#### Hate Speech Detection



**Fall-2021**

**Supervised By**

**Dr. Shahid Iqbal Malik**

# Abstract

With the advent of the internet and numerous social media platforms, citizens now have enormous opportunities to express and share their opinions on various societal and political issues. This phenomenal growth of the internet, social media networks, and messaging platforms provide plenty of opportunities for building intelligent systems, but these are also being heavily misused by certain groups who often disseminate offensive, racial, and hate speeches. Hence, detecting hate speech at the right time plays a crucial role as its spread might affect social fabrics. In recent times, although a few benchmark datasets have emerged for hate speech detection, these are limited in volume and also do not follow any uniform annotation schema. In this project we have done the experiments [1] on the 3 datasets show that the proposed framework for hate speech detection.

# Introduction

With the phenomenal growth in digital technology and the internet, social media have upsurged as a strong platform to allow people to express their opinions on a variety of topics ranging from political, financial, education, sports, defense, religion and other societal issues. Statistics reveals that 6K tweets/second, 200 billion tweets/year are generated on twitter alone, indicating the exponential rise in the consumption of social media. The diversity in language usage across the globe also poses a great challenge due to the variety of linguistic patterns. Social media’s main aim is to connect more people to support them in expressing their right to freedom of speech. However, these mediums are often misused by certain groups to malign others, spreading offensive and hate speeches targeting individuals and/or other groups. This can be considered as a political violence that jeopardizes social stability and peace. Hence, detecting these in proper time and preventing their dis- semination to a larger section is of utmost importance to maintain the harmony in the society and to maintain the law-and-order situations.

United Nations strategy and plan of action on hate speech de- scribes hate speech as any kind of communication in speech, writing or behavior, that attacks or uses pejorative or discriminatory language concerning a person or a group based on who they are, in other words, based on their religion, ethnicity, nationality, race, color, descent, gender or identity factor. So, without suppressing the right to freedom of expression, the focus should be on building robust computational systems which can detect different types of hateful contents that can create disharmony. International Covenant on Civil and Political Rights (ICCPR) is a multi- lateral treaty adopted by United Nations General Assembly. The covenant commits its party to respect the right to freedom of speech along with other fundamental rights for every citizen. As of September 2019, the covenant has 173 parties. Article 19 of it states that:

1. Everyone shall have the right to hold opinions without interference.
2. Everyone shall have the right to freedom of expression. This right shall include freedom to seek, receive and impart information of all kinds, regardless of frontiers, either orally, in writing or print, in the form of art, or through any media of person’s choice. However, an amendment was done and a new article 20 was introduced stating that any advocacy of national, religious or racial hatred that constitutes incitement to discrimination, hostility, violence shall be prohibited under law. There are incidents when viciousness of these messages evolved into genocide, xenophobia and bigotry. Several incidents in the past had evidence of deadly action like mass murder before posting hate messages in online forums. [[1](file:///E:/A%20deep%20neural%20network%20based%20multi-task%20learning%20approach%20to%20hate-converted.docx#_bookmark65)] reported the massive violence in Kenya after hateful messages circulated in media in 2007–2008. In order to build efficient machine learning based hate speech detection system, sufficient amount of labeled data is required. Although there exist a few benchmark datasets, they are often limited by the size, and do not follow any uniform annotation schema.

A brief overview of the related background literature is presented and [discusses](file:///E:/A%20deep%20neural%20network%20based%20multi-task%20learning%20approach%20to%20hate-converted.docx#_bookmark10) in details the proposed methodology. The datasets used for the experiments and definitions of different variants of hate are described. Experimental setup and evaluation metrics are presented in it. The evaluation results and comparisons to the state-of-the-arts.

# Background/ Literature Review

# Proposed Models

The proposed model is evaluated on three different datasets related to hate speech classification, racism, sexism detection, and offensive language detection. Below are the pre-processing and embeddings used in the project:

## Pre-processing

Social media posts contain a lot of noisy texts which are not considered as useful features for the classification. We perform the following steps to remove the noise, and make it ready for machine learning experiments:

1. All the characters like |:,? were removed along with the numbers and URLs.
2. Words are reduced to lower case so that words such as ‘‘HAPPY’’, ‘‘happy’’ and ‘‘Happy’’ will have the same syntax and will utilize the same pre-trained embedding values.
3. Word Segmentation is being done using the Python based word segment to preserve the important features present in hashtag mentions.
4. All the @ (ex.@abc) mentions were replaced with the com- mon token, i.e *user*.
5. The stop words were not removed due to the risk of losing some useful information, and this was also empirically found to be of little or no impact on the classification performance after removing them.

## Embeddings

Below are the 3 types of embeddings used in the project:

1. TFIDF
2. Word2Vec
3. Fast Text

### **TDIDF**

**TF-IDF (term frequency-inverse document frequency)**is a statistical measure that evaluates how relevant a word is to a document in a collection of documents. This is done by multiplying two metrics: how many times a word appears in a document, and the inverse document frequency of the word across a set of documents.

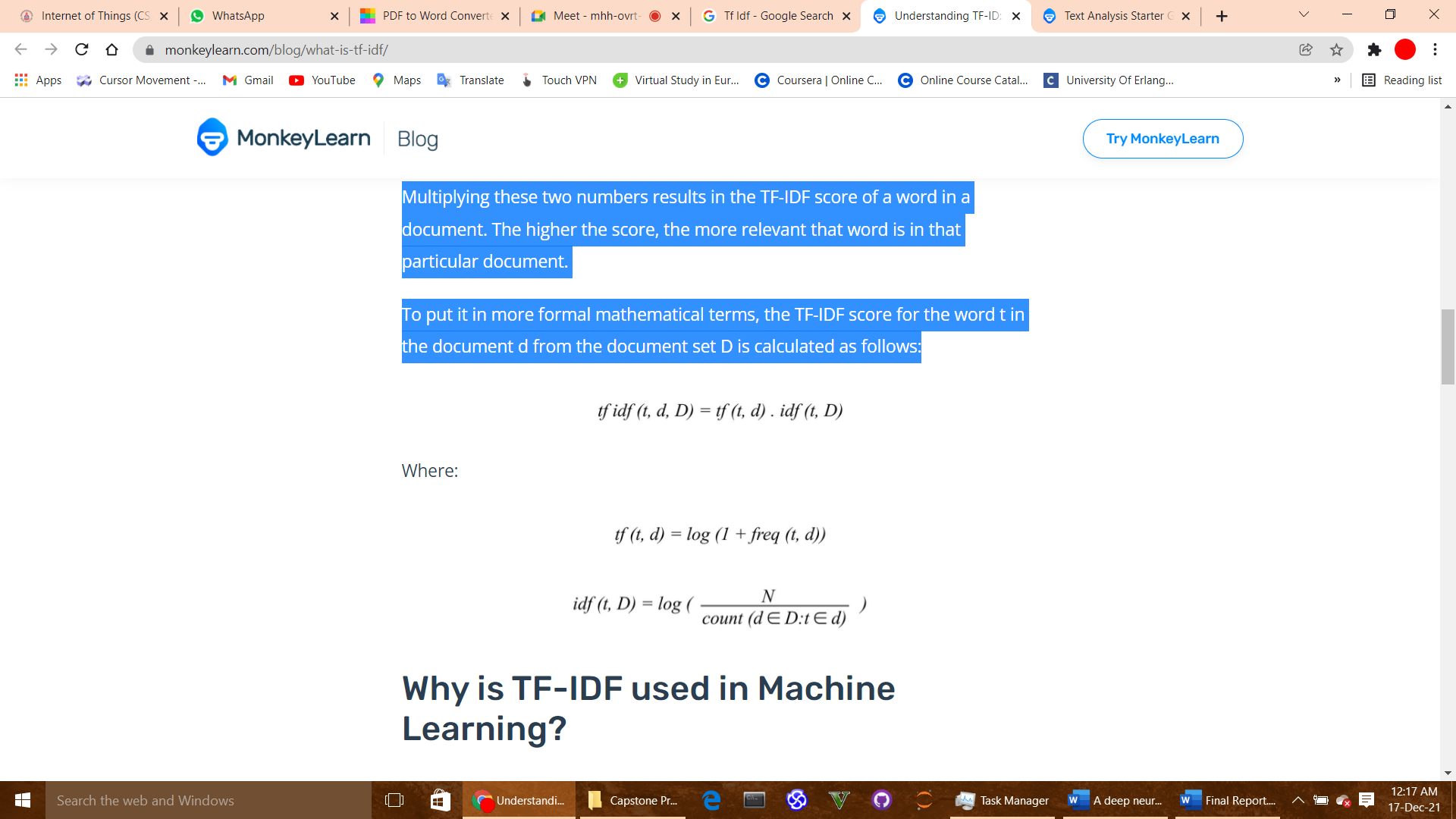
**How TF IDF is calculated**

TF-IDF for a word in a document is calculated by multiplying two different metrics:

* The term frequency of a word in a document. There are several ways of calculating this frequency, with the simplest being a raw count of instances a word appears in a document. Then, there are ways to adjust the frequency, by length of a document, or by the raw frequency of the most frequent word in a document.
* The inverse document frequency of the word across a set of documents. This means, how common or rare a word is in the entire document set. The closer it is to 0, the more common a word is. This metric can be calculated by taking the total number of documents, dividing it by the number of documents that contain a word, and calculating the logarithm.
* So, if the word is very common and appears in many documents, this number will approach 0. Otherwise, it will approach 1.

Multiplying these two numbers results in the TF-IDF score of a word in a document. The higher the score, the more relevant that word is in that particular document. By applying the TF IDF, we generate a matrix of *n x m* size.

To put it in more formal mathematical terms, the TF-IDF score for the word t in the document d from the document set D is calculated as follows:



## Word2Vec

Word embedding is one of the most popular representations of document vocabulary. It is capable of capturing context of a word in a document, semantic and syntactic similarity, relation with other words, etc. Word2Vec is a method to construct such an embedding. It can be obtained using two methods: Skip Gram and Common Bag of Words (CBOW). In this project we have implemented the Skip Gram model.

**Skip Gram**

The Skip-gram model architecture usually tries to achieve the reverse of what the CBOW model does. It tries to predict the source context words (surrounding words) given a target word (the center word). Considering our simple sentence from earlier, “the quick brown fox jumps over the lazy dog”. If we used the CBOW model, we get pairs of (context\_window, target\_word) where if we consider a context window of size 2, we have examples like([quick, fox], brown), ([the, brown], quick), ([the, dog], lazy) and so on. Now considering that the skip-gram model’s aim is to predict the context from the target word, the model typically inverts the contexts and targets, and tries to predict each context word from its target word. Hence the task becomes to predict the context [quick, fox] given target word ‘brown’ or [the, brown] given target word ‘quick’ and so on. Thus, the model tries to predict the context\_window words based on the target\_word.

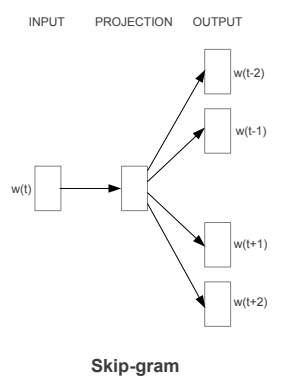


Figure : Skip Gram Pictorial Represemtation

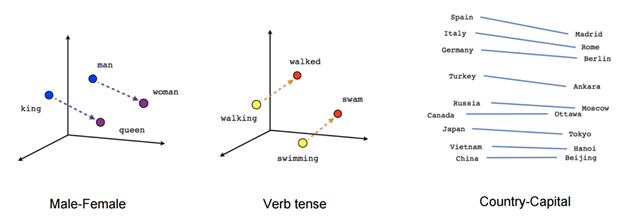


Figure : Pictorial Representations

## Fast Text

FastText is another word embedding method that is an extension of the word2vec model. Instead of learning vectors for words directly, FastText represents each word as an n-gram of characters. So, for example, take the word, “*artificial*” with n=3, the FastText representation of this word is <*ar, art, rti, tif, ifi, fic, ici, ial, al*>, where the angular brackets indicate the beginning and end of the word.

This helps capture the meaning of shorter words and allows the embeddings to understand suffixes and prefixes. Once the word has been represented using character n-grams, a skip-gram model is trained to learn the embeddings. This model is considered to be a bag of words model with a sliding window over a word because no internal structure of the word is taken into account. As long as the characters are within this window, the order of the n-grams doesn’t matter.

FastText works well with rare words. So even if a word wasn’t seen during training, it can be broken down into n-grams to get its embeddings.

Word2vec and GloVe both fail to provide any vector representation for words that are not in the model dictionary. This is a huge advantage of this method.

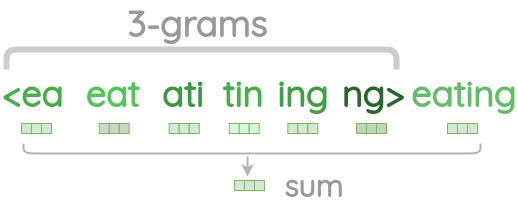


Figure 3: Pictorial Representation

# Implementation Details

## Pre-Processing

For all the datasets we have applied the above written Pre-processing techniques. Below are the screenshots of the code:

Importing the libraries and reading the data csv file in the below screenshot

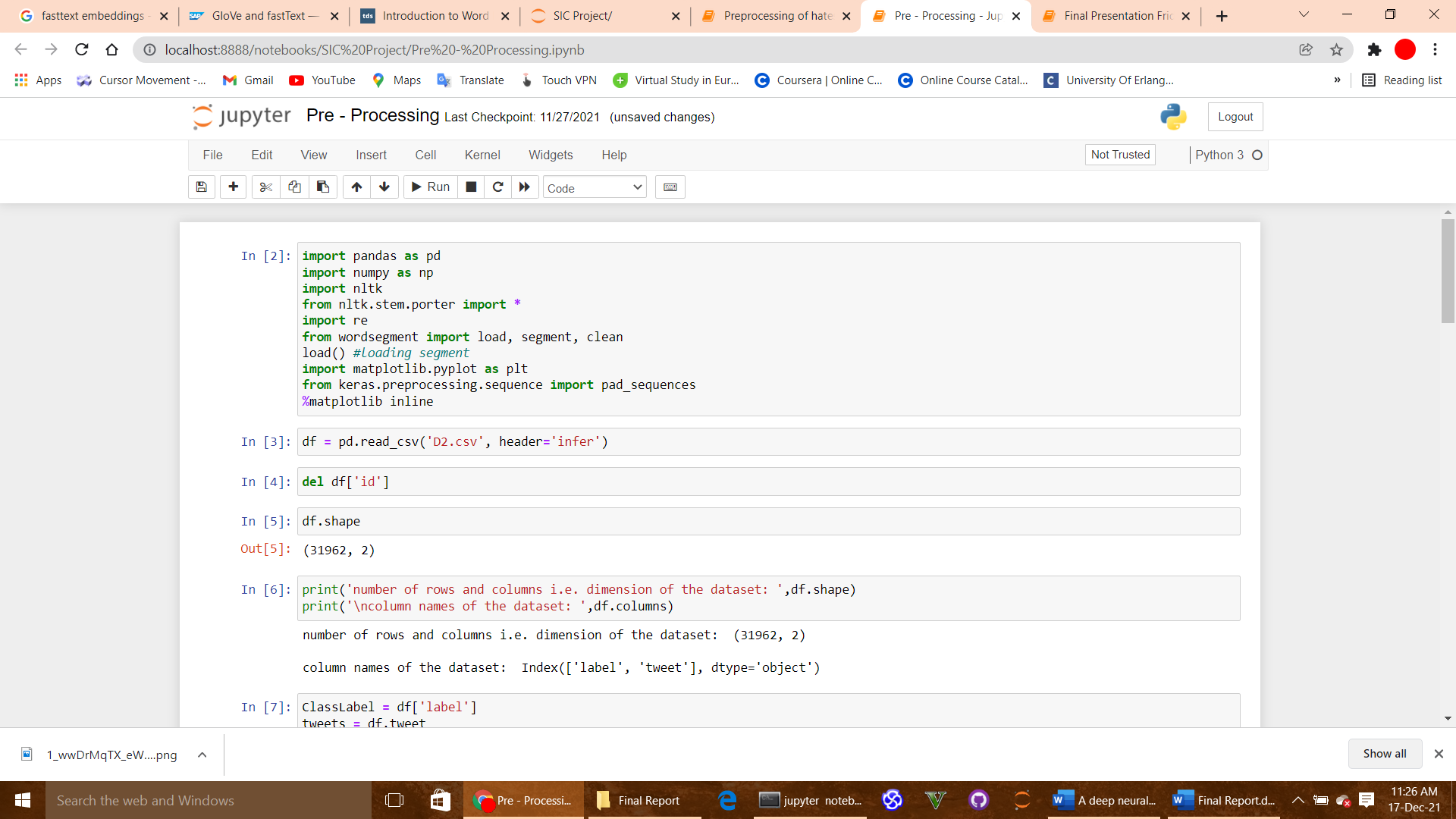


Figure 4: Data Loading

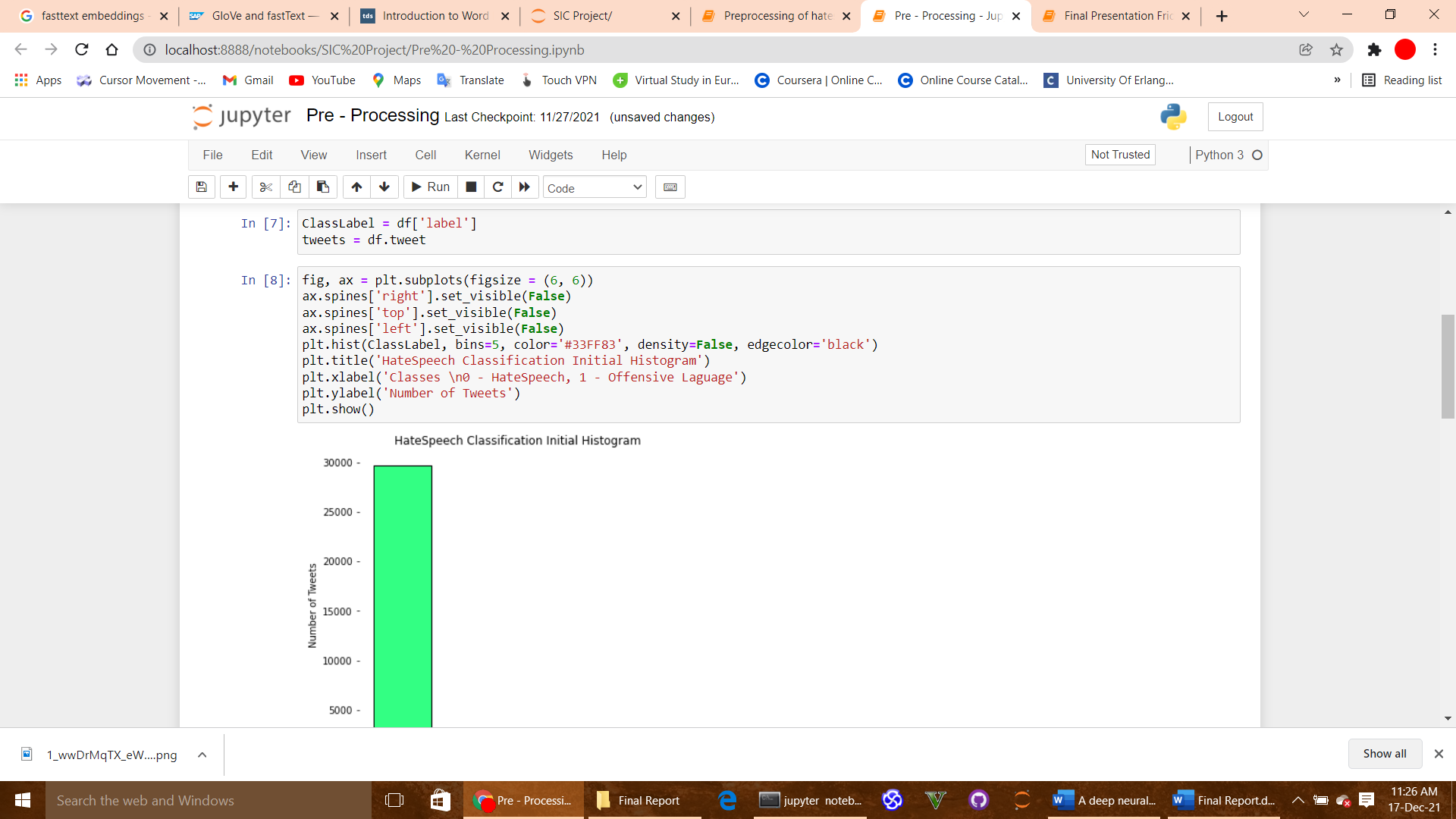


Figure 5: Data Visualization

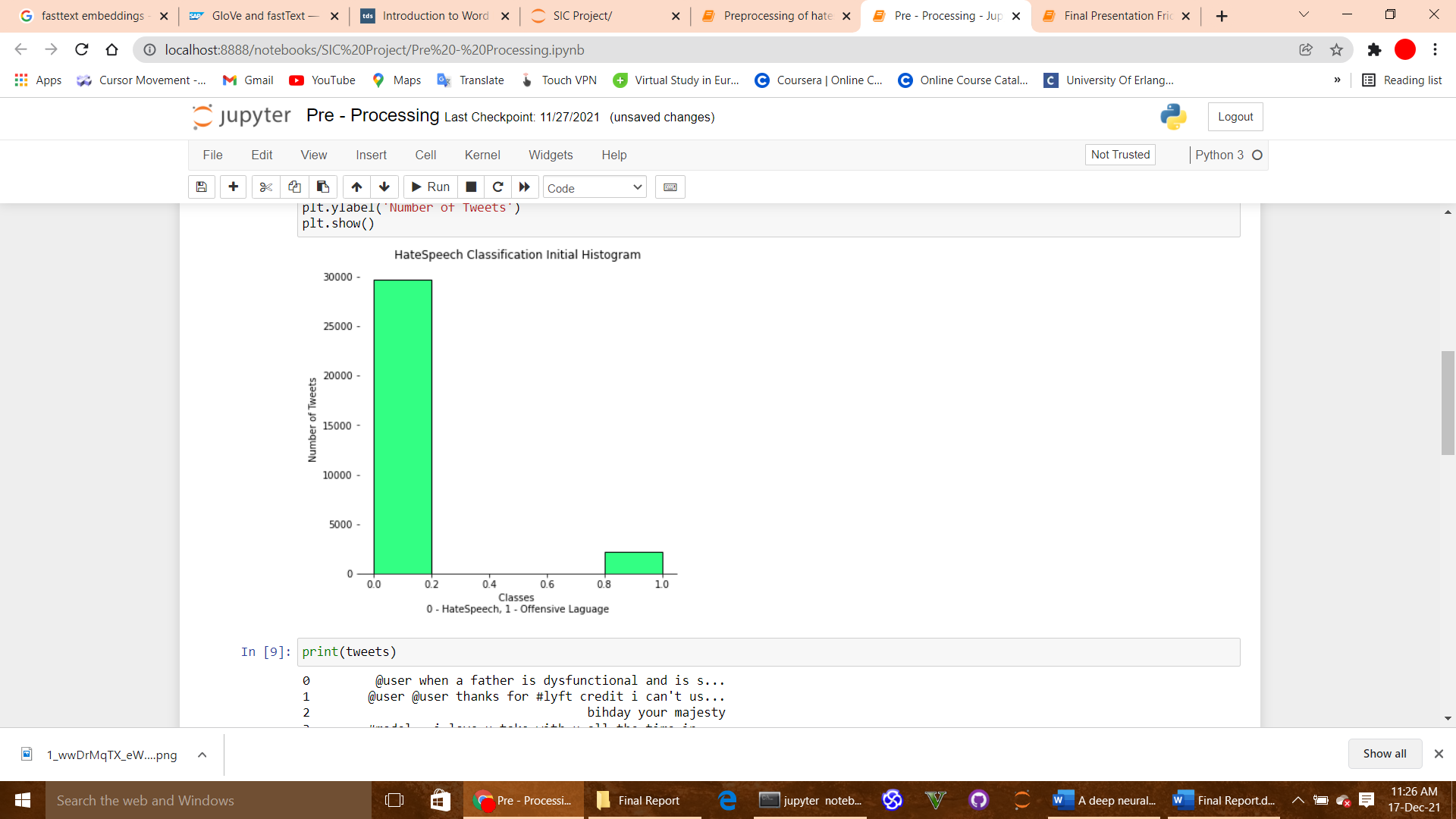


Figure 6: Data Visualization

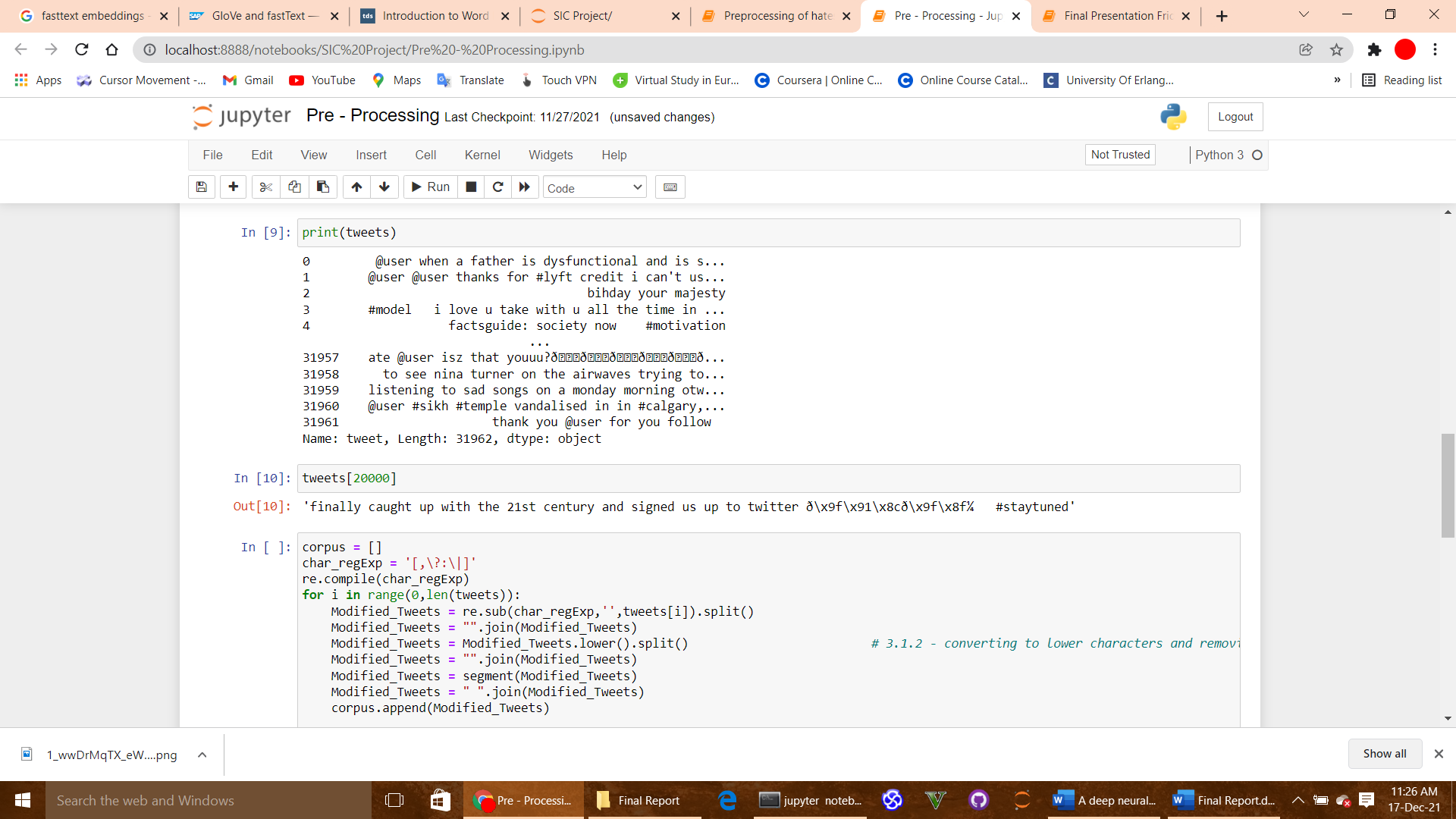


Figure 7: Preprocessing the tweets from the dataset

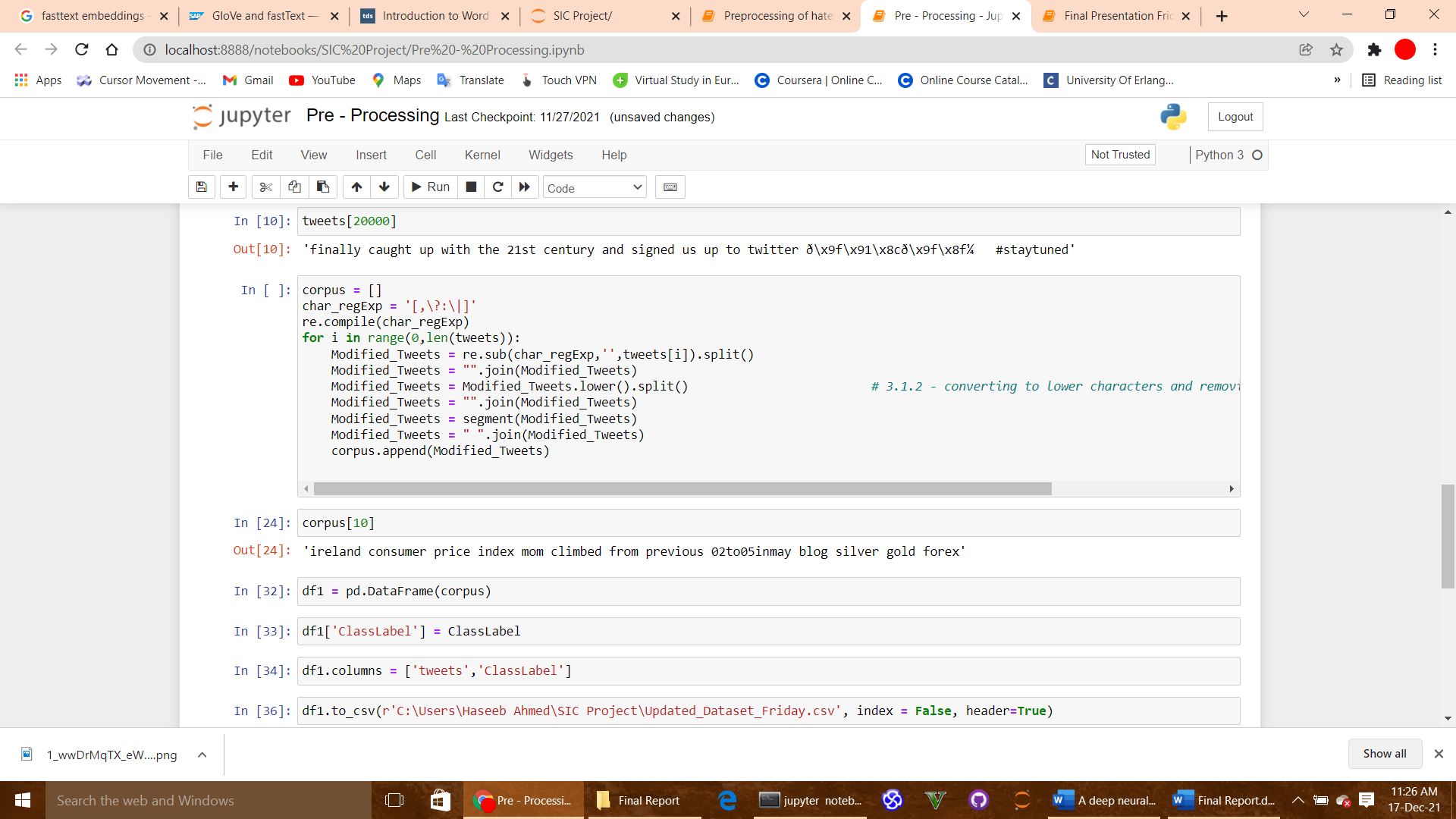


Figure 8: Preprocess data is saves in separate file

### Smote

This is a type of data augmentation for tabular data and can be very effective. Perhaps the most widely used approach to synthesizing new examples is called the Synthetic Minority Oversampling Technique, or SMOTE for short. This technique was described by Nitesh Chawla, et al. Below is the screenshot of code of smote used in the code.

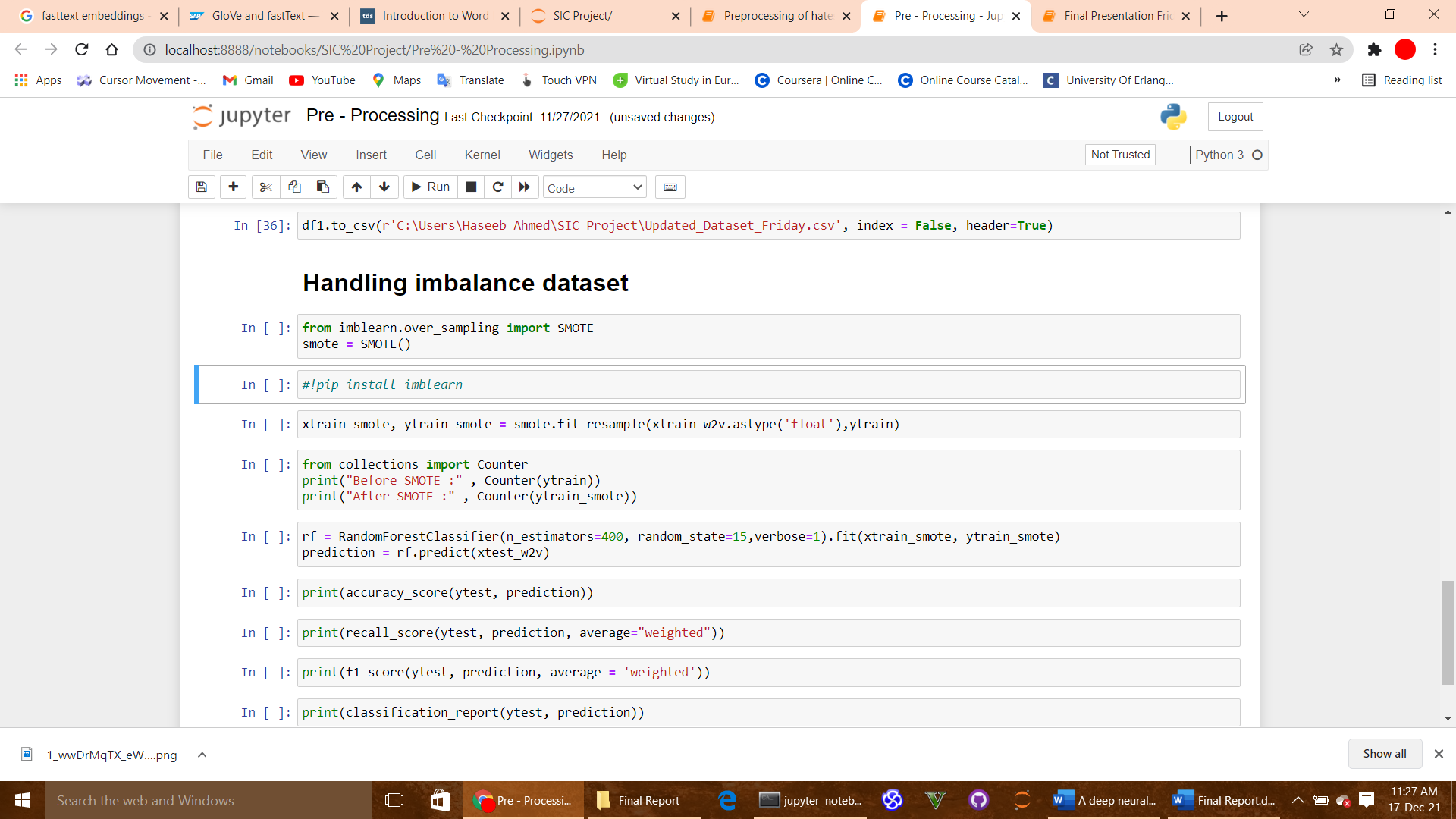


Figure 9: Balancing Code Screenshot

## TF IDF

Contrary to what some may believe, TF IDF is the result of the research conducted by two people. They are Hans Peter Luhn, credited for his work on term frequency (1957), and Karen Spärck Jones, who contributed to inverse document frequency (1972). Below is the code screenshot of TFIDF:

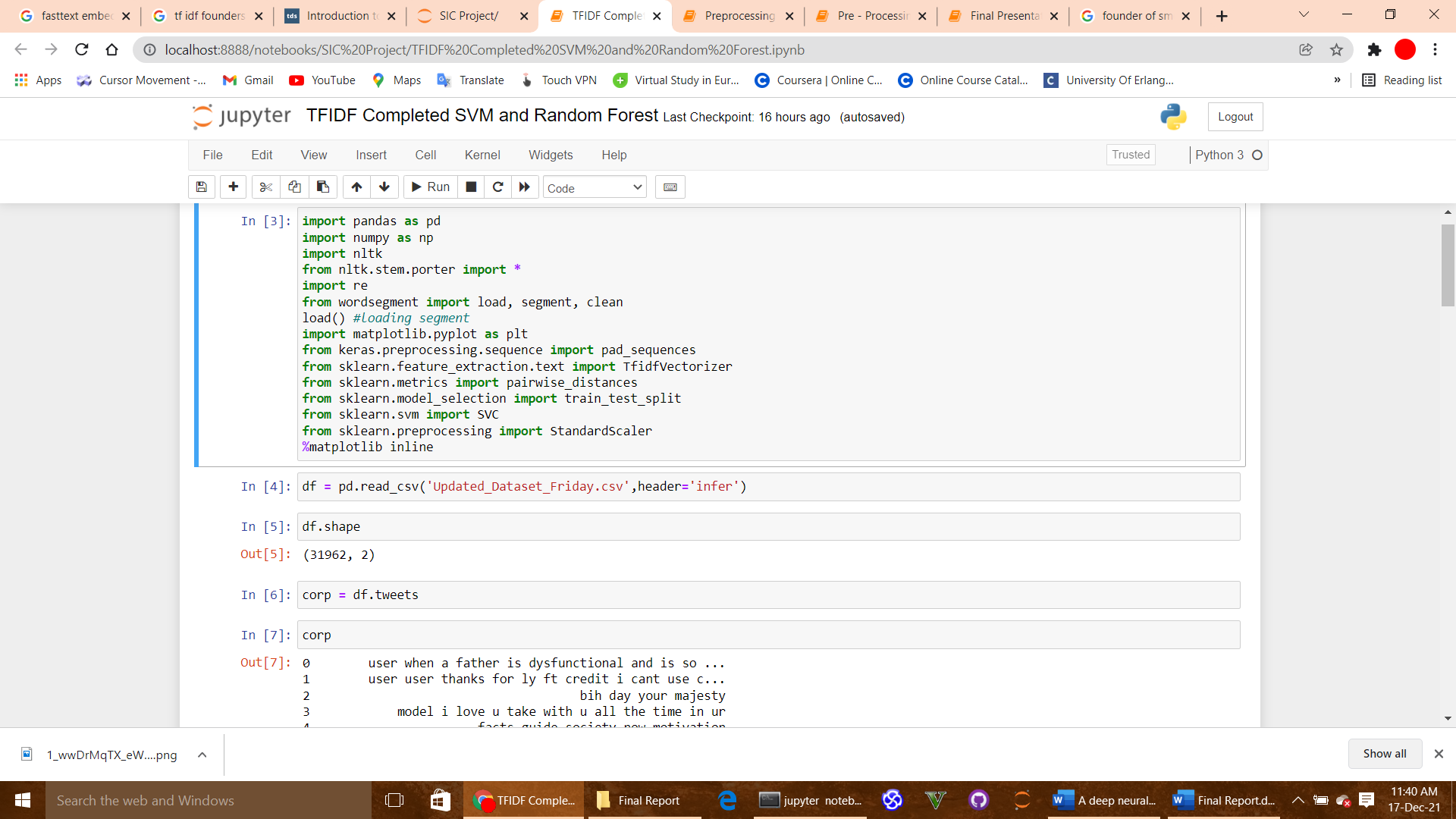


Figure 10: Importing libraries

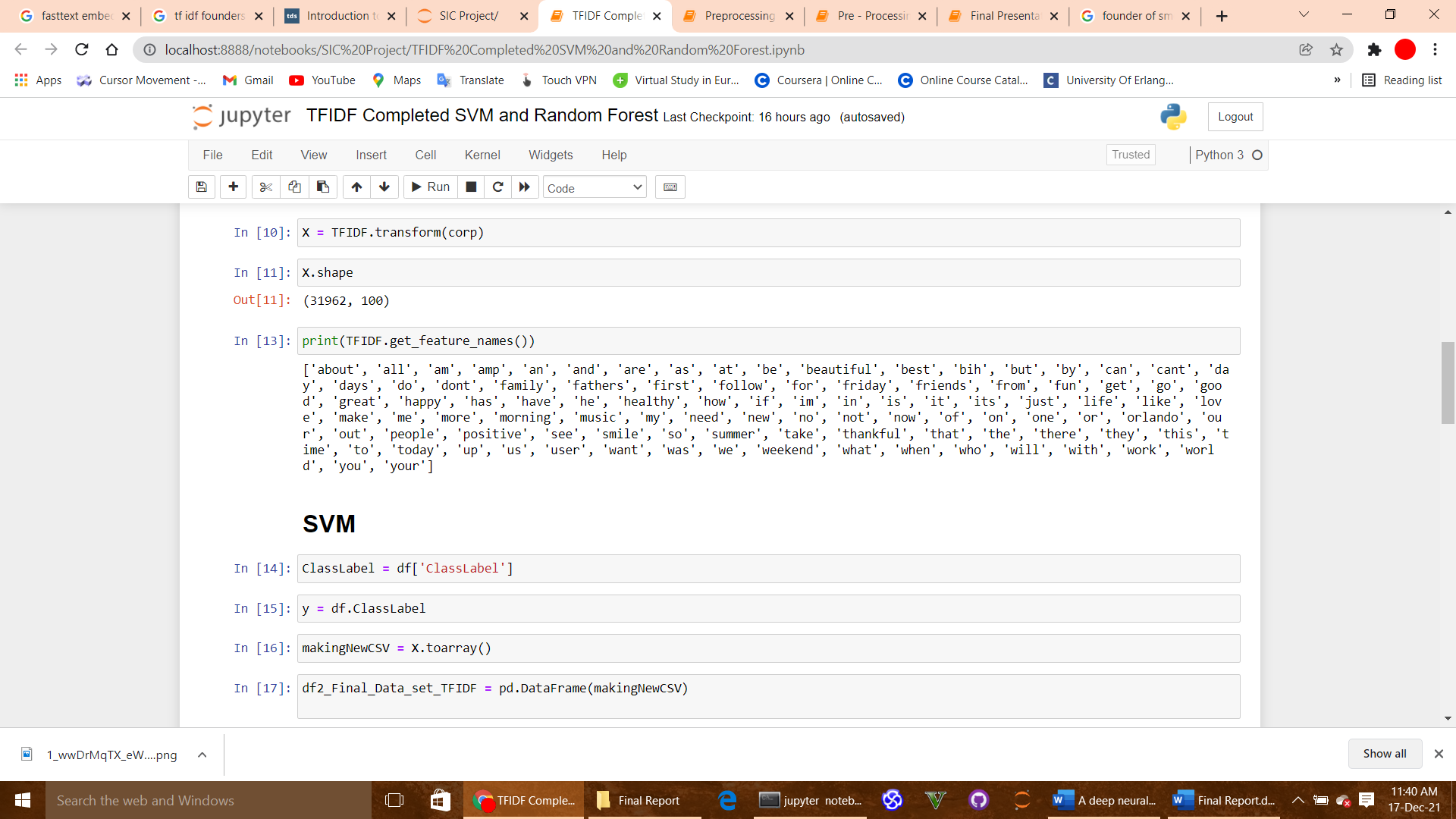


Figure 11: 100 Features Selection

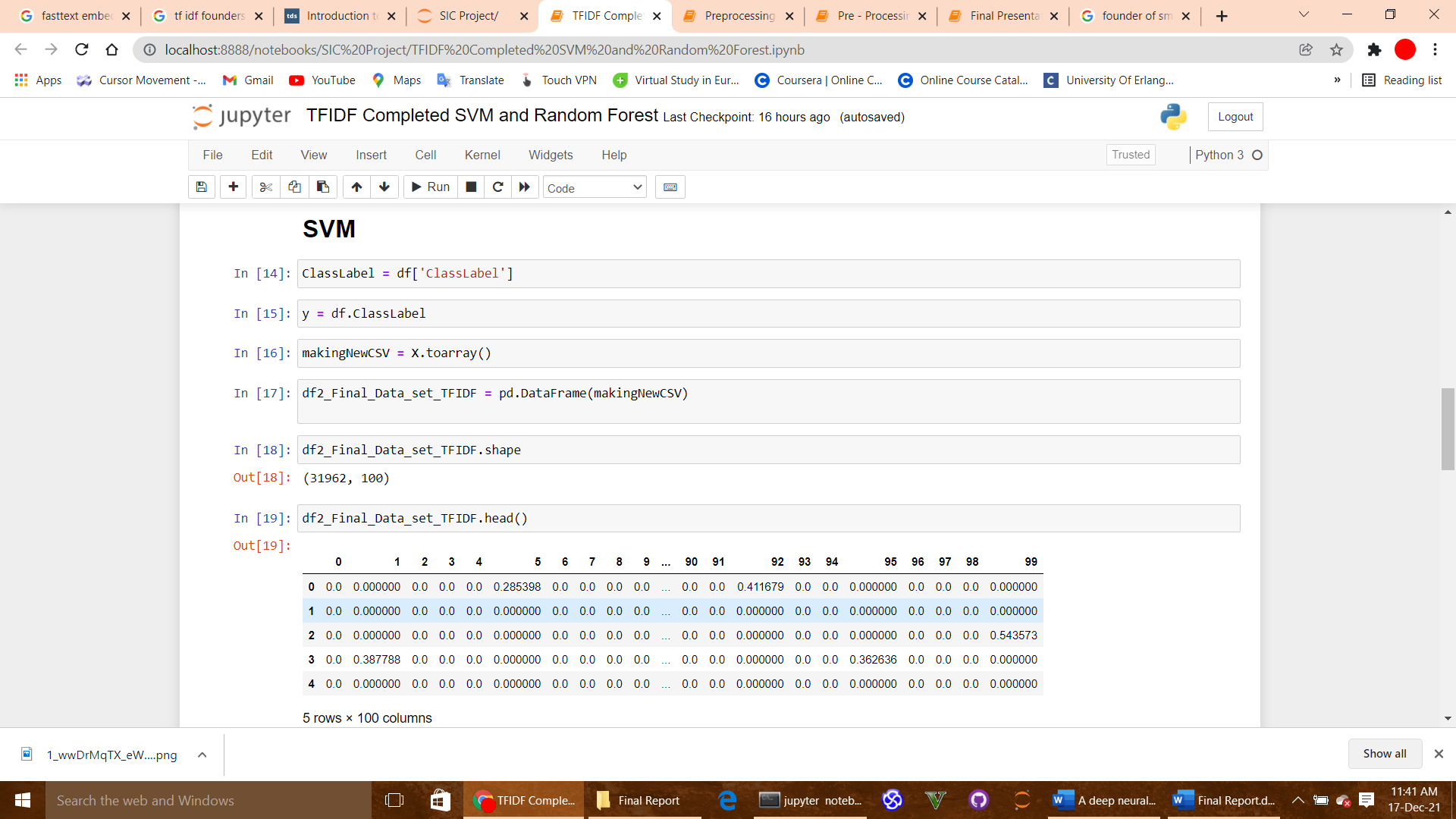


Figure 12: SVM

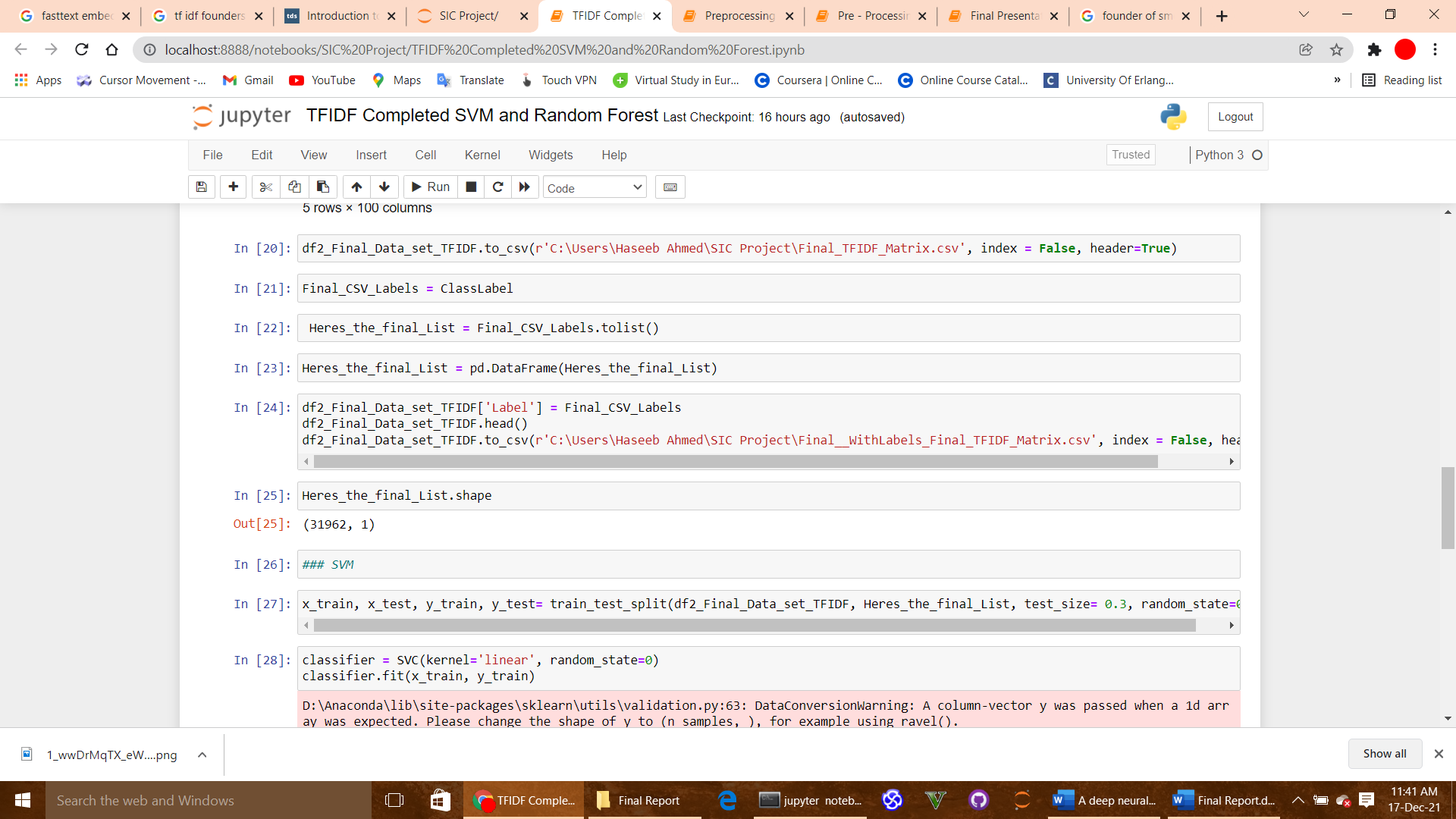


Figure 13: Training the model

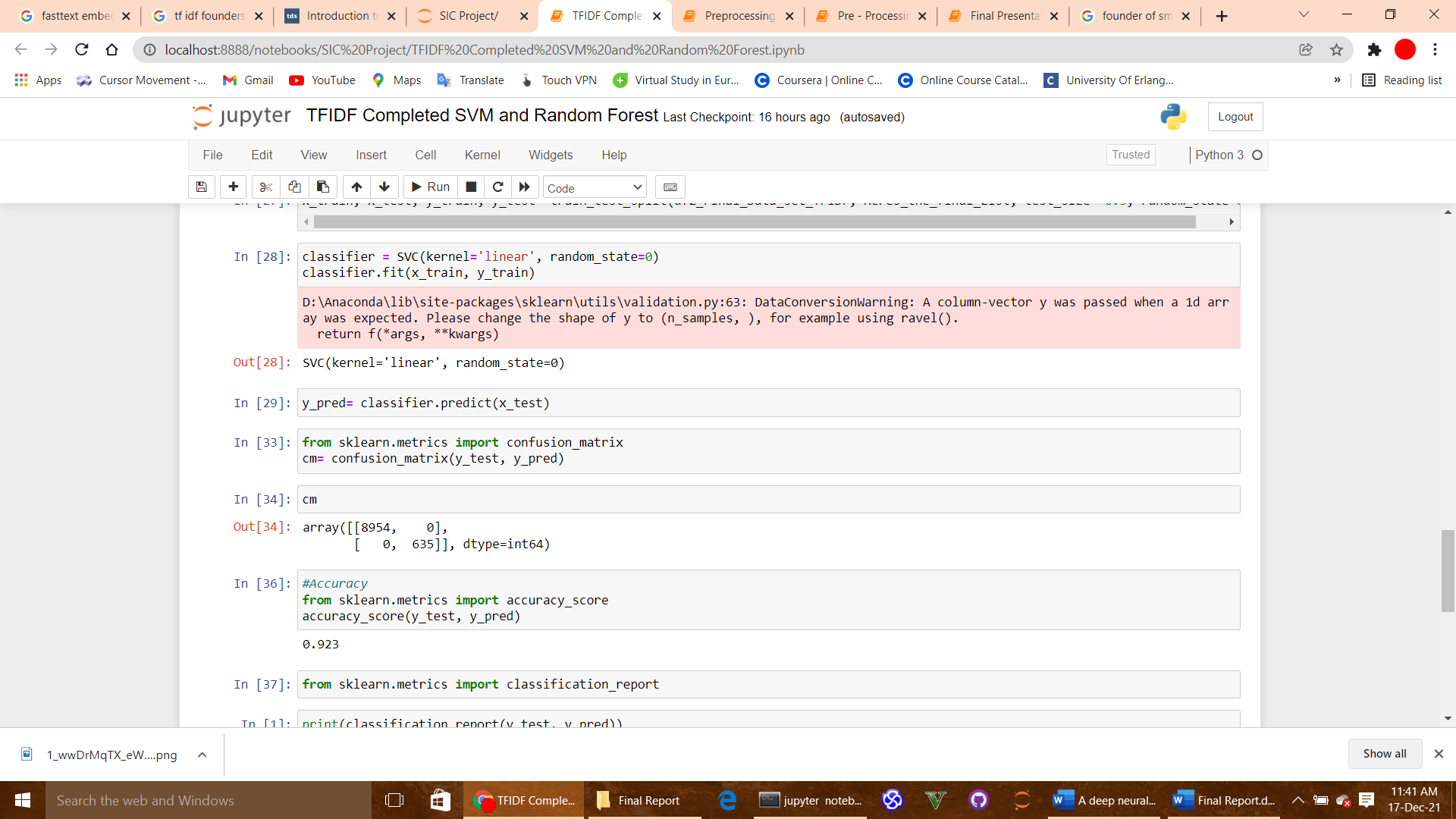


Figure 14: Accuracy Computation

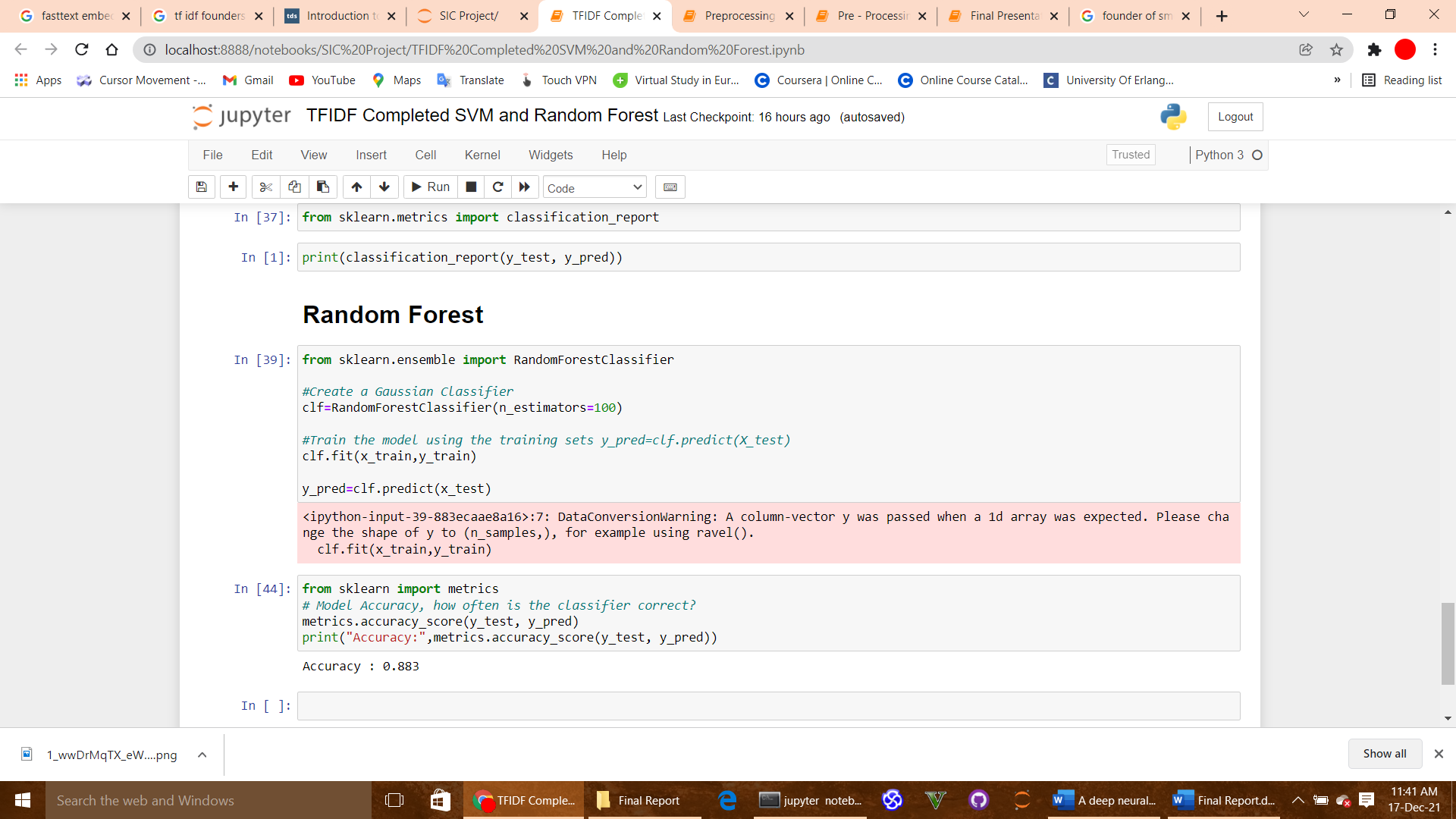


Figure 15: Applying Random Forest

# Result Discussions

## TFIDF

By using the TF IDF embeddings, we have calculated the matrix of 31296 X 100. By using this we applied the SVM Support Vector Machine and Random Forest to calculate its accuracy. By using the SVM on the TF IDF embeddings we get the 0.923 accuracy that means 92.3% accuracy. It means that out of 100 on 92 that system will predict and classify the tweet correctly and 7 out of 100 will not be classified correctly. By using the Random Forest, we get the 0.883 accuracy which means 88.3% accuracy. This means that if using Random Forest, the system will classify the 88 tweets out of 100 correctly and 12 will not be classified correctly out of 100 tweets.

# Conclusion

# References

* 1. [Benesch Susan, Countering dangerous speech to prevent mass violence](http://refhub.elsevier.com/S0950-7051(20)30587-6/sb1) [during Kenya’s 2013 elections, Final Report, 2014, pp. 1–26.](http://refhub.elsevier.com/S0950-7051(20)30587-6/sb1)