**Leveraging Machine Learning for**

**Credit Card Fraud Detection**

**Team Members:**

* Nguyễn Thị Dung
* Nguyễn Quốc Anh
* Nguyễn Quốc Bảo
* Trần Trương Minh Thắng
* Lê Thảo Vy

TABLE OF CONTENTS

CHAPTER I: INTRODUCTION

1. PROBLEM DEFINITION
2. THE PURPOSE OF PROJECT
3. THE SCOPE OF PROJECT
4. DATA SOURCE
5. TECHNICAL APPLICATION

CHAPTER II: PROJECT DESIGN: FROM DATA EXPLORATION TO MODELING

1. Exploratory Data Analysis (EDA)
2. (phần này viết về pyspark) → Cua anh thang
3. Modeling Strategies

CHAPTER III: DATA VISUALIZATION

CHAPTER IV: LIMITATION

REFERENCE

# CHAPTER I: INTRODUCTION

1. **Problem Definition**

Aligned with the introduction of the credit card which is considered to be much more convenient and time-saving than the traditional one, the concerns itself over the fraud and unethical hacking have been highlighted due to its frequent occurrence of credit card fraud cases. This scenario has paved the way for an idea of developing an application of credit card fraud detection as a responsiveness fighting back the cybercrime.

The basic concepts of this credit card fraud detection is based on the use of Big Data Analytics empowering companies to analyze large volumes of data accumulated during financial transactions and other locale – specific information. In this context of the study, Big Data tools are employed to defend against cyber-attacks, facilitating the detection of fraud, identification theft, and aiding in digital forensic analysis.

1. **The Purpose of the Project**

The primary goal of this project is to leverage machine learning to create an effective and reliable credit card fraud detection system. By analyzing transaction data and recognizing patterns indicative of fraud, the system aims to:

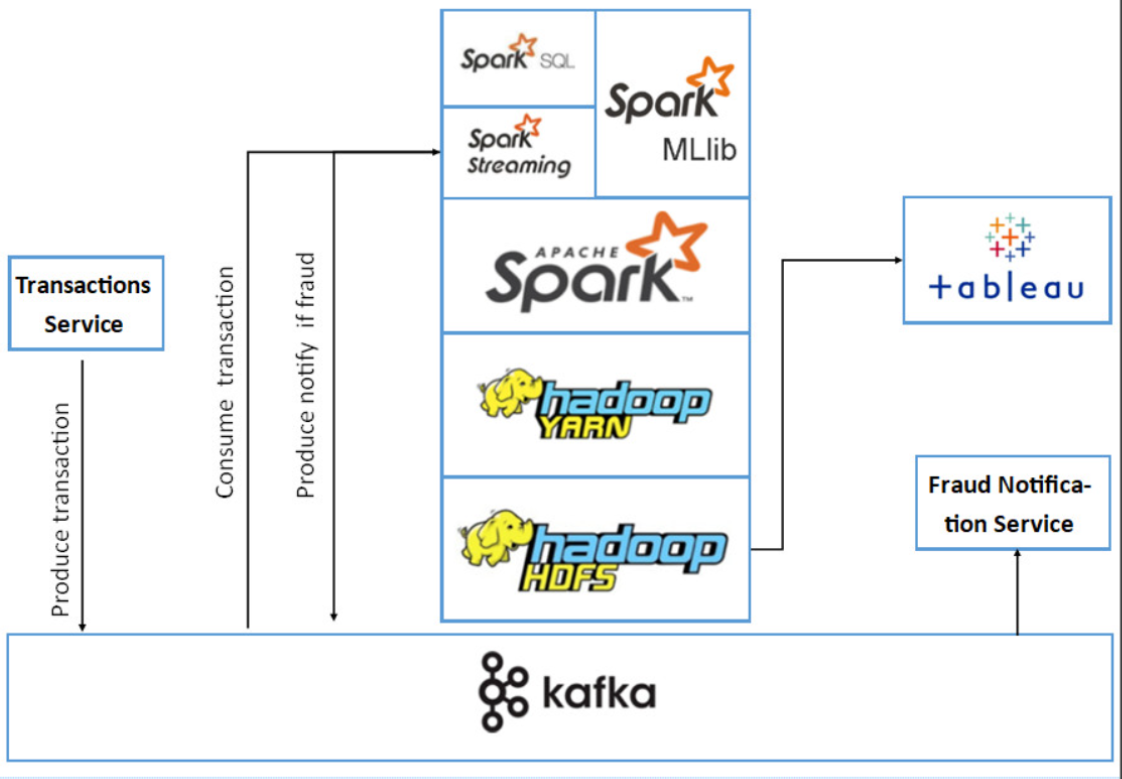
* Identify fraudulent transactions with high accuracy.
* Minimize false positives, ensuring that legitimate transactions are not erroneously flagged.
* Provide actionable insights to financial institutions, enabling prompt response.
* Contribute to the overall security and integrity of financial systems.

1. **The Scope of the Project**

The scope of this project encompasses several key areas:

Target Audience: The intended users of this system include banks, credit card companies, security experts, and data analysts involved in fraud detection.

Project Boundaries: The project is confined to the analysis of a specific dataset of credit card transactions. While the methodologies could be applied elsewhere, the models are tailored to the characteristics of this particular dataset.



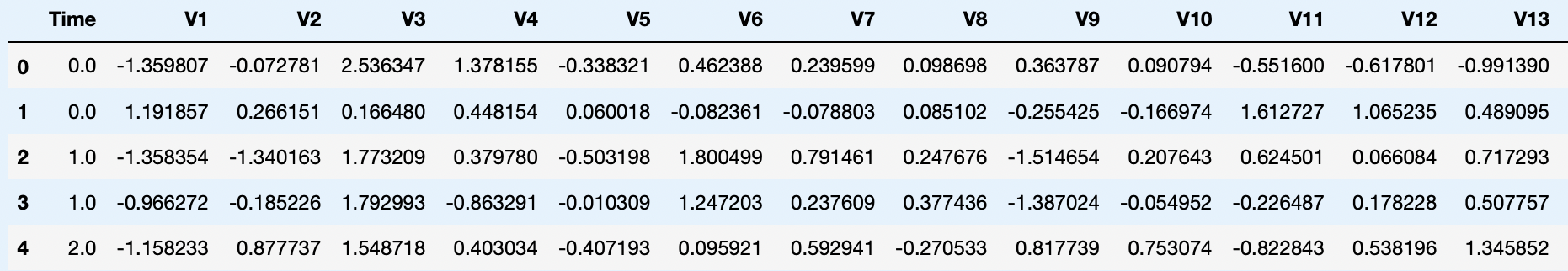
This is the architecture that our team is going to use to solve the problem of the whole project.

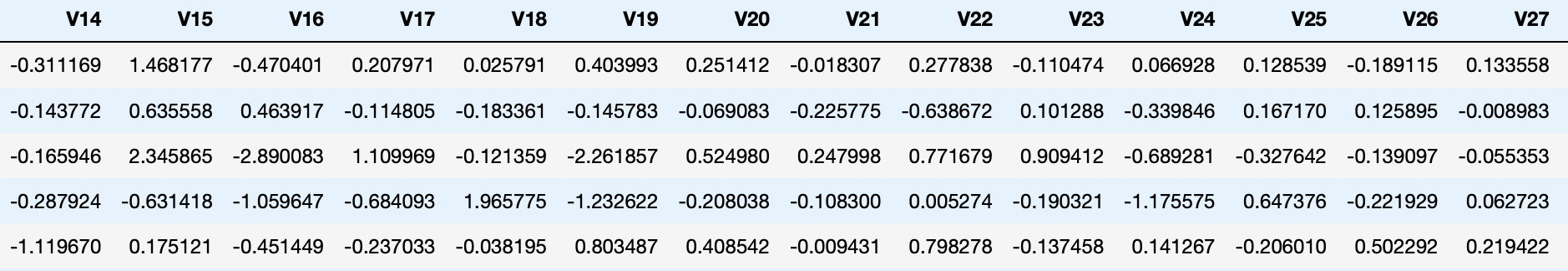
1. **Data source**

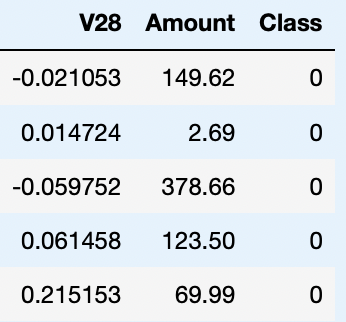
The dataset used in this project consists of credit card transactions over a two-day period. It includes:

* 28 anonymized numerical features obtained through Principal Component Analysis (PCA).
* 'Time' feature representing seconds elapsed between transactions.
* 'Amount' feature indicating the transaction amount.
* 'Class' label, where 1 denotes a fraudulent transaction, and 0 indicates a genuine transaction.

The dataset's imbalance, with fraudulent transactions being a small fraction, poses a unique challenge and necessitates special handling.







1. **Technical tools**

The project employs various tools and technologies, including:

* Data Handling: pandas library for data manipulation and cleaning.
* Model Training and Evaluation: scikit-learn for implementing machine learning models such as Logistic Regression, Decision Trees, and Random Forests.
* Data Visualization: Libraries like Seaborn and matplotlib for visually representing data insights and model performance.
* Imbalanced Data Handling: Techniques like undersampling and oversampling, along with SMOTE, to balance the dataset.

Hardware:

Intel i5( 4 - Cores) Processor

7.5 GB RAM

500 GB SSD

Software:

Technologies to be used: Data Store: HDFS 3.x, Apache Spark 3.x 2.

Programming: Python 3.x 3.

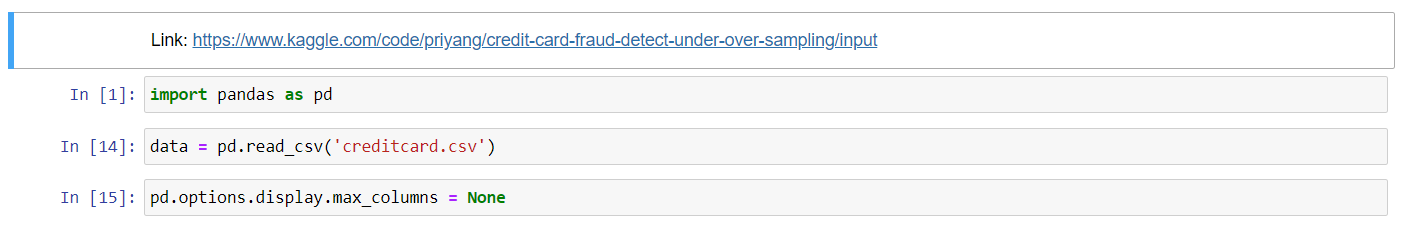
Visualization: Tableau Desktop 2023.2

Chapter I provides a comprehensive introduction to the project, outlining the problem's critical nature, the project's purpose and scope, details of the data source, and the technical methodologies applied. It sets the stage for subsequent chapters, focusing on the architecture, design, data visualization, and final implementation of the credit card fraud detection system.

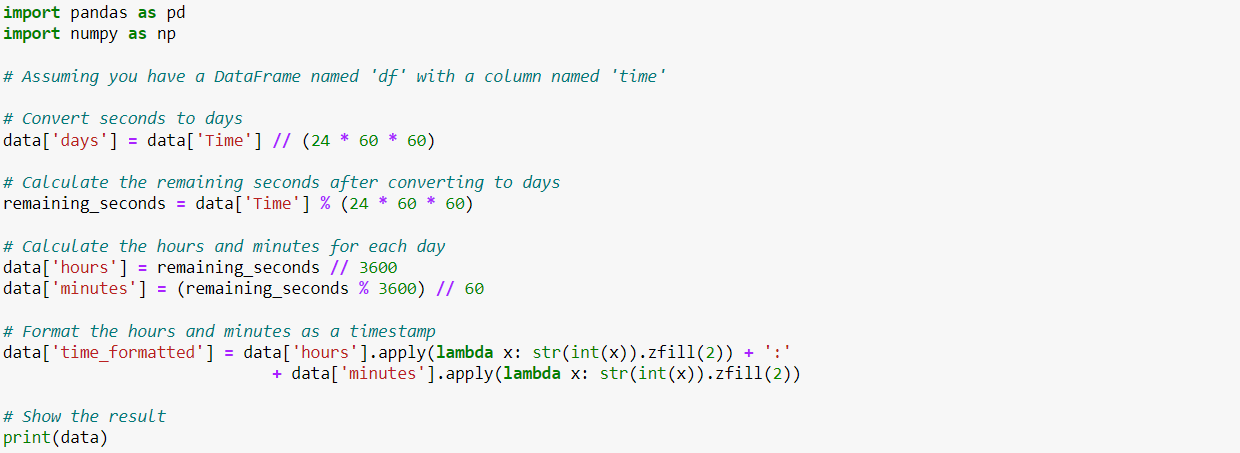
By understanding the context and objectives laid out in this chapter, readers can appreciate the subsequent methodologies and findings, recognizing their relevance and importance in the broader landscape of financial security and fraud detection.

# CHAPTER II: PROJECT DESIGN: FROM DATA EXPLORATION TO MODELING

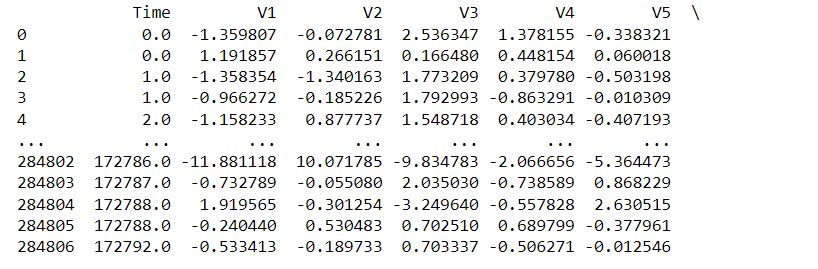
## Exploratory Data Analysis (EDA)



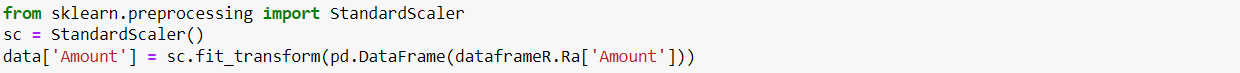
Before processing data, our team made a preparation by formatting the “Time” column as a timestamp.



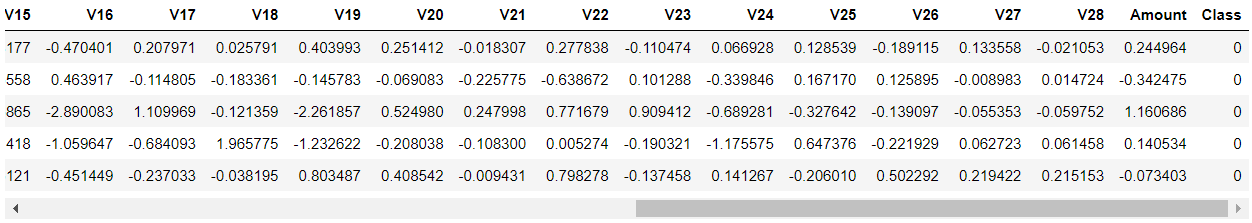
Result:



Next, we are going to transform the “Amount” column by using a scaler method to make values in this column in a range between 0 and 1.



Result:



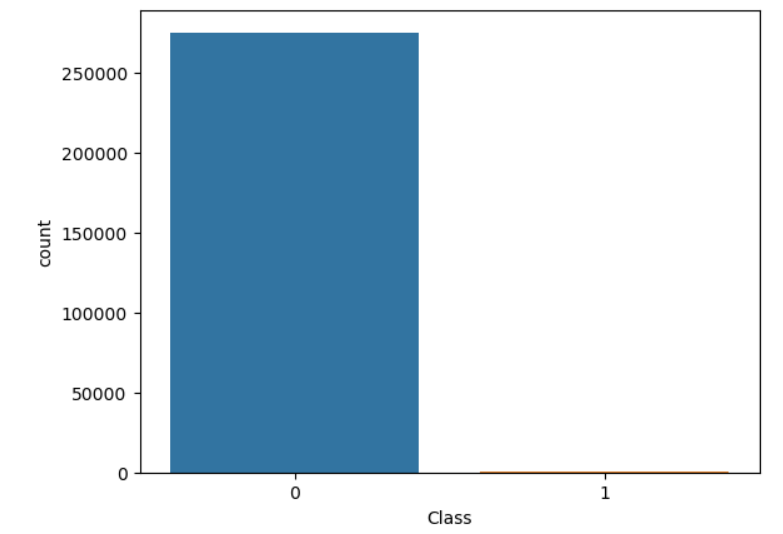
We dropped the “Time” Column



Next, we are going to check the number of fraud transactions in the database by clarifying and counting the values in the “Class” column.

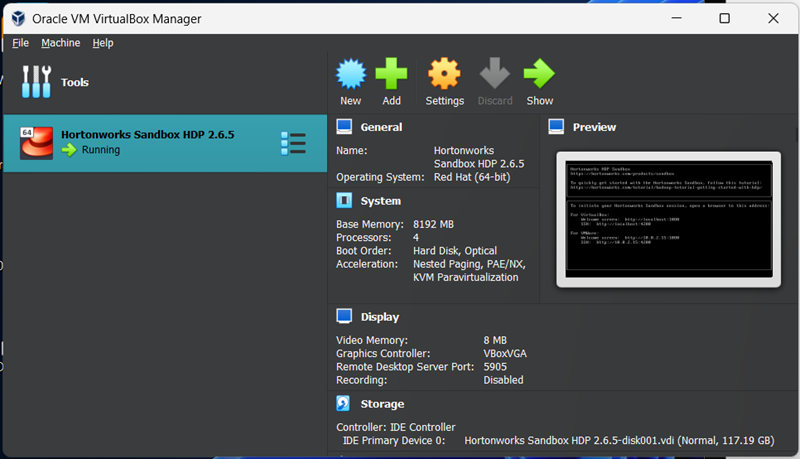


Result:

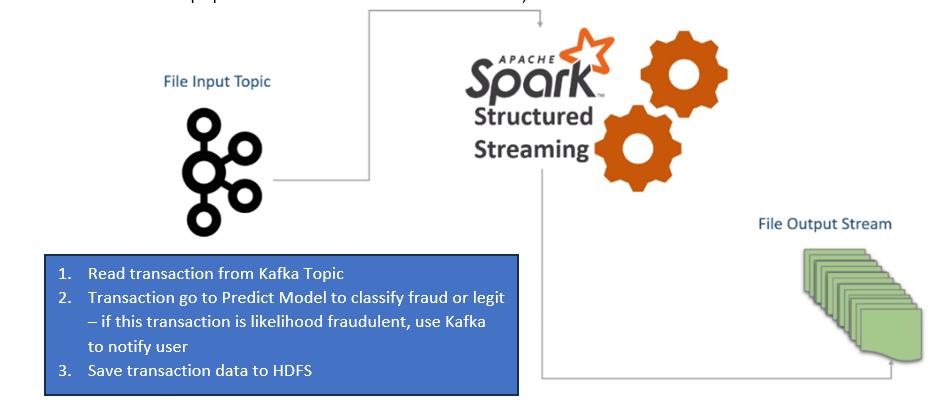


## Hadoop cluster setting

First, we download Hortonworks-sandbox 2.6.5 and virtualbox and install Hortonworks-sandbox 2.6.5 to Virtualbox.



Next, we set up Spark Stream to take raw dataset, and then run Predict Model and save to HDFS file.



Next, our team set up Kafka following the below step:

- Create topic:

Go to folder bin of kafka  
 cd /usr/hdp/current/kafka-broker/bin  
 Now we can create topic

./kafka-topics.sh --create --zookeeper sandbox-hdp.hortonworks.com:2181 --replication-factor 1 --partitions 1 --topic transactions

We run Spark Stream

* Download spark-streaming-kafka-assembly 2.10 jar in link:

<https://search.maven.org/artifact/org.apache.spark/spark-streaming-kafka-assembly_2.10/1.6.0/jar>

* linux command:

wget<https://search.maven.org/remotecontent?filepath=org/apache/spark/spark-streaming-kafka-assembly_2.10/1.6.0/spark-streaming-kafka-assembly_2.10-1.6.0.jar>

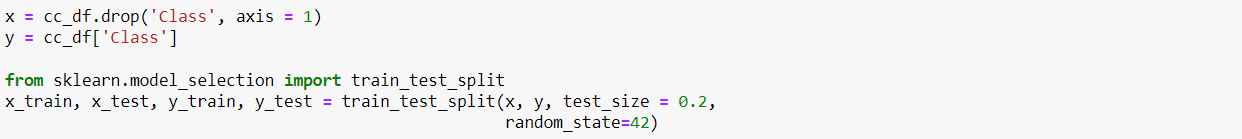
* Download streamingTransactionAnalysis.py in our team github
* Run spark streaming to analysis

spark-submit –jars spark-streaming-kafka-assembly\_2.10-1.5.2.jar /streamingTransactionAnalysis.py

At this point, new transactionChecked.csv is created and ready to visualize.

## Modeling Strategies

We stored the feature matrix in X and response (Target) in vector y. Then, we split the dataset into the training and testing set.



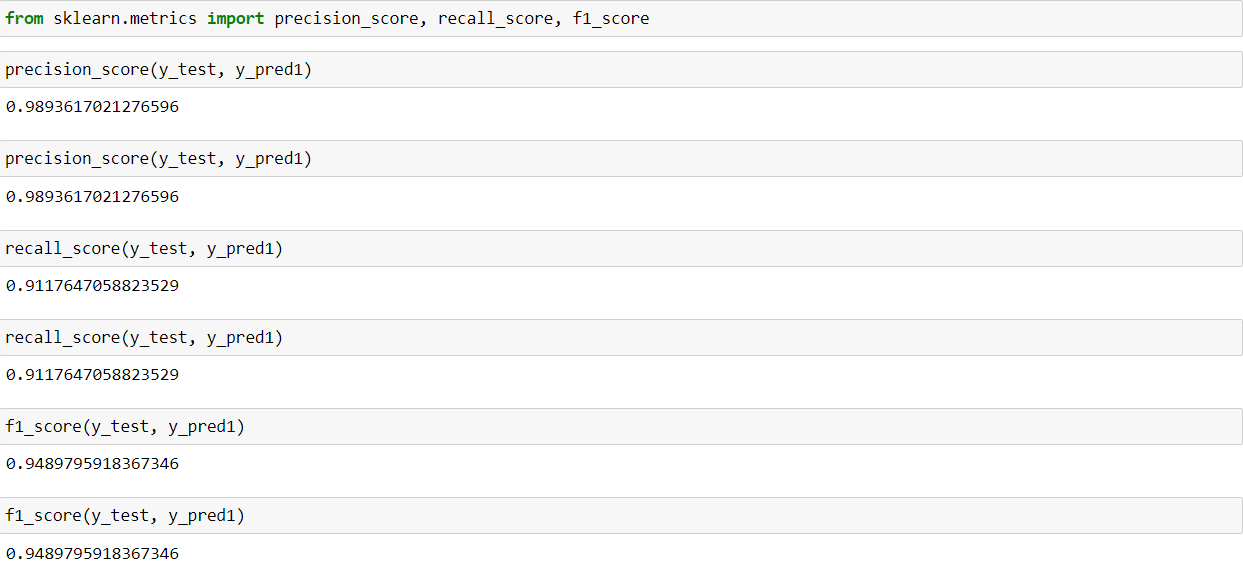
Next, we are going to use undersampling and oversampling to interpret the logistic regression, decision tree classifier, and Random Forest Classifier.

* Undersampling

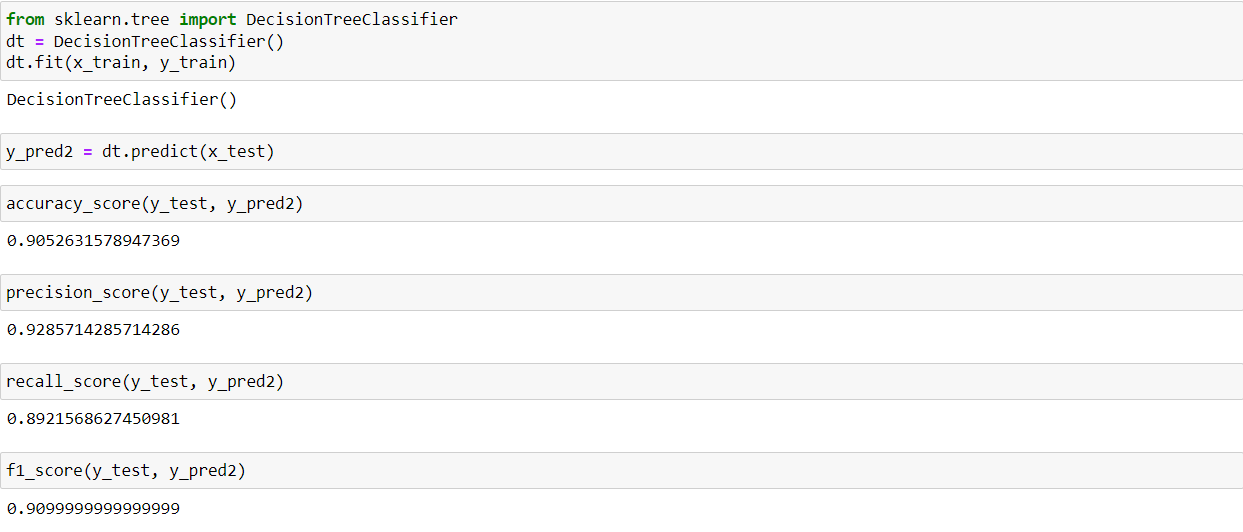


1. Logistic Regression

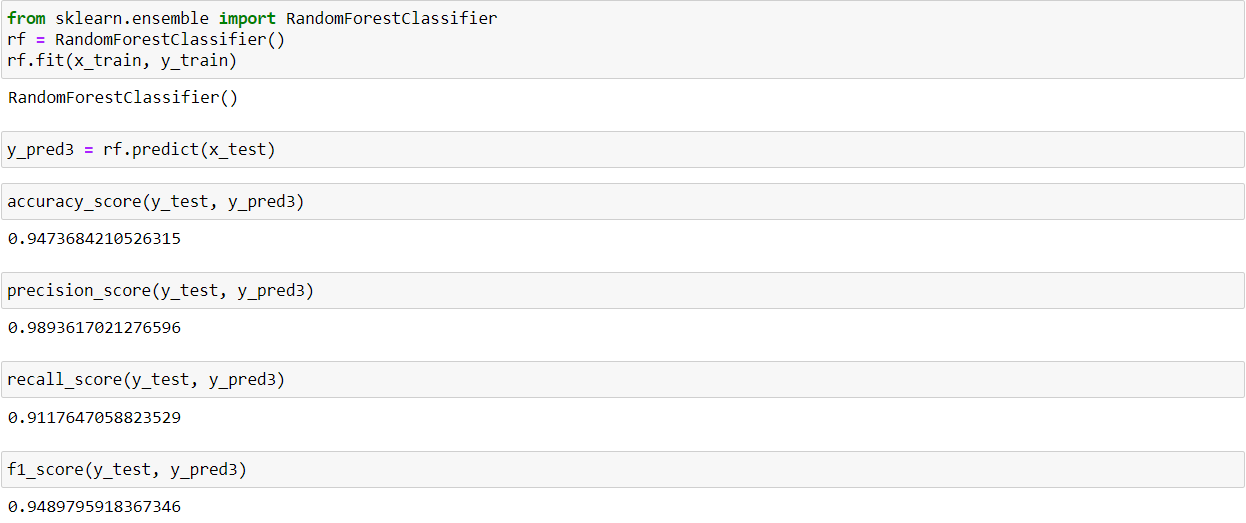




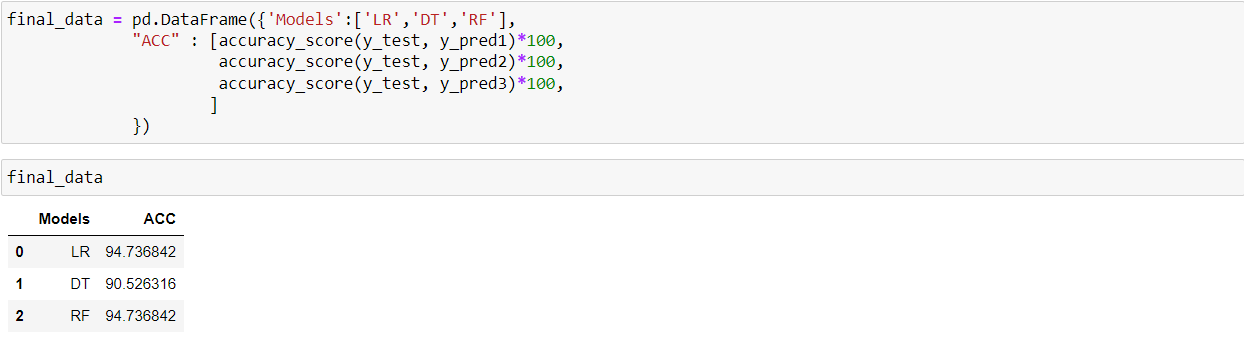
1. Decision Tree Classifier



1. Random Forest Classifier

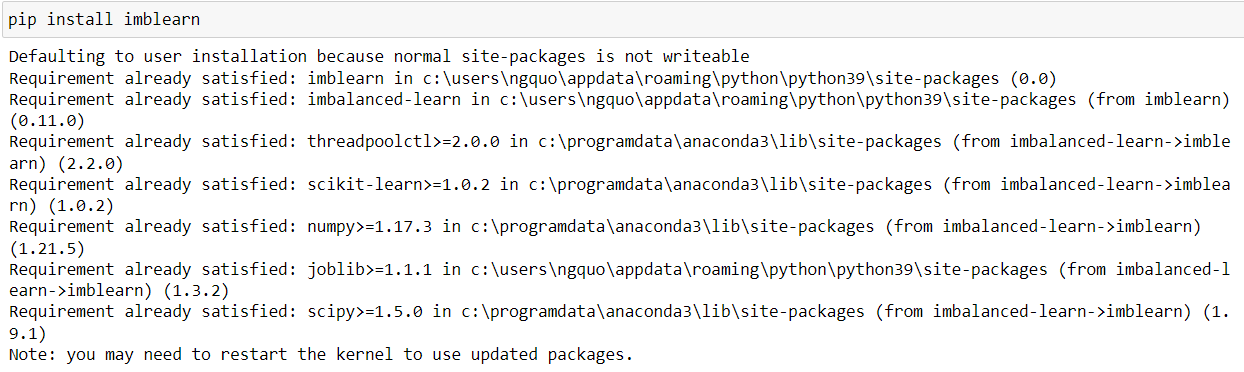


After that, we make a table summarizing the results obtained in previous steps



* Oversampling







1. Logistic Regression

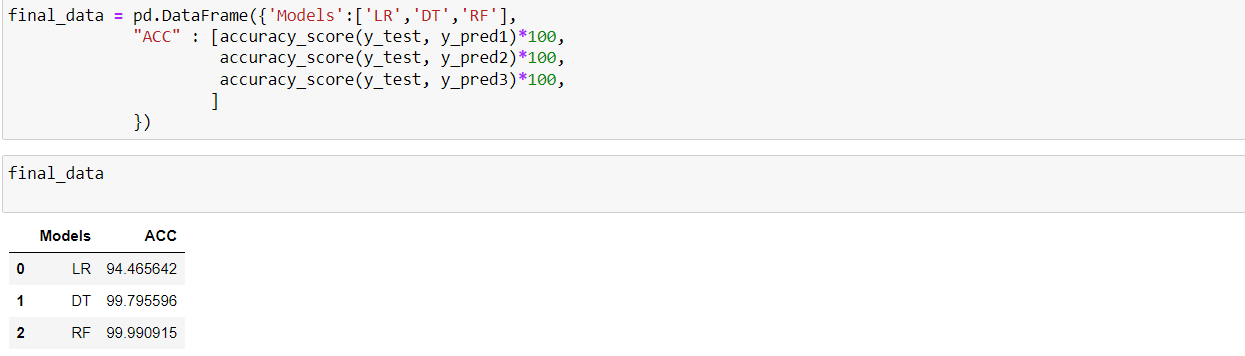


1. Decision Tree Classifier



1. Random Forest Classifier





* Application for testing model

Code:

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score

import streamlit as st

# load data

data = pd.read\_csv('creditcard.csv')

# separate legitimate and fraudulent transactions

legit = data[data.Class == 0]

fraud = data[data.Class == 1]

# undersample legitimate transactions to balance the classes

legit\_sample = legit.sample(n=len(fraud), random\_state=2)

data = pd.concat([legit\_sample, fraud], axis=0)

# split data into training and testing sets

X = data.drop(columns="Class", axis=1)

y = data["Class"]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, stratify=y, random\_state=2)

# train logistic regression model

model = LogisticRegression()

model.fit(X\_train, y\_train)

# evaluate model performance

train\_acc = accuracy\_score(model.predict(X\_train), y\_train)

test\_acc = accuracy\_score(model.predict(X\_test), y\_test)

# create Streamlit app

st.title("Credit Card Fraud Detection Model")

st.write("Enter the following features to check if the transaction is legitimate or fraudulent:")

# create input fields for user to enter feature values

input\_df = st.text\_input('Input All features')

input\_df\_lst = input\_df.split(',')

# create a button to submit input and get prediction

submit = st.button("Submit")

if submit:

# get input feature values

features = np.array(input\_df\_lst, dtype=np.float64)

# make prediction

prediction = model.predict(features.reshape(1,-1))

# display result

if prediction[0] == 0:

st.write("Legitimate transaction")

else:

st.write("Fraudulent transaction")

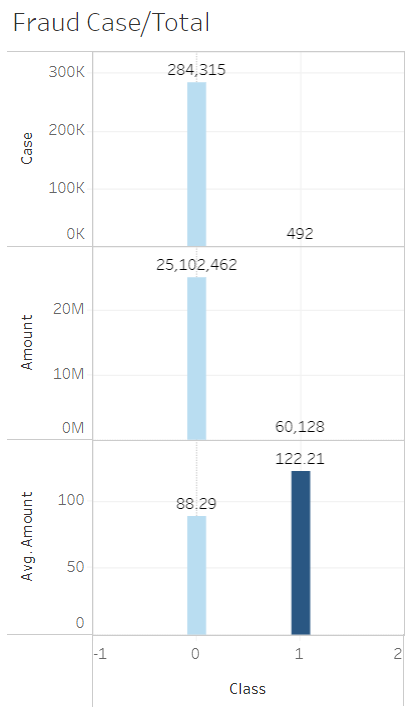
# 

# CHAPTER III: DATA VISUALIZATION

In this chapter, we are going to conduct data visualization from the given available data set “credit card fraud” by using Tableau. This process means extracting the insights which my team used to draw out the further conclusion of our project.

Later, our team also guideline of using tableau to visualize the data of this project.

## Overview of transactions

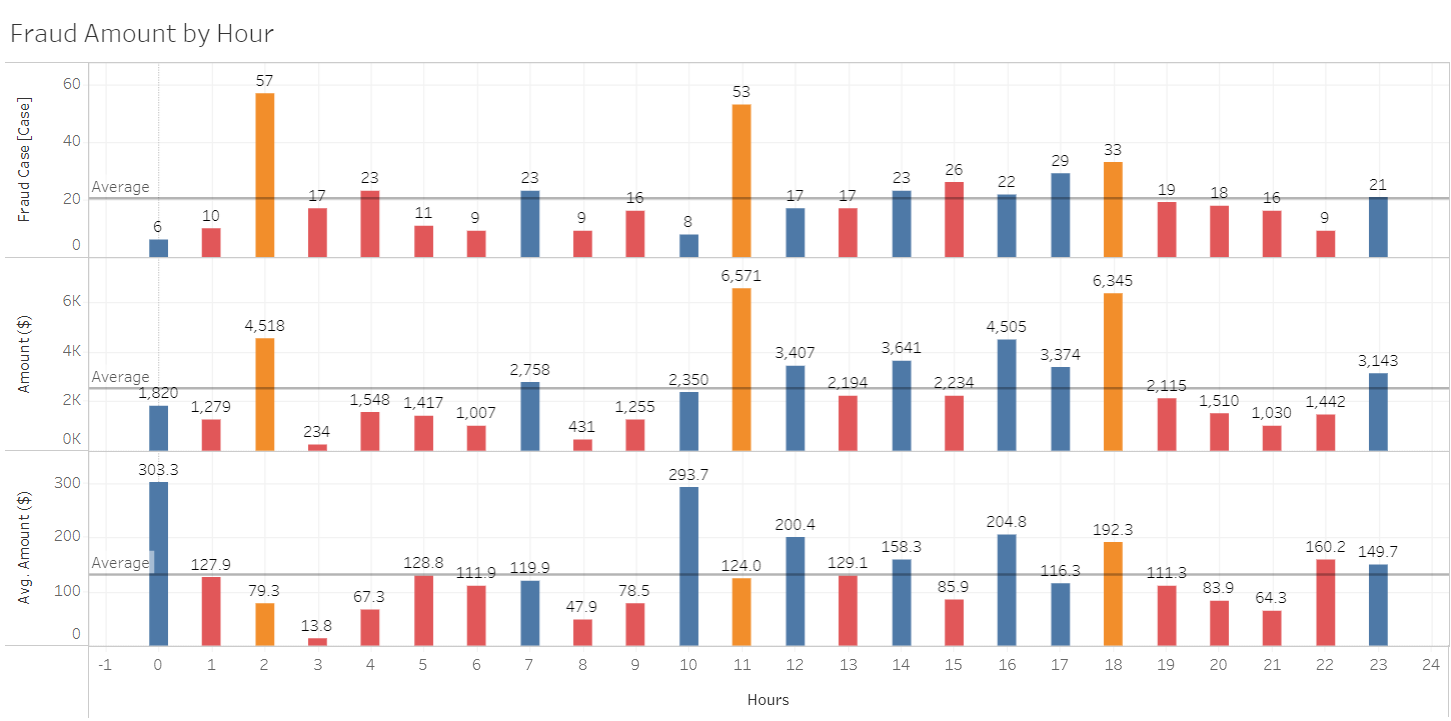


*The comparison between legit and fraud transactions*

In terms of comparison between the legit and fraud transactions, our team focuses on three aspects from the available dataset, including the number of transactions, the amount of money paid, and the average amount within one transaction. In the above figure, the class has been clarified into two groups: “0” representing legit transactions, and “1” representing the fraud ones.

For this dataset, the number of transactions that were legit was dominantly higher than that of fraud ones, with 284,315 times and 492 times respectively. Those figures also reflect the huge gap between the amount of money between valid transactions ($25,102,462) and invalid transactions ($60,128).

Paradoxically, however, the average amount of money traded each time of frauds was moderately higher than that of legit transactions with $122,21 and $88.29 respectively.



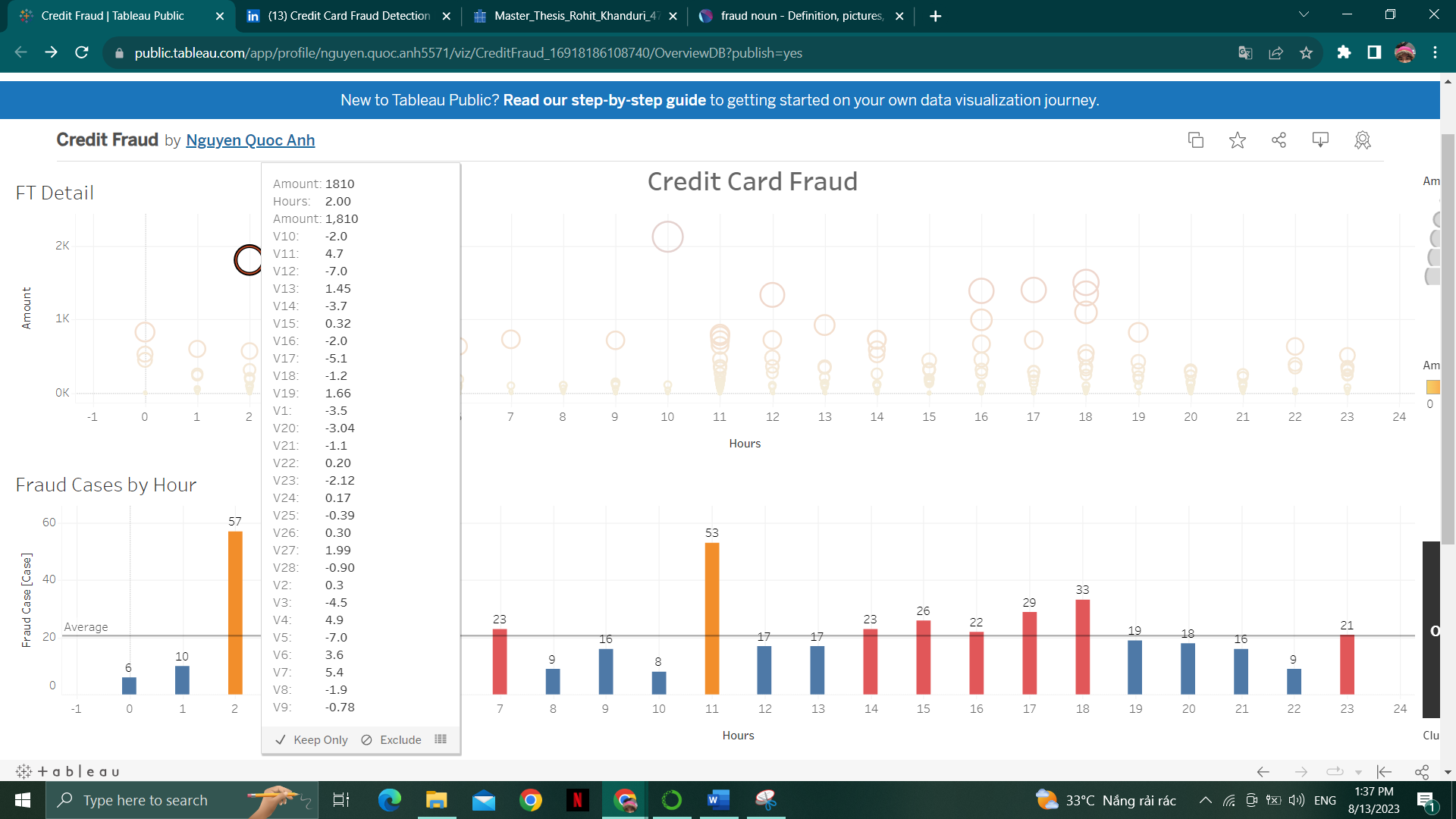
*The occurrence of fraud transactions by hours*

According to given charts, it generally shows that the number of fraud transactions occurred was just around the average of 20 transactions. 2a.m. and 11a.m. were the hours witnessing the highest level of occurrence of fraud transactions, with around 55 transactions. Those fraud transactions occurred regardless of hours within a day.

To all fraud transactions, 2a.m, 11a.m, and 18p.m were the time reporting the highest amount of money traded illegally, with $4,518, $6,571, and $6,345 respectively. Following that was the group of transactions occurring only from 12p.m and 17p.m, reported to be just above the average of $2,505. Whilst the remaining hours reported fraud amounts of money which is in a range of $1000 and $1500.

Moreover, 12a.m and 10a.m had the highest average amount of money within a fraud transaction with $303,3 and $293,7, whereas the remaining average amounts were around the average of $131,4 for a fraud transaction.

## Details of fraud transactions

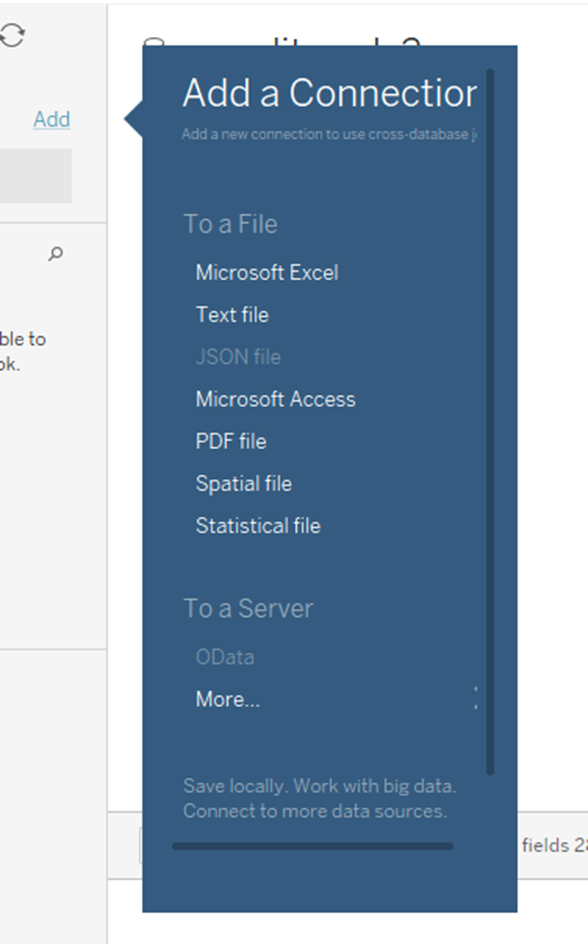


*Fraud transactions with PCA transformation*

This dashboard aims to give a different view of data of fraud transactions and in the meantime, provide the information related to PCA transformation.

## Tableau guideline

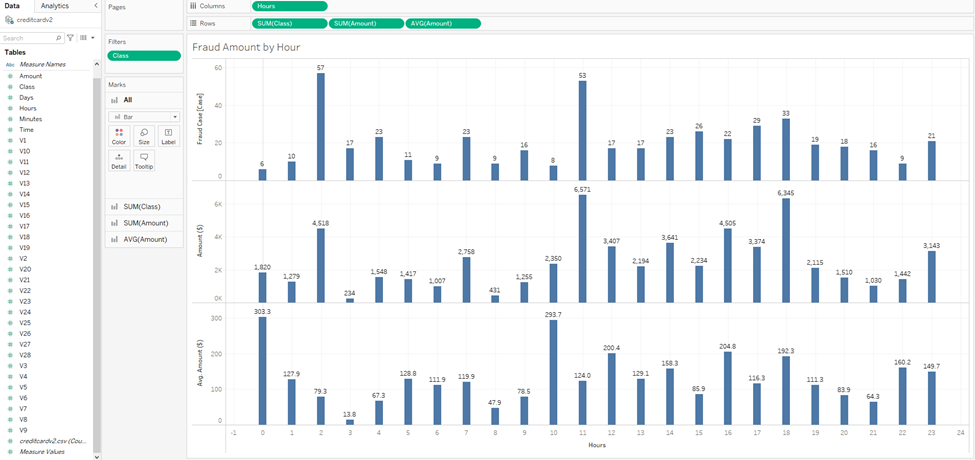
First, we import the csv file of the database to Tableau. To do so, we click “add” and then choose “Text file” to select “creaditcardv2.csv”. This “creaditcardv2.csv” has been adjusted from the original database properly to draw out the desirable results.



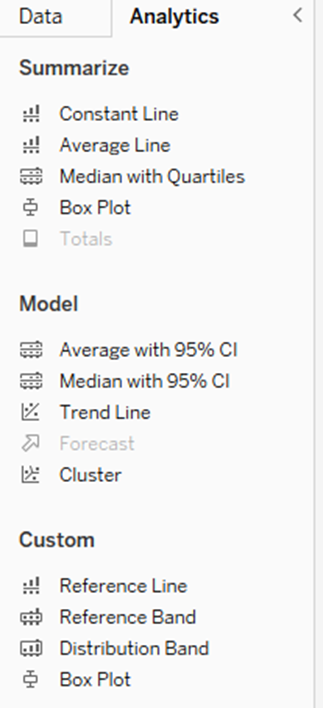
Next, we are going to create a fraud case quantity chart. We drag “Class” and “Amount” into Rows, and then choose suitable Measure: CNT(Class), SUM(Amount), and AVG(Amount). Later, we drag Class into Columns which has been shown in the below figure.

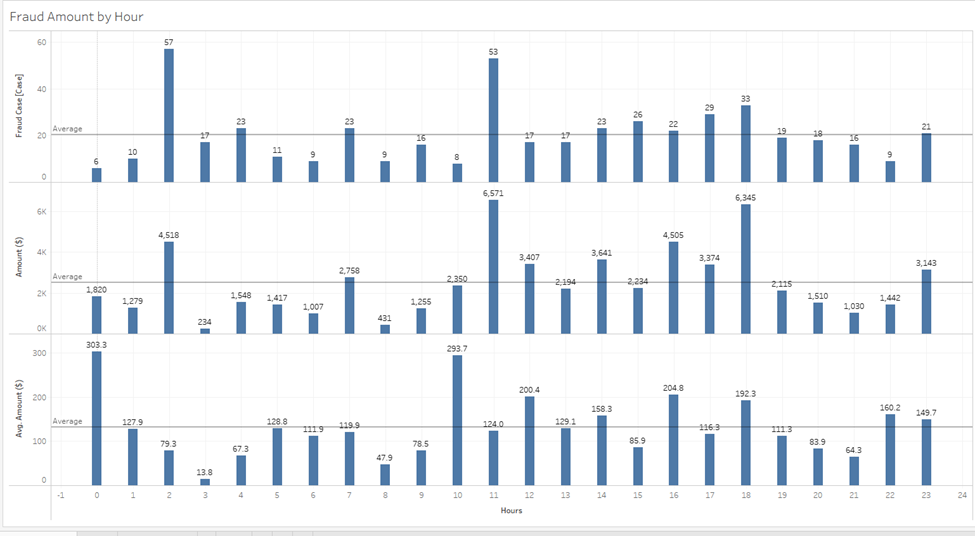


To create a fraud amount by hour chart, our team dragged “Class” and “Amount” into Rows then chose suitable Measure: SUM(Class), SUM(Amount), and AVG(Amount), and then, dragged “Hours” into Columns.

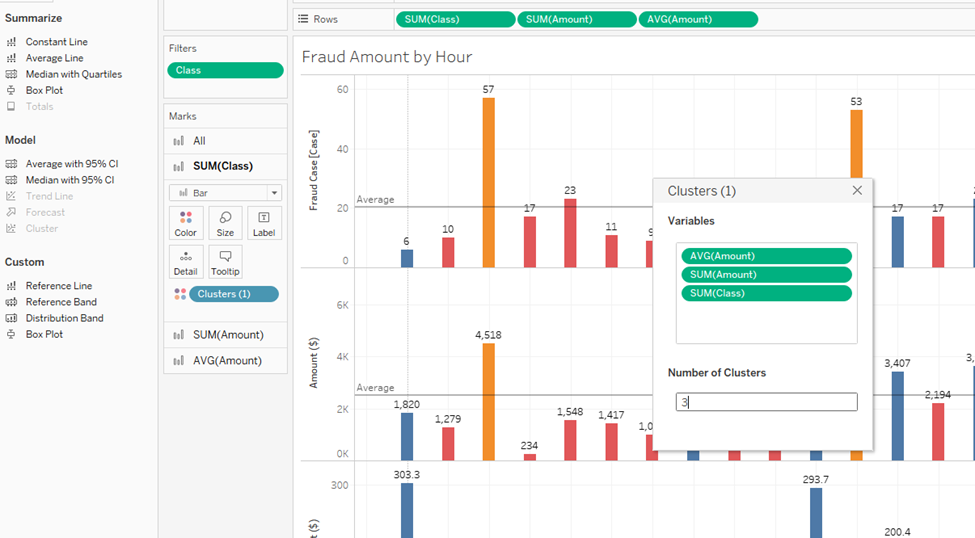


In the next step, we chose the Analytics tab, and then dragged Average Line into the chart and chose “Table”.

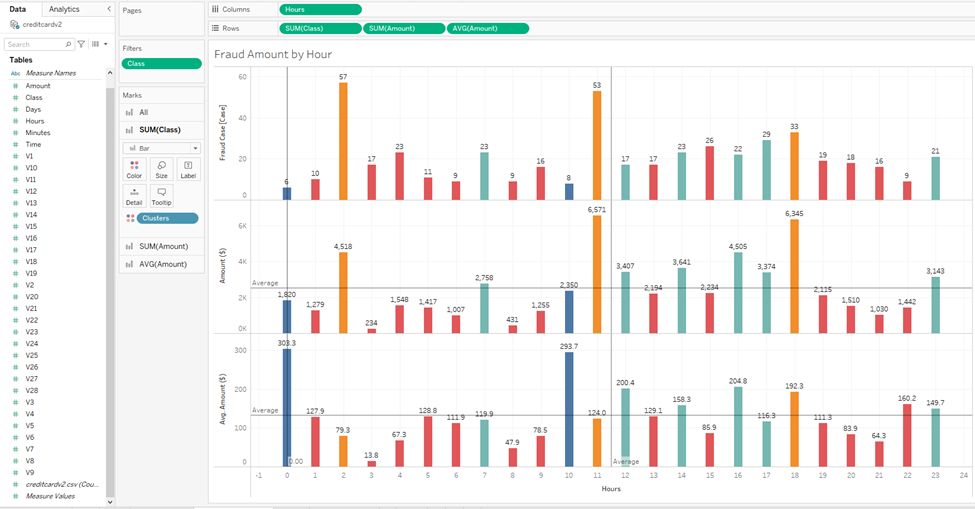




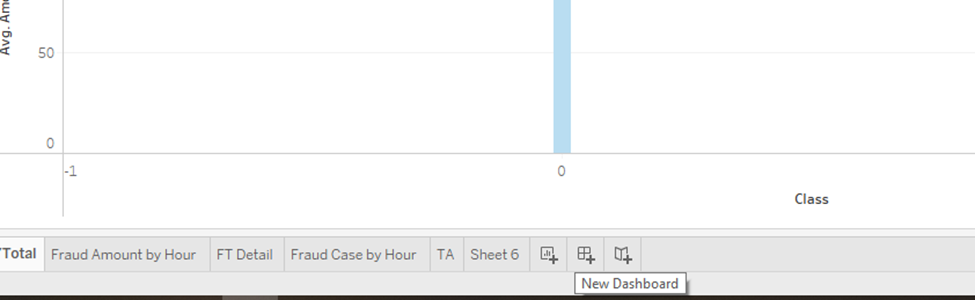
After that, we dragged Cluster into chart, typed “3” into Number of Clusters as the following figure.

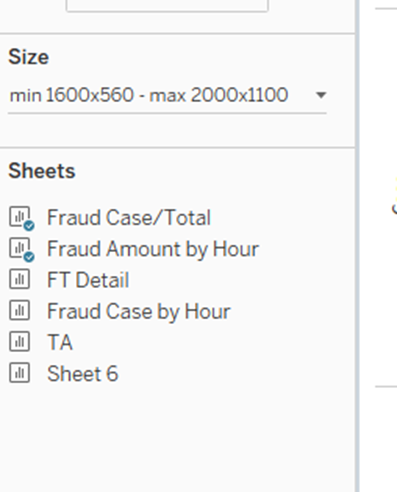


After those steps conducted, we obtain a desirable results shown in the following figure.

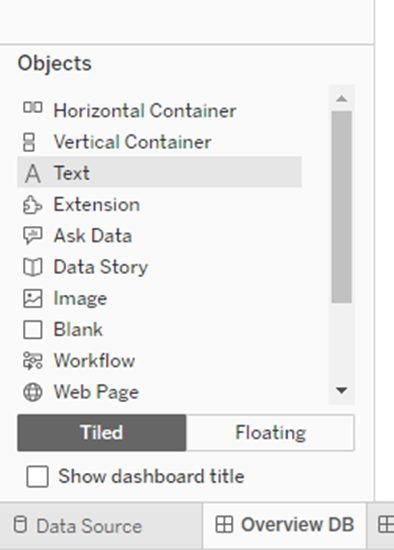


In the Overview page of the dashboard, we click to create a new page for the dashboard. On the left side, we set 2000 x 1100 in the Size menu, and then, dragged two above charts, Fraud Case quantity chart and Total fraud amount by hour into the dashboard.

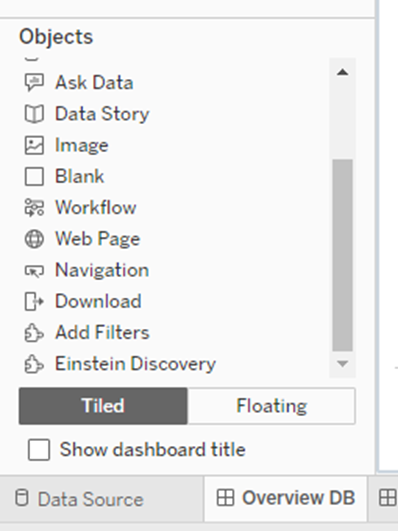




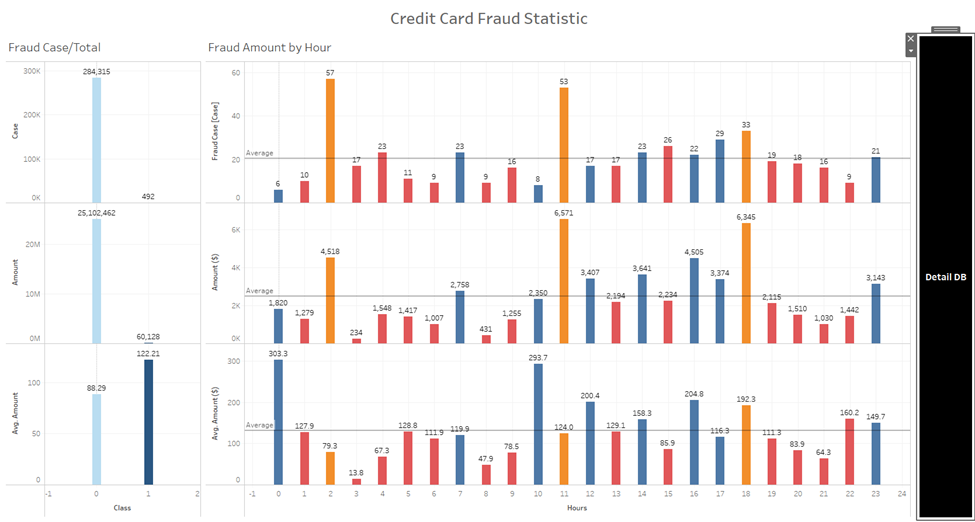
In the Object menu, we dragged “Text” to the top of the dashboard to create Title (Credit Card Fraud Statistics)



Next, we dragged “Navigation” to the right side of Dashboard and link it to Detail Dashboard which we would create later.

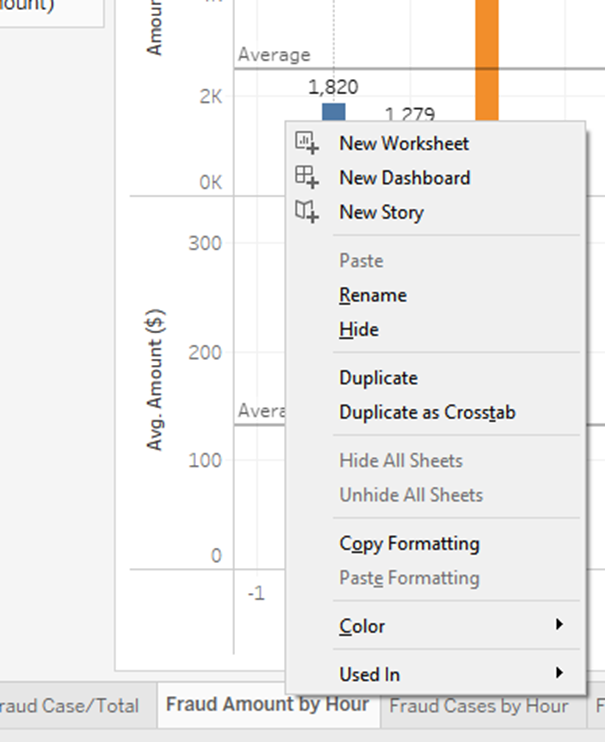


After conducting those steps, we obtain an Overview dashboard.

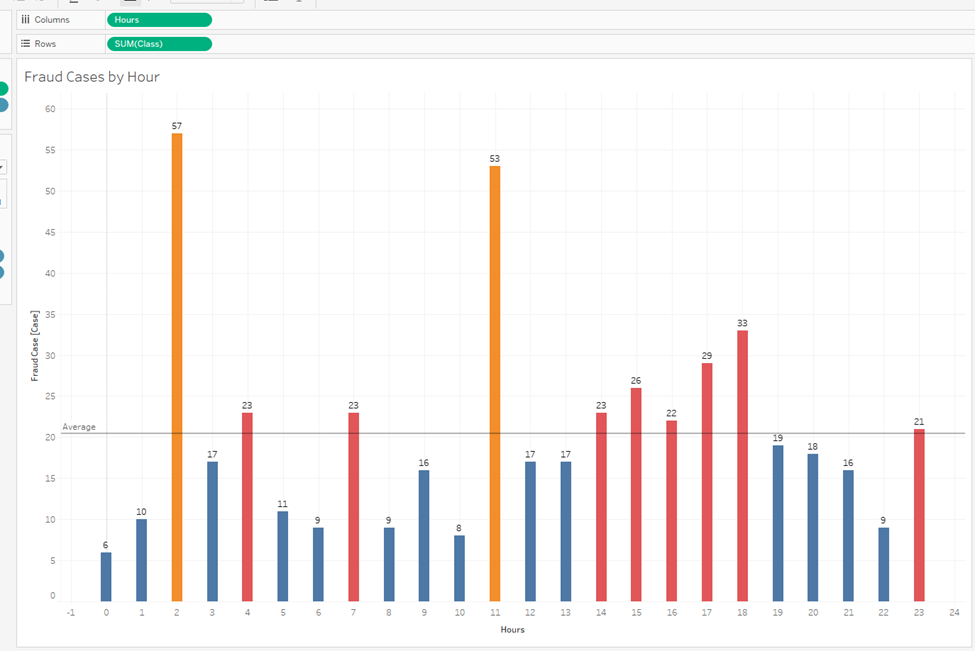


Further, we create a Detail dashboard in the second page. IN this dashboard, we can see details of each transaction by using Tooltips.

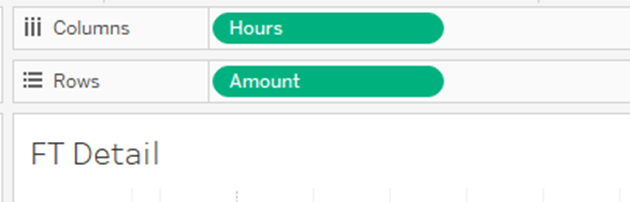
We are going to duplicate the Fraud Amount by Hour which had been created in the previous step.

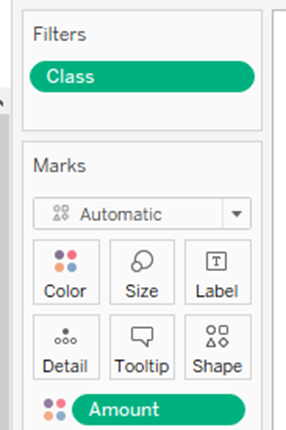


In terms of Columns, we removed others, but keeping SUM(Class).

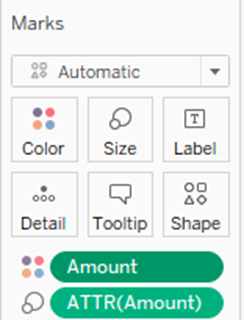


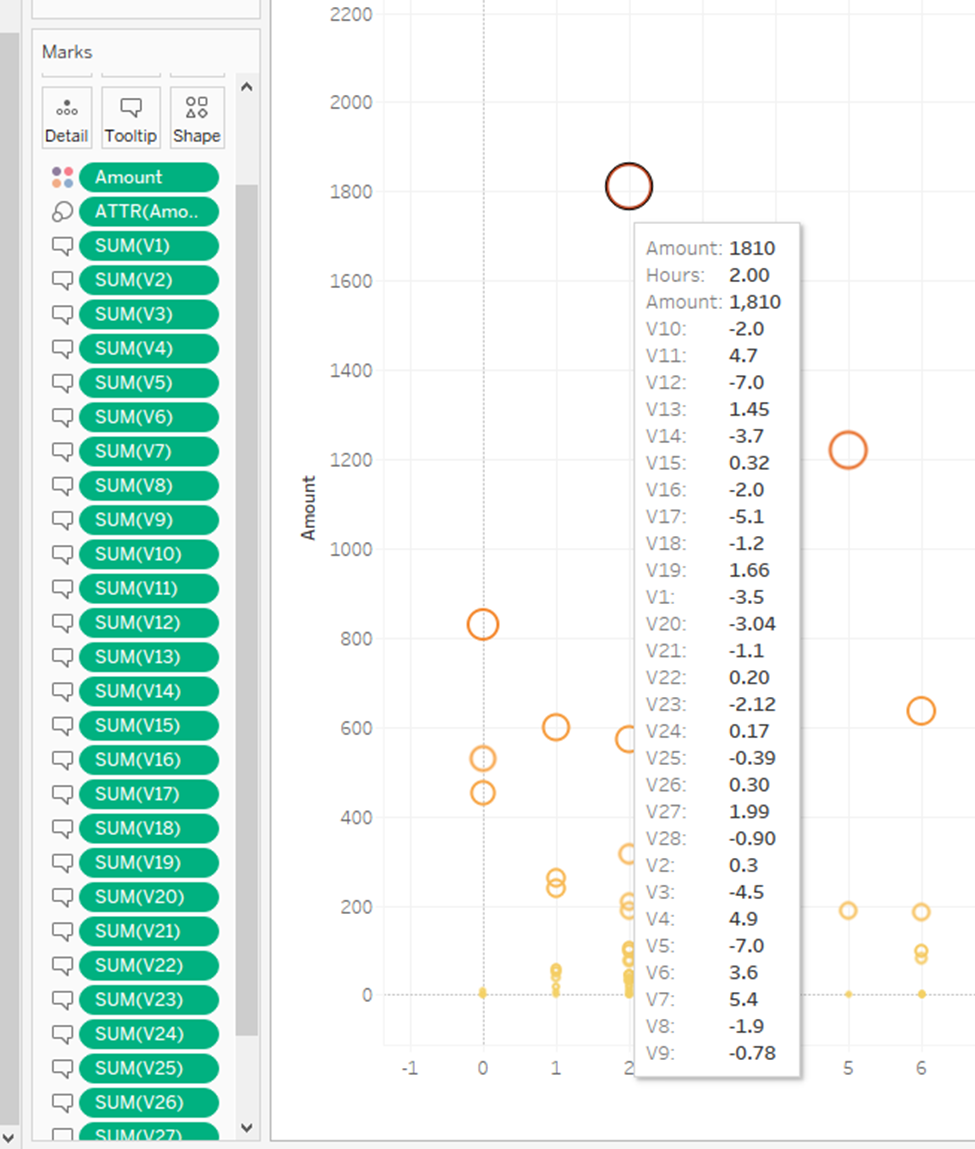
To create a fraud transaction detail Chart, we, first, dragged “Hours” into “Columns”, “Amount” into Hours”. Then, we dragged “Class” into “Filter” then chose “1” and “Amount” into “Color”.



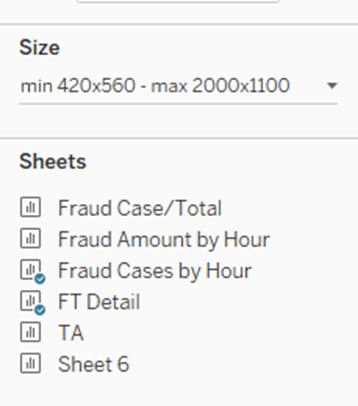


Next, we continued to drag “Amount” into “Size” and V1 to V28 into Tooltips to shw detail about Fraud transactions.

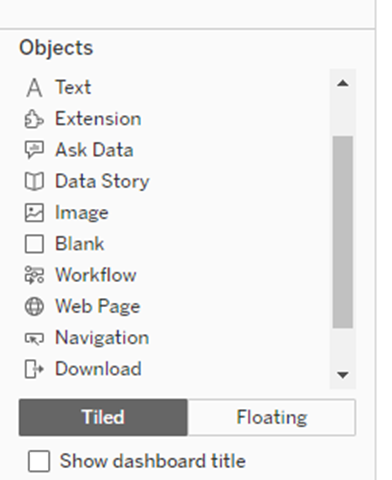




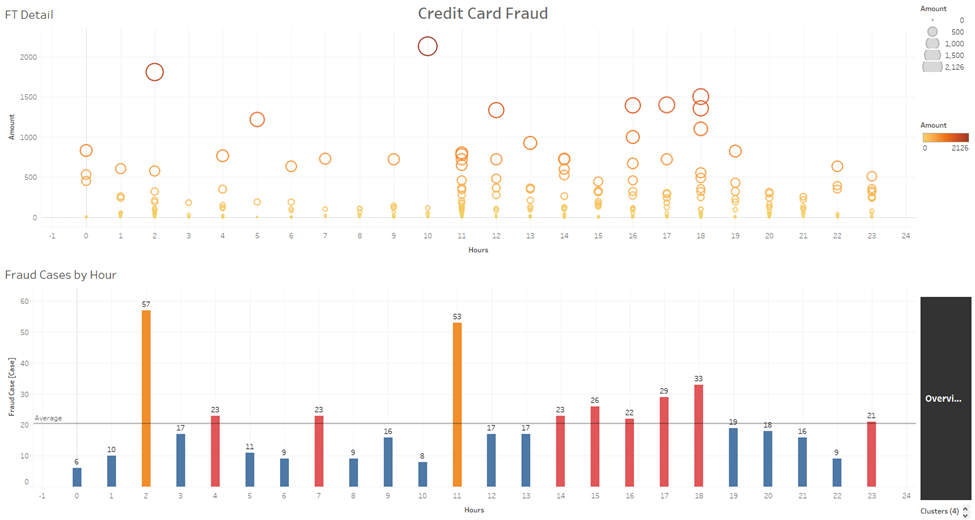
In the next step, we create a Detail dashboard and set Size as 2000 x 1100 pixels. Then, we drag Fraud Case by Hour and FT Detail into the dashboard page.



Next, we dragged Text into the dashboard page to create title and dragged “Navigation” to the right side of the dashboard and navigate to the Overview dashboard.



Finally, we obtain a desirable dashboard.



After creating the dashboard, we publish it to Tableau Public. This is a link to our dashboard: <https://public.tableau.com/app/profile/nguyen.quoc.anh5571/viz/CreditFraud_16918186108740/OverviewDB?publish=yes>

# CHAPTER IV: LIMITATION

**1. Data Imbalance Handling:**

The codes demonstrate both undersampling and oversampling techniques to handle the significant imbalance in the data. While these methods are standard practices, they have inherent limitations:

* Undersampling: Reducing the number of majority class samples might lead to loss of information and may cause the model to underperform.
* Oversampling: Creating synthetic samples can sometimes lead to overfitting, where the model performs well on the training data but poorly on unseen data.
* SMOTE: While SMOTE is used for oversampling, it can introduce noise by creating synthetic samples that are not representative of true fraudulent behavior.

**2. Feature Engineering Limitations:**

The codes include preprocessing like scaling the "Amount" column and transforming the "Time" column. However, given the anonymized nature of most features, there might be missed opportunities for more targeted feature engineering based on domain knowledge. An understanding of what each feature represents could lead to more insightful transformations and interactions.

**3. Modeling Constraints:**

* Algorithm Selection: The codes use Logistic Regression, Decision Tree, and Random Forest. While these are robust algorithms, the selection might not be optimal for this specific problem. More sophisticated algorithms or ensemble methods might yield better results.
* Hyperparameter Tuning: There doesn't appear to be hyperparameter tuning, which could further optimize model performance. Grid search or random search could be employed to find the best parameters.

**4. Evaluation Metrics Consideration:**

The codes employ accuracy, precision, recall, and F1 score for evaluation. While these are valuable, in a highly imbalanced scenario, additional metrics like the Area Under the Precision-Recall Curve (AUC-PR) might provide a more nuanced understanding of performance.

**5. Scalability and Real-Time Deployment:**

While PySpark is mentioned, the provided codes are mostly using standard Python libraries, which might face scalability issues in a real-time, large-scale deployment. Consideration for distributed computing and optimized data pipelines would be necessary for production-ready applications.

**6. Data Quality and Completeness:**

Though the codes include checks for null values and duplicates, real-world data often presents challenges in quality, consistency, and completeness. Robust preprocessing and data cleaning strategies might be needed to handle such issues.

**7. Dependency on External Libraries:**

The codes rely on several external libraries, including imblearn for handling imbalance. Compatibility, updates, and maintenance of these libraries might pose challenges in a production environment.

**REFERENCES**

*For any inquiries or questions related to the project, please feel free to reach out to us via the following email address.*

*trantruongminhthang@gmail.com*

*We look forward to receiving your correspondence and will be more than happy to address your concerns and provide the best assistance possible.*