POLITECNICO
MILANO 1863

Numerical Analysis For Machine Learning
A.Y. 2021-2022

Credit Card Fraud Detection

Authors:
Chizzola Andrea - 10614212
Cominoli Carlo - 10629497

Professor:
Miglio Edie

27 July 2022

Contents

1 Introduction **1**

2 Dataset Description **1**

2.1 Dataset Division 1

3 Classification Algorithms **2**

3.1 K-Nearest Neighbors 2

3.2 Naïve Bayes 2

3.3 Logistic Regression 3

4 Performance Evaluation **4**

4.1 Random Under-Sampling Applied 5

4.2 Random Under-Sampling Not Applied 6

1 Introduction

Detecting fraudulent credit card transactions is perhaps one of the most challenging fields of application for Machine Learning algorithms and techniques. The past decade has seen a huge increase in the number of transactions preformed, especially thanks to the growth of e-commerce platforms and related payment methods. As a direct consequence, it is now of paramount importance for banks and business organizations to be able to correctly authenticate transactions and efficiently detecting the fraudulent ones.

The complexity of the problem is mainly due to the data used to train the classification algorithms as it is very skewed, due to privacy reasons, and it comes in extremely unbalanced datasets, since only a small percentage of the recorded transactions happens to be fraudulent. Many techniques have been developed to deal with these issues, such as Random Under (or Over) Sampling, and allow the definition of a more precise and reliable prediction model.

In the following pages, we decided to study and analyze how some of the most known classification algorithms, namely Naïve Bayes, K-Nearest Neighbors and Logistic Regression, perform when dealing with fraud detection and which techniques could be applied to improve their results.

2 Dataset Description

The dataset selected for the project is the one available on Kaggle[1], which contains the transactions made by European cardholders in September 2013. The features provided, due to confidentiality reasons, are the result of the PCA (Principal Component Analysis) applied to the original data. Since the only features that were not transformed by the PCA were 'Time' (containing the seconds elapsed between each transaction and the first transaction in the dataset) and 'Amount' (indicating the transaction amount), we decided to standardize them. The feature 'Class' is instead responsible for discriminating among valid (0) and fraudulent transactions (1).

2.1 Dataset Division

As expected from a dataset containing credit card transactions, we could immediately see how unbalanced it was, as only 0.173% of the total transactions was labeled as fraudulent.

A possible approach to addressing the problem of class imbalance is to apply a resampling technique to rebalance the class distribution. The method we used for resampling our dataset is called "Random Under-Sampling" (RUS), which involves randomly selecting samples from the majority class (valid transactions in our case) and removing them from the dataset until a more balanced distribution is reached. Not knowing which 'valid to fraudulent' proportion would be more effective, we decided to try three different proportions ('50:50', '66:34' and '75:25') to see how they would affect the training procedure. To better evaluate the effectiveness of

[1]Dataset: https://www.kaggle.com/mlg-ulb/creditcardfraud

the application of RUS, we also decided to compare the obtained results with the ones given by the same dataset but without its application, which better replicate real-world data.

It's to be noted that one of the main disadvantages of RUS is the lost of vast quantities of data, which can make the boundary between minority and majority classes harder to learn, resulting in a worst classification performance.

3 Classification Algorithms

Once the initial dataset has been pre-processed through the under-sampling technique, the obtained datasets were used to train each one of the supervised classification algorithms: K-Nearest Neighbors, Naïve Bayes and Logistic Regression.

3.1 K-Nearest Neighbors

The KNN algorithm is based on the assumption that similar data points (samples) are close to each other. It works by finding the distances between a testing sample and all the other samples in the dataset. Then the k nearest samples are selected and a majority vote is applied to decide the label of the new sample as the predominant one among the k nearest samples.

In our implementation we decided to use the Euclidean Distance to measure the distance among samples and a default k value of 3.

During the implementation and testing of the algorithm we could find that, since all the samples are used also for performing the classification, KNN has the disadvantage of becoming significantly slower as the dimension of the dataset increases.

3.2 Naïve Bayes

The Naïve Bayes algorithm makes two main assumptions. The first one is that each feature is independent from the others, while the second assumption is that each feature makes an equal contribution to the outcome.

The classification is based on the Bayes Theorem which finds the probability of an event occurring given the probability of another event that has already occurred:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

In our case it can alternatively be seen as:

$$P(y \mid x) = \frac{P(x \mid y)P(y)}{P(x)}$$

where $x = (x_1, ..., x_n)$ is a sample and y is a class (0 or 1). Given the assumption on the independence of each feature, the formula can be then expressed as:

$$P(y \mid x_1, ..., x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i \mid y)}{P(x_1)P(x_2)...P(x_n)}$$

Since the denominator is the same for each entry in the dataset and it does not depend on y, it can be removed and substituted by a proportionality:

$$P(y \mid x_1, ..., x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

y can be obtained as:

$$y = \underset{y}{\operatorname{argmax}}\, P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

However, to avoid the multiplication of very small values, in the actual implementation we decided to apply a logarithmic operator, obtaining as final expression for y:

$$y = \underset{y}{\operatorname{argmax}} \log \left(P(y) + \prod_{i=1}^{n} P(x_i \mid y) \right)$$

Finally, each feature is assumed to be sampled from a Gaussian distribution. Hence each conditional probability is given by:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2} \right)$$

where μ_y is the mean and σ_y^2 is the variance.

3.3 Logistic Regression

Differently from a Linear Regression model, Logistic Regression uses a more complex loss function, the *cross entropy* function:

$$Loss = -\frac{1}{N} \sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)$$

coupled with the *sigmoid* function:

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

which will be used instead as activation function. The parameters used are the weights (in the same number as the number of features) and a bias. We initialized the weights with random normal values and the bias with value zero.

During the training procedure, the predicted outcome is computed, for each sample, by feeding the sigmoid function with the weighted samples. The parameters minimizing the loss function are found by iteratively applying gradient descent.

The obtained parameters are then used to perform the classification of unseen data. The outcome predicted using the sigmoid function is by definition a real number between 0 and 1, but, since the desired outcome is binary (fraud or not fraud), a prediction corresponding to 1 is given when the sigmoid function is greater than 0.5, and 0 when it is less than 0.5.

4 Performance Evaluation

For evaluating the effectiveness of each algorithm, we first computed, for each one of them, the related confusion matrix in order to retrieve the number of True Positives (TP), False Positives (FP), True Negatives (TN) and False Negatives (FN). More specifically, a predicted value is said to be a:

- True Positive if it correctly predicts a fraudulent transaction.
- False Positive if it wrongly predicts a fraudulent transaction.
- True Negative if it correctly predicts a valid transaction.
- False Negative if it wrongly predicts a valid transaction.

Then we used the obtained values to calculate some of the metrics typically used for classification models, namely:

- Accuracy is the ratio of the number of correct predictions to the total number of predictions:
$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$
- Sensitivity (or Recall) is the ratio of true positives to all the positives in the dataset:
$$Sensitivity = \frac{TP}{TP + FN}$$
- Specificity is the ratio of the correct negative predictions to the total number of negatives:
$$Specificity = \frac{TN}{TN + FP}$$
- Precision is the ratio of the correct positive predictions to the total number of positive predictions:
$$Precision = \frac{TP}{TP + FP}$$
- F-measure is the harmonic mean of Precision and Sensitivity:
$$F - measure = 2 \cdot \frac{Precision \cdot Sensitivity}{Precision + Sensitivity}$$
- Area Under Curve (AUC) is the measure of the ability of a classifier to distinguish between classes:
$$AUC = \frac{Sensitivity + Specificity}{2}$$

4.1 Random Under-Sampling Applied

Firstly, we decided to test our classification models applying the RUS technique to the dataset.

The results reported in Figure 1 and Figure 2 show how, in terms of accuracy, all algorithms perform better when trained with the 75:25 dataset, while considering precision, the 50:50 training dataset seems to be more effective. For the AUC value, all three splits have a similar effect on the outcome, showing a good ability in identifying the correct class. Except for Naïve Bayes, the other algorithms seem to have a better sensitivity when trained with the 50:50 dataset, while, regarding specificity, KNN and Logistic Regression show good results when using the 75:25 training dataset.

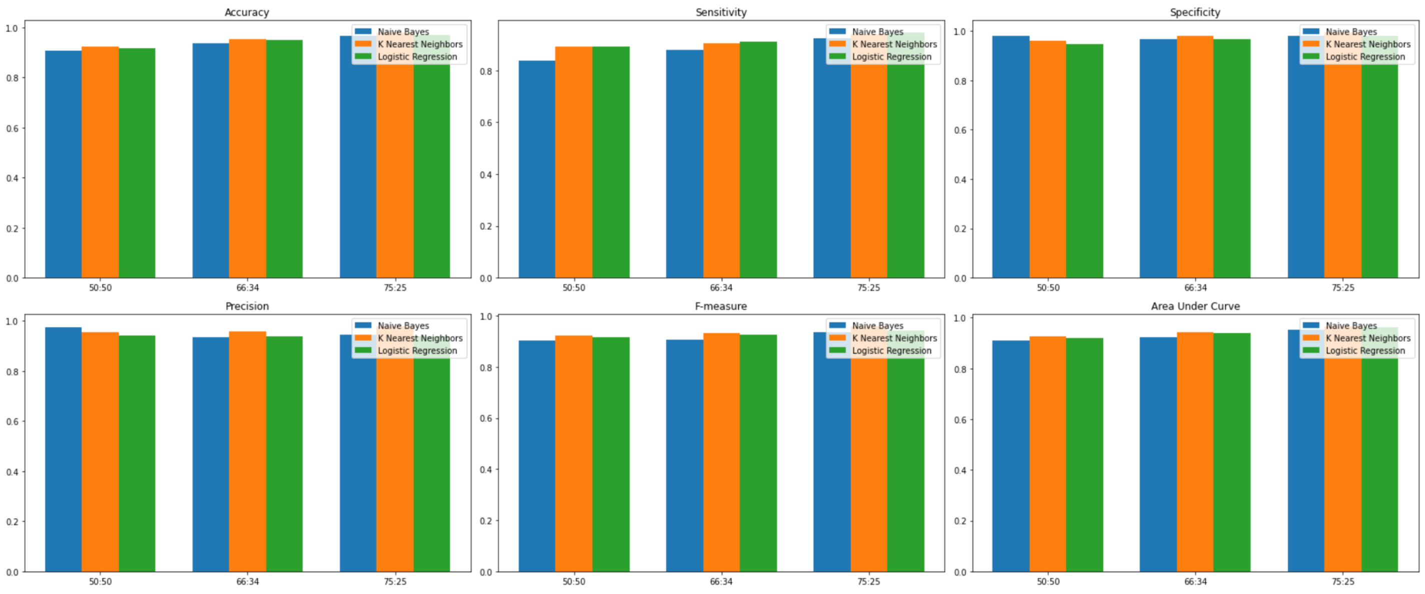
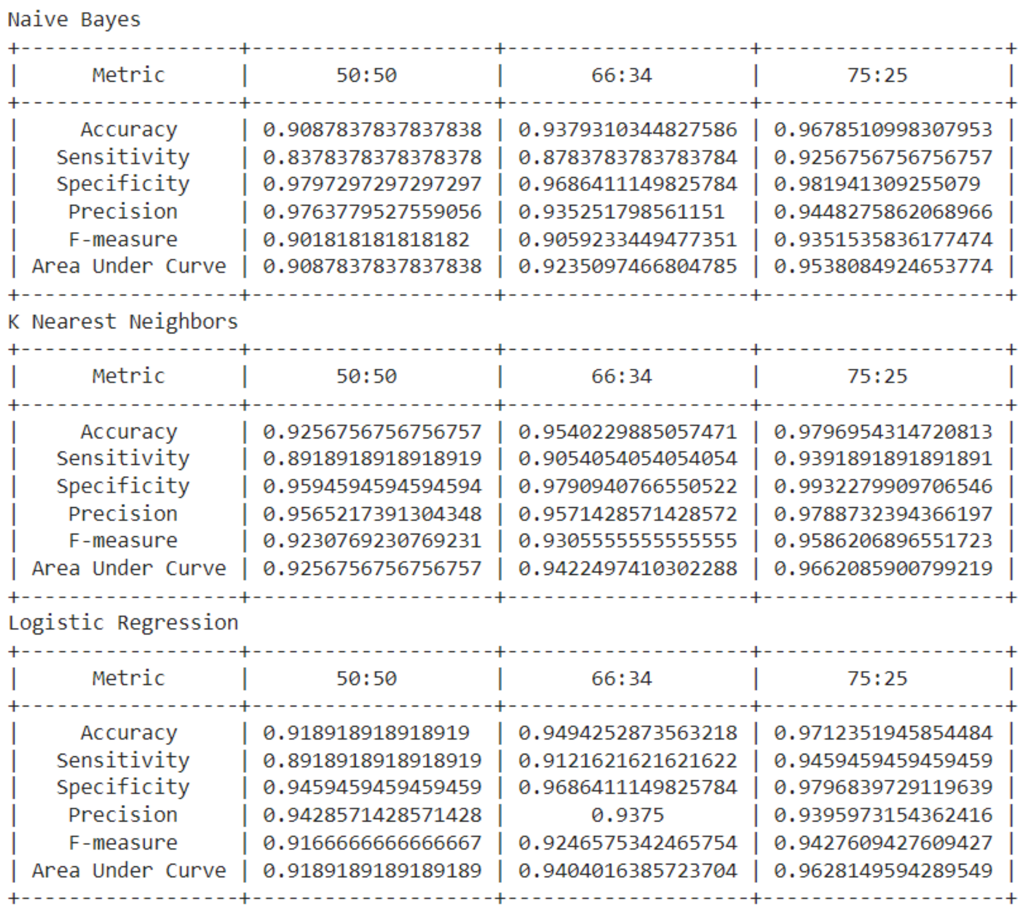


Figure 1: Using random under-sampling



```
Naive Bayes
+-------------------+--------------------+--------------------+--------------------+
|       Metric      |       50:50        |       66:34        |       75:25        |
+-------------------+--------------------+--------------------+--------------------+
|      Accuracy     | 0.9087837837837838 | 0.9379310344827586 | 0.9678510998307953 |
|    Sensitivity    | 0.8378378378378378 | 0.8783783783783784 | 0.9256756756756757 |
|    Specificity    | 0.9797297297297297 | 0.9686411149825784 | 0.981941309255079  |
|     Precision     | 0.9763779527559056 | 0.935251798561151  | 0.9448275862068966 |
|     F-measure     | 0.901818181818182  | 0.9059233449477351 | 0.9351535836177474 |
| Area Under Curve  | 0.9087837837837838 | 0.9235097466804785 | 0.9538084924653774 |
+-------------------+--------------------+--------------------+--------------------+
K Nearest Neighbors
+-------------------+--------------------+--------------------+--------------------+
|       Metric      |       50:50        |       66:34        |       75:25        |
+-------------------+--------------------+--------------------+--------------------+
|      Accuracy     | 0.9256756756756757 | 0.9540229885057471 | 0.9796954314720813 |
|    Sensitivity    | 0.8918918918918919 | 0.9054054054054054 | 0.9391891891891891 |
|    Specificity    | 0.9594594594594594 | 0.9790940766550522 | 0.9932279909706546 |
|     Precision     | 0.9565217391304348 | 0.9571428571428572 | 0.9788732394366197 |
|     F-measure     | 0.9230769230769231 | 0.9305555555555555 | 0.9586206896551723 |
| Area Under Curve  | 0.9256756756756757 | 0.9422497410302288 | 0.9662085900799219 |
+-------------------+--------------------+--------------------+--------------------+
Logistic Regression
+-------------------+--------------------+--------------------+--------------------+
|       Metric      |       50:50        |       66:34        |       75:25        |
+-------------------+--------------------+--------------------+--------------------+
|      Accuracy     | 0.918918918918919  | 0.9494252873563218 | 0.9712351945854484 |
|    Sensitivity    | 0.8918918918918919 | 0.9121621621621622 | 0.9459459459459459 |
|    Specificity    | 0.9459459459459459 | 0.9686411149825784 | 0.9796839729119639 |
|     Precision     | 0.9428571428571428 |       0.9375       | 0.9395973154362416 |
|     F-measure     | 0.9166666666666667 | 0.9246575342465754 | 0.9427609427609427 |
| Area Under Curve  | 0.9189189189189189 | 0.9404016385723704 | 0.9628149594289549 |
+-------------------+--------------------+--------------------+--------------------+
```

Figure 2: Using random under-sampling

4.2 Random Under-Sampling Not Applied

The results reported in Figure 3 and Figure 4 show how, despite having some similar values to the ones previously found, without the application of RUS, the precision and f-measure levels drop drastically especially for Naïve Bayes. Also Logistic Regression faces a decrease in these values, even if to a smaller extent. Another notable difference can be found for the values of sensitivity in KNN and Logistic Regression. A further analysis showed how the precision decrease is mainly due to a high number of False Positives (FP) as, given the strong disproportion between the two classes of transactions, some of the valid ones are considered fraudulent.

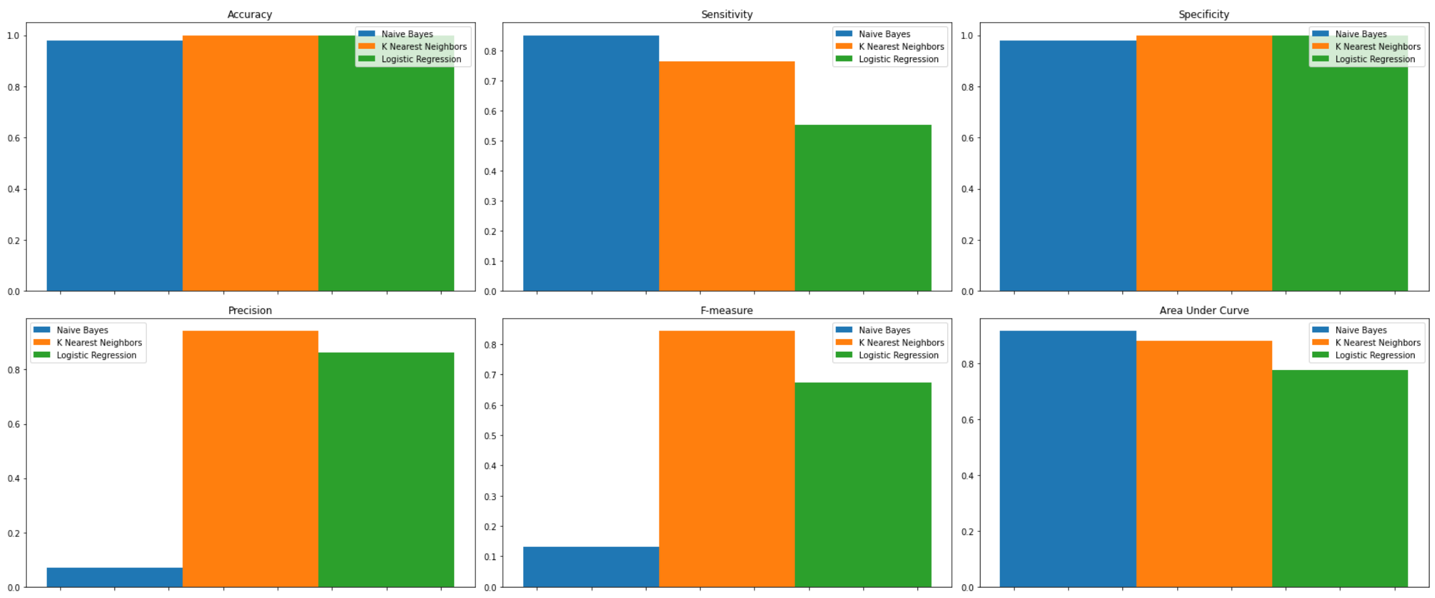
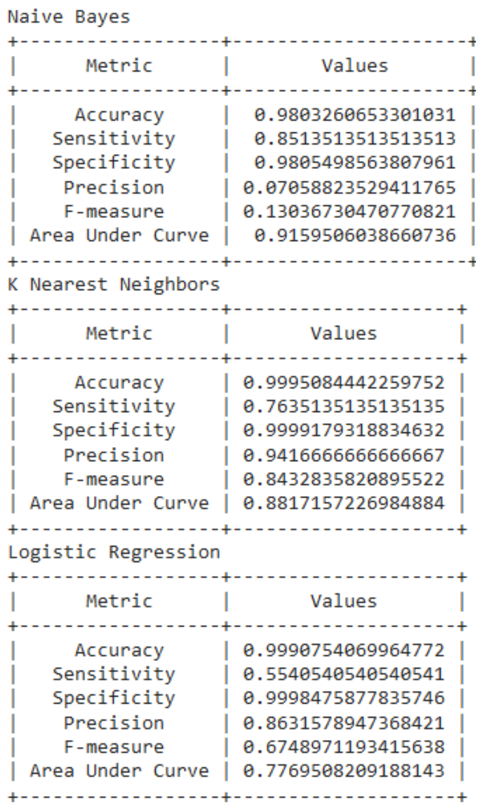


Figure 3: Without under-sampling



```
Naive Bayes
+-----------------+---------------------+
|      Metric     |        Values       |
+-----------------+---------------------+
|     Accuracy    |  0.9803260653301031 |
|   Sensitivity   |  0.8513513513513513 |
|   Specificity   |  0.9805498563807961 |
|    Precision    | 0.07058823529411765 |
|    F-measure    |  0.1303673047077082 1|
| Area Under Curve |  0.9159506038660736 |
+-----------------+---------------------+
K Nearest Neighbors
+-----------------+--------------------+
|      Metric     |       Values       |
+-----------------+--------------------+
|     Accuracy    | 0.9995084442259752 |
|   Sensitivity   | 0.7635135135135135 |
|   Specificity   | 0.9999179318834632 |
|    Precision    | 0.9416666666666667 |
|    F-measure    | 0.8432835820895522 |
| Area Under Curve | 0.8817157226984884 |
+-----------------+--------------------+
Logistic Regression
+-----------------+--------------------+
|      Metric     |       Values       |
+-----------------+--------------------+
|     Accuracy    | 0.9990754069964772 |
|   Sensitivity   | 0.5540540540540541 |
|   Specificity   | 0.9998475877835746 |
|    Precision    | 0.8631578947368421 |
|    F-measure    | 0.6748971193415638 |
| Area Under Curve | 0.7769508209188143 |
+-----------------+--------------------+
```

Figure 4: Without under-sampling