REPORT

Plagiarism Detector

MAY 23, 2022

IET LUCKNOW
LUCKNOW

PlagScan
(A PLAGIARISM DETECTOR)

A
Project Report submitted
in the partial fulfillment of the requirements for the award of the degree
of

Bachelor of Technology
in
Information Technology

By

RITIK KUMAR GUPTA 1805220046
RITIK KUMAR 1805213045
SHUBHAM KUMAR 1805213055

Under the guidance of

Parul Mudita

Supervisor: - Dr. Parul Yadav **Co-Supervisor: -** Er. Mudita Sharan

Department of Computer Science and Engineering
Institute of Engineering and Technology, Lucknow
Dr. A.P.J. Abdul Kalam Technical University, Lucknow, Uttar Pradesh.

CONTENTS

Declaration

We hereby declare that this submission is our own work and that, to the best of our belief and knowledge, it contains no material previously published or written by another person or material which to a substantial error has been accepted for the award of any degree or diploma of university or other institute of higher learning, except where the acknowledgement has been made in the text. The project has not been submitted by us at any other institute for requirement of any other degree.

Submitted by

Date

23/05/2022

RITIK KUMAR GUPTA Ritik

1805220046

IT

RITIK KUMAR Ritik

1805213045

IT

SHUBHAM KUMAR [illegible]

1805213055

IT

CERTIFICATE

This is to certify that the project report entitled **PLAGSCAN** presented by **Ritik Kumar Gupta**, **Shubham Kumar** and **Ritik Kumar** in the partial fulfillment for the award of Bachelor of Technology in Information Technology, is a record of work carried out by them under my supervision and guidance at the Department of Computer Science and Engineering at "Institute of Engineering and Technology, Lucknow".

It is also certified that this project has not been submitted at any other Institute for the awardof any other degrees to the best of my knowledge.

Parul
10/06/2022

Dr. Parul Yadav

Department of Computer Science and Engineering

Institute of Engineering and Technology, Lucknow

[illegible]

Er. Mudita Sharan

Department of Computer Science and Engineering

Institute of Engineering and Technology, Lucknow

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to several individuals and organizations for supporting me throughout my project. First, We wish to express my sincere gratitude to our supervisor, Assistant Professor Dr. Parul Yadav and our co-supervisor Assistant Professor(CF) Er. Mudita Sharan for their enthusiasm, patience, insightful comments, helpful information, practical advice and unceasing ideas that have helped us tremendously at all times in my research and writing of this report

We could not have imagined having a better supervisor and co-supervisor in my project.

We also wish to express our sincere thanks to the Dr. Promila Bahadur and whole IT Department for guiding us in all aspects in this journey.

Project Members

Ritik Kumar Gupta

Shubham Kumar

Ritik Kumar

ABSTRACT

Plagiarism remains one of the most significant problems of digitization. In todays world , it is very easy to copy and paste text from one source to other making it a great hindrance to learning. Many steps are taken for it and one of most famous if to create an another technology to defeat this problem ,"Plagiarism Detection".

Many researchers have proposed/designed/implemented many tools and technologies namely,

1. Jaccard Coefficient
2. Cosine Coefficient
3. Overlap Coefficient
4. Levenshtein Distance
5. Hamming distance
6. Dice Coefficient

including many algorithms and explanations to defy this problem.

Specifically talking to Machine Learning field, we have many algorithms like but here we come with an article of Jason and Paul Claugh which motivates us to make this legacy one step forward by using two unique classification algorithms ,which are "n- gram Containment features" and "Largest common subsequence" bind up with a binary classification technique.

Next, we try to make a full length pipeline to deploy this project on Sagemaker platform using EC2 instances, as proposed in paper. This project helps us in identifying all major problems related to plagiarism and even classifies them as light/heavy plagiarism, that will be bonus point after using it.

List of Figures

List of Tables

Chapter 1
INTRODUCTION

Plagiarism is one of the most common problem in the digital world where everyone has access to internet on their finger tips. According to Oxford dictionary, Plagiarism is just copying others work like they are our own without full acknowledgment or copyright issue from the owner. All types of work or materials like printed, manuscripts, published etc. covered under this definition.

Moving in the same road, we have plagiarism in source codes which is a major issue not just for academics but for bigger businesses as well. Talking specifically about the academia programming, students try to cheat codes by web or so called private freelancers available easily on internet which helps them to cheat, making them passive studs.

According to J. Faidi and S.K. Robinson **[1]** , notable alumnus in this field, plagiarism spectrum should be divided into six levels where level 0 represents directly copied the content by copy-paste formula and level 6 belongs to logic in order to achieve near plagiarised content.

Moreover Arwin and Tahaghoghi **[2]** have mentioned that structured based systems rely on the belief that the similarity of two programs can be estimated from the similarity of their structures. Since structured properties of plagiarized documents vary from its original document it is difficult to detect plagiarism when plagiarism level is 4 or higher **[3].**

1.1 Forms of Plagiarism

There are six forms of plagiarism as given below

a) **Paraphrasing plagiarism**: Some small adjustment like words or syntax are changed, still one can identify original text.

b) **Word-for-word plagiarism**: As it is copying of the source text without giving any credit or acknowledgement.

c) **Plagiarism of the form of a source**: Sometimes people use structure of the source (verbatim or rewritten).

d) **Plagiarism of secondary sources**: Sometime without looking at the original content, people quote or give reference using secondary source.

e) **Plagiarism of authorship**: When one put their name directly on someone's else work and show it as his/her own.

f) **Plagiarism of ideas**: People steal ideas only and writing it in their own words.

1.2 Features used for detecting plagiarism

Some Attribute counting techniques used in plagiarism detection systems

a) Jaccard Coefficient

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

b) Cosine Coefficient

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}},$$

c) Overlap Coefficient

$$\text{overlap}(X,Y) = \frac{|X \cap Y|}{\min(|X|,|Y|)}$$

d) Levenshtein Distance

$$d_{a,b}(i,j) = \min \begin{cases} 0 & \text{if } i=j=0, \\ d_{a,b}(i-1,j)+1 & \text{if } i>0, \\ d_{a,b}(i,j-1)+1 & \text{if } j>0, \\ d_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} & \text{if } i,j>0, \\ d_{a,b}(i-2,j-2)+1 & \text{if } i,j>1 \text{ and } a_i = b_{j-1} \text{ and } a_{i-1} = b_j, \end{cases}$$

e) Dice Coefficient

$$DSC = \frac{2|X \cap Y|}{|X| + |Y|}$$

1.3 Research gaps

But none of these technique are so much reliable in the structural properties of source text and hence they are suffering the problems in handling the plagiarism detection challenges.

- Selecting suitable discriminators which reflect plagiarism
- co-incidental similarity due to content or theme
- extract the inconsistencies and then find possible source texts

1.4 Objectives

- In this project, we build a plagiarism detector that helps in handling these challenges by using 2 new players in the field of plagiarism which we discussed in chapter 3.
- Our PlagScan scans a text file and uses binary classification.
- Detecting plagiarism is an important area of research.
- The task is non-trivial and the differences between paraphrased answers and original work are often not so obvious.

1.5 Organization of report

- The report is organized in form of the flow of our working model and breaks into three Jupyter notebook which are then linked to two python files and one training file.
- Our three notebooks deals with all functions from collecting data to organizing dataset into classification criterion of either plagiarized or not.
- Sagemaker plays an important role in this and we discuss about in upcoming notebooks.

It gives the result that the text file is plagiarized or not which depends on the provided source text. Detecting plagiarism is an important area of research; it is the non-trivial task and the differences between paraphrased answers and original work are often not so obvious which depends on the similarity of the text file to a provided source file.

Chapter 2

LITERATURE REVIEW

Plagiarism is a major problem for research. There are, however, divergent views on how to define plagiarism and on what makes plagiarism reprehensible. In this paper we explicate the concept of "plagiarism" and discuss plagiarism normatively in relation to research. Tools/algorithms/methodology are discussed as follows.

a) Image Recognition system using Keras and Ipython

Deep Neural Network (DNN) IS the latest advancement in the deep learning made task simple for image recognition by as deeply as possible learning. Deep learning is a subdivision of machine learning algorithms, which are excellent in identifying patterns, but usually require more data. The most popular technique used in improving the accuracy of image classification is the Convolutional neural network (CNN). CNN is a special type of neural network that works just like the normal neural network, which initially has a convolution layer.

b) Matrix Multiplication and addition operations using TensorFlow

In the paper, the methods invented by Karatsuba and Strassen are analyzed and implemented; theoretical and real time is calculated. Afterwards the Karatsuba and the Strassen algorithms are merged, and combining these two algorithms a new approach is designed, which can be considered as a method for reducing the time complexity. **[6]**

c) Use of nltk, Tokenization concepts in selection of consecutive

NLTK contains a module called tokenize() which further classifies into two sub-categories: Word tokenize: We use the word_tokenize() method to split a sentence into tokens **or** words. Sentence tokenize: We use the sent_tokenize() method to split a document or paragraph into sentences **[3]**

d) Support Vector Machines

This issue's collection of essays should help familiarize our readers with this interesting new racehorse in the Machine Learning stable. Bernhard Scholkopf, in an introductory overview, points out that a particular advantage of SVMs over other learning algorithms is that it can be analyzed theoretically using concepts from computational learning theory, and at the same time can achieve good performance when applied to real problems**. [4]**

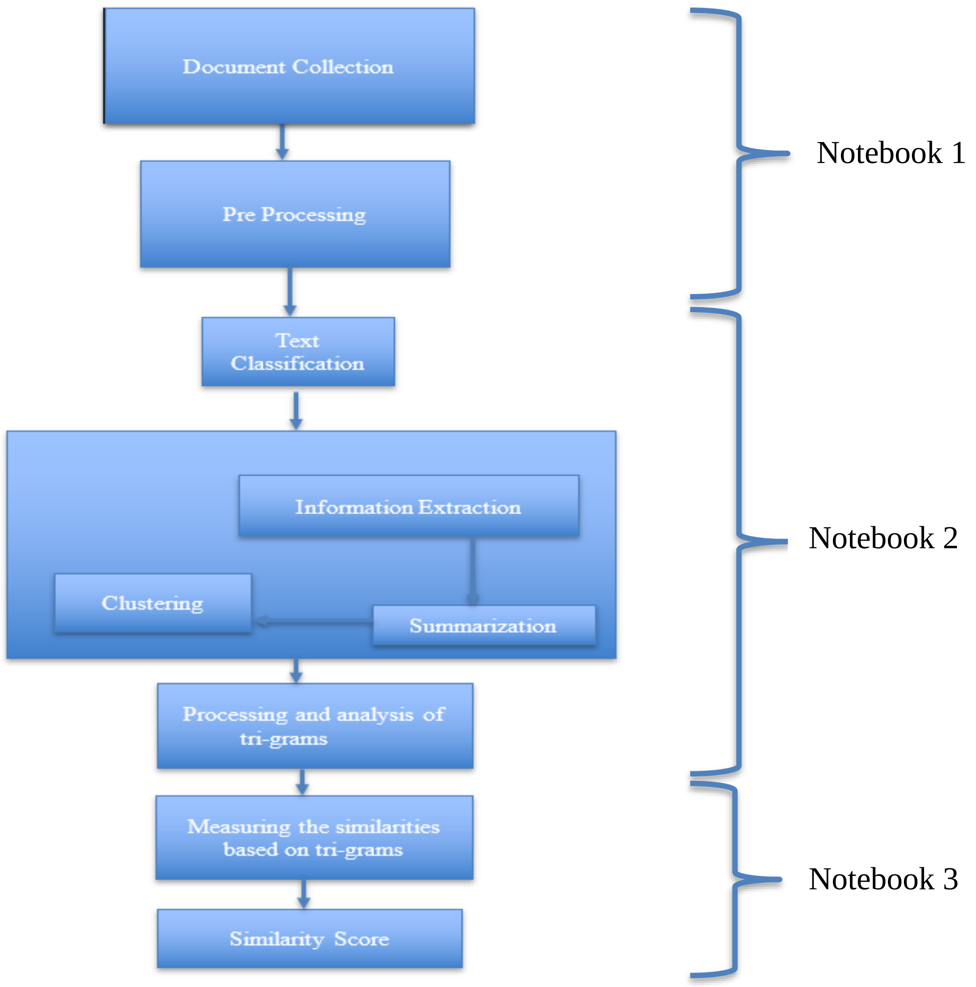
As a result of our normative analysis, we suggest that what makes plagiarism reprehensible and these techniques will help us a lot in determining plagiarism and achieve our goal.

Chapter 3

METHODOLOGY

Dataset is taken from open source project of University of Sheffield named as White Rose in the field of publishing and deploying load data in the Machine learning Resource evaluation. Our project is to detect plagiarism in the text files and classify them as either plagiarized or not in the binary format.

This project will be broken down into three main notebooks



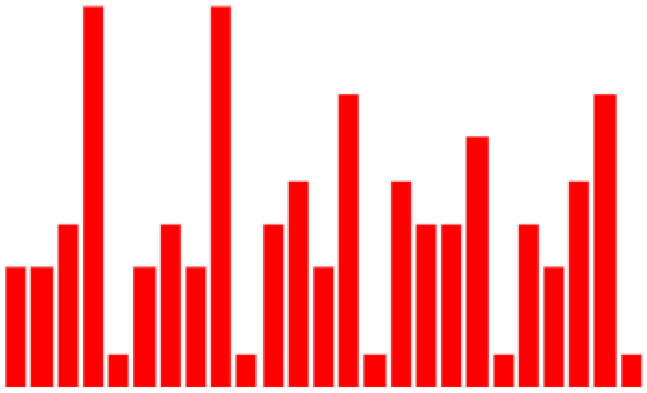
In the notebook 1, we discuss regarding the data pickup, data loading, data cleanising and data preprocessing to get a graphical and tabular insight into the data.

1. Notebook 1 (DataExpl.ipynb)

Steps are given below

a) Using the integrated Jupyter Lab as our prime IDE for implementation, we have divided our source code in three major Jupyter notebooks, in the first notebook, we firstly load the corpus of plagiarism text data.

b) Firstly, we load the corpus data and explore the existing data features and data distribution using popular python libraries on an user-created ipython kernel "my_py_39_ds_env". The kernel is created in a conda vitual core environment with python version 3.9.x



i. Fig 3.1.1

c) Using a new terminal in AWS Sagemaker notebook instance of m1.t3.medium we utilize the following command set using conda package management tool, already a part of AWS Sagemaker

```
conda create -n py_39_ds_env python=3.9
```

d) In order to keep this terminal intact for other notebook usages, we have installed ipykernel using same conda virtual environment and install newly created ipykernel with our username and display name by giing the following command set in the raw terminal

```
python -m ipykernel install --user --name py_39_ds_env --display- name "my_py_39_ds_env"
```

e) The ARN(Amazon resource name) for our notebook-1 is in U.S. East Virgina and have id

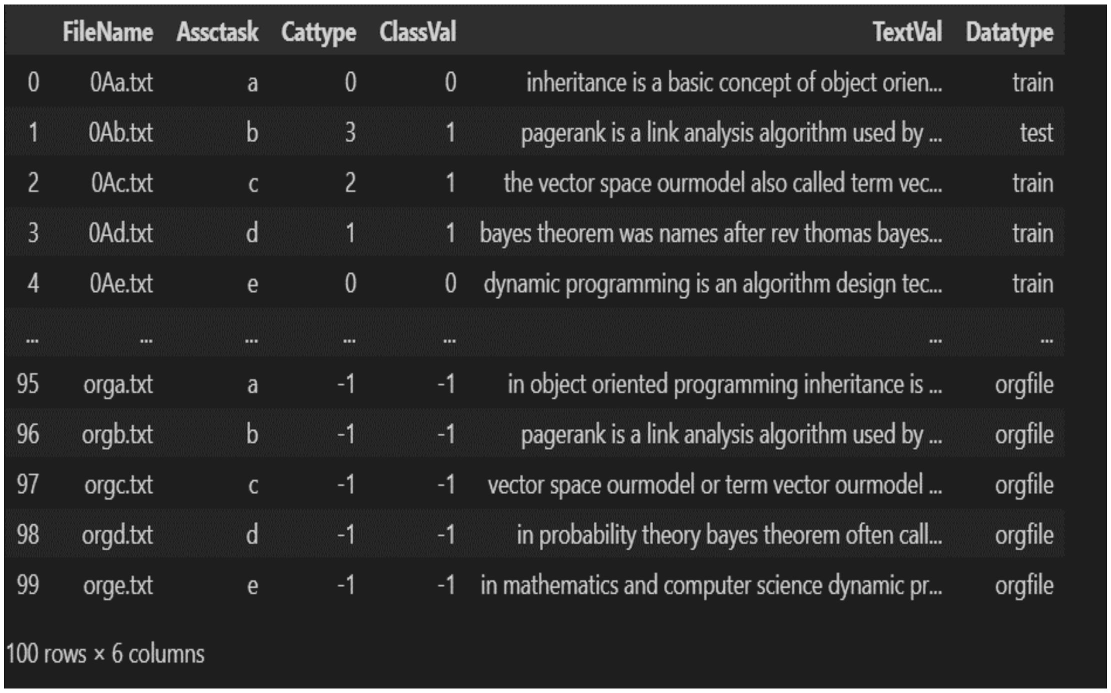
arn:aws:sagemaker:us-east-1:118083503361:notebook-instance/plagsscan

f) The volume size taken is default value minimum 5 GB EBS, EBS is taken to ensure proper working of notebook on cloud hosting deployement, PragScan is created on IAM role with limited access keys and no VPC or tag values so as to keep it at minimum cost.

g) Using matplotlib like plotting python libraries including in numpy and pandas we try to give a visulaise effect like GuiTkinter or GTK.

h) questions are to be treated as 5 asstask i.e.

- Task a
- Task b
- Task c
- Task d
- Task e



	FileName	Assctask	Cattype	ClassVal	TextVal	Datatype
0	0Aa.txt	a	0	0	inheritance is a basic concept of object orien...	train
1	0Ab.txt	b	3	1	pagerank is a link analysis algorithm used by ...	test
2	0Ac.txt	c	2	1	the vector space ourmodel also called term vec...	train
3	0Ad.txt	d	1	1	bayes theorem was names after rev thomas bayes...	train
4	0Ae.txt	e	0	0	dynamic programming is an algorithm design tec...	train
...	...	...	...	...	...	...
95	orga.txt	a	-1	-1	in object oriented programming inheritance is ...	orgfile
96	orgb.txt	b	-1	-1	pagerank is a link analysis algorithm used by ...	orgfile
97	orgc.txt	c	-1	-1	vector space ourmodel or term vector ourmodel ...	orgfile
98	orgd.txt	d	-1	-1	in probability theory bayes theorem often call...	orgfile
99	orge.txt	e	-1	-1	in mathematics and computer science dynamic pr...	orgfile

100 rows × 6 columns

Table 3.1.1

Each task, A-E, is about a topic like "What is inheritance in object-oriented programming?” **[5].**

Every file has an associated plagiarism label/category:

1. nearplg:
   An answer is plagiarized and it is nearly plagiared content.

2. lightplg:
   An answer is plagiarized; is is lightly plagiarized content.

3. majorplg:
   An answer is plagiarized; it is highly plagiarized i.e. directly copy pasted from source text without any efforts.

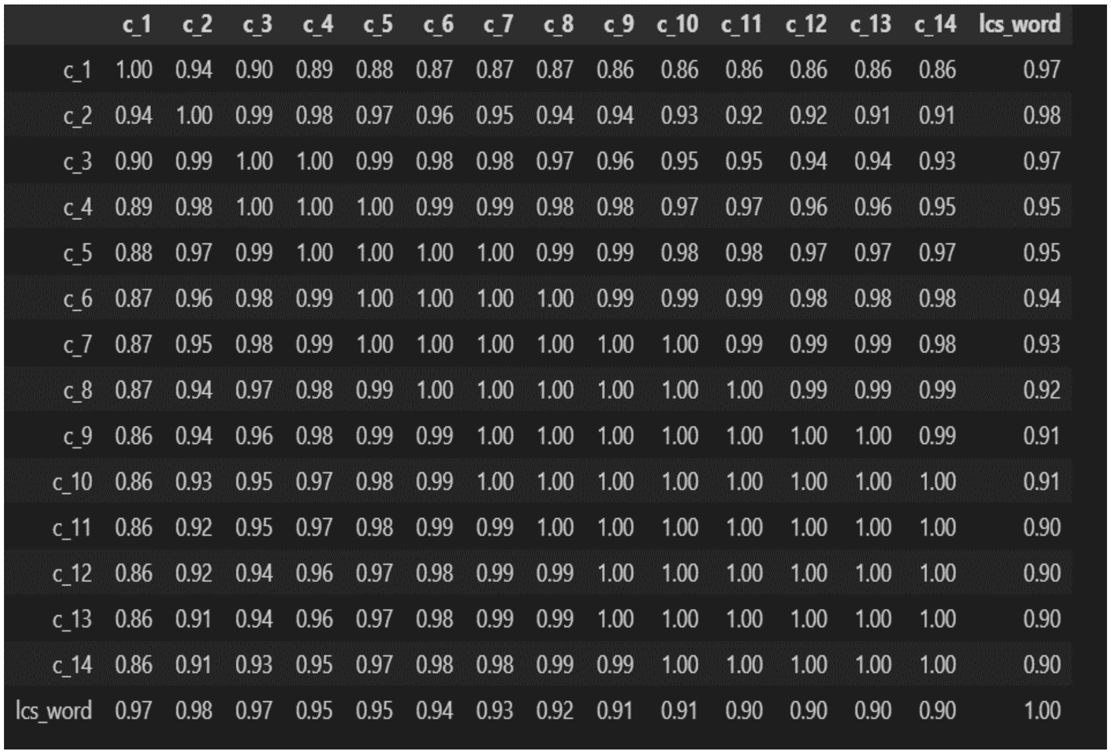
4. notplg:
   An answer is not plagiarized; this means source text is not used to create this answer.

5. org:
   This is a specific original answer.

In this notebook, we use our feature extraction players LCS and containment features to get the maximum output from the dataset we created in previous notebook and use them to create our training and testing dataset.

2. Notebook 2: (Feature Engineering)

Notebook 2 deals with our preparation of dataset and dataframes so as to deploy it afterwards on Sagemaker **[6].**



	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_10	c_11	c_12	c_13	c_14	lcs_word
c_1	1.00	0.94	0.90	0.89	0.88	0.87	0.87	0.87	0.86	0.86	0.86	0.86	0.86	0.86	0.97
c_2	0.94	1.00	0.99	0.98	0.97	0.96	0.95	0.94	0.94	0.93	0.92	0.92	0.91	0.91	0.98
c_3	0.90	0.99	1.00	1.00	0.99	0.98	0.98	0.97	0.96	0.95	0.95	0.94	0.94	0.93	0.97
c_4	0.89	0.98	1.00	1.00	1.00	0.99	0.99	0.98	0.98	0.97	0.97	0.96	0.96	0.95	0.95
c_5	0.88	0.97	0.99	1.00	1.00	1.00	1.00	0.99	0.99	0.98	0.98	0.97	0.97	0.97	0.95
c_6	0.87	0.96	0.98	0.99	1.00	1.00	1.00	1.00	0.99	0.99	0.99	0.98	0.98	0.98	0.94
c_7	0.87	0.95	0.98	0.99	1.00	1.00	1.00	1.00	1.00	1.00	0.99	0.99	0.99	0.98	0.93
c_8	0.87	0.94	0.97	0.98	0.99	1.00	1.00	1.00	1.00	1.00	1.00	0.99	0.99	0.99	0.92
c_9	0.86	0.94	0.96	0.98	0.99	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	0.91
c_10	0.86	0.93	0.95	0.97	0.98	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.91
c_11	0.86	0.92	0.95	0.97	0.98	0.99	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.90
c_12	0.86	0.92	0.94	0.96	0.97	0.98	0.99	0.99	1.00	1.00	1.00	1.00	1.00	1.00	0.90
c_13	0.86	0.91	0.94	0.96	0.97	0.98	0.99	0.99	1.00	1.00	1.00	1.00	1.00	1.00	0.90
c_14	0.86	0.91	0.93	0.95	0.97	0.98	0.98	0.99	0.99	1.00	1.00	1.00	1.00	1.00	0.90
lcs_word	0.97	0.98	0.97	0.95	0.95	0.94	0.93	0.92	0.91	0.91	0.90	0.90	0.90	0.90	1.00

Table 3.2.1

The steps followed are as follows

- Preprocessing the text data, consisting of 100 .txt files of 128 kb storage value of HDD and 124 Kb storage value on S3 buckets.

- Defining the features for comparing the similarities in dataset which includes two players
    - Containment Features
    - Longest Common Subsequence

1) Containment

It is defined as the intersection of the n-gram value count of source Text with the n-gram value count of the modified text divided by the n-gram word count of the modified text.

In the fields of computational linguistics and probability, an n-gram (sometimes also called Q-gram) is a contiguous sequence of n items from a given sample of text or speech. The items can be phonemes, syllables, letters, words or base pairs according to the application. The n-grams typically are collected from a text or speech corpus. When the items are words, n-grams may also be called shingles**. [6]**

Containment is based upon the following parameters:

1.1) A given Data Frame,
1.2) An answer filename
1.3) An n-gram length, n

$$\frac{\sum count(\text{ngram}_A) \cap count(\text{ngram}_S)}{\sum count(\text{ngram}_A)}$$

2) Longest Common Subsequence

LCS Longest common subsequence is longest subsequence which is common in all sequences, also in the original sequences it is not necessary that elements are placed in consecutive order.**[6]**

As there are two given sequences S1 and S2 , there is a common subsequence of S1 and S2 which is Z if Z is a subsequence of both S1 and S2.

Now Z is a strictly increasing sequence of the indices of both S1 and S2. In a strictly increasing sequence, the indices of S1 and S2 chosen from the original sequences must be in ascending order in Z.

If

S1 = {B, C, D, A, A, C, D}

Then, {A, D, B} cannot be a subsequence of S1 as the order of the elements is not the same (ie. not strictly increasing sequence)

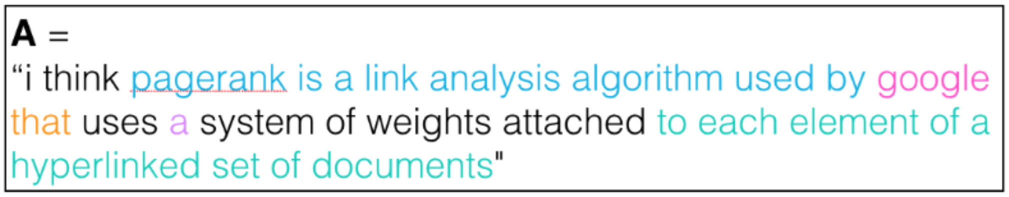
If

S1 = {B, C, D, A, A, C, D}

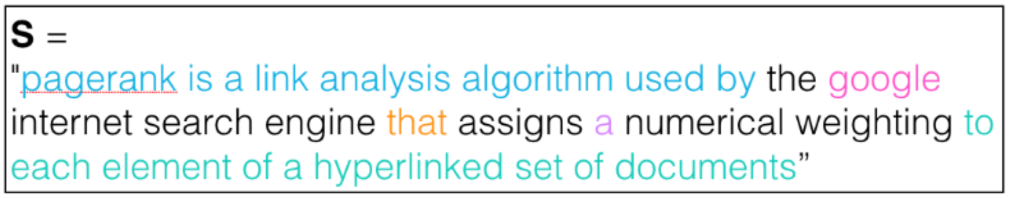
S2 = {A, C, D, B, A, C}

So, the longest common subsequence is. {C, D, A, C} from the common subsequences {B, C}, {C, D, A, C}, {D, A, C}, {A, A, C}, {A, C}, {C, D}.

For using LCS, we can use many methods like Overlapping, Optimal Sequence etc. but here we use Matrix Rule of LCS to find similarity features as proposed in above mentioned research pdf.



A =
“i think pagerank is a link analysis algorithm used by google that uses a system of weights attached to each element of a hyperlinked set of documents"



S =
"pagerank is a link analysis algorithm used by the google internet search engine that assigns a numerical weighting to each element of a hyperlinked set of documents”

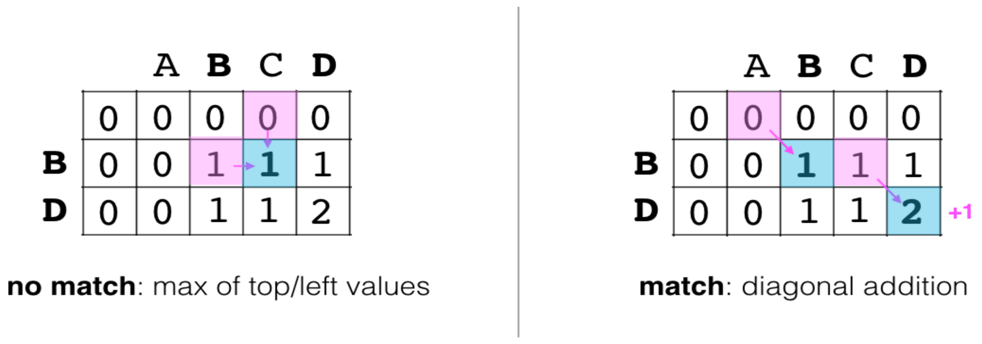


Fig 3.2.2

- Selecting the good features is always appreciated and we extract our good features utilizing the correlation LCS+ n-Gram matrix which is shown in [fig 2]

- Creation of training and testing datasets in form of .csv files with proper naming conventions and proper column and row formatting conventions as according to Sagemaker ret hotdocs **[7].**

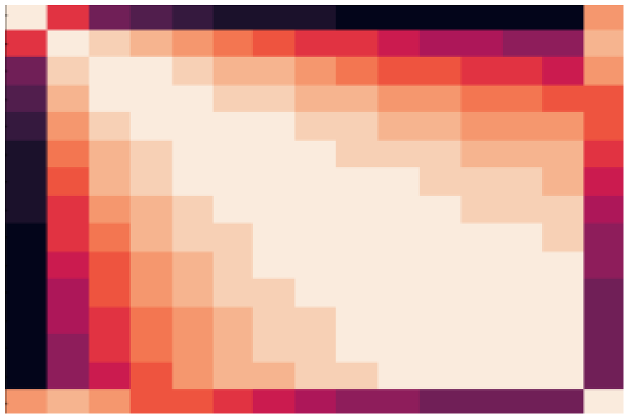
We also tried to visualize this in form of Gui seaborn plot as shown in [fig 3] where for every lcs_word value we have corresponding lcs_value_root showing its correlation value with each -other.

4 features are utilizing in training and corresponding testing dataset which are c_1, c_2, c_3, lcs_word where c_1, c_2, c_3 represents 1-gram, 2-gram, 3-gram containment values respectively for the text modified

.

The four log groups created ,out of which two are directly related to notebook 2 and these are

- /aws/sagemaker/ProcessingJobs
- /aws/sagemaker/NotebookInstances

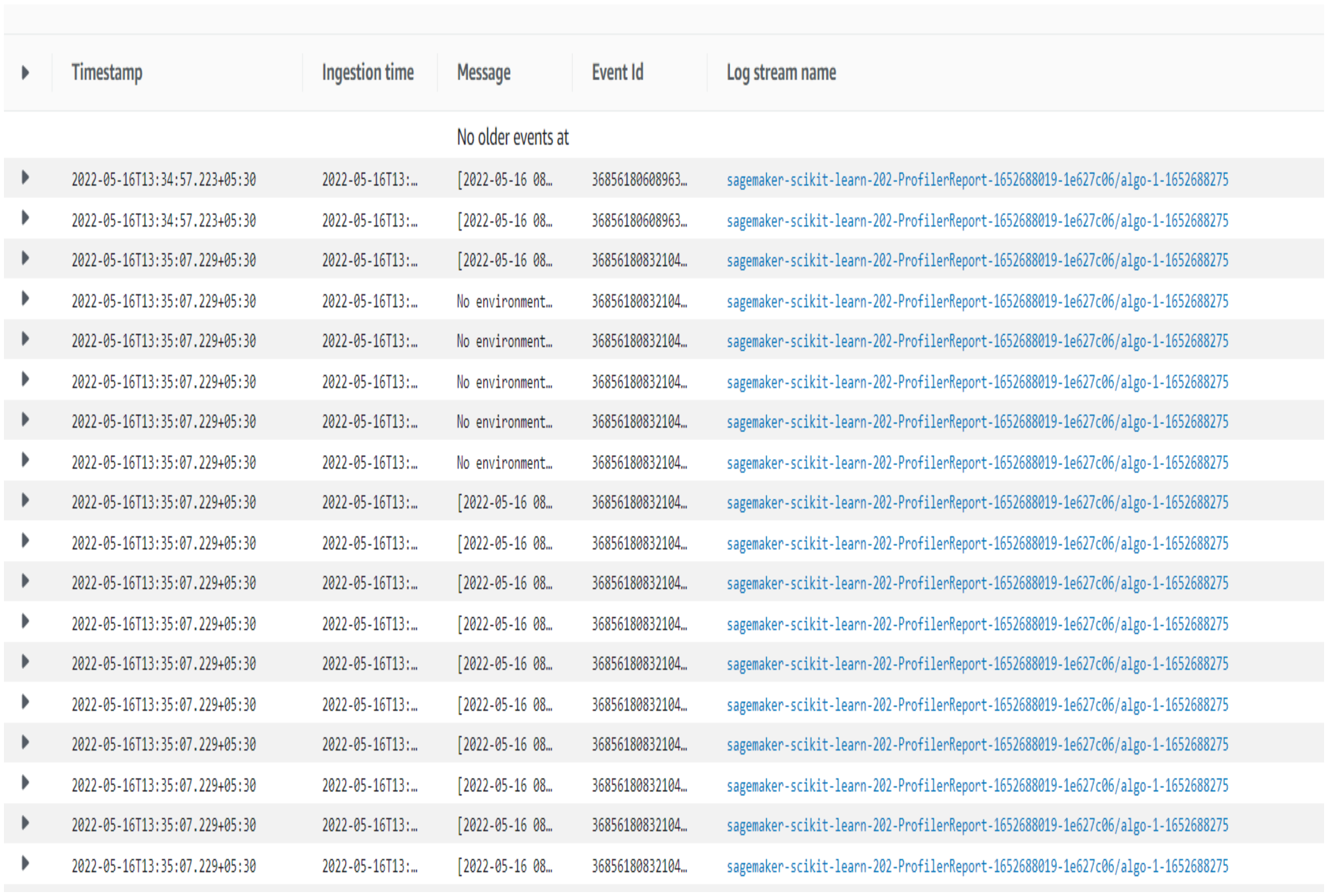


Every log stream is associated with its log stream,

/aws/sagemaker/ProcessingJobs consists of one log stream with scikit learn library as its prefix and profileid and algo id as its suffix

sagemaker-scikit-learn-202-ProfilerReport-1652688019-1e627c06/algo-1-1652688275

Every log stream consists of log events at every timestamp we edit in the .conf file at the creation of notebook instance having their own ingestion time and event id with sub log-streams.



Timestamp	Ingestion time	Message	Event Id	Log stream name
		No older events at		
2022-05-16T13:34:57.223+05:30	2022-05-16T13:…	[2022-05-16 08…	36856180608963…	sagemaker-scikit-learn-202-ProfilerReport-1652688019-1e627c06/algo-1-1652688275
2022-05-16T13:34:57.223+05:30	2022-05-16T13:…	[2022-05-16 08…	36856180608963…	sagemaker-scikit-learn-202-ProfilerReport-1652688019-1e627c06/algo-1-1652688275
2022-05-16T13:35:07.229+05:30	2022-05-16T13:…	[2022-05-16 08…	36856180832104…	sagemaker-scikit-learn-202-ProfilerReport-1652688019-1e627c06/algo-1-1652688275
2022-05-16T13:35:07.229+05:30	2022-05-16T13:…	No environment…	36856180832104…	sagemaker-scikit-learn-202-ProfilerReport-1652688019-1e627c06/algo-1-1652688275
2022-05-16T13:35:07.229+05:30	2022-05-16T13:…	No environment…	36856180832104…	sagemaker-scikit-learn-202-ProfilerReport-1652688019-1e627c06/algo-1-1652688275
2022-05-16T13:35:07.229+05:30	2022-05-16T13:…	No environment…	36856180832104…	sagemaker-scikit-learn-202-ProfilerReport-1652688019-1e627c06/algo-1-1652688275
2022-05-16T13:35:07.229+05:30	2022-05-16T13:…	No environment…	36856180832104…	sagemaker-scikit-learn-202-ProfilerReport-1652688019-1e627c06/algo-1-1652688275
2022-05-16T13:35:07.229+05:30	2022-05-16T13:…	No environment…	36856180832104…	sagemaker-scikit-learn-202-ProfilerReport-1652688019-1e627c06/algo-1-1652688275
2022-05-16T13:35:07.229+05:30	2022-05-16T13:…	[2022-05-16 08…	36856180832104…	sagemaker-scikit-learn-202-ProfilerReport-1652688019-1e627c06/algo-1-1652688275
2022-05-16T13:35:07.229+05:30	2022-05-16T13:…	[2022-05-16 08…	36856180832104…	sagemaker-scikit-learn-202-ProfilerReport-1652688019-1e627c06/algo-1-1652688275
2022-05-16T13:35:07.229+05:30	2022-05-16T13:…	[2022-05-16 08…	36856180832104…	sagemaker-scikit-learn-202-ProfilerReport-1652688019-1e627c06/algo-1-1652688275
2022-05-16T13:35:07.229+05:30	2022-05-16T13:…	[2022-05-16 08…	36856180832104…	sagemaker-scikit-learn-202-ProfilerReport-1652688019-1e627c06/algo-1-1652688275
2022-05-16T13:35:07.229+05:30	2022-05-16T13:…	[2022-05-16 08…	36856180832104…	sagemaker-scikit-learn-202-ProfilerReport-1652688019-1e627c06/algo-1-1652688275
2022-05-16T13:35:07.229+05:30	2022-05-16T13:…	[2022-05-16 08…	36856180832104…	sagemaker-scikit-learn-202-ProfilerReport-1652688019-1e627c06/algo-1-1652688275
2022-05-16T13:35:07.229+05:30	2022-05-16T13:…	[2022-05-16 08…	36856180832104…	sagemaker-scikit-learn-202-ProfilerReport-1652688019-1e627c06/algo-1-1652688275
2022-05-16T13:35:07.229+05:30	2022-05-16T13:…	[2022-05-16 08…	36856180832104…	sagemaker-scikit-learn-202-ProfilerReport-1652688019-1e627c06/algo-1-1652688275
2022-05-16T13:35:07.229+05:30	2022-05-16T13:…	[2022-05-16 08…	36856180832104…	sagemaker-scikit-learn-202-ProfilerReport-1652688019-1e627c06/algo-1-1652688275
2022-05-16T13:35:07.229+05:30	2022-05-16T13:…	[2022-05-16 08…	36856180832104…	sagemaker-scikit-learn-202-ProfilerReport-1652688019-1e627c06/algo-1-1652688275

Table 3.2.2

We can also set the expiry timestamp of these log groups and even can delete them so that we don't be charged extra for using cloudwatch logs.

We can even check our resource health status, systematic canaries, RUM Values and even get insights like of container and application, from the cloudwatch console.

In this notebook, we deploy our model to AWS Sagemaker and run on the targeted EC2 instance to get binary classified values of training dataset

3. Notebook 3 (Training and Deploying)

The notebook 3 deals with whole lot of deployment and training our datast to get corrected output from it.
The major steps we followed while creating this notebook are as follows

- Uploading of the training and testing dataset features which we created in notebook 2 to the S3 bucket. S3 bucket is a simple and secure storage that provides objects storage through web service interface using the scalable storage infrastructure.

- Creation of bucket is done by following code values

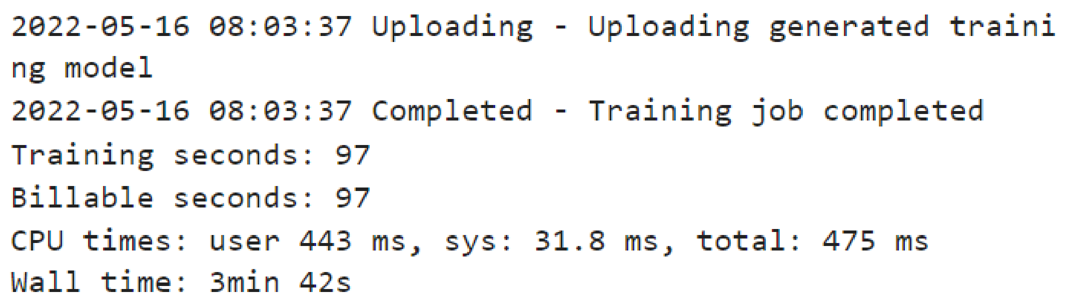
```
sagemaker_session = sagemaker.Session()
role = sagemaker.get_execution_role()
bucket = sagemaker_session.default_bucket()
```

  which automatically creates a "Any size" bucket of minimum 5 Gb for us in the region specified in notebook 1 i.e. US-East-1 ,a particular id is given to the bucket so that it can be accessed easily by other contributors giving access either as "public" or "private" or "inherited" using any git automation tool like gitlab **[8].**

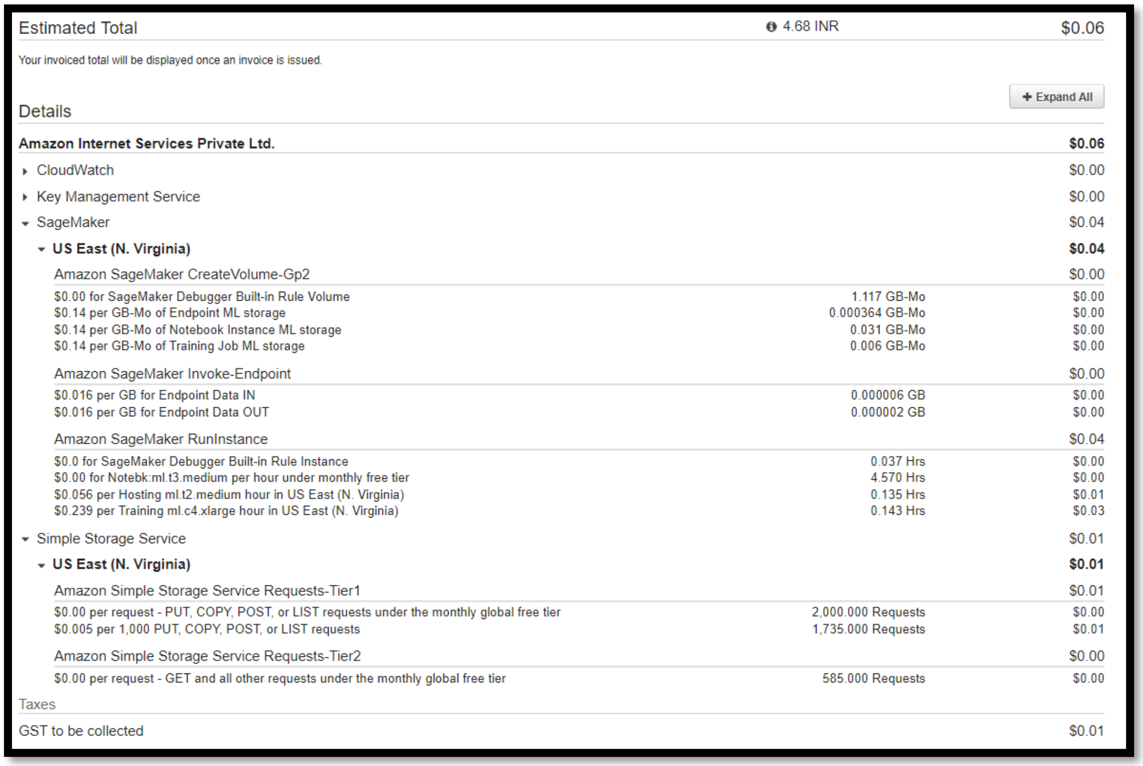
  Permissions by deafault to bucket are as follows

  s3:ListStorageLensConfigurations
  s3:GetStorageLensConfiguration
  s3:GetStorageLensDashboard

  Using AWS access analyzer, we can even download our report for further use in synchronization batch processes.



```
2022-05-16 08:03:37 Uploading - Uploading generated traini
ng model
2022-05-16 08:03:37 Completed - Training job completed
Training seconds: 97
Billable seconds: 97
CPU times: user 443 ms, sys: 31.8 ms, total: 475 ms
Wall time: 3min 42s
```



Estimated Total ⓘ 4.68 INR $0.06

Your invoiced total will be displayed once an invoice is issued.

Details

+ Expand All

Amazon Internet Services Private Ltd.		**$0.06**
▸ CloudWatch		$0.00
▸ Key Management Service		$0.00
▾ SageMaker		$0.04
▾ **US East (N. Virginia)**		**$0.04**
Amazon SageMaker CreateVolume-Gp2		$0.00
$0.00 for SageMaker Debugger Built-in Rule Volume	1.117 GB-Mo	$0.00
$0.14 per GB-Mo of Endpoint ML storage	0.000364 GB-Mo	$0.00
$0.14 per GB-Mo of Notebook Instance ML storage	0.031 GB-Mo	$0.00
$0.14 per GB-Mo of Training Job ML storage	0.006 GB-Mo	$0.00
Amazon SageMaker Invoke-Endpoint		$0.00
$0.016 per GB for Endpoint Data IN	0.000006 GB	$0.00
$0.016 per GB for Endpoint Data OUT	0.000002 GB	$0.00
Amazon SageMaker RunInstance		$0.04
$0.0 for SageMaker Debugger Built-in Rule Instance	0.037 Hrs	$0.00
$0.00 for Notebk:ml.t3.medium per hour under monthly free tier	4.570 Hrs	$0.00
$0.056 per Hosting ml.t2.medium hour in US East (N. Virginia)	0.135 Hrs	$0.01
$0.239 per Training ml.c4.xlarge hour in US East (N. Virginia)	0.143 Hrs	$0.03
▾ Simple Storage Service		$0.01
▾ **US East (N. Virginia)**		**$0.01**
Amazon Simple Storage Service Requests-Tier1		$0.01
$0.00 per request - PUT, COPY, POST, or LIST requests under the monthly global free tier	2,000.000 Requests	$0.00
$0.005 per 1,000 PUT, COPY, POST, or LIST requests	1,735.000 Requests	$0.01
Amazon Simple Storage Service Requests-Tier2		$0.00
$0.00 per request - GET and all other requests under the monthly global free tier	585.000 Requests	$0.00
Taxes		
GST to be collected		$0.01

Fig 3.3.1

- Defining a binary classifier and a classification modek with a training script train.py and for the classification model, we utilize the SVM classification for training and deploying in the sagmeaker

- Finally we have to evaluate our model by calculating predicted and true class labels

  0 represents the file with no plagiarism
  1 represents the file with plagiarism

Then we can modify our search using precision ,recall, f1-score and support values **[9].**

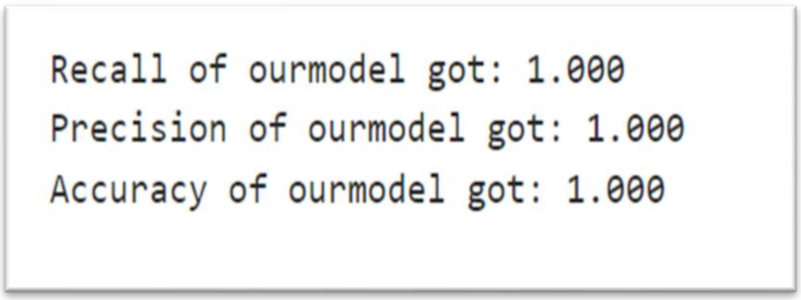
```
Accuracy got: 1.0
              precision    recall  f1-score   support

      notplg       1.00      1.00      1.00        10
           P       1.00      1.00      1.00        15

    accuracy                           1.00        25
   macro avg       1.00      1.00      1.00        25
weighted avg       1.00      1.00      1.00        25


Predicted class labels are:
[1 1 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1 0 1 0 1 1 0 0]

True class labels are:
[1 1 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1 0 1 0 1 1 0 0]
```



```
Recall of ourmodel got: 1.000
Precision of ourmodel got: 1.000
Accuracy of ourmodel got: 1.000
```

1. **entry point:** The path to the Python script SageMaker runs for training
2. **source_dir:** The path to the training script directory source_sklearn

3. **entry_point:** The path to the Python script SageMaker runs for training
4. **source_dir:** The path to the training script directory train_sklearn .

5. **entry_point:** The path to the Python script SageMaker runs for.
6. **source_dir:** The path to the training script directory train_sklearn.

7. **train_instance_count:** The number of training instances
8. **train_instance_type:** The type of SageMaker instance for training.

9. **sagemaker_session:** The session used to train on Sagemaker
10. **hyperparameters (optional):** A dictionary {'name':value, ..} passed

Chapter 4

EXPERIMENTAL RESULTS

On deploying PlagScan on AWS Sagemaker (us-east-1 region) with notable instance of ml.t3.medium along with estimator instance of ml.c4.xlarge and deployer instance of ml.t2.medium on al1-v-1 platform identifier of elastic inference ml.eia1.medium in 5GB of “Any” S3 bucket storage without VPC, we get following measurements on Sklearn.

```
Uploading - Uploading generated training model
Completed - Training job completed
Training seconds: 36
Billable seconds: 36
CPU times: user 343 ms, sys: 23.3 ms, total: 366 ms
Wall time: 2min 41s
```

The accuracy (overall, micro, macro and weighted), precision, recall and f1-score we get in output is as follows.

```
accuracy: 1.0
              precision    recall  f1-score   support

         Non       1.00      1.00      1.00        10
           P       1.00      1.00      1.00        15

   micro avg       1.00      1.00      1.00        25
   macro avg       1.00      1.00      1.00        25
weighted avg       1.00      1.00      1.00        25
```

As we can see our model gets us an accuracy of 1.0 which is a good start i n this project , precision of 1.0 in all micro, macro and weighted average v alues makes the model compatible to get away with all files at any VPC a nd bandwidth value on any type of EC2 instance in ml series.

F1 score is 1.0 in every average weighted value which is a good sign that our model satisfies all conditions of plagiarism identifying with a good no rmalised f1 score which is whatsoever is a must needed task in any ML m odel deployed on cloud hosting development like AWS here.

The billing cost for our 3735 PUT,COPY,POST requests and 585 GET requests of 105 dataset files having 0.031Gb-mo of Notebook Instance ML Storage and 0.006 Gb-mo Training ML Storage on Gp2 EBS with 0.135 hours of hosting ml.t2.medium and 0.143 hours of training ml.c4.xlarge on us-east-1 region zone is as follows.

$0.06 (4.68 INR ac 21-05-22)

Chapter 5

CONCLUSIONS

5.1 Conclusions

Avoiding plagiarism is important and a must needed issue for digital era today and we have to tackle with a good ease and precised techniques. Our model suggests two new features but more can be thought of with good understanding of ML knowledge.

It is important to accept the contributions and information made by other person. We are not deceiving its reader person and making believe that the work is ours. Plagiarism can be thought of as just not a problem but an issue which needs to be handled together with good means.

It is useful to Keep a good record of the sources that we use and to avoid plagiarism we can keep footnotes, work cited lists , and annotated bibliography which can help us to avoid all the problems.

Moreover concepts like jaccard coefficient, picard coefficient etc can also help us in determining the plagiarism.

It is important to keep a check on plagiarism which will encourage the students to believe that they can do their work by themselves, to trust they can be an independent thinkers. Platforms like AWS can be a useful help when we have to deploy our model on the cloud hosting service at a minimum cost but a good accuracy, precision and f1 score value.

5.2 Future Works

Utilize a different and larger dataset to see if this model can be extended to other types of plagiarism. Use language or character-level analysis to find different (and more) similarity features.

Write a complete pipeline function that accepts a source text and submitted text file, and classifies the submitted text as plagiarized or not. Use API Gateway and a lambda function to deploy your model to a web application.

Including the feature of OCR as a open source code on a git version system by which our model can also be able to detect image text and check for the content either plagiarized or not like it will be helpful in cases of pdf files.

Reducing the wall time period for training dataset on Sagemaker and Reduce time and space complexities of user defined functions utilized in helpers.py, problem_unittests.py and train.py file **[Appendix-1]**

Fig 5.2.1

REFERENCES

[1] Clough, P. and Stevenson, M. Developing A Corpus of Plagiarised Short Answers, Language Resources and Evaluation: Special Issue on Plagiarism and Authorship Analysis, In Press

[2] D. Isoc, "Preventing plagiarism in engineering education and research," 2014 International Symposium on Fundamentals of Electrical Engineering (ISFEE), 2014, pp. 1-7, doi: 10.1109/ISFEE.2014.7050618.

[3] J. Yeung, S. Wong, A. Tam and J. So, "Integrating Machine Learning Technology to Data Analytics for E-Commerce on Cloud," 2021 Third World Conference on Smart Trends in Systems Security and Sustainablity (WorldS4), 2021, pp. 105-109, doi: 10.1109/WorldS4.2019.8904026.

[4] Clough, P., Stevenson, M. Developing a corpus of plagiarised short answers, Language Resources and Evaluation, 45 (1), pp. 5-24

[5] Clough, P., Stevenson, M. Developing a corpus of plagiarised short answers. *Lang Resources & Evaluation* 45, 5–2.

[6] P. H. M. Piratelo et al., "Convolutional neural network applied for object recognition in a warehouse of an electric company," 2021 14th IEEE International Conference on Industry Applications (INDUSCON), 2021, pp. 293-299, doi: 10.1109/INDUSCON51756.2021.9529716.

[7] V. B. Kreyndelin and E. D. Grigorieva, "Fast Matrix Multiplication Algorithm for a Bank of Digital Filters," 2021 Systems of Signal Synchronization, Generating and Processing in Telecommunications (SYNCHROINFO, 2021, pp. 1-5, doi: 10.1109/SYNCHROINFO51390.2021.9488350.

[8] M. Sahi, M. A. Maruf, A. Azim and N. Auluck, "A Framework for Partitioning Support Vector Machine Models on Edge Architectures," 2021 IEEE International Conference on Smart Computing (SMARTCOMP), 2021, pp. 293-298, doi: 10.1109/SMARTCOMP52413.2021.00062.

[9] G. K. Masilamani and R. Valli, "Art Classification with Pytorch Using Transfer Learning," 2021 International Conference on System, Computation, Automation and Networking (ICSCAN), 2021, pp. 1-5, doi: 10.1109/ICSCAN53069.2021.9526457.

Appendix-1

- helpers.py

```
import re
import operator
import pandas as pd


def  crtdtype(df,  train_value,  tvalue,  datatype_var,  cmpdfcol,  opcmp,
vcmp,
             sampleno, sampleseed):
    dfsbset = df[opcmp(df[cmpdfcol], vcmp)]
    dfsbset = dfsbset.drop(columns=[datatype_var])
    dfsbset.loc[:, datatype_var] = train_value
          df_sampled    =    dfsbset.groupby(['Assctask',    cmpdfcol],
group_keys=False).apply(
                     lambda    x:    x.sample(min(len(x),    sampleno),
random_state=sampleseed))
    df_sampled = df_sampled.drop(columns=[datatype_var])
    df_sampled.loc[:, datatype_var] = tvalue
    for index in df_sampled.index:
        dfsbset.loc[index, datatype_var] = tvalue
    for index in dfsbset.index:
        df.loc[index, datatype_var] = dfsbset.loc[index, datatype_var]


def trtsdf(clean_df, random_seed=100):
   ndf = clean_df.copy()
   ndf.loc[:, 'Datatype'] = 0
   crtdtype(ndf, 1, 2, 'Datatype', 'Cattype',
             operator.gt, 0, 1, random_seed)
   crtdtype(ndf, 1, 2, 'Datatype', 'Cattype',
             operator.eq, 0, 2, random_seed)
   mapping = {0: 'orgfile', 1: 'train', 2: 'test'}
   ndf.Datatype = [mapping[item] for item in ndf.Datatype]
   return ndf


def processfile(file):
    txtall = file.read().lower()
    txtall = re.sub(r"[^a-zA-Z0-9]", " ", txtall)
    txtall = re.sub(r"\t", " ", txtall)
    txtall = re.sub(r"\n", " ", txtall)
    txtall = re.sub("  ", " ", txtall)
    txtall = re.sub("   ", " ", txtall)
    return txtall


def crtxtcol(df, file_directory='data/'):
    textdf = df.copy()
    text = []
    for row_i in df.index:
        filename = df.iloc[row_i]['FileName']
        file_path = file_directory + filename
        with open(file_path, 'r', encoding='utf-8', errors='ignore') as
file:
            file_text = processfile(file)
            text.append(file_text)
    textdf['TextVal'] = text
    return textdf


def getanssrctxt(df, ansfile):
    srcfname = 'org' + ansfile[-5:]
    sidxno = df[df['FileName'] == srcfname].index.values.astype(int)[
        0]
    aidxno = df[df['FileName'] == ansfile].index.values.astype(int)[0]
    stxt = df.loc[sidxno, 'TextVal']
    atxt = df.loc[aidxno, 'TextVal']
    return stxt, atxt
```

- train.py

```
from __future__ import print_function
from sklearn.externals import joblib
from sklearn.svm import SVC
import argparse
import os
import pandas as pd


def model_fn(ourmodel_dir):
    print("Model Loading")
         ourmodel    =    joblib.load(os.path.join(ourmodel_dir,
"ourmodel.joblib"))
```

```
    print("Done.... :)")
    return ourmodel


if __name__ == '__main__':
    psr = argparse.ArgumentParser()
    psr.add_argument('--ourmodel-dir', type=str,
                     default=os.environ['SM_MODEL_DIR'])
    psr.add_argument('--data-dir', type=str,
                     default=os.environ['SM_CHANNEL_TRAIN'])
    args = psr.parse_args()
    traindirectory = args.data_dir
    datatrain = pd.read_csv(os.path.join(
        traindirectory, "train.csv"), header=None, names=None)
    train_y = datatrain.iloc[:, 0]
    train_x = datatrain.iloc[:, 1:]
    ourmodel = SVC(gamma=2, C=1)
    ourmodel.fit(train_x, train_y)
              joblib.dump(ourmodel,      os.path.join(args.ourmodel_dir,
"ourmodel.joblib"))
```

- problem_unittests.py

```
import numpy as np
import pandas as pd
import re
from unittest.mock import MagicMock, patch
import sklearn.naive_bayes

CSVTEST = 'data/testinfo.csv'


class AssertTest(object):
    def __init__(self, params):
        self.assert_param_message = '\n'.join(
            [str(k) + ': ' + str(v) + '' for k, v in params.items()])

    def test(self, assert_condition, assert_message):
        assert assert_condition, assert_message + \
            '\n\nUnit Test params\n' + self.assert_param_message


def successmsg():
    print('Test cases got passed!')


def tstnumdf(numdf):
    tdf = numdf(CSVTEST)
    assert isinstance(
        tdf, pd.DataFrame), 'Returned type is {}.'.format(type(tdf))
    namecols = list(tdf)
    assert 'Cattype' in namecols, 'No Cattype col got'
    assert 'ClassVal' in namecols, 'No ClassVal col got'
    assert 'FileName' in namecols, 'No FileName col got'
    assert 'Assctask' in namecols, 'No Assctask col got'
    assert tdf.loc[0,
                   'Cattype'] == 1, '`majorplg`failed.'
    assert tdf.loc[2,
                   'Cattype'] == 0, '`notplg` failed.'
    assert tdf.loc[30,
                   'Cattype'] == 3, '`nearplg` failed.'
    assert tdf.loc[5,
                   'Cattype'] == 2, '`lightplg` failed.'
    assert tdf.loc[37, 'Cattype'] == - \
        1, 'orginal file mapping test, failed; should have a Cattype = -
1.'
    assert tdf.loc[41, 'Cattype'] == - \
        1, 'orginal file mapping test, failed; should have a Cattype = -
1.'
    successmsg()


def testctmt(comp_df, cmntfn):
    tval = cmntfn(comp_df, 1, '0Ae.txt')
    assert isinstance(tval, float), 'Returned type is {}.'.format(
        type(tval))
    assert tval <= 1.0, 'Value unnormalized, value should be <=1, got: '
+ \
        str(tval)
    filenames = ['0Aa.txt', '0Ab.txt', '0Ac.txt', '0Ad.txt']
    ng1 = [0.39814814814814814, 1.0,
            0.86936936936936937, 0.59358288770053480]
    ng3 = [0.009345794392523363 8, 0.96410256410256412,
            0.61363636363636365, 0.15675675675675677]
    rsng1= []
    rsng3 = []
    for i in range(4):
        val_1 = cmntfn(comp_df, 1, filenames[i])
        val_3 = cmntfn(comp_df, 3, filenames[i])
        rsng1.append(val_1)
        rsng3.append(val_3)
    assert all(np.isclose(rsng1, ng1, rtol=1e-04)), \
        'n=1 intersection calculation failed'
    assert all(np.isclose(rsng3, ng3, rtol=1e-04)), \
        'n=3 intersection calculation failed'
    successmsg()


def test_lcs(df, lcs_word):
    tstidx = 10
    atxt = df.loc[tstidx, 'TextVal']
    task = df.loc[tstidx, 'Assctask']
    orgfile_rows = df[(df['ClassVal'] == -1)]
    orgfile_row = orgfile_rows[(orgfile_rows['Assctask'] == task)]
    stxt = orgfile_row['TextVal'].values[0]
    tval = lcs_word(atxt, stxt)
    assert isinstance(tval, float), 'Returned type is {}.'.format(
        type(tval))
    assert tval <= 1.0, 'Value should <=1, got: '+str(tval)
    lcs_vals = [0.1917808219178082, 0.8207547169811321,
                               0.8464912280701754,   0.3160621761658031,
0.24257425742574257]
    results = []
    for i in range(5):
        atxt = df.loc[i, 'TextVal']
        task = df.loc[i, 'Assctask']
        orgfile_rows = df[(df['ClassVal'] == -1)]
        orgfile_row = orgfile_rows[(orgfile_rows['Assctask'] == task)]
        stxt = orgfile_row['TextVal'].values[0]
        val = lcs_word(atxt, stxt)
        results.append(val)
    assert all(np.isclose(results, lcs_vals, rtol=1e-05)
               ), 'LCS calculations failed'
    successmsg()


def test_data_split(train_x, train_y, xtst, ytst):
    assert isinstance(train_x, np.ndarray),\
        'train_x not array, instead got type: {}'.format(type(train_x))
    assert isinstance(train_y, np.ndarray),\
        'train_y not array, instead got type: {}'.format(type(train_y))
    assert isinstance(xtst, np.ndarray),\
        'xtst not array, instead got type: {}'.format(type(xtst))
    assert isinstance(ytst, np.ndarray),\
        'ytst not array, instead got type: {}'.format(type(ytst))
    assert len(train_x) + len(xtst) == 95, \
        'Unexpected amount of train + test data. Should be 95 filedata, 
got ' + \
        str(len(train_x) + len(xtst))
    assert len(xtst) > 1, \
        'Unexpected amount of test data. Should be multiple test files.'
    assert train_x.shape[1] == 2, \
        'train_x should have as many columns as selected features, got: 
{}'.format(
            train_x.shape[1])
    assert len(train_y.shape) == 1, \
       'train_y must one dimension, got shape: {}'.format(train_y.shape)
    successmsg()
```