Chapitre 5 **Réalisation**

Ce dernier chapitre mettra fin à cet ouvrage, il représente implémentation et la réalisation du projet. Tout d’abord, nous présentons les outils et technologies utilisées, puis nous allons présenter les principales interfaces graphiques de notre application.

5.1 **Environnement de travail**

Dans cette partie, nous détaillerons, les différentes caractéristiques qui constituent l’environnement spécifique de travail à savoir l’environnement logiciel et matériel.

**5.1. Environnement Matériel :**

De point de vue matériel, nous avons utilisé un ordinateur portable dont la configuration est présentée dans le tableau suivant

**Une image contenant table

Description générée automatiquement**

Environnementmatériel

5.2 Environnement logiciel

Dans cette section, nous énumérons les différents outils et technologies que nous allons utiliser au cours de la réalisation du système.

Outils

Notre application a été mise en œuvre moyennant les outils logiciels suivants :

* Microsoft Visual Studio 2018 Professional, c’est un éditeur de code extensible développé par Microsoft
* Microsoft SQL Server qui est un système de gestion de base de données en langage SQL.
* StarUml, c’est un logiciel de modélisation UML.
* Spring Tool suit (STS) est un IDE étendu pour Eclipse, il se spécialise dans le développement des applications JEE.
* PowerBI : est une solution d'analyse de données de Microsoft. Il permet de créer des visualisations de données avec une interface suffisamment simple.
* Talend : Talend est un éditeur de logiciel spécialisé dans l'intégration de données.
* JasperSoft studio : est un générateur de rapports de développement open source.
* Pentaho workbench : c’est une interface graphique permettant de produire des schémas Mondrian pour les analyses de données OLAP (Online Analytical Processing).

Framework et Technologies

* Springboot : est un micro-Framework open source, il fournit aux développeurs Java une plate-forme pour démarrer avec une application Spring configurable automatiquement.
* Angular : est un Framework Javascript côté client qui permet de réaliser des applications de type "Single Page Application". Il est basé sur le concept de l'architecture MVC.

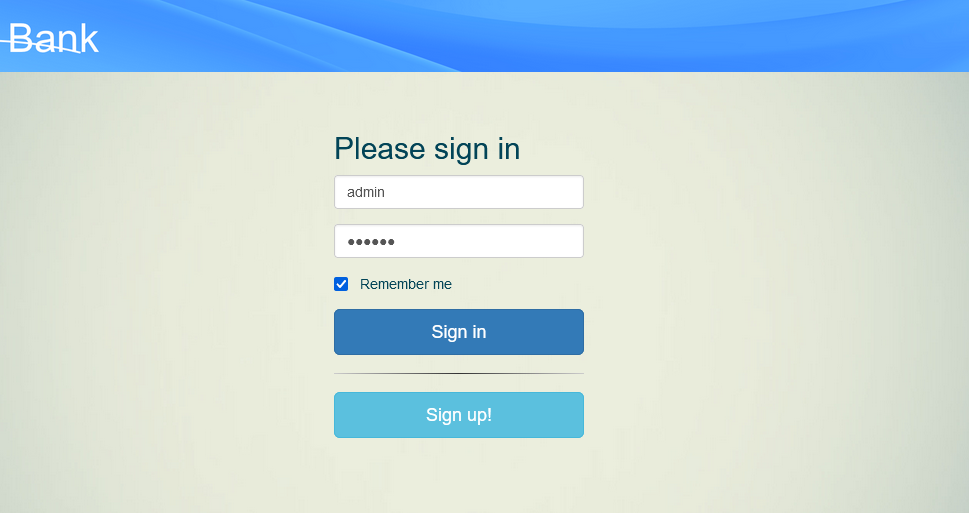
**Travail réalisé**

Pour illustrer le travail réalisé, nous allons montrer les étapes suivies et des interfaces Homme/Machine dans notre application.

1. **Partie dev :**

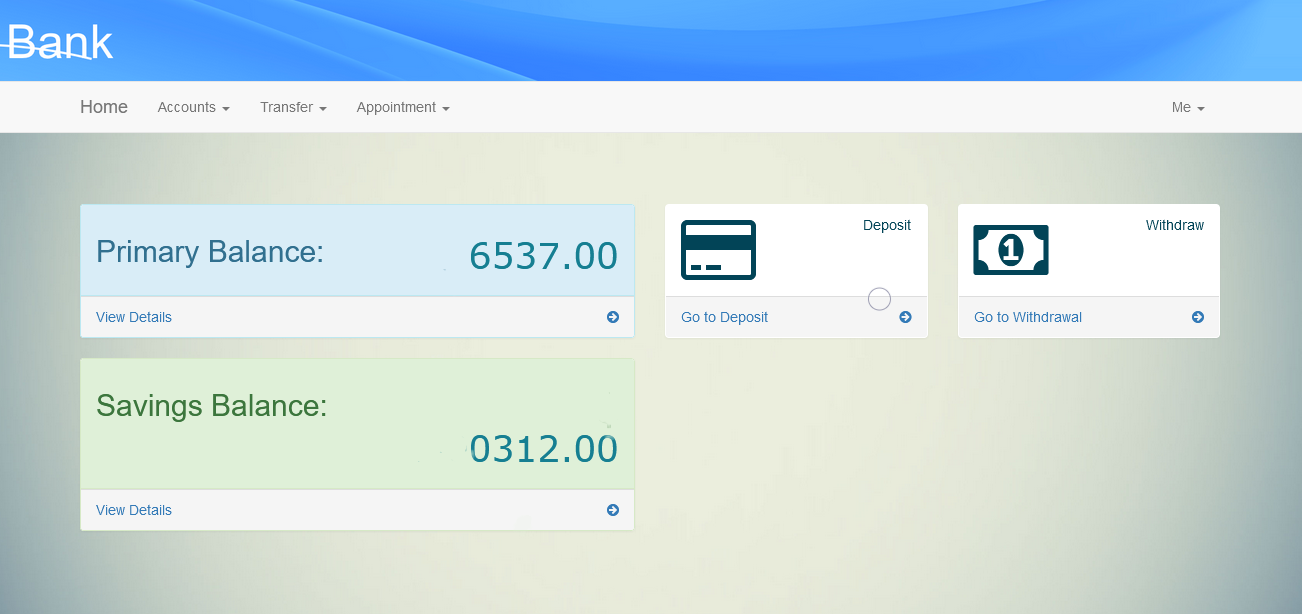
Application bancaire + dash + charts + email notification

La figure suivante illustre la page de d’authentification. Ici, nous demandons à l'utilisateur de mettre son login et mot de passe.



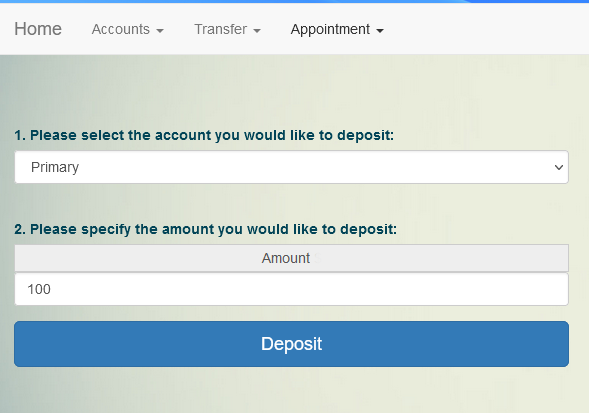
Authentification

La figure suivante montre la page d’accueil de notre application, qui permet le dépôt, le transfert d’argent aussi montre la somme d’argent disponible.



Page d’accueil

Cette interface indique le compte et le montant déposé



Dépôt d’argent

Cette interface présente l’historique des transactions avec leur détailles.

Une image contenant table

Description générée automatiquement

Historique des transactions

La figure suivante montre le tableau de bord de la banque avec des différents graphiques pilotage qui permettent aux gestionnaires de prendre connaissance de l’état de l’évolution des systèmes qu’ils pilotent et d’identifier les tendances.

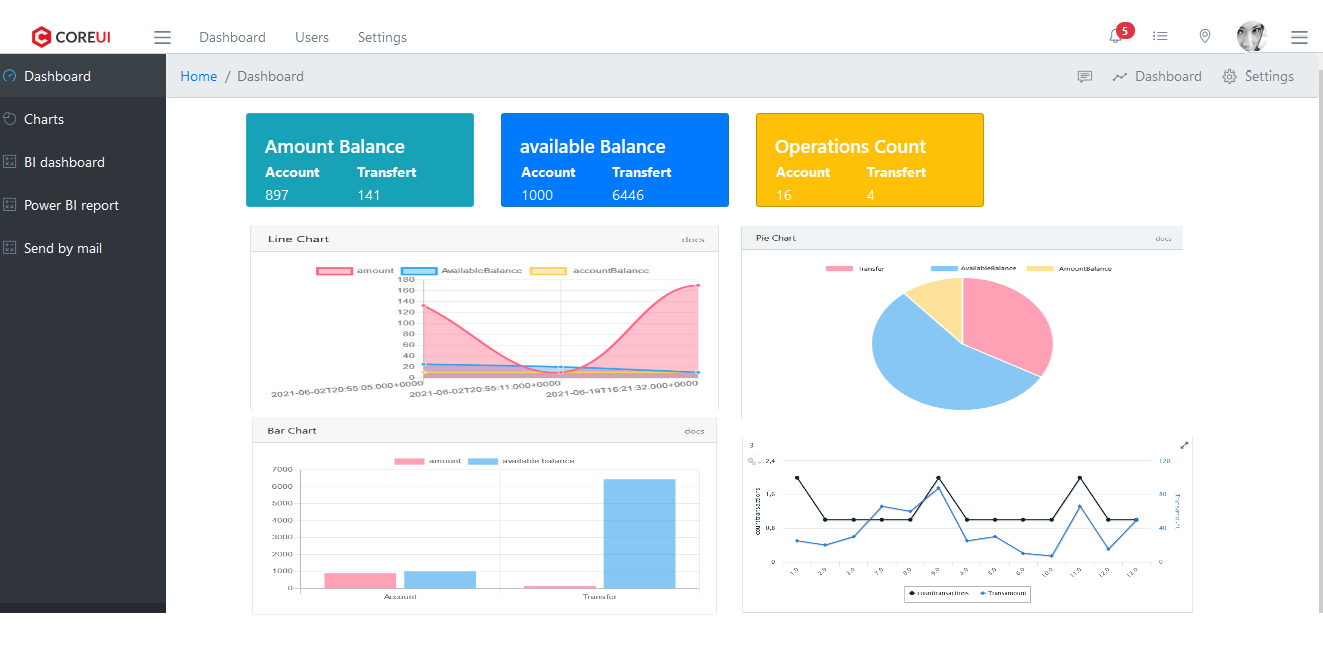


Tableau de bord de la banque

1. **Partie Bi :**

**ETL**

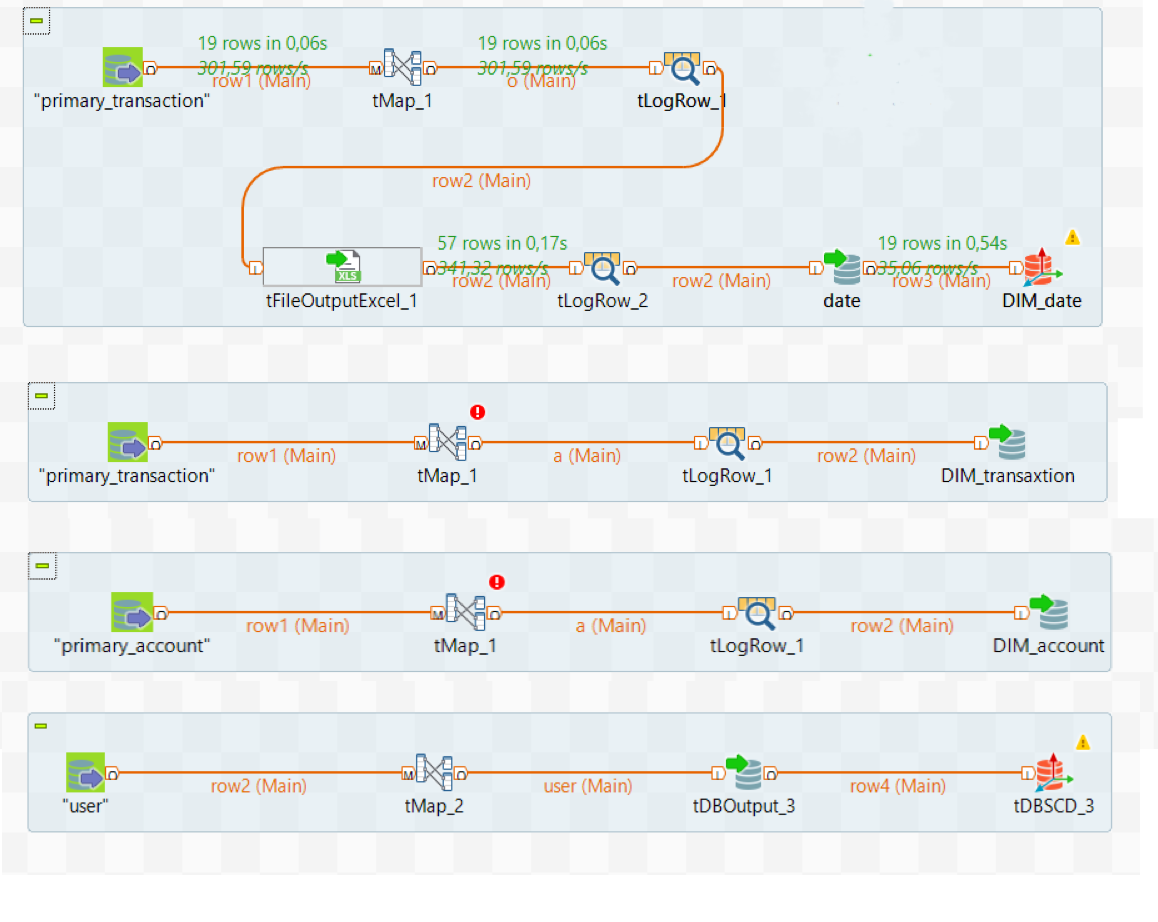
L’implémentation ETL consiste à mettre en place ce qu’on appelle en nomenclature (instance de classification) de Talend des « jobs » permettant d’extraire les données à partir de la base sources, d’effectuer les transformations nécessaires et de les charger dans le Data Warehouse.

1. Extraction et transformation des données

Au cours de cette étape on a eu recours à l’extraction et à la transformation, l’extraction des données se fait par connexion avec l’entrepôt de données avec Talend après, nous procédant à la régulation et la correction des données pour éviter les éventuels problèmes, parmi ces derniers on peut citer :

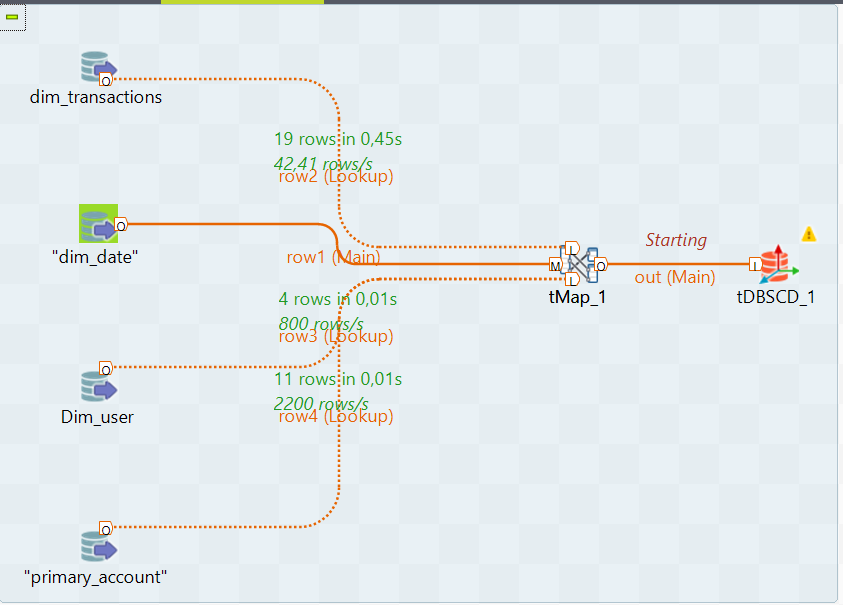
Le séparateur virgule qui engendrera beaucoup de problèmes au cours des prochaines étapes et pour cela nous avons remplacé les virgules par des points, aussi nous avons remarqué une erreur au niveau de la date, qui se présente sous différent format comme dd/mm/yyyy ou mm/dd/yyyy, donc nous avons standardisé ce format.

Après nous avons créé les dimensions d’analyse du dataware house, les figures ci-dessous représentent le travail réalisé d’extraction et de transformation de données avec le logiciel Talend.



1. **Chargement des données**

Après toutes les extractions et transformations effectuées dans les étapes précédentes de l’ETL, les données seront chargées dans notre dataware house qui est installé sur un serveur de base de données, la figure suivante est le schéma relatif au job d’alimentation de la table de fait de notre entrepôt de données.



1. Construction du Cube OLAP

Le Cube OLAP est un tableau multidimensionnel de données qui offre la possibilité de calculs complexes d'analyse des tendances et de modélisation sophistiquée des données.

Nous avons utilisé Pentaho workbench qui c’est une interface graphique permettant de produire des schémas Mondrian pour les analyses de données en multidimension, la figure suivante montre le cube réalisé avec ses différents dimensions et mesures.

Une image contenant texte

Description générée automatiquement

1. Restitution des données

C’est une étape très importante dans notre projet car nous allons visualiser le résultat de notre travail effectuer en illustrant des analyses et des visualisations. Ci-dessous quelques exemples de rapports et tableaux de bord que nous avons réalisés dans notre projet en utilisant Power BI et Jasper Soft studio.

Nous présentons par les figures suivantes quelques exemples de tableau de bord et de rapports élaborés relatifs à l’activité bancaires.

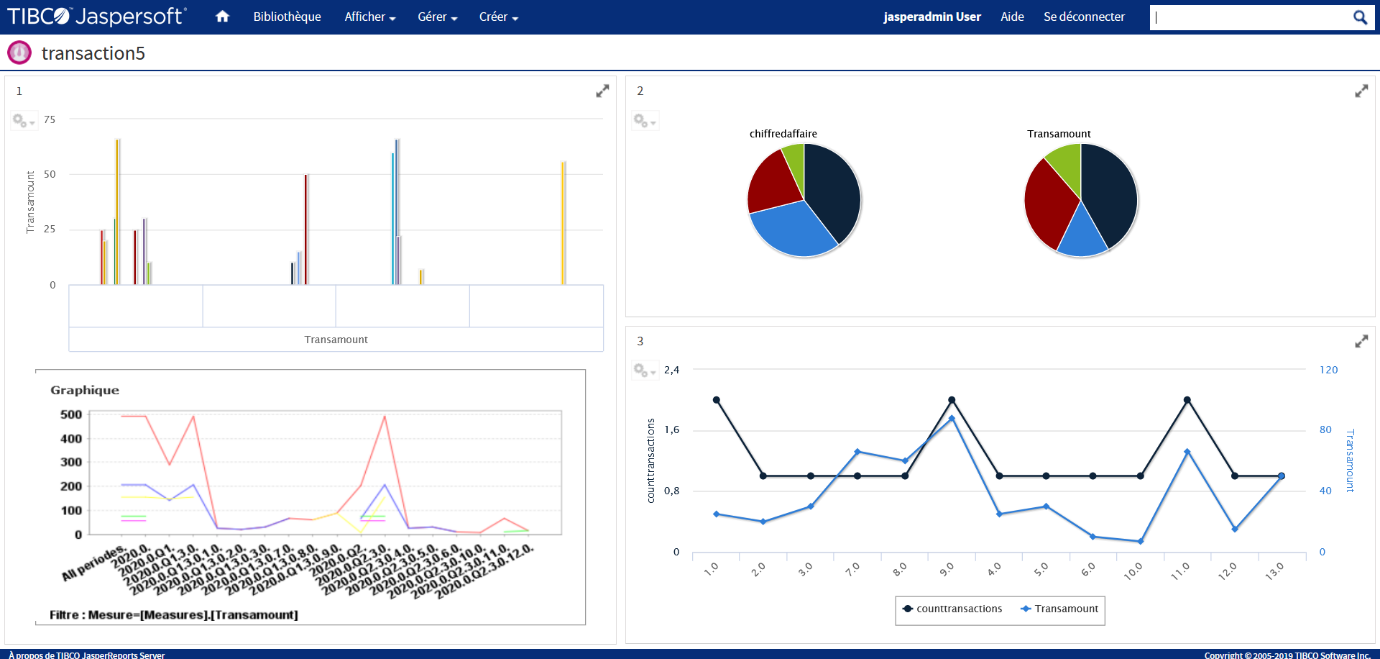
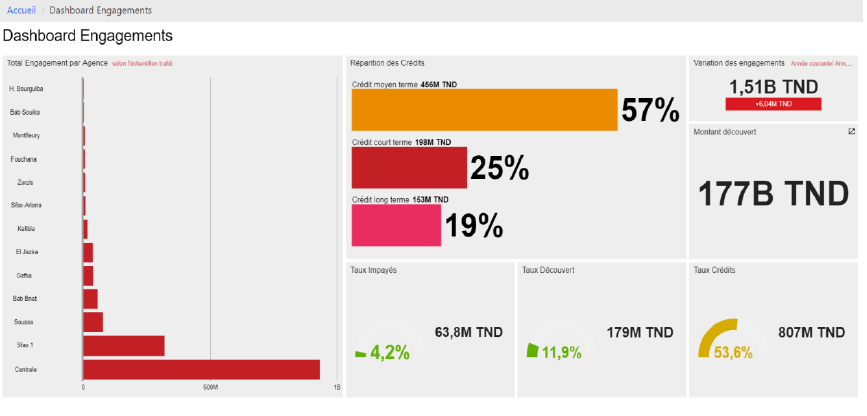


Tableau de bord pour l’évolution des transactions en cours du temp

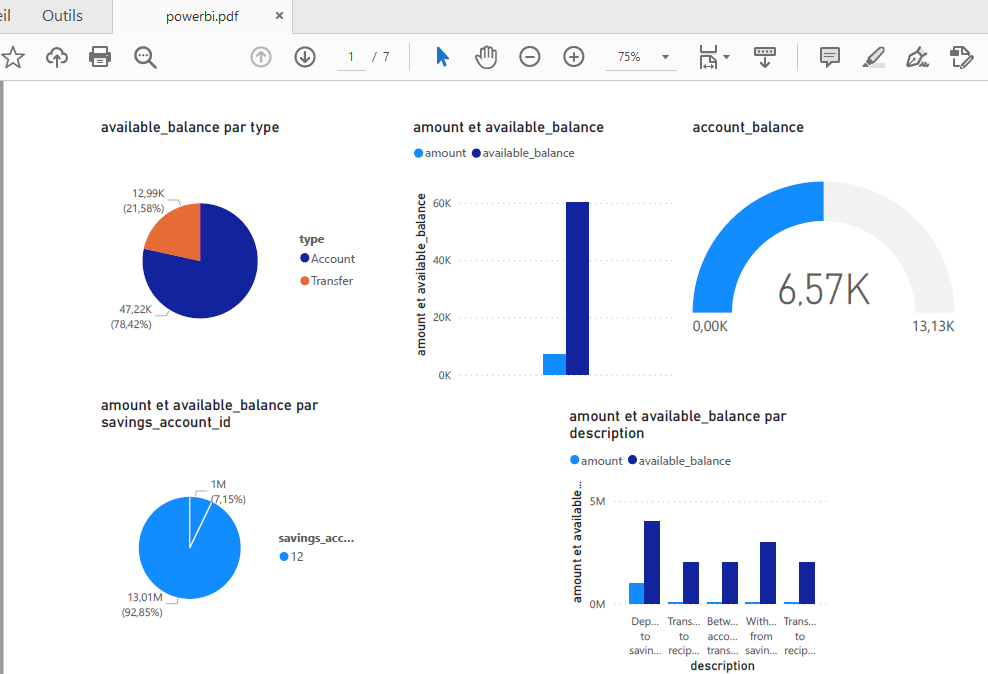
Tableau de bord pour engagement



Une image contenant table

Description générée automatiquement

Nous présentons dans la figure suivante un aperçu du rapport d’analyse des transferts bancaires exporter sous extension PDF et en PNG et qui sont envoyées automatiquement par e-mail aux adresses des exploitants.



1. Partie détection de Fraude :

Le problème à résoudre dans cette partie et de détecter les transactions frauduleuses à l’aide des modèles d'apprentissage automatique.

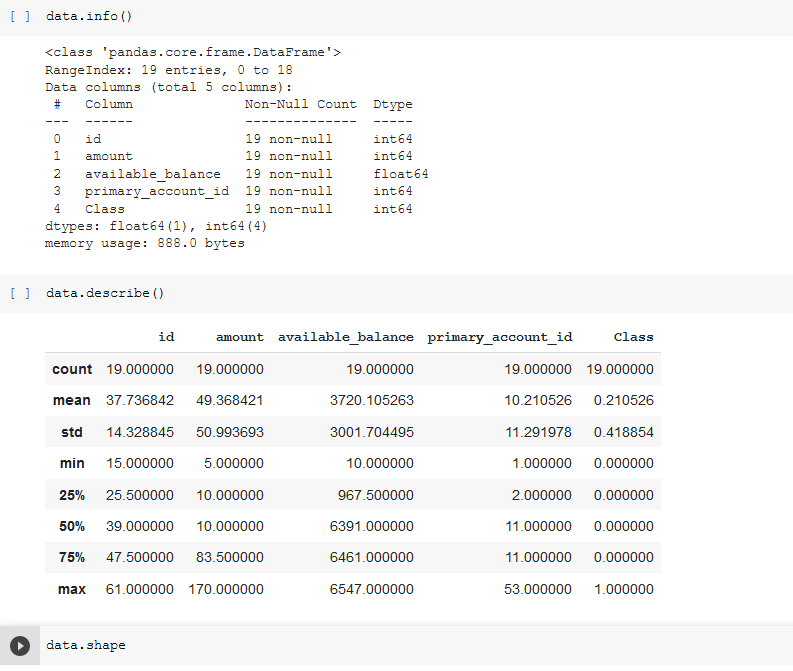
Les transactions frauduleuses constituent une menace importante pour les banques et les clients.

## 

Dans ce qui suit nous allons suivre des étapes ordonnées allons de l’exploration des donnés jusqu’à l’évaluation du modèle passant par des étapes primordiales.

1. Compréhension et exploration des données

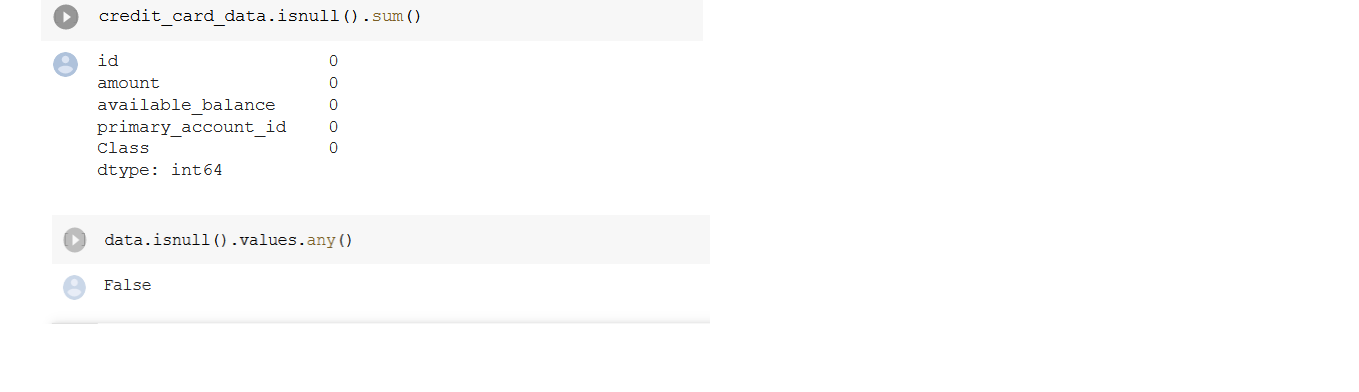
Ici, nous devons charger les données et comprendre les fonctionnalités qui y sont présentes, les données sont chargées à partir de notre base de données de la banque grâce à la librairie Pandas de Python, la figure suivante nous informe sur les données traitées.



### Nettoyage des Données

En science de données, les jeux de données comportent souvent des irrégularités et des erreurs. Ces dernières peuvent être des données manquantes ou aberrantes. Savoir traiter ces données permettra de produire un modèle précis et efficace.

Les données que nous avons choisies d’utiliser sont peu susceptibles d’être parfaitement propres (sans erreurs), dans la figure suivante nous avons vérifié s’il existe des valeurs manquantes.



1. **Analyse des Données :**

Dans cette partie on observe la distribution de nos classes et leur symétrie grâce aux barres graphique qui montre le nombre et le pourcentage des transactions frauduleuses vs non frauduleuses.

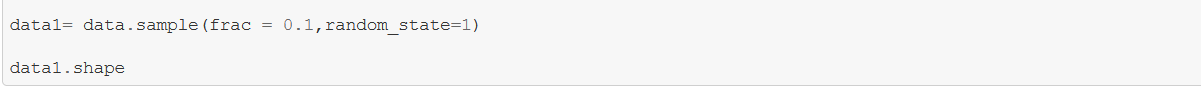
D’après la figure suivante on constate que les classes sont très déséquilibré.

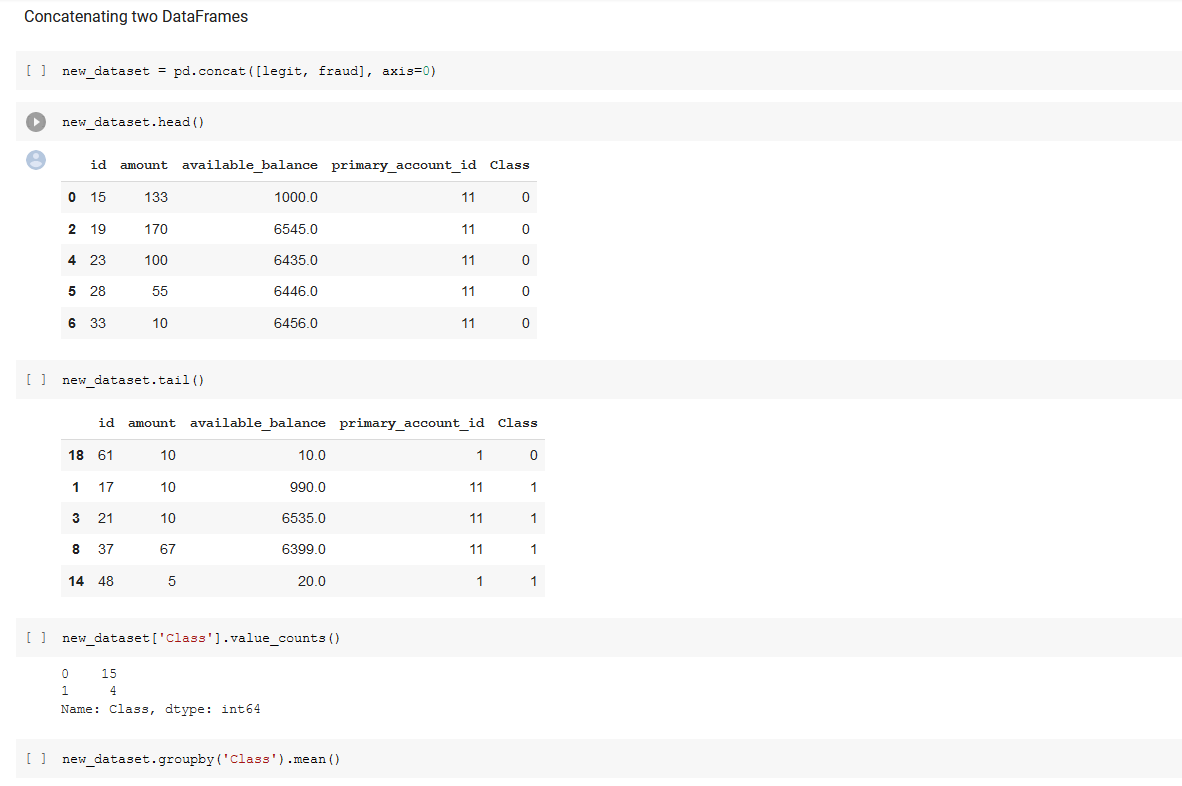


1. **Préparation les données pour la modélisation**

Afin de faire face au déséquilibre des classes utilisées nous allons procéder à la séparation des transactions normal et frauduleuse puis l’échantillonnage et enfin la concaténation des données comme le montre la figure suivante

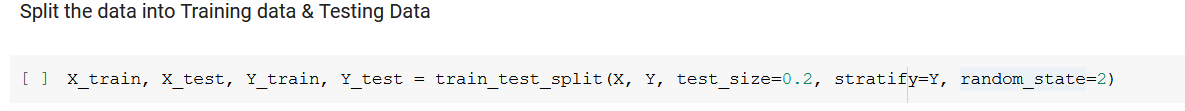






1. Split the data into train and test set

Train/Test Split: Now we are familiar with the train/test split, which we can perform in order to check the performance of our models with unseen data. Here, for validation, we can use the k-fold cross-validation method. We need to choose an appropriate k value so that the minority class is correctly represented in the test folds.



To measure model performance we’ve split the data into two chunks.

80% for Training the model

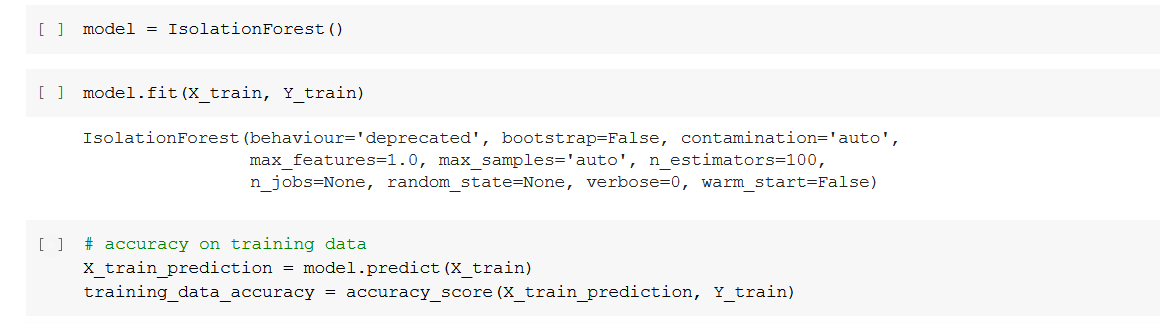
20% for Testing our model

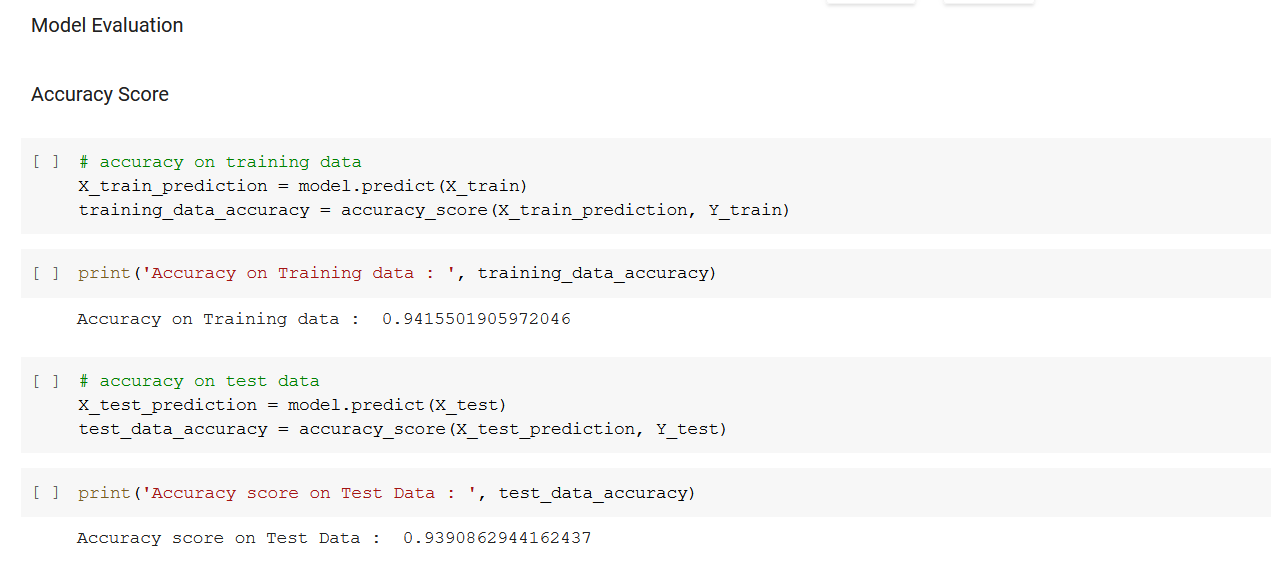
1. Model building

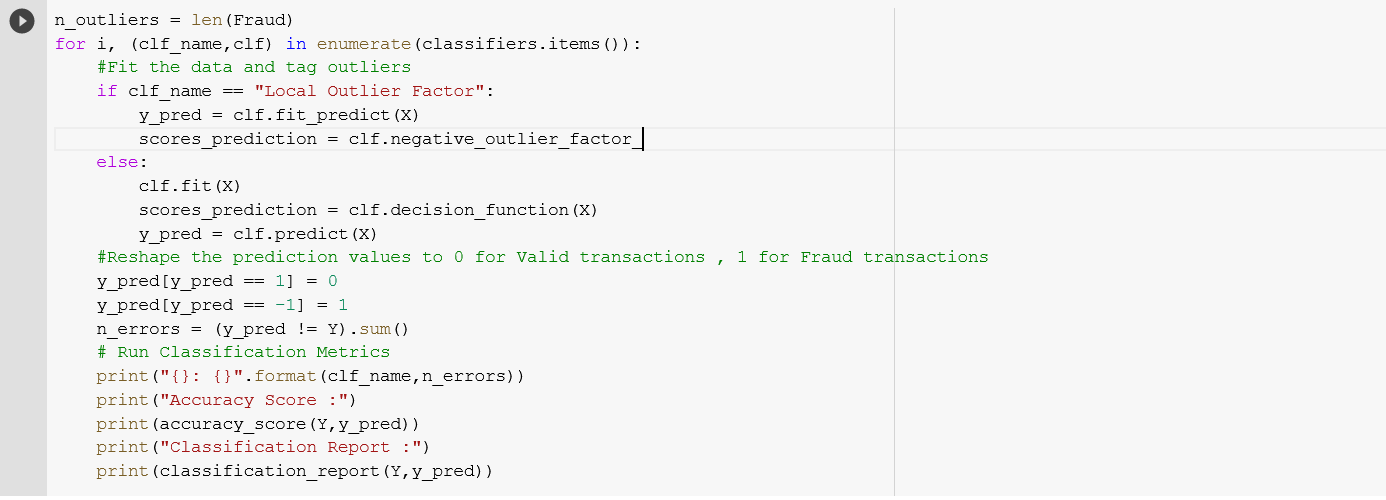
Model-Building/Hyperparameter Tuning: This is the final step at which we can try different models and fine-tune their hyperparameters until we get the desired level of performance on the given dataset. We should try and see if we get a better model by the various sampling techniques.

Une image contenant texte

Description générée automatiquement



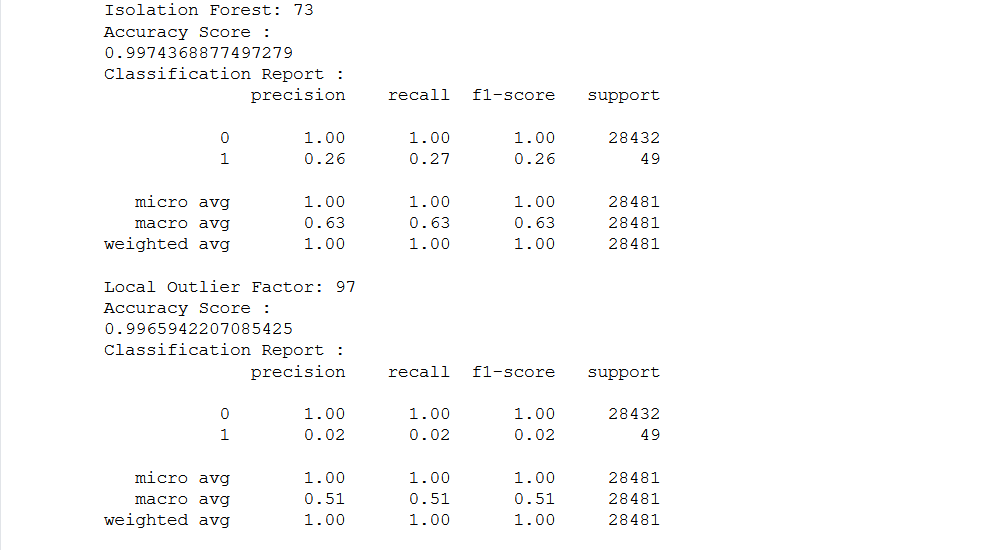




Model Performance – Metrics

we’ve identified the following metrics to measure model performance.

* Precision  
  Fraction of transactions the model classifies as fraud that are actually fraudulent.  
  Precision = TP / (TP + FP)
* Recall  
  Fraction of all fraud that is identify by our model.  
  Recall = TP / (TP + FN)
* F-Measure  
  A good measure to use when we seek a balance between precision and recall and the there is an uneven class distribution.  
  F-Measure = 2 \* Precision \* Recall / (Precision + Recall)



* Train the model with various algorithm such as Logistic regression, SVM, Decision Tree, Random forest, XGBoost etc.
* Tune the hyperparameters with Grid Search Cross Validation and find the optimal values of the hyperparameters

Now it is time to start building the model .The types of algorithms we are going to use to try to do anomaly detection on this dataset are as follows

### Isolation Forest Algorithm :

One of the newest techniques to detect anomalies is called Isolation Forests. The algorithm is based on the fact that anomalies are data points that are few and different. As a result of these properties, anomalies are susceptible to a mechanism called isolation.

This method is highly useful and is fundamentally different from all existing methods. It introduces the use of isolation as a more effective and efficient means to detect anomalies than the commonly used basic distance and density measures. Moreover, this method is an algorithm with a low linear time complexity and a small memory requirement. It builds a good performing model with a small number of trees using small sub-samples of fixed size, regardless of the size of a data set.

Typical machine learning methods tend to work better when the patterns they try to learn are balanced, meaning the same amount of good and bad behaviors are present in the dataset.

How Isolation Forests Work The Isolation Forest algorithm isolates observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature. The logic argument goes: isolating anomaly observations is easier because only a few conditions are needed to separate those cases from the normal observations. On the other hand, isolating normal observations require more conditions. Therefore, an anomaly score can be calculated as the number of conditions required to separate a given observation.

The way that the algorithm constructs the separation is by first creating isolation trees, or random decision trees. Then, the score is calculated as the path length to isolate the observation.

### Local Outlier Factor(LOF) Algorithm

The LOF algorithm is an unsupervised outlier detection method which computes the local density deviation of a given data point with respect to its neighbors. It considers as outlier samples that have a substantially lower density than their neighbors.

The number of neighbors considered, (parameter n\_neighbors) is typically chosen 1) greater than the minimum number of objects a cluster has to contain, so that other objects can be local outliers relative to this cluster, and 2) smaller than the maximum number of close by objects that can potentially be local outliers. In practice, such informations are generally not available, and taking n\_neighbors=20 appears to work well in general.

1. Model evaluation

Model Evaluation: We need to evaluate the models using appropriate evaluation metrics. Note that since the data is imbalanced it is is more important to identify which are fraudulent transactions accurately than the non-fraudulent. We need to choose an appropriate evaluation metric which reflects this business goal.

#### Observations :

* Isolation Forest detected 73 errors versus Local Outlier Factor detecting 97 errors vs. SVM detecting 8516 errors
* Isolation Forest has a 99.74% more accurate than LOF of 99.65% and SVM of 70.09
* When comparing error precision & recall for 3 models , the Isolation Forest performed much better than the LOF as we can see that the detection of fraud cases is around 27 % versus LOF detection rate of just 2 % and SVM of 0%.
* So overall Isolation Forest Method performed much better in determining the fraud cases which is around 30%.
* We can also improve on this accuracy by increasing the sample size or use deep learning algorithms however at the cost of computational expense.We can also use complex anomaly detection models to get better accuracy in determining more fraudulent cases

1. Partie Bi :

ETL + Cube datawarehouse + dashbord

1. Partie Fraude detection :

Data understanding and exploring

Data cleaning.

Exploratory data analysis

Prepare the data for modelling

Split the data into train and test set

Model building

Model evaluation