Improving Classification Accuracy with Imbalanced Data in Python

A report submitted for the course of

APPLICATION DEVELOPMENT – PROJECT EXPLORE

III B. Tech II Semester

By

AkshayaSri Kadiyala	**- 2211IT010005**
CH. Akhil	**- 2211IT010030**
G.Chinna Reddy	**- 2211IT010040**
K.Sannitha	**- 2211IT010048**
K.Pavan Kumar	**- 2211IT010055**

Under the esteemed guidance of

Mrs. Lavanya Tummala
Assistant Professor

DEPARTMENT OF INFORMATION TECHNOLOGY

Malla Reddy University
Maisammaguda, Dulapally, Hyderabad, Telangana 500100
www.mallareddyuniversity.ac.in
2024-25

MALLA REDDY UNIVERSITY

(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that this bonafide record of the Application Development entitled **"Improving Classification Accuracy with Imbalanced Data in Python"** submitted **by Ms. AkshayaSri Kadiyala 2211IT010005** , **Mr. CH. Akhil 2211IT010030** , **Mr. G. Chinna Reddy 2211IT010040**, **Ms. K. Sannitha Reddy 2211IT010048**, **Mr. K. Pavan Kumar 2211IT010055** of **III** year **II** semester to the **Malla Reddy University**, Hyderabad. This bonafide record of work carried out by us under the guidance of our supervision. The contents of this report, in full or in parts, have not been submitted to any other Organization for the award of any Degree.

Internal Guide:
Mrs. Lavanya Tummala
Assistant Professor

Head of the Department
Dr. A.V.L.N. Sujith

External Examiner

Date:

MALLA REDDY UNIVERSITY
(Telangana State Private Universities Act No.13 of 2020 and G.O.Ms.No.14, Higher Education (UE) Department)

INFORMATION TECHNOLOGY

DECLARATION

We certify that

a. The work contained in this report is original and has been done by us under the guidance of our supervisor.

b. The work has not been submitted to any other Organization for any degree.

c. We have followed the guidelines provided by the Organization in preparing the report.

d. We have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Organization.

e. Whenever we have used materials (data, theoretical analysis, figures, and text) from other sources, we have given due credit to them by citing them in the text of the report and giving their details in the references.

AkshayaSri Kadiyala	**- 2211IT010005**
CH. Akhil	**- 2211IT010030**
G. Chinna Reddy	**- 2211IT010040**
K. Sannitha Reddy	**- 2211IT010048**
K. Pavan Kumar	**- 2211IT010055**

ACKNOWLEDGEMENT

We would like take this opportunity to acknowledge everyone who has helped us in every stage of this project.

We ineffably indebted **Mrs. Lavanya Tummula**, Associate Professor, Dept. of IT for conscientious guidance and encouragement to accomplish this Application

We wish to convey our sincere thanks to **Dr. A.V.L.N. Sujith, Head of Department, Dept. of IT, Malla Reddy University** for his continuous support.

We would like to express our special gratitude and thanks to **Dr. VSK Reddy, Vice Chancellor, Malla Reddy University** for his continuous support.

We are also grateful to all administrative staff and the technical staff of Information Technology Department, people who provided us the useful and necessary information needed for this project.

ABSTRACT

In machine learning, handling imbalanced data is a significant challenge, particularly in classification tasks where one class is vastly underrepresented compared to the other. This imbalance can lead to biased models that favor the majority class, ultimately reducing the model’s effectiveness in predicting minority class instances. Since many real-world applications, such as fraud detection, medical diagnosis, and anomaly detection, depend on accurately identifying the minority class, improving classification accuracy under these conditions is crucial. To address this issue, several techniques can be employed. Resampling methods, such as oversampling the minority class and undersampling the majority class, help balance the dataset and improve model learning. Additionally, modifying algorithm parameters, such as adjusting class weights, can provide better representation for the minority class. Machine learning models like Decision Trees, Random Forests, and Gradient Boosting algorithms tend to handle imbalanced data more effectively than traditional linear models. Furthermore, ensemble learning techniques can enhance prediction accuracy by combining multiple models to improve robustness. Another critical aspect is the evaluation of model performance. Standard accuracy metrics can be misleading when dealing with imbalanced data, as a model can achieve high accuracy by simply predicting the majority class most of the time. Instead, alternative metrics such as Precision, Recall, F1-score, and the Area Under the Receiver Operating Characteristic (AUROC) curve provide a more meaningful assessment of classification performance. By integrating these strategies, machine learning practitioners can develop models that perform well across both majority and minority classes, improving overall predictive capabilities. This paper explores various techniques and best practices to enhance classification accuracy for imbalanced datasets, ensuring that models remain reliable and effective in real-world applications.

CONTENTS

LIST OF FIGURES

CHAPTER 1
Introduction

1.1 Context and Significance

Context

Imbalanced data is a common challenge in classification tasks, where one class is significantly underrepresented, leading to biased model performance. This issue is particularly critical in domains such as fraud detection, medical diagnosis, credit risk assessment, and cybersecurity, where failing to detect minority class instances can have severe financial, security, or health-related consequences. Standard machine learning models tend to favor the majority class due to their assumption of a balanced dataset, which results in poor generalization for the minority class. To address this, specialized techniques such as resampling (oversampling or undersampling), cost-sensitive learning, and class weighting are employed to improve the model's ability to learn from imbalanced data, ensuring better classification outcomes.

Significance

Handling class imbalance is essential to building robust and reliable machine learning models, especially in high-stakes applications where misclassification can have significant implications. When left unaddressed, imbalanced data can lead to misleading model performance, where high accuracy masks the poor prediction of the minority class. Techniques such as synthetic data generation (e.g., SMOTE), cost-sensitive algorithms, and ensemble learning methods like boosting and bagging can help mitigate this issue. Additionally, evaluation metrics like Precision, Recall, F1-score, and AUROC provide a more comprehensive understanding of a model’s performance beyond simple accuracy. Implementing these strategies ensures that machine learning models remain effective, ethical, and capable of making fair decisions in real-world scenarios. Finally, leveraging automated hyperparameter tuning and feature engineering can further enhance model performance, ensuring that classifiers are optimized to effectively handle imbalanced datasets.

1.2 Objectives

1. **Understand the Impact of Imbalanced Data** – Analyze how class imbalance affects machine learning models, particularly in classification tasks, and identify the challenges it poses in real-world applications such as fraud detection, medical diagnosis, and cybersecurity.
2. **Explore Techniques for Handling Imbalanced Data** – Investigate various methods, including oversampling, undersampling, cost-sensitive learning, and synthetic data generation (e.g., SMOTE), to improve model training and ensure better representation of minority class instances.
3. **Evaluate Algorithm Performance on Imbalanced Datasets** – Assess the effectiveness of different machine learning algorithms, such as Decision Trees, Random Forests, Gradient Boosting, and ensemble methods, in handling imbalanced data and improving classification accuracy.
4. **Optimize Model Performance Using Advanced Strategies** – Implement class weighting, cost-sensitive learning, and ensemble techniques like bagging and boosting to enhance model robustness and improve predictions for the minority class.
5. **Use Appropriate Evaluation Metrics** – Shift focus from traditional accuracy-based metrics to more insightful measures such as Precision, Recall, F1-score, and AUROC, ensuring a more meaningful assessment of model effectiveness in imbalanced scenarios.
6. **Develop Practical Guidelines for Real-World Applications** – Provide actionable insights and best practices for data scientists and machine learning practitioners to effectively handle imbalanced datasets, ensuring reliable and fair decision-making in AI-driven systems.

1.3 Key Features

1. **Data Preprocessing Techniques** – Methods like **oversampling, undersampling, and SMOTE** help balance datasets and improve model learning.
2. **Cost-Sensitive Learning** – Assigning higher misclassification costs to the minority class to improve detection accuracy.
3. **Ensemble Learning** – Techniques such as **Boosting (e.g., XGBoost, AdaBoost)** and **Bagging (e.g., Random Forests)** enhance model robustness.
4. **Advanced Evaluation Metrics** – Metrics like **Precision, Recall, F1-score, and AUROC** provide a more accurate assessment of model performance.
5. **Explainable AI (XAI) Techniques** – Enhancing model interpretability to build trust in AI-driven decisions, particularly in high-stakes domains.
6. **Deep Learning Approaches** – Leveraging **autoencoders and anomaly detection** to improve minority class detection in complex datasets.
7. **Transfer Learning** – Utilizing pre-trained models to improve classification performance on imbalanced datasets with limited labelled data.
8. **Real-Time Monitoring and Adaptive Learning** – Continuously assessing and adjusting model performance to maintain accuracy as data distributions evolve.

CHAPTER 2
Review Of Relevant Literature

1. **Data-Level Techniques** – Methods such as oversampling, undersampling, and SMOTE help balance datasets, but they come with risks like overfitting (Chawla et al., 2002) or information loss (Batista et al., 2004). Hybrid approaches attempt to mitigate these issues.
2. **Algorithm-Level Strategies** – Cost-sensitive learning (Domingos, 1999) and class-weighted models modify traditional algorithms by prioritizing minority class detection, reducing bias in classification models.
3. **Ensemble Methods** – Boosting techniques like AdaBoost and XGBoost, along with Bagging methods such as Random Forests, improve classification robustness by integrating multiple weak models (Freund & Schapire, 1997).
4. **Evaluation Metrics** – Instead of relying on traditional accuracy, researchers emphasize Precision, Recall, F1-score, AUROC, and Precision-Recall curve**s** for better assessment of model performance on imbalanced datasets (He & Garcia, 2009; Davis & Goadrich, 2006).
5. **Deep Learning Solutions** – Advanced methods such as autoencoders, GANs, and transfer learning have enhanced classification accuracy on imbalanced datasets by generating synthetic samples and extracting complex patterns (Goodfellow et al., 2014; Pan & Yang, 2010).
6. **Real-World Applications** – Addressing class imbalance is essential in domains like fraud detection (Dal Pozzolo et al., 2015), medical diagnosis (Fernández et al., 2018), cybersecurity (Zhang et al., 2019), and financial risk assessment, where improper classification can have significant consequences.

CHAPTER 3
Methodology

3.1 Problem Definition

The problem definition identifies the challenges related to class imbalance in machine learning classification tasks, forming the foundation for developing effective solutions.

Current Issues in Handling Class Imbalance:

- **Biased Model Predictions** – Standard machine learning algorithms tend to favor the majority class, leading to poor generalization for the minority class.
- **Misleading Evaluation Metrics** – Traditional accuracy-based evaluations fail to reflect the model’s actual performance, often overlooking minority class detection.
- **High-Risk Domains** – In critical applications like fraud detection, medical diagnosis, and cybersecurity, misclassification of rare but important instances can have severe consequences.

Scope of the Problem:

- **Need for Improved Learning Techniques** – Developing methods such as resampling, cost-sensitive learning, and ensemble techniques to address class imbalance.
- **Enhancing Evaluation Approaches** – Using advanced metrics like Precision, Recall, and AUROC for better performance assessment.

Purpose of the System:

- To develop machine learning models that effectively handle class imbalance, ensuring accurate predictions.
- To integrate advanced techniques like ensemble learning and cost-sensitive training for improved classification performance.
- To enhance real-world applicability in high-risk domains, reducing misclassification rates.

3.2 Objectives of the Application

The objectives define the key goals of the application, ensuring an effective approach to handling class imbalance in machine learning models.

Primary Objectives:

- **To improve the classification performance of machine learning models** by addressing class imbalance using advanced techniques like resampling, cost-sensitive learning, and ensemble methods.
- **To enhance the detection of minority class instances** in high-risk domains such as fraud detection, medical diagnosis, and cybersecurity, reducing misclassification rates.
- **To implement appropriate evaluation metrics** such as Precision, Recall, F1-score, and AUROC, ensuring a more reliable assessment of model performance.

Secondary Objectives:

- **To explore deep learning approaches** like GANs, autoencoders, and transfer learning for generating synthetic data and improving model robustness.
- **To develop adaptive models** that dynamically adjust to changes in data distributions, ensuring long-term effectiveness.
- **To provide scalable and automated solutions** that can be applied across different real-world applications without extensive manual intervention.

3.3 System Design

The system design stage involves creating a blueprint of the application, including its architecture and functionality.

i. UML Diagrams -

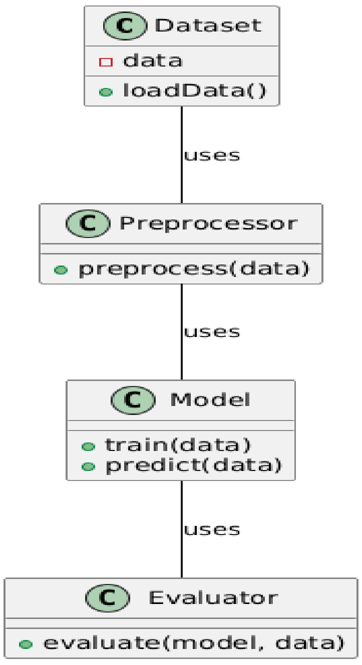


Fig 1: Class Diagram

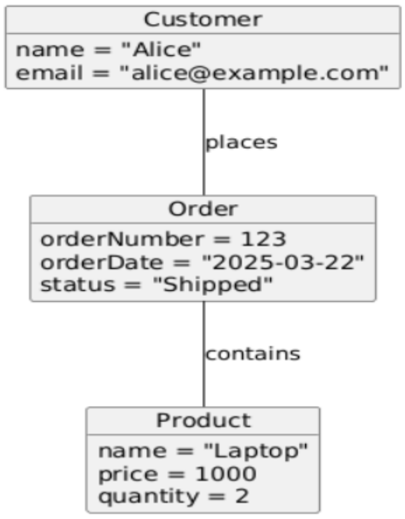


Fig 2: Object Diagram

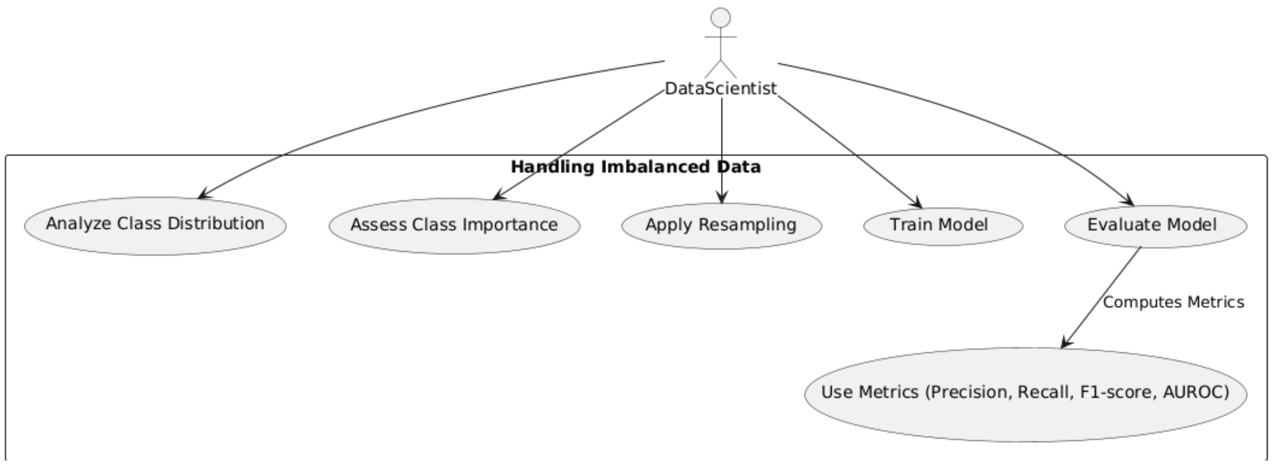


Fig 3: Use Case Diagram

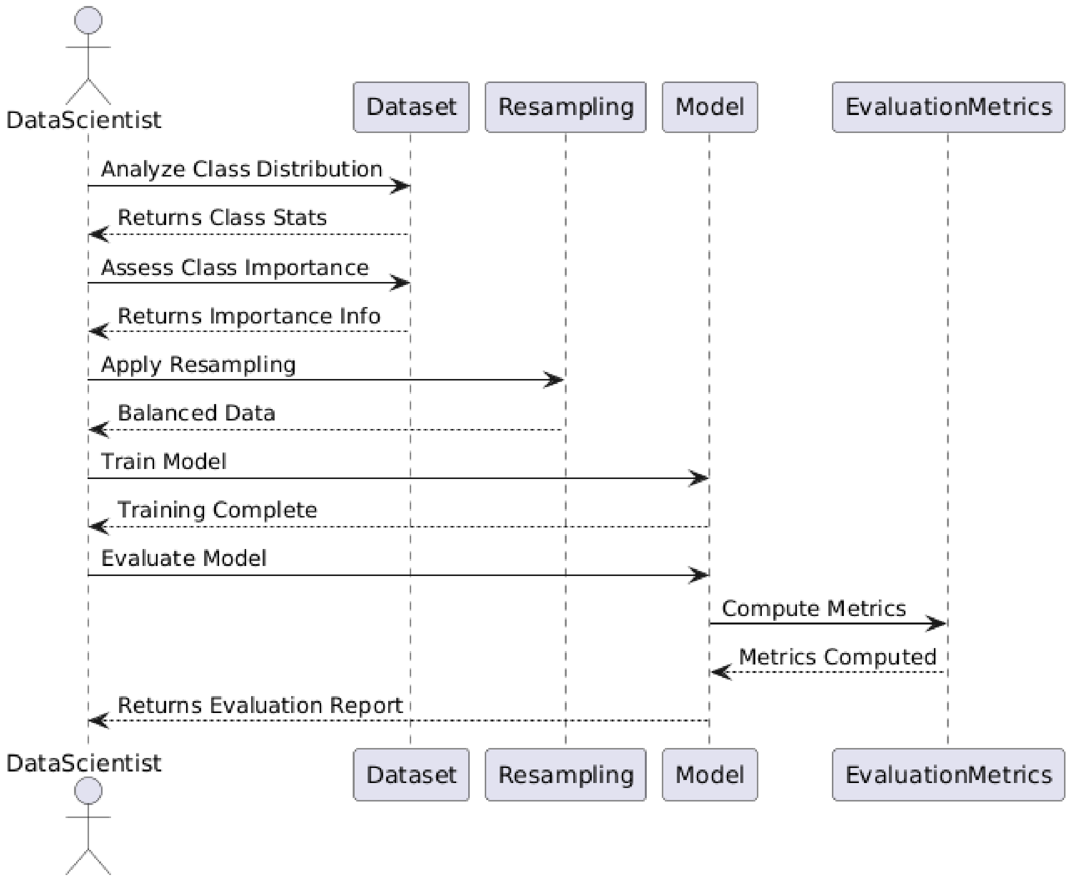


Fig 4: Sequence Diagram

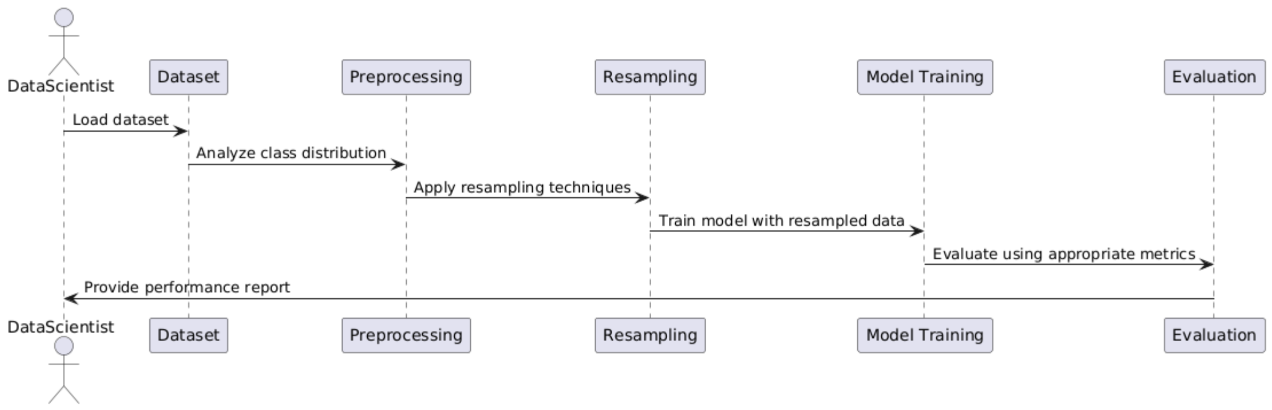

DataScientist
Dataset
Preprocessing
Resampling
Model Training
Evaluation
Load dataset
Analyze class distribution
Apply resampling techniques
Train model with resampled data
Evaluate using appropriate metrics
Provide performance report

Fig 5: Collaboration Diagram

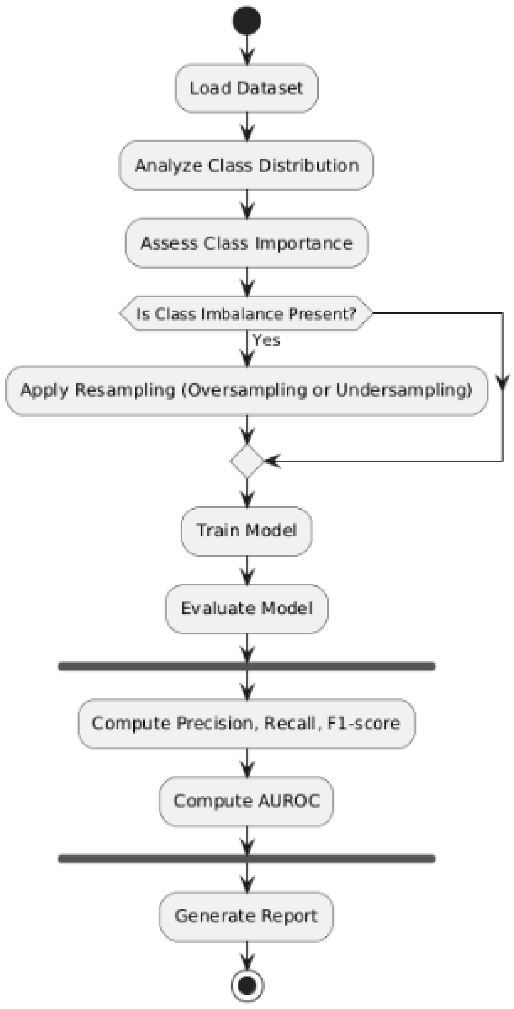

Load Dataset
Analyze Class Distribution
Assess Class Importance
Is Class Imbalance Present?
Yes
Apply Resampling (Oversampling or Undersampling)
Train Model
Evaluate Model
Compute Precision, Recall, F1-score
Compute AUROC
Generate Report

Fig 6: Activity Diagram

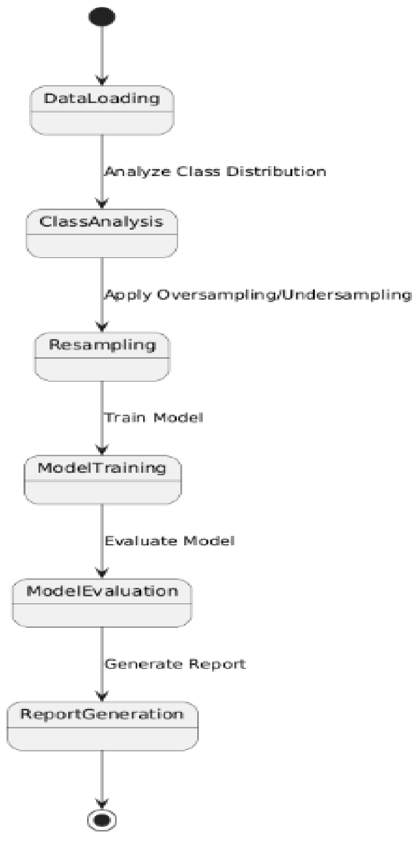


Fig 7: Statechart Diagram

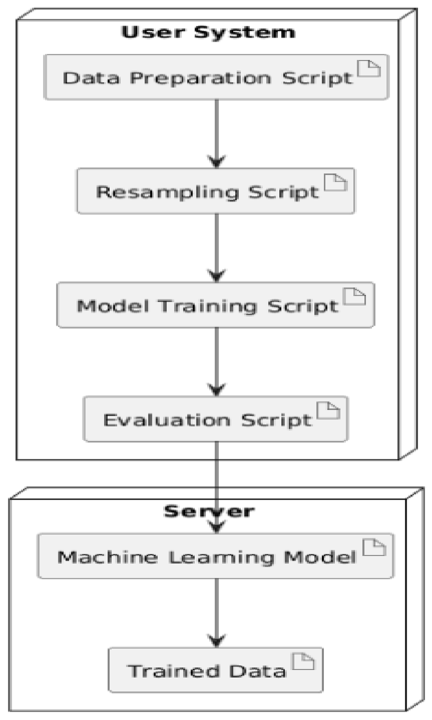


Fig 8: Deployment Diagram

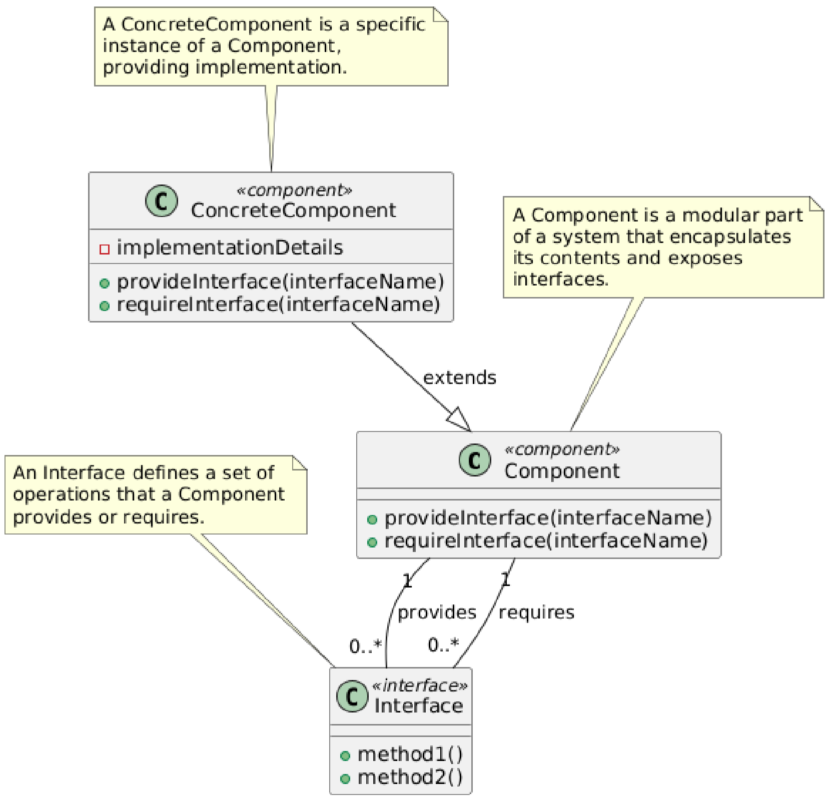

A ConcreteComponent is a specific instance of a Component, providing implementation.
«component» ConcreteComponent
implementationDetails
provideInterface(interfaceName)
requireInterface(interfaceName)
A Component is a modular part of a system that encapsulates its contents and exposes interfaces.
extends
An Interface defines a set of operations that a Component provides or requires.
«component» Component
provideInterface(interfaceName)
requireInterface(interfaceName)
1
1
provides
requires
0..*
0..*
«interface» Interface
method1()
method2()

Fig 9: Component Diagram

ii. Modules and Functionalities

To effectively handle class imbalance in machine learning classification tasks, the system will be divided into key modules with specific functionalities.

1. Data Preprocessing Module

- **Data Cleaning** – Removes noise, duplicates, and irrelevant features from the dataset.
- **Data Balancing Techniques** – Implements oversampling, undersampling, and synthetic data generation (SMOTE, GANs, autoencoders) to address class imbalance.
- **Feature Engineering** – Selects and transforms important features to enhance model performance.
- **Dimensionality Reduction** – Uses techniques like PCA (Principal Component Analysis) to reduce complexity while retaining critical information.

2. Model Training and Optimization Module

- **Cost-Sensitive Learning** – Adjusts model training by assigning higher misclassification costs to the minority class.
- **Class-Weighted Algorithms** – Modifies traditional algorithms (SVM, Decision Trees, Neural Networks) to handle imbalanced datasets.
- **Ensemble Learning Methods** – Utilizes Boosting (AdaBoost, XGBoost) and Bagging (Random Forest) to improve classification accuracy.
- **Hyperparameter Tuning** – Implements grid search and automated tuning techniques to optimize model performance.

3. Deep Learning-Based Imbalance Handling Module

- **Synthetic Data Generation** – Uses Generative Adversarial Networks (GANs) and **autoencoders** to create minority class samples.
- **Transfer Learning** – Applies pre-trained deep learning models to improve performance on imbalanced datasets.
- **Anomaly Detection Techniques** – Leverages autoencoders and unsupervised learning for detecting rare patterns.

4. Evaluation and Performance Metrics Module

- **Traditional Metrics** – Includes accuracy, confusion matrix, and classification reports.
- **Advanced Metrics for Imbalanced Data** – Implements Precision, Recall, F1-score, AUROC, and Precision-Recall curves for more meaningful evaluation.
- **Model Comparison** – Enables performance benchmarking across different techniques to identify the most effective approach.

5. Adaptive Learning and Real-Time Monitoring Module

- **Real-Time Model Evaluation** – Continuously tracks model performance to detect shifts in data distribution.
- **Adaptive Learning Techniques** – Dynamically updates models in response to new imbalanced data patterns.
- **Drift Detection and Model Retraining** – Identifies when retraining is necessary based on data shifts.

6. Visualization and Reporting Module

- **Graphical Representation of Class Distribution** – Provides visual insights into data imbalance before and after processing.
- **Model Performance Dashboards** – Displays key performance metrics in an interactive and interpretable format.
- **Interpretability with Explainable AI (XAI)** – Uses SHAP (Shapley Additive Explanations) and LIME (Local Interpretable Model-agnostic Explanations) for better understanding of model decisions.

7. Deployment and Integration Module

- **API for Model Deployment** – Provides an interface for real-world applications to use the trained models.
- **Scalability and Cloud Integration** – Supports cloud-based solutions to handle large-scale data processing.
- **Integration with Real-World Applications** – Enables implementation **in** fraud detection, healthcare, finance, and cybersecurity systems.

3.4 User Interface

Data Flow Diagram –

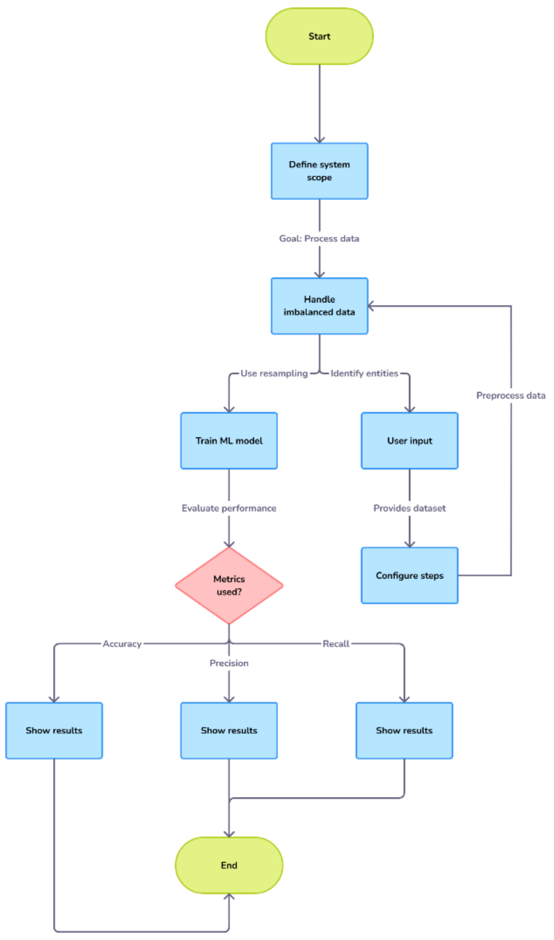


Fig 10: Data Flow Diagram

Database Diagram –

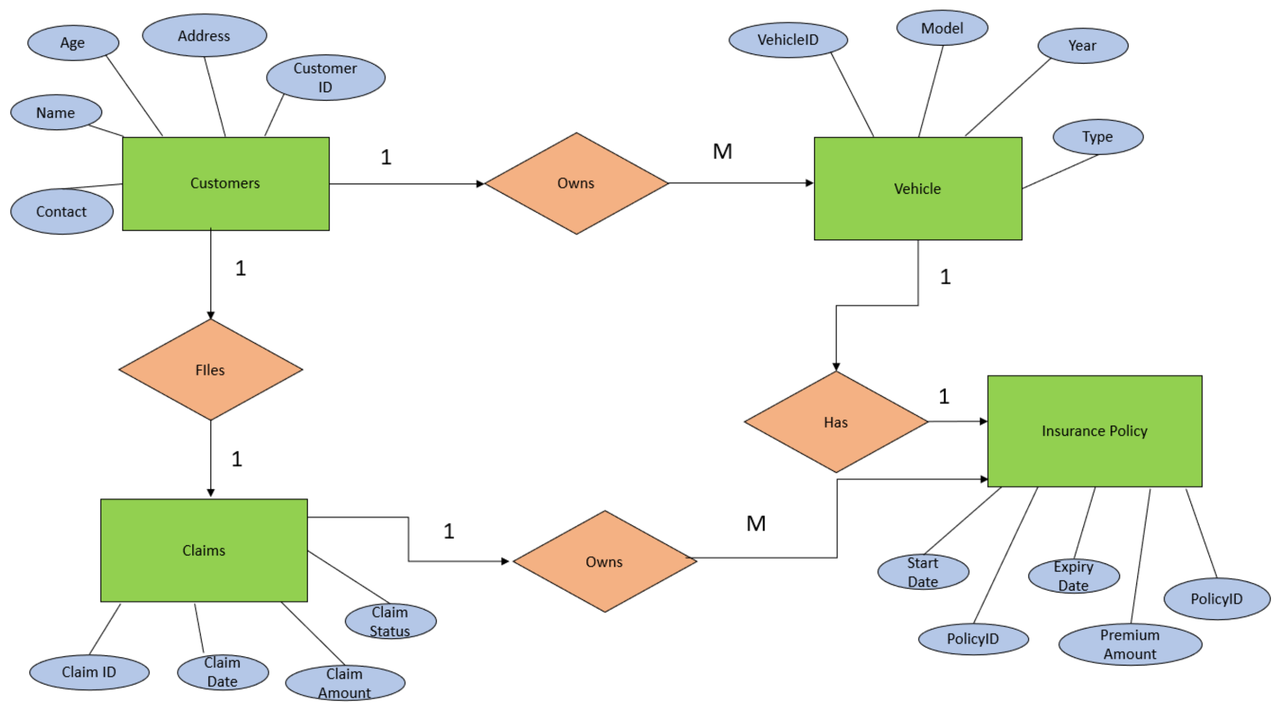
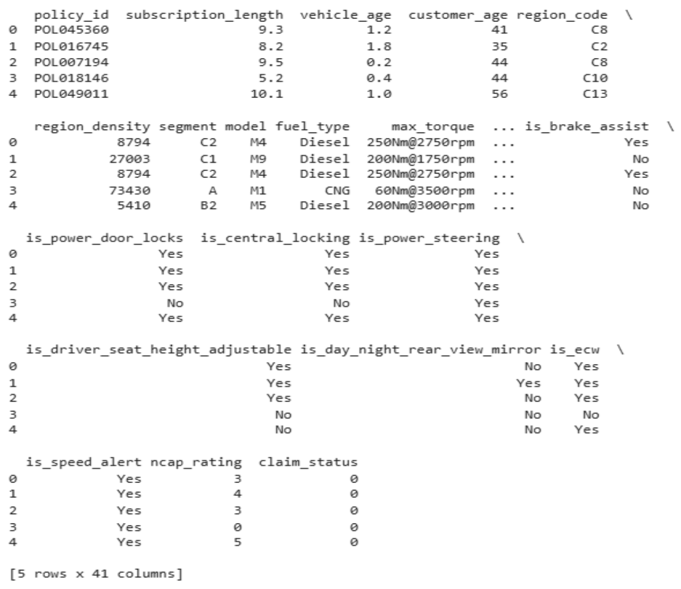


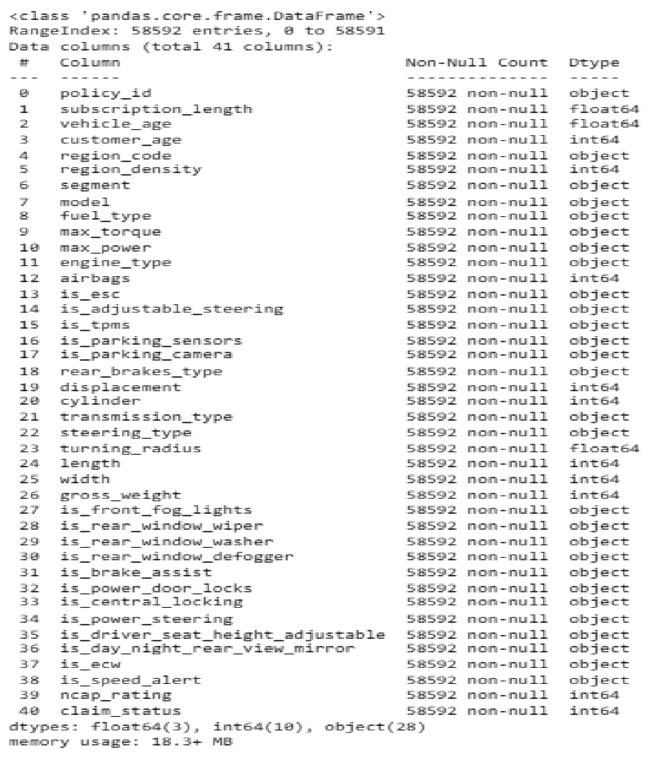
Fig 11: ER Diagram

CHAPTER 4
Results and Discussions



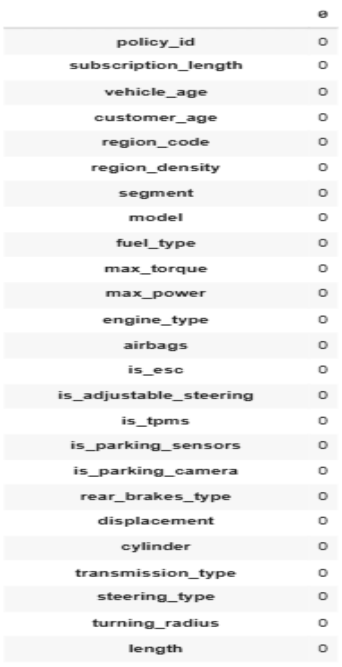
```
   policy_id  subscription_length  vehicle_age  customer_age region_code  \
0  POL045360                  9.3          1.2            41          C8
1  POL016745                  8.2          1.8            35          C2
2  POL007194                  9.5          0.2            44          C8
3  POL018146                  5.2          0.4            44         C10
4  POL049011                 10.1          1.0            56         C13

   region_density segment model fuel_type     max_torque  ... is_brake_assist  \
0            8794      C2    M4    Diesel  250Nm@2750rpm  ...             Yes
1           27003      C1    M9    Diesel  200Nm@1750rpm  ...              No
2            8794      C2    M4    Diesel  250Nm@2750rpm  ...             Yes
3           73430       A    M1       CNG   60Nm@3500rpm  ...              No
4            5410      B2    M5    Diesel  200Nm@3000rpm  ...              No

  is_power_door_locks  is_central_locking is_power_steering  \
0                 Yes                 Yes               Yes
1                 Yes                 Yes               Yes
2                 Yes                 Yes               Yes
3                  No                  No               Yes
4                 Yes                 Yes               Yes

  is_driver_seat_height_adjustable is_day_night_rear_view_mirror is_ecw  \
0                              Yes                            No    Yes
1                              Yes                           Yes    Yes
2                              Yes                            No    Yes
3                               No                            No     No
4                               No                            No    Yes

  is_speed_alert ncap_rating  claim_status
0            Yes           3             0
1            Yes           4             0
2            Yes           3             0
3            Yes           0             0
4            Yes           5             0

[5 rows x 41 columns]
```

Fig 12: Data Set



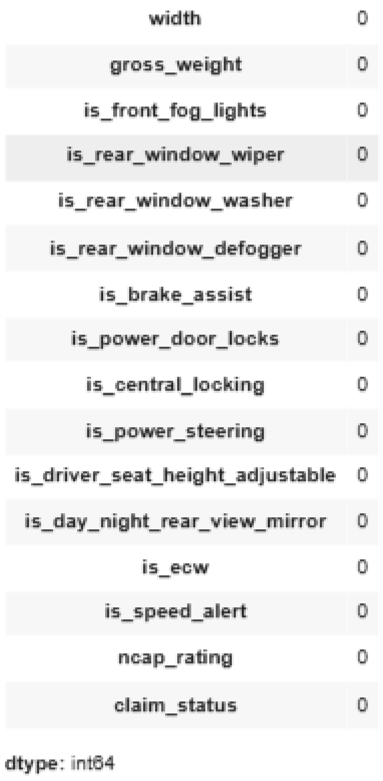
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 58592 entries, 0 to 58591
Data columns (total 41 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   policy_id                         58592 non-null  object
 1   subscription_length               58592 non-null  float64
 2   vehicle_age                       58592 non-null  float64
 3   customer_age                      58592 non-null  int64
 4   region_code                       58592 non-null  object
 5   region_density                    58592 non-null  int64
 6   segment                           58592 non-null  object
 7   model                             58592 non-null  object
 8   fuel_type                         58592 non-null  object
 9   max_torque                        58592 non-null  object
 10  max_power                         58592 non-null  object
 11  engine_type                       58592 non-null  object
 12  airbags                           58592 non-null  int64
 13  is_esc                            58592 non-null  object
 14  is_adjustable_steering            58592 non-null  object
 15  is_tpms                           58592 non-null  object
 16  is_parking_sensors                58592 non-null  object
 17  is_parking_camera                 58592 non-null  object
 18  rear_brakes_type                  58592 non-null  object
 19  displacement                      58592 non-null  int64
 20  cylinder                          58592 non-null  int64
 21  transmission_type                 58592 non-null  object
 22  steering_type                     58592 non-null  object
 23  turning_radius                    58592 non-null  float64
 24  length                            58592 non-null  int64
 25  width                             58592 non-null  int64
 26  gross_weight                      58592 non-null  int64
 27  is_front_fog_lights               58592 non-null  object
 28  is_rear_window_wiper              58592 non-null  object
 29  is_rear_window_washer             58592 non-null  object
 30  is_rear_window_defogger           58592 non-null  object
 31  is_brake_assist                   58592 non-null  object
 32  is_power_door_locks               58592 non-null  object
 33  is_central_locking                58592 non-null  object
 34  is_power_steering                 58592 non-null  object
 35  is_driver_seat_height_adjustable  58592 non-null  object
 36  is_day_night_rear_view_mirror     58592 non-null  object
 37  is_ecw                            58592 non-null  object
 38  is_speed_alert                    58592 non-null  object
 39  ncap_rating                       58592 non-null  int64
 40  claim_status                      58592 non-null  int64
dtypes: float64(3), int64(10), object(28)
memory usage: 18.3+ MB
```

Fig 13: Data Set Information



```
                              0
policy_id                     0
subscription_length           0
vehicle_age                   0
customer_age                  0
region_code                   0
region_density                0
segment                       0
model                         0
fuel_type                     0
max_torque                    0
max_power                     0
engine_type                   0
airbags                       0
is_esc                        0
is_adjustable_steering        0
is_tpms                       0
is_parking_sensors            0
is_parking_camera             0
rear_brakes_type              0
displacement                  0
cylinder                      0
transmission_type             0
steering_type                 0
turning_radius                0
length                        0
```

Fig 14.1: Null Values



```
width                              0
gross_weight                       0
is_front_fog_lights                0
is_rear_window_wiper               0
is_rear_window_washer              0
is_rear_window_defogger            0
is_brake_assist                    0
is_power_door_locks                0
is_central_locking                 0
is_power_steering                  0
is_driver_seat_height_adjustable   0
is_day_night_rear_view_mirror      0
is_ecw                             0
is_speed_alert                     0
ncap_rating                        0
claim_status                       0

dtype: int64
```

Fig 14.2: Null Values

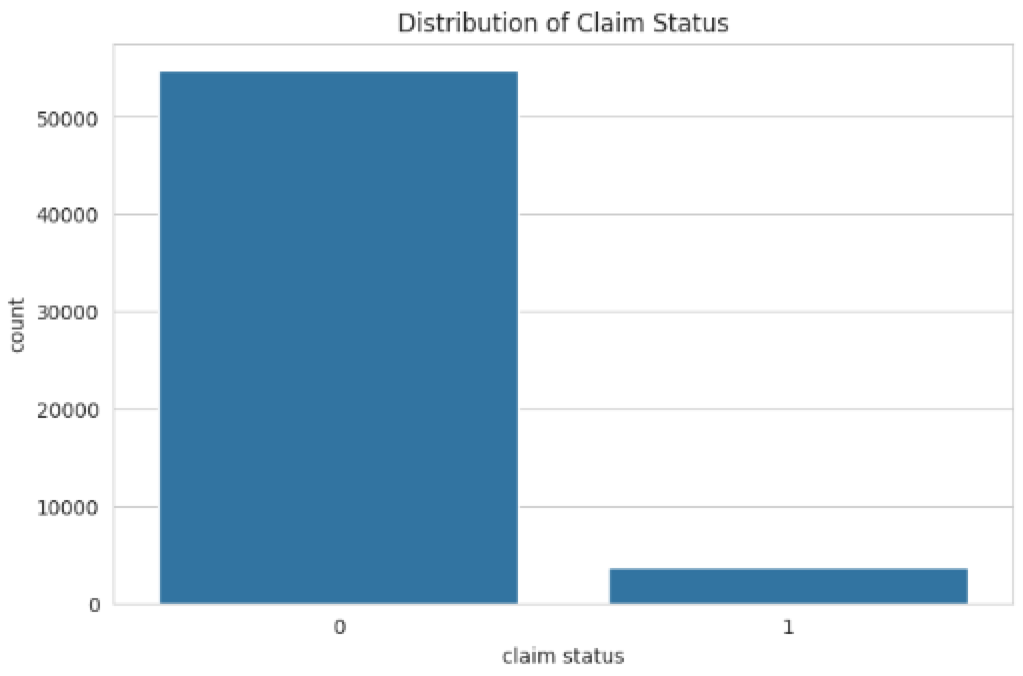


Fig 15: Distribution of the claim_status

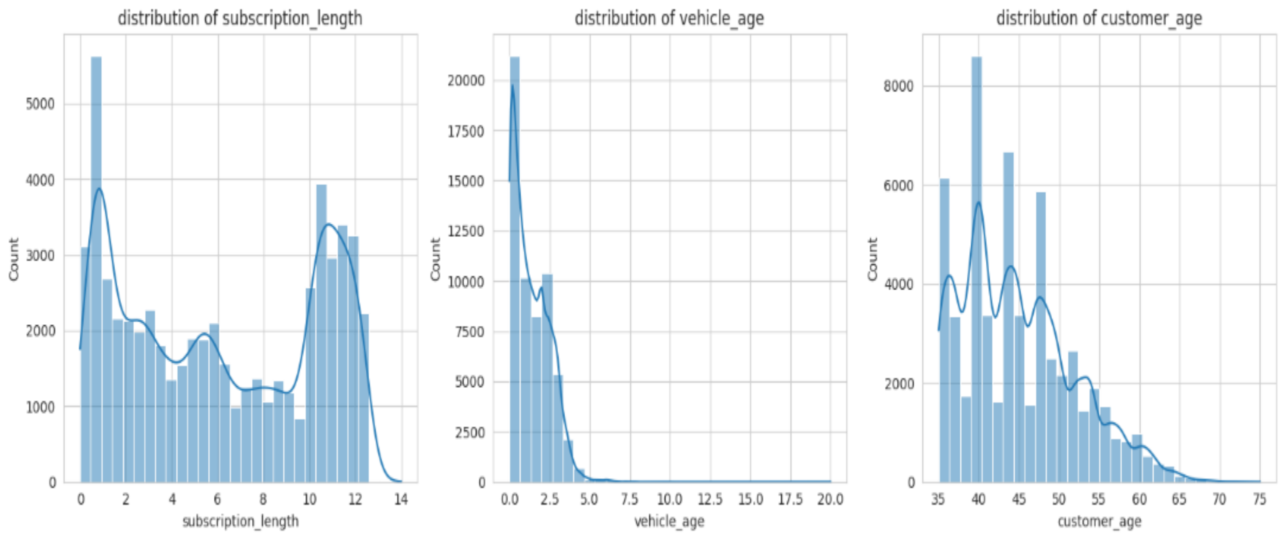


Fig 16: Distribution of the subscription_length, vehicle_age, and customer_age

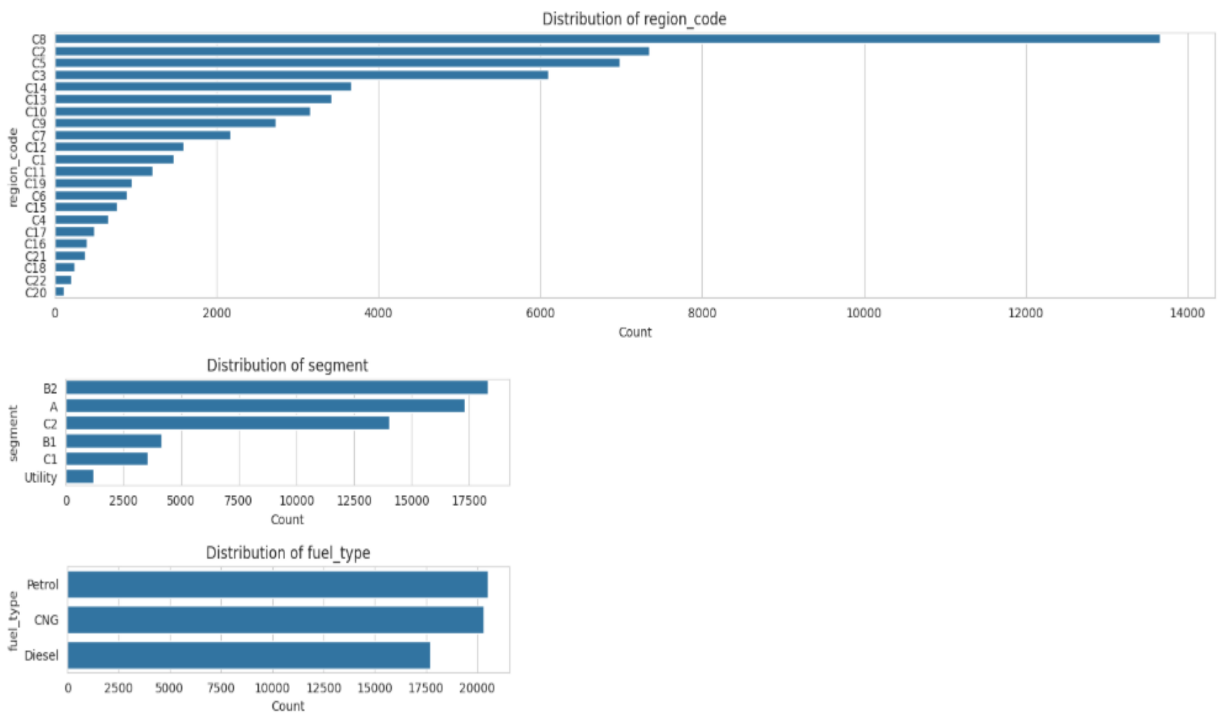
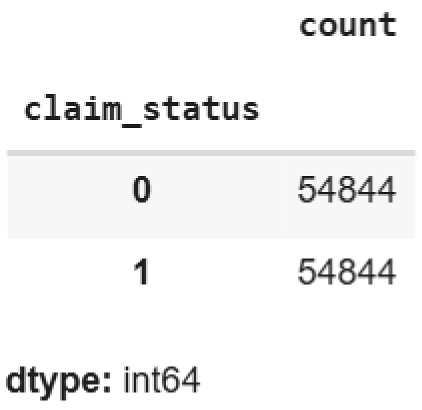


Fig 17: Distribution of region_code, segment, fuel_type



```
              count
claim_status
           0  54844
           1  54844

dtype: int64
```

Fig 18: Oversampled distribution

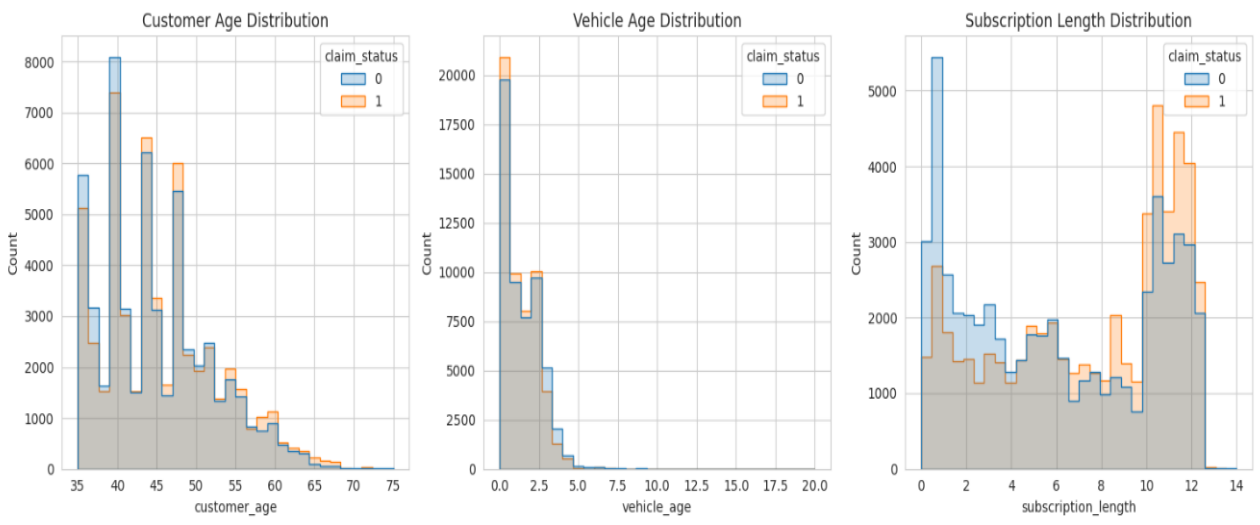
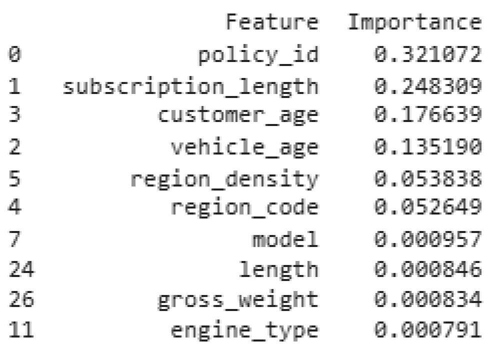
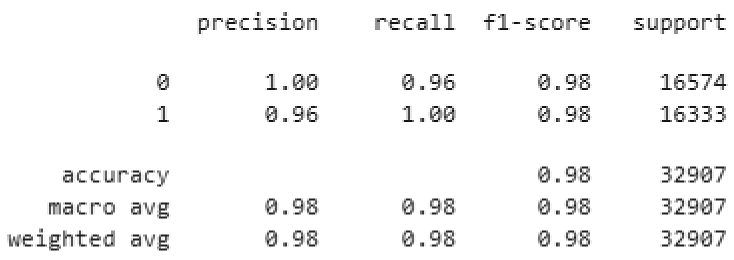


Fig 19: Distribution of customer_age, vehicle_age, and subscription_length



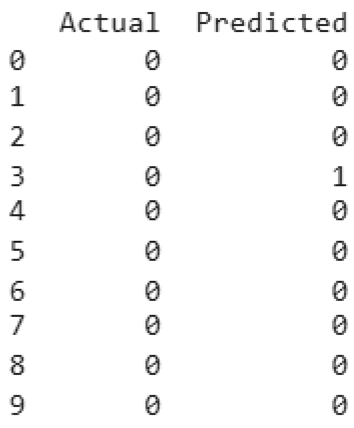
```
              Feature  Importance
0           policy_id    0.321072
1   subscription_length  0.248309
3        customer_age    0.176639
2         vehicle_age    0.135190
5      region_density    0.053838
4         region_code    0.052649
7               model    0.000957
24             length    0.000846
26       gross_weight    0.000834
11        engine_type    0.000791
```

Fig 20.1: Feature Selection



```
              precision    recall  f1-score   support

           0       1.00      0.96      0.98     16574
           1       0.96      1.00      0.98     16333

    accuracy                           0.98     32907
   macro avg       0.98      0.98      0.98     32907
weighted avg       0.98      0.98      0.98     32907
```

Fig 20.2: Feature Selection



```
   Actual  Predicted
0       0          0
1       0          0
2       0          0
3       0          1
4       0          0
5       0          0
6       0          0
7       0          0
8       0          0
9       0          0
```

Fig 21: Model Training

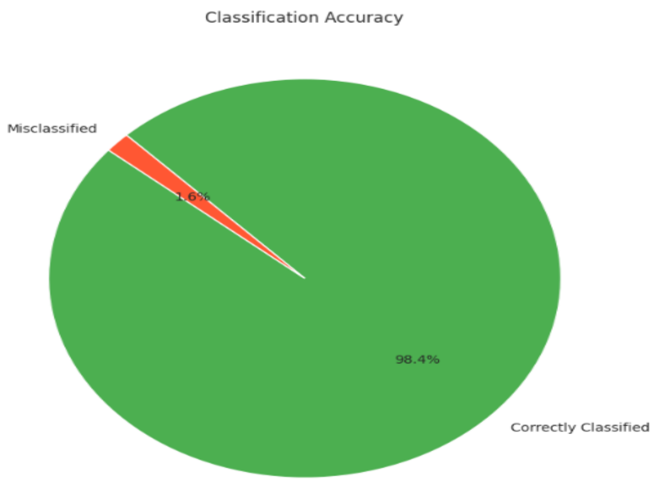


Fig 22: Classification Accuracy

CHAPTER 5
Conclusions and Future Scope of Study

Conclusion:

1. **Effective Handling of Class Imbalance** – Implementing techniques like resampling, cost-sensitive learning, ensemble methods, and deep learning improves the classification of minority class instances.
2. **Improved Model Performance** – Advanced evaluation metrics such as Precision, Recall, F1-score, and AUROC ensure a more accurate assessment of machine learning models on imbalanced datasets.
3. **Real-World Impact** – Addressing class imbalance is crucial in high-risk domains like fraud detection, medical diagnosis, and cybersecurity, where misclassification can have severe consequences.
4. **Deep Learning Advancements** – Techniques like GANs, autoencoders, and transfer learning provide promising solutions for generating synthetic data and enhancing model learning.
5. **Need for Adaptive Models** – Static models struggle with evolving data distributions; therefore, scalable and adaptive learning solutions are required for long-term effectiveness.
6. **Ethical AI Deployment** – Ensuring fair representation of all classes is essential for building ethical and unbiased AI systems in real-world applications.
7. **Hybrid Approaches for Better Accuracy** – Combining multiple techniques, such as cost-sensitive learning with deep learning-based synthetic data generation, leads to more robust models.
8. **Continuous Evaluation and Improvement** – Regular assessment and updating of imbalance-handling models are essential to maintaining accuracy as datasets evolve over time.

Future Scope of Study:

1. **Development of Automated Solutions** – Future research should focus on creating automated and scalable solutions for handling class imbalance across different industries.
2. **Integration of Reinforcement Learning** – Exploring reinforcement learning-based approaches can improve adaptability and decision-making in imbalanced datasets.
3. **Enhanced Synthetic Data Generation** – Advancements in deep learning, such as improved GAN architectures, can create more realistic synthetic samples for minority classes.
4. **Real-Time Monitoring and Adaptive Learning** – Implementing real-time performance tracking and adaptive algorithms can help models dynamically adjust to changing data distributions.
5. **Cross-Domain Applicability** – Ensuring that imbalance-handling techniques generalize well across multiple domains, including healthcare, finance, and cybersecurity.
6. **Ethical and Fair AI Systems** – Future studies should focus on reducing bias in AI models and ensuring fairness, particularly in applications affecting human lives and decision-making processes.
7. **Exploration of Meta-Learning Techniques** – Leveraging meta-learning for improved generalization in imbalanced learning problems can lead to more efficient and adaptive models.
8. **Scalability in Big Data Environments** – Investigating how class imbalance solutions perform in large-scale datasets, ensuring that techniques remain efficient as data grows exponentially.

REFERENCES

Sample Website References:

GeeksforGeeks - Handling Imbalanced Data for Classification.

Google for Developers - Imbalanced datasets.

Kaggle - Best techniques and metrics for Imbalanced Dataset.

Online Platforms:

Sample – Reference Website

ChatGPT – Used for coding.

Colab – Used for development.

APPENDIX

```
import pandas as pd
data = pd.read_csv("/content/Insurance claims data.csv")
print(data.head())
data.info()
data.isnull().sum()
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style("whitegrid")
#plot the distribution of the target variable 'claim_status'
plt.figure(figsize=(8,5))
sns.countplot(x='claim_status',data=data)
plt.title('Distribution of Claim Status')
plt.xlabel('claim status')
plt.ylabel('count')
plt.show()
#selecting numerical columns for analysis
numerical_columns = ['subscription_length', 'vehicle_age','customer_age']
#plotting distributions of numerical features
plt.figure(figsize=(15,5))
for i,column in enumerate(numerical_columns):
  plt.subplot(1, 3, i+1)
  sns.histplot(data[column], bins=30, kde=True)
  plt.title(f'distribution of {column}')
plt.tight_layout()
plt.show()
# selecting some relevant categorical columns for analysis
categorical_columns = ['region_code', 'segment', 'fuel_type']

# plotting distributions of categorical features
plt.figure(figsize=(15, 10))
for i, column in enumerate(categorical_columns, 1):
    plt.subplot(3, 1, i)
    sns.countplot(y=column, data=data, order=data[column].value_counts().index)
    plt.title(f'Distribution of {column}')
    plt.xlabel('Count')
    plt.ylabel(column)
    plt.tight_layout()
    plt.show()
from sklearn.utils import resample
# separate majority and minority classes
majority = data[data.claim_status == 0]
minority = data[data.claim_status == 1]

# oversample the minority class
minority_oversampled = resample(minority, replace=True, n_samples=len(majority),
random_state=42)
# combine majority class with oversampled minority class
oversampled_data = pd.concat([majority, minority_oversampled])
# check the distribution of undersampled and oversampled datasets
oversampled_distribution = oversampled_data.claim_status.value_counts()
oversampled_distribution
# plotting the distribution of 'customer_age', 'vehicle_age', and 'subscription_length' with respect
to 'claim_status'
plt.figure(figsize=(15, 5))
# 'customer_age' distribution
plt.subplot(1, 3, 1)
sns.histplot(data=oversampled_data, x='customer_age', hue='claim_status', element='step',
bins=30)
plt.title('Customer Age Distribution')
# 'vehicle_age' distribution
plt.subplot(1, 3, 2)
sns.histplot(data=oversampled_data, x='vehicle_age', hue='claim_status', element='step', bins=30)
plt.title('Vehicle Age Distribution')
# 'subscription_length' distribution
plt.subplot(1, 3, 3)
sns.histplot(data=oversampled_data, x='subscription_length', hue='claim_status', element='step',
bins=30)
```

```
plt.title('Subscription Length Distribution')
plt.tight_layout()
plt.show()
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder

# encode categorical variables
le = LabelEncoder()
encoded_data = data.apply(lambda col: le.fit_transform(col) if col.dtype == 'object' else col)

# separate features and target variable
X = encoded_data.drop('claim_status', axis=1)
y = encoded_data['claim_status']

# create a random forest classifier model
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X, y)

# get feature importance
feature_importance = rf_model.feature_importances_

# create a dataframe for visualization of feature importance
features_df = pd.DataFrame({'Feature': X.columns, 'Importance': feature_importance})
features_df = features_df.sort_values(by='Importance', ascending=False)

print(features_df.head(10))  # displaying the top 10 important features
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score
from sklearn.ensemble import RandomForestClassifier
# drop 'Policy_id' column from the data
oversampled_data = oversampled_data.drop('policy_id', axis=1)
```

```
# prepare the oversampled data
X_oversampled = oversampled_data.drop('claim_status', axis=1)
y_oversampled = oversampled_data['claim_status']
# encoding categorical columns
X_oversampled_encoded = X_oversampled.apply(lambda col: LabelEncoder().fit_transform(col)
if col.dtype == 'object' else col)
# splitting the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_oversampled_encoded, y_oversampled,
test_size=0.3, random_state=42)
# create and train the Random Forest model
rf_model_oversampled = RandomForestClassifier(random_state=42)
rf_model_oversampled.fit(X_train, y_train)
# predictions
y_pred = rf_model_oversampled.predict(X_test)
print(classification_report(y_test, y_pred))
original_encoded = data.drop('policy_id', axis=1).copy()
encoders = {col: LabelEncoder().fit(X_oversampled[col]) for col in
X_oversampled.select_dtypes(include=['object']).columns}

for col in original_encoded.select_dtypes(include=['object']).columns:
    if col in encoders:
        original_encoded[col] = encoders[col].transform(original_encoded[col])

original_encoded_predictions =
rf_model_oversampled.predict(original_encoded.drop('claim_status', axis=1))
comparison_df = pd.DataFrame({
    'Actual': original_encoded['claim_status'],

'Predicted': original_encoded_predictions
})
print(comparison_df.head(10))
correctly_classified = (comparison_df['Actual'] == comparison_df['Predicted']).sum()
incorrectly_classified = (comparison_df['Actual'] != comparison_df['Predicted']).sum()
classification_counts = [correctly_classified, incorrectly_classified]
```

```
labels = ['Correctly Classified', 'Misclassified']
# create a pie chart
plt.figure(figsize=(8, 8))
plt.pie(classification_counts, labels=labels, autopct='%1.1f%%', startangle=140,
colors=['#4CAF50', '#FF5733'])
plt.title('Classification Accuracy')
plt.show()
```