INSTITUTE OF BUSINESS ADMINISTRATION
KARACHI

PROJECT REPORT

(CLASS IMBALANCE PROBLEM)

[FINAL PROJECT]

GROUP MEMBERS

S.NO	Name	ERP
1	AMBER SHEIKH	27508
2	MAHAM WASEEM	27257

ABSTARCT

The objective of this project is to address class imbalance techniques' impact on machine learning performance using various classification methods. The project employs a systematic approach to constructing an automated pipeline that encompasses data importation, exploratory data analysis (EDA), feature selection, and ultimately the implementation of classification techniques across different machine learning algorithms. This pipeline was replicated across three distinct datasets to bolster understanding of class imbalance techniques in diverse scenarios.

TABLE OF CONTENTS

1 INTRODUCTION

In the domain of machine learning, achieving accurate model predictions is paramount for successful deployment. However, this pursuit comes with various challenges, including feature selection, overfitting, selecting appropriate models within datasets, and particularly addressing class imbalances in datasets. In this project, we confront nearly all of these challenges and apply diverse strategies to manage them effectively.

Class imbalance arises when one class significantly outweighs the other. This imbalance can lead to inaccurate model performance, as algorithms may disproportionately favor the majority class over the minority class. Therefore, it is essential to address this challenge to ensure accurate and reliable performance. Doing so helps mitigate the bias introduced by the dominance of the majority class.

2 DATASETS SNAPSHOT

DATASET NAME	NO OF ROWS	NO OF COLUMNS	CLASS IMBALANCE RATION	TARGET COLUMN	EDA PERFORMED
Churn Dataset	7043	20	27 Is To 73	Churn	Mean Null Value Replacement
Bank Marketing Dataset	41175	21	11 Is To 89	Y	Duplicates Remove
Credit Fraud Dataset	284807	30	0.4 Is To 99.6	Class'	Simple Eda

3 DOMAIN ANALYSIS

3.1 BUSINESS PROCESS RELATED TO DATASET

3.1.1 Telco Churn Data Set:

The telco company collects and preprocesses customer data, which includes demographics and service details. They utilize predictive modeling tools to forecast the likelihood of churn based on factors such as tenure, contract type, and monthly charges. High-risk customers are identified for targeted retention efforts, which may involve offering discounts or personalized recommendations. The company continuously monitors churn metrics, analyzes customer feedback, and iteratively improves its retention strategies to maximize customer lifetime value and drive sustainable growth.

3.1.2 Bank Marketing Dataset:

This dataset provides information about the marketing campaign of a financial institution. It enables the analysis of future strategies to enhance upcoming marketing campaigns for the bank.

3.1.3 Credit Fraud Dataset:

This dataset pertains to credit cards used by European cardholders, containing transactions that occurred over two days, during which 492 frauds out of 284,807 transactions were detected. The primary objective of this dataset is to identify fraudulent activities.

4 Back ground & Literature Review

Understanding class imbalance and exploring various techniques is crucial for developing accurate machine learning models. Now, let's delve deeper into a comprehensive exploration of the challenges posed by class imbalance and the diverse range of methodologies available to address this issue.

4.1 Challenges of Class Imbalance Problem:

4.1.1 Biased Predictions:

Models trained on imbalanced datasets tend to favor the majority class, resulting in accurate predictions for the majority but often inaccurate predictions for the minority class.

4.1.2 Poor Generalization:

Imbalanced datasets often lead to poorly generalized results for new data, particularly for the minority class, due to insufficient representation.

4.1.3 Misclassification of Minority Instances:

Sometimes, the algorithm misidentifies the minority class as outliers or noise, leading to misclassification and overlooking important patterns within the dataset.

4.1.4 Data Sparsity

Data sparsity can lead to misclassification and hinder model learning due to improper patterns in the data.

4.1.5 Model Evaluation Biases:

Performance metrics such as accuracy can be misleading as they may primarily reflect results based on the majority class while neglecting the minority class.

4.2 Techniques to Address Class Imbalance:

4.2.1 Resampling Methods:

4.2.1.1 *Random Under Sampling:*

This technique randomly removes instances from the majority class to balance the dataset.

4.2.1.2 *Random Over Sampling:*

Random oversampling duplicates instances of the minority class to balance the dataset's pattern.

4.2.2 SMOTE

Synthetic Minority Over-sampling Technique (SMOTE) generates synthetic instances of the minority class by interpolating existing minority class instances.

4.2.3 ADASYN

Adaptive Synthetic Sampling (ADASYN) dynamically adjusts the density of classes, focusing on regions with high imbalance.

4.2.4 Algorithmic Methods

4.2.4.1 *Class Weighting:*

In this method, the algorithm assigns higher weights to the minority class to mitigate class discrepancy towards the majority class.

4.2.4.2 *Ensemble Methods:*

Ensemble methods such as bagging and boosting are applied in combination with other classifiers, trained on different subsets of the imbalanced dataset to improve classification performance.

4.2.4.3 *One-Class Learning:*

In this approach, the model is trained to recognize instances of the minority class when treated as noise or anomalies, distinguishing them from the majority class.

4.2.5 Data-Level Techniques:

4.2.5.1 *Feature Engineering:*

Creating or transforming features to better discriminate between classes or mitigate the effects of class imbalance.

4.2.5.2 *GAN:*

Generative Adversarial Networks (GANs) learn synthetic minority class samples by understanding the dataset distribution and creating realistic minority class instances.

5 METHODOLOGY:

In this project, an automated pipeline was created and then applied to three different datasets. The pipeline consists of several functions, which are ultimately called in the master function. Here are the functions included in the pipeline:

5.1 FUNCTION FOR DATA COLLECTION

This function imports the Excel dataset file into Python for further experimentation.

5.2 FUNCTION FOR OBTAINING MAJOR DETAILS OF DATA:

This function provides an overview of the dataset, displaying various statistics and properties. It includes information such as null values, unique values, and the percentage of unique and null values relative to the total number of values in each column. Additionally, it shows the minimum and maximum values found in each column. Overall, this dataset computation assists in exploratory data analysis (EDA). Below is an example:

	type	count	nunique	%unique	null	%null	min	max
gender	object	7021	2	0.028486	0	0.0	Female	Male
SeniorCitizen	int64	7021	2	0.028486	0	0.0	0	1
Partner	object	7021	2	0.028486	0	0.0	No	Yes
Dependents	object	7021	2	0.028486	0	0.0	No	Yes
tenure	int64	7021	73	1.039738	0	0.0	0	72
PhoneService	object	7021	2	0.028486	0	0.0	No	Yes
MultipleLines	object	7021	3	0.042729	0	0.0	No	Yes
InternetService	object	7021	3	0.042729	0	0.0	DSL	No
OnlineSecurity	object	7021	3	0.042729	0	0.0	No	Yes
OnlineBackup	object	7021	3	0.042729	0	0.0	No	Yes
DeviceProtection	object	7021	3	0.042729	0	0.0	No	Yes
TechSupport	object	7021	3	0.042729	0	0.0	No	Yes
StreamingTV	object	7021	3	0.042729	0	0.0	No	Yes
StreamingMovies	object	7021	3	0.042729	0	0.0	No	Yes
Contract	object	7021	3	0.042729	0	0.0	Month-to-month	Two year
PaperlessBilling	object	7021	2	0.028486	0	0.0	No	Yes
PaymentMethod	object	7021	4	0.056972	0	0.0	Bank transfer (automatic)	Mailed check
MonthlyCharges	float64	7021	1585	22.575132	0	0.0	18.25	118.75
TotalCharges	float64	7021	6531	93.020937	0	0.0	18.8	8684.8
Churn	object	7021	2	0.028486	0	0.0	No	Yes

5.3 FUNCTION FOR INVESTIGATING TARGET COLUMN

This function separates the categorical column used in the training and testing of the model.

5.4 FUNCTION FOR INVESTIGATING CATEGORICAL COLUMNS

This function separates categorical and continuous variables.

5.5 FUNCTION FOR LABEL ENCODING AND ONE-HOT ENCODING

This function applies encoding techniques for use in machine learning models. Label encoding assigns unique labels to each category, while one-hot encoding creates binary columns for each category, indicating its presence or absence.

5.6 FUNCTION FOR HANDLING CLASS IMBALANCE

This function addresses class imbalance within the dataset.

- **Apply_smote function**: Creates synthetic samples for the minority class.
- **Apply_adasyn function**: Employs ADASyn to synthetically generate the minority class.
- **Apply_undersampling function:** Involves undersampling to mitigate class imbalance by randomly removing samples.
- **GAN Function:** Specifically targets the minority class by generating synthetic samples using Generative Adversarial Networks (GANs).

5.7 Function for Data Cleaning:

Different data cleaning functions were created based on the situation, including replacing null values with mean and using KNN imputation.

5.8 FUNCTION FOR EDA (Exploratory Data Analysis):

This function conducts EDA on both numerical and categorical features. It generates summary statistics, histograms, and box plots to visualize the distribution of each feature, identifies potential outliers, skewness, or patterns, and displays correlation heatmaps.

5.9 FUNCTION FOR Transformation

Two types of z-score normalization functions were created to scale features within comparable ranges and prevent certain features from dominating others in scale.

5.10 FUNCTION FOR Feature Selection:

This function facilitates feature selection and analysis by removing irrelevant features for predictive modeling.

- **Chisquare 1 function**: Conducts tests for broader analysis based on combinations of categorical features.
- **FStatistics functions:** Calculates association between numerical features relative to a target variable.
- **Mutual_information_regression function**: Computes mutual information between each feature and the target variable.
- **Apply_pca function:** Performs feature extraction and dimensionality reduction using Principal Component Analysis (PCA).

5.11 FUNCTION FOR Storing Results in a DataFrame:

This function imports overall results in the form of a CSV file for graph creation.

5.12 FUNCTION FOR ML Algorithm:

This function streamlines machine learning algorithms and evaluates their performance on given datasets.

- **apply_ML_algov2 function:** Applies specified ML model to training data and evaluates its performance on test data, calculating various evaluation metrics.
- **apply_ML_algov3 function**: Tailored for Naïve Bayes classifier, calculates class weights based on distribution of classes in training data, applies Naïve Bayes model, and evaluates performance on test data.
- **apply_ML_algo_withcv function**: Performs cross-validation for a given machine learning model, printing CV scores along with mean accuracy.

5.13 Special Data Handling on Different Datasets

5.13.1 Churn Dataset

Features are highly correlated, leading to increased number of features after encoding techniques. Patterns to detect class imbalance and effects of class imbalance techniques were monitored on different algorithms.

5.13.2 Bank Marketing Dataset

No null values present, EDA conducted normally. Served as an average dataset for handling class imbalance. Feature selection performed by dropping multicollinear features.

5.13.3 Credit Fraud Dataset

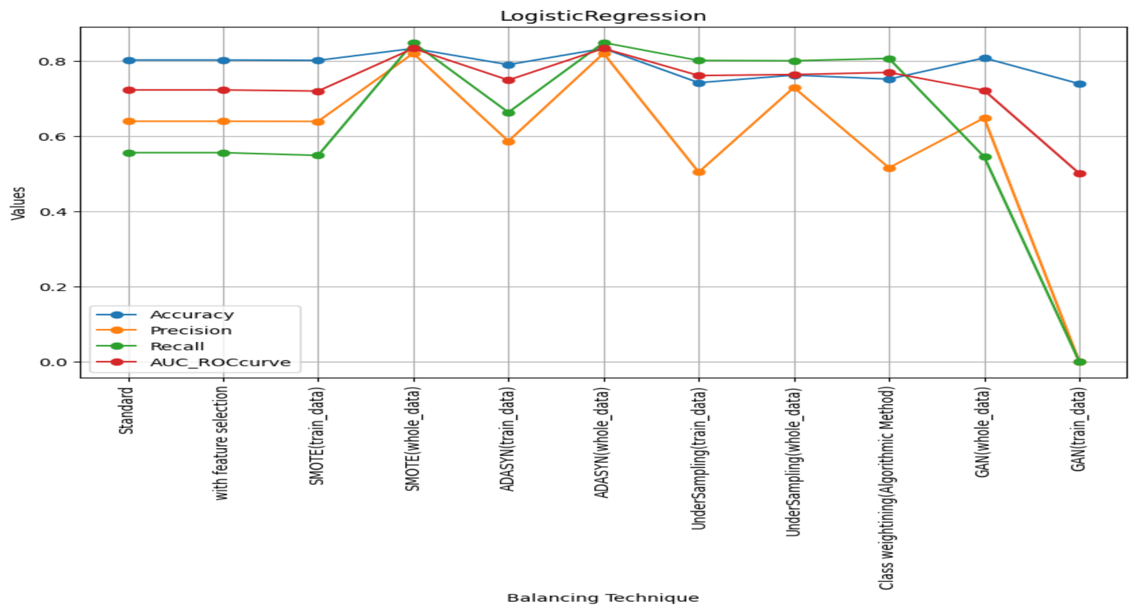
Large dataset with effects of class imbalance. Data handling was a challenging task.

5.14 Pipeline Execution

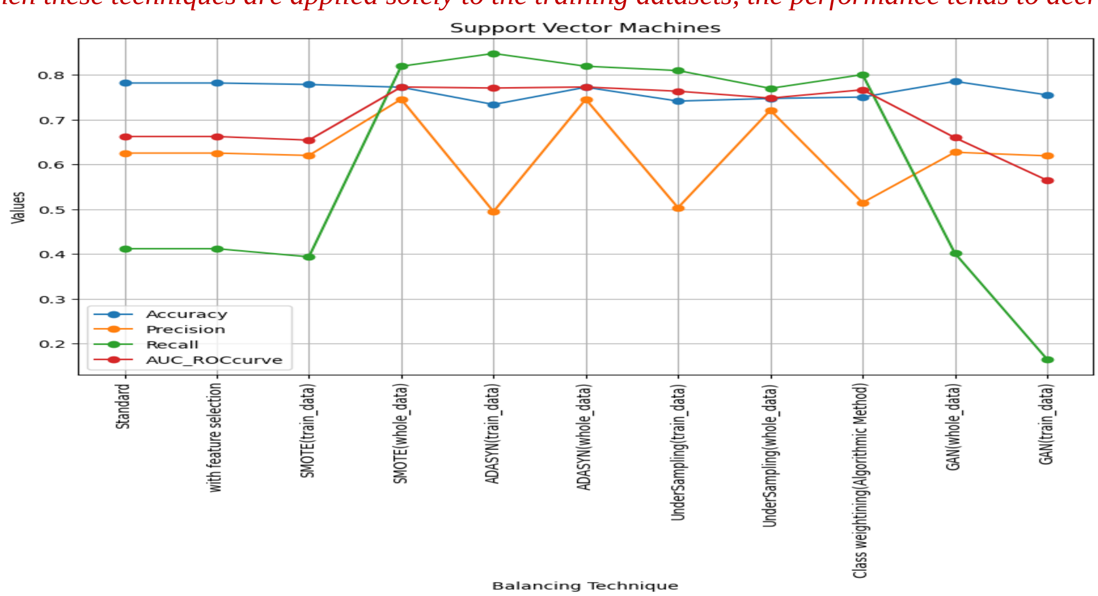
ML execution followed a sequence: baseline model, logistic regression, SVC, decision tree classifier, random forest classifier, and Naïve Bayes. Feature selection was performed, followed by SMOTE, ADASYN, undersampling, class weighting, and GAN techniques, with results checked for each step.

6 RESULTS & ANALYSIS

6.1 CHURN DATASET

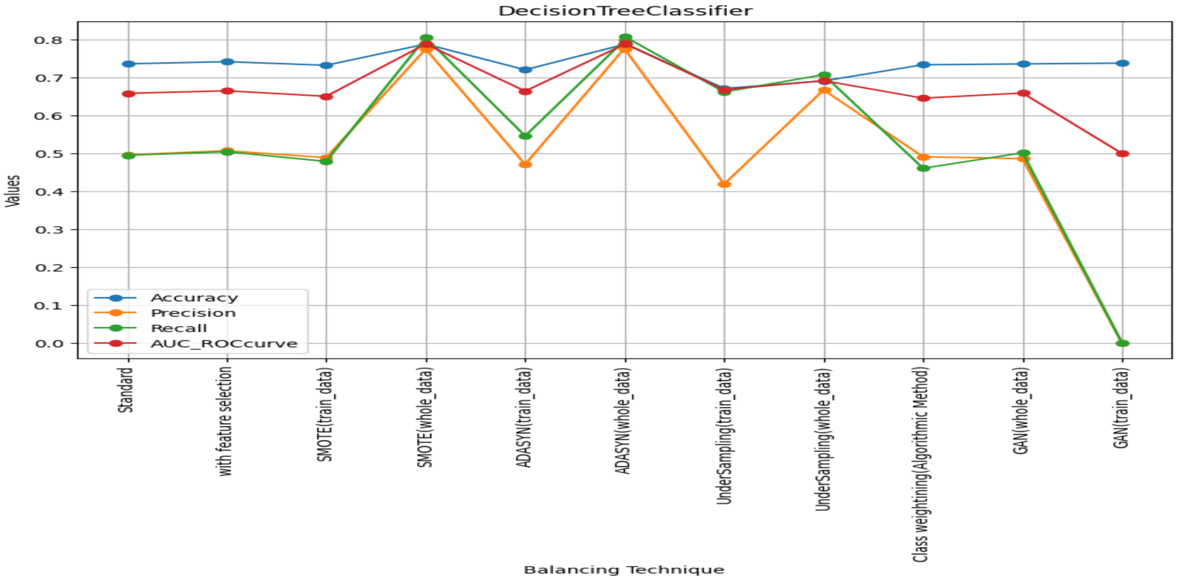


From the graph presented above, it is evident that the performance tends to improve when techniques such as SMOTE, ADASYN, undersampling, and GAN are applied to the entire dataset. Conversely, when these techniques are applied solely to the training datasets, the performance tends to decrease.

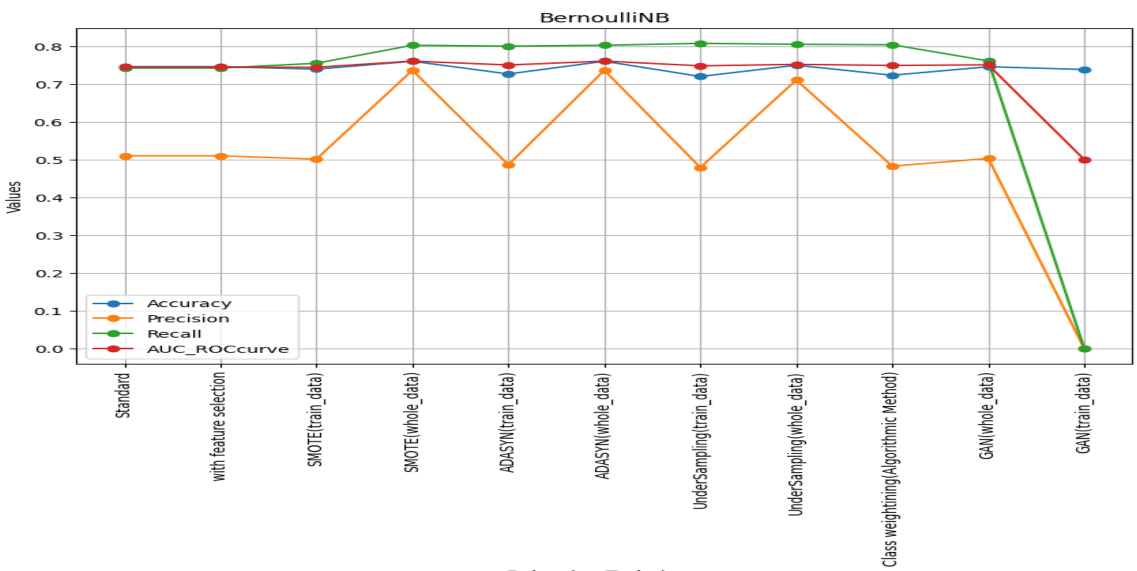


Based on the graph provided above, it can be concluded that the performance shows improvement when applied to the entire dataset for techniques such as SMOTE, ADASYN, undersampling, and GAN.

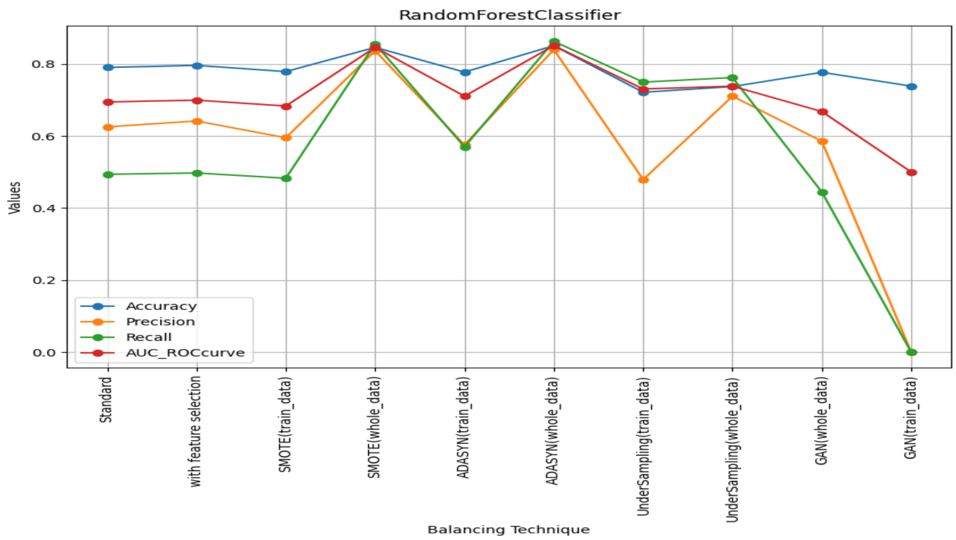
Conversely, the performance tends to decrease when these techniques are applied only to their respective training datasets.



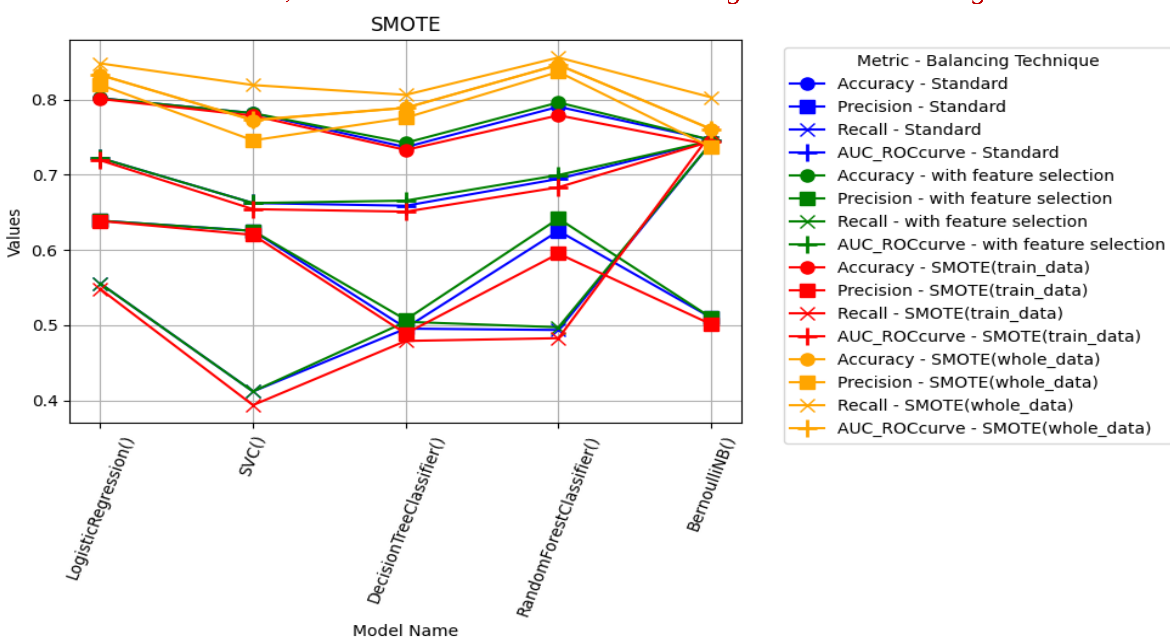
Based on the graph presented above, it appears that the accuracy is increasing in the training dataset while decreasing in the whole dataset. Conversely, there seems to be a vice versa relationship for the whole dataset on techniques such as SMOTE, ADASYN, undersampling, and GAN, where the accuracy tends to increase in the whole dataset and decrease in the respective training datasets.



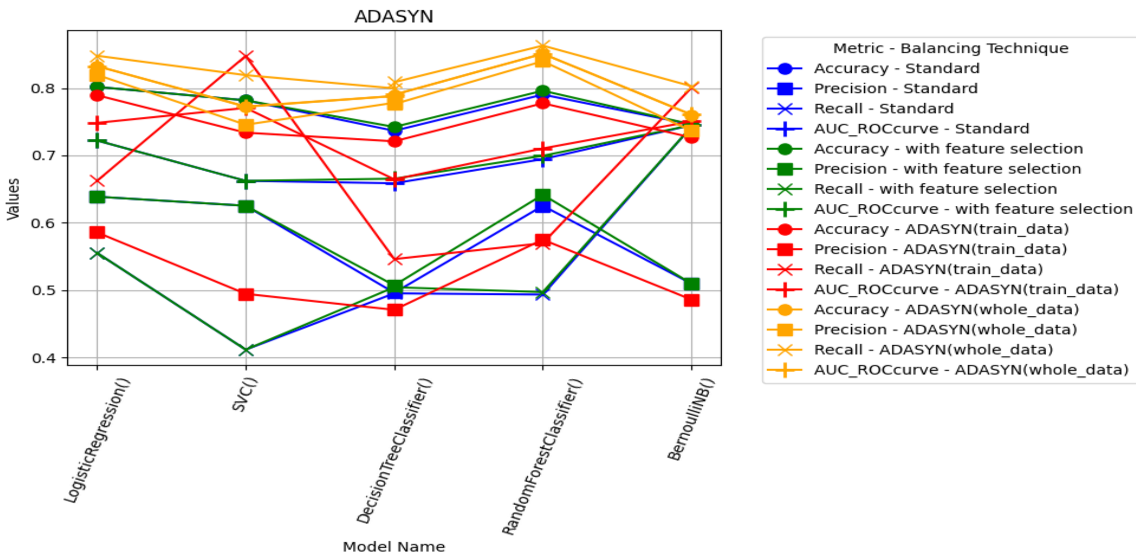
Based on the graph provided above, it can be concluded that the accuracy tends to increase in the training dataset and decrease in the whole dataset. Conversely, there seems to be a vice versa relationship for the whole dataset in techniques such as SMOTE, ADASYN, undersampling, and GAN, where the accuracy tends to increase in the whole dataset and decrease in their respective training datasets.



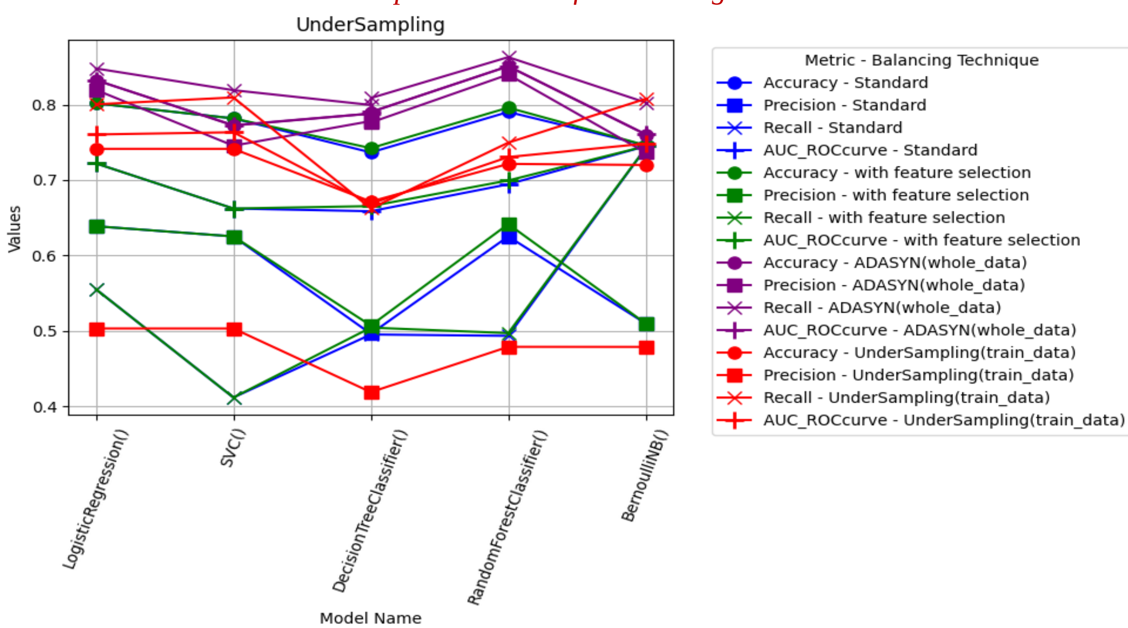
From the graph provided above, it is evident that the performance shows an increasing trend in the whole dataset, whereas it demonstrates a decreasing trend in the training dataset.



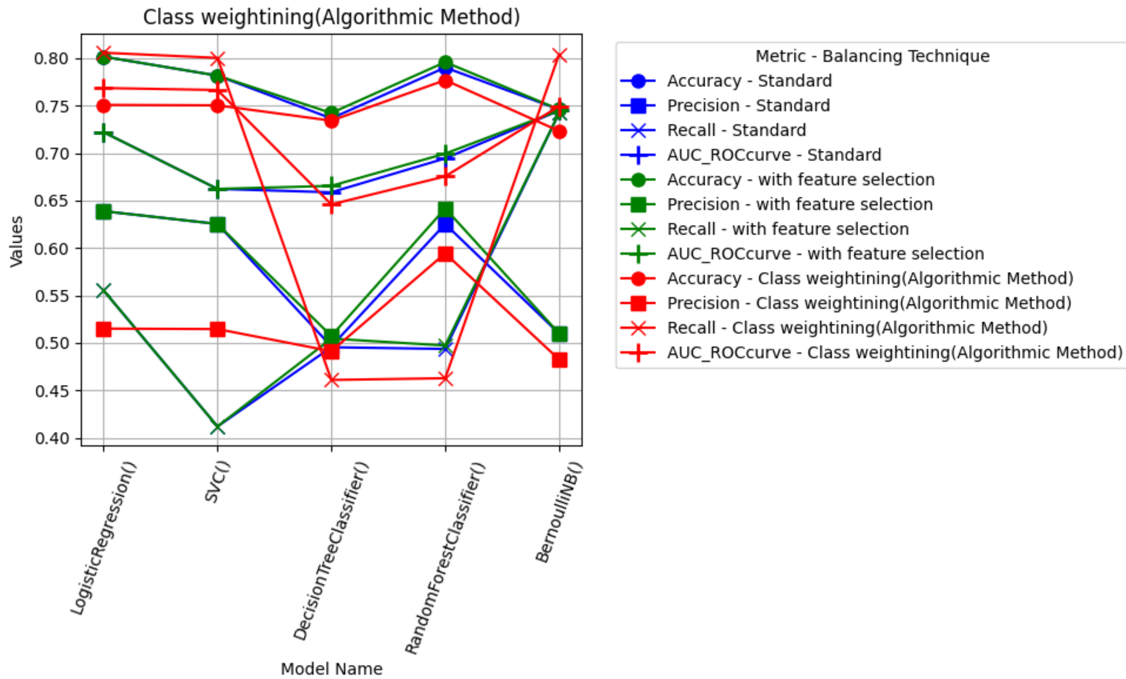
Based on the graph above, it is apparent that logistic regression, Decision Tree Classifier, and Naïve Bayes are exhibiting better performance. Additionally, the performance of the whole dataset surpasses that of the training data.



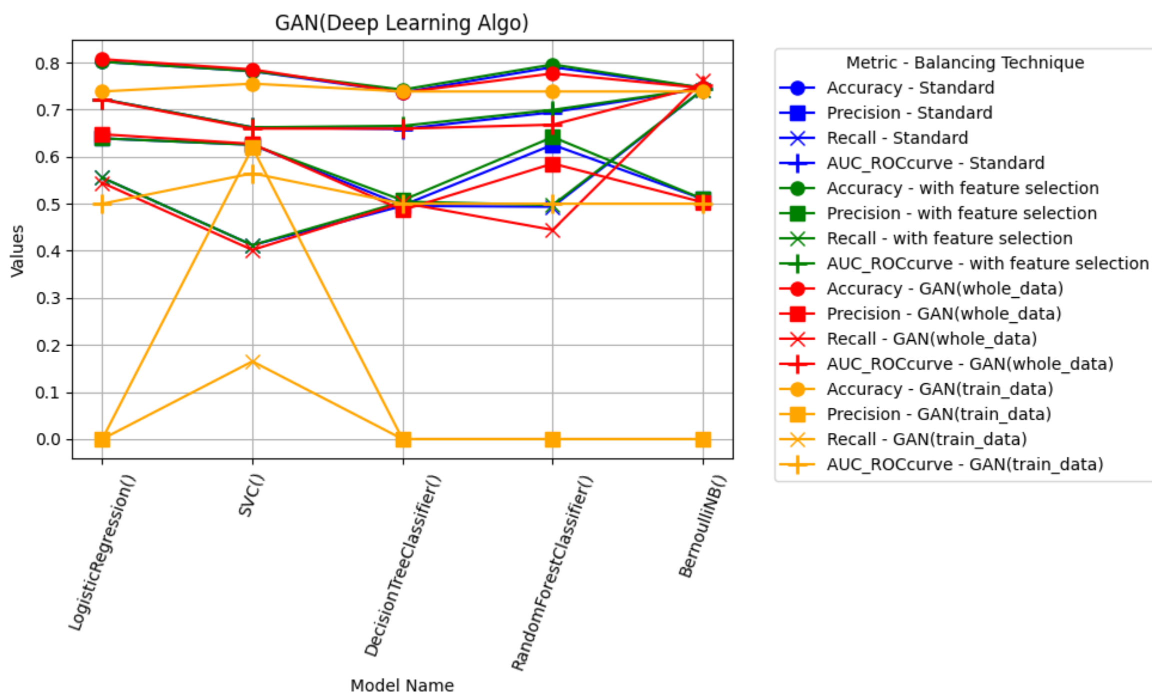
Based on the graph provided above, it can be interpreted that logistic regression and Random Forest Classifier are performing better. Furthermore, the performance of the whole dataset appears to be superior to that of the training data.



Based on the above graph, it can be interpreted that logistic regression and Random Forest Classifier are performing better. Additionally, the performance of the whole dataset is superior to that of the training data.

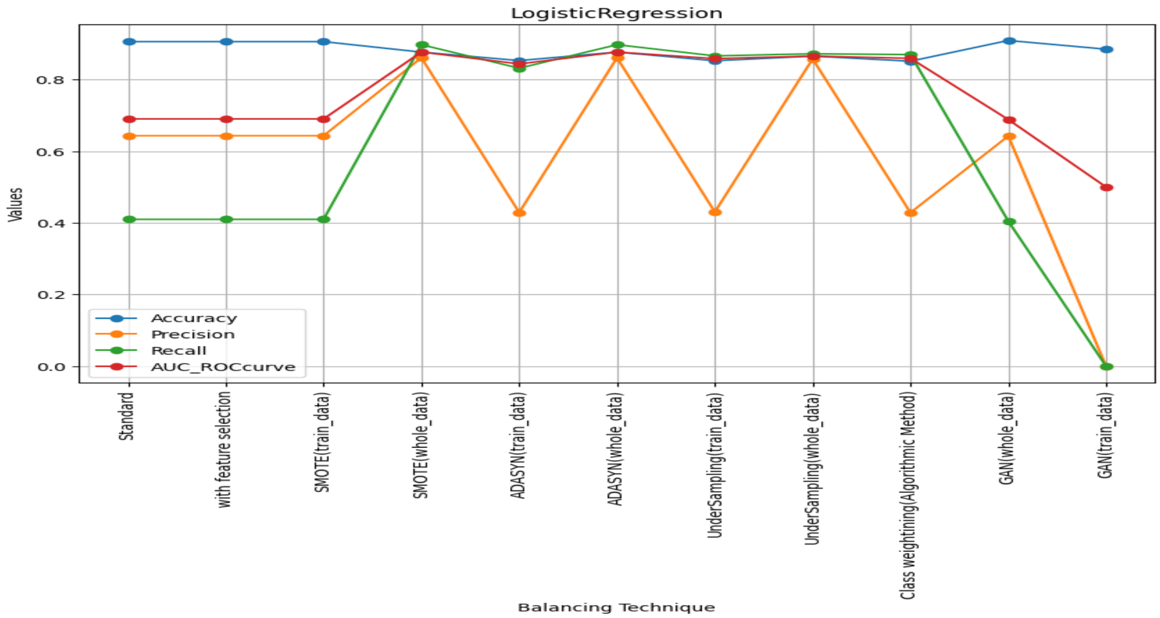


From the above graph, it can be concluded that the performance of algorithmic methods has notably improved on logistic regression, SVC, and Naïve Bayes algorithms.

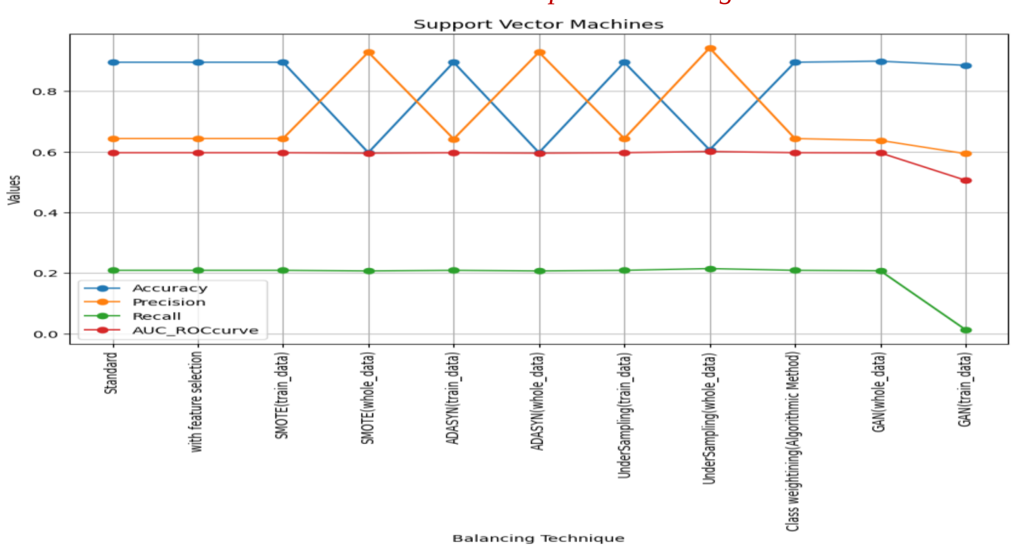


From the above graph, it can be concluded that overall, the performance of the whole dataset is improved compared to the training data in the GAN interpretation.

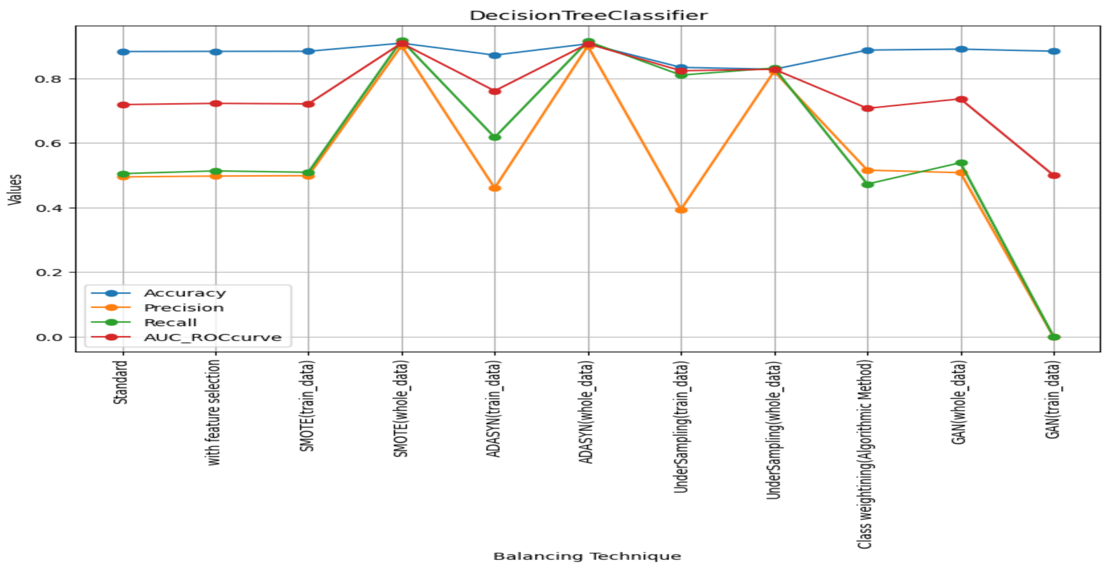
6.2 BANK DATASET



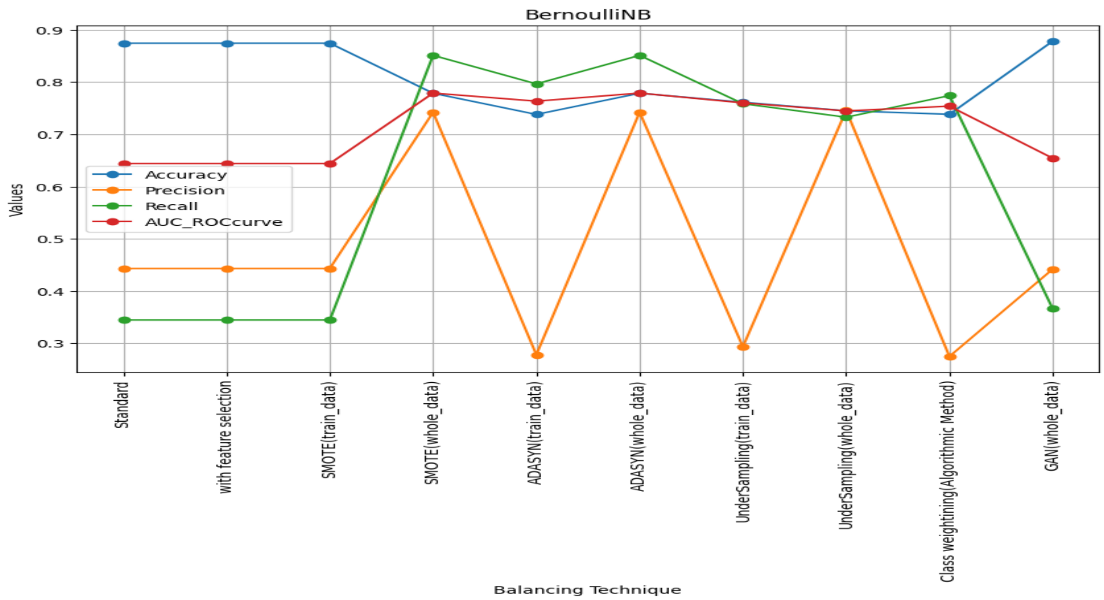
From the graph above, it is apparent that the performance tends to increase on the whole dataset for techniques such as SMOTE, ADASYN, undersampling, and GAN. Conversely, the performance tends to decrease on their respective training datasets.



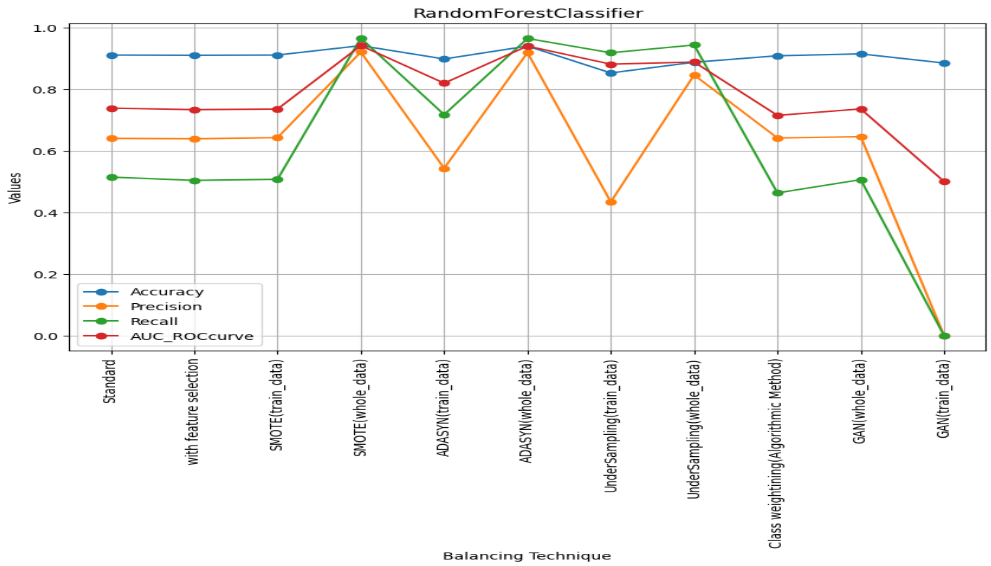
From the graph provided above, it can be observed that the accuracy tends to increase in the training dataset and decrease in the whole dataset. Conversely, there appears to be a vice versa relationship for the whole dataset in techniques such as SMOTE, ADASYN, undersampling, and GAN, where the accuracy tends to increase in the whole dataset and decrease in their respective training datasets.



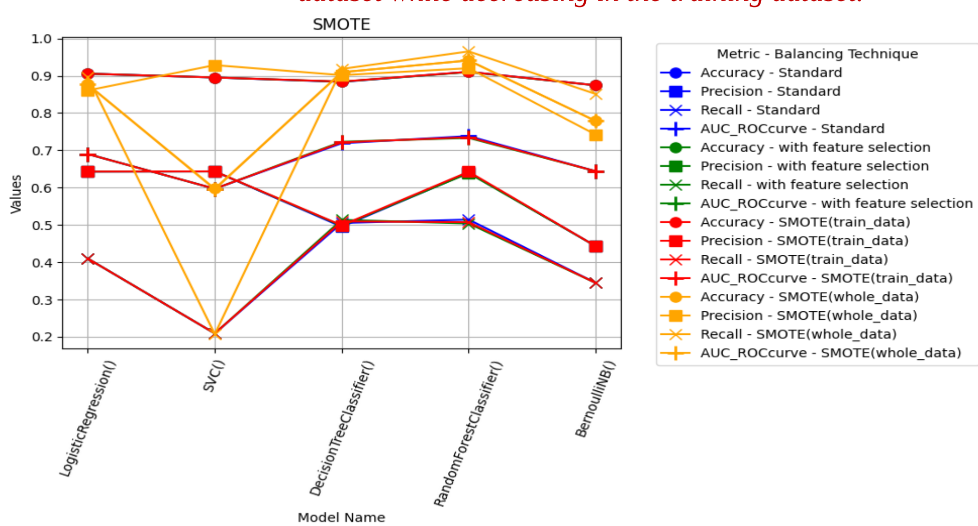
From the above graph, it can be observed that the accuracy tends to increase in the training dataset while decreasing in the whole dataset. Conversely, there seems to be a vice versa relationship for the whole dataset in techniques such as SMOTE, ADASYN, undersampling, and GAN, where the accuracy tends to increase in the whole dataset and decrease in their respective training datasets.



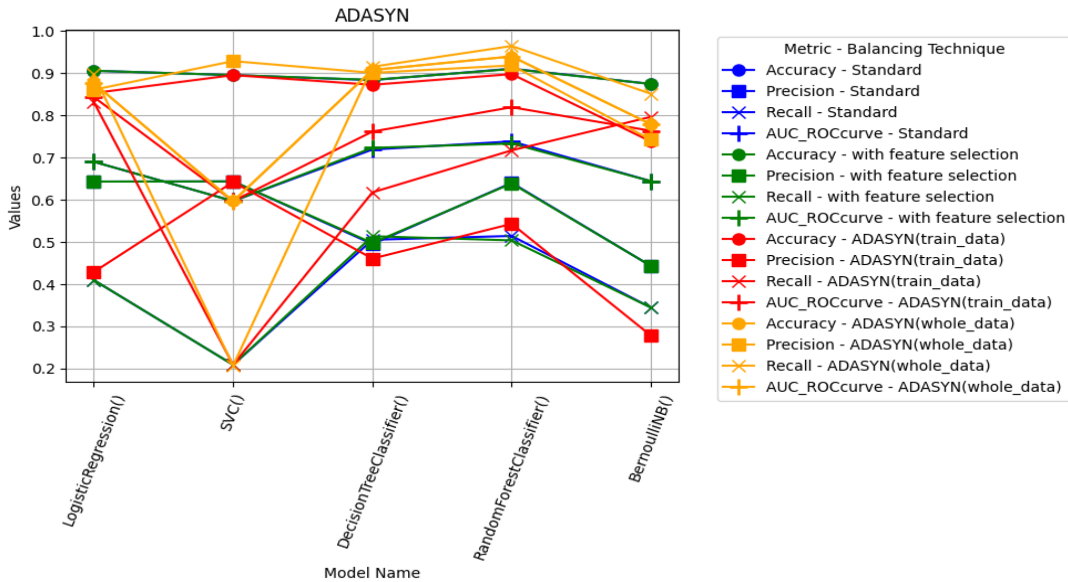
From the above graph, it is evident that the accuracy is increasing in the training dataset and decreasing in the whole dataset. Conversely, there appears to be a vice versa relationship for the whole dataset in techniques such as SMOTE, ADASYN, undersampling, and GAN, where the accuracy tends to increase in the whole dataset and decrease in their respective training datasets.



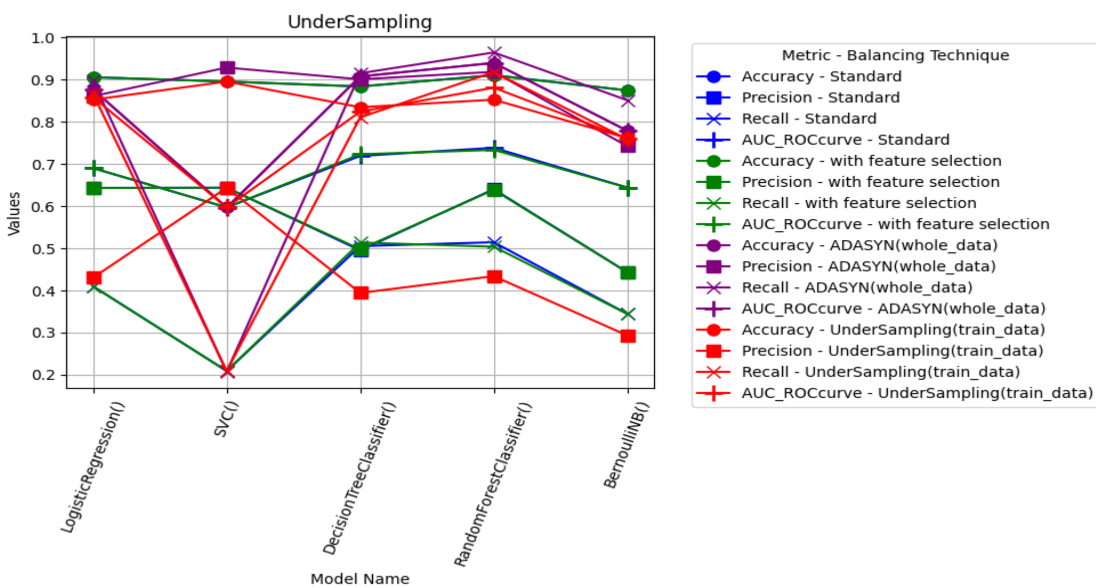
From the graph provided above, it can be inferred that the performance tends to increase in the whole dataset while decreasing in the training dataset.



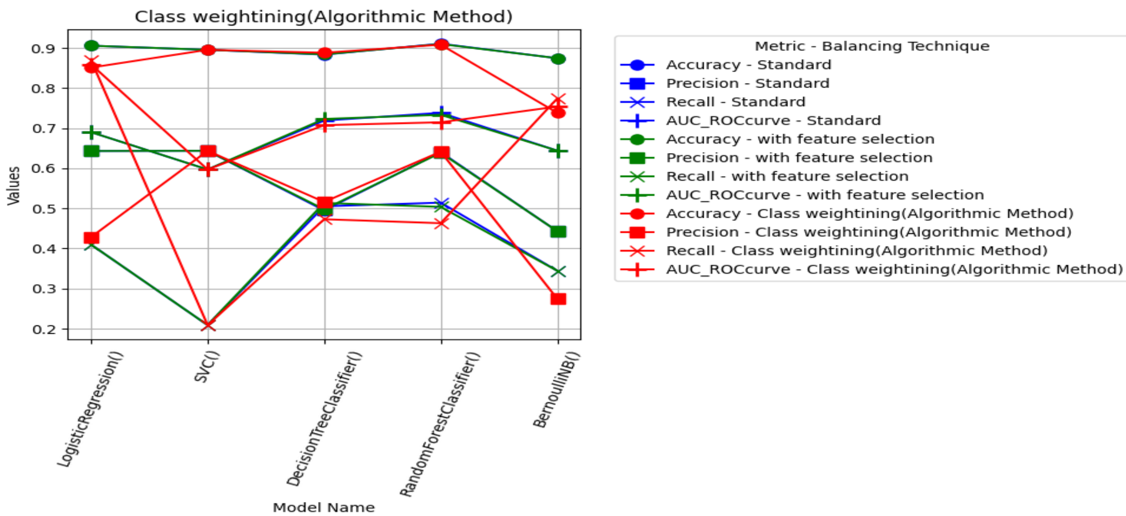
From the above graph, it can be concluded that logistic regression, Decision Tree Classifier, and Random Forest are performing better. Additionally, the performance of the whole dataset is better than that of the training data, except in the case of SVC and Naïve Bayes.



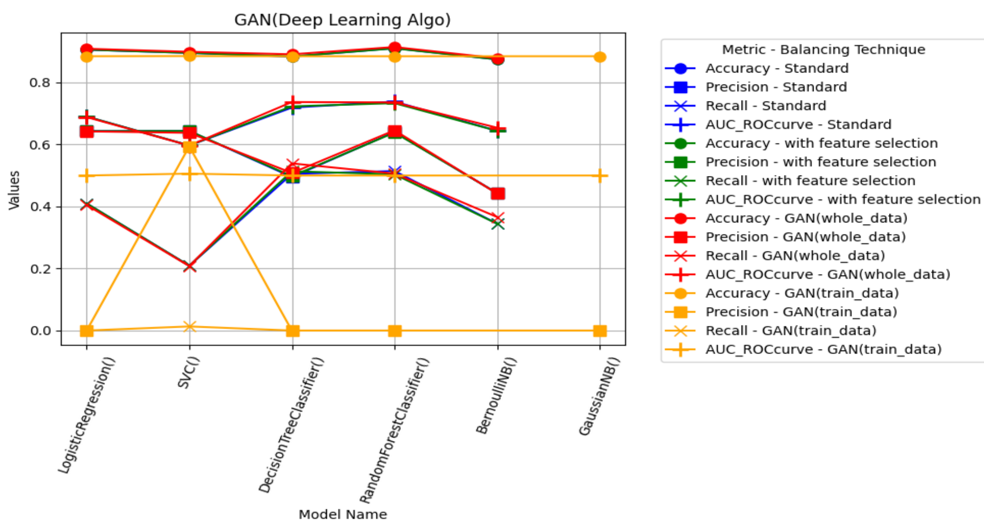
From the above graph, it can be interpreted that logistic regression, Random Forest Classifier, and Decision Tree Classifier are performing better. Additionally, the performance of the whole dataset is better than the training data, except in the case of SVC.



From the above graph, it can be interpreted that logistic regression, Random Forest Classifier, and Decision Tree Classifier are performing better. Additionally, the performance of the whole dataset is better than the training data..

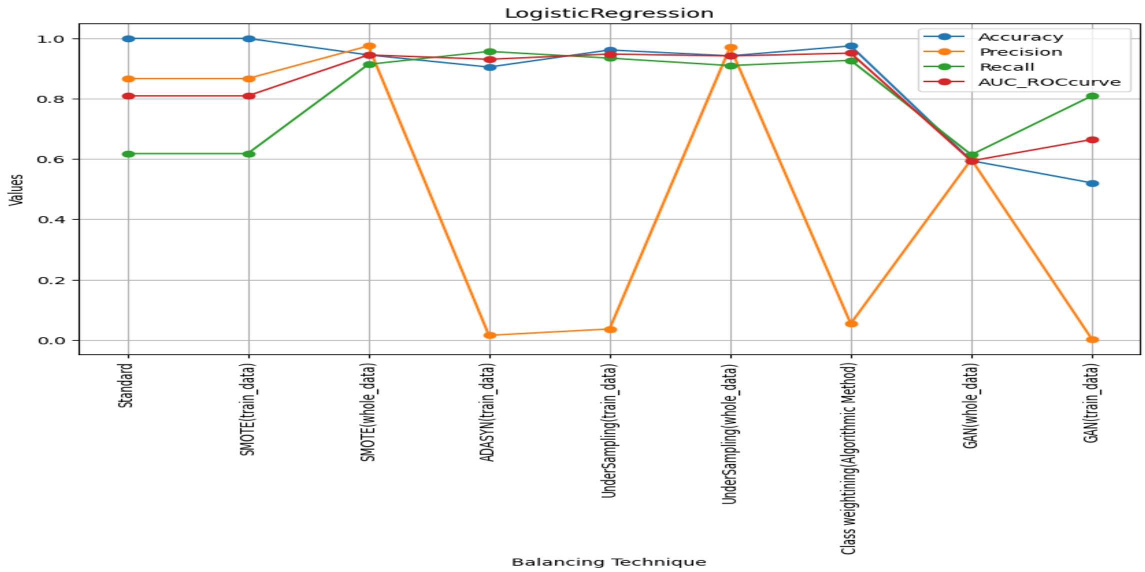


From the above graph, it can be concluded that the performance of algorithmic methods has improved on logistic regression, decision tree classifier, and random forest.

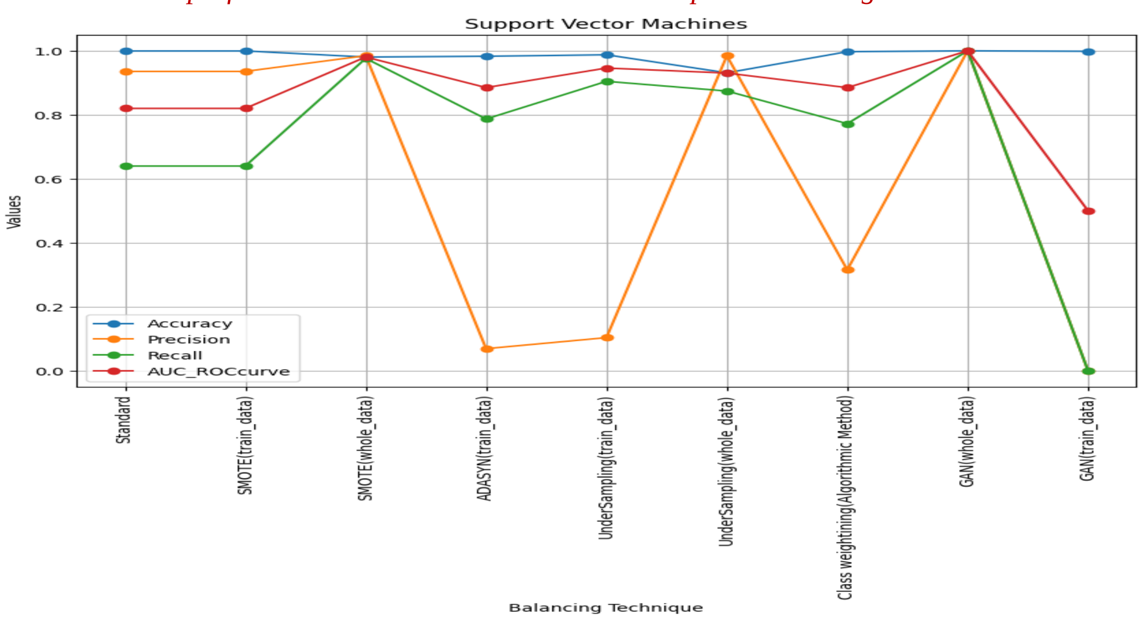


Overall, the performance of the whole dataset is improved compared to the training data in the GAN interpretation.

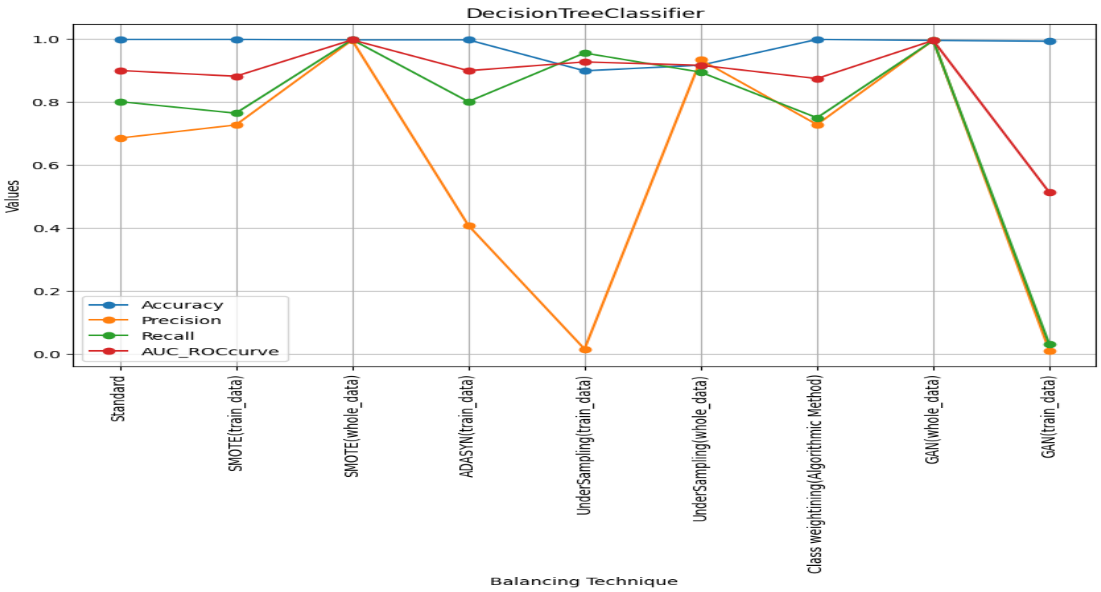
6.3 CREDIT FRAUD DATASET



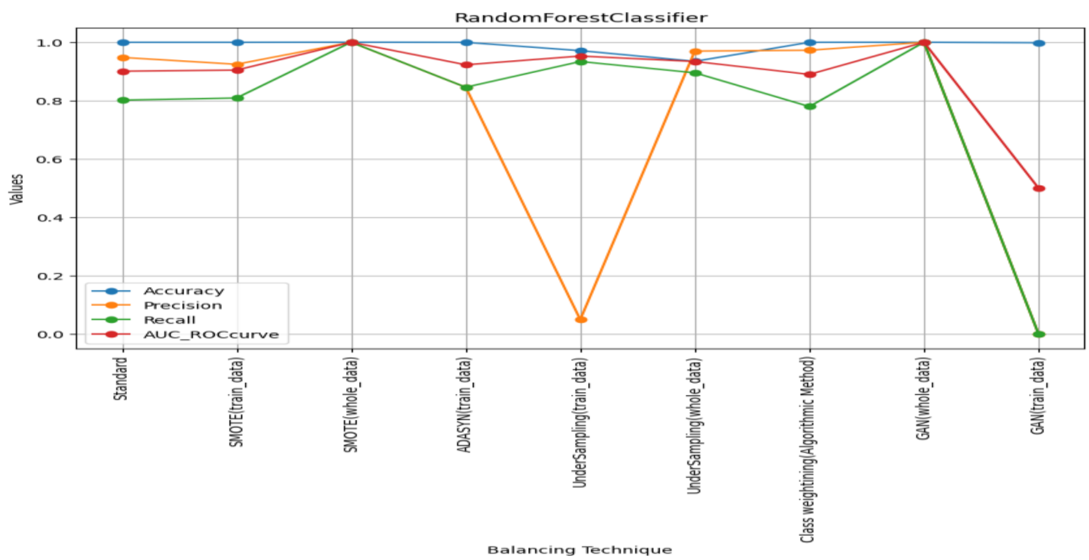
From the graph provided above, it is evident that the performance tends to increase on the whole dataset for techniques such as SMOTE, ADASYN, undersampling, and GAN. Conversely, the performance tends to decrease on their respective training datasets.



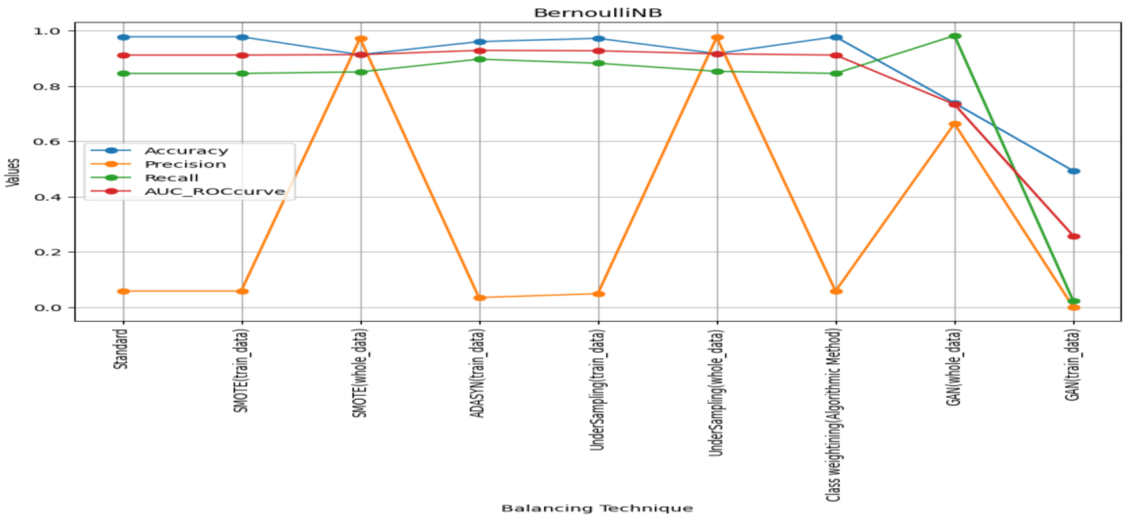
From the above graph, it appears that the performance tends to increase on the whole dataset for techniques such as SMOTE, ADASYN, undersampling, and GAN, while decreasing on their respective training datasets.



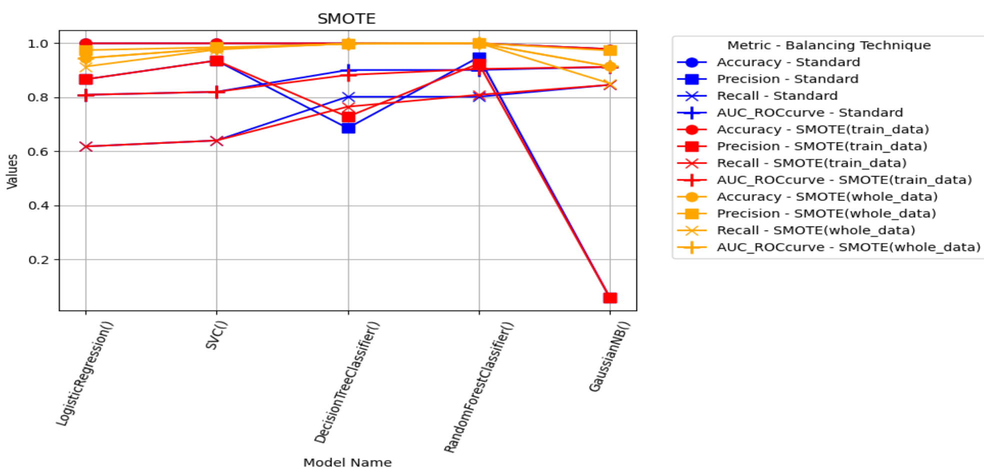
From the above graph, it can be observed that the accuracy tends to increase in the training dataset and decrease in the whole dataset. Conversely, there seems to be a vice versa relationship for the whole dataset in techniques such as SMOTE, ADASYN, undersampling, and GAN, where the accuracy tends to increase in the whole dataset and decrease in their respective training datasets.



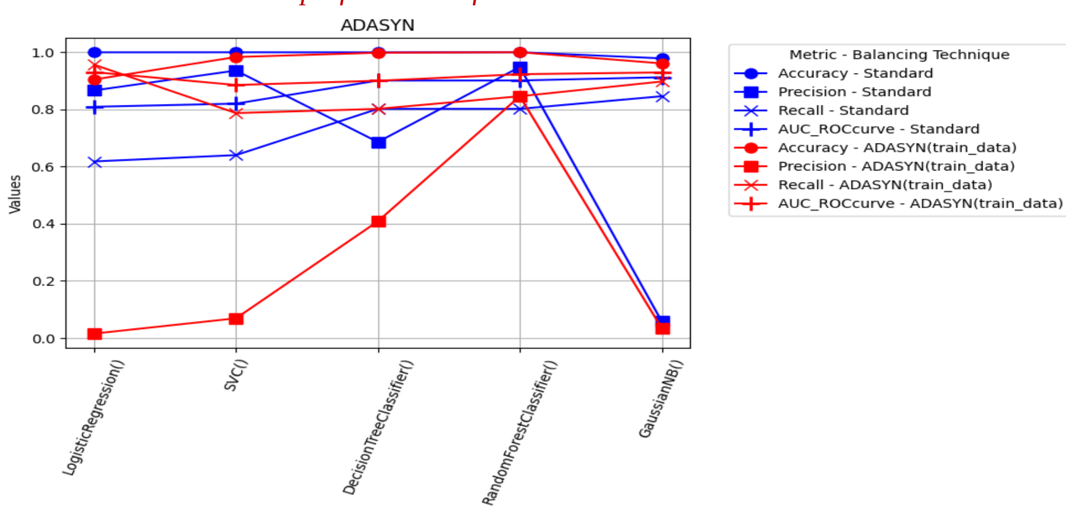
From the graph provided above, it can be inferred that the performance tends to increase in the whole dataset while decreasing in the training dataset.



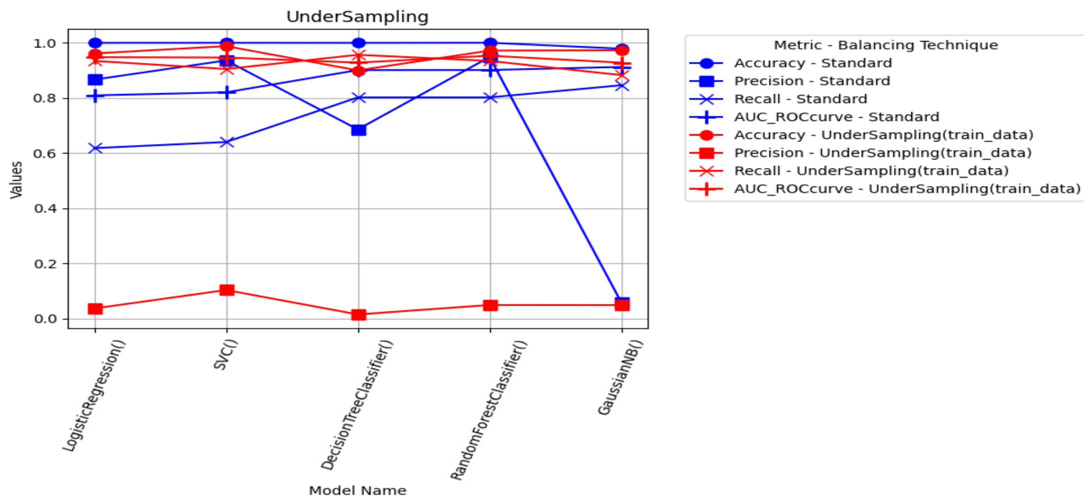
From the above graph, it is evident that the accuracy is increasing in the training dataset and decreasing in the whole dataset. Conversely, there appears to be a vice versa relationship for the whole dataset in techniques such as SMOTE, ADASYN, undersampling, and GAN, where the accuracy tends to increase in the whole dataset and decrease in their respective training datasets.



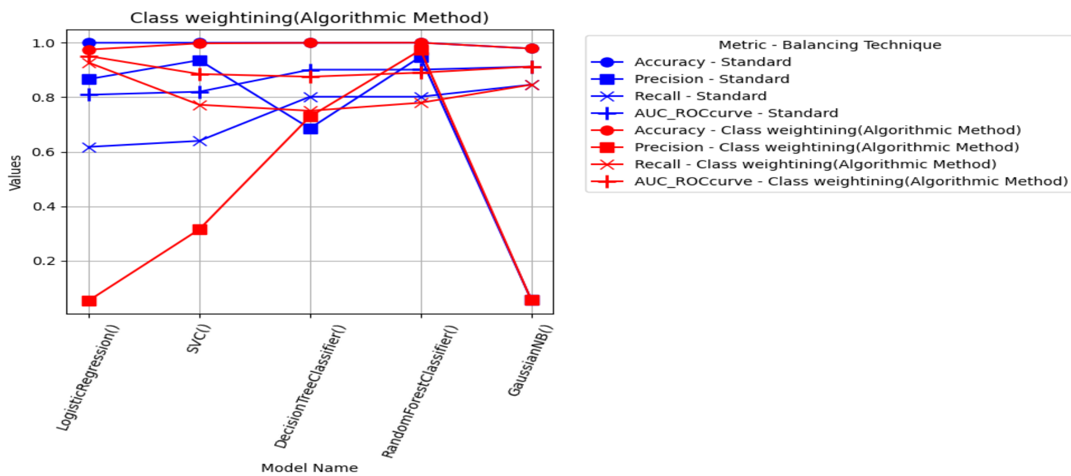
The performance of whole data is better than the train



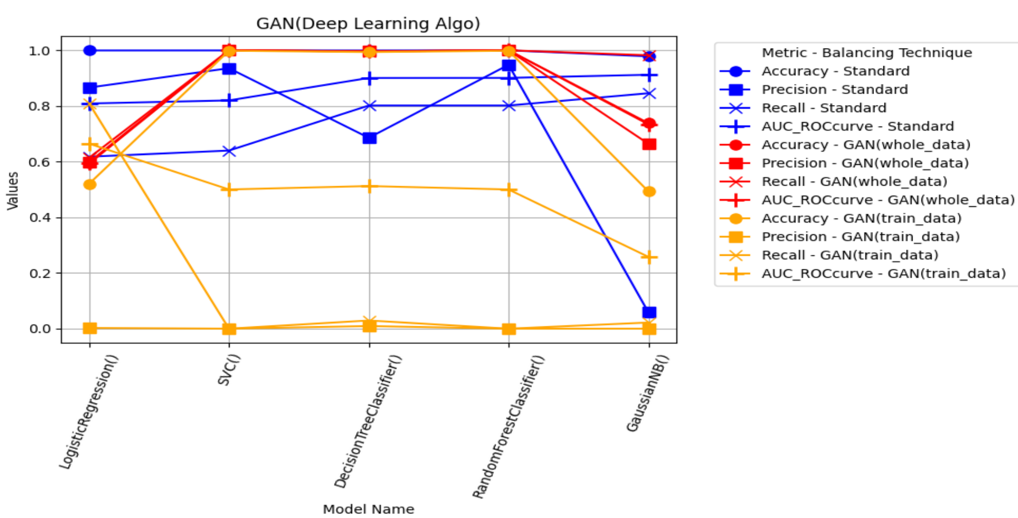
Performance of whole data is better than the train data except in the case of SVC.



Performance of whole data is better than the train data.



From the above graph it can be concluded as the performance of algorithmic method is improved on decision tree classifier, and random forest



Over all the performance of whole data is improved as compared to the train data in GAN interpretation.

7 Conclusion

Based on the information provided, the following conclusions can be drawn:

1. The performance of techniques such as SMOTE, ADASYN, Undersampling, and GAN applied solely on the training data shows a decreasing trend in accuracy while increasing on the whole dataset.

2. Overall, logistic regression appears to be the better classifier compared to other algorithms studied in this project.

These conclusions suggest that while techniques like SMOTE, ADASYN, Undersampling, and GAN may improve performance when applied to the whole dataset, they might not have the same effect when applied solely on the training data. Additionally, logistic regression emerges as the most effective classifier among the algorithms evaluated in this study.

8 ANNEXURE-I DATA SUMMARIES

8.1 CHURN DATASET

	type	count	nunique	%unique	null	%null	min	max
gender	object	7021	2	0.028486	0	0.0	Female	Male
SeniorCitizen	int64	7021	2	0.028486	0	0.0	0	1
Partner	object	7021	2	0.028486	0	0.0	No	Yes
Dependents	object	7021	2	0.028486	0	0.0	No	Yes
tenure	int64	7021	73	1.039738	0	0.0	0	72
PhoneService	object	7021	2	0.028486	0	0.0	No	Yes
MultipleLines	object	7021	3	0.042729	0	0.0	No	Yes
InternetService	object	7021	3	0.042729	0	0.0	DSL	No
OnlineSecurity	object	7021	3	0.042729	0	0.0	No	Yes
OnlineBackup	object	7021	3	0.042729	0	0.0	No	Yes
DeviceProtection	object	7021	3	0.042729	0	0.0	No	Yes
TechSupport	object	7021	3	0.042729	0	0.0	No	Yes
StreamingTV	object	7021	3	0.042729	0	0.0	No	Yes
StreamingMovies	object	7021	3	0.042729	0	0.0	No	Yes
Contract	object	7021	3	0.042729	0	0.0	Month-to-month	Two year
PaperlessBilling	object	7021	2	0.028486	0	0.0	No	Yes
PaymentMethod	object	7021	4	0.056972	0	0.0	Bank transfer (automatic)	Mailed check
MonthlyCharges	float64	7021	1585	22.575132	0	0.0	18.25	118.75
TotalCharges	float64	7021	6531	93.020937	0	0.0	18.8	8684.8
Churn	object	7021	2	0.028486	0	0.0	No	Yes

8.2 BANK MARKETING DATASET

	type	count	nunique	%unique	null	%null	min	max
age	int64	41175	78	0.189435	0	0.0	17	98
job	object	41175	12	0.029144	0	0.0	admin.	unknown
marital	object	41175	4	0.009715	0	0.0	divorced	unknown
education	object	41175	8	0.019429	0	0.0	basic.4y	unknown
default	object	41175	3	0.007286	0	0.0	no	yes
housing	object	41175	3	0.007286	0	0.0	no	yes
loan	object	41175	3	0.007286	0	0.0	no	yes
contact	object	41175	2	0.004857	0	0.0	cellular	telephone
month	object	41175	10	0.024287	0	0.0	apr	sep
day_of_week	object	41175	5	0.012143	0	0.0	fri	wed
duration	int64	41175	1544	3.749848	0	0.0	0	4918
campaign	int64	41175	42	0.102004	0	0.0	1	56
pdays	int64	41175	27	0.065574	0	0.0	0	999
previous	int64	41175	8	0.019429	0	0.0	0	7
poutcome	object	41175	3	0.007286	0	0.0	failure	success
emp.var.rate	float64	41175	10	0.024287	0	0.0	-3.4	1.4
cons.price.idx	float64	41175	26	0.063145	0	0.0	92.201	94.767
cons.conf.idx	float64	41175	26	0.063145	0	0.0	-50.8	-26.9
euribor3m	float64	41175	316	0.767456	0	0.0	0.634	5.045
nr.employed	float64	41175	11	0.026715	0	0.0	4963.6	5228.1
y	object	41175	2	0.004857	0	0.0	no	yes

8.3 CREDIT FRAUD DATASET

	type	count	nunique	%unique	null	%null	min	max
Time	float64	284807	124592	43.746116	0	0.0	0.000000	172792.000000
V1	float64	284807	275663	96.789405	0	0.0	-56.407510	2.454930
V2	float64	284807	275663	96.789405	0	0.0	-72.715728	22.057729
V3	float64	284807	275663	96.789405	0	0.0	-48.325589	9.382558
V4	float64	284807	275663	96.789405	0	0.0	-5.683171	16.875344
V5	float64	284807	275663	96.789405	0	0.0	-113.743307	34.801666
V6	float64	284807	275663	96.789405	0	0.0	-26.160506	73.301626
V7	float64	284807	275663	96.789405	0	0.0	-43.557242	120.589494
V8	float64	284807	275663	96.789405	0	0.0	-73.216718	20.007208
V9	float64	284807	275663	96.789405	0	0.0	-13.434066	15.594995
V10	float64	284807	275663	96.789405	0	0.0	-24.588262	23.745136
V11	float64	284807	275663	96.789405	0	0.0	-4.797473	12.018913
V12	float64	284807	275663	96.789405	0	0.0	-18.683715	7.848392
V13	float64	284807	275663	96.789405	0	0.0	-5.791881	7.126883
V14	float64	284807	275663	96.789405	0	0.0	-19.214325	10.526766
V15	float64	284807	275663	96.789405	0	0.0	-4.498945	8.877742
V16	float64	284807	275663	96.789405	0	0.0	-14.129855	17.315112
V17	float64	284807	275663	96.789405	0	0.0	-25.162799	9.253526
V18	float64	284807	275663	96.789405	0	0.0	-9.498746	5.041069
V19	float64	284807	275663	96.789405	0	0.0	-7.213527	5.591971
V20	float64	284807	275663	96.789405	0	0.0	-54.497720	39.420904
V21	float64	284807	275663	96.789405	0	0.0	-34.830382	27.202839
V22	float64	284807	275663	96.789405	0	0.0	-10.933144	10.503090
V23	float64	284807	275663	96.789405	0	0.0	-44.807735	22.528412
V24	float64	284807	275663	96.789405	0	0.0	-2.836627	4.584549
V25	float64	284807	275663	96.789405	0	0.0	-10.295397	7.519589
V26	float64	284807	275663	96.789405	0	0.0	-2.604551	3.517346
V27	float64	284807	275663	96.789405	0	0.0	-22.565679	31.612198
V28	float64	284807	275663	96.789405	0	0.0	-15.430084	33.847808
Amount	float64	284807	32767	11.504984	0	0.0	0.000000	25691.160000
Class	int64	284807	2	0.000702	0	0.0	0.000000	1.000000

9 ANNEXURE-II-RESULTS OF DATASETS

9.1 CHURN DATASET

Balancing Technique	Model Name	Accuracy	Precision	Recall	AUC_ROC
None	LogisticRegression	0.80	0.64	0.56	0.72
None	SVC	0.78	0.63	0.41	0.66
None	DecisionTreeClassifier	0.74	0.50	0.50	0.66
None	RandomForestClassifier	0.79	0.63	0.49	0.69
None	BernoulliNB	0.75	0.51	0.74	0.74
with feature selection	LogisticRegression	0.80	0.64	0.56	0.72
with feature selection	SVC	0.78	0.63	0.41	0.66
with feature selection	DecisionTreeClassifier	0.74	0.51	0.50	0.67
with feature selection	RandomForestClassifier	0.80	0.64	0.50	0.70
with feature selection	BernoulliNB	0.75	0.51	0.74	0.74
SMOTE(train_data)	LogisticRegression	0.80	0.64	0.55	0.72
SMOTE(train_data)	SVC	0.78	0.62	0.39	0.65
SMOTE(train_data)	DecisionTreeClassifier	0.73	0.49	0.48	0.65
SMOTE(train_data)	RandomForestClassifier	0.78	0.60	0.48	0.68
SMOTE(train_data)	BernoulliNB	0.74	0.50	0.75	0.74
SMOTE(whole_data)	LogisticRegression	0.83	0.82	0.85	0.83
SMOTE(whole_data)	SVC	0.77	0.75	0.82	0.77
SMOTE(whole_data)	DecisionTreeClassifier	0.79	0.78	0.81	0.79
SMOTE(whole_data)	RandomForestClassifier	0.85	0.84	0.86	0.85
SMOTE(whole_data)	BernoulliNB	0.76	0.74	0.80	0.76
ADASYN(train_data)	LogisticRegression	0.79	0.59	0.66	0.75
ADASYN(train_data)	SVC	0.73	0.49	0.85	0.77
ADASYN(train_data)	DecisionTreeClassifier	0.72	0.47	0.55	0.66
ADASYN(train_data)	RandomForestClassifier	0.78	0.58	0.57	0.71
ADASYN(train_data)	BernoulliNB	0.73	0.49	0.80	0.75
ADASYN(whole_data)	LogisticRegression	0.83	0.82	0.85	0.83
ADASYN(whole_data)	SVC	0.77	0.75	0.82	0.77
ADASYN(whole_data)	DecisionTreeClassifier	0.79	0.78	0.80	0.79
ADASYN(whole_data)	DecisionTreeClassifier	0.79	0.78	0.81	0.79
ADASYN(whole_data)	RandomForestClassifier	0.85	0.84	0.86	0.85
ADASYN(whole_data)	BernoulliNB	0.76	0.74	0.80	0.76
UnderSampling(train_data)	LogisticRegression	0.74	0.50	0.80	0.76
UnderSampling(train_data)	SVC	0.74	0.50	0.81	0.76
UnderSampling(train_data)	DecisionTreeClassifier	0.67	0.42	0.66	0.67
UnderSampling(train_data)	RandomForestClassifier	0.72	0.48	0.75	0.73
UnderSampling(train_data)	BernoulliNB	0.72	0.48	0.81	0.75

UnderSampling(whole_data)	**LogisticRegression**	**0.76**	**0.73**	**0.80**	**0.76**
UnderSampling(whole_data)	**SVC**	**0.75**	**0.72**	**0.77**	**0.75**
UnderSampling(whole_data)	**DecisionTreeClassifier**	**0.69**	**0.67**	**0.71**	**0.69**
UnderSampling(whole_data)	**RandomForestClassifier**	**0.74**	**0.71**	**0.76**	**0.74**
UnderSampling(whole_data)	**BernoulliNB**	**0.75**	**0.71**	**0.81**	**0.75**
Class weightining(Algorithmic Method)	**LogisticRegression**	**0.75**	**0.52**	**0.81**	**0.77**
Class weightining(Algorithmic Method)	**SVC**	**0.75**	**0.51**	**0.80**	**0.77**
Class weightining(Algorithmic Method)	**DecisionTreeClassifier**	**0.73**	**0.49**	**0.46**	**0.65**
Class weightining(Algorithmic Method)	**RandomForestClassifier**	**0.78**	**0.59**	**0.46**	**0.68**
Class weightining(Algorithmic Method)	**BernoulliNB**	**0.72**	**0.48**	**0.80**	**0.75**
GAN(whole_data)	**LogisticRegression**	**0.81**	**0.65**	**0.54**	**0.72**
GAN(whole_data)	**SVC**	**0.79**	**0.63**	**0.40**	**0.66**
GAN(whole_data)	**DecisionTreeClassifier**	**0.74**	**0.49**	**0.50**	**0.66**
GAN(whole_data)	**RandomForestClassifier**	**0.78**	**0.59**	**0.44**	**0.67**
GAN(whole_data)	**BernoulliNB**	**0.75**	**0.50**	**0.76**	**0.75**
GAN(train_data)	**LogisticRegression**	**0.74**	**0.00**	**0.00**	**0.50**
GAN(train_data)	**SVC**	**0.76**	**0.62**	**0.17**	**0.56**
GAN(train_data)	**DecisionTreeClassifier**	**0.74**	**0.00**	**0.00**	**0.50**
GAN(train_data)	**RandomForestClassifier**	**0.74**	**0.00**	**0.00**	**0.50**
GAN(train_data)	**BernoulliNB**	**0.74**	**0.00**	**0.00**	**0.50**

9.2 BANK MARKETING DATASET

Balancing Technique	Model Name	Accuracy	Precision	Recall	AUC_ROC
None	**LogisticRegression**	**0.64**	**0.41**	**0.69**	**0.64**
None	**SVC**	**0.64**	**0.21**	**0.60**	**0.64**
None	**DecisionTreeClassifier**	**0.50**	**0.51**	**0.72**	**0.50**
None	**RandomForestClassifier**	**0.64**	**0.51**	**0.74**	**0.64**
None	**BernoulliNB**	**0.44**	**0.34**	**0.64**	**0.44**
with feature selection	**LogisticRegression**	**0.64**	**0.41**	**0.69**	**0.64**
with feature selection	**SVC**	**0.64**	**0.21**	**0.60**	**0.64**
with feature selection	**DecisionTreeClassifier**	**0.50**	**0.51**	**0.72**	**0.50**
with feature selection	**RandomForestClassifier**	**0.64**	**0.50**	**0.73**	**0.64**
with feature selection	**BernoulliNB**	**0.44**	**0.34**	**0.64**	**0.44**

SMOTE(train_data)	LogisticRegression	0.64	0.41	0.69	0.64
SMOTE(train_data)	SVC	0.64	0.21	0.60	0.64
SMOTE(train_data)	DecisionTreeClassifier	0.50	0.51	0.72	0.50
SMOTE(train_data)	RandomForestClassifier	0.64	0.51	0.74	0.64
SMOTE(train_data)	BernoulliNB	0.44	0.34	0.64	0.44
SMOTE(whole_data)	LogisticRegression	0.86	0.90	0.88	0.86
SMOTE(whole_data)	SVC	0.93	0.21	0.60	0.93
SMOTE(whole_data)	DecisionTreeClassifier	0.90	0.92	0.91	0.90
SMOTE(whole_data)	RandomForestClassifier	0.92	0.97	0.94	0.92
SMOTE(whole_data)	BernoulliNB	0.74	0.85	0.78	0.74
ADASYN(train_data)	LogisticRegression	0.43	0.83	0.84	0.43
ADASYN(train_data)	SVC	0.64	0.21	0.60	0.64
ADASYN(train_data)	DecisionTreeClassifier	0.46	0.62	0.76	0.46
ADASYN(train_data)	RandomForestClassifier	0.54	0.72	0.82	0.54
ADASYN(train_data)	BernoulliNB	0.28	0.80	0.76	0.28
ADASYN(whole_data)	LogisticRegression	0.86	0.90	0.88	0.86
ADASYN(whole_data)	SVC	0.93	0.21	0.60	0.93
ADASYN(whole_data)	DecisionTreeClassifier	0.90	0.92	0.91	0.90
ADASYN(whole_data)	DecisionTreeClassifier	0.92	0.96	0.94	0.92
ADASYN(whole_data)	RandomForestClassifier	0.74	0.85	0.78	0.74
ADASYN(whole_data)	BernoulliNB	0.43	0.87	0.86	0.43
UnderSampling(train_data)	LogisticRegression	0.64	0.21	0.60	0.64
UnderSampling(train_data)	SVC	0.39	0.81	0.82	0.39
UnderSampling(train_data)	DecisionTreeClassifier	0.43	0.92	0.88	0.43
UnderSampling(train_data)	RandomForestClassifier	0.29	0.76	0.76	0.29
UnderSampling(train_data)	BernoulliNB	0.86	0.87	0.87	0.86
UnderSampling(whole_data)	LogisticRegression	0.94	0.21	0.60	0.94
UnderSampling(whole_data)	SVC	0.82	0.83	0.83	0.82
UnderSampling(whole_data)	DecisionTreeClassifier	0.85	0.94	0.89	0.85
UnderSampling(whole_data)	RandomForestClassifier	0.75	0.73	0.74	0.75
UnderSampling(whole_data)	BernoulliNB	0.43	0.87	0.86	0.43
Class weightining(Algorithmic Method)	LogisticRegression	0.64	0.21	0.60	0.64
Class weightining(Algorithmic Method)	SVC	0.52	0.47	0.71	0.52
Class weightining(Algorithmic Method)	DecisionTreeClassifier	0.64	0.46	0.71	0.64
Class weightining(Algorithmic Method)	RandomForestClassifier	0.27	0.77	0.75	0.27

Class weightining(Algorithmic Method)	**BernoulliNB**	**0.64**	**0.40**	**0.69**	**0.64**
GAN(whole_data)	**LogisticRegression**	**0.64**	**0.21**	**0.60**	**0.64**
GAN(whole_data)	**SVC**	**0.51**	**0.54**	**0.74**	**0.51**
GAN(whole_data)	**DecisionTreeClassifier**	**0.65**	**0.51**	**0.74**	**0.65**
GAN(whole_data)	**RandomForestClassifier**	**0.44**	**0.37**	**0.65**	**0.44**
GAN(whole_data)	**BernoulliNB**	**0.00**	**0.00**	**0.50**	**0.00**
GAN(train_data)	**LogisticRegression**	**0.59**	**0.01**	**0.51**	**0.59**
GAN(train_data)	**SVC**	**0.00**	**0.00**	**0.50**	**0.00**
GAN(train_data)	**DecisionTreeClassifier**	**0.00**	**0.00**	**0.50**	**0.00**
GAN(train_data)	**RandomForestClassifier**	**0.00**	**0.00**	**0.50**	**0.00**
GAN(train_data)	**BernoulliNB**	**0.64**	**0.41**	**0.69**	**0.64**

9.3 CREDIT FRAUD DATASET

Balancing Technique	Model Name	Accuracy	Precision	Recall	AUC_ROC
None	**LogisticRegression**	**0.87**	**0.62**	**0.81**	**0.87**
None	**SVC**	**0.94**	**0.64**	**0.82**	**0.94**
None	**DecisionTreeClassifier**	**0.69**	**0.80**	**0.90**	**0.69**
None	**RandomForestClassifier**	**0.95**	**0.80**	**0.90**	**0.95**
None	**BernoulliNB**	**0.06**	**0.85**	**0.91**	**0.06**
with feature selection	**LogisticRegression**	**0.87**	**0.62**	**0.81**	**0.87**
with feature selection	**SVC**	**0.94**	**0.64**	**0.82**	**0.94**
with feature selection	**DecisionTreeClassifier**	**0.73**	**0.76**	**0.88**	**0.73**
with feature selection	**RandomForestClassifier**	**0.92**	**0.81**	**0.90**	**0.92**
with feature selection	**BernoulliNB**	**0.06**	**0.85**	**0.91**	**0.06**
SMOTE(train_data)	**LogisticRegression**	**0.97**	**0.91**	**0.94**	**0.97**
SMOTE(train_data)	**SVC**	**0.98**	**0.98**	**0.98**	**0.98**
SMOTE(train_data)	**DecisionTreeClassifier**	**1.00**	**1.00**	**1.00**	**1.00**
SMOTE(train_data)	**RandomForestClassifier**	**1.00**	**1.00**	**1.00**	**1.00**
SMOTE(train_data)	**BernoulliNB**	**0.97**	**0.85**	**0.91**	**0.97**
SMOTE(whole_data)	**LogisticRegression**	**0.02**	**0.96**	**0.93**	**0.02**
SMOTE(whole_data)	**SVC**	**0.07**	**0.79**	**0.88**	**0.07**
SMOTE(whole_data)	**DecisionTreeClassifier**	**0.41**	**0.80**	**0.90**	**0.41**
SMOTE(whole_data)	**RandomForestClassifier**	**0.85**	**0.85**	**0.92**	**0.85**
SMOTE(whole_data)	**BernoulliNB**	**0.04**	**0.90**	**0.93**	**0.04**
ADASYN(train_data)	**LogisticRegression**	**0.04**	**0.93**	**0.95**	**0.04**
ADASYN(train_data)	**SVC**	**0.10**	**0.90**	**0.95**	**0.10**
ADASYN(train_data)	**DecisionTreeClassifier**	**0.01**	**0.96**	**0.93**	**0.01**
ADASYN(train_data)	**RandomForestClassifier**	**0.05**	**0.93**	**0.95**	**0.05**
ADASYN(train_data)	**BernoulliNB**	**0.05**	**0.88**	**0.93**	**0.05**

ADASYN(whole_data)	**LogisticRegression**	**0.97**	**0.91**	**0.94**	**0.97**
ADASYN(whole_data)	**SVC**	**0.98**	**0.87**	**0.93**	**0.98**
ADASYN(whole_data)	**DecisionTreeClassifier**	**0.93**	**0.90**	**0.92**	**0.93**
ADASYN(whole_data)	**DecisionTreeClassifier**	**0.97**	**0.90**	**0.93**	**0.97**
ADASYN(whole_data)	**RandomForestClassifier**	**0.98**	**0.85**	**0.92**	**0.98**
ADASYN(whole_data)	**BernoulliNB**	**0.05**	**0.93**	**0.95**	**0.05**
UnderSampling(train_data)	**LogisticRegression**	**0.32**	**0.77**	**0.88**	**0.32**
UnderSampling(train_data)	**SVC**	**0.73**	**0.75**	**0.87**	**0.73**
UnderSampling(train_data)	**DecisionTreeClassifier**	**0.97**	**0.78**	**0.89**	**0.97**
UnderSampling(train_data)	**RandomForestClassifier**	**0.06**	**0.85**	**0.91**	**0.06**
UnderSampling(train_data)	**BernoulliNB**	**0.60**	**0.61**	**0.59**	**0.60**
UnderSampling(whole_data)	**LogisticRegression**	**1.00**	**1.00**	**1.00**	**1.00**
UnderSampling(whole_data)	**SVC**	**1.00**	**1.00**	**1.00**	**1.00**
UnderSampling(whole_data)	**DecisionTreeClassifier**	**1.00**	**1.00**	**1.00**	**1.00**
UnderSampling(whole_data)	**RandomForestClassifier**	**0.66**	**0.98**	**0.73**	**0.66**
UnderSampling(whole_data)	**BernoulliNB**	**0.00**	**0.81**	**0.66**	**0.00**
Class weightining(Algorithmic Method)	**LogisticRegression**	**0.00**	**0.00**	**0.50**	**0.00**
Class weightining(Algorithmic Method)	**SVC**	**0.01**	**0.03**	**0.51**	**0.01**
Class weightining(Algorithmic Method)	**DecisionTreeClassifier**	**0.00**	**0.00**	**0.50**	**0.00**
Class weightining(Algorithmic Method)	**RandomForestClassifier**	**0.00**	**0.02**	**0.26**	**0.00**
Class weightining(Algorithmic Method)	**BernoulliNB**	**0.87**	**0.62**	**0.81**	**0.87**
GAN(whole_data)	**LogisticRegression**	**0.94**	**0.64**	**0.82**	**0.94**
GAN(whole_data)	**SVC**	**0.69**	**0.80**	**0.90**	**0.69**
GAN(whole_data)	**DecisionTreeClassifier**	**0.95**	**0.80**	**0.90**	**0.95**
GAN(whole_data)	**RandomForestClassifier**	**0.06**	**0.85**	**0.91**	**0.06**
GAN(whole_data)	**BernoulliNB**	**0.87**	**0.62**	**0.81**	**0.87**
GAN(train_data)	**LogisticRegression**	**0.94**	**0.64**	**0.82**	**0.94**
GAN(train_data)	**SVC**	**0.73**	**0.76**	**0.88**	**0.73**
GAN(train_data)	**DecisionTreeClassifier**	**0.92**	**0.81**	**0.90**	**0.92**
GAN(train_data)	**RandomForestClassifier**	**0.06**	**0.85**	**0.91**	**0.06**
GAN(train_data)	**BernoulliNB**	**0.97**	**0.91**	**0.94**	**0.97**