Comparative Analysis of Different Resampling Techniques in Employee Attrition Prediction

Himanshi Choudhary

Indira Gandhi Delhi Technical University for Women

New Delhi, India

Email- himanshichoudhary565@gmail.com

Mehak Arora

Indira Gandhi Delhi Technical University for Women

New Delhi, India

Email- mehakarora2112@gmail.com

1. ABSTRACT

Machine learning is extensively used across different domains to make data-driven decisions; one such domain is HR Analytics. Employee attrition refers to the departure of employees for both voluntary and involuntary reasons. Employee attrition becomes a significant concern for organizations because considerable costs and risks are involved in recruiting new employees and it is efficient to retain the trained, experienced employees. Predicting the employees who are prone to attrition allows the company to take proactive measures to improve retention. This paper attempts to compare various resampling techniques used to deal with imbalanced dataset and applies them with various supervised learning models to conclude the best predicting model for attrition. The project involved the use of IBM HR Analytics dataset on which feature reduction techniques like PCA and chi square test were applied before the training of models on the imbalanced dataset. The said models include: Logistic Regression, SVM (Support Vector Machine), KNN (K-Nearest Neighbors), Decision Trees, Random Forest Classifier and XG Boost. In the next step, resampling techniques: Random Oversampling, SMOTE, ADASYN, Under-sampling, Tomek-Links, ENN, Hybrids, were used to balance the dataset and the models were trained again to make comparisons. The performance of classifiers improved with the use of a balanced dataset. Results obtained were compared on the basis of performance metrics, accuracy and AUC-ROC score which show that the Random Forest model performed best with an accuracy of 98.18% and AUC-ROC score of 99.99% when cross validation was performed after using Random Oversampling for balancing.

Keywords- Machine Learning, Data Balancing, Data Resampling, HR Analytics, Employee Attrition, Under Sampling, Over Sampling, Hybrid balancing techniques, comparative analysis.

2. INTRODUCTION

It is really crucial for an organization to have an idea of employees who are likely to depart from the organization in the near future, since more attrition significantly affects the overall working and management of the company. It costs an employer an average of 33% of an employee's yearly salary for their exit [9]. Retention rates of 90% and above are considered good however, according to a study one-third of new employees leave after 6 months [9]. High attrition rates in certain cases may bring down the overall reputation of the organization and loss of productive employees results in inefficient functioning due to continuous changes in workforce. Ongoing staff turnover can disrupt company operations. So, it is really important for them to figure out which employees may certainly leave the company so that appropriate measures can be taken beforehand to stop employees who are a great asset from departing.

Accurate attrition prediction can help organizations identify which employees are more likely to leave and which are likely to stay. Data imbalance is a significant issue in our dataset, as in real-world scenarios, employees are more likely to stay in the organization than to leave, as suggested by the dataset. The class indicating employees staying (Class 0) is much more prevalent than the class indicating employees leaving. Consequently, the different machine learning models we used performed exceptionally well for the majority class (employees staying), with a recall above 0.9, whereas the recall for the minority class (employees leaving) was very low, highlighting a serious class imbalance.

Various studies have been done to predict the attrition with different models and balancing techniques. A research study has adopted SMOTE, ADASYN along with XGBoost and hyperparameter tuning [3]. Other work included random oversampling, random undersampling and SMOTE for balancing the dataset with addition to the classification algorithms, KNN, logistic regression, decision tree, random forest and AdaBoost for analysis of their performance metrics [2]. Another research used SMOTE and conducted a comparative study of predicting employee attrition by using the four advanced machine learning techniques ETC (Extra Trees Classifier), SVM, LR, and DTC along with hyperparameter tuning where the proposed optimized ETC approach achieved a decent accuracy of 93%, [8]. A recent study [4], implemented SMOTE and hyperparameter tuning in the XGBoost model and achieved an accuracy of 0.85.

In summary, due to the evident problem of the imbalanced dataset, a need for comprehensive comparison among the popular resampling methods and models as a way to handle this problem efficiently,

arises. Therefore, our research problem is to analyze different resampling techniques while using various machine learning classifiers to provide a comparative view to fill the gaps in the literature.

We used statistical techniques like Chi Square test to identify features which are significant in attrition prediction, Principal Component Analysis (PCA) was also used in dimensionality reduction which significantly improved the complexity of features and furthermore, hyperparameter tuning using Randomized Search CV for each model was done to improve model performance. Different evaluation metrics for each model and resampling techniques were used for effective identification of the best performing model. For undersampling, we used Random Undersampling, Tomek Links and Edited Nearest Neighbors (ENN). Undersampling techniques remove data instances belonging to the majority class. Undersampling is also referred to as "down-sizing" in literature. Oversampling is done using Random Oversampling, Synthetic Minority Oversampling (SMOTE) and Adaptive Synthetic Sampling (ADASYN). The purpose of oversampling is to raise the number of cases of the minority class to equal the number of occurrences of the actual majority class. Oversampling is "upsizing" the minority class. At last hybrid methods - (SMOTE-TOMEK) and (SMOTE-ENN) were also tested. Both oversampling and undersampling techniques are combined in hybrid approaches. The balance between eliminating instances of the majority class and producing instances of the minority class is thus achieved through the use of hybrid approaches. These methods will be implemented on the IBM HR Analytics Datasets and will be compared for best performance. We performed EDA on both resampled dataset and the original dataset after each resampling technique so as to get an overview of how different features are getting impacted on resampling, revealing how the distribution of each feature changes due to resampling. The oversampling techniques and hybrid sampling performed better than undersampling. This approach will enable organizations to better identify employees who are likely to leave, rather than only those who are likely to stay with utmost accuracy.

3. METHODOLOGY

3.1. Dataset Description & Analysis

The dataset used in this work is IBM HR Employee Attrition [1]. The dataset contains 35 features and 1470 data entries. Out of these 35 features, 9 features are categorical features and remaining 26 are numerical features.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Age                       1470 non-null   int64
 1   Attrition                 1470 non-null   object
 2   BusinessTravel            1470 non-null   object
 3   DailyRate                 1470 non-null   int64
 4   Department                1470 non-null   object
 5   DistanceFromHome          1470 non-null   int64
 6   Education                 1470 non-null   int64
 7   EducationField            1470 non-null   object
 8   EmployeeCount             1470 non-null   int64
 9   EmployeeNumber            1470 non-null   int64
 10  EnvironmentSatisfaction   1470 non-null   int64
 11  Gender                    1470 non-null   object
 12  HourlyRate                1470 non-null   int64
 13  JobInvolvement            1470 non-null   int64
 14  JobLevel                  1470 non-null   int64
 15  JobRole                   1470 non-null   object
 16  JobSatisfaction           1470 non-null   int64
 17  MaritalStatus             1470 non-null   object
 18  MonthlyIncome             1470 non-null   int64
 19  MonthlyRate               1470 non-null   int64
 20  NumCompaniesWorked        1470 non-null   int64
 21  Over18                    1470 non-null   object
 22  OverTime                  1470 non-null   object
 23  PercentSalaryHike         1470 non-null   int64
 24  PerformanceRating         1470 non-null   int64
 25  RelationshipSatisfaction  1470 non-null   int64
 26  StandardHours             1470 non-null   int64
 27  StockOptionLevel          1470 non-null   int64
 28  TotalWorkingYears         1470 non-null   int64
 29  TrainingTimesLastYear     1470 non-null   int64
 30  WorkLifeBalance           1470 non-null   int64
 31  YearsAtCompany            1470 non-null   int64
 32  YearsInCurrentRole        1470 non-null   int64
 33  YearsSinceLastPromotion   1470 non-null   int64
 34  YearsWithCurrManager      1470 non-null   int64
dtypes: int64(26), object(9)
memory usage: 402.1+ KB
```

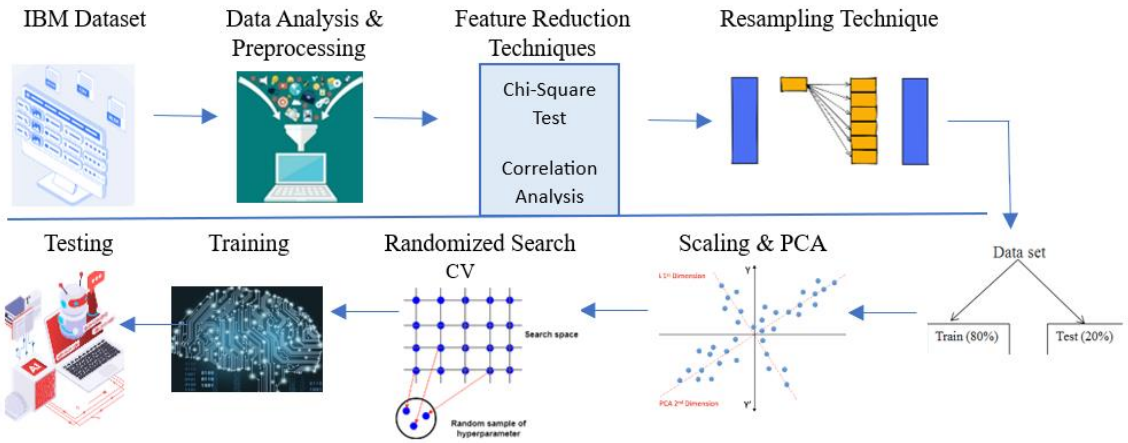

Figure 1. Methodology followed in our project.

The dataset was analyzed and features with only one unique value[‘employeeCount', 'Over18', 'StandardHours'] were removed. Furthermore, [‘Marital Status’, ‘EmployeeNumber’] features were dropped.

Fig. 2 shows the distribution of classes, ‘Yes’ and ‘No’ in the target variable, ‘Attrition’. It clearly depicts that the instances of negatives make up the majority of the attrition data.

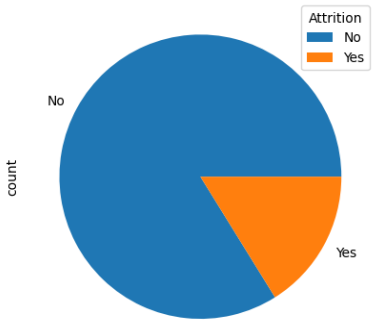

Figure 2. Visualizing the target variable

3.2. Data Preprocessing and Feature Engineering

I. Chi-square statistical test was used to study the relevance of categorical features in predicting the target variable. Smaller p-values indicate stronger evidence against the null hypothesis that the feature has no effect. Hence, a value of 0.05 was chosen as the standard measure. The feature, [‘Gender’], was dropped in accordance with the test.

II. Correlation analysis was done for feature reduction. Features that showed low correlation with attrition were removed which included, [‘PerformanceRating’, ‘HourlyRate’, ‘PercentSalaryHike’]. Heatmap was drawn for studying the correlation between the numerical features. [‘JobLevel’, ‘MonthlyIncome’] showed high correlation with each other (0.95). Keeping both may have not provided additional value and can lead to overfitting as a result of which, ‘JobLevel’ was removed. Fig. 3 depicts the heatmap.

III. The categorical features were Label encoded.

IV. Standard scaling was applied to the dataset. Principal component analysis (PCA) technique was implemented as a part of feature engineering for dimensionality reduction. However, this step followed after every resampling technique because after resampling, the dataset gains a different distribution and applying PCA after this stage ensures the principal components reflect the true variance in the resampled data.

3.3. Resampling Methods

Before applying the resampling techniques, the dataset was preprocessed as described previously. After generating resampled data using several techniques, we compared performance of each model on raw data and the resampled data. Then we applied PCA on the resampled data so as to reduce dimensionality of features.

3.3.1 Under Sampling

The aim is to eliminate majority class instances in the data for balancing.

I. **Random Under Sampling-**
Random Under Sampling [2] is a simple method. In this technique, the instances of majority class (0-no attrition took place) are chosen at random and eliminated till the count of instances of minority class (1-attrition happened) matches with that of majority class.
We performed this by importing “Random Under Sampler” from “imblearn.under_sampling module” of “imbalanced-learn” library in python.

II. **Tomek Links**
This involves measuring distances. It finds desired samples of data from the majority class that is having the lowest Euclidean distance with the minority class data and then removes it.
We performed this by importing “TomekLinks” from “imblearn.under_sampling” module of “imbalanced-learn” library.

III. **ENN**
ENN incorporated KNN (K-nearest Neighbors) in order to eliminate majority class instances. ENN determines the nearest neighbors of each instance in the majority class within a defined radius. The radius is determined by the number of neighbors, k. The present instance's class label is compared to those of its nearest neighbors. If a significant fraction of the neighbors are from a different class (the minority class), it means that the current instance is likely to be an outlier. Thus, the instance is removed from the dataset.
It is implemented by using “EditedNearestNeighbours” class from “imblearn.under_sampling” module of “imbalanced-learn” library.

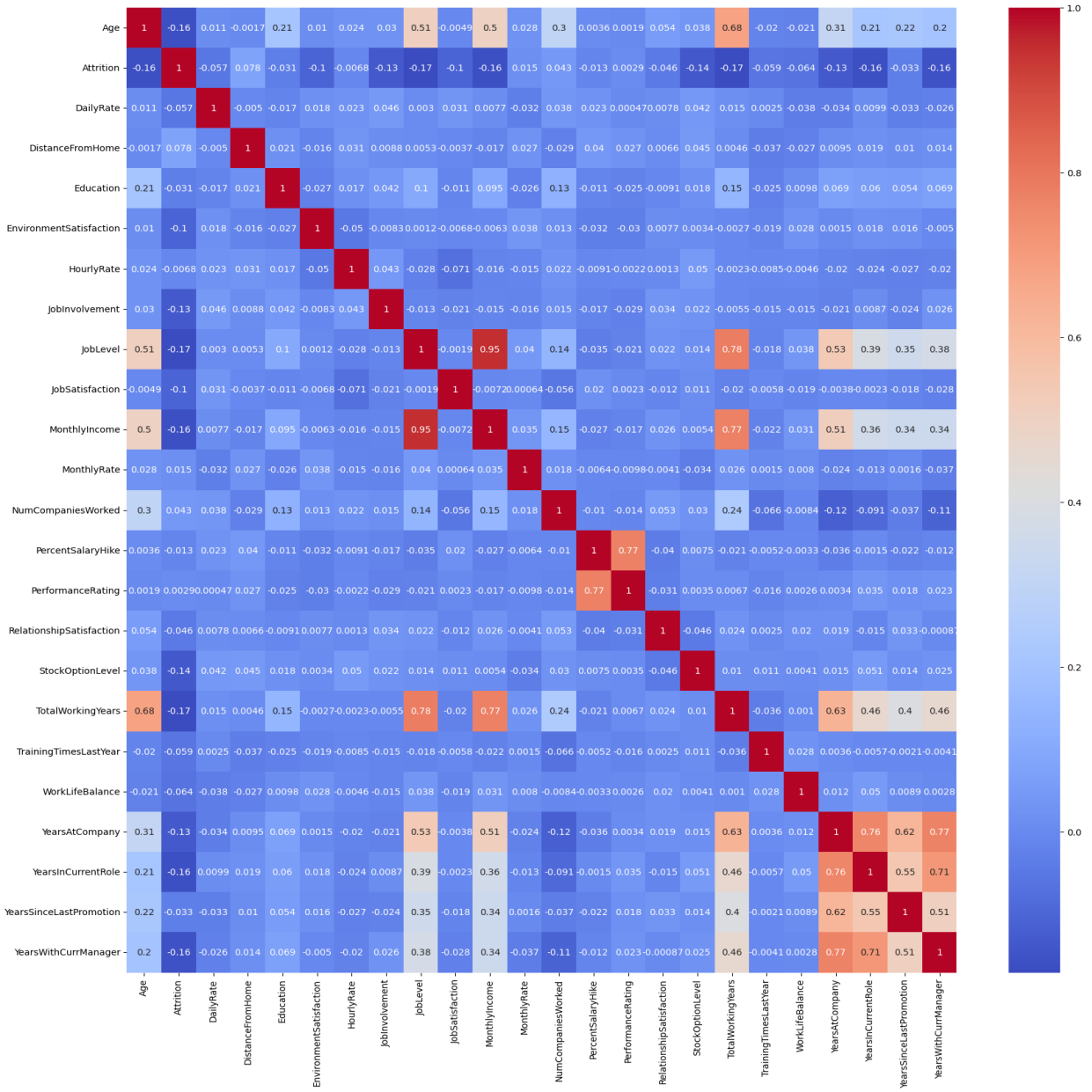

Figure 3. Heatmap to study correlations between numerical feature

3.3.2 Over Sampling

It aims to generate instances of the minority class to balance the class distribution.

I. Random Oversampling

Random Oversampling [2] is a simple method in which the instances of minority class (1-attrition took place) are chosen at random for replication till the count of instances of minority class matches with that of majority class (0-no attrition happened).
We performed this by importing "RandomOverSampler" from "imblearn.over_sampling" module of "imbalanced-learn" library. This performed the best. Fig. 4 is the demonstration of the effect of resampling on ["Age",'MonthlyIncome','YearsAtCompany','DistanceFromHome] features in our dataset.

II. SOMTE

It is an over-sampling approach in which the minority class is over-sampled by creating "synthetic" examples rather than by over-sampling with replacement. [10]
Interpolation between multiple minority class instances that lie together is the basis for creating new instances in SMOTE [2].
Implemented by importing "SMOTE" class from "imblearn.over_sampling"

III. ADASYN

It generates synthetic examples by focusing on misclassified instances and hard-to-classify data points. This process improves the quality of the predictions by increasing the amount of data for minority classes while taking into account the specific difficulties encountered by the model.
Implemented by using the "ADASYN" class from "imblearn.over_sampling"

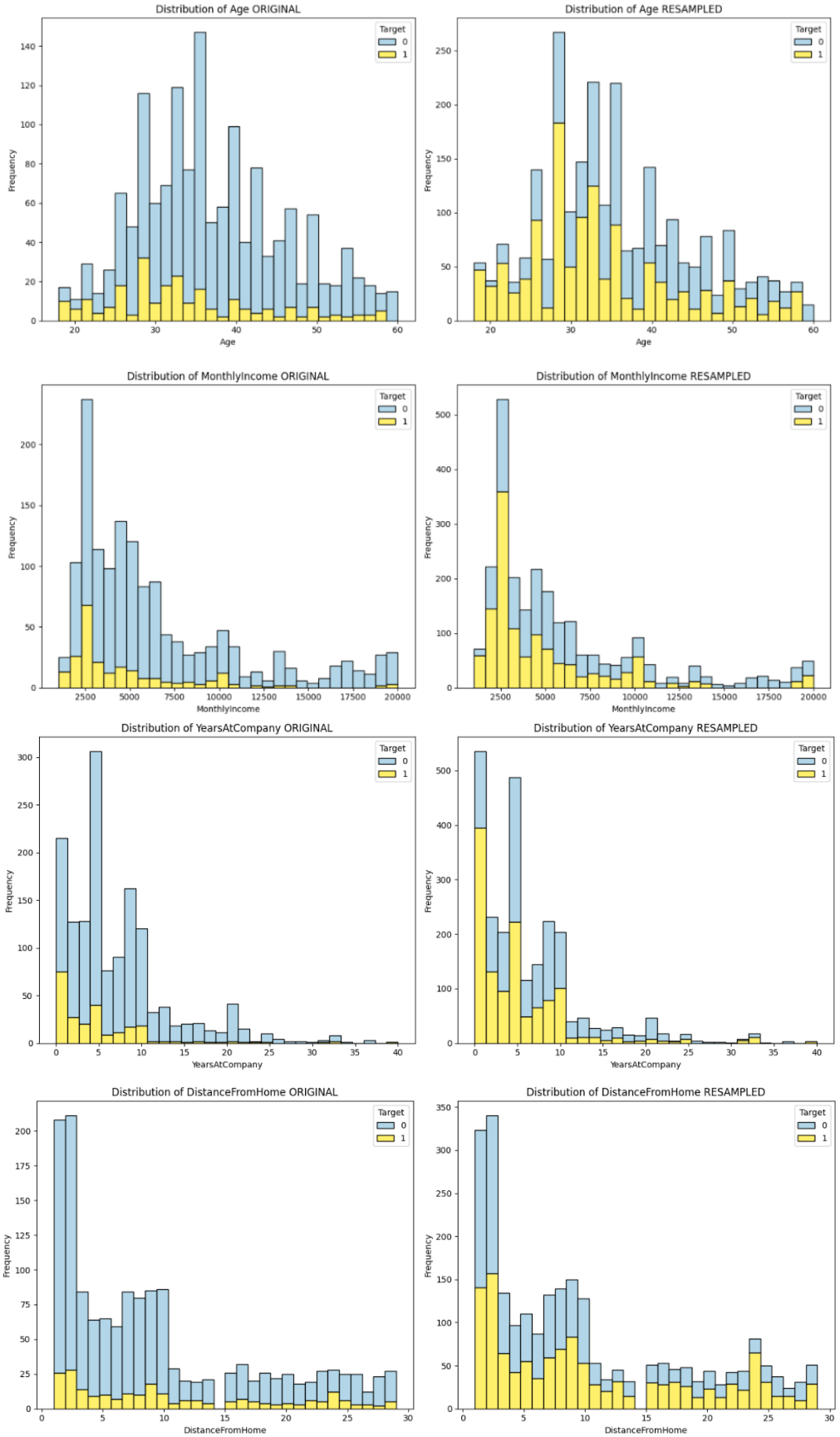

Figure 4. Features before & after Random Oversampling

3.3.3 Hybrid Methods

They are a combination of undersampling and oversampling methods. They assist in striking a balance between the creation of minority class instances and the removal of majority class instances.

I. **SMOTE-TOMEK LINKS**
The SMOTE-Tomek Links technique refines the dataset by eliminating unclear cases using Tomek Links after using SMOTE to create synthetic samples. This combination produces a cleaner, more balanced dataset, which enhances the performance of classification systems.
Implemented by using the "SMOTETomek" class from "imblearn.combine"

II. **SMOTE-ENN**
In order to balance the class distribution, SMOTE first creates synthetic samples in SMOTE-ENN. The dataset is then cleaned up using ENN by eliminating noisy or ambiguous occurrences, which enhances the classes' quality and separability. By combining these two elements, the dataset should become more balanced and noise-free, which will improve the performance of machine learning models.
Implemented by using the "SMOTEENN" class from "imblearn.combine"

3.4 Experimental Setup

The models we used for comparative analysis of each resampling techniques to find the best performing model are Logistic Regression [5], Random Forest Classifier [2], KNN, XGB Classifier [3], SVC, Decision Tree Classifier.

I. **Logistic Regression**
The likelihood of a binary result on one or more predictor variables (features) is modelled using this statistical technique in machine learning. When there are only two potential values for the dependent variable, which are typically represented by the numbers 0 and 1, it is utilized. In our data 0 represents attrition did not happen while 1 represents attrition did happen.
Logistic Function (Sigmoid Function)
The core of logistic regression is the logistic function,
which maps any real-valued number into the (0, 1) interval. The function is:

$$S(z) = \frac{1}{1 + e^{-z}}$$

Here, z is the combination of input features.

II. **Decision Tree Classifier**
A Decision Tree Classifier is a model that makes predictions by recursively splitting data based on feature values to form a tree structure, where each path from the root to a leaf represents a classification rule.
It generates a structure resembling a tree by **recursively dividing** the data according to feature values. The full dataset is represented by the top node of the tree, also known as the **root node**. Subsets of the dataset are then created according to the values of individual features. Based on one of the features, each internal node of the tree represents a **decision rule**; the branches show the results of these decisions. This procedure keeps going until a stopping criterion—like a **maximum tree depth** or a **minimum number of samples per leaf**—is met or the data in the **leaf nodes are pure**.

III. **Random Forest Classifier**
For classification tasks, Random Forest is a flexible and potent ensemble learning technique. It creates a more reliable and accurate classification model by combining the results of several decision trees.
Ensemble Learning: Random Forest is an ensemble approach, meaning it makes predictions by combining several models (decision trees). The theory behind this is that performance can be enhanced by combining predictions from many models.
Decision Trees: A Random Forest's fundamental construction elements are decision trees. Based on feature values, each tree is trained to divide data into classes. While a single decision tree may be overfitting, a forest—a group of trees—generalizes more effectively.
The Bootstrap Aggregating Bagging method: Random Forest employs bootstrapping for data sampling, whereby every tree is trained on a random portion of the training dataset (with replacement). As a result, the trees now exhibit diversity.

Randomness of Features: Just a random subset of features is taken into account at each split during the tree-building process, significantly diversifying the trees. The **final prediction** is determined by majority voting. Each tree votes for a class, and the class with the most votes is chosen.

IV. **K Nearest Neighbor Classifier**
A data point is assigned a class by the K-Nearest Neighbors (KNN) classifier based on the most prevalent class among its **'k' closest** neighbors. It computes **the distances between each training point and the query point**, finds the 'k' closest ones, and then chooses the most **common class label** among these neighbors in order to classify. KNN is easy to use and doesn't need a training phase, but it has trouble processing high-dimensional data and can be slow with big datasets.

V. **XG Boost Classifier**
A potent machine learning model that use boosting to increase prediction accuracy is the XGBoost (Extreme Gradient Boosting) classifier. It **sequentially** constructs an ensemble of decision trees, with each tree **fixing mistakes from the preceding** ones. XGBoost is able to extract intricate patterns and relationships from the data by fusing together several trees. It frequently achieves state-of-the-art outcomes in contests and real-world

activities, and is renowned for its **efficiency, high performance, and flexibility**.

VI. SVC

For classification problems, the Support Vector Classifier (SVC) is a potent machine learning model. The method finds the **optimum hyperplane** in the feature space to divide various classes. The objective is to **increase the margin** that separates the **classes' closest points** from the hyperplane. By utilizing various **kernel functions,** such as linear, polynomial, or radial basis function (RBF), SVC can handle both linear and non-linear classification cases. Although it might be computationally demanding for huge datasets, it works well with high-dimensional data.

Cross Validation

Cross-validation is a technique used to assess a model's performance and ensure its generalizability across different subsets of data. By splitting the dataset into multiple folds and training the model on some folds while validating it on others, cross-validation helps mitigate overfitting and provides a more reliable estimate of the model's effectiveness. Averaging the results from these different folds gives a clearer picture of the model's overall performance and robustness. This process helps in selecting the best model and tuning its parameters effectively.

HyperParameter Tuning

We used Randomized Search CV to efficiently explore a wide range of hyperparameters by sampling from specified distributions. This method evaluates a fixed number of random combinations of hyperparameters, which helps to identify the best-performing configuration without exhaustively searching the entire space. Once the optimal parameters are found, the model is trained on the dataset using these parameters to ensure the best performance and generalization.

3.5 Evaluation Matrices

A confusion matrix, classification report, accuracy score, AUC-ROC score along with ROC curve was generated for every combination of resampling techniques and classifiers to gather the performance metrics overview.

Accuracy measures the proportion of correctly predicted instances among all the instances. However, it can be deceptive in unbalanced datasets since it may get influenced by the dominance of the majority class. Thus, it becomes crucial to take other metrics like Precision, Recall and F1 score into consideration. AUC-ROC becomes the key metric for evaluation, especially in cases of imbalanced dataset as it gives the overall performance measure.

After studying the various classification reports for all the models, it was observed that using the imbalanced dataset, the majority class (0) had much higher values for precision, recall and F1 score when compared to that of the minority class (1). This anomaly is highlighted in the AUC-ROC scores as well. Thus, both accuracy and AUC-ROC are taken into consideration to compare various combinations of balancing techniques and classifiers.

The following section covers the difference generated in some of the stated matrices before and after resampling, and among the various classifiers.

4. RESULTS & DISCUSSION

4.1 Performance Comparison

Table 1 and 2 illustrate the results of accuracy and AUC-ROC scores for predictions done based upon the imbalanced(raw) dataset as well as that of various combinations of classifiers and resampling techniques for comparison.

Furthermore, Figure 5 depicts the classification report and ROC curve for the Random Forest Classifier which proves to be the best performing model when random oversampling was applied. Though we have studied the report and graph for every combination, only the comparison of the performance before and after resampling and **cross validation** for the best performing model is shown.

4.2 Analysis of Results

It can be observed from the first statistical column in each table where PCA was applied on imbalanced(raw) dataset along with hyperparameter tuning for the respective classifier, that while the accuracy can be considered good (0.84 approx), the ROC-AUC score is under par (0.57 approx). This anomaly is generated because the model is influenced by the dominance of the majority class, it performs better for the majority class (class 0) but struggles with the minority class (class 1).

This was also apparent from studying the classification report where the precision, recall and F1 scores for minority class was lesser when compared to the majority class.

The ROC-AUC score has improved considerably with the implementation of the balancing techniques for all the classifiers, as can be seen from Table 2.

4.2.1 Accuracy Score Analysis

SVM, KNN, and logistic regression all do reasonably well in the raw dataset, with accuracies of about 0.85. However, when applied to the raw data, decision trees exhibit the lowest accuracy (0.7891), which could be a sign of problems like overfitting or underfitting. Even though it isn't the best, Random Forest has a respectable accuracy of 0.833.

Random Forest performs the best for the oversampled dataset, especially when Random Oversampling is used. At 0.9818, it had the highest accuracy. At 0.9534, XG boost displayed the second-highest accuracy. While KNN's performance decreases with ADASYN (0.8434), it still performs well with Random Oversampling (0.9068). Random Oversampling considerably increases Decision Tree accuracy (0.8906), but SMOTE and ADASYN cause a discernible decrease in accuracy.

TABLE 1. ACCURACY SCORE

MODELS	PCA+RSCV on RAW dataset	PCA+RSCV on OVERSAMPLED dastaset			PCA+RSCV on UNDERAMPLED dataset			PCA+RSCV with HYBRID Techniqu	
		Random Oversampling	SMOTE	ADASYN	Random Undersampling	Tomek Links	ENN	SMOTE-TomekLinks	SMOTE-ENN
Logistic Regression	0.8775	0.7672	0.8238	0.7974	0.7157	0.8759	0.826	0.8119	0.8061
XGBoost	0.8425	0.9534	0.8421	0.8517	0.7368	0.8613	0.8067	0.8547	0.8469
KNN	0.8503	0.9068	0.838	0.8434	0.7263	0.8248	0.7729	0.8311	0.8503
Decision Trees	0.7891	0.8906	0.7651	0.7724	0.6526	0.7554	0.7729	0.7393	0.8061
SVC	0.8571	0.9109	0.8704	0.8622	0.7052	0.8248	0.7971	0.8653	0.8605
Random Forest	0.833	**0.9818**	0.9056	0.891	0.7473	0.8515	0.8309	0.9033	0.9265

TABLE 2. ROC-AUC SCORE

	PCA+RSCV on RAW dataset	PCA+RSCV on OVERSAMPLED dastaset			PCA+RSCV on UNDERAMPLED dataset			PCA+RSCV with HYBRID Techniqu	
		Random Oversampling	SMOTE	ADASYN	Random Undersampling	Tomek Links	ENN	SMOTE-TomekLinks	SMOTE-ENN
Logistic Regression	0.6687	0.7672	0.8239	0.7972	0.7156	0.6704	0.6546	0.812	0.806
XGBoost	0.5537	0.9534	0.8421	0.851	0.736	0.6288	0.6271	0.8547	0.8463
K-Nearest Neighbors	0.5664	0.9069	0.8381	8.444	0.7258	0.5246	0.5526	0.8312	0.8397
Decision Trees	0.5385	0.8907	0.7652	0.7718	0.652	0.581	0.6352	0.7393	0.8069
Support Vector Mach	0.579	0.9109	0.8704	0.8614	0.7055	0.5	0.5832	0.8654	0.8595
Random Forest	0.5476	**0.9999**	0.9681	0.9602	0.7826	0.8026	0.8243	0.9659	0.9747

Table 1&2. Depict scores for Accuracy and ROC-AUC

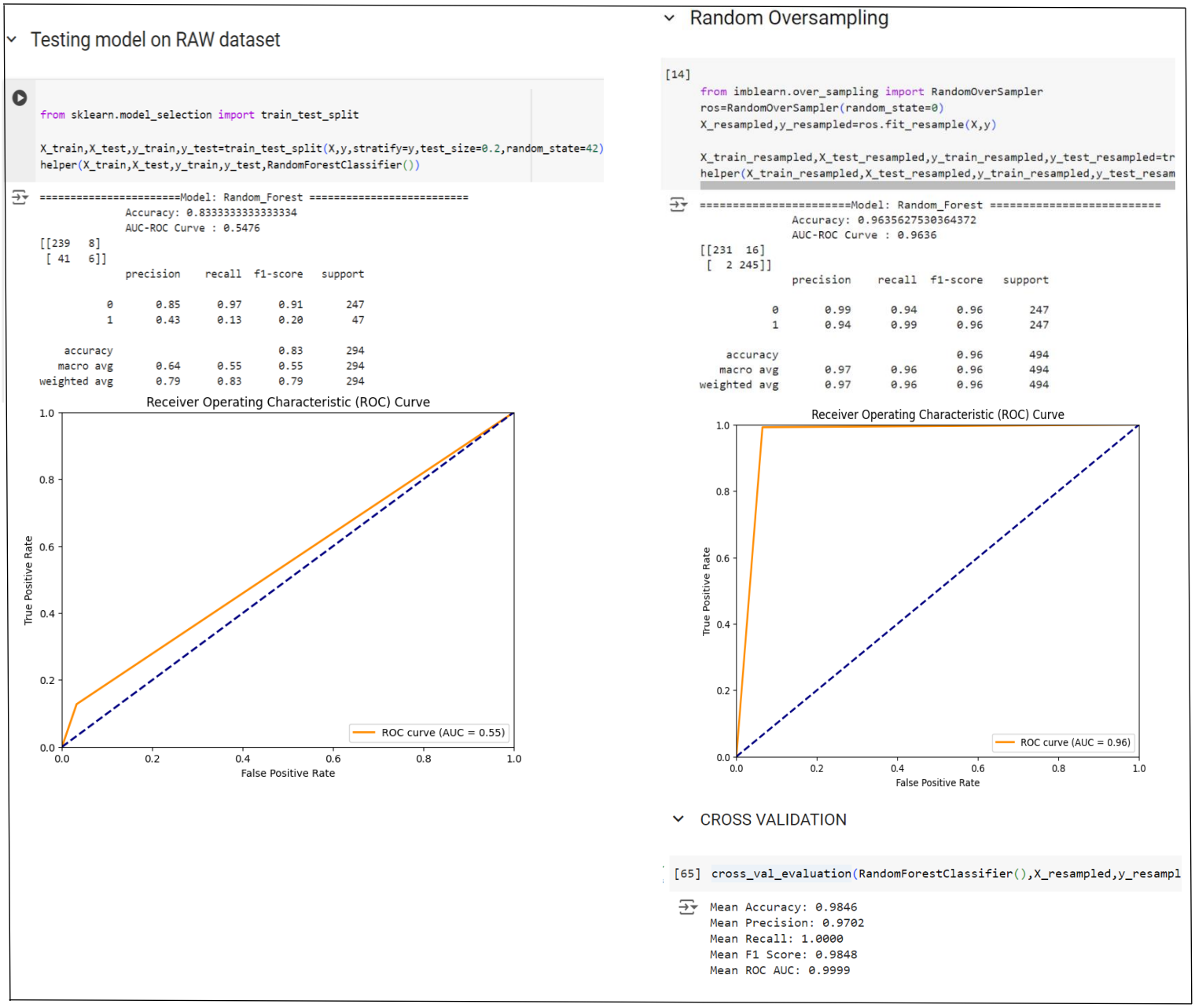

Figure 5. Results of Random Forest Classifier before & after Random Oversampling

Across all models, Tomek Links perform better than Random Undersampling in the undersampled dataset. When Tomek Links are used, the accuracy of SVM and logistic regression in particular is 0.8248 and 0.8759, respectively, demonstrating their resilience to class imbalance. However, with both undersampling strategies, Decision Trees perform far worse, highlighting how sensitive they are to data reduction.

SMOTE-Tomek Links and SMOTE-ENN produce competitive results when hybrid techniques are used, frequently outperforming individual oversampling or undersampling techniques. The best performing algorithms under SMOTE-Tomek Links are Random Forest and SVC, with accuracy values of 0.9033 and 0.8653, respectively. Random Forest yielded the best results for SMOTE-ENN, with an accuracy of 0.9265.

4.2.2 ROC-AUC Score Analysis

Logistic regression has the greatest ROC-AUC value (0.6687) in the raw dataset, showing stronger discriminative capacity at the expense of lesser accuracy. The comparatively low ROC-AUC scores of other models, such as Decision Trees and SVM, point to possible problems with successfully classifying data.

Random Forest exhibits better performance in the oversampled dataset once more. ROC-AUC values for Random Forest are almost ideal when SMOTE (0.9681) and Random Oversampling (0.9999) are used. Oversampling is beneficial for XGBoost and SVC as well, particularly when using Random Oversampling.

Random Undersampling produces higher ROC-AUC values for the undersampled dataset than Tomek Links and ENN, especially for SVM and Logistic Regression. Nevertheless, undersampling presents challenges for Decision Trees and KNN, indicating their incapacity to process fewer data points efficiently. On the other hand, For Random Forest Random Undersampling was least successful.

In conclusion, the hybrid approaches—that is, SMOTE-Tomek Links and SMOTE-ENN—perform exceptionally well in terms of ROC-AUC for all models, with Random Forest and SVC at the front of the pack. Random Forest with SMOTE-ENN (0.9747) has the best ROC-AUC score, closely followed by SMOTE-Tomek Links (0.9659).

5. CONCLUSION

Different sampling approaches have a substantial effect on model performance. For example, Random Oversampling and SMOTE improved model accuracy and ROC-AUC scores, particularly for XGBoost and Random Forest. ADASYN also increases scores, though not as consistently. Random undersampling reduced performance. Hybrid approaches such as SMOTE-Tomek Links and SMOTE-ENN work well, with the latter frequently producing the top scores across multiple models.
Across all sampling strategies, Random Forest and XGBoost regularly outperform other models, especially when it comes to oversampling and hybrid techniques. While stable across different approaches, logistic regression and SVC fall short of ensemble models such as Random Forest and XGBoost in terms of performance. However, undersampling is a problem for Decision Trees and KNN, suggesting that more advanced techniques are required to handle imbalanced data. A solid compromise between oversampling and undersampling is provided by the hybrid techniques (SMOTE-Tomek Links and SMOTE-ENN), which frequently result in enhanced model performance. In this investigation, the most successful methods are XGBoost and Random Forest, especially when Random Oversampling, SMOTE-Tomek Links, or SMOTE-ENN are used.

Overall Random Forest model performed best with an accuracy of 98.18% and AUC-ROC score of 99.99% when cross validation was performed after using Random Oversampling for balancing.

6. ACKNOWLEDGEMENTS

We would like to express our profound gratitude to Dr. Ritu, professor in Indira Gandhi Delhi Technical University and Dr. Nidhi Grover for their invaluable guidance and support throughout the course of this research.

7. REFERENCES

[1] Dataset: https://www.kaggle.com/datasets/pavansubhasht/ibm-hr-analytics-attrition-dataset

[2] Vinoodhini, D. (2022, April). Effective Classification Of Ibm Hr Analytics Employee Attrition Using Sampling Techniques. In *2022 Second International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT)* (pp. 1-6). IEEE.

[3] Konar, K., Das, S., & Das, S. (2023, January). Employee attrition prediction for imbalanced data using genetic algorithm-based parameter optimization of XGB Classifier. In *2023 International Conference on Computer, Electrical & Communication Engineering (ICCECE)* (pp. 1-6). IEEE.

[4] Priyana, I., Alamsyah, N., Sarifiyono, A. P., & Rusnendar, E. (2024, June). Predictive Boosting for Employee Retention with SMOTE and XGBoost Hyperparameter Tuning. In *2024 International Conference on Smart Computing, IoT and Machine Learning (SIML)* (pp. 92-97). IEEE.

[5] Sharma, S., & Sharma, K. (2023, June). Analyzing Employee's Attrition and Turnover at Organization Using Machine Learning Technique. In *2023 3rd International Conference on Intelligent Technologies (CONIT)* (pp. 1-7). IEEE.

[6] Mitravinda, K. M., & Shetty, S. (2022, December). Employee Attrition: Prediction, Analysis Of Contributory Factors And Recommendations For Employee Retention. In *2022 IEEE International Conference for Women in Innovation, Technology & Entrepreneurship (ICWITE)* (pp. 1-6). IEEE.

[7] Fallucchi, F., Coladangelo, M., Giuliano, R., & William De Luca, E. (2020). Predicting employee attrition using machine learning techniques. *Computers*, *9*(4), 86.

[8] Raza, A., Munir, K., Almutairi, M., Younas, F., & Fareed, M. M. S. (2022). Predicting employee attrition using machine learning approaches. *Applied Sciences*, *12*(13), 6424.

[9] 19 Employee Retention Stastics That Will surprise you. 2024, apollotechnical.com- available online..

[10] Nitesh, V. C. (2002). SMOTE: synthetic minority over-sampling technique