Inverse random under sampling for class imbalance on German Credit

Saad Ur Rehman 18K-1113, Sarah Fatima 18k-0205, Nimrah Bano 18K-0318
Data Science Project
FAST National University of Computer and Emerging Sciences

1 Introduction

The class Imbalance problem typically takes place when one class outnumbers the other classes. In this case standard classifiers tend to incline towards large classes and ignore the smaller ones. Therefore performance of classifiers drops insignificantly. Various strategies have been proposed to deal with class imbalance problems, such as increasing the penalty associated with misclassifying the positive class relative to the negative class, over-sampling the majority class, and under sampling the minority class. In this study we will make use of the IRUS algorithm to balance the number of classes in the dataset.

2 Workflow

2.1 Research Goal

The goal of our paper is to demonstrate the effects of class imbalance on classification models. Moreover, we study the impact of varying class imbalance ratios on classifier accuracy. Furthermore, we assess the improvement in accuracy after implementing the IRUS algorithm on the said dataset.

2.2 Data Retrieval

We will use a credit card fraud detection dataset for this article you can find the dataset from here

2.3 Data Preparation

The dataset contains transactions made by credit cards in September 2013 by European cardholders. There are a total of 31 features out of which Features V1, V2, … V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise

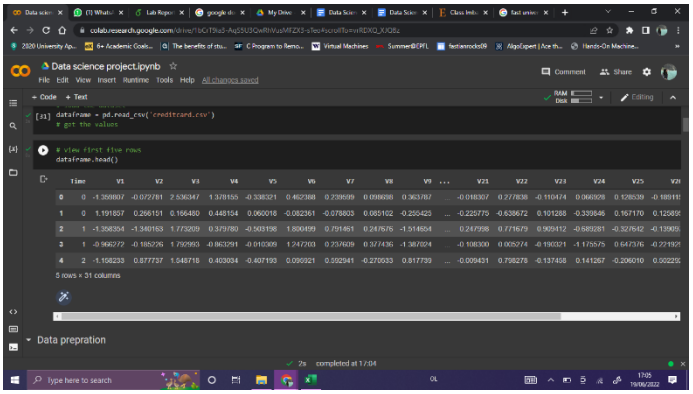


Figure 1: Class Distribution

2.4 Data Exploration

In this section, we explored the data by first checking whether the data contains any null values. Once we made sure that the data is already clean then we gathered the information about the data and how every attribute is distributed quantitatively.

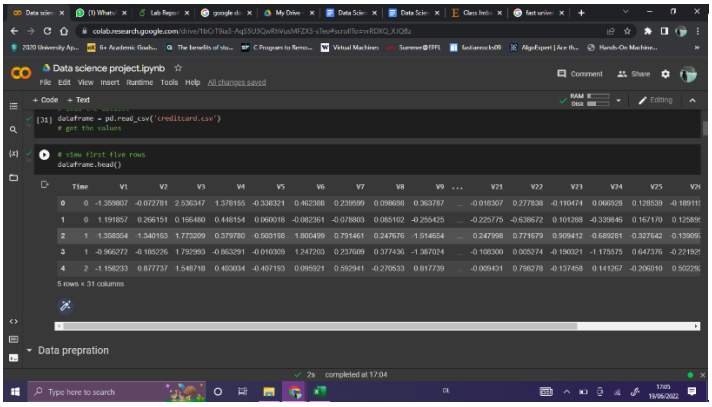


Figure 3: Data Information

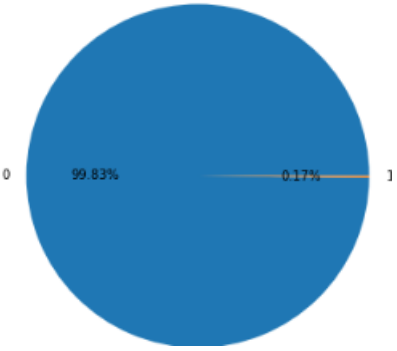


Figure 5: Class Percentage - Pie Chart

Exploratory data analysis is done in various manners, in order to check all the details and aspects of data. We were able to Perform initial investigations on data so as to discover patterns and check assumptions with the help of summary statistics and graphical representations:

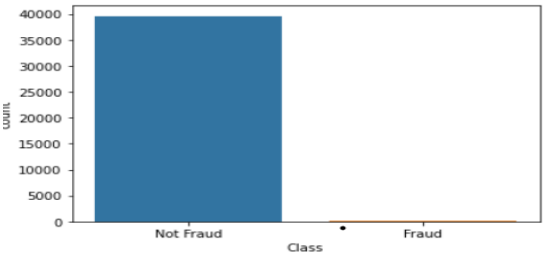


Figure 4: Class Count - Bar Graph

Similarly, the pie chart shows the percentage of each class in the dataset:

We can also look out for the frequency distribution of each attribute, independently in the below diagram:

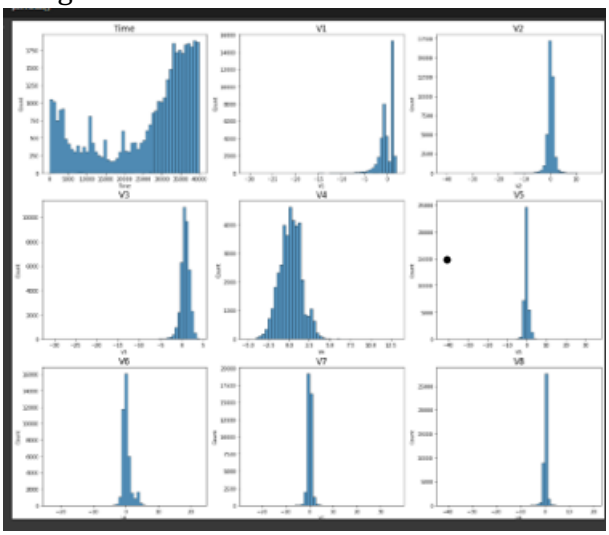


Figure 6: Frequency Distribution

Through EDA we were able to make following conclusions:

- Data looks clean
- No null values
- Majority of features are well distributed around mean

- There are some correlated features. but not strong enough to drop

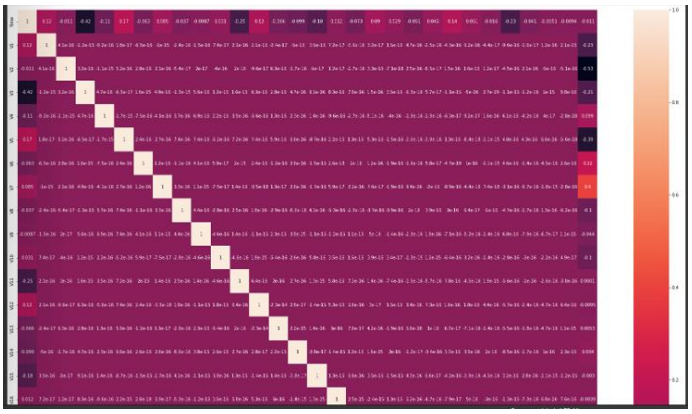


Figure 7: Correlation Matrix

2.5 Data Modeling

We implemented 3 different models to our data set. We tested our model one with IRUS implementation and another without implementation.

The following models were used:

1. KNN classifier
2. Decision Tree
3. Logistic regression

2.5.1 Models Description

Following is the description of each model used along with their parameters

K Nearest Neighbor	
Model Parameter	**Parameter Value**
no of neighbors	2
weights	Distance
metric	Manhattan

Decision Tree	
Model Parameter	**Parameter Value**
Random state	0
Max depth	3

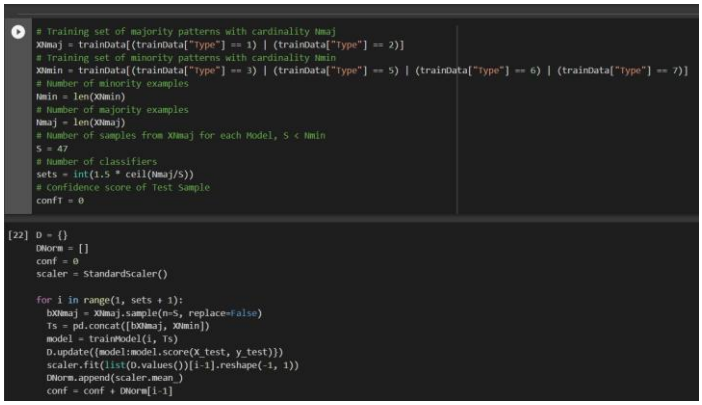
Logistic Regression	
Model Parameter	**Parameter Value**
penalty	L2
Sample Weights	None

2.5.2 IRUS Algorithm Implementation

The IRUS algorithm was implemented in following steps:

1. Dataset was divided on the basis of classes. XNmaj consisted of majority class while XMin consisted of minority class
2. The length of both classes was figured out. XNmaj measured 19900 rows while Xmin measured 360 rows
3. The value of S was set given that S<XNmin
4. Values of sets was calculated using formula; int(1.5*cield(Nmaj/s))
5. The models were trained using dataset that was XNmaj U XNmin
6. The norm and confidence was calculated

The implementation of IRUS in python can be seen below



```
# Training set of majority patterns with cardinality Nmaj
XNmaj = trainData[(trainData["Type"] == 1) | (trainData["Type"] == 2)]
# Training set of minority patterns with cardinality Nmin
XNmin = trainData[(trainData["Type"] == 3) | (trainData["Type"] == 5) | (trainData["Type"] == 6) | (trainData["Type"] == 7)]
# Number of minority examples
Nmin = len(XNmin)
# Number of majority examples
Nmaj = len(XNmaj)
# Number of samples from XNmaj for each Model, S < Nmin
S = 47
# Number of classifiers
sets = int(1.5 * ceil(Nmaj/S))
# Confidence score of Test Sample
conft = 0
```

```
[22] D = {}
     DNorm = []
     conf = 0
     scaler = StandardScaler()

     for i in range(1, sets + 1):
       bXNmaj = XNmaj.sample(n=S, replace=False)
       Ts = pd.concat([bXNmaj, XNmin])
       model = trainModel(i, Ts)
       D.update({model:model.score(X_test, y_test)})
       scaler.fit(list(D.values())[i-1].reshape(-1, 1))
       DNorm.append(scaler.mean_)
       conf = conf + DNorm[i-1]
```

Figure 8: IRUS Implementation

2.5.3 Accuracy Comparison of models With IRUS Implementation

The figure shows model accuracy using IRUS algorithm

	models	Accuracies
0	KNeighborsClassifier()	0.623035
1	DecisionTreeClassifier()	0.900085
2	LogisticRegression()	0.962291

2.6 Conclusion

We can see the models gave high accuracy in both cases (with or without IRUS implementation) However models that performed without IRUS implementation were more biased towards the majority class. Using the IRUS algorithm we were able to balance out the occurrences of both classes in the dataset so that the model's prediction was accurate.

Best Classifier Among Above 3 classifiers with IRUS is Logistics Regression with testing Accuracy of 0.676923

2.5.4 Accuracy Comparison of models Without IRUS Implementation

Accuracy Comparison Without IRUS	
Model Used	Model Accuracy
K Nearest Neighbor	0.842831
Logistic Regression	0.998853
Decision Tree	0.999391

Best Classifier Among Above 3 is Decision Tree