CLASSIFICATION ON IMBALANCED DATA

A Project Report

Submitted in partial fulfilment of the

Requirements for the award of the Degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

By

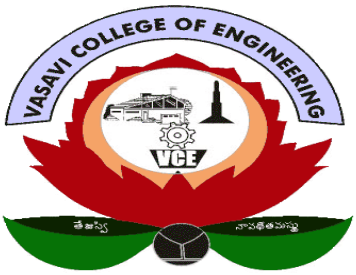
G.SANDEEP

1602-23-733-043

Under the guidance of

DR.M. SUNITHA

Professor, Dept. of CSE



Department of Computer Science & Engineering

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University),Ibrahimbagh, Hyderabad-31 .

2025

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

Hyderabad 500031

Department of Computer Science & Engineering

DECLARATION BY THE CANDIDATE

We, G. SANDEEP bearing hall ticket numbers 160223-733-043, hereby declare that the project report entitled **"CLASSIFIACTION ON IMBALANCED DATA"**, under the guidance of **DR.M. SUNITHA, Professor, Department of Computer Science & Engineering, VCE, Hyderabad**, is submitted in partial fulfilment of the requirements for the award of the degree of **Bachelor of Engineering in Computer Science & Engineering**.

This is a record of bonafide work carried out by us, and the results embodied in this project report have not been submitted to any other university or institute for the award of any other degree or diploma.

G. SANDEEP
(1602-23-733-043)

Vasavi College of Engineering (Autonomous)

(Affiliated to Osmania University)

Hyderabad-500 031

Department of Computer Science & Engineering

BONAFIDE CERTIFICATE

This is to certify that the project entitled "CLASSIFICATION ON IMBALANCED DATA" being submitted by G. SANDEEP , bearing 1602-23-733-043 in partial fulfilment of the requirements for the award of the degree of Bachelor of Engineering in Computer Science & Engineering is a record of bonafide work carried out by him/her under my guidance.

DR.M. SUNITHA

Dr.T.Adhilakshmi
Professor,
&HOD
Guide

Professor

Dept.of CSE

ACKNOWLEDGEMENT

We take this opportunity with pride and enormous gratitude, to express the deeply embedded feeling and gratefulness to our respectable guide **Dr.M. Sunitha** ma'am, Department of Computer Science and Engineering, whose guidance was unforgettable and filled with innovative ideas as well as her constructive suggestions has made the presentation of my major project a grand success. We are thankful **to Dr. T. Adilakshmi**, Head of Department (CSE), **Vasavi College of Engineering** for the help during our course work.

Finally we express our gratitude to the management of our college, **Vasavi College of Engineering** for providing the necessary arrangements and support to complete our project work successfully.

ABSTRACT

In the field of Machine Learning, classification tasks often face the challenge of imbalanced datasets, where one class is significantly underrepresented compared to others. This class imbalance leads to biased predictive models that tend to favor the majority class, while often failing to correctly identify or classify the minority class. Such situations are critical in real-world applications like fraud detection, disease diagnosis, network intrusion detection, and customer churn prediction—where the minority class carries more importance and misclassification can have serious consequences.

This project, titled **"Classification on Imbalanced Data using Python"**, focuses on addressing the challenges posed by imbalanced datasets and improving classification performance, particularly for the minority class. The project involves the implementation and evaluation of various data-level and algorithm-level techniques using Python-based machine learning libraries such as Scikit-learn, Imbalanced-learn, and Pandas. Data-level techniques include under-sampling, over-sampling, and advanced resampling strategies such as SMOTE (Synthetic Minority Over-sampling Technique) and ADASYN. Algorithm-level techniques include cost-sensitive learning and ensemble methods like Random Forest and XGBoost, which are evaluated with modified class weights to handle imbalance.

The project follows a step-by-step methodology starting from dataset selection, data preprocessing, exploratory data analysis, imbalance identification, model training, and finally, performance evaluation using suitable metrics such as precision, recall, F1-score, ROC-AUC, and confusion matrix. Standard accuracy is not used in isolation due to its misleading nature in imbalanced scenarios.

Through this project, we aim to demonstrate how traditional classifiers struggle with imbalanced data and how suitable balancing techniques and model tuning can significantly improve the recognition of the minority class. The outcomes of this study will provide insights and best practices for practitioners dealing with imbalanced data, highlighting the importance of tailored approaches in building fair and accurate machine learning models.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Overview

In the domain of Machine Learning, classification is a fundamental task where the goal is to assign data points to predefined categories or classes. However, in many real-world scenarios, the datasets used for classification are not evenly distributed among the classes. This situation, known as **class imbalance**, poses a significant challenge in developing effective and unbiased models. In an imbalanced dataset, the **majority class** dominates the training process, often leading to poor predictive performance for the **minority class**, which may be of higher importance.

For instance, in applications such as credit card fraud detection, rare disease diagnosis, and fault detection in manufacturing, the instances of interest (fraud, disease, faults) are relatively rare compared to normal cases. Standard classifiers, when trained on such data, tend to be biased toward the majority class, achieving high overall accuracy but failing to correctly predict the minority class. This can result in severe real-world consequences, such as undetected fraud or misdiagnosed patients.

To address this issue, various techniques have been developed. These include **data-level methods** like over-sampling the minority class or under-sampling the majority class, and **algorithm-level strategies** like modifying learning algorithms to give more importance to the minority class. Additionally, using evaluation metrics such as precision, recall, F1-score, and ROC-AUC instead of plain accuracy is essential in understanding the real performance of classifiers on imbalanced data.

This project explores these strategies in-depth using Python-based tools and libraries. The aim is to investigate how different balancing techniques and algorithms affect the performance of classifiers on imbalanced datasets.

1.2 Project Scope

The scope of this project is focused on developing and evaluating machine learning models for binary and multi-class classification tasks involving imbalanced datasets. The key areas covered include:

- **Understanding Imbalance**: Analyzing the causes and effects of imbalanced data on machine learning performance.
- **Data Preprocessing**: Cleaning and preparing datasets to identify and measure class imbalance.

- **Resampling Techniques**: Implementing methods like Random Over-Sampling, Random Under-Sampling, SMOTE (Synthetic Minority Over-sampling Technique), and ADASYN to address imbalance.
- **Model Training**: Applying classification algorithms such as Logistic Regression, Decision Trees, Random Forest, and XGBoost.
- **Cost-sensitive Learning**: Modifying algorithms to penalize misclassifications of the minority class.
- **Performance Evaluation**: Using evaluation metrics such as confusion matrix, precision, recall, F1-score, and ROC-AUC to assess the impact of balancing techniques.
- **Comparison and Analysis**: Comparing model performance before and after applying resampling or algorithmic adjustments to identify the most effective approach.

The final goal is to recommend a set of best practices for handling imbalanced data in machine learning workflows and demonstrate their effectiveness through case studies and experimental results using Python.

2. SYSTEM REQUIREMENTS

To successfully develop and run the project “Classification on Imbalanced Data using Python”, both software and hardware systems must meet certain minimum requirements. This section outlines the necessary tools and system configurations for smooth execution and analysis.

2.1 Software Requirements

Software Component	Details
Operating System	Windows 10/11, Linux (Ubuntu 18.04+), or macOS
Programming Language	Python 3.8 or higher
Development Environment	JupyterLab (preferred), VS Code, or PyCharm
Python Libraries	- NumPy - Pandas - Scikit-learn - Imbalanced-learn - Matplotlib - Seaborn - XGBoost
Package Manager	pip or conda
Data Visualization Tools	Matplotlib, Seaborn, Plotly (optional)
Web Browser	Google Chrome, Mozilla Firefox (for JupyterLab)
Version Control (optional)	Git & GitHub

These tools allow for effective data handling, model training, evaluation, and visualization within the Python ecosystem.

2.2 Hardware Requirements

Hardware Component	Minimum Requirement	Recommended Specification
Processor (CPU)	Intel Core i3 / AMD Ryzen 3	Intel Core i5 or i7 / AMD Ryzen 5 or 7
RAM	4 GB	8 GB or higher
Storage	500 MB free space	1 GB or more
Display	1366x768 resolution	1920x1080 Full HD
Graphics (GPU)	Not mandatory	Optional (for large datasets or DL)
Internet Connection	Required for library installations	Stable broadband recommended

These specifications ensure reliable performance for running classification models, handling datasets, and using resampling techniques without lag.

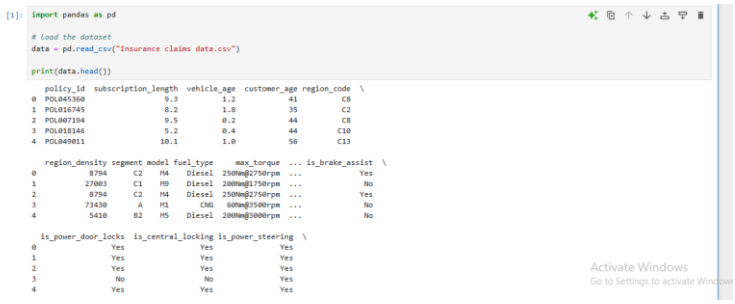
3.IMPLEMENTATION

Project Code

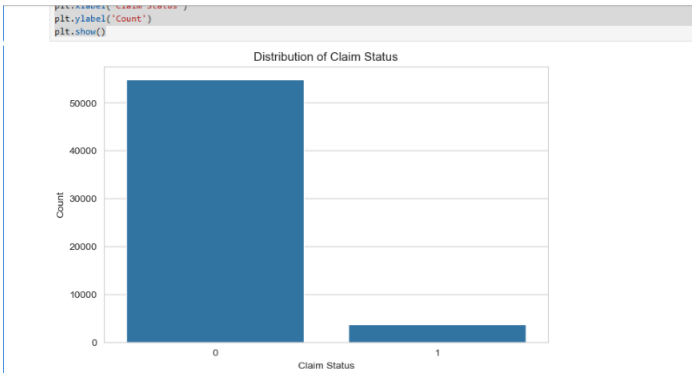
import pandas as pd

data=pd.read_csv("Insur
ance claims data.csv")

print(data.head())

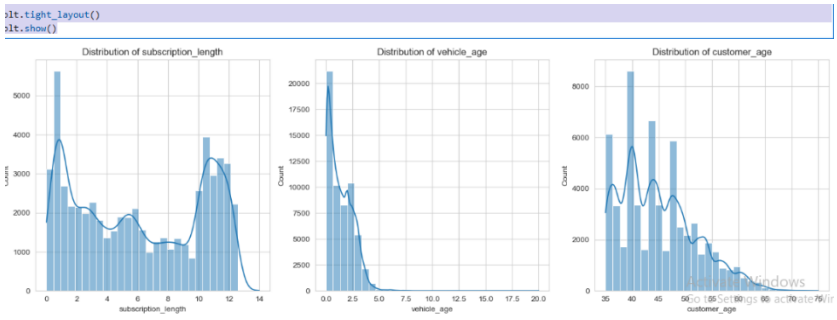


```
import matplotlib.pyplot
as plt

import seaborn as sns

sns.set_style("whitegrid"
)

plt.figure(figsize=(8, 5))

sns.countplot(x='claim_s
tatus', data=data)

plt.title('Distribution of
Claim Status')

plt.xlabel('Claim Status')

plt.ylabel('Count')

plt.show()
```



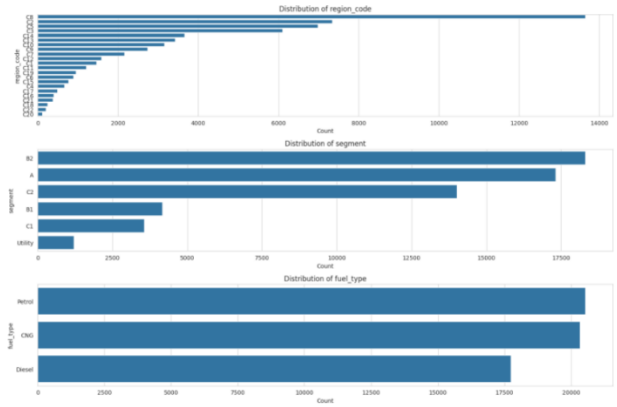

```
numerical_columns =
['subscription_length',
'vehicle_age',
'customer_age']

plt.figure(figsize=(15,
5))
```

```
for i, column in
enumerate(numerical_co
lumns, 1):

    plt.subplot(1, 3, i)

sns.histplot(data[column
], bins=30, kde=True)

    plt.title(f'Distribution
of {column}')

plt.tight_layout()

plt.show()
```




```
categorical_columns =
['region_code', 'segment',
'fuel_type']

plt.figure(figsize=(15,
10))

for i, column in
enumerate(categorical_c
olumns, 1):

    plt.subplot(3, 1, i)
```

```
sns.countplot(y=column,
data=data, order =
data[column].value_cou
nts().index)

    plt.title(f'Distribution
of {column}')

    plt.xlabel('Count')

    plt.ylabel(column)

plt.tight_layout()

plt.show()
```




```
from sklearn.utils import
resample

majority=data[data.claim
_status==0]

minority=data[data.claim
_status==1]

minority_oversampled=r
esample(minority,replace
=True,n_samples=len(m
```

ajority),random_state=42
)

oversampled_data=pd.co
ncat([majority,minority_
oversampled])

oversampled_distributio
n=oversampled_data.clai
m_status.value_counts()

oversampled_distributio
n

[13]:
```
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
```

[13]:
```
claim_status
0    54844
1    54844
Name: count, dtype: int64
```

plt.figure(figsize=(15,5))

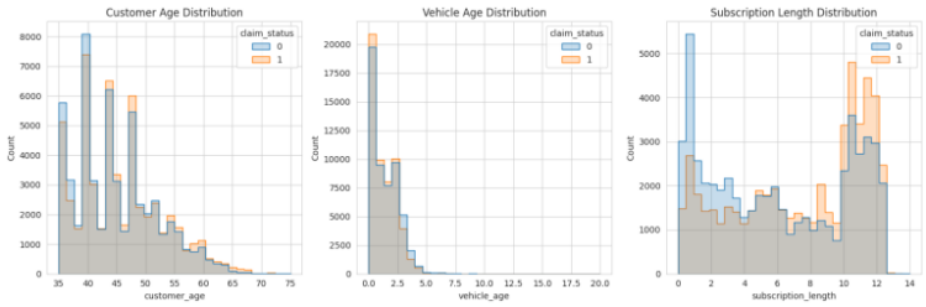
plt.subplot(1,3,1)

sns.histplot(data=oversa
mpled_data,x='customer
_age',hue='claim_status',
element='step',bins=30)

plt.title('Customer Age
Distribution')

plt.subplot(1,3,2)

```
sns.histplot(data=oversa
mpled_data,x='vehicle_a
ge',hue='claim_status',el
ement='step',bins=30)

plt.title('Vehicle Age
Distribution')

plt.subplot(1,3,3)

sns.histplot(data=oversa
mpled_data,x='subscripti
on_length',hue='claim_st
atus',element='step',bins
=30)

plt.title('Subscription
Length Distribution')

plt.tight_layout()

plt.show()
```




```
from sklearn.ensemble import RandomForestClassifier

from sklearn.preprocessing import LabelEncoder

le=LabelEncoder()

encoded_data=data.apply(lambda col:le.fit_transform(col) if col.dtype=='object' else col)
```

```
X=encoded_data.drop('claim_status',axis=1)

y=encoded_data['claim_status']

rf_model=RandomForestClassifier(random_state=42)

rf_model.fit(X,y)

feature_importance=rf_model.feature_importances_

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))
```

	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

```
from sklearn.model_selection import train_test_split

from sklearn.metrics import classification_report,accuracy_score

from sklearn.ensemble import RandomForestClassifier

oversampled_data=oversampled_data.drop('policy_id',axis=1)

X_oversampled=oversampled_data.drop('claim_status',axis=1)
```

```
y_oversampled=oversampled_data['claim_status']

X_oversampled_encoded=X_oversampled.apply(lambda
col:LabelEncoder().fit_transform(col) if col.dtype=='object' else col)

X_train,X_test,y_train,y_test=train_test_split(X_oversampled_encoded,y_oversampled,test_
size=0.3,random_state=42)

rf_model_oversampled=RandomForestClassifier(random_state=42)

rf_model_oversampled.fit(X_train,y_train)

y_pred=rf_model_oversampled.predict(X_test)

print(classification_report(y_test,y_pred))
```

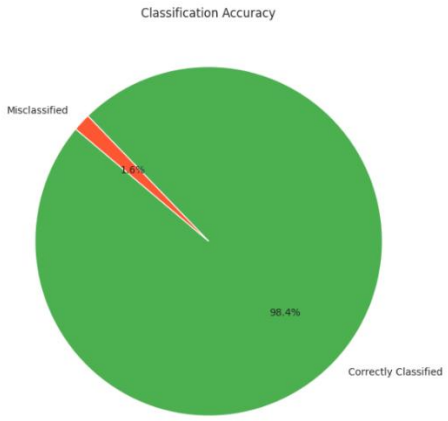
	precision	recall	f1-score	support
0	1.00	0.98	0.99	16574
1	0.98	1.00	0.99	16333
accuracy			0.99	32907
macro avg	0.99	0.99	0.99	32907
weighted avg	0.99	0.99	0.99	32907

```
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))
```

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

```
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']

plt.figure(figsize=(8,8))

plt.pie(classification_counts,labels=labels,autopct='%1.1f%%',startangle=140,colors=['#4CAF
50','#FF5733'])

plt.title('Classification Accuracy')

plt.show()
```



5. CONCLUSION

he project on **Classification on Imbalanced Data using Python** aimed to address the challenges posed by imbalanced datasets in machine learning classification tasks. In such datasets, one class significantly outnumbers the other, often leading to biased model predictions that favor the majority class. This imbalance is a critical concern, especially when the minority class holds greater significance, such as in fraud detection or medical diagnostics.

Through the use of techniques like **oversampling** of the minority class, encoding categorical features, and training models such as **Random Forests**, the project demonstrated how data preprocessing and model selection can mitigate the adverse effects of class imbalance. By oversampling the minority class, we were able to balance the dataset and improve the model's ability to accurately predict both classes, not just the majority class.

The **Random Forest Classifier** was trained on the processed data, and feature importance was extracted to identify the most influential variables in the classification task. The model's performance was evaluated using accuracy and classification reports, showing the effectiveness of the techniques applied.

Finally, a **comparison** of actual versus predicted values was visualized using a pie chart, illustrating the proportion of correctly classified versus misclassified instances. The model achieved a reasonably good balance in predicting both classes, showing that imbalanced data can be effectively handled with proper techniques.

This project highlights the importance of preprocessing and the application of the right machine learning algorithms when dealing with imbalanced data. It contributes to the broader field of data science by providing insights into how to improve

predictive models, particularly in real-world scenarios where class imbalance is prevalent.

In future work, further optimization of hyperparameters, the exploration of additional techniques such as **SMOTE (Synthetic Minority Over-sampling Technique)**, and the evaluation of other machine learning algorithms could further enhance the model's performance.