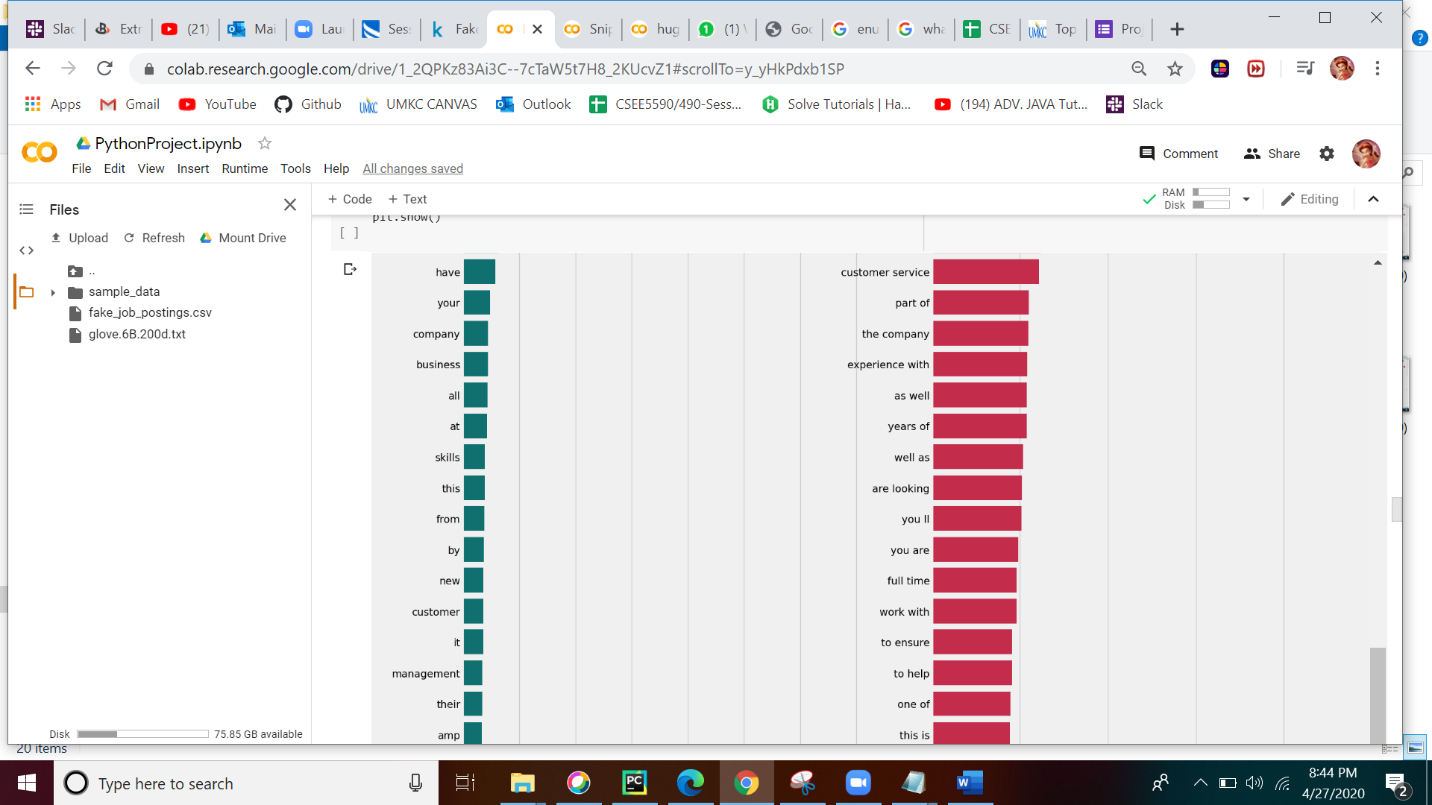


**Data preprocessing:**

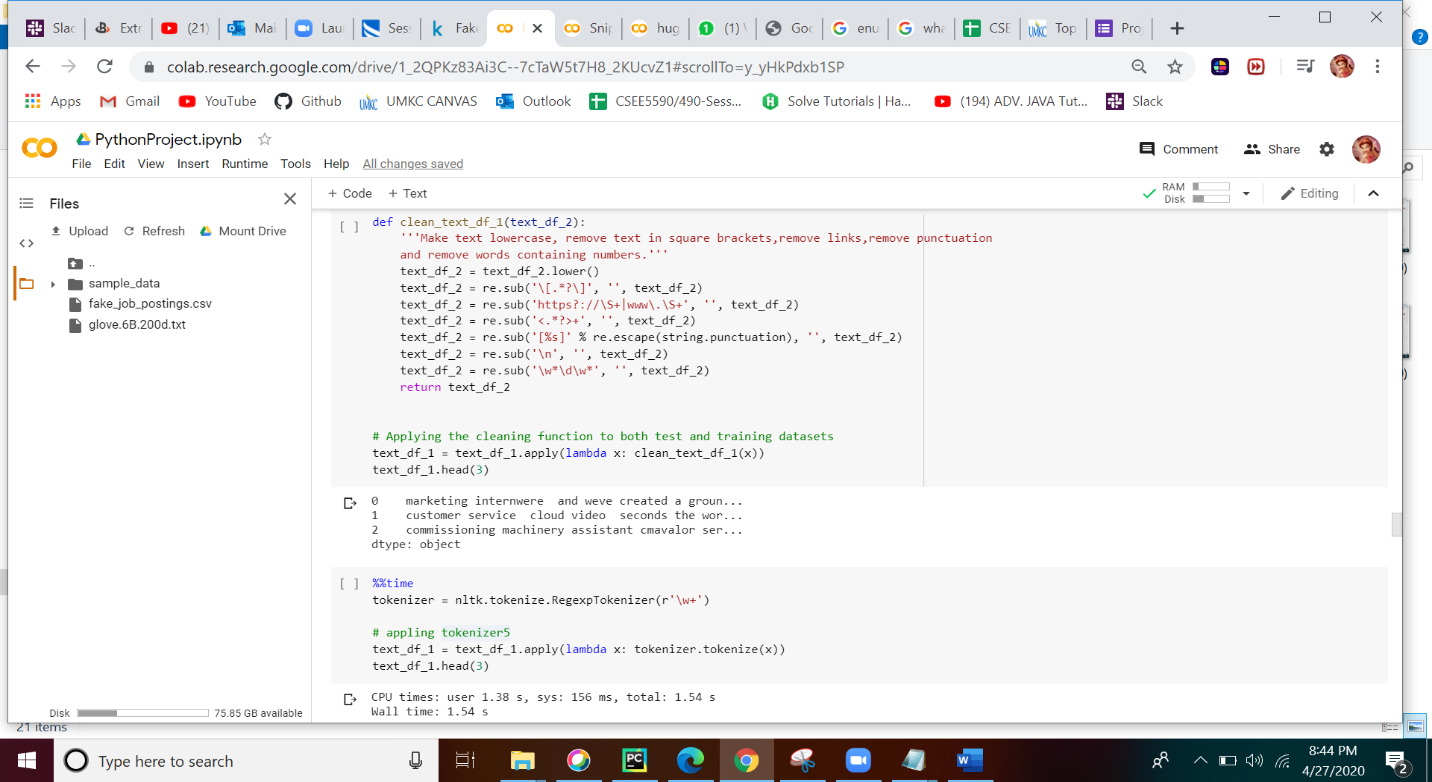
In this module, we have implemented pre-processing by removing all the punctuations, stop words, numbers, cleaning the data, tokenizing the data.

Initially, we have performed N-gram analysis to eliminate stop words.

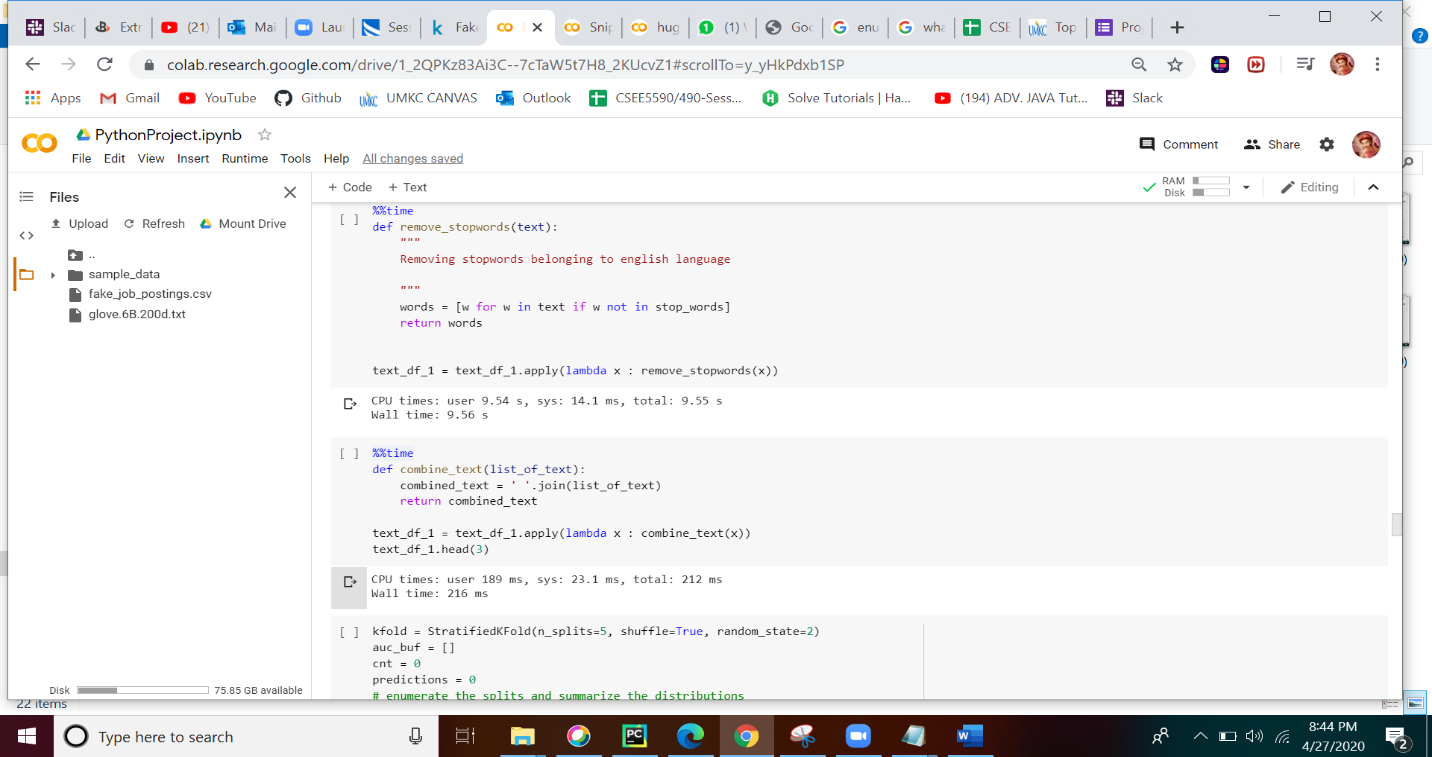


The output for the N-gram analysis is as follows:

We have performed the text cleanup, the code for this is as follows:

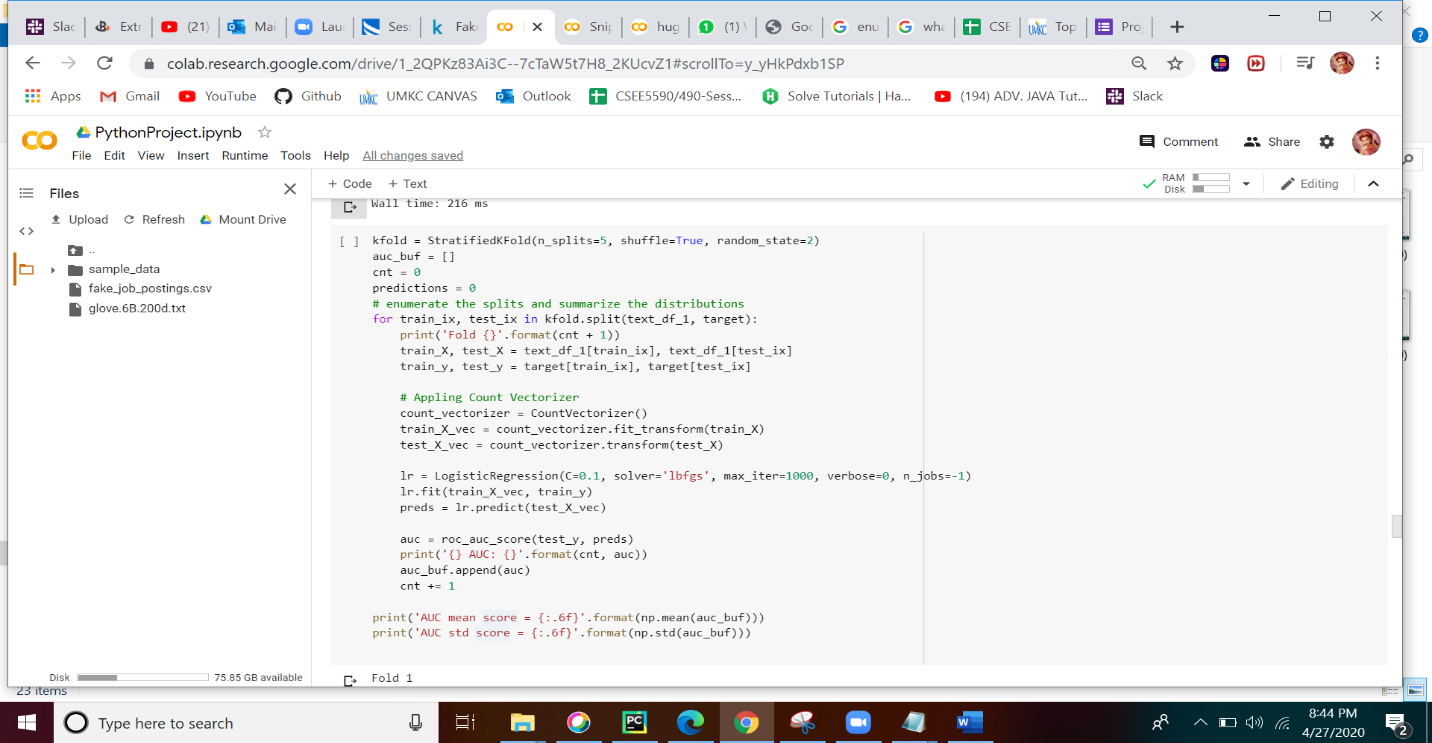


In order to perform tokenization, we have used tokenizer5 technique and removed stop words in English.



**Applying the dataset to various models:**

We have created a baseline model by enumerating the splits and summarized all the distributions.

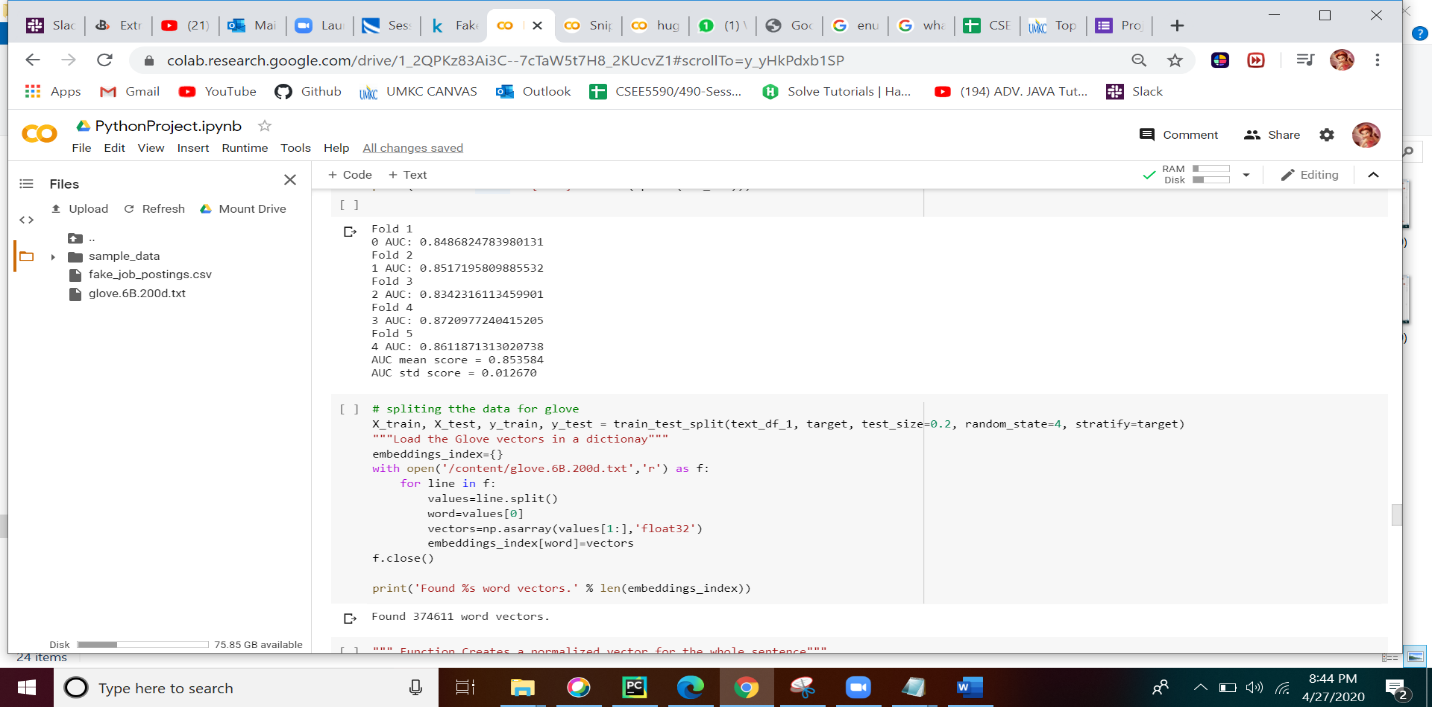


After the baseline model, we have applied the dataset to three different models namely

* + Count Vectorization
  + Glove Model
  + BERT technique

**Count Vectorization:**

It is nothing but an one hot encoding where the tokens are represented by the vectors and feeds to the model. The code which we implemented is as follows:

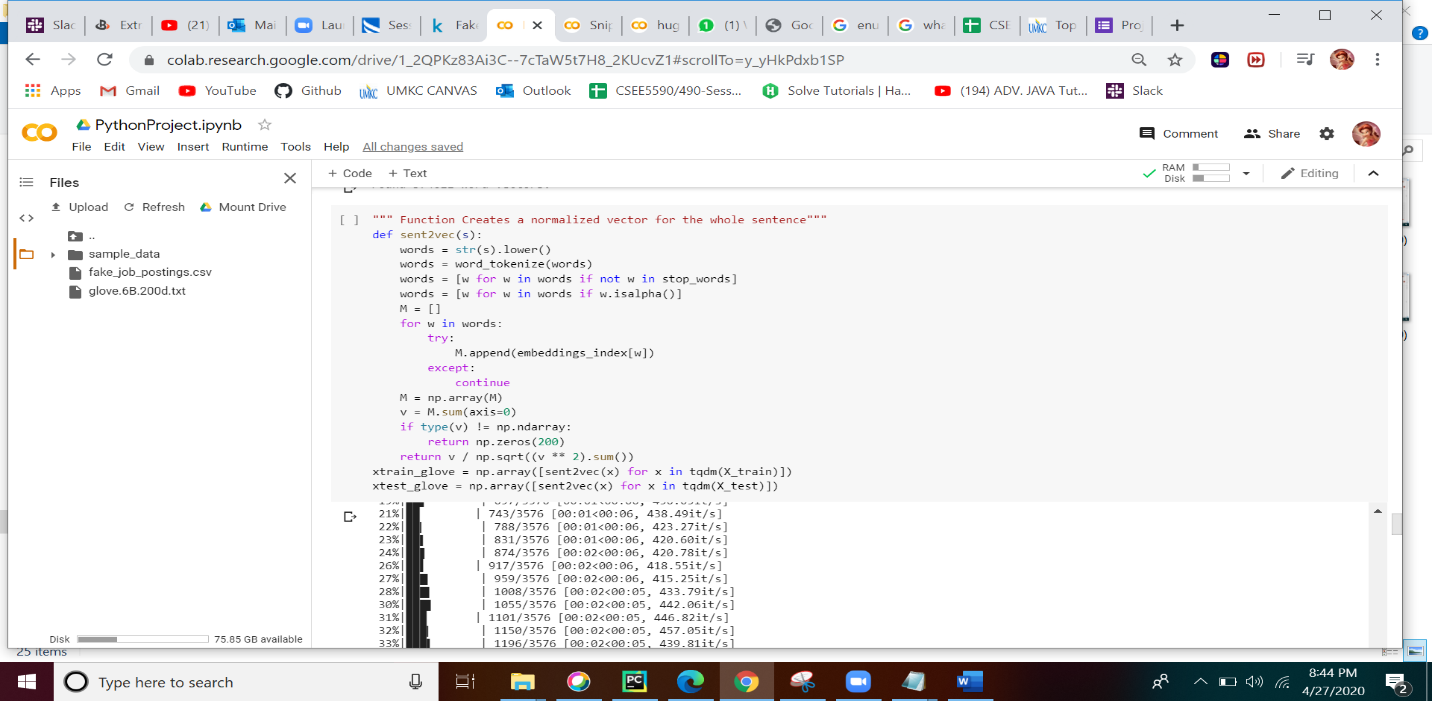
****

The AUC score for count vectorization model is 0.85.

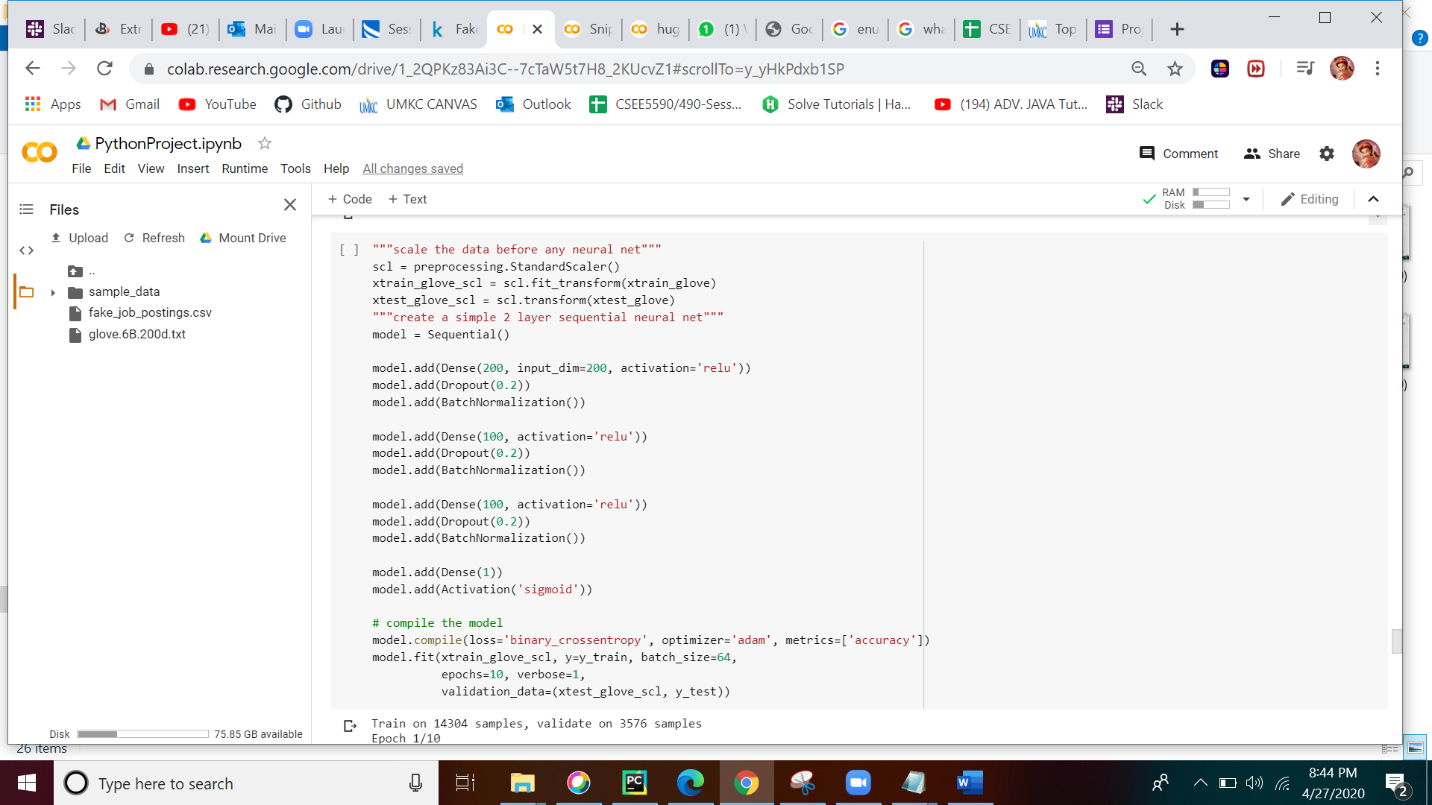
**Glove Model:**

We have used GloVe pretrained model and out of three different varieties available we tried 200D here. In order to use the GloVe technique, we have split the data and loaded the GloVe vectors into dictionary. After we loaded the vectors, a simple 2 layer neural network on Glove features is created.

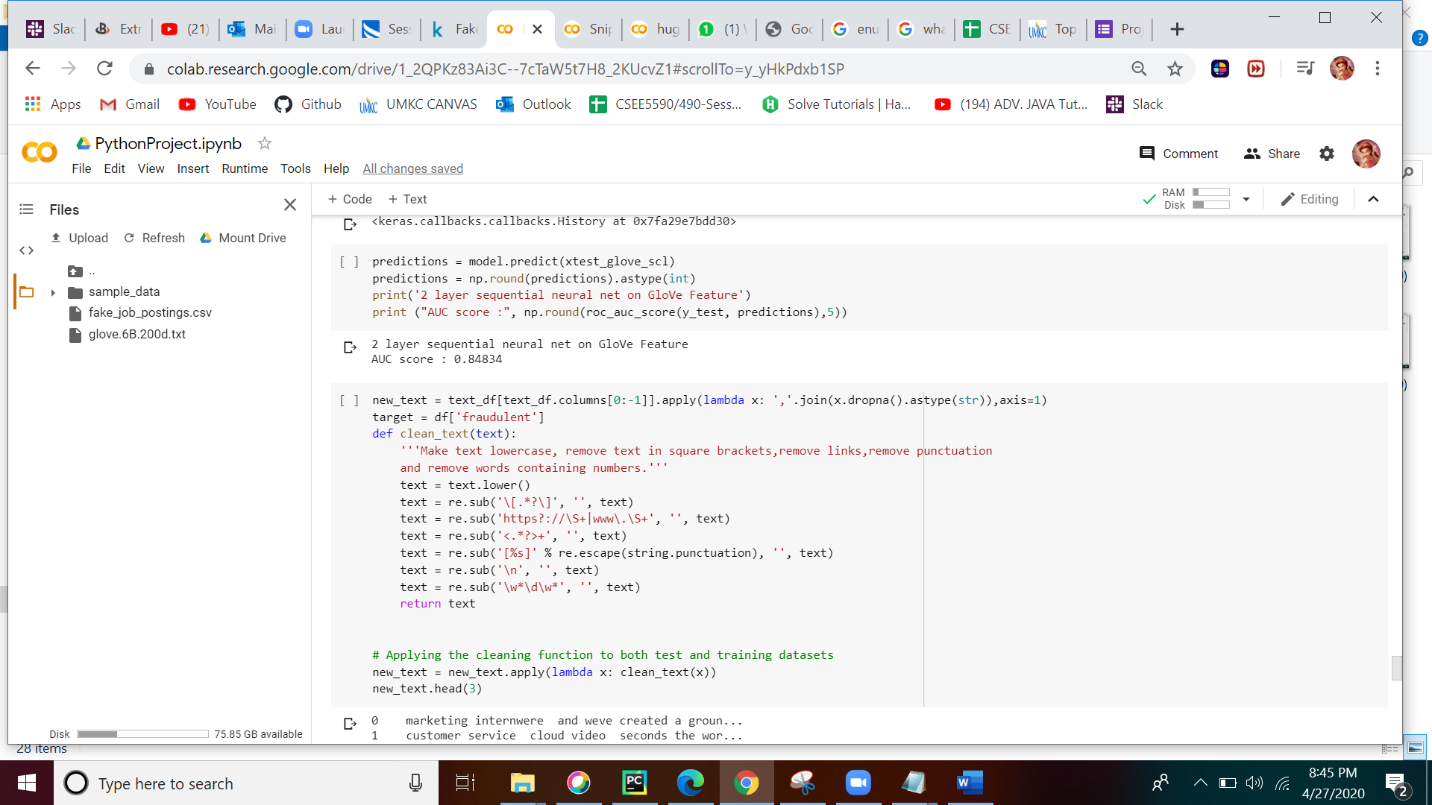
The code for using pre-trained model is represented as follows:



The code for splitting the data, loading the Glove Vectors into dictionary and creating a 2 layer neural net is as follows:

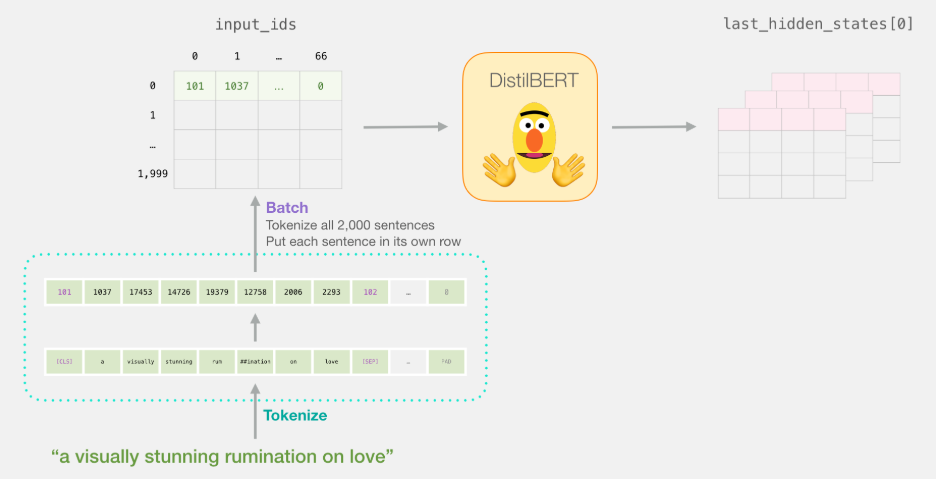


In order to predict the score, we used AUC score (Area under the ROC curve). For the GloVe method we have got an AUC score of 0.8 which is slightly lesser than the count vectorization method.



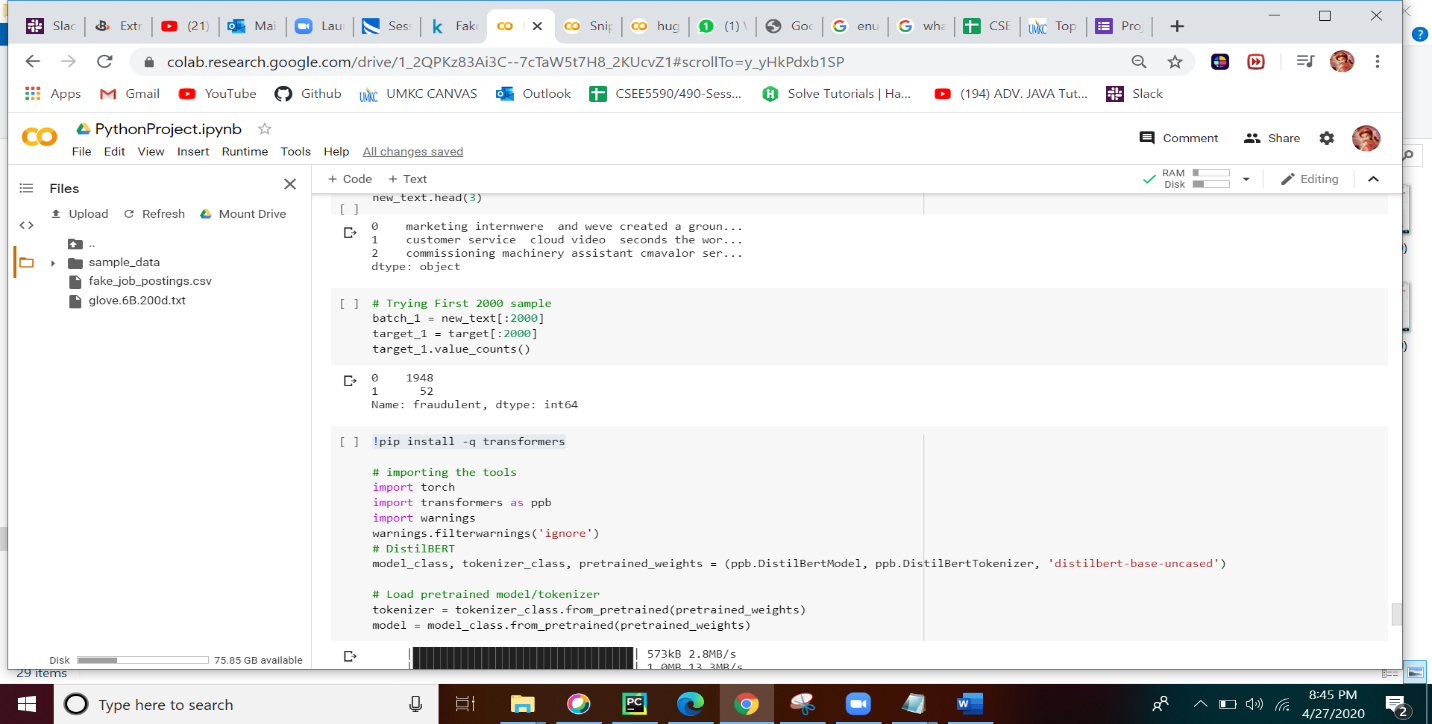
**Simple BERT model:**

BERT is an AI algorithm which is researched by the google, where in the normal NLP techniques, we eliminate the stop words, but in order to save the context of the words, BERT maintains Bi-directional relationship and add pre-trained weights. All the output vectors are given to the BERT model and results a vector of size 768 for each word.This can be explained by an architecture.

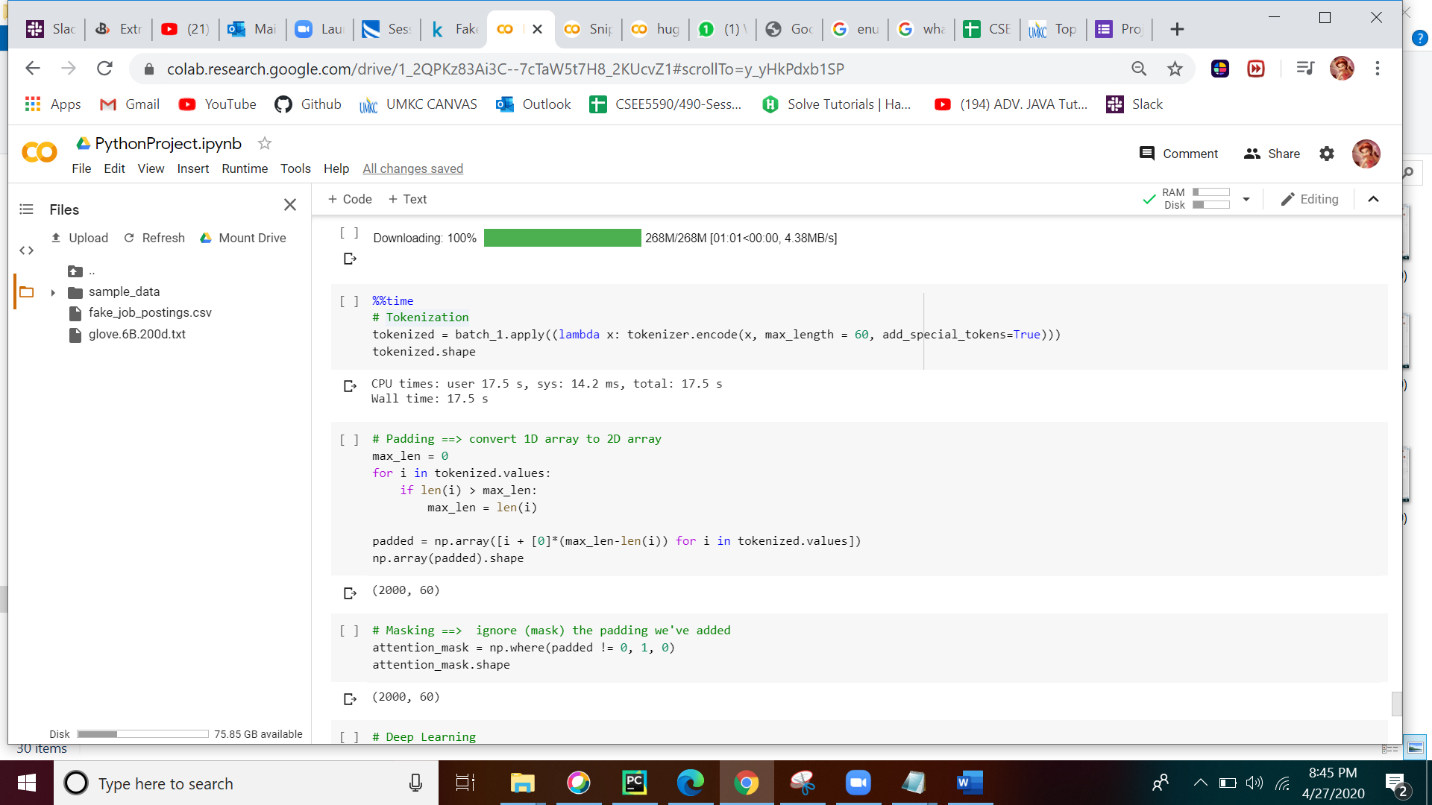


Code for BERT implementation:

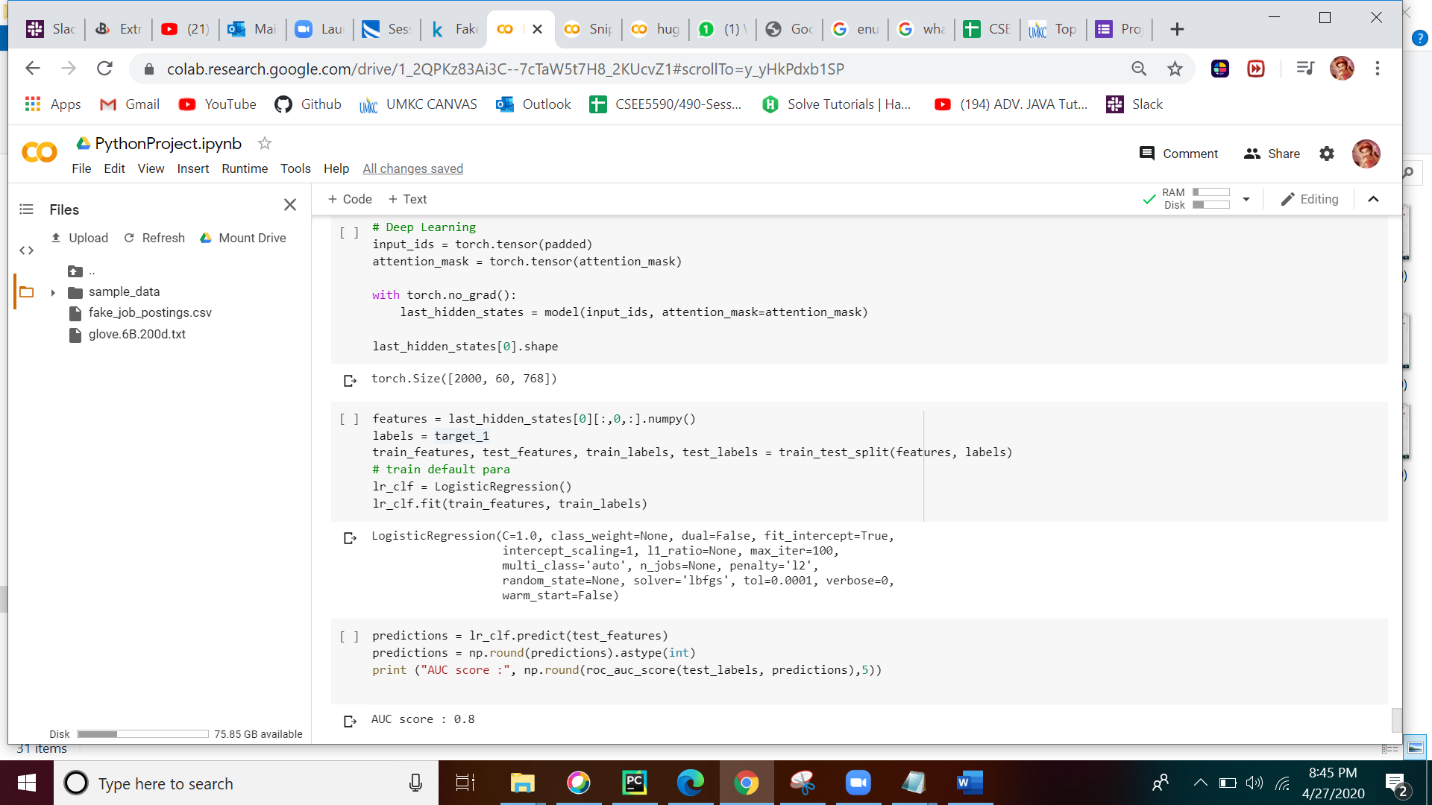
Initially, we are only implementing on 2000 samples.



Tokenizing, padding and masking the data



Applying to the BERT model and fitting a logistic regression.



The AUC score for the BERT model on 2000 samples is almost 0.8 but if trained on all the samples it may result in more AUC score.By this we can say that BERT performance better than other models.

**Advantages:**

* Through Count Vectorization we were able to preprocess the text data prior generating the vector representation
* Glove, basically took less time for training and testing of Data and we also observed that we can stop training when improvements have become small.
* Bert model can be used for a wide variety of language tasks. If you want to fine tune the original model based on your own dataset you can do just by adding a single layer on top of the core model.

**Contribution:**

* Design :Haritha Bheemineni
* Code Execution : Satya Anusha PrattiPati/Yamini Dontireddy
* User Manuals: Vamshi Jakkula

**References:**

* <https://www.kaggle.com/shivamb/real-or-fake-fake-jobposting-prediction>
* <https://nlp.stanford.edu/seminar/details/jdevlin.pdf>
* [**https://nlp.stanford.edu/projects/glove/**](https://nlp.stanford.edu/projects/glove/)