

Data Duplication Removal Using Machine Learning

A PROJECT REPORT

Submitted in partial fulfillment of the

Requirement for the award of the

Degree of

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

&

ELECTRONICS AND COMPUTER ENGINEERING

by

Vatsal Panchal 18BEC1125

S. Harshaavardhini 18BLC1053

Angelina Das 18BLC1076

Under the Guidance of

Dr. Vijayakumar P



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF ELECTRONICS ENGINEERING

VELLORE INSTITUTE OF TECHNOLOGY

CHENNAI - 600127

November 2020

CERTIFICATE

This is to certify that the Project work titled “**Data Duplication Removal Using Machine Learning**” that is being submitted by *S. Harshaavardhini (18BLC1053)*, *Angelina Das (18BLC1076)* and *Vatsal Panchal (18BEC1125)* is in partial fulfilment of the requirements for the award of **Bachelor of Technology in Electronics and Communication Engineering**, is a record of bonafide work done under my guidance. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for award of any degree or diploma and the same is certified.

Dr. VIJAYAKUMAR P

Guide

The thesis is satisfactory / unsatisfactory

Internal Examiner

External Examiner

Approved by

Approved by

Approved by

**Dr. Vetrivelan P
A**

Dr. Thiripurasundari D

Dr. Sivasubramanian

PROGRAM CHAIR

PROGRAM CHAIR

DEAN

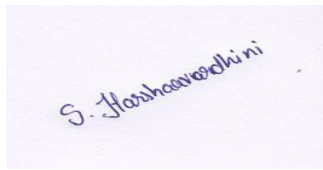
B. Tech. Electronics and
Communication Engineering

B. Tech. Electronics and
Computer Engineering

School of Electronics
Engineering

ACKNOWLEDGEMENTS

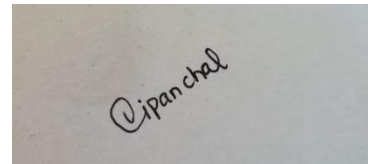
We sincerely thank our Dean, **Dr. Sivasubramanian A**, Program Chair of ECE, **Dr. Vetrivelan P** and Program Chair of ECM, **Dr. Thiripurasundari D** for their support and providing us with the required facilities to complete this project. This acknowledgement of gratitude gives us an opportunity to thank all those who have lent us a helping hand. This project has been a product of motivation and encouragement from various sources. We would like to place on record my deep gratitude towards **Dr. Vijayakumar P** who gave us the opportunity to work under him.



S. Harshaavardhini- 18BLC1053



Angelina Das - 18BLC1076



Vatsal Panchal - 18BEC1125

TABLE OF CONTENTS

CHAPTER NO.		TITLE	PAGE NO.
		LIST OF TABLES	4
		LIST OF FIGURES	4
		ABSTRACT	5
1		INTRODUCTION	6 - 10
	1.1	Objectives	6
	1.2	Background and Literature Survey	7
	1.3	Need for the project	9
	1.4	Organization of Report	10
2		DATA DEDUPLICATION USING MACHINE LEARNING	11 - 13
	2.1	Methodology	11
	2.2	Algorithm	13
	2.3	Challenges Faced	13
3		RESULTS	14 - 19
4		CONCLUSION AND FUTURE WORK	20
	4.1	Conclusion	20
	4.2	Future Work	20
5		REFERENCES AND BIODATA	21 - 22

LIST OF TABLES

SL. NO.	TITLE	PG. NO.
1	Literature Survey	7

LIST OF FIGURES

SL. NO.	TITLE	PG. NO.
1.1	Common deduplication methods	10
2.1	Block diagram of the system	11
3.1	Distribution of Data	14
3.2	Dataset used	15
3.3	Date of birth converted to stanford format	15
3.4	F1 scores from the ML training model after each iteration	16
3.5	F1 score of the ML testing model	16
3.6	Time taken by ML model	16
3.7	The dataset obtained after applying Hash	17
3.8	F1 scores for the Hash training model	17
3.9	F1 scores for the hash testing model	17
3.10	Time taken by hash model	18
3.11	Output csv file of ML model	18
3.12	Output csv file of Hash model	19

ABSTRACT

In computing, data deduplication is a technique for eliminating duplicate copies of repeating data. A related and somewhat synonymous term is single-instance (data) storage. This technique is used to improve storage utilization and can also be applied to network data transfers to reduce the number of bytes that must be sent. In the de-duplication process, unique chunks of data, or byte patterns, are identified and stored during a process of analysis. As the analysis continues, other chunks are compared to the stored copy and whenever a match occurs, the redundant chunk is replaced with a small reference that points to the stored chunk. Given that the same byte pattern may occur dozens, hundreds, or even thousands of times (the match frequency is dependent on the chunk size), the amount of data that must be stored or transferred can be greatly reduced. One of the most common forms of data de-duplication implementations works by comparing chunks of data to detect duplicates. For that to happen, each chunk of data is assigned an identification, calculated by the software, typically using cryptographic hash functions. In many implementations, the assumption is made that if the identification is identical, the data is identical, even though this cannot be true in all cases due to the pigeonhole principle; other implementations do not assume that two blocks of data with the same identifier are identical, but actually verify that data with the same identification is identical. If the software either assumes that a given identification already exists in the de-duplication namespace or actually verifies the identity of the two blocks of data, depending on the implementation, then it will replace that duplicate chunk with a link.

Chapter I

INTRODUCTION

Data Deduplication, often called Dedup for short, is a feature that can help reduce the impact of redundant data on storage costs. When enabled, Data Deduplication optimizes free space on a volume by examining the data on the volume by looking for duplicated portions on the volume. Duplicated portions of the volume's dataset are stored once and are (optionally) compressed for additional savings. Data Deduplication optimizes redundancies without compromising data fidelity or integrity. Data de-duplication is a technique for eliminating duplicate copies of repeating data. Cloud computing is a powerful technology that provides a way of storing voluminous data that can easily be accessed anywhere and at any time. With the continuous and exponential increase of the number of users and the size of their data, data de-duplication becomes more and more a necessity for cloud storage providers. Data de-duplication is a specialized data compression technique that removes redundant data. Examples of few of these methods are Checksums, parity bit, repetition code, Cryptographic hash functions, Error-correcting codes, Checksum and Cyclic Redundancy Checks (CRC). Data Deduplication helps storage administrators reduce costs that are associated with duplicated data. Large datasets often have a lot of duplication, which increases the costs of storing the data. For example:

- User file shares may have many copies of the same or similar files.
- Virtualization guests might be almost identical from VM-to-VM.
- Backup snapshots might have minor differences from day to day.

The space savings that you can gain from Data Deduplication depend on the dataset or workload on the volume. Datasets that have high duplication could see optimization rates of up to 95%, or a 20x reduction in storage utilization.

1.1. OBJECTIVES:

The following are the objectives of this project:

- Detecting and removing duplicates using Machine Learning by calculating the digest of files which takes less time than other pre-implemented methods.
- The project proposes an efficient method for detecting and removing duplicates using machine learning algorithms.
- Storage optimization by de-duplication.

- We read the file with duplicate data and store all the unique entries in it in another new file.
- The input dataset consists of a lot of duplicate entries.

The goal of this Project is to use a machine learning approach to remove those duplicate entries.

1.2. BACKGROUND AND LITERATURE SURVEYS:

SL. NO.	RESEARCH PAPER	SUMMARY
1	Ellappan, Manogar & Murugappan, Abirami. (2014). A study on data deduplication techniques for optimized storage. 161-166. 10.1109/ICoAC.2014.7229702.	In recent years, the explosion of data such as text, image, audio, video, data centers and backup data lead to a lot of problems in both storage and retrieval processes. The enterprises invest a lot of money for storing the data. Hence, an efficient technique is needed for handling the enormous data. There are two existing techniques for eliminating the redundant data in the storage system such as data deduplication and data reduction. Data deduplication is one of the best techniques which eliminates redundant data, reduces the bandwidth and also minimizes the disk usage and cost. Various research papers have been studied from the literature, as the result, this paper attempts to summarize various storage optimization techniques, concepts and categories using data deduplication. In addition to this, chunk based data deduplication techniques are surveyed in detail.
2	Research on Chunking Algorithms of Data De-duplication Proceedings of the 2012 International Conference on Communication, Electronics and Automation Engineering, 2013, Volume 181 ISBN : 978-3-642-31697-5 Cai Bo, Zhang Feng Li, Wang Can	Data de-duplication is a technology of detecting data redundancy, and is often used to reduce the storage space and network bandwidth. Now it is one of the hottest research topics in the backup storage area. In this paper, five representative chunking algorithms of data de-duplication are introduced and their performance on real data set is compared. The experiment result shows that the performance of these methods is improved obviously from the whole-file chunking to the TTTD chunking. According to the analysis of their features, we can provide some references for backup storage systems to choose the best chunking algorithm for eliminating data redundancy.

3	Supriya, S.; Mythili, S. Study on data deduplication in cloud computing. International Journal of Advanced Research in Computer Science . Sep/Oct2017, Vol. 8 Issue 8, p209-213. 5p.	Cloud computing provides scalable, low-cost and location-independent services over the internet. The services provided ranges from simple backup services to cloud storage infrastructures. The fast growth of data volumes has greatly increased the demand for techniques for saving disk space and network bandwidth. Cloud storage services like Dropbox, Mozy, Google Drive choose a deduplication technique where the cloud server stores only a single copy of redundant data and creates links to the copy instead of storing actual copies. The security of users data become a new challenge. Hence the users encrypt the data before outsourcing to the cloud. Conventional encryption techniques are incompatible with deduplication while convergent encryption resolves this problem effectively. Various research papers have been studied from the literature, as a result, this paper attempts to survey data deduplication techniques in cloud storage along with concepts, categories and methods used in data deduplication.
4	Optimizing the Cloud Storage by Data Deduplication: A Study Zuhair S. Al-sagar, Mohammad S. Saleh, Aws Zuhair Sameen ³	In the last few years, the digital data, such images, audio, video and files are exploding. A lot of problems in storage and performance are appearing. Lots of money is spent on these problems. Until the scientists focus on the problems and the needs to solve these problems. Nowadays there are many techniques used for eliminating the redundant data in the storage. One of the best techniques is the deduplication data. In this paper a study on previous researches will do. Focus on the gaps and the problems that they could not solve or introduce them as future work. Finally, we propose a new method depending on the literature to fill the gaps. This method based on the checking the data, whether it is in the cloud storage before storing it
5	Leesakul, W. Townsend, P and Xu, J (2014). Dynamic data deduplication in Cloud storage (2014)	Cloud computing plays a major role in the business domain today as computing resources are delivered as a utility on demand to customers over the Internet. Cloud storage is one of the services provided in cloud computing which has been increasing in popularity. The main advantage of

		<p>using cloud storage from the customers' point of view is that customers can reduce their expenditure in purchasing and maintaining storage infrastructure while only paying for the amount of storage requested, which can be scaled-up and down upon demand. With the growing data size of cloud computing, a reduction in data volumes could help providers reducing the costs of running large storage system and saving energy consumption. So data deduplication techniques have been brought to improve storage efficiency in cloud storages. With the dynamic nature of data in cloud storage, data usage in cloud changes overtime, some data chunks may be read frequently in a period of time, but may not be used in another time period. Some datasets may be frequently accessed or updated by multiple users at the same time, while others may need the high level of redundancy for reliability requirements. Therefore, it is crucial to support this dynamic in cloud storage. However current approaches are mostly focused on static schemes, which limits their full applicability in dynamic characteristics of data in cloud storage. In this paper, they propose a dynamic deduplication scheme for cloud storage, which aiming to improve storage efficiency and maintaining redundancy for fault tolerance.</p>
--	--	--

1.3. NEED FOR PROJECT:

Duplicate data is constantly created across organisations as customer records get inputted, migrated, and manipulated. If an organisation doesn't regularly deduplicate data, they'll potentially waste valuable resources and impact customer relationships. Maintaining relevant, accurate, and reliable data depends largely on deduplication. Even a 1% duplicate rate can have costly impacts across an organisation. A typical email system might contain 100 instances of the same 1 megabyte (MB) file attachment. If the email platform is backed up or archived, all 100 instances are saved, requiring 100 MB of storage space. With data deduplication, only one instance of the attachment is stored; each subsequent instance is referenced back to the one saved copy. In this example, a 100 MB storage demand drops to 1 MB. Thus data deduplication is really important. And since we are using both machine learning and hashes, the results produced by the project is beneficial. Data deduplication is generally done using one of these techniques.

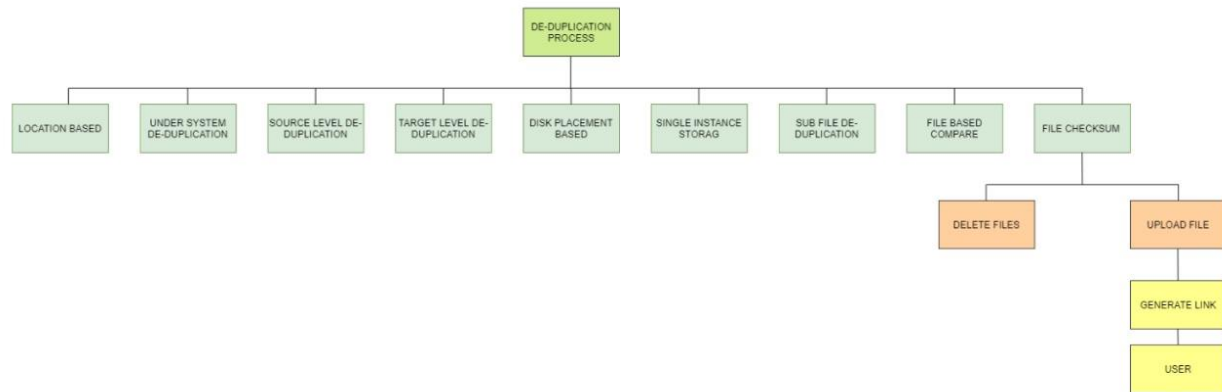


Fig 1.1: Common deduplication methods

1.4. ORGANIZATION OF REPORT:

The remaining chapters of the project report are described as follows:

- Chapter 2 contains the methodology and design approach used in the project.
- Chapter 3 compiles the results obtained after the project was implemented.
- Chapter 4 concludes the report with discussions about the results obtained and their future implications.

CHAPTER II

DATA DEDUPLICATION USING MACHINE LEARNING

2.1 METHODOLOGY

We perform and analyse data deduplication in two methods. Firstly, we implement a pure ML based algorithm which is used to find the duplicate entries. Secondly, we will formulate an algorithm using both checksum and Machine learning approaches. We will find the checksum of each entry in the file, and then use Machine learning to predict if a given entry is unique.

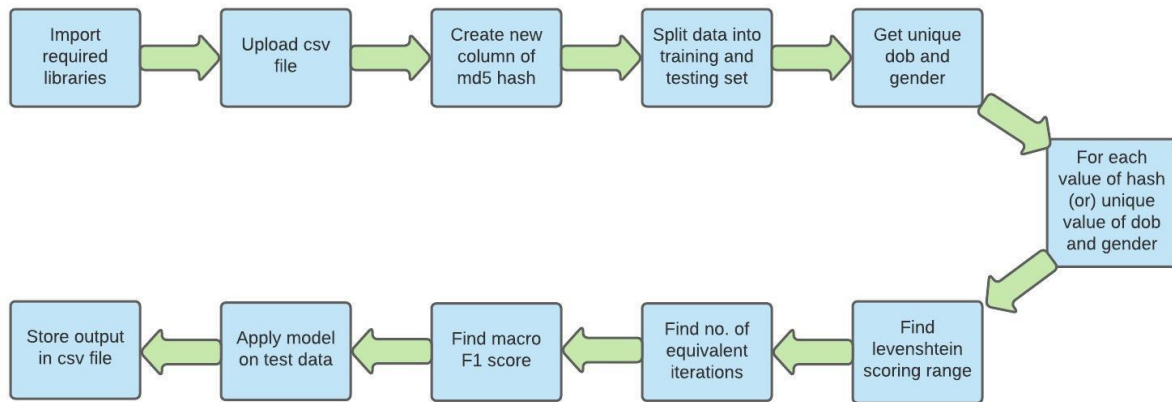


Fig. 2.1: Block diagram of the system

The process of deduplicating is divided into 4 parts.

Phase 1: Data Preprocessing

1. The date of birth column is converted to standard datetime format.
2. Lists of unique dates of birth as well as unique genders are obtained
3. In order to help the model learn through evaluation..
4. The input dataset was pre-processed to classify entries as either duplicate (0) or occurring for the first time (1). Hence, a labeled (binary) column is present in the dataset.
5. The dataset is randomly divided into a training and a test set, keeping the ratio of classes/labels same in both the sets.

Phase 2: Levenshtein Distance

1. It calculates the number of changes required to convert one string into another.
2. The allowed transformations are deletion, substitution and insertion.
3. The Levenshtein distance, when normalized, can be used efficiently to verify a match between listed proper names in a machine learning result.

Phase 3: Training

1. A learning function is created for the purpose of classifying entries as either duplicate (1) or occurring for the first time (0).
2. Parameters of the learning function -
 - a. training data
 - b. Levenshtein_scoring_range
 - c. no. of equivalent iterations before stopping
3. Levenshtein_scoring_range is the number of values of levenshtein score to iterate the model over.
4. No. of equivalent iterations before stopping is the maximum number of increments in the levenshtein score that the model makes despite no improvement in the accuracy.

Phase 4 : Model Building

1. For the first method a loop is run over every unique date of birth in the data.
2. Followed by looping over every unique gender.
3. To identify duplicate entries, we run text matching over entries with the same DOB and Gender.
4. Once the entries with the same DOB and Gender are indexed out as a dataframe.
5. For the second method we run a loop over every hash generated.
6. Every entry in that dataframe is matched with every other entry in the same dataframe to obtain a list of levenshtein scores for each entry.
7. For 2 or more same entries, the first entry will be considered as unique.
8. While the others will be treated as its duplicates.
9. The minimum levenshtein score for an entry will determine if that entry is close to at least one of the other entries with the same DOB and Gender.
10. This minimum value will be selected as the final levenshtein score for that entry.
11. Machine learning is done to identify the optimum maximum value of levenshtein score beyond which an entry will be classified as unique.
12. The algorithm will iterate over a passed range of numbers (levenshtein_scoring_range).
13. Since we not only want the duplicate entries to be removed, we also want the first time entries to remain
14. Hence the appropriate evaluation metric for this problem would be Macro F1-score.
15. It calculates F1-score for each label/class, and finds their unweighted mean.

2.2 ALGORITHM

This is the algorithm followed to deduplicate the data imputed through the csv file:

Step 1. Install and import all the required libraries.

Step 2. Upload the csv file with duplicate entries.

Step 3. Create a new column called hash containing the md5 hash of each row.

Step 4. Store unique values of date of birth, gender.

Step 5. Dataset split into training and test dataset.

Step 6. For the first method, for every unique entry of gender and birthdate.

Step 7. For the second method we take a hash of each row.

Step 8. Levinshtein scoring range and no. of equivalent iterations are found.

Step 9. The loop varied over levenshtein_scoring_range to identify the optimum maximum value of levenshtein score beyond which an entry will be classified as unique using Macro F1 score.

Step 10. Model applied on test dataset to get deduplicated values.

Step 11. Result stored in csv files.

2.3 CHALLENGES FACED

One method for de-duplicating data relies on the use of cryptographic hash functions to identify duplicate segments of data. If two different pieces of information generate the same hash value, this is known as a collision. The computational resource intensity of the process can be a drawback of data deduplication. Another concern is the interaction of compression and encryption. The goal of encryption is to eliminate any discernible patterns in the data. Thus encrypted data cannot be de-duplicated, even though the underlying data may be redundant.

CHAPTER III

RESULTS

The data set we used consisted of five columns: first name (fn), last name (ln), date of birth (dob), gender (gn) and a column for our reference which says if the given entry is duplicated (is_duplicate). It has a total of 150 entries. Originally the data set contained more than 35 % duplicate data, i.e, there are 97 original entries and 53 duplicate entries. In Fig. 3.1, we compare the frequency of duplicate and unique data. We can see that the number of duplicate data is fairly high.

```
data = pd.read_csv('input.csv')  
data['is_duplicate'].value_counts().plot(kind='bar')
```

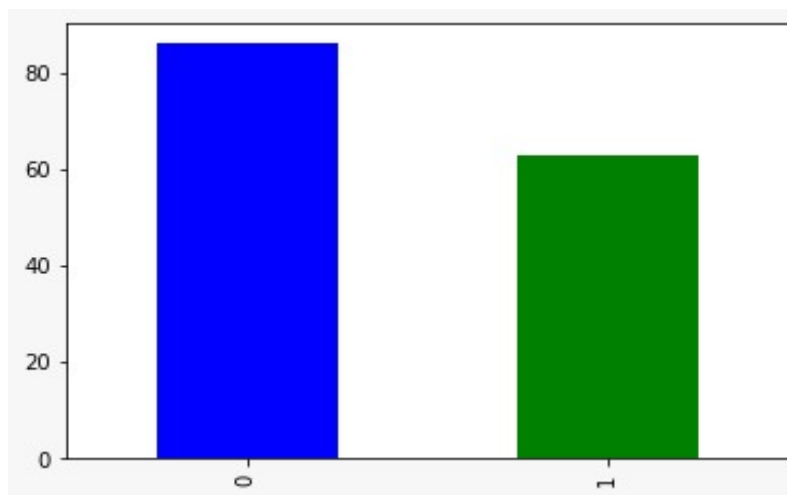


Fig. 3.1: Distribution of Data

The image below (Fig 3.2) gives us a glimpse of the data set being used. The data now contains five columns. The last column is a manually made column which states if a given file is duplicate or not. Here '0' refers to an original entry while '1' refers to a duplicate entry.

```
data.head()
```

	ln	dob	gn	fn	is_duplicate
0	SMITH JR	01-03-1968	F	WILLIAM	0
1	ROTHMEYER JR	01-03-1968	F	WILLIAM	0
2	ASBY JR	01-03-1968	F	WILLIAM	0
3	SALTER JR	01-03-1968	F	WILLIAM	0
4	SALTER JR	01-03-1968	F	WILLIAM	1

Fig. 3.2: Dataset used

The original dataset had the date of birth in the MM/DD/YYYY format. Hence it is converted to its standard format. Fig 3.3 , shows us the conversion of the DOB column to standard date time format. This is an important preprocessing step. It helps bring uniformity to the dataset.

```
In [6]: ##### The dob is converted to standard datetime format.  
data.dob = pd.to_datetime(data.dob)
```

```
In [7]: data.head()
```

```
Out[7]:
```

	ln	dob	gn	fn	is_duplicate
0	SMITH JR	1968-01-03	F	WILLIAM	0
1	ROTHMEYER JR	1968-01-03	F	WILLIAM	0
2	ASBY JR	1968-01-03	F	WILLIAM	0
3	SALTER JR	1968-01-03	F	WILLIAM	0
4	SALTER JR	1968-01-03	F	WILLIAM	1

Fig. 3.3: Date of birth converted to standard format

Macro F1-score (short for macro-averaged F1 score) is used to assess the quality of problems with multiple binary labels or multiple classes. Macro F1-score will give the same importance to each label/class. It will be low for models that only perform well on the common classes while performing poorly on the rare classes. In Fig. 3.4, the Macro F1 scores of the ML based model for the training data are displayed. We see that the F1 score decreases after 3 iterations. We can

see that there is no improvement in the score post 3 iterations. Hence we can conclude that 3 iterations are enough.

```
F1-score after 0 iterations : 0.6611481975967958
F1-score after 1 iterations : 0.6825155783630937
F1-score after 2 iterations : 0.6778147071042793
F1-score after 3 iterations : 0.6663580246913581
F1-score after 4 iterations : 0.6335796178343949
```

Fig. 3.4: F1 score from the MLtraining model after each iteration.

After training the model with training dataset, we test it with the test dataset. Fig. 3.5, contains the Macro F1 score computed for the testing dataset, which is used to determine the effectiveness of our ML model. The F1 score is slightly less when compared to the value we got from the training.

```
print('F1-score on test set:',accuracy_score(test.is_duplicate, predictions))
```

```
F1-score on test set: 0.625
```

Fig. 3.5: F1 scores of the ML testing model.

To test the efficiency of the two methods from other than the output we obtained, we calculated the time taken for the model to run. We can see the turnaround time for our ML model in Fig. 3.6. The ML model took a total of 3.946 seconds to run on a model of 150 entries.

```
Time taken:
```

```
Out[17]: 3.946988105773926
```

Fig 3.6: Time taken by ML model

The images till Fig 3.6 were related to the Machine Learning model. For the second method we combine the ML model with the checksums. First all the values in a row were concatenated to

form a new column called 'concat'. Using concat we find the md5 hash of each row as shown in Fig 3.7.

```
import hashlib
import base64
data = data.assign(concat = data.dob.astype(str) + data.gn + data.fn + data.ln)
data['hash']=data['concat'].astype(str).str.encode('UTF-8').apply(lambda x: base64.b64encode(hashlib.md5(x).digest()))
data
data.head()
```

	ln	dob	gn	fn	is_duplicate	name	concat	hash
0	SMITH JR	1968-01-03	F	WILLIAM	0	WILLIAM SMITH JR	1968-01-03FWILLIAMSMITH JR	b'wKkNy25BkUeZf06o2Jj6zw=='
1	ROTHMEYER JR	1968-01-03	F	WILLIAM	0	WILLIAM ROTHMEYER JR	1968-01-03FWILLIAMROTHMEYER JR	b'N2hmeSiSkC0bi/kbID2YdA=='
2	BLAND III	1962-02-21	F	WILLIAM	1	WILLIAM BLAND III	1962-02-21FWILLIAMBLAND III	b'LTt/hA+IZH7xu0hhuwL5og=='
3	BLAND JR	1962-02-21	F	BILL	0	BILL BLAND JR	1962-02-21FBILLBLAND JR	b'pQf0T22zXX76fYM0LCLMVg=='
4	BLAND	1962-02-21	F	WILLIAM	1	WILLIAM BLAND	1962-02-21FWILLIAMBLAND	b'XhFtQh04LCumDBUz5FGfxA=='

Fig 3.7: The dataset obtained after applying Hash

In Fig. 3.8, the Macro F1 scores of the hash model for the training data are displayed. We see that the F1 score remains constant for all the 4 iterations. Hence we can conclude that 3 iterations are enough

```
F1-score after 0 iterations : 0.6611481975967958
F1-score after 1 iterations : 0.6611481975967958
F1-score after 2 iterations : 0.6611481975967958
F1-score after 3 iterations : 0.6611481975967958
```

Fig 3.8: F1 scores for the Hash training model.

Like in the ML model, after training it with the training dataset, we test it with the test dataset. Fig. 3.9, contains the Macro F1 score computed for the testing dataset. The F1 score for both the models are equal.

```
print('F1-score on test set:',accuracy_score(test.is_duplicate, predictions))
F1-score on test set: 0.625
```

Fig 3.9: F1 scores for the Hash training model

The hash model also took very less time to run. But it is relatively much faster than the ML model by over 2.256 seconds. The figure 3.10 tells us the run time of the second model.

Time taken:
1.6901249885559082

Fig 3.10: Time taken by hash model

Figures 3.11 and 3.12 are the csv files we got as output on running the code. As we can see from fig 3.11, the first csv file, that we obtained from the ML model has around 97 entries while in the hash model (fig 3.12) we got around 103 entries.

A	B	C	D	E	F	G	H	I	J	K
71	RAJ	1964-03-05	M	PATEL	1964-03-05MPATELRAJ	b368R85hWUeYg6nWSquSVw==				
72	MICHAELSON JR	1947-09-04	M	ROBERT	1947-09-04MROBERTMICHAELSON JR	bJlYKIGHxOAEli8G1PgskQ==				
73	DEEPIKA	1955-09-15	F	AJIT	1955-09-15FAJITDEEPIKA	bIeNZ9vZgoPCMSNkCPCMG==				
74	VIDYA	1963-05-15	F	BALAN	1963-05-15FBALANVIDYA	bGPFJUr1CEHt9GmK5GSKg==				
75	CLARK	1945-09-06	M	JOHN	1945-09-06MJOHNCLARK	bStqewbKkrtienRDJBo0/nw==				
76	CLARK	1944-12-16	M	JOHN	1944-12-16MJOHNCLARK	bYs7CBNo6gTPXsgpauxiaQ==				
77	MICHAELSON JR	1947-09-04	M	BOB	1947-09-04MBOBMICHAELSON JR	bEcp44P5sZEn6M3f5xok1Q==				
78	VIDYA	1962-09-10	F	KHAN	1962-09-10FKHANVIDYA	bDUA7m0himdcBms5jeShdkG==				
79	SALTER JR	1968-01-03	F	WILLIAM	1968-01-03FWILLIAMSSALTER JR	bC/nkx7UR9QEB55yKl4sRPg==				
80	MANISH	1937-07-10	M	PATEL	1937-07-10MPATELMANISH	b9JRRki28VAltdQaAVpt7zg==				
81	DUNCAN JR	1953-10-25	F	THOMAS	1953-10-25FTHOMASDUNCAN JR	bSYO19a8mmPhQCInTT5beow==				
82	CAIN JR	1946-01-20	M	JOHN	1946-01-20MJOHNCAIN JR	b2l9JUnkSH4in1eQoyW5hgA==				
83	PRIYANKA	1968-02-09	F	SHARMA	1968-02-09FSHARMAPRIYANKA	bim/phmL1SO3+2FTHgaFXkjg==				
84	BLAND	1962-02-21	F	WILLIAM	1962-02-21FWILLIAMBLAND	bXhFtQh04LCumDBUz5FGfxA==				
85	VIDYA	1962-08-02	F	OBEROI	1962-08-02FOBEROIVIDYA	bKvMzXa+PnltrHDifqtngQ==				
86	ASBY JR	1968-01-03	F	WILLIAM	1968-01-03FWILLIAMASBY JR	bT2cGPhRDatV88ac5FMEQ==				
87	MICHAELSON JR	1951-01-05	M	ROY	1951-01-05MROYMICHAELSON JR	b7rxWacJ4HCGZ3/1o7K1hg==				
88	ANSHUL	1953-10-20	F	SHARMA	1953-10-20FSHARMAANSHUL	bTgmBk9VgA3Ea/uCzrbTN5w==				
89	FAGEN JR	1937-07-10	M	GEORGE	1937-07-10MGEORGEFAGEN JR	bXQipOaYYXG09qE/7lYT3Q==				
90	GRIFFIN JR	1937-07-05	M	CLARENCE	1937-07-05MCLARENCEGRIFFIN JR	b+yRtEFrCk3gQn9jK7E5A==				
91	PRERNA	1945-11-17	F	CHOPRA	1945-11-17FCHOPRA PRERNA	bbsmW2CluyQ7Ba5uEME6zUw==				
92	DEEPIKA	1960-10-12	F	DAS	1960-10-12FDASDEEPIKA	b6pS6s2S5i4VzLgxtu4nA==				
93	GRIFFIN JR	1937-10-07	M	DARL	1937-10-07MDARLGRIFFIN JR	bOxMT5FFSNsJ1W7U7pY1Dg==				
94	RAJ	1945-10-05	M	SHARMA	1945-10-05MSHARMARAJ	b290E3U9EwPdVhWcSaOKw==				
95	SALTER JR	1968-01-03	F	WILLIAM	1968-01-03FWILLIAMSSALTER JR	bC/nkx7UR9QEB55yKl4sRPg==				
96	MELVIN JR	1937-07-10	M	HAROLD	1937-07-10MHAROLDMELVIN JR	bYEBnyjKGoB5D/IT584W95Q==				
97	BLAND III	1962-02-21	F	WILLIAM	1962-02-21FWILLIAMBLAND III	bLTtrhA+IZH7xu0hhuwL5og==				

Fig 3.11: Output csv file of ML model

	A	B	C	D	E	F	G	H	I	J	K	L
79	VIDYA	1963-05-15	F	BALAN	1963-05-15FBALANVIDYA	b'gPFJuRICEHt9G/mK5GSKg==						
80	CLARK	1945-09-06	M	JOHN	1945-09-06MJOHNCLARK	b'SqewbKKrtienRDJBo0/yw==						
81	CLARK	1944-12-16	M	JOHN	1944-12-16MJOHNCLARK	b'Ys7CBNo6gTfPXsgpauXlQ==						
82	MICHAELSON JR	1947-09-04	M	BOB	1947-09-04MBOBMICHAELSON JR	b'Ecp44P5sZEn6M3fv5xok1g==						
83	MANISH	1947-11-03	F	CHOPRA	1947-11-03MCHOPRAMANISH	b'SicsN1NyCHaLX FHpBA0WpQ==						
84	VIDYA	1962-09-10	F	KHAN	1962-09-10FKHANVIDYA	b'DUA7m0hmdcBmsSjeShdKg==						
85	SALTER JR	1968-01-03	F	WILLIAM	1968-01-03FWILLIAMSSALTER JR	b'C/nkx7UR9QE855yK14sRPg==						
86	MANISH	1937-07-10	M	PATEL	1937-07-10MPATELMANISH	b'9JRRkI28VAltDQaVpT7zg==						
87	DUNCAN JR	1953-10-25	F	THOMAS	1953-10-25FTHOMASDUNCAN JR	b'SYO19a8mmPhQCnTT5beow==						
88	CAIN JR	1946-01-20	M	JOHN	1946-01-20MJOHNCAIN JR	b'2l9JUnkSH4in1eQoyWShgA==						
89	PRIYANKA	1968-02-09	F	SHARMA	1968-02-09FSHARMAPRIYANKA	b'm/phmL1SO3+2FTHgaFXkjg==						
90	BLAND	1962-02-21	F	WILLIAM	1962-02-21FWILLIAMBLAND	b'XhFQh04LCumDBUz5FGfxA==						
91	VIDYA	1962-08-02	F	QBEROI	1962-08-02FOBEROIVIDYA	b'kvMzXa+PoltrHDIqtpQ==						
92	ASBY JR	1968-01-03	F	WILLIAM	1968-01-03FWILLIAMASBY JR	b'T2cGPhRdatV88ac5FfMEQ==						
93	MICHAELSON JR	1951-01-05	M	ROY	1951-01-05MROYMICHAELSON JR	b'7rxWaCj4HCGZ3/1o71K1ng==						
94	ANSHUL	1953-10-20	F	SHARMA	1953-10-20FSHARMAANSHUL	b'tqmBk9VgA3Ea/uCzrbTN5w==						
95	FAGEN JR	1937-07-10	M	GEORGE	1937-07-10MGEORGEFAGEN JR	b'QXipQaYYXG08qE/71YtT3Q==						
96	GRIFFIN JR	1937-07-05	M	CLARENCE	1937-07-05MCLARENCEGRIFFIN JR	b'+yRTEFrCk3gOln9jK7E5A==						
97	PREERNA	1945-11-17	F	CHOPRA	1945-11-17FCHOPRAPREERNA	b'tzmW2CluyQ7B55fME6zUy==						
98	DEEPIKA	1960-10-12	F	DAS	1960-10-12FDASDEEPIKA	b'6pS6s2S5i4NzLxGxtu4nA==						
99	GRIFFIN JR	1937-10-07	M	DARL	1937-10-07MDARLGRIFFIN JR	b'OxMT5FFSNsJ1W7UYpLY1Dg==						
100	RAJ	1945-10-05	M	SHARMA	1945-10-05MSHARMARAJ	b'z9OE3U9EewPdvKhCsaOKw==						
101	SALTER JR	1968-01-03	F	WILLIAM	1968-01-03FWILLIAMSSALTER JR	b'C/nkx7UR9QE855yK14sRPg==						
102	MELVIN JR	1937-07-10	M	HAROLD	1937-07-10MHAROLDMELVIN JR	b'YEBnvjKGoB5D/1584W95Q==						
103	BLAND III	1962-02-21	F	WILLIAM	1962-02-21FWILLIAMBLAND III	b'LTUfA+I2H7xu0hhuwL5og==						

Fig 3.12: Output csv file of Hash model

From the outputs obtained, we observed that the ML model took 3.94 seconds to execute.

Though the hash model took less time to execute. The ML model gave a nearly accurate csv file with all the 97 unique entries. The Hash model took 1.69 seconds to execute. It had just one loop in its function. While the hash gave 103 entries in the csv file. That is it gave 4 more duplicate values in the csv file when compared to the first model. This is because the levenshtein distance and macro F1 scores wasn't a good method for the hash model. Hence we can conclude that though the ML model was relatively slow, it gave better results and thus the best model for our dataset.

CHAPTER IV

CONCLUSION AND FUTURE WORKS

4.1 Conclusion

Our project successfully recognised the original values from the csv filled with duplicate values in both ML technique and the ML & hash technique .The deduplicated data frames were then saved in separate csv files.The ML model was the better model when compared to the hash model. Even though it was slow, it gave accurate results. With the emerging need for Automation, AI and various ML based approaches come in handy for realising our needs. Our approach combines the usage of both previously used methods like hashing and the upcoming field of machine learning.This helps us build more effective models with reduced error rates.Making it beneficial for the long run.

4.2 Future works

De-duplication is useful regardless of workload type. Maximum benefit is seen in virtual environments where multiple virtual machines are used for test/dev. and application deployments. When this is implemented with this method, it's efficiency can increase.Some relational databases such as Oracle and SQL do not benefit greatly from de-duplication, because they often have a unique key for each database record, which prevents the de-duplication engine from identifying them as duplicates. Thus if this is integrated with such databases, they will vastly improve. De-duplication benefits both cloud service providers and users.

CHAPTER V

REFERENCES AND BIODATA

REFERENCES

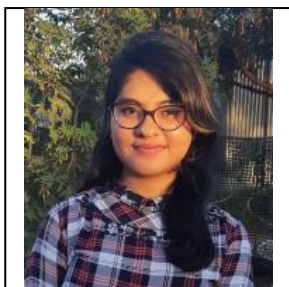
Research Papers:

- [1] Ellappan, Manogar & Murugappan, Abirami. (2014). A study on data deduplication techniques for optimized storage. 161-166. 10.1109/ICoAC.2014.7229702.
- [2] Research on Chunking Algorithms of Data De-duplication. Proceedings of the 2012 International Conference on Communication, Electronics and Automation Engineering, 2013, Volume 181. ISBN : 978-3-642-31697-5. Cai Bo, Zhang Feng Li, Wang Can
- [3] Supriya, S.; Mythili, S. Study on data deduplication in cloud computing. International Journal of Advanced Research in Computer Science . Sep/Oct2017, Vol. 8 Issue 8, p209-213. 5p.
- [4] Optimizing the Cloud Storage by Data Deduplication: A Study. Zuhair S. Al-sagar, Mohammad S. Saleh, Aws Zuhair Sameen3
- [5] Leesakul, W. Townsend,P and Xu, J (2014). Dynamic data deduplication in Cloud storage (2014)

Websites:

- [1] <https://support.esri.com/en/technical-article/000020408>
- [2] <https://itsfoss.com/checksum-tools-guide-linux/>
- [3] <https://mkyong.com/java/how-to-generate-a-file-checksum-value-in-java/>
- [4] <https://ant.apache.org/manual/Tasks/checksum.html>
- [5] <https://searchstorage.techtarget.com/definition/data-deduplication>
- [6] https://en.wikipedia.org/wiki/Data_deduplication

BIODATA



Name : Angelina Das
Mobile Number : 9903115320
E-mail : angelina.das2018@vitstudent.ac.in
Permanent Address : Eastern High, Newtown, Kolkata-700156.



Name : Vatsal Panchal
Mobile Number : 6354203300
E-mail : vatsal.panchal2018@vitstudent.ac.in
Permanent Address : Sector 26 84-A, Reliance Greens, Jamnagar, Gujarat



Name : S. Harshaavardhini
Mobile Number : 9136000640
E-mail : s.harshaavardhini2018@vitstudent.ac.in
Permanent Address : 1A-32, Kalpataru Riverside, Panvel, Navi Mumbai.