

Configuration Manual

MSc Research Project Data Analytics

Tushar Patil Student ID: X19199988

School of Computing National College of Ireland

Supervisor: Bharathi Chakravarthi

National College of Ireland Project Submission Sheet School of Computing



Student Name:	Tushar Patil
Student ID:	X19199988
Programme:	Data Analytics
Year:	2020-2021
Module:	MSc Research Project
Supervisor:	Bharathi Chakravarthi
Submission Due Date:	16/08/2021
Project Title:	Configuration Manual
Word Count:	1247
Page Count:	21

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

<u>ALL</u> internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Tushar Patil
Date:	16th August 2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).Attach a Moodle submission receipt of the online project submission, to
each project (including multiple copies).You must ensure that you retain a HARD COPY of the project, both for

your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only										
Signature:										
Date:										
Penalty Applied (if applicable):										

Contents

1	Intr	oduction	1								
2	System Specification										
3	Tools and Technologies										
4	Environmental Setup										
5	Dat	a Selection and Collection	3								
6	Imp 6.1	lementation Installing and Importing Required Packages	4 4								
	6.2	Exploratory Data Analysis	5								
	6.4	Data Cleaning and Preprocessing	8								
	6.5	Feature Selection	8								
	6.6	Data Split: Train, validation and Test	10								
	6.7	Case Study 1: Training Models on Unbalanced Data	10								
		6.7.1 Training Logistic Regression	11								
		6.7.2 Training Random Forest	11								
		6.7.3 Training XGBoost Model	11								
	6.8	Data Augmentation and Balancing Using CT-GAN	12								
	6.9	Case Study 2: Training Models on balanced Data	14								
		6.9.1 Training Logistic Regression + CT-GAN	14								
		6.9.2 Training Random Forest + CT-GAN	15								
	0.10	$6.9.3 \text{Training XGBoost Model} + \text{CT-GAN} \dots \dots$	15								
	6.10	Testing Performance of All Models on Test Dataset	15								
		6.10.1 Testing Logistic Regression	10								
		6.10.2 Testing Logistic Regression + CI-GAN	10								
		6.10.4 Testing Random Forest + CT CAN	17								
		6.10.5 Tosting XCBoost Model	11 18								
		6.10.6 Testing XGBoost Model $+$ CT CAN	18								
	6.11	Evaluation Using Visualization	19								
7	Con	clusion	21								

Configuration Manual

Tushar Patil X19199988

1 Introduction

The goal of this paper is to provide a quick overview of the processes required in putting this project into action. The main objective of this research was to test the efficiency of the proposed approach of using CT-GAN for the data augmentation method to handle class imbalance issues in credit card fraud prediction tasks. We have implemented the proposed architecture with help of 3 different classifiers and tested the performance of the same for both unbalanced and with balanced data. The subsequent sections of this handbook discuss the tools and strategies that were utilized to achieve the defined goals.

2 System Specification

The system configuration on which research work has been carried out is mentioned below.

- Operating System: Windows 10 Home
- System Type: 64 bit
- Installed Memory (RAM): 16 GB
- Hard Drive: 500 GB SSD
- Processor: Intel® Core[™] i5-9300H CPU @ 2.40GHz
- GPU: GeForce GTX 1650 4GB

3 Tools and Technologies

The Python programming language was utilized to complete this project, with Google Colaboratory serving as the coding platform for developing and processing our code. Google Colaboratory is a free cloud platform provided by Google. It is free for usage and provides free access to GPU/TPU for python coding.

• Python 3.7.11

4 Environmental Setup

As mentioned above, we have used Google Colaboratory for the development of this research, which does not require any environmental set-up on a local machine. The following steps could be followed to configure and run the coding files associated with this research. Step 2 and 3 are alternatives to each other, and either one should be followed. After Initialization of the session, the runtime type should be changes to GPU for faster processing. ¹



Figure 1: Step 1: Google search for 'google colab'

CO Welcome to Colaboratory File Edit View Insert Runtime	Toola Help								co Stare 👾
Table of contents C Gring started Data sensor Machine learning More resources Machine learning examples Sectors	× + Code Colabor - Ze - Frn - Ea	Examples Filter notebooks Title O Overview of Colub	Recent	Google Drive च	GitH Last opened .	ND First opened +	Upload Fr C		Connect + 🖌 Editing A
		CO Welcome to Colab A X19199968_THESI A X19199968_THESI					2 2 4 8 6		
20		x19199968_TH438 ute the code in the above	S.FNAL cell, select it with a cli	ck and then either press	11 August	13 August New notabo	R Cancel	ie keyboard	

Figure 2: Step 2: Open an existing Jupyter Notebook or create a new one.

🍐 Untitled3.ipynb 🕆	
File Edit View Insert Runt	
Open in playground mode	
New notebook	
Open notebook	Ct/+0
Resame	
Mone	
Move to the bio	
Sine	Corl+S
Download	
	Cul+P

Figure 3: Step 3: Upload an existing Jupyter notebook from the local system.

¹https://colab.research.google.com/notebooks/intro.ipynb?utm_source=scs-index# recent=true

5 Data Selection and Collection

The dataset utilized in this research has been fetched from the open-source dataset platform Kaggle which is a public repository. The source URL of the data is given below. Also, the features of the selected dataset are discussed below.²



Figure 4: kaggle Dataset

Data	columns	(total	31 column	<).
#	Column	Non-Nul	11 Count	Dtype
	COLUMN	NOT-NU.		осуре
9	Time	284807	non-null	float64
1	V1	284807	non-null	float64
2	V2	284807	non-null	float64
à	V3	284807	non-null	float64
4	V4	284807	non-null	float64
5	V5	284807	non-null	float64
6	V6	284807	non-null	float64
	V7	284807	non-null	float64
, Я	VR	284807	non-null	float64
q	va	284807	non-null	float64
10	V10	284807	non-null	float64
11	V11	284807	non-null	float64
12	V12	284807	non-null	float64
13	V13	284807	non-null	float64
14	V14	284807	non-null	float64
15	V15	284807	non-null	float64
16	V16	284807	non-null	float64
17	V17	284807	non-null	float64
18	V18	284807	non-null	float64
19	V19	284807	non-null	float64
20	V20	284807	non-null	float64
21	V21	284807	non-null	float64
22	V22	284807	non-null	float64
23	V23	284807	non-null	float64
24	V24	284807	non-null	float64
25	V25	284807	non-null	float64
26	V26	284807	non-null	float64
27	V27	284807	non-null	float64
28	V28	284807	non-null	float64
29	Amount	284807	non-null	float64
30	Class	284807	non-null	int64
dtyp	es: floa	+64(30)	int64(1)	

Figure 5: Data Description

The selected dataset has 31 columns, out of which 28 columns are resulting components of PCA transformation. PCA transformation is carried out for hiding sensitive customer information. Time and amount are only non-transformed features present in data. We have a binary target variable-'Class'.

²https://www.kaggle.com/mlg-ulb/creditcardfraud

6 Implementation

6.1 Installing and Importing Required Packages

In this section of code, we have installed all the required packages for this research. All the packages are installed and imported in the coding environment using 'pip'. Figure:-6,7



Figure 6: Installing required packages



Figure 7: Importing installed packages

6.2 Importing selected Data from Kaggle

For getting our data, we have used kaggle API to remove the dependency of uploading the data into code each time. The code for the same is as given below. Figure:- 8



Figure 8: Importing dataset from kaggle

6.3 Exploratory Data Analysis

It is important to understand the patterns and nature of data before going ahead with any pre-processing step. We have done exploratory data analysis using visualizations in this section. The code for the same is as following. Figure:- 9



Figure 9: Understanding Features in dataset

To understand the distribution of the target variable, we have used a bar graph from **Matplotlib** package. The following code snippet gets the plot of target variable distribution. Figure:- 10, 11



Figure 10: Getting distribution of target feature



Figure 11: distribution of target feature

To understand the correlation between features of data, we are using correlation matrix. We have also plotted correlation matrix for subsample of original data to understand the correlations between feature more accurately. Figure:- 12, 13,14

To understand the distribution and nature of all the features, we have used the **dataprep** package from **Pandas** to get an automatic report for our data. Figure:- 15

CO A X19199988_THESIS_CODE.ipyr	hb τ∕r ⊨Help All-chargestuneed	🗖 Comment 🎿 Share 🗢 🊱
Table of contents ×		🗸 RAM 🔜 🔹 🖌 Editing 🔺
Auther: Turbar Patil INSTALLING AND IMPORTING REQUIRED LIBRARES IMPORTING DATA FROM KAGGLE LEVELONATORY DATA ANALYSIS (EDA) DATA GENMEN AND REPORTED FOR A	 aprecise correction match to dust to identify the correlation between features. phr.fiper(fiper(fiper(sec.5), N)) # After the match to set the valence in the segme radius) to from matk - opticapier(fiper(sec.5), N)(Sec.2017), Apreparation)) between - set.between(fiper(fiper(sec.5))) here - set.between(fiper(fiper(sec.5))), restricted(fiper(fiper(sec.5))), phole(b)). 	
EEATINE OF CETINN	P Correlation Matrix for original data	_
DAT MAN, YA ALATAR, YEN'S TATT CALE ETIN'S HANDOOT BOOKT MANDOOT FURST HENDOOT BOOKT DATA ALATEST TO HEND CALE ETIN'S HANDOOT FURST HANDOOT FURST HONOTON AND TATLE HANDOOT FURST HANDOOT FURST & CEAL		-10 -13 -13 -43 -43
Section	Teine và	-1.00
	1s completed at 2118	

Figure 12: Correlation matrix for original dataset

cc	A X19199988_THESIS_CODI File Edit View Insert Runtime	E.ipynt Tools	o ☆ Help ∉	VI changes	saved															Comm	int 1 1. S	ihare 🏚	۲
=	Table of contents		+ Code																	V RAM Disk	.		^
م د	Auther: Tushar Patil INSTALLING AND IMPORTING REQUIR LIBRARIES	ED	(9)	#getting DF_FOR_O DF_FOR_O	subsampl DRRELATIO DRRELATIO	e of data N = DF_RAN N	for better .sample(fr	understan HC=.1, nan	ding of th dom_state=	e correlat 500, repla	ions in da: ce-False)												
	IMPORTING DATA FROM KAGGLE				Time				¥4		¥6		va		V10				V14		V16		v
	EXPLONATORY DATA ANALYSIS (EDA			40235																			-1 5057
	DATA CLEANING AND PREPROCESSIN	IG		99961	67359.0	.0 580562	1 211205	1.059214	.0 182297	0 135511	.0 492895	0.565321	0 136202	.0 356534	.0.019415	1 119288	0 205313	.0 481179	.0 209570	0 294874	0.681219	.0 338145	0.5175
	FEATURE SELECTION			106203					-0.032005		.0 703687	0.642463	-0.042483	-0 184567		-0.657908							0.0953
	DATA TRAIN, VALIDATION, TEST SPLIT			2675	2215.0	.0 353843	0.403437	1 577453	0 716591	.0.048707	.0 694315	0 258382	0.029425	0.014559	.0 383817	.0 930883	.0 391990	-1 080915	0.078405	0.075558	.0 576965	0.351137	.0.3352
	CASE STUDY:- 1			219193																			-0.8746
	LOGISTIC REGRESSION																						
	KANDOM FOREST			68489																			0.4554
				163210	115747.0	0.058577	.0 163884	0 775043	-1 952921	.0.071904	.0.214163	-0 161480	0.019826	.0.643299	0.032144	-1 371468	-0.913125	0 309835	-0.611309	-0 287019	1 217989	.0 124617	-0.8567
	CASE STUDY- 2			69157																			-1 9474
	LOGISTIC REGRESSION + CT-GAN			13164	23114.0	1 216481	0.481945	0.438229	0.838736	.0.031806	.0.534867	.0.023666	.0 195950	1 180086	.0 723376	1 253078	-1 717418	2 728536	0.999879	0.511618	0 115485	0.954941	.0.5829
	RANDOM FOREST + CT-GAN			29095																			1 7233
	XGBOOST MODEL + CT-GAN			28481 row	s x 31 colu	mns																	
	TESTING PERFORMANCES OF MODEL USING TESTING DATASET.		l	4																			×
	LOGISTIC REGRESSION		[10]																				
	LOGISTIC REGRESSION + CT-GAN			plt.figu	re(figsiz																		
	RANDOM FOREST																						
	RANDOM FOREST + CT-GAN			mask = n heatman	p.triu(np = sos.hea	.ones_like	(DF_FOR_CO B_CORRELAT	RELATION.	corr(), dt	ype+np.boo	1)) . vmax=1. ;	innot=True	cman='Re	86')									
	XGBOOST MODEL			heatmap.	set_title	('Correlat		for subsa	mpled data	\n', fontd	ict-{'font:		pad=16);										
	XGBOOST MODEL + CT-GAN							Corr	elation M	latrix for	subsame	led data											
8	EVALUATION BASED ON ROC CURVE AND VARIOUS METRICS FOR EACH MODEL			Tra .				com			subsuring	neu uut	•				-10						
										1s com	pleted at 21	08											• ×

Figure 13: Getting subsample of data



Figure 14: Correlation matrix for subsample of data

Q	A X19199988_THESIS_CODE.i File Edit View Insert Runtime T	pynb ☆ ools Help ;							🔲 Comment 🛛 🕰 Shi	are 🌣 🍘
=	Table of contents	< + cod							✓ RAM Disk =	Editing
۹	Auther: Tushar Patil	•							1 *	• • • ·
ò	INSTALLING AND IMPORTING REQUIRED		#dataprep create_rep	package ort(DF_RAW)						
	IMPORTING DATA FROM KAGGLE				DataBran Banart	Ownedaw Madebbe -	Interactions Corrobations	Mircine Makes		<u>^</u>
•	EXPLONATORY DATA ANALYSIS (EDA)		_		Баштер Кероп	Overview variables =	intelacions correlations	Missing values		
	DATA CLEANING AND PREPROCESSING									
	FEATURE SELECTION									
	DATA TRAIN, VALIDATION, TEST SPLIT			Overview						
	CASE STUDY:-1									- I
	LOGISTIC REGRESSION				Dataset Statistics			Dataset Insights		
	RANDOM FOREST			Number of Variables	31		v2 and v13 have similar distribution	ations	Similar Distribution	
	XGBOOST MODEL			Number of Rows	28	4807	vs) and v9 have similar distribut	tions	Similar Distribution	
	DATA AUGMETATION USING CT-GAN			Missing Cells	0		vs) and v11 have similar distribution	utions	Similar Distribution	
	CASE STUDY:-2			Missing Cells (%)	0.0	1%	vr and v14 have similar distribution	utions	Similar Distribution	
	LOGISTIC REGRESSION + CT-GAN			Duplicate Rows	10	81	vz and vis have similar distribution	utions	Similar Distribution	
	RANDOM FOREST + CT-GAN			Duplicate Rows (%)	0.4	1%	v7 and v16 have similar distribution	itions	Similar Distribution	
	XGBOOST MODEL + CT-GAN			Total Size in Memory	67	AMB	v7 and v18 have similar distribution	itions	Similar Distribution	
	TESTING PERFORMANCES OF MODELS USING TESTING DATASET			Average Pow Size in Memory	24	10.8	vz and vis have similar distribution	itions	Similar Distribution	
	LOGISTIC REGRESSION			Variable Tuner			v12 and v16 have similar distrib	utions	Similar Distribution	
	LOGISTIC REGRESSION + CT-GAN			Values (ype)	Nu Ca	merical: 30 tegorical: 1	(v13) and (v15) have similar distrib	utions	Similar Distribution	
	RANDOM FOREST							1 2 3 4 5 6 7		
	RANDOM FOREST + CT-GAN									
	XGBOOST MODEL									-
	XGBOOST MODEL + CT-GAN									
=	EVALUATION BASED ON ROC CURVE AND VARIOUS METRICS FOR EACH MODEL			Variables	()a	atad at 73.00				
					 Is compi 	eleu al 21.00				• *

Figure 15: Auto explanatory analysis

6.4 Data Cleaning and Preprocessing

Data cleaning is one of the most important stages in the data analysis process. We are checking for any missing or duplicate values in our data and removing the same from the data. Also, we are scaling our Time and Amount features to keep them at the same scale as other variables. Figure:- 16

cc	A X19199988_THESIS_COD File Edit View Insert Runtime	DE.ipyn Tools	b ☆ Help	All change	s saved														C	3 Commer	nt 🚉 Sh	are 🌣	0
	Table of contents X + Code + Text Recorder + Text																						
	Auther: Tushar Patil INSTALLING AND IMPORTING REQUI LIBRARIES IMPORTING DATA FROM KAGGLE	RED	- DA We	TA CI are cle	LEANIN aning our	G AND data of a	PREPF	ROCES:	SING te values	in this s	ection.										↓ ∞ ⋿	∕⊌∎	
	EXPLONATORY DATA ANALYSIS (EDA DATA CLEANING AND PREPROCESSI FEATURE SELECTION DATA TRAIN, VALIDATION, TEST SPLI CASE STUDY: 1 LOGISTIC REGRESSION RANDOM FOREST	N) ING		<pre>[] storup Dumlinates (rem data. DB[EXALdrop_deplIcates(subsct-store, keep='first', inplace-True) Forders for missing values is data. df percent_missing-colonati-mem(D_RAM_insull(),sam() * 100 / lem(DF_RAM)) df percent_missing-colonati-mem(D_RAM_insull(),sam() * 100 / lem(DF_RAM)) df percent_missing-colonati-mem(D_RAM_insull(),sam() * 100 / lem(DF_RAM)) df percent_missing-trunspace(percentage for each feature in data***********************************</pre>																			
	XGBOOST MODEL						Percentage																
	DATA AUGMETATION USING CT-GAN																	V26 V27 1	V28 Amount				
	CASE STUDY:- 2			Null_Pe	ercentage																		
	LOGISTIC REGRESSION + CT-GAN																						
	RANDOM FOREST + CT-GAN			#scalin	g of Amount	and Time 4																	
	XGBOOST MODEL + CT-GAN			DF_RAN['scaled_amon	int'] = sca	aler.fit_tr	ransform(DF	_RAM[*Amou	int'].value	s.reshape(
	TESTING PERFORMANCES OF MODE USING TESTING DATASET.	LS		DF_RANE	'scaled_time	<pre>2'] = scale</pre>	er.fit_tran	nsform(DF_F	WW['Time']	.values.re	shape(-1,1												
	LOGISTIC REGRESSION					,																	
	LOGISTIC REGRESSION + CT-GAN			DF_RAN																			
	RANDOM FOREST																						
	RANDOM FOREST + CT-GAN									vo		vo		v10	• • • • • • • •	V12	415	*14	V15	010	V1/	v18	
	XGBOOST MODEL																						
	XGBOOST MODEL + CT-GAN				1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	-0.166974	1.612727	1.065235	0.489095	-0.143772	0.635558	0.463917	-0.114805	-0.183361	-0.14
	EVALUATION BASED ON ROC CURVE AND VARIOUS METRICS FOR EACH				-1.358354 -0.966272	-1.340163 -0.185226	1.773209 1.792993	0.379780	-0.503198 -0.010309	1.800499 1.247203	0.791461 0.237609	0.247676	-1.514654 -1.387024	0.207643	0.624501	0.066084	0.717293	-0.165946 -0.287924	2.345865 -0.631418	-2.890083 -1.059647	1.109969 -0.684093	-0.121359 1.965775	
										1s comp	leted at 21:0	8											• ×

Figure 16: Data cleaning and preprocessing

6.5 Feature Selection

To get rid out non-required features from our data, we are using **SelectKBest** feature selection technique from **Scikit-learn** package for getting most relevant features for our further analysis. Below, given code explains the steps involved for getting the top 14 best features from our dataset. Figure:- 17, 18, 19

To verify our selection of features, we are getting correlation between remaining features. Figure:- 20

X19199988_THESIS_CODE	ipynb ☆	🖽 Comment 🔐 Share 🏚 🙆
File Edit View Insert Runtime	Tools Help All changes saved	· · · · · · · · · · · · · · · · · · ·
ble of contents	X + Code + Text	Reconnect - Fediting A
Auther: Tushar Patil NSTALLING AND IMPORTING REQUIR	+ FEATURE SELECTION	
UBRARIES		
MPORTING DATA FROM KAGGLE	In this section we are using KBestUlass model for detection and selection of best predictors from our data. We are	
EXPLONATORY DATA ANALYSIS (EDA)	removing other reatures from data for further analysis as they are requiridum towards our target variable.	
DATA CLEANING AND PREPROCESSIN		
FEATURE SELECTION		
DATA TRAIN, VALIDATION, TEST SPLIT	douring function for fractive selection design besteries	
PASE STUDY-1	X = datafile.drop(response_variable, axis-1)	
	target = datafile[response_variable]	
LOGISTIC REGRESSION	# Univariate Selection using KBestClass feature extraction	
RANDOM FOREST	test = SelectKBest(score_func-f_classif, k-14)	
XGBOOST MODEL	<pre>fit = test.fit(X, target)</pre>	
DATA AUGMETATION USING CT-GAN		
CASE STUDY:- 2	set_printoptions(precision-3)	
LOGISTIC REGRESSION + CT-GAN	features = fit.transform(X)	
RANDOM FOREST + CT-GAN		
XGBOOST MODEL + CT-GAN	X_indices - np.arange(X.shape[-1]) scores = fit.scores	
TESTING PERFORMANCES OF MODEL	scores /= scores.max()	
USING TESTING DATASET.	plt.bar(X_indices, scores, width=5) nlt_title("K_host_ceatures" fination=20)	
LOGISTIC REGRESSION	pit.table('Feature')	
LOGISTIC REGRESSION + CT-GAN	plt.ylabel("Feature importance score")	
RANDOM FOREST		
RANDOM FOREST + CT-GAN	<pre>selected_features = fit.scoresargsort()[-14:][::-1]</pre>	
XGBOOST MODEL	return X.iloc[:, selected_features].copy()	
XGBOOST MODEL + CT-GAN	[] #calling the function for feature selection.	
EVALUATION BASED ON ROC CURVE	<pre>Features_kbest = Feature_selection_model_kBest(DF_RAW, 'Class')</pre>	
AND VARIOUS METRICS FOR EACH	print(Features_Kbest.columns)	
NUULL	 1s. completed at 21.08 	• ×
	X19199988_THESIS_CODE. File Edit Vew Inset Bundine Tele Pail Tele Tele Pail Tele Tele Pail Tele Tele Pail Tele Tel	A XY019998g THESIS COOLEYT + the Addataset and the Addataset

Figure 17: Defining a function for selection of K best Features using SelectKBest method



(a) Scores for all features

(b) List of selected features

Figure 18: feature selection: scores and features

0	#keeping DF_RAW.d DF_CLEAN DF_CLEAN	the above cop(DF_RAW ED=DF_RAW	selected f .columns.di	eatures an fference(b	d our targ est_featur	get variabl Yes), 1, in	e for furt place=True	her analys)	is. Dropin	ng other fe	eatures fro	m datafram	le.			
C⇒		V1	V2	V3	V4	V5	٧7	V9	V10	V11	V12	V14	V16	V17	V18	Class
		-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.239599	0.363787	0.090794	-0.551600	-0.617801	-0.311169	-0.470401	0.207971	0.025791	
		1.191857	0.266151	0.166480	0.448154	0.060018	-0.078803	-0.255425	-0.166974	1.612727	1.065235	-0.143772	0.463917	-0.114805	-0.183361	
		-1.358354	-1.340163	1.773209	0.379780	-0.503198	0.791461	-1.514654	0.207643	0.624501	0.066084	-0.165946	-2.890083	1.109969	-0.121359	
		-0.966272	-0.185226	1.792993	-0.863291	-0.010309	0.237609	-1.387024	-0.054952	-0.226487	0.178228	-0.287924	-1.059647	-0.684093	1.965775	
		-1.158233	0.877737	1.548718	0.403034	-0.407193	0.592941	0.817739	0.753074	-0.822843	0.538196	-1.119670	-0.451449	-0.237033	-0.038195	
	284802	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-4.918215	1.914428	4.356170	-1.593105	2.711941	4.626942	1.107641	1.991691	0.510632	
	284803	-0.732789	-0.055080	2.035030	-0.738589	0.868229	0.024330	0.584800	-0.975926	-0.150189	0.915802	-0.675143	-0.711757	-0.025693	-1.221179	
	284804	1.919565	-0.301254	-3.249640	-0.557828	2.630515	-0.296827	0.432454	-0.484782	0.411614	0.063119	-0.510602	0.140716	0.313502	0.395652	
	284805	-0.240440	0.530483	0.702510	0.689799	-0.377961	-0.686180	0.392087	-0.399126	-1.933849	-0.962886	0.449624	-0.608577	0.509928	1.113981	
	284806	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	1.577006	0.486180	-0.915427	-1.040458	-0.031513	-0.084316	-0.302620	-0.660377	0.167430	
	283726 ro	ws × 15 colu	mns													

Figure 19: Reduced dataframe after removal of redundant features

0	#ge plt mas heat heat	tting co .figure k = np.1 tmap = s tmap.set	orrelatio (figsize= triu(np.o sns.heatm t_title('	ons for =(20, 8) ones_lik map(DF_C 'CORRELA	reduced) e(DF_CLE LEANED.c TION MAT	data aff ANED.com corr(), r RIX FOR	ter feat rr(), dt mask=mas CLEANED	ure sele ype=np.b k, vmin= AND RED	ool)) -1, vmax UCED DAT	=1, anno A', font	ot=True, tdict={'	cmap='B fontsize	rBG') ':18}, p	oad=16);			
C→					CORR	ELATIO	N MAT	RIX FO	R CLEA	NED A	ND RE	DUCED	DATA				
	5 -																- 1.00
	s -	0.0069															-0.75
	5 -	-0.0081	0.0053														0.75
	≸ -	0.0023	-0.0015	0.0028													- 0.50
	₽-	-0.007	0.0052	-0.0069	0.0017												
	5 -	-0.0092	0.0074	-0.012	0.0047	-0.0087											- 0.25
	\$ -	0.0018	-0.00027	-0.0036	0.0022	-0.0012	-0.0049										
	5 - 7	0.00082	0.00062	-0.0096	0.0028	-0.006	-0.014	-0.013									- 0.00
	<u>1</u>	0.001	-0.00063	0.0023	-0.0012	0.00041	0.0025	-0.00022	0.00084								
	77	-0.0015	0.0023	-0.0059	0.0034	-0.0023	-0.0062	-0.0024	-0.0069	0.0056							0.25
	4-	-0.0027	0.0027	-0.003	0.0028	-0.001	-0.0038	0.002	0.00017	0.0077	-0.01						0.50
	9 - 719	-0.0033	0.004	-0.0044	0.0033	-0.0024	-0.0059	-8.6e-05	-0.0037	0.0048	-0.0074	-0.0091					0.50
	5.	-0.0035	0.0032	-0.0082	0.0037	-0.0045	-0.0088	-0.0023	-0.0079	0.0074	-0.013	-0.014	-0.0091				0.75
	8 - 7	-0.0035	0.0025	-0.0035	0.0023	-0.0027	-0.0043	-0.00037	-0.0025	0.0021	-0.0035	-0.0045	-0.0054	-0.0053			
	Class	-0.094	0.085	-0.18	0.13	-0.088	-0.17	-0.094	-0.21	0.15	-0.25	-0.29	-0.19	-0.31	-0.11	a'	1.00
		V1	V2	V3	√4	V5	√7	V9	V10	vi1	V12	V14	V16	V17	V18	Class	
												+	code	+ lext			

Figure 20: Correlation between reduced dataframe

6.6 Data Split: Train, validation and Test

Once our data is ready for application of machine learning models, we are devising our data into train, validation and test subsets for further usage. Figure:- 21

co	X19199988_THESIS_CODE.i File Edit View Insert Runtime T	pynb ☆ xxis Hebp Last_sixed at 22.59	🗖 Comment 🏔 Share 🗢 🌘
	able of contents	K + Code + Text	Connect - 🖌 Editing 🔨
	Auther: Tushar Patil INSTALLING AND IMPORTING REQUIRED LIBRARIES IMPORTING DATA FROM KAGGLE EXPLONATORY DATA ANALYSIS (EDA) DATA CLEANING AND PREPROCESSING	 DATA TRAIN, VALIDATION, TEST SPLIT Before applying our models and other techniques we are dividing our data into train, validation and test set. We would be using train and validation set for tuning and augmentation purpose. Test set would be only used for final testing of model performance. 	î v⊗ L ∕ D I :
	FATURE SELECTION DATA TRAIN, VALONTON, TEST SPLIT CASE STUDY: 1 LOGISTIC REGRESSION FANDOM FOREST XXEBOOST MODEL DATA AUGMETATION USING CFGAN GAE STUDY: 2 LOGISTIC REGRESSION + CFGAN TRANDOM FOREST L CFGAN XXEBOOST MODEL + CFGAN TESTING FRIFORMARETS OF MODELS USING TESTING ONASET.	<pre>[] #function for splitting the data into train, validate and test set. def data_split(feature, target): train, test, target_traind, sarget_test = train_test_split(feature, target, test_size = 0.30, random_state = 735) print(target, solidation) = train_test_split(train, target_train, test_size = 0.20, random_state = 735) print(train head()) print(train head()) print(train head()) print(train head()) print(train head()) print(traget_validation head()) feature_train = train feature_train = train feature_train = train feature_train = validation feature_train = validation feature_train, target_train, feature_validation, target_validation, feature_test, target_test</pre>	
	LOGISTIC REGRESSION LOGISTIC REGRESSION + CT GAN RANDOM FOREST + CT GAN XGB00ST MODEL XGB00ST MODEL + CT GAN EVALUATION BASED ON ROC CURVE AND VARIOUS METRICS FOR EACH MODEL	<pre>[] #culling_down_drivet_splitting=forection fearpretaits = 0F_CLEANED('Class') targetdata = 0F_CLEANED('Class') feature_train_target_train_feature_validation_target_validation_feature_test,target_test = data_split(featuredata,targetdata) 100000 BSI18 100000 BSI18 100000 BSI18 100000 2118509 - 0.00000 0 - 0.00010 - 0.00000 - 0.100105 100000 2118509 - 0.00000 - 0.00010 - 0.100105 200007 112027 - 0.00000 - 0.00000 - 0.100105 200007 112027 - 0.00000 - 0.00000 - 0.00010</pre>	

Figure 21: Splitting data into train, validation and test set

6.7 Case Study 1: Training Models on Unbalanced Data

Modelling is the most crucial part of our research work. For testing our proposed methodology, we are training our selected models on unbalanced data and verifying their performance using a validation dataset.

6.7.1 Training Logistic Regression

In this section, we have trained our Logistic regression model on unbalanced data.

-	LO	GISTIC REGRE	SSION											
		<pre># Logistic Regr LR_before_balan logistic_pred = log_sccuracy=d #model_sccuracy=d #model_evalatio print("consist print(" print(" # auc scores fo auc_score1 = nprint("Auc score</pre>	ession mode cing = Log cing.fit(f LR_before etrics.acc s=[log_acc n for Logi n matrix f 	el applicat isticRegres eature_trai balancing. uracy_score uracy] stic regrss or Logistic mort of Logi Regression e(target_va stic Regres	ion on Tr. sion() n,target_ predict(fr (target_v ion model regression stic regression stic reg	<pre>sin and Val. crain) sature_vali sature_vali on model on ") con model on ") ssSion mode ss</pre>	idation data. dation) logistic_pred))*100 umbalanced data: \r 1 on umbalanced data red(:1) ata: ',auc_score1)	",metric : \n", c	zs.confusion_ ∶lassi≁icatio	matrix(target_ m_report(targe	validation, 1 t_validation,	ogistic_pre logistic_pr	-d)) -ed))	
		Confusion matri [[39655 6] [25 36]] Classification 0 1 accuracy macro avg weighted avg AUC score for Lu	x for Logi Report of f precision 1.00 0.86 0.93 1.00 .00 .00 .00 .00 .00 .00 .00 .00 .0	stic regress ogistic re recall 1.00 0.59 0.80 1.00 sression on	sion mode: gression m fl-score 1.00 0.70 0.85 1.00 	t on unbalad support 39661 61 39722 39722 39722 39722	nced døta: balanced døta: .7950063261551484							

Figure 22: Training of logistic regression model on unbalanced data

6.7.2 Training Random Forest

In this section, we have trained our Random forest model on unbalanced data.

•	RAI	NDOM FORE	ST								
 # Random Forest classification model application on Train and Validation data. R_forest_clf_pefore_balancingsFit(Fature_train_target_train) R_forest_clf_pefore_balancings.Fit(Fature_train_target_train) rfc_pred=R_forest_clf_before_balancing.predict(feature_validation) 											
	rfc_accuracy=(metrics.accuracy_score(target_validation, rfc_pred))*100 Model_accuracies.append(rfc_accuracy)										
		<pre>#model evalat print("Classi print(" print("</pre>	ion for Rando fication Repo	om Forest m ort of Rand	odel. Iom forest		ation model on unbalanced data: \n", classification_report(target_validation,rfc_pred))				
		print("Confus print("****** print("	ion matrix fo	or Random F	orest mod	el on unbal *****") ")	<pre>lanced data: \n",metrics.confusion_matrix(target_validation, rfc_pred))</pre>				
		<pre># auc scores auc_score1 = print("AUC sc</pre>	for Random Fo roc_auc_score ore for Rando	orest (target_va om Forest o	lidation, on unbalan	<pre>rfc_pred[: ced data: '</pre>	:]) ",auc_score1)				
			n Report of F precision	andom fore recall	st classi f1-score	fication mo support	odel on unbalanced data:				
			1.00	1.00	1.00	39661					
			0.96		0.85						
		accuracy			1.00						
		macro avg	0.98	0.89	0.93						
		weighted avg	1.00	1.00	1.00						
		Confusion mat [[39659 [14 47	rix for Rando 2]]]	m Forest m	odel on u	nbalanced d	data:				
			Random Fores	t on unbal	anced data	a: 0.88522	206879533555				

Figure 23: Training of Random forest model on unbalanced data

6.7.3 Training XGBoost Model

In this section, we have trained our XGBoost model on unbalanced data.



Figure 24: Training of XGBoost model on unbalanced data

6.8 Data Augmentation and Balancing Using CT-GAN

Data augmentation using CT-GAN is the most crucial part of our research project. We are using CTGAN package to generate synthetic samples of data and use for to balance our training data set. The code used for implementation of CT-GAN is explained in this section. Figure:- ??,??,??

ſ																		^ ↓
	DA	TA AL	JGMET	ATION	USING	CT-GA	N											
	Ma			I to mode	l the min	ority alay		a far bala		data Wa	would be							
	we	are usir	ig CI-GAN	vio mout		only clas				uala. we	would be	using mil	lory class					
	Sall	iples in		anning uat	a to augi	lient the	new samp	nes.										
	0	#creatin	ng dataframe	e from trai	ning set fo	or augment	ation purpos											
		DF_FOR_A	WG = pd.con	ncat([featu	re_train, t	arget_tra	in], axis=1,	join="inne										
					5.													
		minority minority	_class_df= /_class_df	DF_FOR_AUG	[DF_FOR_AUG	['Class']	==1]											
	C⇒		V1	V2	va	VA	V5	V7	vo	V10	V11	V12	V14	V16	V17	V18	Class	
		118308	-0 430330	0.985633	0 645789	0 317131	0 616332	1.078234	-0 492856	-1 039638	-0 395608	-0 664684	-0 660968	0 530852	0 278142	0 355530		
		149587	1.954852	1.630056	-4.337200	2.378367	2.113348	0.653745	1.217608	-2.829098	3.504568	-3.918200	-4.704509	1.854772	6.024397	3.531250		
		6774	0.447396	2.481954	-5.660814	4.455923	-2.443780	-4.716143	-0.718326	-5.390330	6.454188	-8.485347	-7.019902	-4.649864	-6.288358	-1.339312		
		68522	0.206075	1.387360	-1.045287	4.228686	-1.647549	-2.943678	-1.181743	-3.096504	3.200912	-2.450832	-6.397170	-2.427373	-4.448472	-1.212220		
		14197	-16.598665	10.541751	-19.818982	6.017295	-13.025901	-14.118865	-4.099551	-9.222826	6.329365	-8.952191	-9.825054	-7.541687	-14.259599	-5.035052		
		154697	-4.221221	2.871121	-5.888716	6.890952	-3.404894	-7.739928			5.350890	-9.299807	-6.106552	-6.250629		-4.192780		
		84543	-3.975216	0.581573	-1.880372	4.319241	-3.024330	-1.909559	-2.752611	-3.550385	4.838964	-6.040235	-6.221945	-5.073643	-10.441009	-3.755525		
		154234	-23.984747	16.697832	-22.209875	9.584969	-16.230439	-33.239328	-10.842526	-19.836149	3.223233	-10.895134	0.116303	-7.606425	-18.108261	-7.511866		
		8296	-2.125490	5.973556	-11.034727	9.007147	-1.689451	-7.810441	-5.902828	-12.840934	12.018913	-17.769143	-19.214325	-10.266609	-15.503392	-5.494928		
		10630	-5.187878	6.967709	-13.510931	8.617895	-11.214422	-9.462533	-4.897006	-11.786812	9.369079	-15.094163	-11.852161	-10.688242	-18.388811	-6.898840		
		273 rows	× 15 columns															

Figure 25: Isolation of minority class samples from data for data augmentation

[]	<pre>#inializing the CTGANSynthesizer function of CT-GAN package and provide the required input and tuning parameters for training of CT-GAN. ctgan = CTGANSynthesizer() ctgan.fit(minority_class_df, column_names, epochs=150)</pre>
[]	<pre># generating new samples using our trained CT-GAN model. minority_class_df_new=pd.DataFrame() minority_class_augmented_df=minority_class_df i = 1 while i < 10: augmented_df = ctgan.sample(30000) concat_df=[minority_class_augmented_df,augmented_df] minority_class_augmented_df=pd.concat(concat_df) i += 1</pre>
0	<pre># Merging newly generated samples into original training dataset for balancing the same. concat_df=[DF_RAW,minority_class_augmented_df] balanced_df_GAN=pd.concat(concat_df) balanced_df_GAN</pre>

Figure 26: Using CT-GAN for generation of synthetic minority class samples and merging them to original data



Figure 27: Balanced training data after merging augmented data

6.9 Case Study 2: Training Models on balanced Data

To verify the efficiency of our proposed approach to sole class imbalance using CT-GAN we would be training another set of classifiers on balanced test data and validating their performances on a validation dataset. The code used for training our models on balanced data is as follows. Figure:- 28,29,30,31

Creating feature and target sets for training our models.

•	CASE STUDY:- 2
	APPLYING MACHINE LERANING MODELS:- TRAINED ON BALANCED DATA
	We are training our classifiers on newly balanced dataframe and validating their performance on validation set. Validation set is used to fine tune the performance of models.
	<pre>[] # seperating our predictors and target variable for model application. X_train = balanced_df_GAN.drop(['Class'], inplace=False, axis=1) y_train = balanced_df_GAN['Class']</pre>

Figure 28: Creating feature and target set for training

6.9.1 Training Logistic Regression + CT-GAN

In this section, we have trained our Logistic regression model on balanced data.

LO	GISTIC REGRE	ESSION + C	T-GAN							
0	# Logistic Reg LR_after_balan LR_after_balan logistic_pred	ression mode cing = Logis cing.fit(X_t = LR_after_b	l applicat ticRegress rain,y_tra alancing.p	ion on Tr ion() in) redict(fe	ain(balanced ature_valida	d) and Validation data. ation)				
log_accuracy=(metrics.accuracy_score(target_validation, logistic_pred))*100 Model_accuracies=[log_accuracy]										
	<pre>#model evalati print("Confusi print(" print("t******* print("classif print("t******* print(" # auc scores f auc_score1 = r print("AUC sco</pre>	on for Logis on matrix fo ************ Gor Logistic for Logistic roc_auc_score re for Logis	tic regrss r Logistic ********* rt of Logi ********* Regression (target_va tic Regres	ion model regression stic regression stic regression stic regression stic regression regression stic regression regressi	on model aff ") *****") ession model *****") ") logistic_pu ^ balancing	ter balancing: \n",metr l after balancing: \n", red[:]) : ",auc_score1)	ics.confusion_matri	x(target_validati ort(target_valida	ion, logistic_pred	1)) :d))
C⇒	Confusion matr [[39646 15 [11 50]	ix for Logis]]	tic regres	sion mode	l after bala	ancing:				
	********	******	*******	******						
	Classification	Report of L precision	ogistic re recall	gression n f1-score	nodel after support	balancing:				
	0 1	1.00 0.77	1.00 0.82	1.00 0.79	39661 61					
	accuracy macro avg weighted avg	0.88 1.00	0.91 1.00	1.00 0.90 1.00	39722 39722 39722					
	*****	*******	******	******						
	AUC score for	Logistic Reg	ression af	ter balan	ing: 0.909	96469629288547				

Figure 29: Training of logistic regression model on balanced data

6.9.2 Training Random Forest + CT-GAN

In this section, we have trained our Random Forest model on balanced data.

▼ RAN	NDOM FOREST	+ CT-GA	N						
O	<pre># Random Forest (R_forest_clf_aft) R_forest_clf_aft) rfc_pred=R_fores rfc_accuracy=(mei Model_accuracies</pre>	classifica er_balanci er_balanci t_clf_afte trics.accu .append(rf	ation model ing=RandomFo ing.fit(X_to er_balancing uracy_score fc_accuracy	applicat prestClas rain,y_tr g.predict (target_v)	ion on Train(bala sifier(n_estimato ain) (feature_validati alidation, rfc_pr	nced) and Validat rs=100) on) ed))*100			
	<pre>#model evalation print("Classific print(" print("************************************</pre>	for Rando ation Repo	om Forest mo ort of Rande	odel. om forest	classification m ") *****")		ing: ∖n", classificat	ion_report(targe	t_validation,rfc_pred))
	<pre>print("Confusion</pre>			orest mod		g: \n",metrics.co	onfusion_matrix(target	_validation, rfc	_pred))
	print("*********	********	**********	*******	*****")				
	print(
	auc_score1 = roc	_auc_score	e(target_vai	lidation,	rfc_pred[:])				
	print("AUC score	for Rando	om Forest a	fter bala	ncing: ",auc_scor	e1)			
D•	Classification Re p	eport of R recision	tandom fores recall f	st classi F1-score	fication model af support	ter balancing:			
		1.00	1.00	1.00	39661				
		1.00	1.00	1.00					
	accuracy			1.00					
	macro avg	1.00	1.00	1.00					
	weighted avg	1.00	1.00	1.00					
	Confusion matrix [[39661 0] [0 61]]	for Rando	om Forest mo	odel afte	r balancing:				
	AUC score for Ram	ndom Fores		lancing:					

Figure 30: Training of Random Forest model on balanced data

6.9.3 Training XGBoost Model + CT-GAN

In this section, we have trained our XGBoost Model on balanced data.

✓ XGBOOST MODEL + CT-GAN
 # XGE0ost classification model application on Train(balanced) and Validation data XGE after_balancing - xgb.XGEClassifier() XGE after_balancing.ift(x train, y, train) XGE pred-XGE_after_balancing.predict(feature_validation)
<pre>#model evalation for XGBoost model. #model evalation for XGBoost model. print("Classification Report of XGB classification model after balancing: \n", classification_report(target_validation,XGB_pred)) print("</pre>
pint("Confusion matrix for XGB model after balancing: \n",metrics.confusion_matrix(target_validation, XGB_pred)) pint(""""""""""""""""""""""""""""""""""""
<pre># auc score for XGBoost auc_score1 - roc_auc_score(target_validation, XGB_pred[:]) print('Auc score for XGBoost after balancing: ',auc_score1)</pre>
[→ Classification Report of XGB classification model after balancing: precision recall f1-score support
0 1.00 1.00 1.00 39661 1 0.76 0.84 0.80 61
accuracy 1.00 39722 macro avg 0.88 0.92 0.90 39722 weighted avg 1.00 1.00 1.00 39722
Confusion matrix for XGB model after balancing: [[39645 16] [10 51]]
AUC score for XGBoost after balancing: 0.9178310773973359

Figure 31: Training of XGBoost Model on balanced data

6.10 Testing Performance of All Models on Test Dataset

Once we have obtained both set of models: Trained on unbalanced data, trained on balanced data, we are testing their performance on test set of data. We have collected the performance parameters into a single dataframe for further visualization and summarized results.



Figure 32: Defining function and dataframe to collect the performance parameters for all models

6.10.1 Testing Logistic Regression

In this section, we are testing our Logistic regression model using a test dataset.

₹ L	OGISTIC REGRESSION
•	<pre>print("*******"Logistic Regression performance on testing data after trained on Unbalanced data.*****") print("\n") unbalanced_logistic_pred_test = LR_before_balancing.predict(feature_test)</pre>
	<pre>#model evalation for Logistic regression model trained on unbalanced data tested on testing data. print("Confusion matrix for Logistic regression model trained on unbalanced data: \n',metrics.confusion_matrix(target_test, unbalanced_logistic_pred_test)) print("") </pre>
	print("Classification Report of Logistic regression model trained on unbalanced data: \n", classification_report(target_test,unbalanced_logistic_pred_test)) print(" ")
	<pre>#RECORDING EVALUATION METRICS FOR LR get_model_performance_parameters(target_test,unbalanced_logistic_pred_test,'LOGISTIC_REGRESSION','LR')</pre>
	🕞 *******Logistic Regression performance on testing data after trained on Unbalanced data.*****
	Confusion matrix for logistic regression model trained on unbalanced data: [[48967 12] [60 79]]
	Classification Report of Logistic regression model trained on unbalanced data: precision recall fi-score support
	0 1.00 1.00 1.00 84979 1 0.87 0.57 0.69 139
	accuracy 1.00 85118
	ասւլյանը 0.55 0.76 0.04 05116 weighted avg 1.00 1.00 1.00 85118
	LOGISTIC_REGRESSION parameters recorded

Figure 33: Testing logistic regression model on test data.

6.10.2 Testing Logistic Regression + CT-GAN

In this section, we are testing our Logistic regression + CT-GAN model using a test dataset.



Figure 34: Testing logistic regression + CT-GAN model on test data.

6.10.3 Testing Random Forest

In this section, we are testing our Random Forest model using a test dataset.

•	✓ RANDOM FOREST									
		<pre>print("****** print("\n") unbalanced_rf #model evalat print("confus print(" print("'classi print(" print("***** #RECORDING EV get_model_per</pre>	**Random Fore ic_pred=R_fore ion for Randc ion matrix fc fication Repo fication Repo ALUATION METR formance_para	est model p est_clf_bef om Forest m or Random F ort of Rand ert of Rand reters (tar	erformance ore_balanc odel. orest mode 	<pre>on testing data after trained on Unbalanced data.*******) ing.predict(feature_test) l trained on Unbalanced data: \n", metrics.confusion_matrix(target_test, unbalanced_rfc_pred)) *****') ") classification model trained on Unbalanced data: \n", classification_report(target_test,unbalanced_rfc_pred)) *****' nbalanced_rfc_pred, 'RANDOM_FOREST', 'RF')</pre>				
		*******Randod Confusion mat [[84977 [36 103 ******** Classificatio 0 1	m Forest mode rix for Rando 2]]] n Report of R precision 1.00 0.98	l performa m Forest m andom fore recall 1.00 0.74	nce on tes odel train ******* st classif f1-score 1.00 0.84	ting data after trained on Unbalanced data.****** ed on Unbalanced data: iCation model trained on Unbalanced data: support 139 139				
		accuracy macro avg weighted avg **************	0.99 1.00 ****************	0.87 1.00	1.00 0.92 1.00	85118 85118 85118				

Figure 35: Testing Random Forest model on test data.

6.10.4 Testing Random Forest + CT-GAN

In this section, we are testing our Random Forest + CT-GAN model using a test dataset.

•	RA	NDOM FOREST + CT-GAN									
		<pre>] print(********Random Forest model performance on testing data after trained on balanced data.*******) print(*\n') balanced_ffc_pred=R_forest_clf_after_balancing.predict(feature_test) #model evalation for Random Forest model.</pre>									
		print("Confusio print("******** print("	n matrix fo **********	r Random F	orest mode *********	<pre>el trained on balanced data: \n',metrics.confusion_matrix(target_test, balanced_rfc_pred)) ")</pre>					
		print("Classifi print(" print("********				<pre>classification model trained on balanced data: \n", classification_report(target_test,balanced_rfc ") ******)</pre>	pred))				
		<pre>#RECORDING EVALUATION METRICS FOR RF+CTGAN get_model_performance_parameters(target_test,balanced_rfc_pred,'RANDOM_FOREST + CT-GAN', 'RF+CT-GAN')</pre>									
		*******Random Forest model performance on testing data after trained on balanced data.*****									
		Confusion matrix for Random Forest model trained on balanced data: [[84979 0] [0 139]]									
			Report of R	andom fore	st classif f1-score	fication model trained on balanced data:					
			1.00 1.00	1.00 1.00	1.00 1.00	84979 139					
		accuracy macro avg weighted avg	1.00 1.00	1.00 1.00	1.00 1.00 1.00	85118 85118 85118					
RANDOM_FOREST + CT-GAN parameters recorded											

Figure 36: Testing Random Forest + CT-GAN model on test data.

6.10.5 Testing XGBoost Model

In this section, we are testing our XGBoost model using a test dataset.

✓ XGBOOST MODEL										
₽	<pre>print("********XGBoost model performance on testing data after trained on unbalanced data.**********) print("\n")</pre>									
	UNBALANCED_XGB_pred=XGB_before_balancing.predict(feature_test)									
	<pre>model evalation for X6Boost model trained on unbalanced data, tested on testing data print("confusion matrix for X6Boost model trained on unbalanced data: \n",metrics.confusion_matrix(target_test, UNBALANCED_X6B_pred)) print("") print("")</pre>									
<pre>print("Classification Report of XGBoost classification model trained on unbalanced data: \n", classification_report(target_test,UNBALANCED_ print("") print("""") #RECORDING EVALUATION NETRICS FOR XGBOOST get_model_performance_parameters(target_test,UNBALANCED_XGB_pred, 'XGBOOST', 'XGB')</pre>										
	Confusion matrix for XGBoost model trained on unbalanced data: [[84972 7] [39 100]]									
	Classification Report of XGBoost classification model trained on unbalanced data: precision recall fi-score support									
	0 1.00 1.00 1.00 84979 1 0.93 0.72 0.81 139									
	accuracy 1.00 85118 macro avg 0.97 0.86 0.91 85118									
	weighted avg 1.00 1.00 85118									
	XGBOOST parameters recorded									

Figure 37: Testing XGBoost model on test data.

6.10.6 Testing XGBoost Model + CT-GAN

In this section, we are testing our XGBoost + CT-GAN model using a test dataset.

▼ XGBOOST MODEL + CT-GAN										
	<pre>print("********XGBoost model performance on testing data after trained on balanced data.*****************) print("\n")</pre>									
	BALANCED_XGB_pred=XGB_after_balancing.predict(feature_test)									
	<pre>#model evalation for XGBoost model trained on unbalanced data, tested on testing data print("Confusion matrix for XGBoost model trained on balanced data: \n",metrics.confusion_matrix(target_test, BALANCED_XGB_pred)) print("""")</pre>									
	<pre>print("Classification Report of XGBoost classification model trained on balanced data: \n", classification_report(target_test,BALANCED_XGB_pred)) print("</pre>									
	<pre>#RECORDING EVALUATION METRICS FOR XGB00ST+CT-GAN get_model_performance_parameters(target_test,BALANCED_XGB_pred,'XGB00ST + CT-GAN','XGB+CT-GAN')</pre>									
D	*********XGBoost model performance on testing data after trained on balanced data.*****************************									
Confusion matrix for XGBoost model trained on balanced data: [[84956 23] [22 117]]										
	Classification Report of XGBoost classification model trained on balanced data: precision recall f1-score support									
	0 1.00 1.00 1.00 84979 1 0.84 0.84 139									
	accuracy 1.00 85118 macro avg 0.92 0.92 85118 weighted avg 1.00 1.00 85118									
	XGBOOST + CT-GAN parameters recorded									

Figure 38: Testing XGBoost + CT-GAN model on test data.

6.11 Evaluation Using Visualization

In this section, we have summarized the various evaluation metrics calculated for each mode after testing on the test dataset. Figure:- 39 shows the summarised metrics in a single dataframe.

- EVALUATION BASED ON ROC CURVE AND VARIOUS METRICS FOR EACH MODEL												
		MODEL.	_PERFOROMANCE_DF									
			MODEL_CASE	MODEL_ABB	PRECISION	RECALL	F1_SCORE	AUC_SCORE	GM	FP_RATE	TP_RATE	
			LOGISTIC_REGRESSION	LR	86.813187	56.834532	68.695652	78.410206	70.242344	[0.0, 0.00014121135810023653, 1.0]	[0.0, 0.5683453237410072, 1.0]	
			DGISTIC_REGRESSION + CT-GAN	LR+CT-GAN	80.714286	81.294964	81.003584	90.631596	81.004105	[0.0, 0.0003177255557255322, 1.0]	[0.0, 0.8129496402877698, 1.0]	
			RANDOM_FOREST		98.095238	74.100719	84.426230	87.049183	85.258007	[0.0, 2.353522635003942e-05, 1.0]	[0.0, 0.7410071942446043, 1.0]	
			RANDOM_FOREST + CT-GAN	RF+CT-GAN	100.000000	100.000000	100.000000	100.000000	100.000000	[0.0, 0.0, 1.0]	[0.0, 1.0, 1.0]	
			XGBOOST		93.457944	71.942446	81.300813	85.967104	81.997519	[0.0, 8.237329222513797e-05, 1.0]	[0.0, 0.7194244604316546, 1.0]	
			XGBOOST + CT-GAN	XGB+CT-GAN	83.571429	84.172662	83.870968	92.072798	83.871506	[0.0, 0.00027065510302545337, 1.0]	[0.0, 0.841726618705036, 1.0]	

Figure 39: Summarized performance parameters

By using all the performance parameter collected into single dataframe we have used bar graph to carry out comparative analysis. The code for bar plot is as Figure:- 40



Figure 40: Code for plotting bar graph using values for all models.

By using all the TP and FP parameters collected into data frame we have plotted a AUC-ROC curve for further comparative analysis. The code for AUC-ROC is as Figure:-41, 42



Figure 41: AUC-ROC curve graph for all models.



Figure 42: AUC-ROC curve

7 Conclusion

The whole implementation procedure of this project has been outlined in a succinct, thorough, and sequential way using the information presented in the preceding parts. The needed packages have been indicated wherever they were used. All the codes are commented and divided into sections for better readability.