horizontal line

**Ali & Humberto Company**

Montreal, Toronto, Canada

Prepared for University of Toronto - Class 3760

Credit Risk Assessment in Cloud PoC

**10th November 2021**

# GOALS

1. Run the Credit Risk Assessment model on a public cloud provider
2. Select the most suitable cloud provider for the project
3. Provide a low-latency solution for users to query the credit score (API)
4. Present the results and justify the choices.

# SPECIFICATIONS

Use the Kaggle Credit Score Database <https://www.kaggle.com/c/home-credit-default-risk>

Create a model that returns the credit risk with performance above 75% accuracy (AUC) using the existing database

The GitHub code is available at <https://github.com/humbertosouza/3760creditrisk>

Select the cloud provider that best suits the needs

# MILESTONES

## General Design - PowerPoint, Draw.io, pictures

1. Explain and explore the problem behind the credit risk model.
2. Provide a high-level view of the chosen architecture in draw.io
3. Add supportive information such as graphs, pictures and detail configurations

## Execution

Select between GCP, AWS and Azure. Justify the choice in the PDF file.

Implement the designed solution in the chosen cloud provider.

# THE HOME CREDIT RISK ASSESSMENT

## Credit Default Risk as a Tool for Standardized Decision-Making

Credit risk tools are part of any financial institution to evaluate if money should be borrowed for all kinds of purposes. Some are purpose-built and more accurate for a given business case. \

## Credit Default Risk as an Inclusion Tool

Diverse initiatives across the globe have been designed and executed, some of them being even being awarded the Nobel Prize. In this case, the “Home Credit” has posted in Kaggle a problem that would provide credit to the unbanked population using alternative data such as telco and transactional information, to evaluate default likelihood.

## Input files

2.6 Gbytes dataset distributed amongst 10 CSV files are the input data, totalling 346 columns.

## Outputs

The JSON result indicating the approval/denial and the default likelihood of a given profile sent through an API endpoint

## Enhanced output via API (if there is time)

A simple API that returns the following functions

url.com/reload/N - Allows uploading the CSV files. The N identifies the files to upload where:

POST url.com/get\_credit\_score\_v1

**Parameters**

Authorization: Bearer token

Form-data:

* Record: assessment\_records.csv
* Logged\_user: SSO user email

**returns**

{"score":"Approved","default\_probability":"0.3067828"}

If failed to start returns

404 or 500

## The model

The model has been created by Ali and includes the following steps:

* Data cleansing
* Feature Engineering
* Prediction engine: XGBoost

**Problem Statement**

This is a supervised binary classification problem since the labels are provided in the application\_train table (supervised), and the label is a binary variable with 0 (repaying the loan) and 1 (having difficulty repaying the loan).

**Date Source**

**application\_train**: This table includes the information for each loan application represented by an id of loan (SK\_ID\_CURR). The applicatoin\_train table includes a TARGET column: 1: client with payment difficulties: he/she had late payment more than X days on at least one of the first Y installments of the loan, 0: the loan was repaid)

In addition to application\_train that includes the current application information, there are two other sources of data related to each customer historical transactions and records obtained from Bureau and Home Credit previous application, presented in the following tables:

1. **Bureau**: This table includes information for all client's previous credits provided by other financial institutions that were reported to the Credit Bureau. Each credit in the bureau table is represented by a bureau id (SK\_ID\_BUREAU) which is related to the one id of the loan application (SK\_ID\_CURR). One SK\_ID\_CURR can have 0,1,2 or more related previous credits (SK\_ID\_BUREAU) in a bureau table showing a one-to-many relationship.

2. **Bureau\_balance**: This table includes information related to the monthly balance of previous credits in the Credit Bureau. This table has one row for each month of history of every previous credit reported to Credit Bureau – i.e the table has (#loans in sample, #of relative previous credits, #of months where we have some history observable for the previous credits) rows.

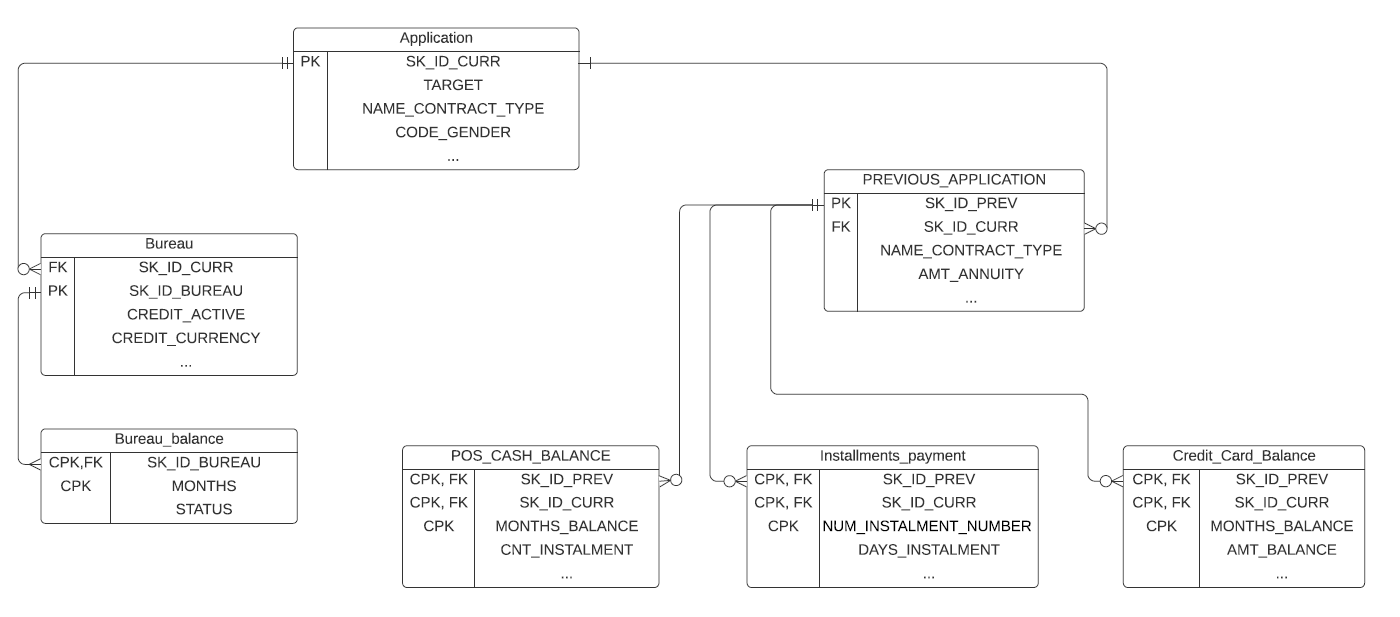
3. **previous\_application**: This table includes all previous applications at Home Credit which are represented by an id of loan (\_\_SK\_ID\_PREV\_\_). One SK\_ID\_CURR can have 0,1,2 or more related previous credits (SK\_ID\_PREV) in the previous\_application table showing a one-to-many relationship.

4. **POS\_CASH\_BALANCE**: This table includes the monthly balance of the previous point of sale (POS) with Home Credit.

5. **credit\_card\_balance**: This table includes monthly balance snapshots of previous credit cards that the applicant has with Home Credit

6. **installments\_payments**: This table includes repayment history for the previously disbursed credits related to the loans in the Home Credit database.

The following Entity Relation Diagram (ERD) shows how different tables are related:



## Preprocessing of data

As shown in the above ERD each SK\_ID\_CURR is related to multiple SK\_ID\_BUREAU in Bureau tables and multiple SK\_ID\_PREV in Home Credit tables. In order to develop a machine learning model, first, we need to flatten out the database. It means aggregating the information from 2 bureau and 4 Home Credit tables to have one merged table. Each row in the final merged table represents one loan application (SK\_ID\_CURR). Manual Feature Engineering has been used to flatten out the data and have one row per application.

**Machine learning**

The dataset of this problem is significantly imbalanced with 91% of data not-defaulted and 9% being defaulted. The challenge of working with an imbalanced dataset is that most machine learning algorithms perform poorly on the minority class that is more important to detect in credit risk management. We have tried boosted algorithms (XGBoost, LightGBM, Catboost) and compared the performance for this dataset.

## The results

The model was simplified for implementation in GCP and usage with XGBoost.

Compared to the original Ali’s code, it can be observed that the performance was xxx.

# 

# SYSTEM ARCHITECTURE

## Total Cost of Ownership

This credit risk model is a proof of concept. However, the successful implementation may allow Home Credit to update it with the latest data, retrain it and link it to other systems via API if necessary.

An on-premises solution would imply buying and maintaining hardware and software for as long as the PoC is active.

Public cloud usage takes out the initial hardware and specialists to set up the equipment. On the other hand, the cost accumulates along the time and is often assigned to OPEX cost centres albeit it could be designed to a CAPEX cost centre.

## Selection of the cloud provider

A comparison between Google Cloud Platform - GCP and Amazon Web Services - AWS has been performed.

Google Cloud platform was chosen thanks to an overall better pricing model and availability of components. Added to that, the implementation in GCP implies less dependency and favours future migration to other public cloud providers.

The Credit Risk model is simple and relatively small, widening the tools available to be considered in both cloud offerings, not allowing the company to be in a good position to negotiate better terms and pricing for this project, considered a proof of concept.

Key components

| **Component** | **Expected Usage** | **AWS** | **GCP** | **Comment** |
| --- | --- | --- | --- | --- |
| Bucket per Gb | 5 Gb inbound; 4 Gb moving within cloud | 2.28 USD / 12 | **USD 0.115 x 12** |  |
| ML Engine per query | 2 runs, notebook, 8 hours of nodes running | **9.63 USD x 12** | USD 4.72 **X 12** |  |
| API Availability per query | NA | NA | NA |  |
| noSQL Database per Gb + access | 30 Mbytes monthly  AWS DynamoDB;  GCP FireStore | **0.30 USD** | **Free** | Audit Database: 1000x 30 = 30000 x 1 KB = 30Mb |
| Functions (exclude Free Tier) | 500 ms of running code per req, 2 Gb of Ram, 1200 reqs per day | **0.62 USD x 12** | **Free** | Doubled the figures, still free for GCP. $3.6 if 10x |
| CloudFront | NA | NA | NA | Relevant when using Lambda@Edge |

It is possible to save X% when GCP is adopted.

A point to consider is the developers’ familiarity with the chosen public cloud provider platform. The same should be considered for integrators if the business sector has a preference. Being this consultancy is a small company, and no further integrations are considered at this point, this was not considered for the present study.

**CLOUD TOOLS SELECTION**

Apart from the costs and selected above, there are a number of supportive websites that went through similar questions. They are listed below.

When to Choose GCP App Engine vs Cloud Run vs Cloud Function

<https://www.sphereinc.com/blogs/when-to-choose-app-engine-vs-cloud-functions-or-cloud-run-in-gcp/>

Cloud Run VS Cloud Functions: What’s the lowest cost?

<https://medium.com/google-cloud/cloud-run-vs-cloud-functions-whats-the-lowest-cost-728d59345a2e>

For the databases, Sourceforge provides interesting side-by-side comparison

<https://sourceforge.net/software/compare/Amazon-DynamoDB-vs-Google-Cloud-Firestore-vs-Openredis-vs-Oracle-Database/>

# SYSTEM DESIGN

Attach below the Draw.io design

## General description

A comparison between Google Cloud Platform - GCP and Amazon Web Services - AWS has been performed.

## Code automation

Explain in a few words how Terraform can be leveraged to build the whole environment. Show a dry-run if it is the case.

# SYSTEM IMPLEMENTATION

## google-gcloud-sdk

Installed on a Ubuntu 20.04. The snap package requests --classic parameter to get installed, and it means the snaps can have access to access outside the snap environment. Although Google is a trusted company, It may not be recommended to be run on computers containing sensitive information.

Service account for accessing the endpoint

Local commands

## Local commands

sudo snap install google-cloud-sdk --classic

gcloud init --console-only

gcloud iam service-accounts create svc-credituser

gcloud projects add-iam-policy-binding credit-risk-assessment --member="serviceAccount:svc-credituser@credit-risk-assessment.iam.gserviceaccount.com" --role="roles/cloudfunctions.invoker"

gcloud projects add-iam-policy-binding credit-risk-assessment --member="serviceAccount:svc-credituser@credit-risk-assessment.iam.gserviceaccount.com" --role="roles/storage.admin"

gcloud projects add-iam-policy-binding credit-risk-assessment --member="serviceAccount:svc-credituser@credit-risk-assessment.iam.gserviceaccount.com" --role="roles/storage.objectViewer"

mkdir ~/gcloud

cd gcloud

gcloud iam service-accounts keys create svc\_tester\_key.json --iam-account=[svc-tester@credit-risk-assessment.iam.gserviceaccount.com](mailto:svc-credituser@credit-risk-assessment.iam.gserviceaccount.com)

export GOOGLE\_APPLICATION\_CREDENTIALS="/home/humberto/gcloud/svc\_credituser\_key.json"

Extract the key for usage in curl

curl https://northamerica-northeast1-credit-risk-assessment.cloudfunctions.net/function-7test -H "Authorization: bearer $(gcloud auth print-identity-token)"

Or for using apps (every few hours)

gcloud auth print-identity-token

**Sending a CSV via CURL**

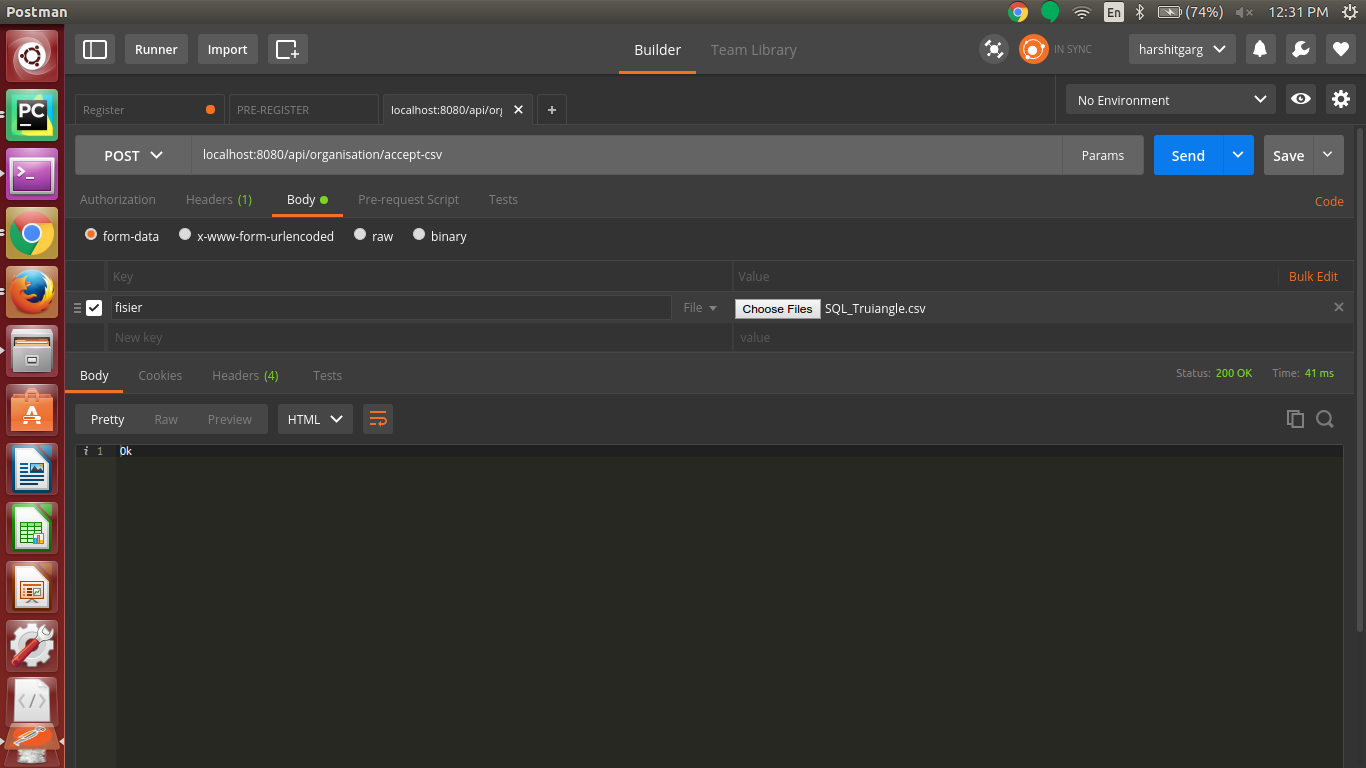
This is to send the payload for evaluating a single record.

curl -L -H "Authorization: bearer $(gcloud auth print-identity-token)" --data-binary @record\_req1.csv "https://northamerica-northeast1-credit-risk-assessment.cloudfunctions.net/function-4test"

**Sending a CSV using POSTMAN**

From <https://stackoverflow.com/questions/43199786/add-csv-file-to-http-post>

1. Select the POST method and type the url
2. In the Body menu header, click on form-data check-box
3. In the key-value form that comes,  
   3.1 add the key as fisier  
   3.2 Choose the type as File from the dropdown near the key  
   3.3 A file chooser button opens, click it, browse the file and upload it  
   3.4 Click on Send method to send the request



https://cloud.google.com/products/calculator/#id=cf7a29b6-b82f-48b4-b2f5-729a99e113fc

# 