REVA UNIVERSITY
Bengaluru, India

SCHOOL OF COMPUTER SCIENCE AND APPLICATIONS

Minor Project Progress Report – II

Auto Insurance Fraud Detection Using Classification Models and Oversampling Techniques

Master of Science in Data Science - M.Sc.(DS)

Submitted by
Disha Shet Dessai
(R22DG014)

Saurabh Sain
(R22DG049)

Under the guidance of

Internal Guide
Dr. Hemanth K. S
Professor

25th January 2024

Rukmini Knowledge Park, Kattigenahalli, Yelahanka, Bengaluru-560064
www.reva.edu.in

CERTIFICATE

This is to certify that the Minor project work entitled "**Auto Insurance Fraud Detection Using Classification Models and Oversampling Techniques** " submitted to the School of Computer Science and Applications, REVA University in partial fulfillment of the requirements for the award of the Degree of MSc in Data Science in the academic year 2023-2024 is a record of the original work done by **(Saurabh Sain (R22DG049)** and **Disha Shet Dessai(R22DG014))** under my supervision and guidance and that this Minor project work has not formed the basis for the award of any Degree / Diploma / Associate ship / Fellowship or similar title to any candidate of any University.

Place: Bengaluru, India

Date: 25/01/2024

Internal Guide Signature: Signature of the Program Co-Ordinator

Signature of the Director

Submitted for the University Examination held on __________________

Internal Examiner External Examiner

DECLARATION

I hereby declare that this Minor project work entitled "**Auto Insurance Fraud Detection Using Classification Models and Oversampling Techniques**" under the guidance of **Dr. Hemant K. S** and that this Minor project work has not formed the basis for the award of any Degree / Diploma / Associate ship / Fellowship or similar title to any candidate of any University.

Signature of the Candidate	Signature of the Candidate
Name: Saurabh Sain	Name: Disha Shet Dessai
SRN No: R22DG049	SRN No: R22DG014

Date: 25/01/2024

Countersigned by

Signature of the guide

TABLE OF CONTENTS

ABSTRACT

In the landscape of machine learning, the quest for heightened accuracy in classification models remains ongoing. This research delves into the enhancement of classification model performance by delving into the hyperparameter space of diverse resampling methods. Imbalanced datasets pose a significant challenge, and resampling techniques such as oversampling, undersampling, and their combinations emerge as pivotal tools for addressing this issue. The primary objective is to systematically optimize the hyperparameters linked with these resampling techniques, aiming to maximize the overall accuracy of the classification model. This study focuses specifically on the context of auto insurance fraud detection, where accurate classification is crucial for identifying fraudulent claims. The exploration of resampling methods and their hyperparameter fine-tuning serves as a valuable contribution towards achieving robust and effective fraud detection systems in the insurance domain.

INTRODUCTION

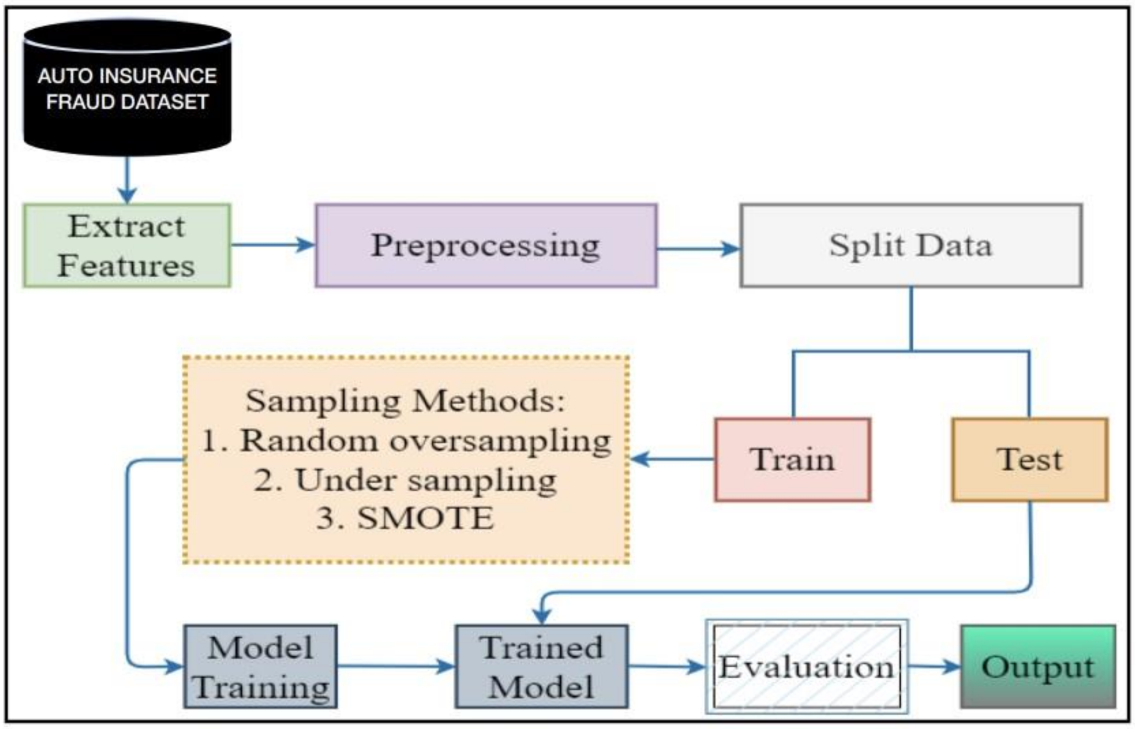
Auto insurance fraud represents a pervasive challenge with far reaching consequences for the insurance industry. This issue encompasses a variety of deceptive practices, including staged accidents, exaggerated claims, and involvement of organized crime. These fraudulent activities lead to substantial financial losses for insurance companies, necessitating increased premiums for policyholders and eroding trust within the insurance system.

The impact of auto insurance fraud extends beyond financial considerations. Legitimate policyholders face the consequences through increased premiums, reflecting the industry's efforts to offset losses incurred due to fraudulent activities. The erosion of trust within the insurance system is a consequential byproduct, as honest customers bear the collective burden of dishonest practices. Various oversampling techniques and algorithms have been applied, such as Random Over Sampling (ROS) and Synthetic Minority Oversampling Technique (SMOTE). However, these methods have disadvantages.

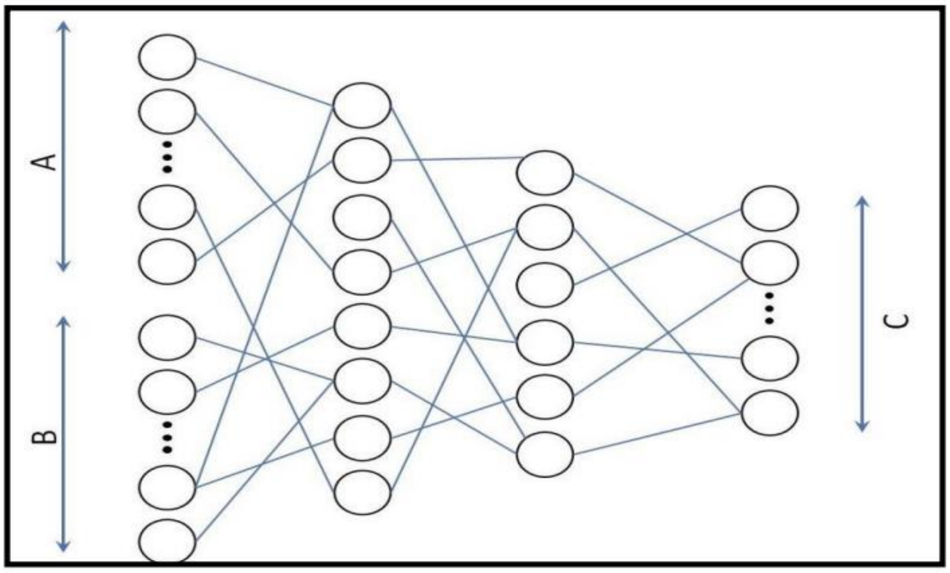
In recent years, there has been growing interest in applying deep learning methods, particularly when dealing with complex and nonlinear relations. Deep learning is suitable for big, multidimensional data and allows simultaneous classification, recognition, and prediction. Recently, generative adversarial networks (GANs) have been introduced as an alternative method for generating synthetic data on minority classes based on their complex distribution. The proposed method is evaluated by training 17 different classifiers, comparing the CTGAN, SMOTE, RUS, ROS, BSMO, ADASYN. By comparing these techniques, we are finding the best over sampling method for insurance detection and also we are hyper tuning the parameters of this techniques. As there is no such oversampling technique and its not even discovered for auto insurance this paper helps to find out the best technique.

ARCHITECTURE DIAGRAM

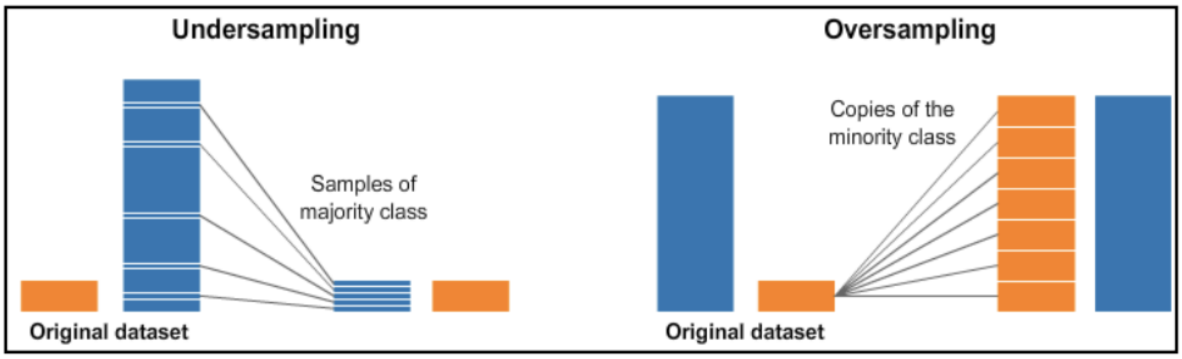
MODEL ARCHITECTURE



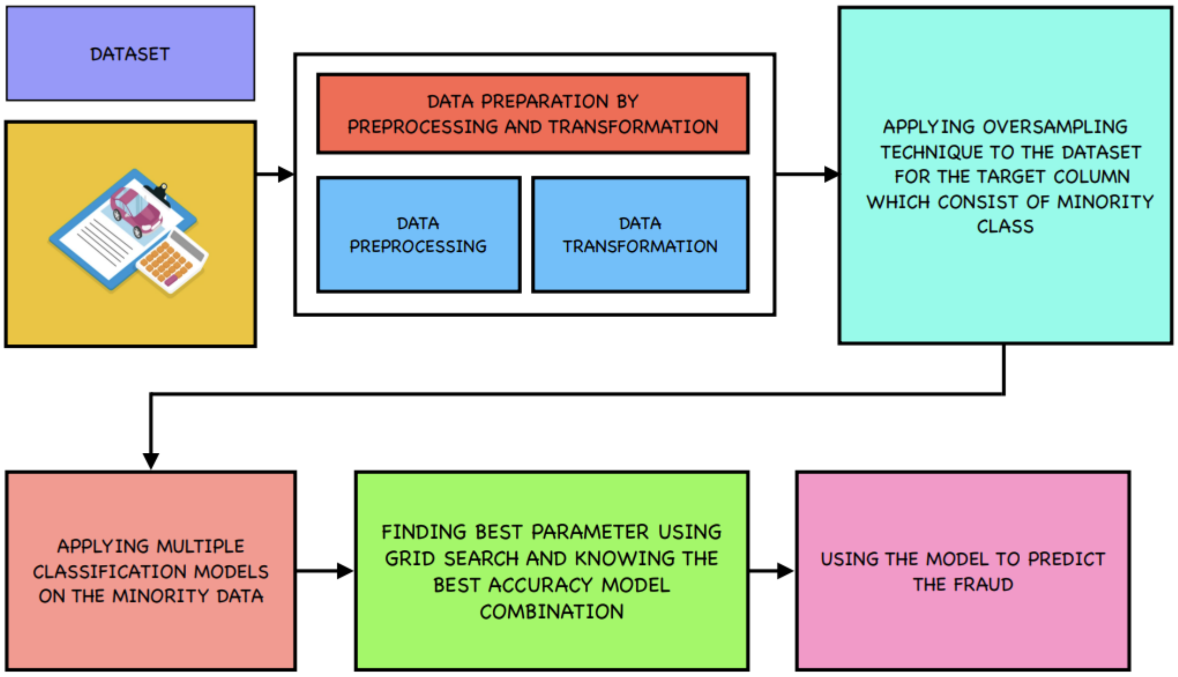
SMOTE ARCHITECTURE



RANDOM UNDER SAMPLING AND OVERSAMPLING TECHNIQUE



MODEL FLOW


DATASET
DATA PREPARATION BY PREPROCESSING AND TRANSFORMATION
DATA PREPROCESSING
DATA TRANSFORMATION
APPLYING OVERSAMPLING TECHNIQUE TO THE DATASET FOR THE TARGET COLUMN WHICH CONSIST OF MINORITY CLASS
APPLYING MULTIPLE CLASSIFICATION MODELS ON THE MINORITY DATA
FINDING BEST PARAMETER USING GRID SEARCH AND KNOWING THE BEST ACCURACY MODEL COMBINATION
USING THE MODEL TO PREDICT THE FRAUD

METHODOLOGY

A. About Dataset Description

In this research, we leverage the "carclaims" dataset obtained from "Angoss Knowledge Seeker", a publicly available automobile insurance dataset, to investigate fraud detection in the insurance domain. The dataset comprises 15,420 samples, each representing an insurance claim associated with automobile incidents. Notably, there is a significant class imbalance within the dataset, with 14,497 instances classified as legitimate claims and a relatively smaller subset of 923 instances characterized as fraudulent. This high class imbalance poses a considerable challenge for accurate and robust fraud detection, as the fraud class is underrepresented compared to legitimate claims.

The dataset encompasses various features, including temporal information such as the month, week, and day of the accident and claim, as well as categorical attributes like the vehicle make, accident area, and policy type. Continuous variables such as age, deductible, and driver rating provide additional insights into the characteristics of the claims. With a focus on the target variable "FraudFound," indicating the presence or absence of fraud in the claim, this dataset serves as a valuable resource for exploring the effectiveness of fraud detection methods, particularly in the context of imbalanced data, and evaluating the proposed methodologies using advanced techniques such as Conditional Tabular GAN (CTGAN) and others for oversampling to address the class imbalance issue.

B. Dataset Description

1. Month: The month in which the accident occurred.

2. WeekOfMonth: The week of the month in which the accident occurred.

3. DayOfWeek: The day of the week on which the accident occurred.

4. Make: The make or brand of the vehicle involved in the accident.

5. AccidentArea: The geographical area or location where the accident occurred.

6. DayOfWeekClaimed: The day of the week when the insurance claim was reported.

7. MonthClaimed: The month in which the insurance claim was reported.

8. WeekOfMonthClaimed: The week of the month in which the insurance claim was reported.

9. Sex: Gender of the policyholder or involved party.

10. MaritalStatus: Marital status of the policyholder.

11. Age: The age of the policyholder.

12. Fault: Indicates whether the policyholder was at fault in the accident.

13. PolicyType:Type of insurance policy held by the policyholder.

14. VehicleCategory: The category of the vehicle (e.g., sedan, SUV).

15. VehiclePrice: The price category of the vehicle.

16. PolicyNumber: Unique identifier for the insurance policy.

17. RepNumber: The representative or agent number associated with the policy.

18. Deductible: The amount of money that the policyholder is required to pay before the insurance coverage kicks in.

19. DriverRating: Rating associated with the driving history or skills of the policyholder.

20. Days: PolicyAccident : The number of days the policy has been active before the accident.

21. Days: PolicyClaim: The number of days the policy has been active before the insurance claim was filed.

22. PastNumberOfClaims: The number of previous insurance claims filed by the policyholder

23. AgeOfVehicle: The age of the vehicle involved in the accident.

24. AgeOfPolicyHolder: The age of the policyholder.

25. PoliceReportFiled: Indicates whether a police report was filed for the accident.

26. WitnessPresent: Indicates whether there were any witnesses present at the time of the accident.

27. AgentType: Type or category of the insurance agent.

28. NumberOfSuppliments: The number of supplementary insurance supplements associated with the policy.

29. AddressChangeClaim: Indicates whether there was a change of address when the claim was filed.

30. NumberOfCars: The number of cars covered by the insurance policy.

31. Year: The year in which the data was recorded.

32. BasePolicy: The base insurance policy type.

33. FraudFound: Binary indicator (yes/no) of whether fraud was found in the insurance claim. This is the target variable for fraud detection.

C. Preprocessing

Preparing the data for hyperparameter tuning of resampling techniques and classification models to attain increased accuracy is an essential first step. The data must be clean, normalized, and appropriate for training strong, reliable models in order for preprocessing to be effective.

When classification categories are not nearly evenly distributed, machine learning algorithms struggle to learn. Given the extreme imbalance in the data, some sort of balancing must be done in order for the model to be trained effectively. Undersampling the majority class, oversampling the minority class, or combining the two are common techniques for modifying the class distribution. When applied to imbalanced datasets, the popular oversampling technique known as Synthetic Minority Oversampling Technique (SMOTE) has shown to be effective. change the category and values for string into numerical. We have also changed the category values for string to numerical.

TECHNIQUES USED

A. Classification models

In machine learning, A classification model is a sort of machine learning model that is trained to predict the categorical class labels of instances based on their attributes. A classification algorithm's purpose is to learn a mapping between input features and class labels. This mapping enables the model to make predictions about previously unknown instances.

i. **Logistic Regression**: Logistic Regression is a binary classification approach that predicts the likelihood that an instance belongs to a specific class. It simulates the relationship between the independent factors and the likelihood of a specific result. It employs the logistic function (sigmoid function) to compress the output into a range of 0 to 1, expressing probabilities.

ii. **Random Forest**: Random Forest is an ensemble method for making more accurate and resilient forecasts by combining many decision trees. It grows a forest of trees and then combines their forecasts to provide a more stable and accurate conclusion. Each tree in the forest is trained on a different subset of the training data and votes on the final forecast.

iii. **SGD Classifier:** SGD is an optimization technique that is utilized in a variety of machine learning algorithms, including classifiers.It iteratively changes the model parameters, minimizing the cost function by considering one training sample at a time. It is especially helpful with huge datasets and online learning scenarios.

iv. **Passive Aggressive Classifier** is a form of online learning system that is developed for streaming data. It updates its model depending on new data instances and responds fast to changes. It's especially beneficial in instances where data distribution changes over time.

v. **K-Nearest Neighbors (KNN) Classifier:** KNN is a straightforward, instance-based learning method that may be used for classification and regression problems. It classifies a data point in the feature space based on the majority class of its k-nearest neighbors. The 'k' value chosen affects the smoothness of the decision border.

vi. **Decision Tree:** Decision Trees split the dataset recursively based on the most significant attribute at each node. Nodes represent feature conditions, and leaves represent class labels. They make decisions by moving from the root to the leaf of the tree.

vii. **Extra Trees Classifier**: Extra Trees is a Random Forest-like ensemble approach that constructs several decision trees with random splits. It adds unpredictability by utilizing different random thresholds for each feature at each split. This randomization frequently results in faster training and can increase generalization.

viii. **Gradient Boosting:** Gradient Boosting successively constructs a sequence of weak learners (usually decision trees). Each tree corrects the faults of the previous ones, resulting in a more accurate model. To generate the final prediction, it integrates the projections of all trees.

ix. **Gaussian Naive Bayes:** Naive Bayes is a probabilistic classifier that is based on the Bayes theorem and assumes feature independence. The Gaussian Naive Bayes version is based on the assumption that the features have a Gaussian distribution. It computes the likelihood of a sample belonging to a class based on the likelihood of its features belonging to that class.

x. **Linear Support Vector Machine (Linear SVM):** Linear SVM creates a hyperplane in the feature space that best divides classes. It seeks to maximize the margin between classes, which is defined as the distance between the hyperplane and the nearest data points from each class. It works well in three-dimensional spaces.

xi. **AdaBoost** combines several weak classifiers to build a strong classifier. It gives weights to misclassified cases, emphasizing difficult-to-classify samples. Weak learners are sequentially taught, and their predictions are weighted differently.

xii. **Bagging (Bootstrap Aggregating):** Bagging (Bootstrap Aggregating) is the process of training several instances of the same learning algorithm on different subsets of the training data. The final prediction is frequently an average or voting of individual model projections. A bagging algorithm is an example of Random Forest.

xiii. **Linear Discriminant Analysis (LDA):** LDA is a technique for classification and dimensionality reduction. It seeks to identify the linear combinations of features that best distinguish between various classes. It is presumptively assumed that the data is normally distributed and that the classes share a covariance matrix.

xiv. **Quadratic Discriminant Analysis (QDA):** QDA is similar to LDA, except it relaxes the assumption of all classes having a common covariance matrix. Each class can have its own covariance matrix. It may be more versatile than LDA, but it may necessitate more data.

B. Oversampling Techniques Used:

Oversampling is a machine learning approach used to address the problem of class imbalance in classification issues. When the number of instances in one class is much smaller than the number of instances in another class, class imbalance develops. A model may be biased towards the dominant class in such instances, resulting in poor performance on the minority class. Oversampling is one method for balancing the class distribution and improving the model's ability to classify cases from the minority class properly.

1. <u>SMOTE or Synthetic Minority Over-sampling Technique</u>

SMOTE, or Synthetic Minority Over-sampling Technique, is a frequently used method in machine learning and data mining. It is specifically intended to handle the problem of class imbalance in datasets. Class imbalance arises when one class in a classification issue has significantly fewer occurrences than the other(s). This can result in biased models that favor the dominant class and perform poorly when forecasting the minority class. SMOTE is an oversampling strategy that attempts to address this issue by creating synthetic samples from the minority class, thereby balancing the class distribution and increasing the performance of machine learning models.

The basic idea underlying SMOTE is to generate synthetic samples by interpolating existing minority class samples. The approach entails recognizing specific minority class instances and creating new synthetic samples that are similar but not identical to the current minority class examples. This is accomplished via a combination of feature space similarity and random picking of locations along the line segments that connect pairs of minority class examples.

The mathematical definition of SMOTE entails comprehending how new synthetic samples are generated based on feature space similarity to existing minority class cases. To keep things simple, assume a dataset with only two characteristics (X1 and X2).
Assume we have a minority class instance A = (A1, A2), and its nearest neighbor B = (B1, B2). To create a synthetic sample C = (C1, C2) between A and B, apply the following equation:

$$C = A + rand() * (B - A)$$

Where: A = (A1, A2) represents the coordinates of instance A. B = (B1, B2) represents the coordinates of its nearest neighbor B. rand() generates a random number between 0 and 1. C = (C1, C2) represents the coordinates of the new synthetic sample. This equation essentially computes a point along the line segment joining A and B based on a random factor generated by rand(). By varying this random factor, multiple synthetic samples can be generated within the feature space proximity of A and B.

SMOTE is an effective strategy for correcting class imbalances in machine learning datasets by creating synthetic samples for the minority class. Its mathematical formulation is interpolating existing instances to generate new synthetic samples within the feature space vicinity. While it provides significant benefits for enhancing model performance, careful application and parameter tuning are required to enhance its usefulness.

2. ROS or Random oversampling

Random oversampling is a technique used in data resampling approaches to cope with imbalanced datasets, particularly in classification difficulties. This strategy includes randomly duplicating the observations of the minority class to match the size of the majority class. The goal is to build a more balanced dataset, which can lead to more accurate and unbiased models when used with machine learning techniques.

The random oversampling technique is described by the following equation: Denote *Nminority* as the number of instances in the minority class. *Nmajority* as the number of instances in the majority class. *Oversampling_Ratio* as the desired ratio of the number of minority class instances to the majority class instances after oversampling. The number of instances to be added to the minority class (*Noversample)* can be calculated as:

$$Noversample=(Oversampling\_Ratio \times Nmajority)-Nminority$$

This formula calculates how many synthetic instances must be created to attain the desired balance. It is crucial to note that, while Random Oversampling helps to reduce class imbalance during training, it may also introduce some level of overfitting because the model detects redundant instances. Care should be made to assess the model's performance on a different, uneven validation or test set.

3. <u>RUS or Random Under-Sampling</u>

RUS resolves class imbalance by eliminating instances from the majority class at random. This strategy reduces the majority class's dominance, although it may result in knowledge loss if significant instances are deleted. RUS is an easy technique, but it may not adequately convey the complexities of the minority class.

Random undersampling works by selecting a random subset of the majority class, reducing the overall size of the majority class while preserving the randomness of the selection procedure. This ensures that the selected samples represent the full dataset. The decreased majority class dataset is joined with the minority class dataset to produce a more balanced dataset.

To calculate the number of samples to be eliminated from the majority class, we have used the following equation:

$$N\_majority\_samples = (N\_total\_samples - N\_minority\_samples)$$

N_majority_samples is the number of samples in the majority class; *N_total_samples* is the total number of samples in the dataset; and *N_minority_samples* is the number of samples in the minority class. Once we've calculated the number of samples to be eliminated from the majority class, we can select them at random without replacement. This means that each sample has an equal chance of being chosen, and once selected, the sample cannot be chosen again.

4. ADASYN or Adaptive Synthetic Sampling

ADASYN is an adaptive technique for creating synthetic minority class samples based on the distribution of the minority class in the vicinity of each minority class sample. It accomplishes this by taking a sample from the majority class and perturbing it to produce a new sample from the minority class. This perturbation is performed in such a way that the new sample is identical to the original minority class sample in terms of feature space. The ADASYN undersampling technique can be represented by the following equation:

$$X\_syn = X\_maj + (X\_min - X\_maj) * r$$

Where: X_syn is the synthetic minority class sample. X_maj is the chosen majority class sample. X_min is the minority class sample. r is a random value generated from a uniform distribution in the range [0, 1]. This equation shows how the synthetic minority class sample (X_syn) is created by adding a scaled difference between the minority and majority class samples (X_min, X_maj) to the majority class sample. The random value (r) ensures that the perturbation is adaptive and reflects the minority class's local distribution.

IMPLEMENTATION

	SMOTE	ADASYN	RUS	ROS
LR	0.81	0.81	0.73	0.75
SGD	0.50	0.76	0.49	0.72
PsvAgr	0.51	0.51	0.49	0.62
KNN	0.85	0.85	0.53	0.87
DT	0.91	0.92	0.66	0.96
ExTr	0.92	0.93	0.71	0.98
GB	0.85	0.86	0.75	0.79
GNB	0.70	0.75	0.74	0.68
LSVC	0.79	0.76	0.50	0.75
AdaBoost	0.84	0.84	0.74	0.76
Bagging	0.92	0.92	0.72	0.97
RF	0.93	0.93	0.76	0.98
LDA	0.80	0.81	0.77	0.75
QDA	0.56	0.76	0.48	0.50

Table (a) *Overall accuracy using oversampling techniques*

Table (a) displays classification performance metrics (most likely F1 scores) for several oversampling approaches paired with various classification algorithms. SMOTE, ADASYN, RUS (Random Under-Sampling), and ROS (Random Over-Sampling) are among the oversampling approaches. The classification algorithms include LR (Logistic Regression), SGD (Stochastic Gradient Descent), PsvAgr (Passive Aggressive), KNN (K-Nearest Neighbors), DT (Decision Tree), ExTr (Extra Trees), GB (Gradient Boosting), GNB (Gaussian Naive Bayes), LSVC (Linear Support Vector Classification), AdaBoost, Bagging, RF (Random Forest), LDA (Linear Discriminant Analysis), and QDA (Quadratic Discriminant Analysis).

The values in the table are F1 scores, which are a statistic for measuring classification performance that combines precision and recall. Here's a quick rundown of the F1 scores for each oversampling method and classification algorithm combination: Logistic Regression works well with the majority of oversampling approaches, particularly SMOTE and ADASYN. Stochastic Gradient Descent performs differently across oversampling approaches, with ADASYN and ROS scoring higher. Passive Aggressive performs similarly across oversampling approaches, with lower overall scores when compared to other algorithms. K-Nearest Neighbors performs well, particularly when combined with SMOTE, ADASYN, and RUS. Decision Tree performs well with high F1 scores across all oversampling strategies. Extra Trees regularly outperforms, with strong scores across all oversampling methods. Gradient Boosting performs well, especially with ROS. Gaussian Naive Bayes performs moderately among oversampling strategies. Linear Support Vector Classification (LSVC) works reasonably well, especially with SMOTE and ROS.

On the Auto Insurance Fraud dataset, the Random Forest algorithm and the Random Over-Sampling (ROS) technique outperformed each other due to complementary strengths. Random Forest, being an ensemble of decision trees, excels at catching non-linear patterns in data, which is critical in fraud detection because fraudulent activities might have complicated relationships. The ROS technique, which involves oversampling the minority class (fraudulent instances), tackles the problem of class imbalance in fraud detection datasets, resulting in a more balanced representation and preventing the model from being biased toward the majority class. The combination of Random Forest and ROS is particularly helpful in dealing with the obstacles provided by imbalanced data and identifying complicated patterns, resulting in higher fraud detection accuracy.

The higher performance achieved on the Auto Insurance Fraud dataset using the Random Forest and Random Oversampling (ROS) techniques can be attributed to their synergistic effects. Random Forest's ensemble of decision trees, trained on randomized subsets of data using feature randomization and bagging, excels in capturing complicated, non-linear relationships within the dataset. The ROS technique, by resolving the class imbalance inherent in fraud detection datasets, guarantees that cases of fraud, which are infrequent, are adequately represented during training. This dual strategy reduces overfitting, improves the model's applicability to potential cases of fraud, and ensures forecast stability. The complementary nature of Random Forest's ensemble learning and ROS's alleviation of class imbalance collectively contribute to the model's superior performance, demonstrating the efficacy of this combined strategy in enhancing fraud detection accuracy on the Auto Insurance Fraud dataset.

Similarly, the successful conclusion obtained with the Extra Trees (ExTr) algorithm in conjunction with the ROS approach can be attributed to similar causes. Extra Trees, similar to Random Forest, uses ensemble learning and randomization in feature selection, providing resilience against overfitting and the ability to capture non-linear correlations. When used with ROS to reduce class imbalance, the combo tackles the unique issues of fraud detection datasets. This combination improves fraud detection by leveraging Extra Trees' intrinsic capabilities and ROS's ability to mitigate uneven class distribution.

	PARAMETES		OVERALL_ACCURACY	MEAN
LR	C=0.001		0.94	0.938
SGD	loss=hinge	penalty=elasticnet	0.94	0.763
PsvAgr	C=0.001		0.06	0.763
KNN	n_neighbors= 19	weights=uniform	0.94	0.937
DT	max_depth=10		0.94	0.919
ExTr	max_depth=10	n_estimators=200	0.94	0.936
GB	learning_rate=0.1	n_estimators=200	0.94	0.937
GNB	NA		0.60	0.509
LSVC	C=1000		0.94	0.938
AdaBoost	n_estimators=10		0.93	0.934
Bagging	n_estimators=200		0.93	0.928
RF	max_depth=10	n_estimators=10	0.94	0.937
LDA	tol=0.0001		0.94	0.932
QDA	reg_param=0.5	tol=0.0001	0.92	0.801

Table (b) *Best parameters & accuracy without using oversampling techniques*

In machine learning, parameters are the internal variables that a model learns from training data. These parameters are changed throughout the learning process to improve the model's performance on a certain task. The model learns the values of these parameters based on the patterns and correlations in the training data, rather than setting them manually. Model parameters are the internal variables that the model acquires from training data. Examples include weights in a linear regression model, coefficients in a logistic regression model, and split points in a decision tree.

In Table (b) model parameters are specific to the algorithm used and are changed during the training process to reduce the discrepancy between projected and actual outputs. Hyperparameters are external configuration settings for the model. Unlike model parameters, are not learned from data but rather are predetermined prior to training. Learning rates, regularization strengths, tree depths, and an algorithm's iteration count are some examples. Hyperparameters are critical for influencing the behavior of the learning algorithm and should be carefully selected using approaches such as grid search or random search.

The optimal performance obtained in the Auto Insurance Fraud dataset using Logistic Regression (LR) with a regularization parameter (C) of 0.001 and Leave-One-Out Stratified Cross-Validation (LSCV) with C=1000 can be attributed to careful parameter tuning and the dataset's inherent characteristics. A low regularization value (C=0.001) in LR indicates a preference for a simpler model to avoid overfitting and accommodate possible noise in the data. On the other hand, using LSCV with a higher C value (C=1000) indicates a readiness to allow for a more complicated model capable of capturing intricate patterns within the data. This balanced strategy tackles the trade-off between model complexity and generalization, improving the model's capacity to detect fraudulent patterns in the auto insurance data set. The ideal parameters for fraud detection are carefully chosen based on the dataset's complexity and the need for a comprehensive model.

The table (b) is a summary of many machine learning models, including their hyperparameter setups and associated performance metrics. This table highlights the performance of different machine learning models on a given job. The "Overall Accuracy" shows each model's accuracy across the entire dataset, but the "Mean Accuracy" may refer to cross-validation or another evaluation method, depending on the context. The process of determining the best settings for hyperparameters is known as hyperparameter tuning or optimization. It frequently requires experimentation and validation on a separate validation dataset or using techniques such as cross-validation.

Random Forest, a popular ensemble learning technique for classification and regression applications, has achieved significant success because to its unique properties. The approach, which consists of decision trees trained independently on random subsets of the training data, uses bagging via bootstrap aggregation to increase tree diversity, resulting in better generalization to unseen data. Feature randomization during tree construction reduces inter-tree correlation, increasing resilience and minimizing overfitting.

Notably, Random Forest excels in handling nonlinear relationships, which are critical in fraud detection cases when fraudulent patterns depart from linear trends. Its capacity to resolve skewed data distributions often observed in fraud detection datasets, together with its variance reduction skills, demonstrates its potential for consistent and trustworthy predictions. The algorithm's inbuilt feature significance measure significantly enhances its utility. The algorithm's feature importance measure improves its efficacy in detecting vehicle insurance fraud by identifying major contributors to model predictions.

	OVERALL_ACCURACY	MEAN_ACCURACY
LR	0.94	0.938
SGD	0.90	0.761
PsvAgr	0.94	0.938
KNN	0.94	0.935
DT	0.90	0.887
ExTr	0.92	0.924
GB	0.94	0.934
GNB	0.60	0.509
LSVC	0.94	0.938
AdaBoost	0.93	0.933
Bagging	0.92	0.924
RF	0.93	0.933
LDA	0.94	0.932
QDA	0.93	0.684

Table (c) *Default parameters without using oversampling techniques accuracy*

The optimal performance achieved on the Auto Insurance Fraud dataset using Logistic Regression (LR), Support Vector Machines with Polynomial Kernel (PsvAgr), and Linear Support Vector Classification (LSVC) with default parameter configurations indicates that these algorithms are well-suited to the dataset's specific characteristics. Logistic Regression is well-known for its ease of use and interpretability, particularly in instances involving linear connections. Support Vector Machines, particularly those with a polynomial kernel, excel at detecting complex non-linear patterns. Linear Support Vector Classification, a linear model variant of SVM, performs well when the underlying relationships in the data are linear. The default parameter selections may be consistent with the dataset's underlying structure, contributing to the reported optimal results. The effectiveness of these models demonstrates their compatibility with the Auto Insurance Fraud dataset. They accurately detect and generalize trends without considerable parameter modification.

This table (c) compares classification performance parameters such as accuracy, overall accuracy, and mean accuracy for several classification algorithms that do not include any oversampling approaches. The classification algorithms include LR (Logistic Regression), SGD (Stochastic Gradient Descent), PsvAgr (Passive Aggressive), KNN (K-Nearest Neighbors), DT (Decision Tree), ExTr (Extra Trees), GB (Gradient Boosting), GNB (Gaussian Naive Bayes), LSVC (Linear Support Vector Classification), AdaBoost, Bagging, RF (Random Forest), LDA (Linear Discriminant Analysis), and QDA (Quadratic Discriminant Analysis).

Logistic Regression (LR), Passive Aggressive (PsvAgr), K-Nearest Neighbors (KNN), Decision Tree (DT), Extra Trees (ExTr), Gradient Boosting (GB), Linear Support Vector Classification (LSVC), AdaBoost, Bagging, Random Forest (RF), and Linear Discriminant Analysis (LDA) all have high accuracy, indicating good overall performance. Stochastic

Gradient Descent (SGD) has significantly poorer accuracy than other techniques. Gaussian Naive Bayes (GNB) is relatively less accurate. Most algorithms, including LR, PsvAgr, KNN, DT, ExTr, GB, LSVC, AdaBoost, Bagging, RF, and LDA, achieve excellent overall accuracy, indicating effective classification across classes. SGD and GNB achieve slightly lower overall accuracy. LR, PsvAgr, KNN, DT, ExTr, GB, LSVC, AdaBoost, Bagging, RF, and LDA all achieve high mean accuracy, showing strong performance across classes. SGD and GNB have significantly lower mean accuracy. QDA reveals a considerable decline in mean accuracy, indicating difficulties in correctly identifying occurrences, particularly for certain classes.

In conclusion, without oversampling strategies, classification algorithms often perform well, with high accuracy and overall accuracy across classes. The algorithm chosen may be determined by specific needs, such as computing efficiency, interpretability, and class performance.

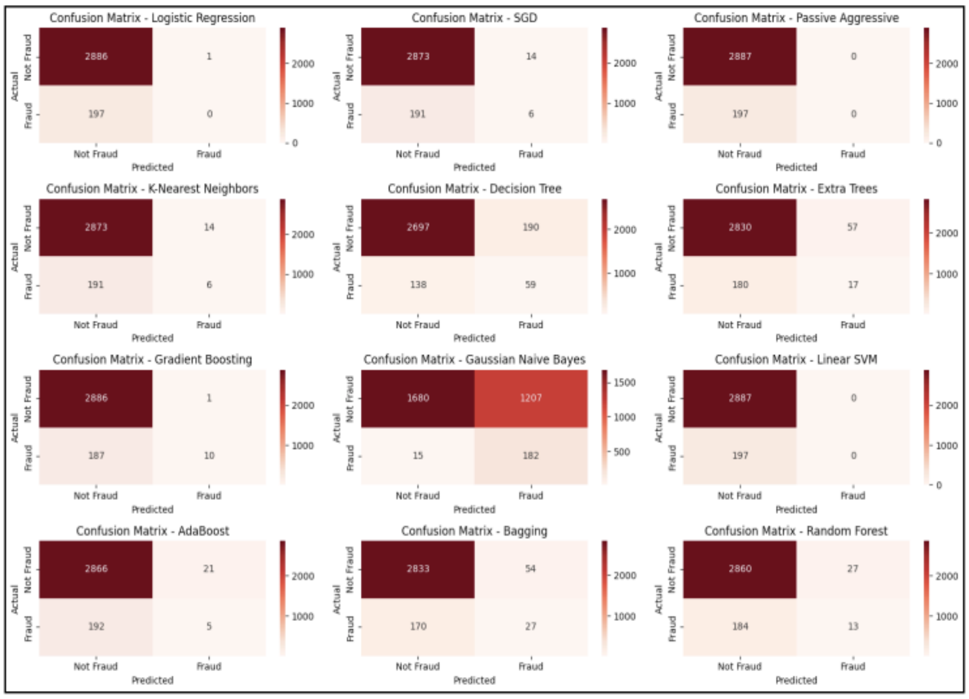


Fig 3. *Confusion Matrix for the models implemented*

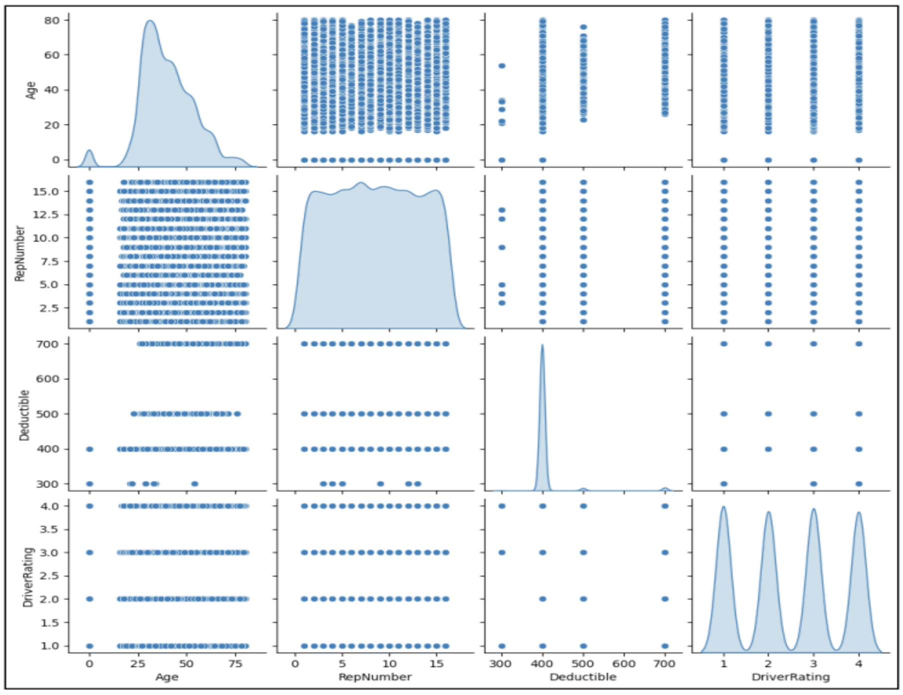


Fig 1. *Pairwise relationships between variables in a dataset*

The pairwise relationships between variables in a dataset through a grid of scatter plots in Fig 1 is Each variable is plotted against every other variable in the DataFrame. The diag_kind='kde' argument specifies that the diagonal plots should show a kernel density estimate (KDE) of the distribution of each variable instead of the default histogram. KDE is a non-parametric way to estimate the probability density function of a random variable.

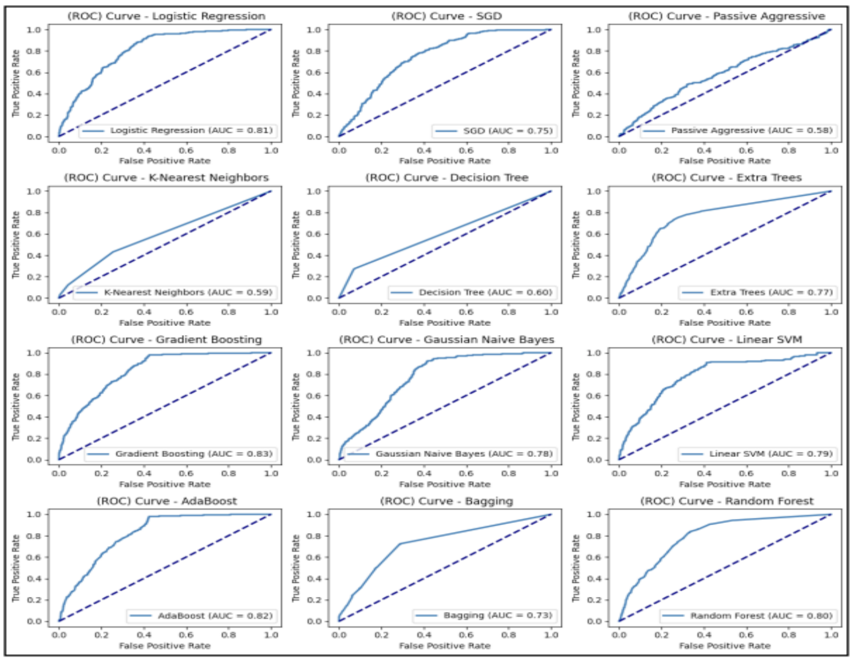


Fig 2. *Roc curves for each model implemented*

In this study Fig 2 shows that, we employed the Receiver Operating Characteristic (ROC) curve as the evaluation metric for our machine learning models. ROC curve analysis provided us with valuable insights into the model performance across various algorithms. Our findings highlight the effectiveness of certain models in accurately predicting the target variable. Additionally, we demonstrated the power of data preprocessing and algorithm selection in improving the model's overall performance.

Our results in Fig 2 indicate that the Gradient Boosting model achieved the highest AUC-ROC value of 0.83, demonstrating its superiority in distinguishing between positive and negative classes. The Gradient Boosting model's sequential learning approach, ability to handle complex data, flexibility, and regularization techniques contribute to its outstanding performance. This finding highlights the importance of employing advanced ensemble learning methods, such as Gradient Boosting, to improve model performance and achieve accurate predictions. Additionally, it is important to note that the high AUC-ROC value of 0.83 is not a standalone result but should be interpreted in the context of the problem and the other models' performance. Comparing the Gradient Boosting model's performance to other models, such as logistic regression or decision trees, can provide a more comprehensive understanding of its effectiveness.

SCOPE FOR FUTURE ENHANCEMENT

1. **Further Exploration of Oversampling Techniques:** The study focused on a subset of oversampling techniques, including SMOTE, ADASYN, RUS, and ROS. Future research could explore additional oversampling methods or combinations of techniques to assess their impact on classification model performance.

2. **Similar Techniques used for other datasets:** In this project we have used "carclaims" dataset, similarly this project can be taken forward for other fraud detection datasets.

3. **Advanced Deep Learning Approaches:** While the study touched upon the use of GANs, particularly CTGAN, for generating synthetic data, there is potential for further exploration of advanced deep learning methods. This could involve leveraging more sophisticated GAN architectures or exploring other deep learning techniques to handle complex and non-linear relationships in fraud detection.

4. **Ensemble Methods:** The research primarily focused on individual classification algorithms. Future studies could investigate the effectiveness of ensemble methods that combine multiple algorithms to enhance overall model performance. Ensemble methods, such as stacking or boosting, could provide synergies in fraud detection.

5. **Real-world Implementation and Validation:** The research utilized a specific dataset for experimentation. Future work could involve the implementation and validation of the proposed techniques in real-world scenarios within the insurance industry. This could include collaboration with insurance companies to assess the practical applicability and impact of the proposed methods.

6. **Explainability and Interpretability:** While achieving high accuracy is crucial, the interpretability of the models is also important, especially in industries like insurance where decisions have significant consequences. Future research could delve into methods for making machine learning models more interpretable without compromising on accuracy.

7. **Dynamic Adaptation:** Auto insurance fraud patterns may evolve over time. Future research could explore dynamic adaptation techniques that allow the model to continuously learn and update its understanding of fraud patterns based on changing data distributions.

8. **Integration of External Data Sources:** Incorporating additional external data sources, such as public records or social media data, could enhance the model's ability to detect fraudulent activities. Future research could explore the integration of diverse data sources to improve fraud detection accuracy.

9. **Ethical Considerations and Bias Mitigation:** As with any machine learning application, addressing ethical considerations and potential biases is crucial. Future research could focus on developing methods to mitigate biases in fraud detection models and ensure fair treatment of all policyholders.

10. **Scalability:** The scalability of the proposed techniques should be assessed for larger datasets and in scenarios with increased computational demands. Research in this area could explore efficient algorithms and parallel computing techniques.

11. **Benchmarking and Standardization:** Establishing benchmarks and standardizing evaluation metrics for fraud detection models could facilitate comparisons across different studies. Future research could contribute to the development of standardized practices in the evaluation of machine learning models for insurance fraud detection.

CONCLUSION

This comprehensive project shows the optimization of classification model performance by meticulously examining resampling approaches and hyperparameter tuning. The study looks at a variety of oversampling methods, including SMOTE, ADASYN, RUS, and ROS, as well as classification algorithms like Logistic Regression, Stochastic Gradient Descent, K-Nearest Neighbors, Decision Tree, Extra Trees, Gradient Boosting, Gaussian Naive Bayes, Linear Support Vector Classification, AdaBoost, Bagging, Random Forest, Linear Discriminant Analysis, and Quadratic Discriminant Analysis.

Furthermore, using the Auto Insurance Fraud dataset, the study emphasizes the synergy between Random Forest and Random Over-Sampling (ROS), underlining their combined strengths in dealing with imbalanced data and identifying detailed, non-linear patterns. The success is attributable to Random Forest's ensemble learning capabilities and ROS's capacity to minimize class imbalance, which results in higher fraud detection accuracy.

In conclusion, the study emphasizes the need of using proper oversampling approaches and hyperparameter tuning to improve classification model accuracy. These findings shed light on the complex interactions between resampling approaches, classification algorithms, and parameter configurations, providing a thorough guide for practitioners looking to optimize machine learning models in the context of unbalanced datasets.