1. **PROBLEM STATEMENT**

To detect fraudulent credit cards using Isolation Forest algorithm and Local Outlier Factor.

1. **INTRODUCTION**

Credit-card-based purchases can be categorized into two types: 1) physical card and 2) virtual card. In a physical-card based purchase, the cardholder presents his card physically to a merchant for making a payment. To carry out fraudulent transactions in this kind of purchase, an attacker has to steal the credit card. If the cardholder does not realize the loss of card, it can lead to a substantial financial loss to the credit card company. In the second kind of purchase, only some important information about a card (card number, expiration date, secure code) is required to make the payment. Such purchases are normally done on the Internet or over the telephone. To commit fraud in these types of purchases, a fraudster simply needs to know the card details. Most of the time, the genuine cardholder is not aware that someone else has seen or stolen his card information. The only way to detect this kind of fraud is to analyze the spending patterns on every card and to figure out any inconsistency with respect to the “usual” spending patterns. Fraud detection based on the analysis of existing purchase data of cardholder is a promising way to reduce the rate of successful credit card frauds. Since humans tend to exhibit specific behaviorist profiles, every cardholder can be represented by a set of patterns containing information about the typical purchase category, the time since the last purchase, the amount of money spent, etc. Deviation from such patterns is a potential threat to the system.

1. **OBJECTIVES**
2. To select an appropriate dataset of credit card holders on which algorithm can be performed.
3. To apply local outlier factor and Isolation Forest Algorithm.
4. **THEORY**

**Python**

Python is an [interpreted](https://en.wikipedia.org/wiki/Interpreted_language), [high-level](https://en.wikipedia.org/wiki/High-level_programming_language), [general-purpose programming language](https://en.wikipedia.org/wiki/General-purpose_programming_language). Created by [Guido van Rossum](https://en.wikipedia.org/wiki/Guido_van_Rossum) and first released in 1991, Python has a design philosophy that emphasizes [code readability](https://en.wikipedia.org/wiki/Code_readability), notably using [significant whitespace](https://en.wikipedia.org/wiki/Significant_whitespace). It provides constructs that enable clear programming on both small and large scales. Van Rossum led the language community until stepping down as leader in July 2018.

Python features a [dynamic type](https://en.wikipedia.org/wiki/Dynamic_type) system and automatic [memory management](https://en.wikipedia.org/wiki/Memory_management). It supports multiple [programming paradigms](https://en.wikipedia.org/wiki/Programming_paradigm), including [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming), [imperative](https://en.wikipedia.org/wiki/Imperative_programming), [functional](https://en.wikipedia.org/wiki/Functional_programming) and [procedural](https://en.wikipedia.org/wiki/Procedural_programming). It also has a comprehensive [standard library](https://en.wikipedia.org/wiki/Standard_library).

Python interpreters are available for many [operating systems](https://en.wikipedia.org/wiki/Operating_system). [CPython](https://en.wikipedia.org/wiki/CPython), the [reference implementation](https://en.wikipedia.org/wiki/Reference_implementation) of Python, is [open source](https://en.wikipedia.org/wiki/Open-source_software) software and has a community-based development model, as do nearly all of Python's other implementations. Python and CPython are managed by the non-profit [Python Software Foundation](https://en.wikipedia.org/wiki/Python_Software_Foundation).

**Jupyter notebook**

Jupyter is a [nonprofit organization](https://en.wikipedia.org/wiki/Nonprofit_organization) created to "develop [open-source software](https://en.wikipedia.org/wiki/Open-source_software), open-standards, and services for [interactive computing](https://en.wikipedia.org/wiki/Interactive_computing) across dozens of programming languages". Spun-off from [IPython](https://en.wikipedia.org/wiki/IPython)in 2014 by [Fernando Pérez](https://en.wikipedia.org/wiki/Fernando_P%C3%A9rez_(software_developer)), Project Jupyter supports execution environments in several dozen languages. Project Jupyter's name is a reference to the three core programming languages supported by Jupyter, which are [Julia](https://en.wikipedia.org/wiki/Julia_(programming_language)), [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) and [R](https://en.wikipedia.org/wiki/R_(programming_language)), and also an homage to [Galileo's notebooks](https://en.wikipedia.org/wiki/Galileo_Galilei) recording the discovery of the [moons of Jupiter](https://en.wikipedia.org/wiki/Moons_of_Jupiter). Project Jupyter has developed and supported the interactive computing products Jupyter Notebook, Jupyter Hub, and Jupyter Lab, the next-generation version of Jupyter Notebook.

Jupyter [Notebook](https://en.wikipedia.org/wiki/Notebook_interface) (formerly IPython Notebooks) is a [web-based interactive](https://en.wikipedia.org/wiki/Rich_Internet_application) computational environment for creating Jupyter notebook documents. The "notebook" term can colloquially make reference to many different entities, mainly the Jupyter web application, Jupyter Python web server, or Jupyter document format depending on context. A Jupyter Notebook document is a [JSON](https://en.wikipedia.org/wiki/JSON) document, following a versioned schema, and containing an ordered list of input/output cells which can contain code, text (using [Markdown](https://en.wikipedia.org/wiki/Markdown)), mathematics, plots and rich media, usually ending with the ".ipynb" extension.

Throughout the financial sector, machine learning algorithms are being developed to detect fraudulent transactions. In this project, that is exactly what we are going to be doing as well. Using a dataset of of nearly 28,500 credit card transactions and multiple unsupervised anomaly detection algorithms, we are going to identify transactions with a high probability of being credit card fraud. In this project, we will build and deploy the following two machine learning algorithms:

* Local Outlier Factor (LOF)
* Isolation Forest Algorithm

Furthermore, using metrics such as precision, recall, and F1-scores, we will investigate why the classification accuracy for these algorithms can be misleading.

In addition, we will explore the use of data visualization techniques common in data science, such as parameter histograms and correlation matrices, to gain a better understanding of the underlying distribution of data in our data set.

**Local Outlier Factor (LOF)**

The anomaly score of each sample is called Local Outlier Factor. It measures the local deviation of density of a given sample with respect to its neighbors. It is local in that the anomaly score depends on how isolated the object is with respect to the surrounding neighborhood.

**Isolation Forest Algorithm**

The IsolationForest ‘isolates’ observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature.

Since recursive partitioning can be represented by a tree structure, the number of splittings required to isolate a sample is equivalent to the path length from the root node to the terminating node.

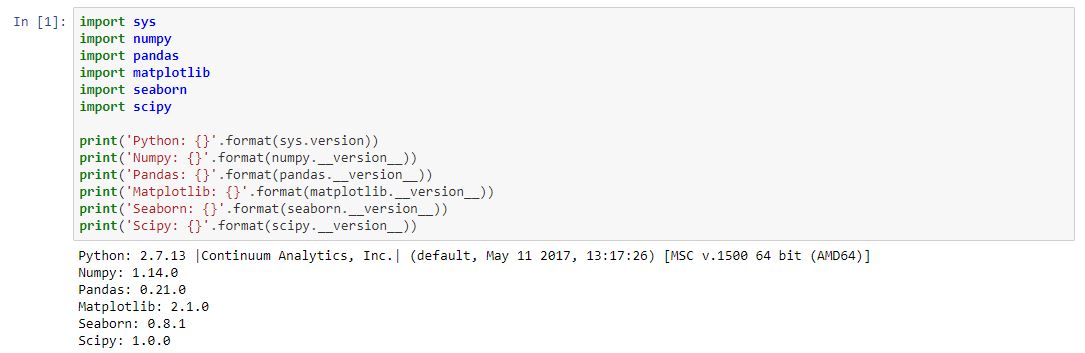
This path length, averaged over a forest of such random trees, is a measure of normality and our decision function

Random partitioning produces noticeably shorter paths for anomalies. Hence, when a forest of random trees collectively produce shorter path lengths for particular samples, they are highly likely to be anomalies.

1. **RESULT**

**1. Importing Necessary Libraries**

To start, let's print out the version numbers of all the libraries we will be using in this project. This serves two purposes - it ensures we have installed the libraries correctly and ensures that this tutorial will be reproducible.

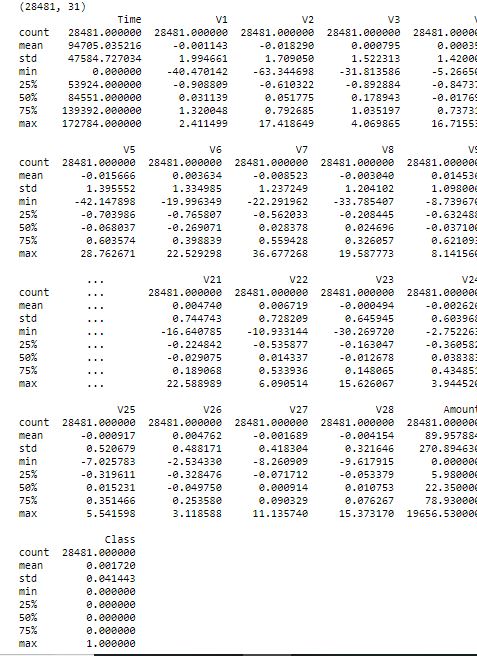


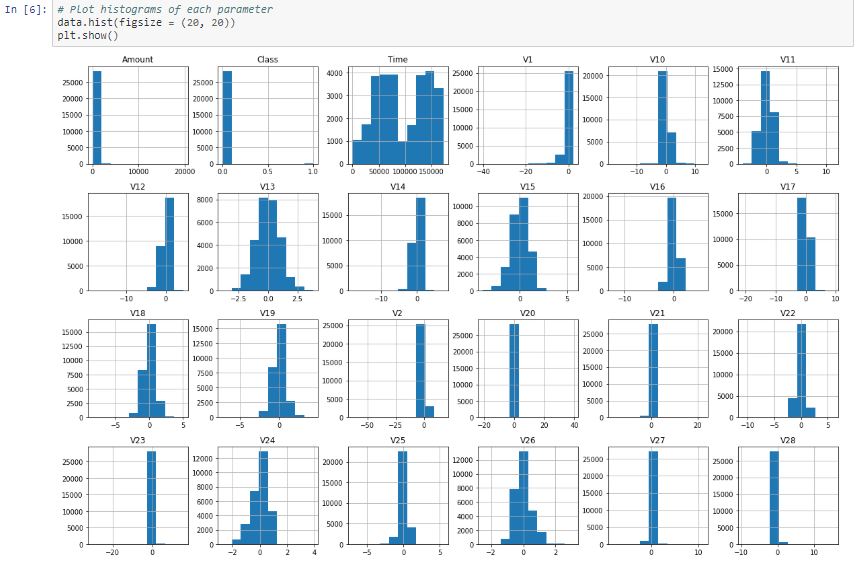


### 2. The Data Set

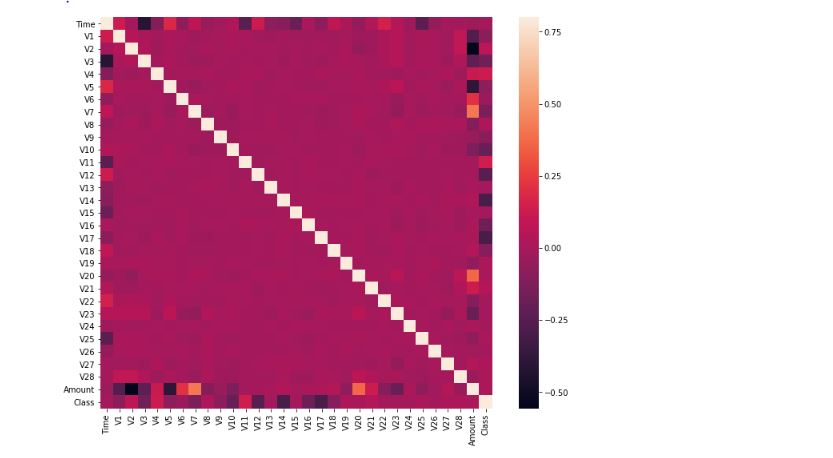
In the following cells, we will import our dataset from a .csv file as a Pandas DataFrame. Furthermore, we will begin exploring the dataset to gain an understanding of the type, quantity, and distribution of data in our dataset. For this purpose, we will use Pandas' built-in describe feature, as well as parameter histograms and a correlation matrix.

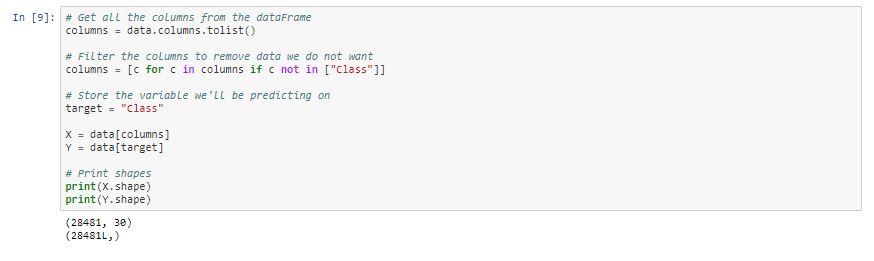












## 3. Unsupervised Outlier Detection

Now that we have processed our data, we can begin deploying our machine learning algorithms. We will use the following techniques:

**Local Outlier Factor (LOF)**

The anomaly score of each sample is called Local Outlier Factor. It measures the local deviation of density of a given sample with respect to its neighbors. It is local in that the anomaly score depends on how isolated the object is with respect to the surrounding neighborhood.

**Isolation Forest Algorithm**

The IsolationForest ‘isolates’ observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature.

Since recursive partitioning can be represented by a tree structure, the number of splittings required to isolate a sample is equivalent to the path length from the root node to the terminating node.

This path length, averaged over a forest of such random trees, is a measure of normality and our decision function

Random partitioning produces noticeably shorter paths for anomalies. Hence, when a forest of random trees collectively produce shorter path lengths for particular samples, they are highly likely to be anomalies.



Local Outlier Factor: 97

0.9965942207085425

precision recall f1-score support

0 1.00 1.00 1.00 28432

1 0.02 0.02 0.02 49

avg / total 1.00 1.00 1.00 28481

Isolation Forest: 71

0.99750711000316

precision recall f1-score support

0 1.00 1.00 1.00 28432

1 0.28 0.29 0.28 49

avg / total 1.00 1.00 1.00 28481

1. **CONCLUSION**

We have successfullydetected fraud credit card using isolation forest algorithm

1. **REFERENCES**
2. <https://en.wikipedia.org/wiki/Python_(programming_language)>
3. <https://en.wikipedia.org/wiki/Project_Jupyter>
4. <https://www.geeksforgeeks.org/isolation forest-algorithms/>
5. <https://www.kaggle.com>
6. https://ieeeexplore.org/document/6554038