IBM COURSERA ADVANCED DATA SCIENCE CAPSTONE

AI POWERED FRAUD DETECTION IN FINANCIAL TRANSACTION

ARCHITECTURAL DECISIONS DOCUMENT

Adapted from “The Lightweight IBM Cloud Garage Method for Data Science”

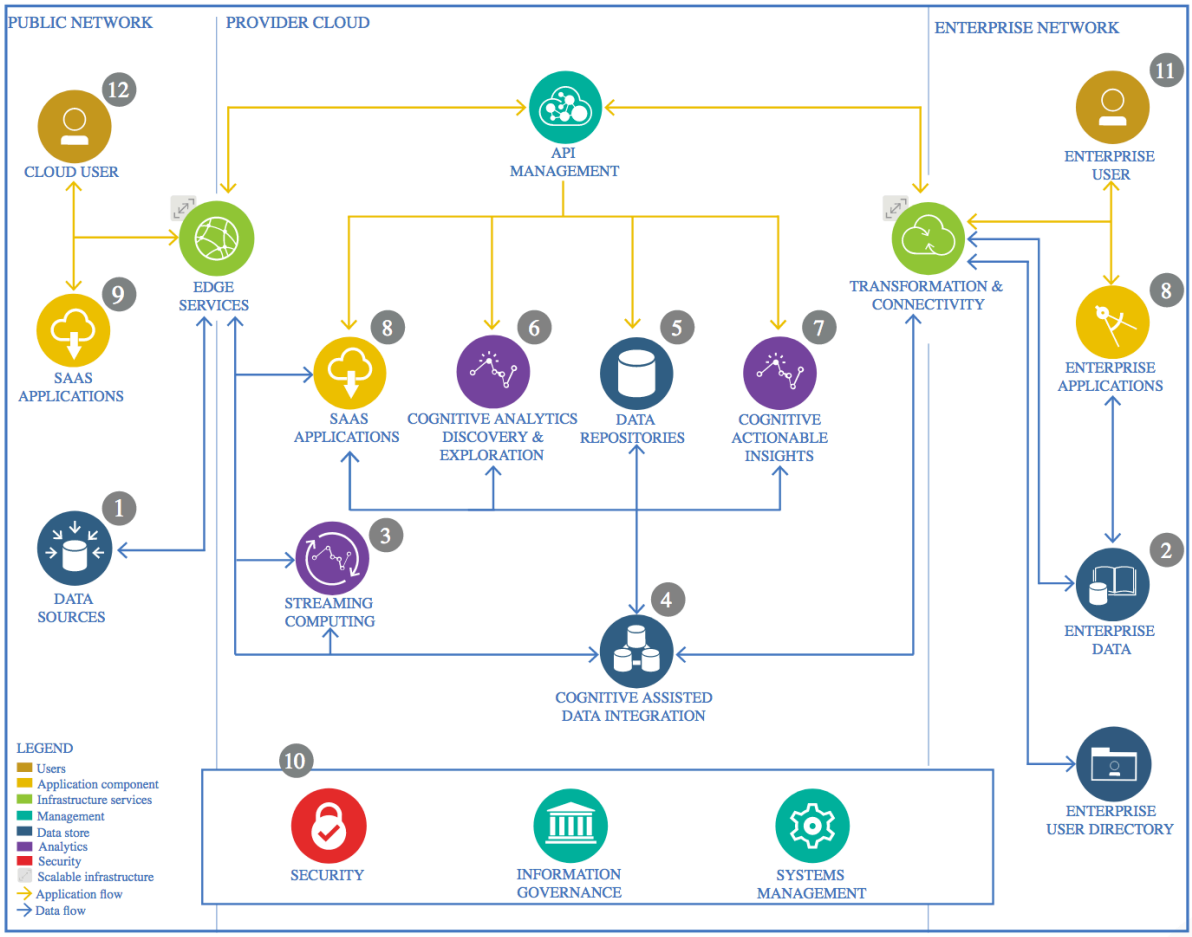
Author:
Francesco di Cugno
March 2020

1 Context

The case study chosen aims at demonstrating how to develop machine learning tools for anomaly detection for extremely rare events. The dataset used is (NTNU, 2017).

The scenario supposes a consultant who is employed to develop a model for automatic anomaly detection.

2 Architectural Components Overview



IBM Data and Analytics Reference Architecture. Source: IBM Corporation

2.1 Data Source

2.1.1 Technology Choice

Data source for model development and testing use payments DB in CSV format. According to (NTNU, 2017) the data has the following structure:

1. *step* - maps a unit of time in the real world. In this case 1 step is 1 hour of time. Total steps 744 (30 days simulation).
2. *type* - CASH-IN, CASH-OUT, DEBIT, PAYMENT and TRANSFER.
3. *amount* - amount of the transaction in local currency.
4. *nameOrig* - customer who started the transaction
5. *oldbalanceOrg* - initial balance before the transaction
6. *newbalanceOrig* - new balance after the transaction
7. *nameDest* - customer who is the recipient of the transaction
8. *oldbalanceDest* - initial balance recipient before the transaction. Note that there is not information for customers that start with M (Merchants).
9. *newbalanceDest* - new balance recipient after the transaction. Note that there is not information for customers that start with M (Merchants).
10. *isFraud* - This is the transactions made by the fraudulent agents inside the simulation. In this specific dataset the fraudulent behavior of the agents aims to profit by taking control or customers accounts and try to empty the funds by transferring to another account and then cashing out of the system.
11. *isFlaggedFraud* - The business model aims to control massive transfers from one account to another and flags illegal attempts. An illegal attempt in this dataset is an attempt to transfer more than 200.000 in a single transaction.

2.1.2 Justification

The scenario supposed refers to a consultant employed to develop a model for a customer. As model development happens offline, a snapshot of the data is needed. CSV file are easy to be generated and are independent from the client architecture. Text files can also be easily compressed and transferred.

2.2 Data Integration

2.2.1 Technology Choice

ETL, data cleansing and feature creation has been done in Python using Jupiter Notebooks.

2.2.2 Justification

Given the amount of transactions in the data set, a distributed solution for model development and training was desirable. Python offers a straightforward integration with Spark that provides parallel computing capabilities, easy to use APIs for data processing, and a SQL layer for data querying. Furthermore, a number of packages are available in case integration with third applications is needed. As the data cleansing and feature selection and/or generation is quite an iterative task, Jupyter notebook allows to take notes about the decision taken for later use.

2.3 Data Repository

2.3.1 Technology Choice

Preprocessed can be stored using a combination of Apache Parquet and numpy standard serialization.

2.3.2 Justification

Raw data has been stored in a cloud object storage. That simplifies the execution of parallel elaborations using Apache Spark.

Apache Parque offers a simple column-based storage format. The framework simplifies the data manipulation and aggregation. Access to data is much faster compared to the CSV and it is possible to store structured data. It is fully integrated in Pyspark and it provide fast serialization/deserialization of Spark RDDs. The framework was the preferred choice to persist processed data. For those cases in which data was processed with scikit-learn, data needs to be converted from RDD to numpy array. The conversion of 4 mil rows takes a good portion of the elaboration time. Numpy serialization can be used to persist numpy arrays of features vectors.

2.4 Discovery and Exploration

2.4.1 Technology Choice

Data exploration was conducted using a Jupyter Notebook that allows to easily describe the analysis and take notes about findings. Due to the size of the dataset, the initial data exploration can be conducted on a 10% sample using Python frameworks such as Pandas, Seaborn, and Matplotlib.

2.4.2 Justification

Jupyter Notebook that allows to easily describe the analysis and take notes about findings. Using a variety of common Data Science Python frameworks such as Pandas, Seaborn, and Mathplotlib allows fast prototyping.

2.5 Actionable Insights

2.5.1 Technology Choice

Predictive models can be developed in Python using Pyspark MLLib, scikit-learn and Keras. SystemML can be used to run Keras models over Spark.

2.5.2 Justification

The proposed frameworks offer implementations of most of the machine learning algorithms. MLLib is natively integrated with Spark and allows to fit and test models on a spark cluster.

2.6 Applications / Data Products

2.6.1 Technology Choice

The developed models can be deployed in “Watson Machine Learning” and expose the service via a REST API.

2.6.2 Justification

REST API provide a great deal of flexibility and abstraction. REST can handle different types of call, return different type of results, and introduce little overhead. Most of the enterprise widely used programming languages provide APIs to use REST services.

2.7 Security, Information Governance and Systems Management

2.7.1 Technology Choice

Secure and controlled access to the APIs is managed via Watson Machine Learning.

2.7.2 Justification

Watson Machine Learning provides a secure gateway that allows steering the access to the APIs. Every client can be provided with an own token to be used while using the service.

3 Bibliography

NTNU, T. @. (2017). *Synthetic Financial Datasets For Fraud Detection (Version 2).* Retrieved 02 01, 2019, from Kaggle: https://www.kaggle.com/ntnu-testimon/paysim1