UNIVERSITÀ CATTOLICA DEL SACRO CUORE
Economics | Mathematics, Physics and Natural Sciences
Data Analytics for Business

Credit card fraud detection through isolation forest

Graduand: Davide Alberto Lupis
Student ID: 5004776

Supervisor: Dr. Francesco Ballarin

Academic Year 2021–2022

Abstract

Credit card fraud detection through isolation forest

Davide Alberto Lupis

In this work, we will follow the steps of different references about Fraud Detection and discuss the application of Isolation Forest. The first part will give a brief introduction to Fraud Detection and the pro and cons of Unsupervised Learning. The succeeding chapters will be focused on the intuition and the mathematical formalization of the algorithm with a final application on simulated transactions using data provided in the Reproducible Machine Learning for Credit Card Fraud Detection Handbook [Le Borgne et al. 2022]. The core discussion will focus on the implementation of isolation forest inside of what is called a Fraud Detection System, by now FDS, with its pros and cons.
Keywords: isolation-forest, unsupervised-learning, anomaly-detection

I acknowledge professor Ballarin for being collaborative and understanding in the development of this work.

Contents

Introduction

As we can see in Figure 1, financial losses caused by credit card fraudulent activities are worth billions and compromise the growth and stability of the economic system.

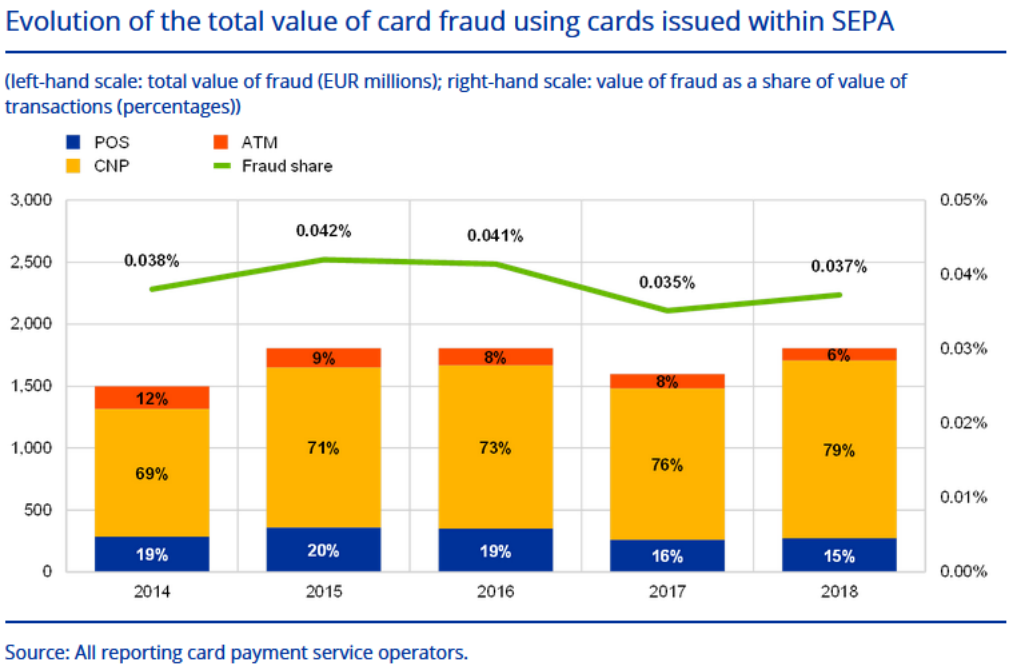

Figure 1: European Central Bank. 6th report on card fraud. August 2020.

The term “fraud” is commonly used for many forms of misconduct but the legal definition of fraud is very specific. In the broadest sense, fraud can encompass any crime for gain that uses deception as a principal modus-operandi [Westphal (2008)]. The closest analytical definition of fraud is the term “anomaly”. An anomaly is an irregular piece of information that, for one reason or another, does not fit with the rest of the data. It follows that the problem is business specific but it can be translated into a statistical problem, upon the right assumptions and limitations. As the economy progresses also frauds start become more complex, arriving to a point where the use of simple rules is simply not enough to prevent them. The set of all qualitative and quantitative analysis to prevent or flag anomalies is defined under the broad term “Fraud Analytics”. Fraud analytics is when the analysis relies on “critical thinking” skills to integrate the output of diverse methodologies into a cohesive and actionable analysis of products. The analysis process requires a combined effort of: knowledge, skills, and abilities [Westphal (2008)]. The use of data driven strategies made fraud analytics more complete and powerful, but the industry still faces many challenges [Baesens et al. (2015)]:

- Concept Drift
  A shift in the operative paradigm that makes the existing techniques not useful or impossible to maintain, a few example are macroeconomic conditions and shift in technology.
- Statistical Accuracy The ability of the model to fit the data, this might decay for several reasons like concept drift or a population change during time.
- Interpretability The ability to derive business conclusion from the outcome
- Operational Efficiency The operational process is considered scaleable and efficient if it feasible to apply for a massive customer base without problems.
- Regulatory Compliance The acceptance and legal permission of the procedure from both final users and policy makers.

Fraud Analytics heavily relies on information and rules for detecting anomalies, the extraction of information is heavily conditioned by the process in use and by the quality and quantity of data collected. Therefore Anomaly detection starts from Data Collection. As the use of databases increased , fraud analytics has become more and more a data driven industry with a strong statistical approach. Many analytical approaches rely on historical data for detecting patterns/behaviours that are often related to frauds. Some examples of analytical techniques are [Baesens et al. (2015)]:

- Rule Based Algorithm Rules based on simple metrics i.e."Number of transactions greater than N in last 24Hours"

- Traditional Statistical Models Generalized linear model, often logistic regressions
- Network Graphs Analytical instrument that is beneficial for studying relations and defining strength between the existing entities in an organization like products, clients, and services.
- Machine Learning/Deep Learning
  - Supervised
  - Unsupersived

In the last decade, machine learning became a valid alternative/addition to traditional models and rule based algorithm, especially for it's flexibility in the processing of data and the rise of efficient frameworks to build models. Among the many methods, supervised machine learning stands out for performance and operational efficiency. The biggest advantage of machine learning models is their flexibility with the processing of data, for example they adapt very well to unstructured information like text,images and it is easier to impute missing values or handle non linear data. Although supervised machine learning is effective, it requires a label [James et al. (2013)]. The label indicates the record is associated with a fraud or not, and this information can be difficult or to obtain since a fraud label is created after a claim or block in the payment process. For this and other reasons, supervised machine learning is often paired with other statistical methods, as well as unsupervised machine learning. The main advantage of unsupervised learning is the absence of a label. At a direct comparison of performance unsupervised learning models are usually less effective than supervised ones, but very useful when combined with other methods. Nature and in depth definition of supervised and unsupervised machine learning will be better explained in the Chapter 1. Chapter 2 will be focused on the formal definition of Isolation Forest, while this section will give an high level view about the intuition behind the method. Isolation Forest [Liu et al. (2008)], by now IF, is an ensemble model. We call ensemble models, a collection of smaller or simple models that jointly contribute to the final prediction. The business intuition is the following: "If anomalies are indeed data with a different behaviour inside the feature space, it is expected to be easier to isolate them from the rest using randomised and independent splits". A more computer scientist definition of IF is the following: IF is built using an ensemble of random trees, and anomalies are points with the shortest number of splits required to isolate them (path length). Once the isolation forest is built, the algorithm can be used to identify outliers by finding instances out of a multidimensional trend. The instances with a high anomaly score are considered as outliers or anomalies.

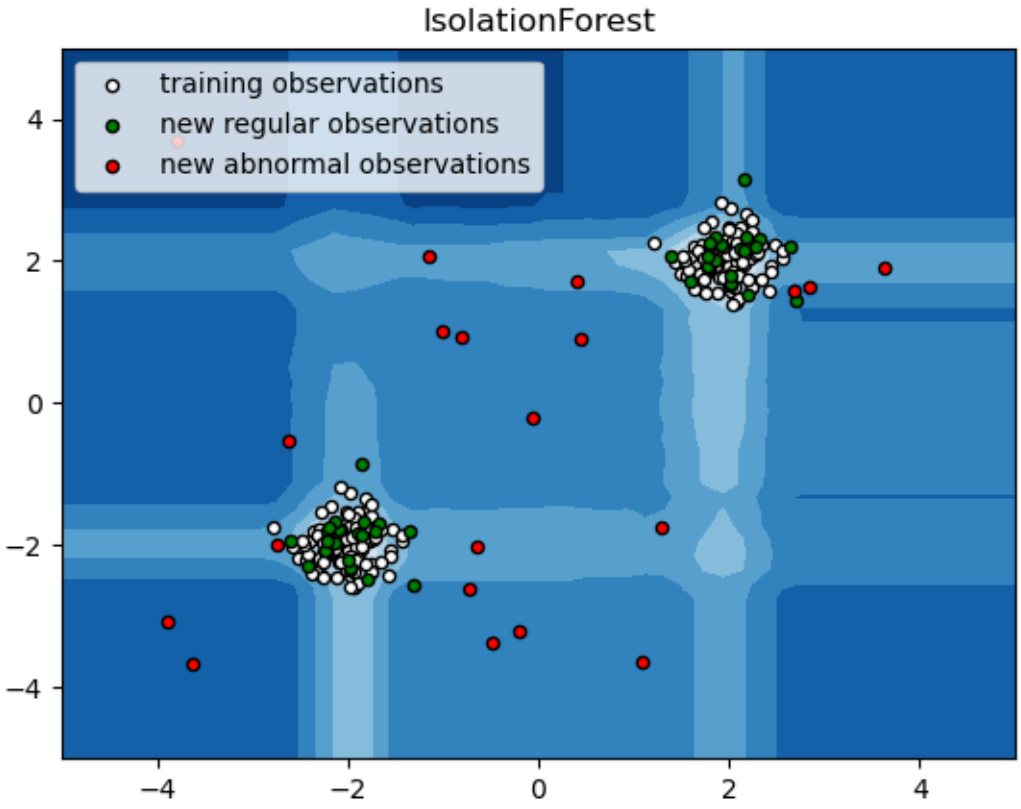

Figure 2: Scikit-learn Documentation of Isolation Forest

Example of IF in a 2 dimensional space can be seen in Figure 2

Loosely speaking, we update our knowledge of fraud based on the latent information of the data. Even if the IF is one of the most promising algorithm for anomaly detection, it is important to keep in mind that one single model will never be the answer to a structured business problem. Whatever the technique, the most important aspect of every solution is the understanding of the goal and a meaningful manipulation of data.

Chapter 1

Unsupervised Anomaly Detection

1.1 Definition of Unsupervised Learning

Loosely speaking Unsupervised Learning is the term used to describe all set of algorithms which task is to create or modify a representation of data. While in supervised learning we try to approximate a function $f(x)$, that best captures the relation between an input vector X and an output $y = f(x)$, in unsupervised learning we do not have a predefined label or output y in the data. The generic unsupervised case has a set of N observations $(x_1, x_2, \ldots, x_N)$ from a random vector X having joint density $Pr(X)$. The goal is to directly infer the properties of this probability density without the help of a supervisor providing correct answers or degree-of-error for each observation [Hastie et al. (2009)]. It is of immediate understanding that unsupervised learning is difficult to grasp and the number of possible solutions is heavily dependent by task. Here we list a few macro categories:

- Clustering
- Dimensional Transformation
- Outlier Detection

that we are going to discuss in the rest of this section.

1.1.1 Clustering

In clustering we try to identify patterns inside the data, and the final goal is to create a pseudo label among all the instances that shares a particular behaviour or characteristic. Clustering can be seen as an optimization problem, in which we do not use a label but the general notion of diversity given some constraints. Here few examples of constraints:

- Number of Clusters

- Density of Cluster

- Inter-cluster Variance

- Intra-cluster Variance

The general notion of diversity can change depending on the task, but it can be expressed as distance, dissimilarity or variability. The term diversity implies a relaxed definition of the common terminology of distance. In the mathematical sense, the term distance is used to describe a situation with precise properties like symmetry.
An example of clustering formalization is K-Means. We can think of K-means as an heuristic, iterative algorithm designed to optimize the euclidean distance J inside and between the clusters given the constraint of $K \in \mathbb{N}$, number of cluster defined a priori and randomly initialized.

The objective function, is defined iterating each element i using the following expression:

$$J = \sum_{k=1}^{K} \sum_{i=1}^{N} r_{i,k} ||x_i - \mu_l||^2$$

with r as indicator function of the kth cluster in the iteration.

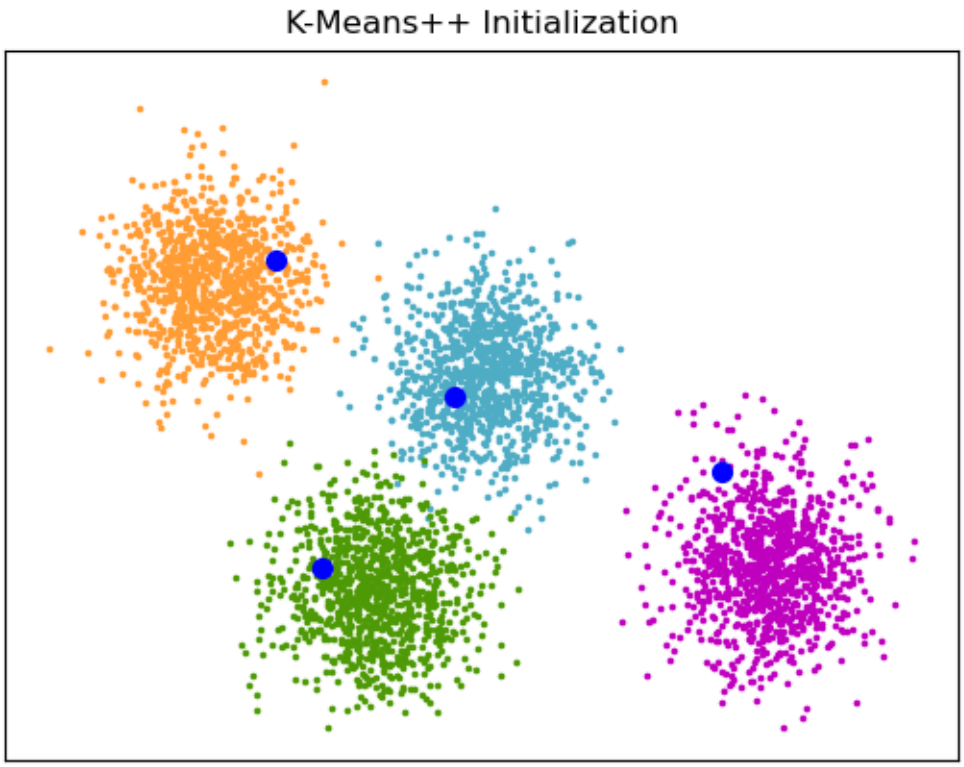

Figure 1.1: Scikit-learn Documentation of Kmeans++

A final considerations about this algorithm is its sensitivity to the initialization of the centroids, which are repeatedly updated according to the optimization. We can see in Figure 1.1 a representation of data and centroids. Unlike Figure 1.1, Data are not always well separated, and the choice of K heavily depends on the user choices.

1.1.2 Dimensional Transformation

In Dimensional Transformation tasks the final goal is to find better representation of the original inputs. The term transformation is used to describe any process in which we change the original dimensions. In the majority of cases, the task is to find the most significant representation, in p dimension, in k dimensions with $k < p$. In this last case we talk about dimensional reduction. An example of dimensional reduction technique is Principal Component Analysis.

Principal Component Analysis (PCA) is a linear technique that uses a set of orthogonal linear combinations of the original features, called principal components, to represent the data in a new coordinate system.

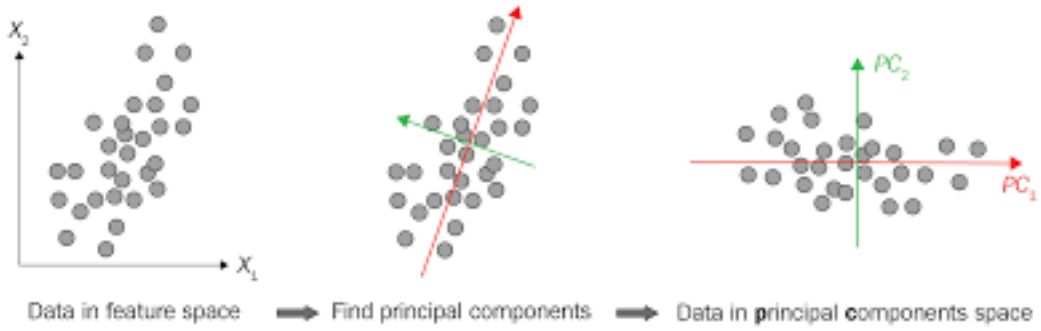

Figure 1.2: Steps of PCA

In formal terms, PCA can be defined as an linear dimensionality reduction technique that aims to find the directions (principal components) in the data that maximize the variance. Given a dataset X, collection of all sample x, by now iterated with notation x_i, of size $N \times D$, where N is the number of instances and D is the number of features, the goal of PCA is to find a new set of feature space Z of size $N \times k$, where k is the number of principal components, such that $Z = X * W$, where W is a matrix of size $D \times k$ containing the principal components as its columns.

PCA can be mathematically defined as the process of finding the eigenvectors of the covariance matrix of X, which are then used as the principal components. The eigenvectors are the directions of the data that maximize the variance.

Mathematically, we can express the PCA as the following steps:

1. Mean-centering the data by subtracting the mean from each feature:

$$X_{centered} = X - \frac{1}{N} \sum_{i=1}^{N} x_i$$

2. Calculating the covariance matrix:

$$\Sigma = \frac{1}{N} X_{centered}^T X_{centered}$$

3. Finding the eigenvectors and eigenvalues of the covariance matrix

$$\Sigma v_i = \lambda_i v_i$$

4. Sorting the eigenvectors by their corresponding eigenvalues in decreasing order.

5. Using the eigenvectors with the highest eigenvalues as the principal components, and project the data onto the new feature space by multiplying it with the matrix of eigenvectors

$$Z = X_{centered} V_k$$

PCA is a linear technique, which means that it only captures linear relationships between the features. Therefore, it may not work well when the data has non-linear relationships.

1.1.3 Outlier Detection

Outlier detection, also known as anomaly detection, is the process of identifying instances in a dataset that are unusual or deviate from the expected behavior. These instances, referred to as outliers, can be caused by errors in data collection, measurement, or entry, as well as by rare or unexpected events. Outliers can be harmful to the performance of many machine learning models, as they can lead to overfitting or poor generalization. There are several different techniques used in outlier detection, including density-based methods, statistical methods, proximity-based methods, and model-based methods. Each technique has its own advantages and limitations, and the appropriate technique will depend on the characteristics of the dataset and the desired outcome. Mahalanobis distance is a measure of similarity between data points that takes into account the covariance structure of the data. It is commonly used in outlier detection as a way to identify instances that are significantly different from the rest of the data. The Mahalanobis distance between a point x and the mean vector m of a dataset X is defined as:

$$D^2(x, m) = (x - m)^T \Sigma^{-1} (x - m),$$

where Σ is the covariance matrix of the dataset, and Σ^{-1} is the inverse of the covariance matrix.

The Mahalanobis distance is a powerful measure of similarity because it accounts for the correlation between features. Unlike the Euclidean distance, which gives equal weight to all features, the Mahalanobis distance gives more weight to features that are highly correlated with each other. This means that it can identify outliers that may be missed by other distance measures. In other words we have an outlier zone based on the trend and mean. If a data point is distant from the mean but inside the trend is not flagged.

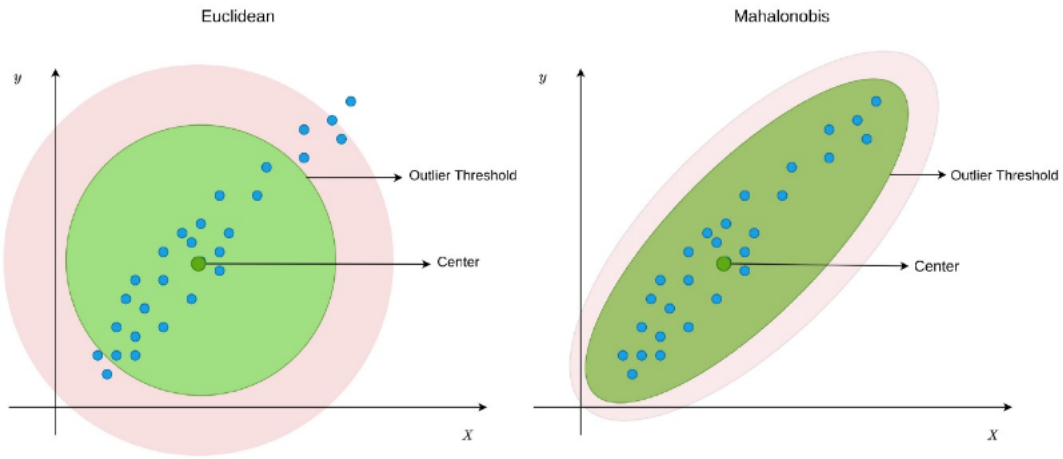

Figure 1.3: Anomaly Space for Euclidean vs Mahalanobis

As we see in Figure 1.3, The Mahalanobis distance is used to detect an outlier by computing the distance between each point and the distribution of the data. The instances whose distance exceeds a certain threshold are flagged as outliers. The threshold can be chosen using an appropriate criterion, such as a chi-squared distribution, or the k-nearest neighbor rule.

1.2 Performance Metrics of Unsupervised Learning

Unsupervised Learning is not focused on making precise prediction of an existing outcome $y = f(x)$. For example, in the task of clustering we measure the quality of the clusters (according to the specific task) and the availability of true labels. In the following we discuss separately the cases in which true labels are either available or not.

1.2.1 Unsupervised Measures without True Labels

A common example of unsupervised learning evaluation without the presence of true label is clustering evaluation. Measures like the Silhouette Score, denoted by now as S, allows analysts to evaluate the quality of clustering results. Silhouette Score measures the similarity of a data point to its own cluster compared to other clusters. A high Silhouette Score indicates that a data point is well-matched to its own cluster and poorly-matched to neighboring clusters,

while a low Silhouette Score indicates the opposite. Rousseeuw 1987 We proceed to describe in more formal terms how the score is computed. For each data point i:

$$s(x_i) = \frac{d_1(x_i) - d_2(x_i)}{\max\{d_2(x_i), d_1(x_i)\}}$$

where $d_2(x_i)$ is the average distance from point x_i to all other points in the same cluster, and $d_1(x_i)$ is the average distance from point i to all points in the nearest neighboring cluster. The maximum of a(i) and $b(i)$ is taken to ensure that $s(i)$ is always between -1 and 1. Then, the overall silhouette score S is computed as the mean of all $s(i)$:

$$S = \frac{1}{n} \sum_{i=1}^{n} s(i)$$

1.2.2 Unsupervised Measures with True Labels

For the case of Anomaly Detection, we rely on the same techniques adopted in supervised learning. The presence of a true label set is mandatory for computing the following measures:

- True Positive Rate (TPR) and False Positive Rate (FPR): TPR, also known as recall, is the proportion of true anomalies that were correctly identified, while FPR is the proportion of normal instances that were incorrectly identified as anomalies. A balanced trade-off between TPR and FPR is desired, as a high TPR may result in a high number of false positives.

- Precision: Precision is the proportion of correctly identified anomalies among all the instances that were identified as anomalies. A high precision indicates a low rate of false positives.

- F1-score: The F1-score is the harmonic mean of precision and recall. It is used to balance the trade-off between precision and recall, as it considers both metrics equally.

- Area Under the Receiver Operating Characteristic (ROC) curve (AUC-ROC): ROC curve is a graphical representation of the TPR and FPR trade-off for every possible cut-off. As we see in Figure 1.4, the AUC-ROC is a scalar value that ranges from 0 to1, where a value of 1 indicates a perfect classification, and a value of 0.5 indicates a random classification. AUC-ROC is commonly used to evaluate anomaly detection algorithms, as it is insensitive to class imbalance.

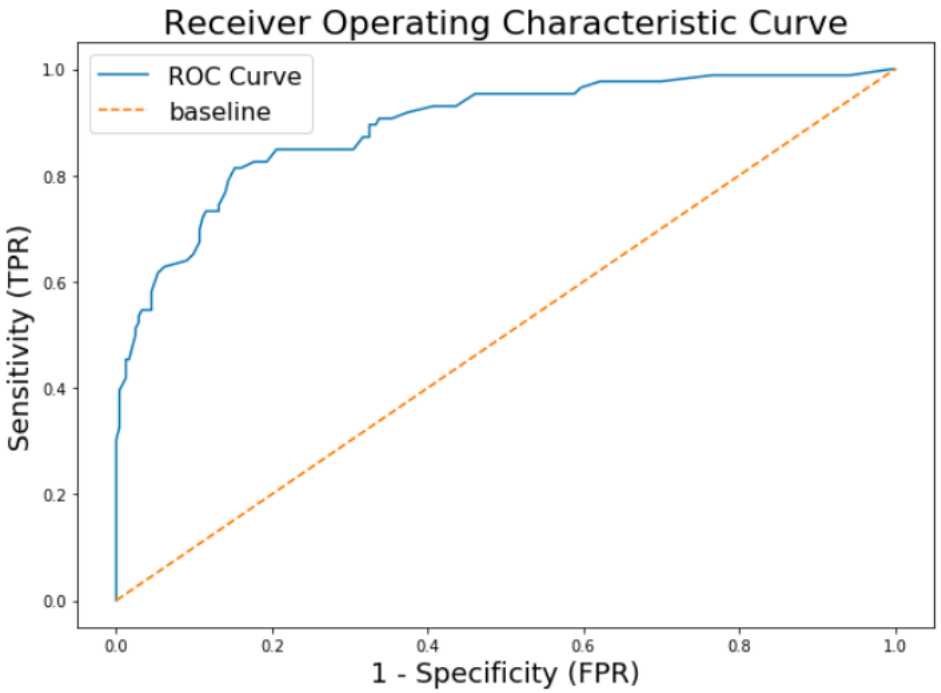

Figure 1.4: Example of AUC Plot

The shift from Supervised to Unsupervised is due to the presence of extremely rare categories in the outcome. Such imbalance create a bias in the supervised training process, and the solutions of Over-sampling likely lead to over-fitting. On the other hand Under-sampling, the act of reducing the data to the dimension of the rare events, will cause insufficient sample to make robust predictions. Cost Sensitive Loss functions are a possible solution to the problem, but considerably complicate the model and its understanding. The whole framework of anomaly detection via unsupervised learning is built on the intuition that extremely rare events always deviate from a general trend in the data, if such trend can be spotted and we can compute a "distance/dissimilarity" measure we can make prediction.

Chapter 2

Isolation Forest

2.1 Introduction

The algorithm was first introduced in the paper "Isolation Forest" by Liu et al. 2008 as an ensemble-based method for anomaly detection. This method isolates anomalies according to the following assumptions:

- Anomalies are a minority consisting of fewer instances
- Anomalies have attribute-values that are different from normal instances

The method mimics a Random-Forest approach using the isolation trees.

Definition of isolation tree

An isolation tree *iTree* is a tree such that any node $T \in iTree$ satisfies the following property:

- T is either an external-node with no child, or an internal-node with one test and exactly two daughter nodes (T_l, T_r)
- A test consists of an attribute q and a split value p,both randomly selected, such that the test $q < p$ divides data points into T_l and T_r.
- has a height limit l

Definition of Path Length

The path length of a data point in an isolation tree,by now $h(x)$, is defined as the number of edges traversed from the root node to the leaf node that isolates the data point. In other words, the number of splits required to isolate a sample.

Definition of Isolation Forest

An Isolation Forest *iForest*, is an ensemble of *iTree* that return an anomaly score based on path length of each *iTree*. The final anomaly score, for each data point, is a function of the expected value of the path length and is used to make a final judgment on the instance.

Definition of anomaly score

The anomaly score $s(x, n)$ of a data point x in X, with X dataset of size n, defined as:

$$s(x, n) = 2^{\frac{-E(x)}{c(n)}}$$

- $E(x)$ is the average path length of x over the isolation trees
- $c(n)$ is a normalizing factor that also depends on the number of instances. In the original paper is presented as follows:

$$c(n) = 2H(n-1) - \frac{2(n-1)}{n}$$

A normalizing factor $c(n)$, the average of h(x) given data set of n instances, is required to ensure the anomaly scores fall within a range of [0,1]. Without this normalization, the anomaly scores would depend solely on the number of trees in the forest and would increase with the number of trees, making comparisons with other datasets or models almost impossible. The role of the function H is not part of the core intuition of the algorithm but favorites the normalization.

- $H(n)$ harmonic number and is defined as

$$ln(n) + ec$$

where $ec = 0.5772156649$ (Euler's constant)

As we can see in Figure 2.1 there is a relationship between the score and the Expected value of path length

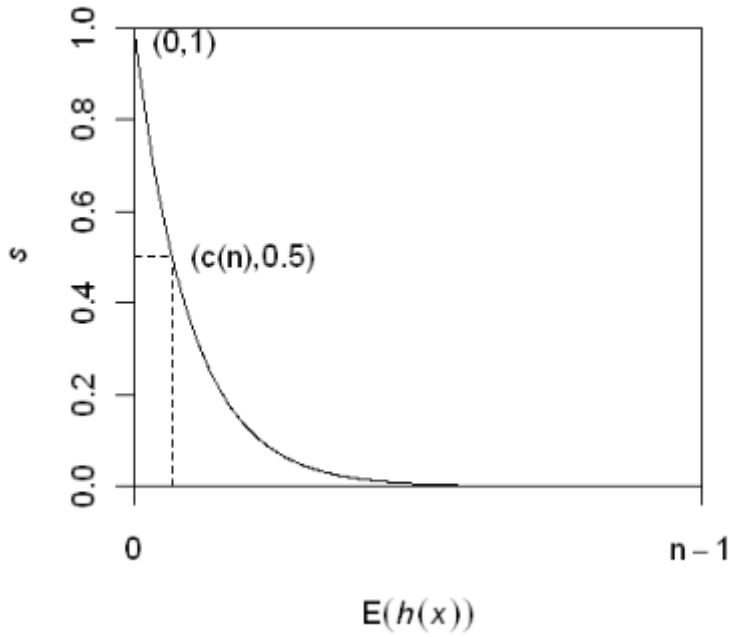

Figure 2.1: The relationship of expected path length E(h(x)) and anomaly score

Using the anomaly scores, we are able to make the following assessment:

- if instances return a value of s very close to 1, then they are definitely anomalies.
- if instances have s much smaller than 0.5, then they are quite safe to be regarded as normal instances
- if all the instances return s close to 0.5, then the sample does not really have any distinct anomaly trait.

We remark again on the small differences between the paper and the official scikit-learn implementation, applied in order to be consistent with the best practices of the framework. The conclusions we apply using the python implementation are so inverted.

- Anomalies are values with a low score since they are easy to isolate. If instances have a score output much smaller than a general cut-off, then they have distinct anomaly traits.

2.2 Formalization of Algorithm

We now proceed to the formal definition of the algorithms, this compact representation summarizes the steps and sub-algorithm, like Isolation Tree and Path Length presented in the original paper:

Algorithm 1 : $iForest(X, t, \psi)$

Inputs: X - input data, t - number of trees, ψ - sub-sampling size

Output: a set of t *iTrees*

1: **Initialize** $Forest$
2: set height limit $l = ceiling(\log_2 \psi)$
3: **for** $i = 1$ to t **do**
4: $\quad X' \leftarrow sample(X, \psi)$
5: $\quad Forest \leftarrow Forest \cup iTree(X', 0, l)$
6: **end for**
7: **return** $Forest$

Figure 2.2: Isolation Forest of original paper

Algorithm 2 : $iTree(X, e, l)$

Inputs: X - input data, e - current tree height, l - height limit

Output: an iTree

if $e \geq l$ or $|X| \leq 1$ **then**
  return *exNode*$\{Size \leftarrow |X|\}$
else
  let Q be a list of attributes in X
  randomly select an attribute $q \in Q$
  randomly select a split point p from max and min values of attribute q in X
  $X_l \leftarrow filter(X, q < p)$
  $X_r \leftarrow filter(X, q \geq p)$
  return *inNode*$\{Left \leftarrow iTree(X_l, e + 1, l),$
    $Right \leftarrow iTree(X_r, e + 1, l),$
    $SplitAtt \leftarrow q,$
    $SplitValue \leftarrow p\}$
end if

Figure 2.3: Isolation Tree of original paper

Algorithm 3 : $PathLength(x, T, e)$

Inputs : x - an instance, T - an iTree, e - current path length; to be initialized to zero when first called

Output: path length of x

1: **if** T is an external node **then**
2: return $e + c(T.size)$ {$c(.)$ is defined in Equation 1}
3: **end if**
4: $a \leftarrow T.splitAtt$
5: **if** $x_a < T.splitValue$ **then**
6: return $PathLength(x, T.left, e + 1)$
7: **else** $\{x_a \geq T.splitValue\}$
8: return $PathLength(x, T.right, e + 1)$
9: **end if**

Figure 2.4: Path Length of original paper

To be consistent with the figure Figure 2.4, we redefine what the original paper called Equation 1.

$$c(n) = 2H(n-1) - \frac{2(n-1)}{n}$$

As we see in Figure 2.5, if there is a strong trend in the data and a multitude of observations, everything outside of the trend should stand out and therefore be easier to isolate [Liu et al. 2008]

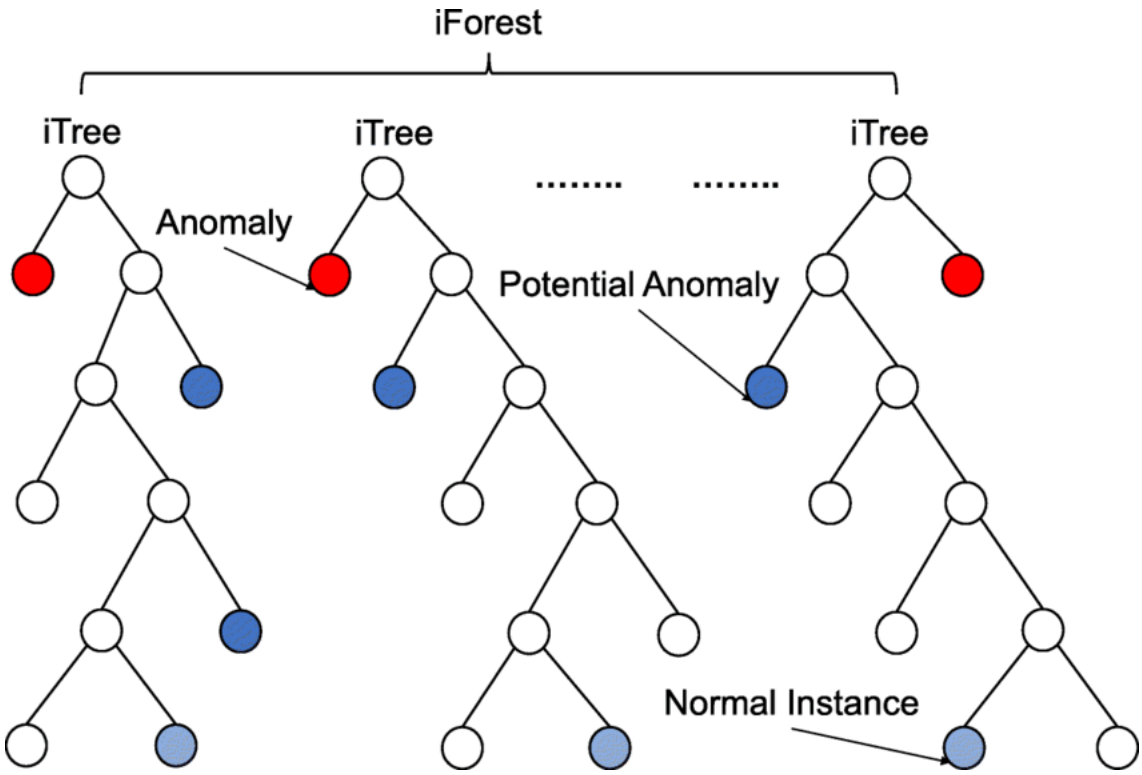

Figure 2.5: Isolation Trees and Anomaly Flag

2.3 Variations

Within a few years from the original publication, a few variations came out to overcome possible issues and reduce sensitivity to parameters. The most important variation is Extended Isolation Forest (EIF) proposed in [Hariri et al. 2018], which addresses the issue of slicing only parallel to one of the axes, since the separation is done using a simple inequality. As we can see in Figure 2.7, EIF's branching method allows the slicing of data via hyperplanes, this creates non-parallel segmentation. Instead of selecting the feature and value, it selects a random slope b for the branching cut and a random intercept a. The slope can be generated from $N(0, 1)$ Gaussian distribution, and the intercept is generated from the uniform distribution with bounds coming from the sub-sample of data to be split. The branching criteria for the data splitting for a given point x, and the value of its feature q, is

$$(q - a) * b < 0$$

.

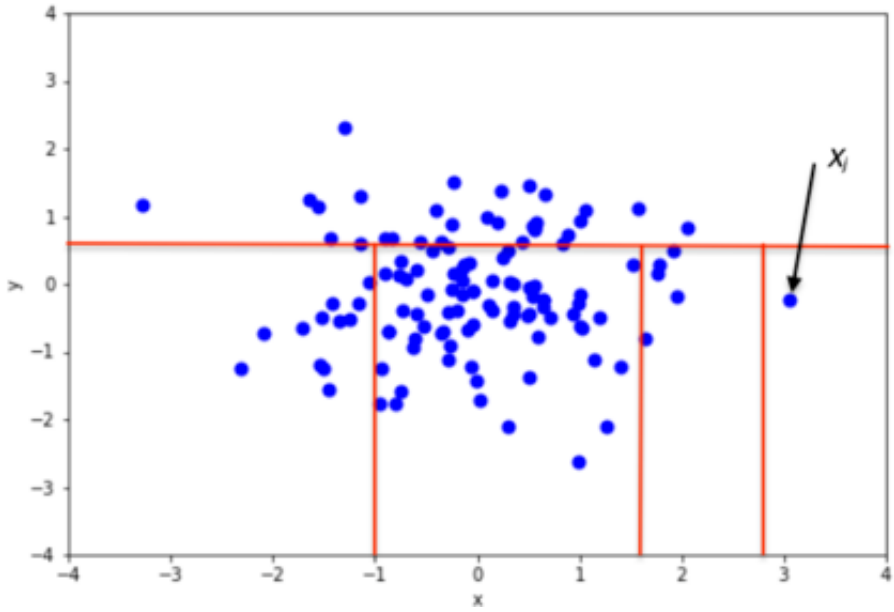

Figure 2.6: Traditional Isolation Tree slice

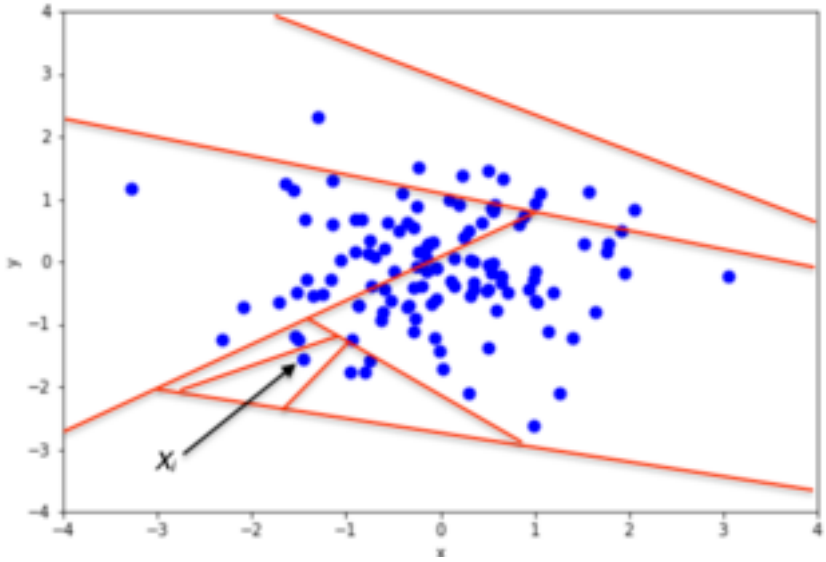

Figure 2.7: Extended Isolation Tree slice

Another variation is the Multi-dimensional Isolation Forest (MIF) proposed in [Hariri et al. 2018]. The authors propose a new method to handle high-dimensional data and for detecting anomalies in multi-dimensional data. The algorithm is based on the idea of using multi-dimensional partitioning to iso-

late anomalies in high-dimensional data. This last variation has not been yet implemented in scikit-learn and it will not be used in this work.

2.4 Parameters

There are two input parameters to the Isolation Forest algorithm. They are the sub-sampling size ϕ and the number of trees t. Sub-sampling is important for the efficiency of the algorithm and also to be sure that there is a representative set of data in each iTree. Its value must be tuned or set using prior knowledge, and in this sense is the only major drawback of isolation forest. [Liu et al. 2008] The original authors of isolation forest have tested the robustness of the algorithm to a change in subsample size and number of features; the algorithm was shown to be robust as long as data were representative of the original distribution. As for the number of trees, the ensemble logic helps convergence since a single iTree will unlikely give robust results.

The python implementation also requires a prior guess of **contamination** for hard classification, which is the expected value of anomalies in the data. It can be defined using business knowledge or statistical approaches. It must be specified that contamination is not influencing the fitting process but simply controls the threshold for the decision function when a scored data point should be considered an outlier. The final score is hopefully a skewed distribution with a slight peak in the tail, just like the following:

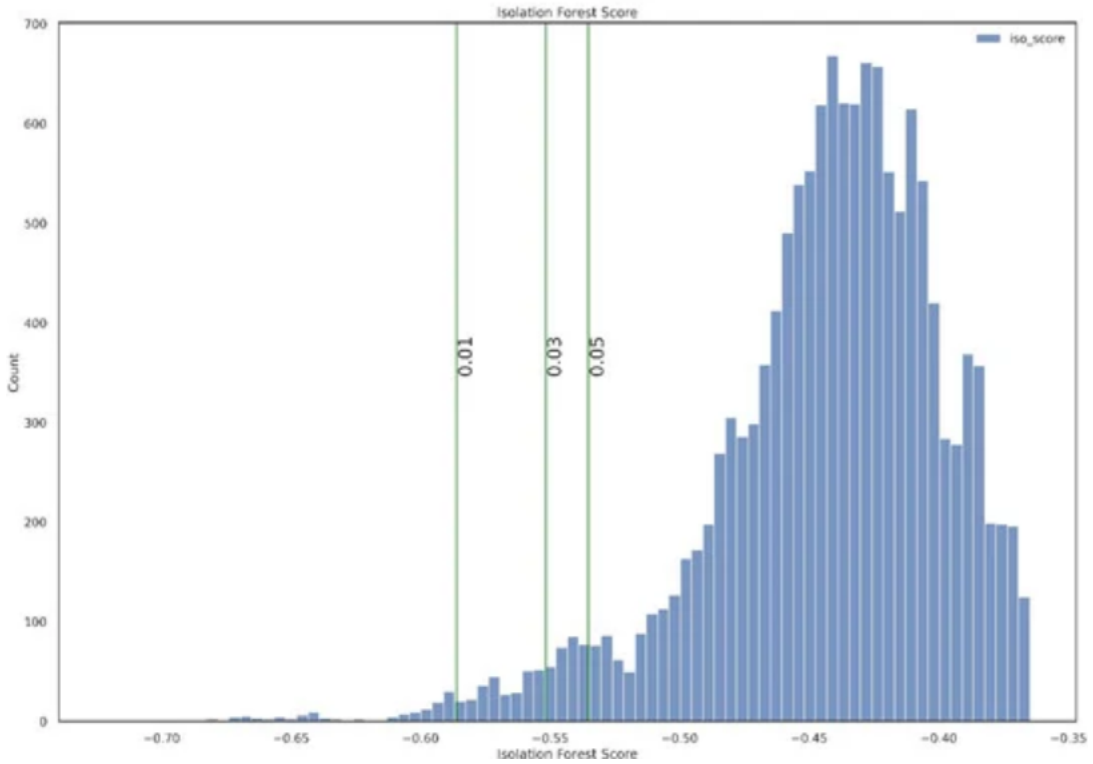

Figure 2.8: Anomaly Score Distribution according to Scikit-learn output (opposite score of original)

As we saw in Figure 2.8, setting a threshold at a given point of the distribution creates a final cutoff for the binary outcome. In this framework, we consider the parameter contamination as a percentile.

This last characteristic of isolation forest to be flexible in the final cut-off is extremely useful in a production environment. In a situation of alert, the contamination can increase and avoid losses due to a major change in the fraud concept or a drop in performance. The result is a much quicker update than traditional business rules and other supervised models, without the immediate need for refitting.

2.5 Advantages

As for every algorithm, complexity is one of the main points for determining effectiveness and feasibility. The time complexity of the Isolation Forest algorithm depends on the number of trees in the forest and the size of the data.

According to the original paper the advantages of Isolation Forest are mostly due to the use of trees and in particular can be summarized by the following properties:

- Non-Parametric: it does not require any prior knowledge about the distribution of the anomalies nor their presence, in fact, the goal is not focused on defining anomalies but to create a standard representation of a process/-pattern within the data.
- Scale invariant: able to handle high-dimensional data without the need to scale them.
- Versatile and Robust: can handle both numerical and categorical data in high dimensional feature space, without the problem of multicollinearity. Contrary to existing methods where a large sampling size is more desirable, the isolation method works best when the sampling size is kept relatively small and with essential features. Thus, sub-sampling provides a favorable environment for iForest to work well.
- Simple: the algorithm is very simple and it never requires complicated transformations since it is based on a convergence of random split of multiple isolation trees

Furthermore, the use of trees eliminates the need for a prior definition of distance or density, a requirement in many unsupervised clustering techniques like Kmeans or DBSCAN.

From a business perspective, isolation forest act as an ensemble of investigators that jointly monitor several KPIs. Although feature selection is an advantage, inside an unsupervised framework it might not be possible. According to the original paper, isolation forest also suffer the "curse of dimensionality". The proposed solution by the authors was to increase the number of isolation trees and use feature sampling to reduce noise.

2.6 Limitations

The most evident limitation of Isolation Forest is the need for an explicit pattern in the data since without separation in the original space the algorithm is completely incapable of isolating the anomalies. To make isolation more efficient, heavy feature engineering is suggested, and ordinal data do not guarantee better predictions. The most favorable feature is the one that incentives separation and a compact trend. The original algorithm has also the minor drawback of parallel segmentation of space, which might be difficult in the case of curve patterns.

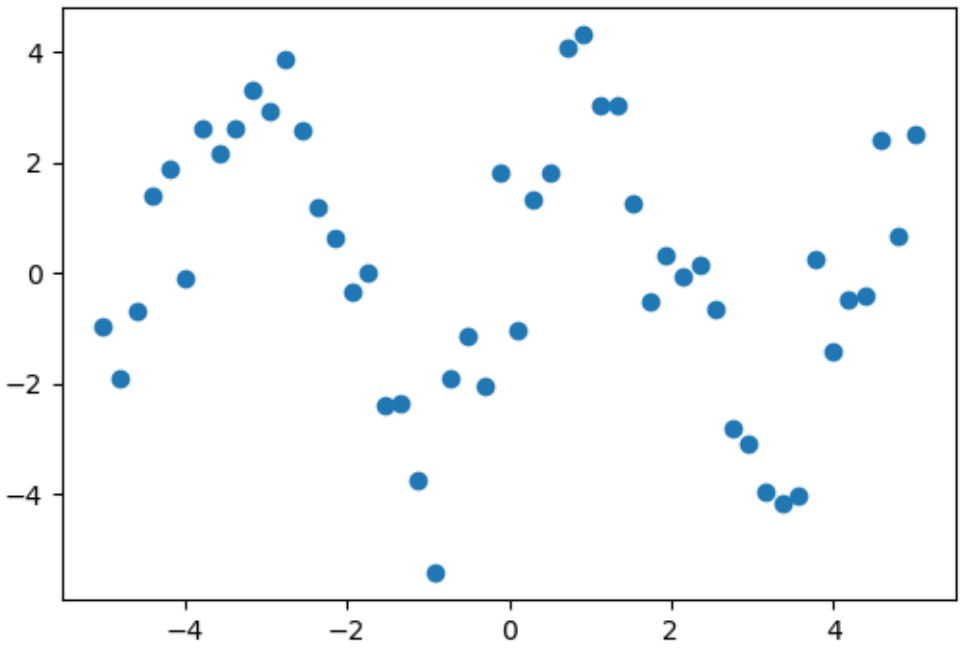

Figure 2.9: Example of curve trend that can limit Isolation Forest predictions

Since the simple inequality is not the best fit for a good separation in the case of curves like in Figure 2.9, further splits will be required without feature engineering. For the task of anomaly detection, the algorithm does not have further major drawbacks besides the one previously outlined. With the correct definition of the task and the correct assumption, an isolation forest is often able to find outliers.

Chapter 3

Fraud Detection Systems

3.1 Business Context

Due to privacy concerns and bureaucracy in data management, it is not easy to obtain data. Therefore, it has become best practice to simulate transactions and initiate the Pseudonymization of real data. These approaches have both pros and cons, a simple advantage is the freedom of experimenting with real methodologies and without privacy issues. The major disadvantage is the creation of non-actionable models, which often need a long validation time before being fitted on real data. In this work, we exploit the simulation framework to evaluate a few methodologies related to isolation forest. The majority of resources are focused on statistics and algorithms, without a proper introduction and definition of the process in which they take part. This lack of context often leads to misleading interpretations or useless models. Here we proceed to define, in a precise way, the business terminology and the technical language of Fraud Detection.

Two types of transactions can be distinguished: those that require a physical card, referred to as card-present (CP) scenarios, and those that do not, referred to as card-not-present (CNP) scenarios. CP scenarios include transactions at physical locations such as stores or cash points. CNP scenarios, on the other hand, encompass transactions made through the internet, phone, or mail. The differentiation between CP and CNP scenarios is crucial as the methods used to exploit a card vary depending on whether or not a physical card is needed. Additionally, it is important to note that in recent times, fraudsters tend to focus more on exploiting weaknesses in CNP scenarios than in CP scenarios as we can see in Figure 3.1 This is likely because CP scenarios have been around for a long time and have become quite resistant to fraud attempts [Le Borgne et al. 2022].

Card-present fraud occurs when a fraudster is able to make a fraudulent transaction using a physical payment card, either at a point-of-sale or an ATM. In this setting, fraud scenarios are typically categorized into three types: lost or stolen cards, counterfeited cards, and cards not received.

- Lost or Stolen Card: The card belongs to a legitimate customer and falls into the hands of a fraudster due to a loss or theft. This is the most common type of fraud in the card-present setting and allows a fraudster to make transactions as long as the card is not blocked by its legitimate owner. In this scenario, the fraudster typically tries to spend as much and as quickly as possible.

- Counterfeited Card: A fake card is produced by a fraudster by imprinting the information of a card. Such information is usually obtained by skimming the card of the legitimate customer without them noticing. Since the legitimate owners are not aware of the existence of a copy of their card, the source of the fraud may be more difficult to identify, as the fraudster can wait a long time before making use of the fake card. The increased use of chip-and-PIN technology has reduced this type of fraud.

- Card not Received: The card is intercepted by a fraudster before it reaches the mailbox of a legitimate customer. This may happen if a customer orders a new card, which is intercepted, or if a fraudster manages to order a new card without the knowledge of the legitimate customer, and have it delivered to a different address. In the former case, the customer may quickly warn the bank that their card was not received and have it blocked. The latter case can be harder to detect since the customer is not aware that a new card was ordered.

Card-not-present frauds are transactions that are conducted remotely, without the use of a physical card. These types of frauds can occur through mail, phone, or online transactions and typically involve the use of only some of the information on a card, such as a card number, expiration date, security code, and billing information.

Compared to card-present frauds, there is less statistical data available on the causes of CNP fraud. However, it is known that most of these frauds are a result of illegally obtained payment credentials, either from data breaches or directly from cardholders through phishing or scam messages. These stolen credentials are often sold on underground web marketplaces and used by criminal groups. It is worth noting that the group of criminals who steal data is usually different from the group who perpetrate fraud [Le Borgne et al. 2022].

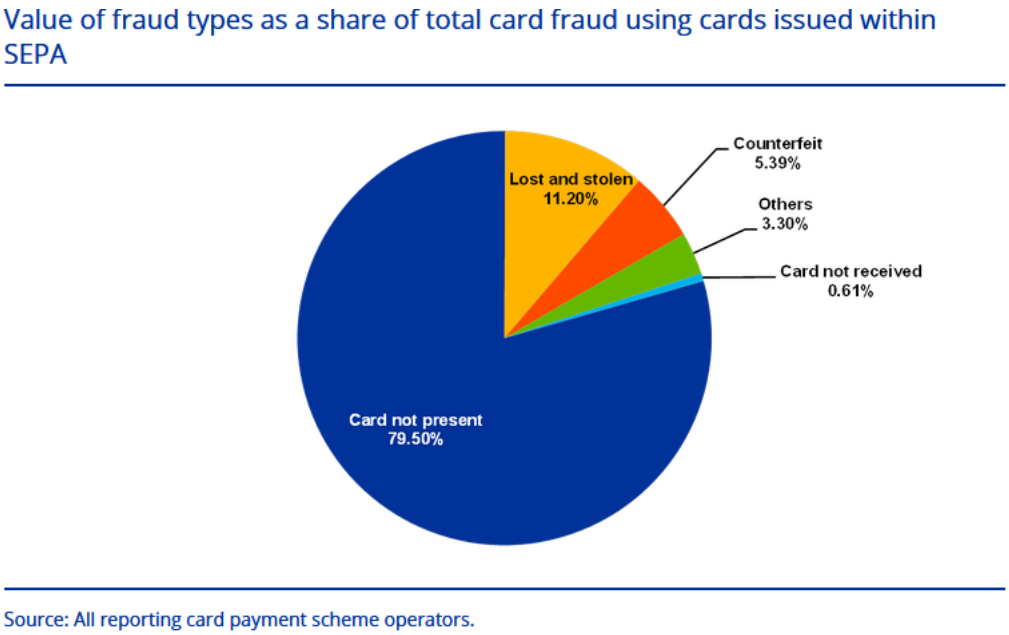

Figure 3.1: European Central Bank. 6th report on card fraud. August 2020.

3.2 Design of the Fraud Detection System

The payment system is designed as a sequence of steps to connect the two parties, in this case a consumer and a generic service provider. when a transaction is taking place. A fraud detection system (FDS) is a system that is designed to detect and prevent fraudulent transactions. It typically consists of several layers, including transaction-blocking rules, scoring rules, and a data-driven model. The system uses information available when a payment is requested but also relies on previous information depending on the case. In Figure 3.2, we can summarize how a real FDS is built.

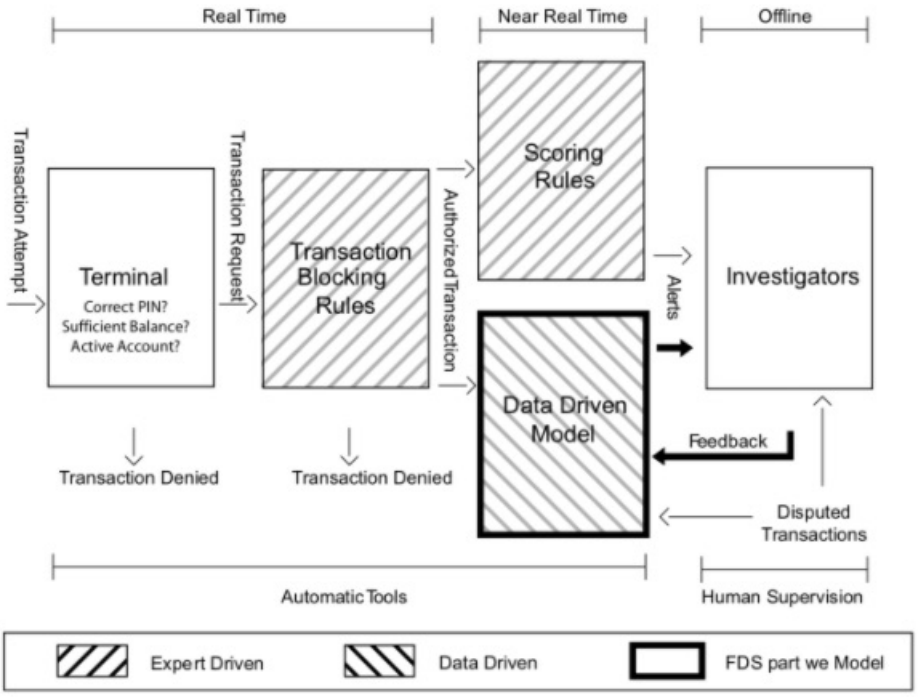

Figure 3.2: Design of a Fraud Detection System.

The transaction-blocking rules are manually designed by investigators and are meant to block transaction requests that are perceived as fraud. Scoring rules are also manually designed and assign a score to each authorized transaction, with higher scores indicating a higher likelihood of fraud. The Data Driven Model is trained from a set of labelled transactions and estimates the probability for each transaction to be a fraud. It is usually a supervised learning task, but we have seen in Chapter 1 that is not always the case. The primary goal of a Data Driven Model is to return precise alerts, as investigators might ignore further alerts when too many false alarms are reported. In recent systems, transactions associated with very high-risk scores can bypass investigators and be directly sent to the cardholder for feedback requests (e.g. by SMS).

3.3 Data Generating Process

The simulation will consist of the following five main steps:

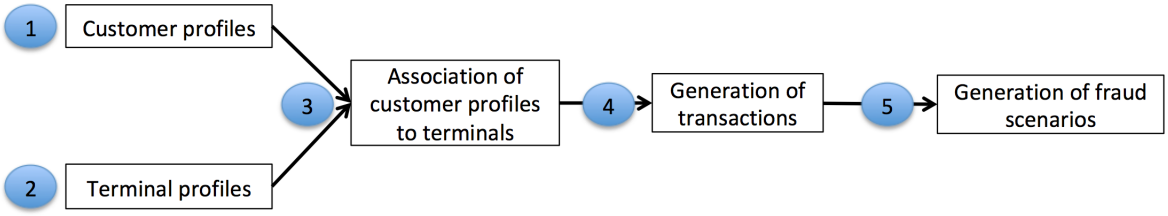

Figure 3.3: Flow of simulated transactions.

1. Generation of customer profiles: Every customer is different in their spending habits. This will be simulated by defining some properties for each customer. The main properties will be their geographical location, their spending frequency, and their spending amounts.

2. Generation of terminal profiles: Terminal properties will simply consist of a geographical location.

3. Association of customer profiles to terminals: We will assume that customers only make transactions on terminals that are within a radius of their geographical locations. This makes the simple assumption that a customer only makes transactions on terminals that are geographically close to their location. This step will consist of adding a feature list-terminals to each customer profile, that contains the set of terminals that a customer can use.

4. Generation of transactions: The simulator will loop over the set of customer profiles, and generate transactions according to their properties (spending frequencies and amounts, and available terminals).

5. Generation of fraud scenarios: This last step will label the transactions as legitimate or genuine. This will be done by following three different fraud scenarios, which are introduced in the next section.

3.4 Fraud Scenarios

The last step of the simulation involves adding fraudulent transactions to the data set, using the following fraud scenarios created by [Le Borgne et al. 2022]:

- Scenario 1: Any transaction whose amount is more than 220 is a fraud. This scenario is not inspired by a real-world scenario. Rather, it will provide an obvious fraud pattern that should be detected by any baseline fraud

detector. This will be useful to validate the implementation of a fraud detection technique and will randomly appear and fade over time.

- Scenario 2: Every day, a list of two terminals is drawn at random. All transactions on these terminals in the next 28 days will be marked as fraudulent. This scenario simulates a criminal use of a terminal, through phishing for example. Detecting this scenario will be possible by adding features that keep track of the number of fraudulent transactions on the terminal. Since the terminal is only compromised for 28 days, additional strategies that involve concept drift will need to be designed to efficiently deal with this scenario.
- Scenario 3: Every day, a list of 3 customers is drawn at random. In the next 14 days, 1/3 of their transactions have their amounts multiplied by 5 and marked as fraudulent. This scenario simulates a card-not-present fraud where the credentials of a customer have been leaked. The customer continues to make transactions, and transactions of higher values are made by the fraudster who tries to maximize their gains. Detecting this scenario will require adding features that keep track of the spending habits of the customer. As for scenario 2, since the card is only temporarily compromised, additional strategies that involve concept drift should also be designed.

3.5 Formalization of the Machine Learning Approach

The formalization of the problem goes according to the context we just described and we will apply the best practice of unsupervised learning to solve it. Given that we know the ground truth we will use the Area Under the Receiver Operating Characteristics, by now ROC-AUC,a performance measurement for the classification problems at various threshold settings. ROC is a probability curve and AUC represents the degree or measure of separability. It tells how much the model is capable of distinguishing between classes. Higher the AUC, the better the model is at predicting 0 classes as 0 and 1 classes as 1. This metric is widely accepted as the standard measure of model performance and it does not depend on any specific cut-off. Furthermore, it will be a fair representation of the ability of our anomaly score to discriminate between fraud and non-fraud transactions according to a real imbalanced scenario.

3.6 Model Scope

The scope of the model is to evaluate the likelihood of the current transaction being anomalous using an anomaly score. It is part of a Data-Driven Layer

in a Fraud Detection System and works in a close to real-time scenario. In order to be successful, the model is expected to recognize each of the fraud scenarios introduced in section 3.4. The model is built on simulated transactions and assumes every user has at least 1 month of the transaction history. The reason for such requirements is connected to the original assumption of the model described in Chapter 2. In order to spot anomalies the model needs a pattern, and the pattern is built using not only using the information of the current transaction but also the previous ones. The final features of the model should explicit a benchmark between the current transaction and the "normal" behaviour. Without transaction history, there is no way to create features. This model is therefore a P.O.C. (Proof of Concept) for fraud detection and it cannot be used for production without adaptation to real transactions.

3.7 Data Assessment

We are using simulated transaction data, generated by Le Borgne et al. 2022. The scenario data-set is based on 5000 customers, 10000 terminals and 183 days of transactions (which corresponds to a simulated period of 183 days). The final number of transactions is in the order of millions (1754155) with 0.8% (14681) frauds and can vary from simulation to simulation if required. At the time of the transaction, the following details are available:

Name of Variable	**Description**
TRANSACTION ID	*Unique identifier for each transaction*
TX DATETIME	*Date and time of the transaction*
CUSTOMER ID	*Unique identifier for the customer involved in the transaction*
TERMINAL ID	*Unique identifier for the terminal used in the transaction*
TX AMOUNT	*Amount of the transaction*
TX TIME SECONDS	*Number of seconds to complete the transaction*
TX TIME DAYS	*Number of days between the last and the current transaction*
TX FRAUD	*Indicator of whether the transaction is fraudulent (1) or not (0)*
TX FRAUD SCENARIO	*Scenario that led to the transaction being flagged as fraudulent*

Table 3.1: Variable Description

3.8 Model Choice

According to the previous consideration, the model choice is Isolation Forest. The advantage and drawbacks of the model have been described in Chapter 2.

3.9 Feature Engineering

The task of detecting fraud is far from trivial, and in the previous chapters we have listed all the major challenges of the case. In order to build a powerful model, feature engineering is particularly important, and its starts by making hypotheses on what can be useful to approximate. According to several resources, the major areas of investigation are the following:

- Customer Spending Behaviour
- Terminal Behaviour
- Pattern in Transaction Time

Another important aspect is the time component. The book "Python Feature Engineering Cookbook" [Galli 2020] explains the usefulness of window aggregation when working with transaction data. This technique involves grouping transactions into a set period of time, such as days or weeks, and then calculating various statistics within each window. This can be useful for uncovering patterns or trends that may not be immediately apparent when looking at the data as a whole. By using window aggregation, we can also deal with concept drift and find patterns or trends that may change over time. Overall window aggregation is a powerful technique that can be used to extract valuable insights from transactional data, and is highly recommended for any credit card fraud detection tasks. We now proceed to define the reasoning behind every area of investigation.

3.9.1 Customer Spending Behavior

Spending behaviour refers to the way a customer cash-flow is distributed across different time windows. The idea is to compare the current transaction with the ”normal” pattern in the feature space. Accordingly: for every customer id, on different time windows going from 1 to 30 days, every transaction/row has been enriched with the following statistics :

Name of Variable	Description
CUSTOMER ID NB TX	*Number of tx*
CUSTOMER ID AVG AMOUNT	*Average tx amount*
CUSTOMER ID MAX AMOUNT	*Maximum tx amount*
CUSTOMER ID VC AMOUNT	*Variation coefficient of tx amount*
CUSTOMER ID QT70 AMOUNT	*70th percentile of tx amount*
CUSTOMER ID QT90 AMOUNT	*90th percentile of tx amount*

Table 3.2: Customer Spending Variable Description

3.9.2 Terminal Behavior

Terminal behavior refers to the different level of risk a payment system has been known for in the past. In the fraud context it very likely that a compromised terminal (a terminal that is frauding or is victim of fraud) will still be active for a period of time before is shut down or return to normality. The risk score will be defined as the average number of fraudulent transactions that occurred on a terminal ID over a time window. As for customer ID transformations, we will use several window sizes, from 1 to 30 days.

Name of Variable	Description
TERMINAL ID NB TX	*Number of tx registered*
TERMINAL ID RISK	*Ratio tx flagged in the past over total in period*

Table 3.3: Terminal Behavior Variable Description

3.9.3 Pattern in Transaction Time

Features related to time are not powerful when used without context, but extremely useful when combined with other features. Several fraudsters actively operate in a specific time window or occasion. In order to trace the presence of a time pattern, the following features are created:

Name of Variable	Description
TX IN NIGHT	*Binary Flag if tx is in night*
TX IN WEEKEND	*Binary Flag if tx is in weekends*
TX TIME DAY	*Day of month when tx occurred*
TX TIME HOUR	*Hour of day when tx occurred*
TX TIME MINUTE	*Minute when tx occured*
TX TIME SECONDS	*Seconds when tx occured*

Table 3.4: Time Variable Description

3.9.4 Explicit Flags

This is a special type of feature created for the sole purpose of facilitating isolation forest. The logic goes as follows: Given the split is actually random and the model is an ensemble of trees, the use of a binary flag will act as a sort of meta-classifier. The current transaction is confronted with the previous statistics and will return a 1 if a particular condition is met, otherwise 0. For example if the current amount is the maximum amount in the current time period. In this way, we have an extra layer of investigation to increase the chance of outliers standing out from the trend.

3.9.5 Final note on Feature Engineering

The result of the feature engineering process is a list of 64 features. In a supervised framework we could have done a feature selection process but, given the unsupervised approach and the flexibility of the model, we do not need to go further. In a future version, some dimensionality reduction or even clustering could be applied. To summarize, thanks to feature engineering now every row is independent of the other but brings information about the previous transactions; this approach makes now possible to compare each new transaction with a multivariate distribution and facilitates the recognition of a trend.

3.10 Model Validation

As for model validation, we have previously explained why we use AUC as our main model performance indicator. The strategy to validate the model can be based on different contexts:

1. Standard Cross Validation with shuffle splits

2. Scenario Cross Validation

3. Time Cross Validation

Cross-Validation (CV), Figure 3.4, evaluate learning algorithms by dividing data into different segments: one used to learn or train a model and the other used to validate the model.

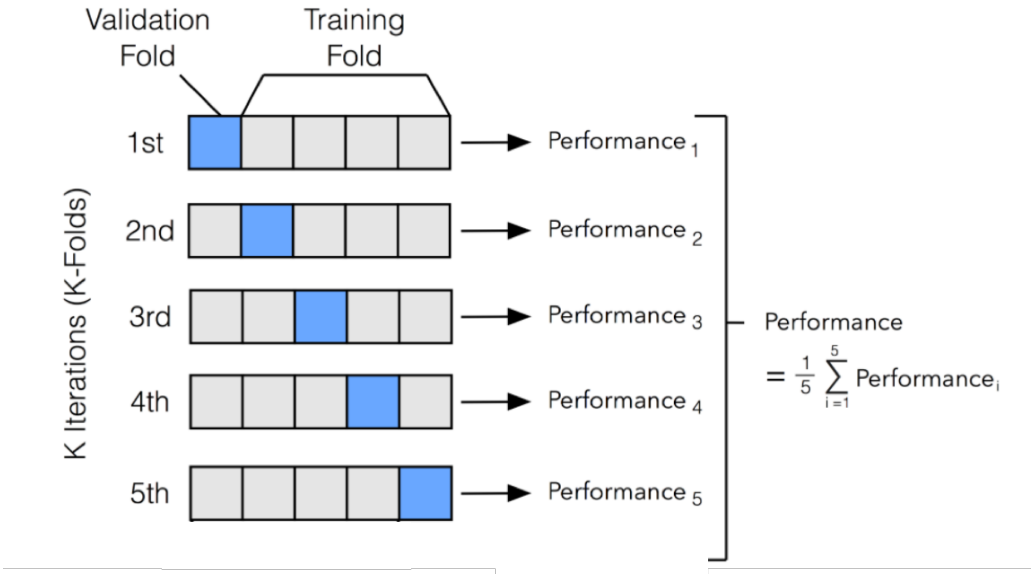

Figure 3.4: Example of 5 Fold Cross Validation.

The performance metric is the same for each session, and the way data is divided into validation and training can vary from case to case.

3.10.1 Independent Cross Validation

In this first case, we assume data are iid, and therefore it is not relevant to the way we split data. This CV method will asses the general score that is expected from the model regardless of scenario or time.

3.10.2 Scenario Cross Validation

The Scenario cross-validation will be a slight modification of the traditional version, and it is based on the idea of testing how the model behaves to a concept drift or simply in a different scenario.

3.10.3 Time Cross Validation

The last cross-validation technique will examine the time component. For each time window, we train the model with the previous transactions up to a fixed period and test in the next. This cross-validation is the most appropriate for the case of anomaly detection since it mimics real dynamics. There are 2 types of Time Cross-Validation as we see in Figure 3.5.

- Expanding.
- Sliding.

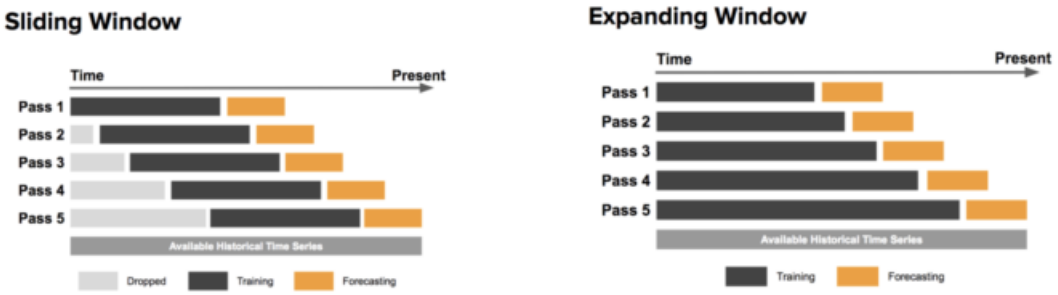

Figure 3.5: Example of Time Cross Validation.

Furthermore, we will test the performance in time and given different feature sets.

As a benchmark, we also compute random guesses and fixed predictions in order to compare the real capacity of the model to discriminate between the classes. Algorithm 1 and 2 summarize the process.

Algorithm 1 Sliding Time Cross Validation

Let $t \in T$ be a chunk of total time period of transactions

1: **for** t in T **do**
2: let $X_{t_{train}}$ be a collection of data up to t
3: iF $\leftarrow$ IsolationForest.fit($X_{t_{train}}$)
4: let $X_{t_{test}}$ be a collection of data after t
5: $AUC_t \leftarrow$ iF.evaluate($X_{t_{test}}$)
6: $t \leftarrow t$+1 period
7: **end for**

3.11 Architecture of Data Driven Model

The process of creating an anomaly score is not only a matter of statistical models, an important part is played by the flow of data and processes around the collection and use of raw data. As we can see in Figure 3.6 the model is just one part of the process. Raw data about the transaction enter into a stream of aggregation and feature engineering/ transformation labeled as ETL (Extract Transform and Load). The second part is a Scoring process, in our case we choose to work in a close-to-realtime framework. Assuming a model fitted on training

Algorithm 2 Sliding Time Cross Validation with subset features

Let $\psi \in F$ feature set according to an area of investigation
Let randomguess be a model that returns random results
Let dummymodel be a model that returns always no fraud as result
Let $\psi \in F$ feature set according to an area of investigation
for ψ in F **do**
  Initiate a Sliding Time Cross-Validation as T ordinated by time
  for t in T **do**
    let $X_{t_{train}}$ be a collection of data up to t
    iF $\leftarrow$ IsolationForest.fit($X_{t_{train}}$)
    let $X_{t_{test}}$ be a collection of data after t
    $AUC_r\psi_t \leftarrow$ randomguess.evaluate($X_{t_{test}}$)
    $AUC_d\psi_t \leftarrow$ dummymodel.evaluate($X_{t_{test}}$)
    $AUC_f\psi_t \leftarrow$ iF.evaluate($X_{t_{test}}$)
    $t \leftarrow t$+1 period
  end for
  Collect **E**[$AUC_r\psi_t$] dummymodel
  Collect **E**[$AUC_d\psi_t$] randomguess
  Collect **E**[$AUC_f\psi_t$] iF
  Slide to next ψ
end for

data, the new raw transaction pass through the ETL process and generates a payload, a theoretical application establishes a connection with a cloud service where the model is deployed (IBM Watson Machine Learning API) and score the payload. The final result is the anomaly score.

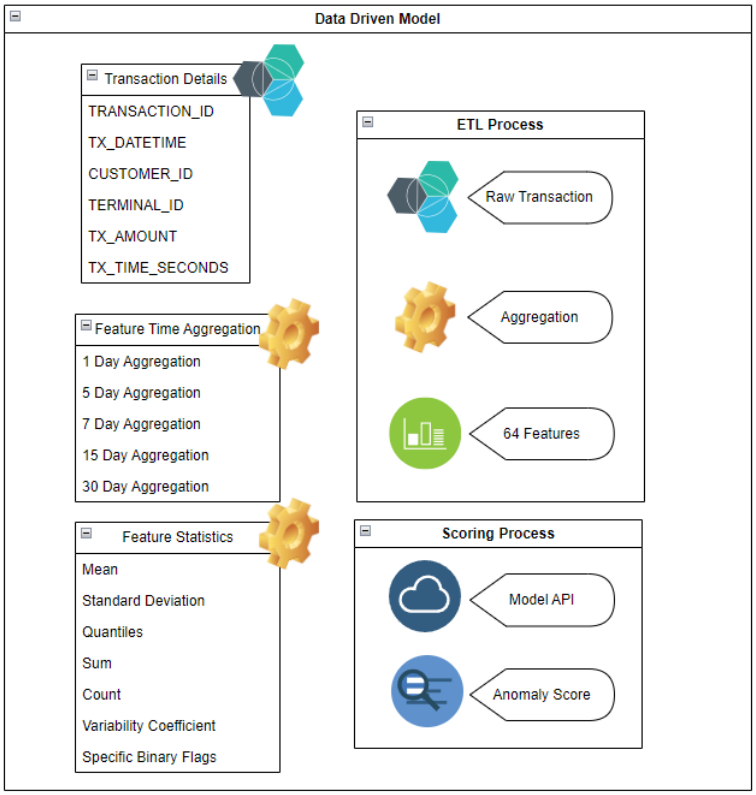

Figure 3.6: Example Data-Driven Model in Fraud Detection System.

Chapter 4

Model Results

4.1 Summary of Business Context

Before showing the model results, we now proceed to summarize all the steps in the model development and data preprocessing.

A fraud detection system is made of many layers, one of them is a data-driven model and Isolation Forest has been chosen as a possible candidate to overcome the following issues of supervised models:

- Drop in performance given a Drift in Fraud concept
- Need of a label in the training set
- Difficult Preprocessing and HyperParameter Tuning

The limitation of the approach is the required transactional history. Without this condition, we cannot build a significant set of features and therefore the model is incapable of discriminating between the classes.

To test such a model we have used a dataset of simulated transactions created by Le Borgne et al. 2022. The scenario data-set is based on

- 5000 customers
- 10000 terminals
- 183 days
- 1754155 transactions
- 14681 frauds (0.8 percent of total transactions)

The frauds are not always the same, in order to better mimic real dynamics a set of 3 scenarios have been created and randomly appear during the period of time.

- Scenario 1: Any transaction whose amount is more than 220 is a fraud. This scenario is not inspired by a real-world scenario. Rather, it will provide an obvious fraud pattern that should be detected by any baseline fraud detector. This will be useful to validate the implementation of a fraud detection technique.
- Scenario 2: Every day, a list of two terminals is drawn at random. All transactions on these terminals in the next 28 days will be marked as fraudulent. This scenario simulates a criminal use of a terminal, through phishing for example. Detecting this scenario will be possible by adding features that keep track of the number of fraudulent transactions on the terminal. Since the terminal is only compromised for 28 days, additional strategies that involve concept drift will need to be designed to efficiently deal with this scenario.
- Scenario 3: Every day, a list of 3 customers is drawn at random. In the next 14 days, 1/3 of their transactions have their amounts multiplied by 5 and marked as fraudulent. This scenario simulates a card-not-present fraud where the credentials of a customer have been leaked. The customer continues to make transactions, and transactions of higher values are made by the fraudster who tries to maximize their gains. Detecting this scenario will require adding features that keep track of the spending habits of the customer. As for scenario 2, since the card is only temporarily compromised, additional strategies that involve concept drift should also be designed.

The analysis assessed the ability of Isolation Forest to

- Discriminate between classes
- Remain stable in time
- Perform according to a specific Fraud Scenario

Furthermore, different features have been tested to see the performance of the model given different areas of investigation.

- Customer Spending Behaviour
- Terminal Behaviour
- Pattern in Transaction Time

4.2 Results of Time Cross Validation

The model has been validated using a time-rolling window in which, the first 2 months are used for training the next 1 month is used for testing. The results are encouraging, as we see in Figure 4.1 and Figure 4.2.

The first model was a random guess and the AUC of 0.5 clearly indicates a poor classification performance, the second model called dummy model is trying to exploit the imbalanced proportion between the classes by always predicting 0. The dummy model is also incapable of distinguish between the classes given an AUC of 0.5. Isolation Forest is clearly the better model with a performance of 0.87.

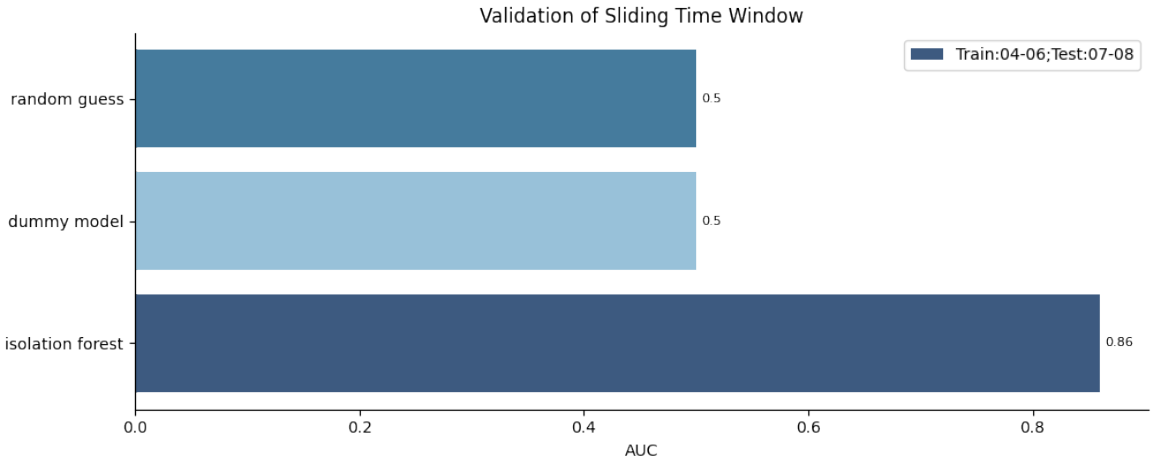

Figure 4.1: AUC of Model in Train and Test Period

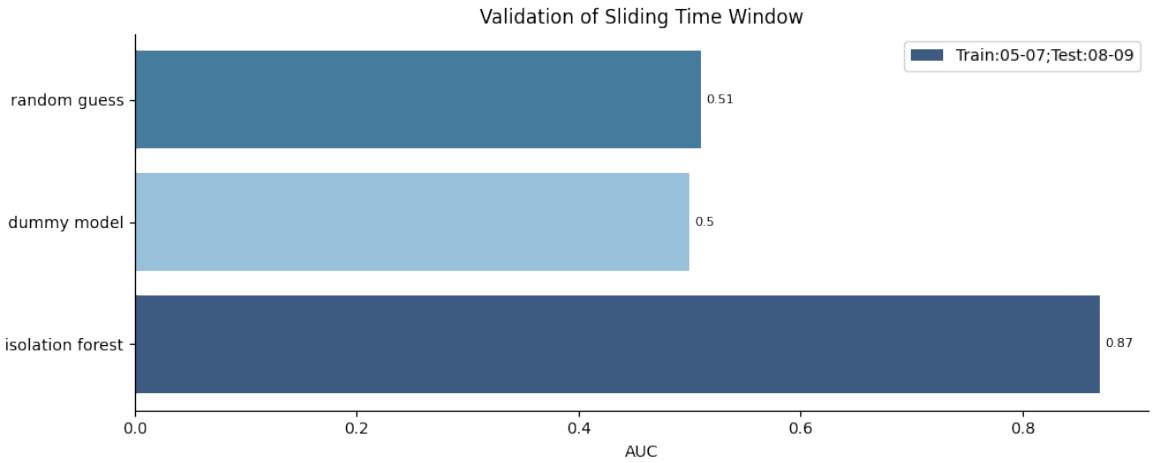

Figure 4.2: AUC of Model in Train and Test Period

4.3 Results of Fraud Scenario Analysis

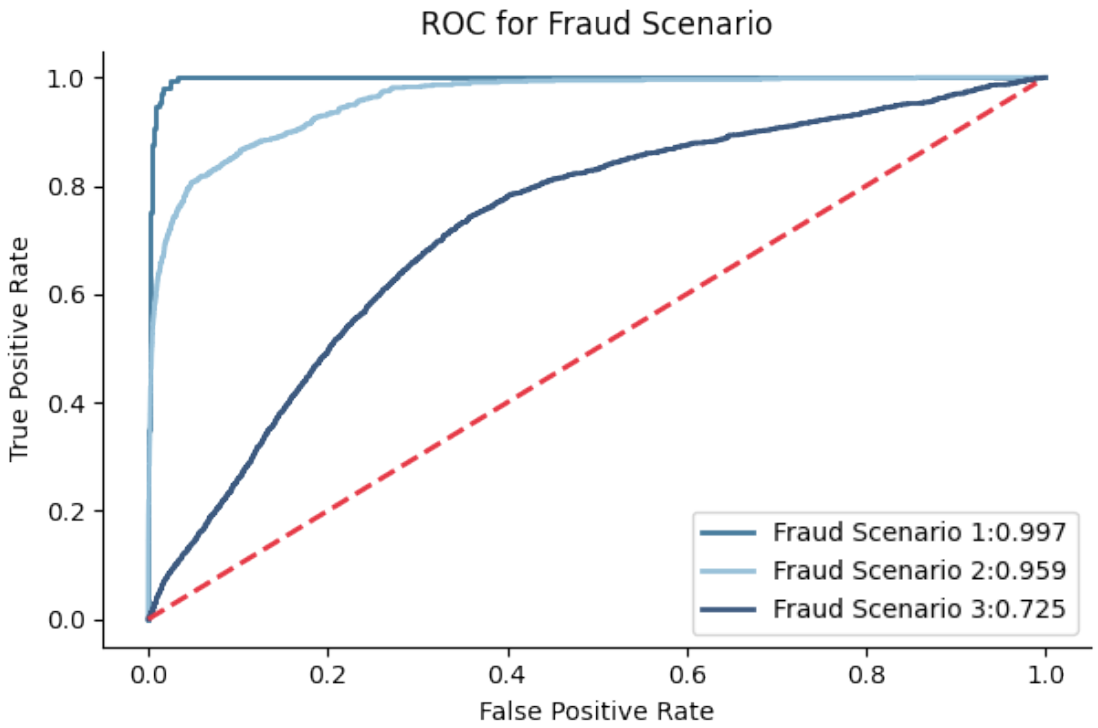

Figure 4.3: ROC for all the Fraud Scenarios

A further argument of discussion was the ability of the model to adapt to specific scenarios. As we see in Figure 4.3 the model perfectly adapts to the first scenario with an AUC of 0.99, which confirms the ability of the model to recognize simple patterns in payments. The second and the third case are progressively more challenging and the performance is still good, with an AUC of 0.95 and 0.72. These results are evidence that Isolation Forest can actually compete with supervised models and overcome their issues. Furthermore, it can be implemented as a meta-model for other supervised techniques.

4.4 Results of Feature Validation

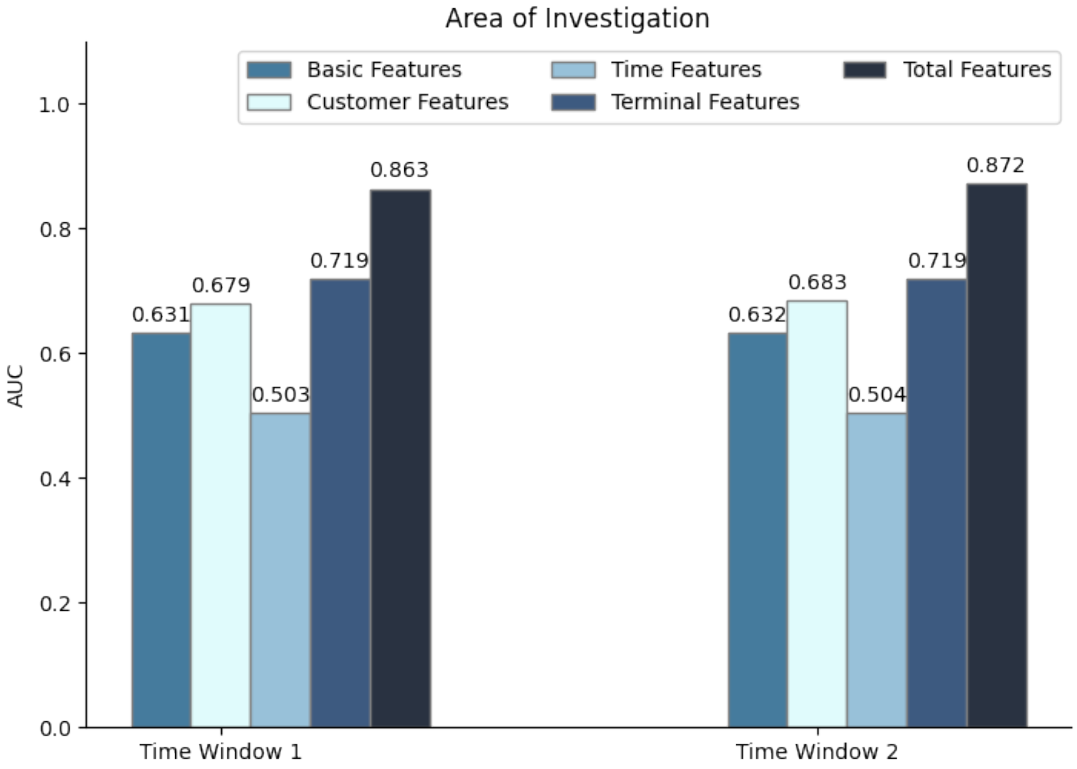

Figure 4.4: Summary of Model Validation for time and features

As we can see in Figure 4.4 the model is stable in performance across time and features. The 2-time windows are created using the period between April and September and they will give and idea of how stable is the model in two different time scenarios.

- Basic Features regarding TX-Amount and Time of TX performs moderately well in both time periods with an overall AUC around 0.63

- Customer Spending Features performs well in both time period, with an overall AUC around 0.68

- Time Features performances are stable in both periods and around 0.50 overall, which is an indication of bad ability in detecting anomalies

- Terminal Features perform well in both time periods and demonstrate a good ability in detecting anomalies with an overall AUC around 0.72

The best model performances across time are given by a mix of all area of investigation, with a stable overall AUC of 0.86 around the 2 periods.

4.5 Real Time Example on random User

Figure 4.5 is linked to the project with code for development and an end-to-end example of model training on historical data and a stream of scoring for simulated transactions.

Figure 4.5: QR Code to Project

The Figure 4.6 shows the result of the architecture defined in Figure 3.6 applied to a random User N:

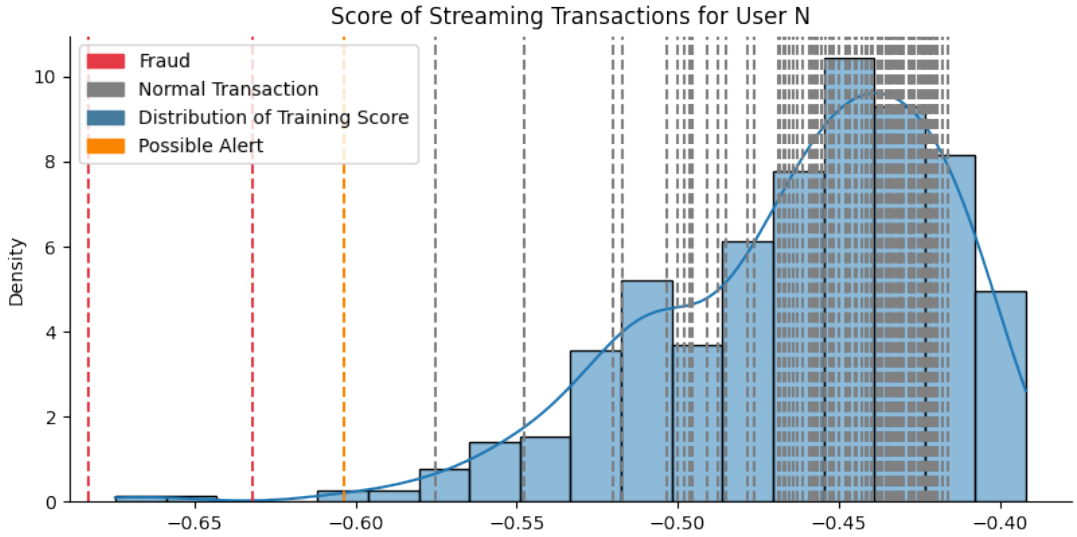

Figure 4.6: End-to-end streaming of score and transaction with Isolation Forest

We can see the scores created from training have indeed the desired skewed distribution defined in the chapter 2, Figure 2.8, and each time a transaction takes place it enters the ETL and Scoring process. The color of each transaction tell us if the current transaction is a fraud, since we know the ground truth also for test data. The result is that the majority of normal transactions are concentrated around the peak of the training distribution, while frauds are in the tail.

The chosen possible alert, contamination, is the value for which we decide if the current transaction can be considered normal or not, and the change of such value do not required a refit process.

Chapter 5

Conclusion

Developing an effective fraud detection system is a complex process that typically requires more than one technique. The majority of tools fully approved by regulators are explicit rules, often inconsistent with the fast pace of time and technology. Although simulated transactions are purely demonstrative, they are a valuable tool for experimenting new techniques and support the development of fraud detection models. Investigators should supervise the development of new rules and features to spot anomalies, and seek continuous feedback from users and ex former fraudsters. As part of a Data Driven Layer, Isolation Forest can help existing methods to perform better and avoid explicit rules that might be fooled by more experienced fraudsters. Creating Features related to customer spending and information about previous risks over time seems a key element also in difficult fraud scenarios and prevent a drift in performance. Class Imbalance seems also a problem for many supervised models, and isolation forest performed well and stayed consistent. Furthermore, the use of a flexible alert can be of a good use in moments when policy makers or management requires more attention or less friction in the payment system. Investing in data collection and cloud computing solutions can help minimize the latency between registration and validity of transactions. However, it is essential to ensure data protection and address the trade-off between compliance and efficiency. [Harish and Santhoshkumar 2021]

Overall, the field of fraud detection in the banking sector is rapidly evolving, and continued research and development are necessary to stay ahead of fraudsters and protect the financial system.

Bibliography

Baesens, Bart, Veronique Van Vlasselaer, and Wouter Verbeke (2015). *Fraud analytics using descriptive, predictive, and social network techniques: a guide to data science for fraud detection*. John Wiley & Sons.

Galli, Soledad (2020). *Python Feature Engineering Cookbook: Over 70 recipes for creating, engineering, and transforming features to build machine learning models*. Packt Publishing Ltd, 2020.

Hariri, Sahand, Zhihua Chen, and Huan Liu (2018). "Extended Isolation Forest". *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, pp. 1097–1102.

Harish, NA and R Santhoshkumar (2021). "Fraud detection in the banking sector using machine learning: A systematic literature review". *Journal of Financial Crime*.

Hastie, Trevor et al. (2009). *The elements of statistical learning: data mining, inference, and prediction*. Vol. 2. Springer.

James, Gareth et al. (2013). *An introduction to statistical learning*. Vol. 112. Springer.

Le Borgne, Yann-Aël et al. (2022). *Reproducible Machine Learning for Credit Card Fraud Detection - Practical Handbook*. Université Libre de Bruxelles.

Liu, Fei Tony, Kai Ming Ting, and Zhi-Hua Zhou (2008). "Isolation Forest", pp. 413–422.

Rousseeuw, Peter J (1987). "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis". *Journal of computational and applied mathematics* 20, pp. 53–65.

Westphal, Christopher (2008). *Data Mining for Intelligence, Fraud & Criminal Detection: Advanced Analytics & Information Sharing Technologies*. CRC Press.