Monash University

FIT5202 - Data processing for Big Data (SSB 2024)

Assignment 2B: Using Real-time Streaming Data and Machine Learning to Predict and Visualise Loan Default

Due: **Thursday, 8 February 2024, 11:55 PM**
Worth: 10% of the final marks

Background

Monash **Lo**an **Co**rporation(**MoLoCo**, an imaginary company) is an established loan servicing company offering various home loans, consumer credit and other unsecured loans defined as below.

- Home loan: applicants borrow money from a bank or company to purchase a property(house/apartment), usually the property is used as a security (called a mortgage). If an applicant can't repay the loan, the lender has the right to confiscate the property and sell it.
- Consumer credit: applicants borrow money to purchase goods or services (e.g. an iPhone), and the goods are not secured against the loan. This type of credit usually attracts a higher interest rate. Usually the applicants are required to make a periodical payment set by the contract. As a simple example, if you purchase a $1000 phone with a 20% interest rate over 12 months, you will repay $100 each month. The size of consumer credit is usually small.
- Other unsecured loans: Other types of loans do not require any type of collateral. For example, in an unsecured business loan, applicants borrow money to start a business. If the business fails, the lender could lose money depending on the terms and contracts.

Over many years of operation, they have accumulated many applicants and collected large amounts of data from applicants and applications.

MoLoCo is still using loan assessors and risk managers to process applications, which is slow and inefficient. It plans to execute a digital transformation strategy to optimise operational costs and improve applicant experience. In the first stage, it will utilise big data processing, machine learning and streaming processing to manage loan risk.

The Problem and Project

One of the main risks of a loan business is **default**, which means applicants borrow money from the company but fail to repay for some reason. Traditionally, it takes the loan assessor several days to process a loan application. As a part of the company's digital transformation strategy, it plans to launch a web-based loan application portal and aims to provide a response to a loan application in real-time.

In part A of this assignment, we have already developed a machine learning model to predict loan default. **In this part B**, we will create proof-of-concept **streaming applications** to demonstrate the integration of machine learning models, Kafka, and Spark streaming. Assuming applicants are browsing the website and continuously applying for loans, we will use our model to predict loan default and make decisions. Then, we will visualise the data to give the company a better understanding of its business operation.

What you need to get started

1) Your saved ML model from assignment 2 Part A(A2A).
2) application_data_stream.csv (available from Moodle): We will use this file to simulate incoming loan applications. The schema is the same as A1 and A2A.
3) previous_application_static.csv (available from Moodle): Previous loans of applicants linked by id_app. If id_app cannot be found in this dataset, it means the applicant is new. The schema is the same as A1 and A2A.
4) Value_dict.csv (identical to A1 and A2A). You may or may not need it, depending on the features in your model.

What you need to achieve

MoLoCo requires a proof-of-concept application for ingesting loan application data and predicting the potential default. To achieve this, you need to simulate the streaming data production using Kafka and then build a streaming application that ingests the data and integrates the machine learning model to predict the potential defaults.

A compulsory interview/demo will be arranged during lab 12 after the submission to discuss your proof-of-concept application. Failure to attend this interview will result in 0 marks.

Architecture

The overall architecture of the assignment is represented by the following figure.

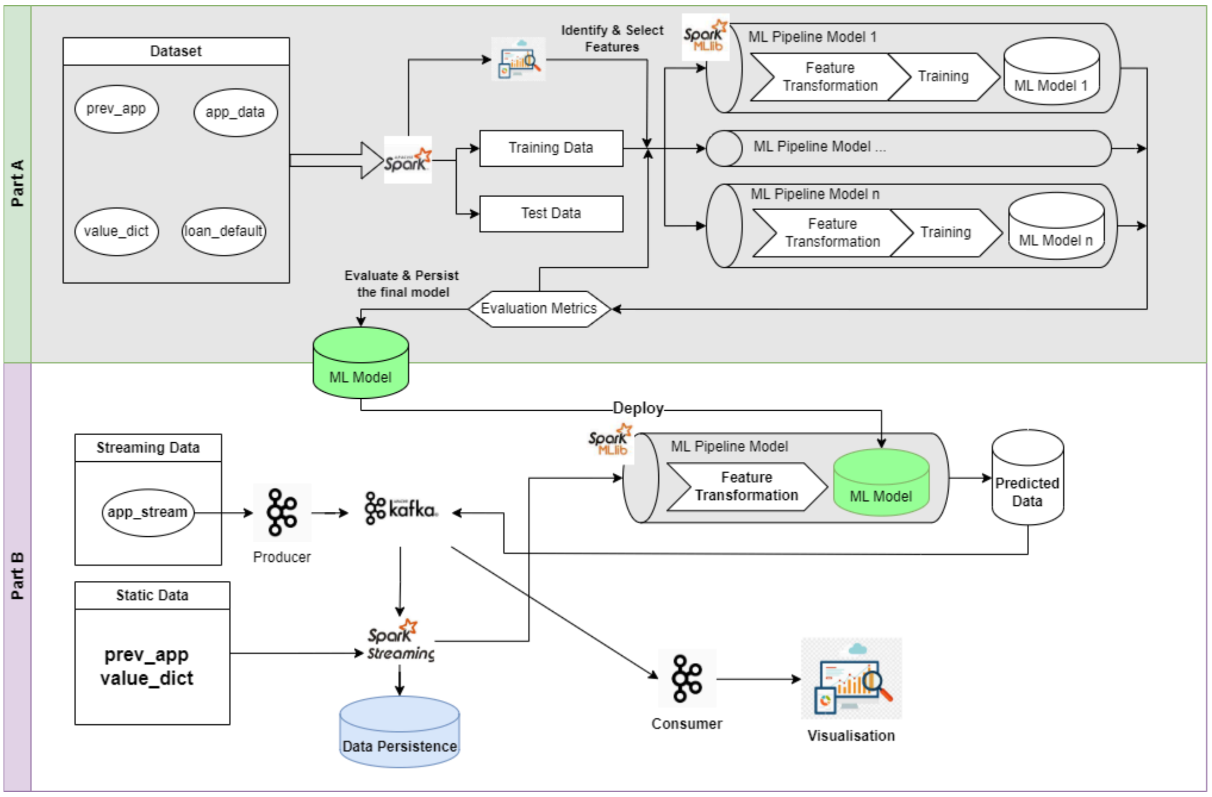


Fig 1: Overall architecture for assignment 2 (We work on Part B)

Part B of the assignment consists of creating data streams in Kafka and joining static data frames with streaming data to create features, using Spark streaming/MLLib to predict potential default, and visualising predicted data.

1. In task 1 (producing the streaming data), you will use the CSV module, Pandas library or other libraries to read and publish loan application data to the Kafka stream.
2. In task 2 (streaming data application), you need to use Spark Structured Streaming and PySpark ML/DataFrame to process the data streams.
3. For task 3 (visualisation), you can use the spark, Pandas library, matplotlib, plotly or other libraries to read the data from the Kafka stream and visualise it.

Please follow the steps to document the processes and write the codes in the Jupyter Notebook.

Getting Started

- Download the dataset from Moodle.
- Download three ipynb template files from Moodle
  - ***A2B-Task1_producer.ipynb*** file for streaming data production
  - ***A2B-Task2_spark_streaming.ipynb*** file for consuming and processing data using Spark Structured Streaming
  - ***A2B-Task3_consumer.ipynb*** file for consuming the data using Kafka and visualise

Important Note:
If you couldn't complete A2A, we still encourage you to attempt this assignment. You can use the "target" column in application_data_stream as the prediction wherever needed. Similarly, if you can't get the streaming join working, you can also do the same. However, if you decide to choose this alternative, the maximum mark you'll receive is capped at 7.9/10(D).

Part 1. Producing the data (10%)

In this task, we will implement **one** Apache Kafka producer to simulate real-time data streaming. Spark is not allowed/required in this part since it's simulating a streaming data source.

1. Your program should send one batch of applications **every 5 seconds**. One batch consists of a **random 100-500 rows** from the application_data_stream dataset. Note that only the number of rows needs to be random, you can read the file sequentially.
   As an example, in the first and second batches, assuming we generate random numbers 100 and 400, the first batch will send records 1-100 from the CSV file, and the second batch will send records 101-500.
   The CSV shouldn't be loaded to memory at once to conserve memory (i.e. Read rows as needed).
2. Add an integer column named '**ts**' for each row, a Unix timestamp in seconds since the epoch (UTC timezone). Spead your batch out evenly for 5 seconds.
   For example, if you send a batch of 100 records at 2024-02-01 00:00:00 (ISO format: YYYY-MM-DD HH:MM:SS) -> (ts = 1704027600):
    - Record 1-20: ts = 1704027600
    - Record 21-40: ts = 1704027601
    - Record 41-60: ts = 1704027602
    - ....
3. Send your batch to a Kafka topic with an appropriate name.

All the data except for the 'ts' column should be sent in **String** type, without changing to other data types. In many streaming processing applications, the data sources usually have little to no processing power (e.g. sensors). To simulate this, we shouldn't do any processing/transformation at the producer.

Save your code in ***Assignment-2B-Task1_producer.ipynb***.

Part 2. Streaming application using Spark Structured Streaming (50%)

In this task, we will implement Spark Structured Streaming to consume the data from task 1 and perform predictive analytics.

Important:
- **This task uses PySpark Structured Streaming with PySpark Dataframe APIs and PySpark ML.**
- **You also need your pipeline model from A2A to make a prediction.**

1. Write code to create a SparkSession with the following requirements: 1) use **four cores** with **a proper application name**; 2) Melbourne timezone; 3) a checkpoint location has been set.
2. Similar to assignment 2A, write code to define the data schema for the data files, following the data types suggested in the metadata file. Load the static datasets(previous_application_static and value_dict) into data frames. (You can reuse your code from 2A.)
3. Using the Kafka topic from the producer in Task 1, read the streaming data with Spark Streaming, assuming all data comes in the **String** format. Except for the 'ts' column, you shall receive it as an **Int** type.
4. Then, transform the streaming data format into proper types following the metadata file schema, similar to assignment 2A. Perform the following tasks:
    a) For the 'ts' column, convert it to the timestamp format, we will use it as **event_time**.
    b) If the data is late for more than 1 minute, discard it.
5. Join the **static** data frames with the streaming data frame, perform data type/column conversion according to your ML model and print out the Schema. (Again, you can reuse code from 2A).
6. Load your ML model, and use the model and Spark to perform the following:
    a) **Every 10 seconds**, print the total number of applications and number of potential default applications (prediction = 1) in the **last 1 minute.**
    b) Every 15 seconds, show the total requested credit (sum of credit where default=0) in the last 15 seconds.
    c) Every 1 minute, show the top 10 largest loan applications with the potential of default.
7. Write a Parquet file to store the following data:
    a. Persist the raw data: Persist your **prediction results** along with **application data** and **event_time** in **Parquet format**; after that, read the Parquet file and show the first 10 records.
    b. Persist the 15-second requested credit (6b) in another parquet file.
8. Read the two parquet files from 7a and 7b as a data stream, and send the records to two topics with appropriate names.
   (Note: **You shall read the parquet files as a streaming data frame and send messages to the Kafka topic when new data appears in the parquet files. The parquet files serve as an intermediate storage in this use case.)**

Save your code in ***Assignment-2B-Task2_spark_streaming.*ipynb**.

Part 3. Consuming data using Kafka and Visualise (20%)

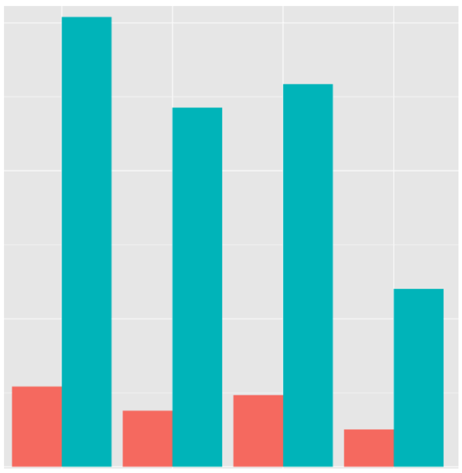
In this task, we will implement an Apache Kafka consumer to consume the data from Part 2.

Important:

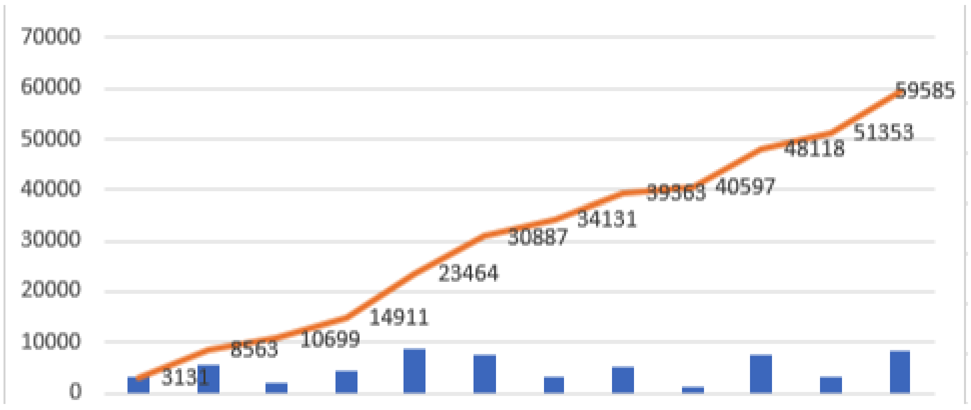
- **In this part, use Kafka consumer to consume the streaming data published from task 2.8.**

- **This part doesn't require parallel processing, you don't need to use Spark. Please use pandas or Python functions to do the simple calculations for visualisation.**

1. Plot a bar chart showing the number of applications and the number of predicted default applications side-by-side(see the illustration below as an example). For every batch, your diagram will be updated with two additional bars. The x-axis is datetime, y-axis is the number of applications.



2. With streaming data from 7b's Kafka topic, plot a cumulative line chart and bar chart of required credits, and annotate every 15 seconds with a numerical valued label. Please see the figure below as an example.



Note: Both plots shall be real-time plots, which will be updated if new streaming data comes in from Part 2. The figures are for illustrative purposes only. Feel free to use your own design/library of choice.

Save your code in ***Assignment-2B-Task3_consumer.ipynb***.

Part 4: Demo and Interview (20%)

IMPORTANT: Interview is compulsory and it is when we mark your A2B. No marks will be awarded if the interview is not attended.

Demo/interview session time will be announced/arranged in lab 12, and please pay attention to the unit announcement email and Ed forum.

Each demo is roughly 10 minutes. You have 5-6 minutes to show your application; then, your marker will ask 3-4 questions to assess your understanding. Please come to your allocated demo session a few minutes early and ensure your laptop/application works correctly.

Demo/Interview is marked on a 5-level scale:

The demo is working and the student has a competent understanding	20
Working demo, partial understanding	15
The demo is not working, partial understanding	10
The demo is not working, low understanding	5
No attendance or can't answer most of the questions	0

Assignment Marking

The marking of this assignment is based on the quality of work you have submitted rather than just quantity. The marking starts from zero and goes up based on the tasks you have completed and their quality, for example, how well the code submitted follows *programming standards, code documentation, presentation of the assignment, readability of the code, reusability of the code, organisation of code, and so on*. Please find the PEP 8 -- Style Guide for Python Code here.

Submission

You should submit your final version of the assignment solution via Moodle. You must submit the following:

- A **zip** file named based on your authcate name (e.g. abcd1234). The zip file should contain
    - ***Assignment-2B-Task1_producer*.ipynb**
    - ***Assignment-2B-Task2_spark_streaming*.ipynb**
    - ***Assignment-2B-Task3_consumer.ipynb***

    The file in submission should be a ZIP file and *not any other kind of compressed folder (e.g. .rar, .7zip, .tar)*. Please **do not include the data files** in the ZIP file.
- The assignment submission should be uploaded and finalised by Thursday, 8 February 2024, 11:55 PM.

Other Information

Where to get help

You can ask questions about the assignment in the Assignments section in the Ed Forum accessible on the unit's Moodle Forum page. This is the preferred venue for assignment clarification-type questions. You should check this forum regularly, as the responses of the teaching staff are "official" and can constitute amendments or additions to the assignment specification. Also, you can attend scheduled consultation sessions if the problem and the confusion are still not solved.

Searching and learning on commercial websites/forums (e.g. Quora, Stack Overflow) is allowed. However, you should not post/ask assignment questions on those forums.

Plagiarism and collusion

Plagiarism and collusion are severe academic offences at Monash University. Students must not share their work with any other students. Students should consult the policy linked below for more information.

https://www.monash.edu/students/academic/policies/academic-integrity

See also the video linked on the Moodle page under the Assignment block.

Students involved in collusion or plagiarism will be subject to disciplinary penalties, which can include:

- The work not being assessed
- A zero grade for the unit
- Suspension from the University
- Exclusion from the University

Late submissions

There is a **10% daily penalty, including weekends,** for a late submission. Also, the cut-off date is seven days after the due date. Submissions will only be accepted after the cut-off date if you have a special consideration.

Note: Assessments submitted more than seven calendar days after the due date without approved special consideration will receive a mark of zero (0) for that assessment task.

ALL Special Consideration, including within the semester, is now handled centrally. Students MUST submit an online Special Consideration form via Monash Connect. For more details, please refer to the **Unit Information** section in Moodle.

Mark Release and Review

- Mark will be released within 10 business days after the submission deadline.
- Reviews and disputes regarding the mark will be accepted a maximum of 7 days after the release date (including weekends).

Generative AI Statement

As per the University's policy on the guidelines and practice pertaining to the usage of Generative AI, all use of generative AI is **restricted** for this assessment. You should **not** use generative artificial intelligence (AI) to generate any materials or content in relation to the assessment task.

The teaching team restricts all use of generative AI to ensure that students apply their own critical thinking and reasoning skills when working on the assessments. In addition, generative AI tools may produce inaccurate content and this could have a negative impact on students' comprehension of big data topics.

Data source acknowledgement:

The dataset is a remix based on several real-world and synthetic datasets. We thank the authors/owners for sharing the original datasets.