ID2221 Data-Intensive Computing Task 4 - Project Report

Group2: Siwei Zhang (siweizh@kth.se), Xuanqi Wang (xuanqi@kth.se), Yudong Wang (yudongwa@kth.se).

This report consists of two parts. The first part summarizes what we did in project and the second part gives the instructions on how to run our codes.

Part 1: Streaming Default Prediction System with Spark

This part covers the background and dataset, what we have done, and our results.

1. The Background and Dataset

Credit risk refers to the risk that a contracted payment will not be made. In simple words, it is the risk of borrower not repaying loan, credit card or any other type of loan. From the bank's perspective, it is important to recognise the credit risk. One major part of the credit risk is the Probability of Default (PD). PD means the likelihood that a borrower will default on debt.

In this project, we built a real-time analyzer for predicting whether the loan applicant is likely to default based on the data information of the loan applicant.

The dataset we used is One Tianchi Competition Dataset. The dataset comes from the loan records of a credit platform, with a total data volume of 0.8 million, including 47 columns of variable information, of which 15 are anonymous variables.

The dataframe schema in the dataset is given as follows.

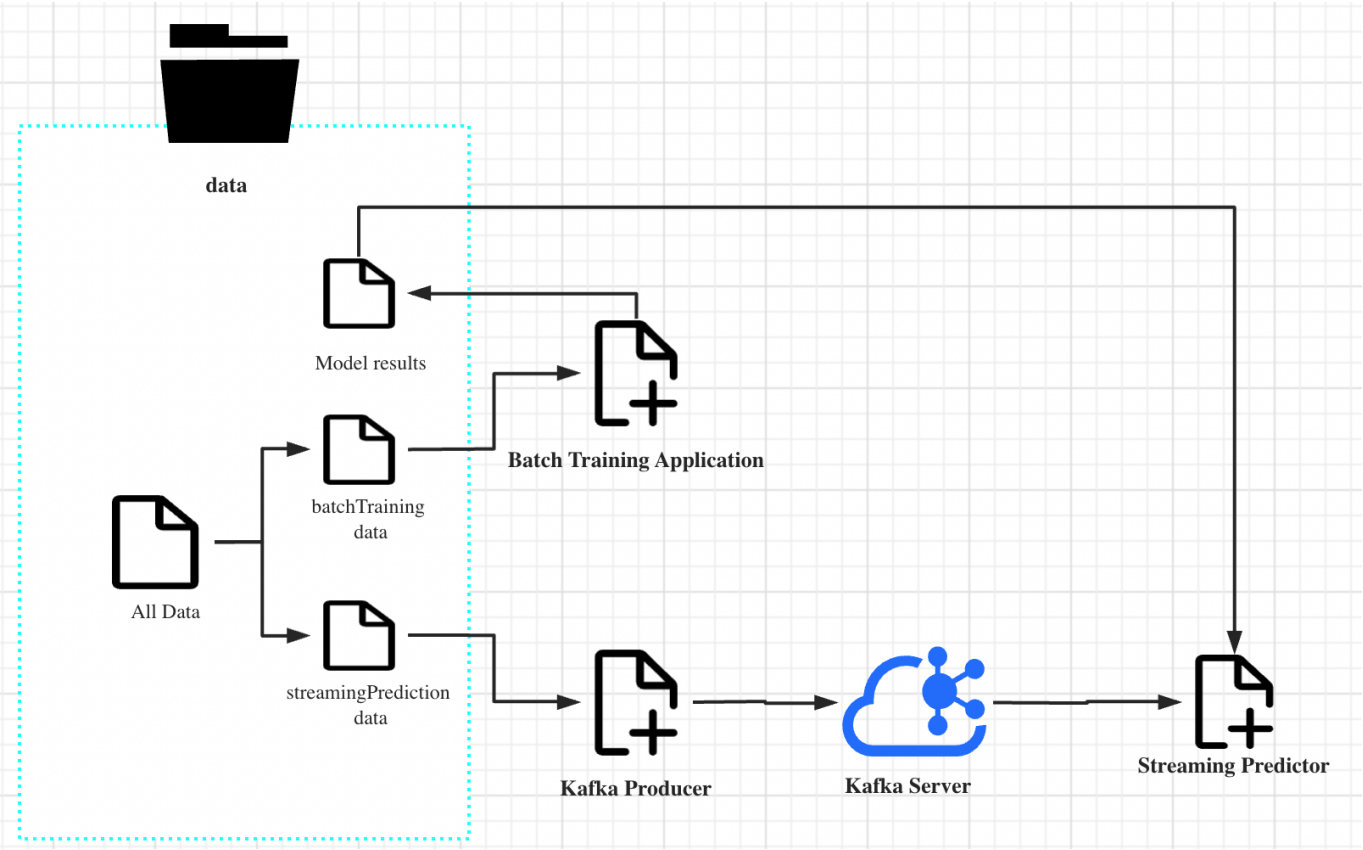
```
['id', 'loanAmnt', 'term', 'interestRate', 'installment', 'grade',
'subGrade', 'employmentTitle', 'employmentLength', 'homeOwnership',
'annualIncome', 'verificationStatus', 'issueDate', 'isDefault',
'purpose', 'postCode', 'regionCode', 'dti', 'delinquency_2years',
'ficoRangeLow', 'ficoRangeHigh', 'openAcc', 'pubRec',
'pubRecBankruptcies', 'revolBal', 'revolUtil', 'totalAcc',
'initialListStatus', 'applicationType', 'earliesCreditLine', 'title',
'policyCode', 'n0', 'n1', 'n2', 'n3', 'n4', 'n5', 'n6', 'n7', 'n8',
'n9', 'n10', 'n11', 'n12', 'n13', 'n14']
```

Each feature has own connotations. Since this is not our focus, in the following context, we will explain some of them if necessay.

2. Detailed Design and Implementation

We designed and implemented a streaming default prediction system based on PySpark and Kafka.

The whole system architecture is given in Figure 1:



- Figure 1: The system architecture.

The system has four components: `Kafka Server`, `Kafka Producer`, `Batch Training Application`, and `Streaming Predictor`.

Data Proprocessing

The whole dataset is 0.8 million, whose `id` is one identifier from 1 to 800,000.

We extracted the first 20000 (id is from 1 to 20000) as `BatchTrainingData.csv`, which was used to train the Linear Regression Model in `Batch Training Application`.

We extracted the next 20000 (id is from 20000 to 40000) as `streamingPredictionData.csv`, which was read by Kafka Producer in chunk and was sent to Kafka Server for simulating the streaming data.

Kafka Server

The `Kafka Server` is one dockerized kafka instance, which we tailored this for our purpose. We mapped the localhost port to the port in the container so that `Kafka Producer` and `Streaming Predictor` can still run normally for localhost Kafka server if docker is not installed.

Batch Training

The `Batch Training` used the `BathTrainingData.csv` to train a linear regression model and saved the model.

In this file, we only used the following features to train the model.

```
["id","loanAmnt", "term", "interestRate", "installment", "homeOwnership",
       "annualIncome", "verificationStatus", "dti", "delinquency_2years",
       "ficoRangeLow", "ficoRangeHigh", "openAcc", "pubRec", "revolBal",
"totalAcc","isDefault", "grade"]
```

The `id` is one identifier and was not used to train. The `isDefault` is the real binary classification label.

The following features are numerical data:

```
["id","loanAmnt", "term", "interestRate", "installment", "homeOwnership",
       "annualIncome", "verificationStatus", "dti", "delinquency_2years",
       "ficoRangeLow", "ficoRangeHigh", "openAcc", "pubRec", "revolBal",
"totalAcc"]
```

Only the feature `dti` has missing values and we used `pyspark.ml.feature.Imputer` to fill missing values. After filling missing values, we first used `pyspark.ml.feature.VectorAssembler` to aggegate these numerical features into one vector and used `pyspark.ml.feature.StandardScaler` to standardize this vector.

Only the feature `grade` is the categorial data and we used `pyspark.ml.feature.StringIndexer` and `pyspark.ml.feature.OneHotEncoder` consecutively to map this column to numerical binary vector.

Then, we use `pyspark.ml.feature.VectorAssembler` to combine numerical and categorial features into one vector and added `pyspark.ml.classification.LogisticRegression` to get the model.

The whole pipeline was:

```
pipeline = Pipeline(stages=[imputer, indexer, onehoter,
num_features_assembler, scaler, scaled_cate_assembler,lr])
```

Kafka Producer

The `Kafka Producer` read the small chunk data from the whole data and sent it (JSON) into Kafka to simulate the streaming data.

Streaming Predictor

The `Streaming Predictor` received data from `Kafka Producer` and used the model generated by `Batch Training Application` to predict the real-time default probability and output the results to the console.

3. Our Results

Firstly, we run the `batchTraining Application` process and got the results in Figure 2:

```
(base) xuanqiwang@p230-n146 Project % python batchTraining.py
[num_features is ['id', 'loanAmnt', 'term', 'interestRate', 'installment', 'homeOwnership', 'annualIncome', 'verificationStatus', 'dti', 'delinquency_2years', 'ficoRangeLow', 'ficoRa]
ngeHigh', 'openAcc', 'pubRec', 'revolBal', 'totalAcc', 'label'],

 cat_features is ['grade']
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
22/10/15 19:55:55 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
22/10/15 19:56:02 WARN package: Truncated the string representation of a plan since it was too large. This behavior can be adjusted by setting 'spark.sql.debug.maxToStringFields'.
The model training on the batch data has been completed and the following is some metrics:

Accuracy of the model: 0.800
+--------------------+--------------------+
|                 FPR|                 TPR|
+--------------------+--------------------+
|                 0.0|                 0.0|
|6.871134986570054E-4|0.002005012531328...|
|0.001061902679742...|0.005263157894736842|
|0.001436691860828284|0.008521303258145364|
|0.001873945905428...|0.011528822055137845|
|0.002436129677056...|0.014035087719298246|
|0.003185708039227...|0.015789473684210527|
|0.003872821537884...|0.017794486215538845|
|0.004372540445999125| 0.02055137844611529|
|0.004934724217627584| 0.02305764411027569|
|0.005496907989256043| 0.02556390977443609|
|0.006246486351427322| 0.02731829573934837|
|0.006933599850084328|0.029323308270676692|
|0.007745643075769879|0.030827067669172932|
|0.008307826847398338| 0.03333333333333333|
| 0.00893247548254107| 0.03558897243107769|
| 0.00949465925416953|  0.0380952380952381|
|0.010056843025797988|  0.0406015037593985|
|0.010619026797426448|  0.0431077694235589|
|0.011118745705540634| 0.04586466165413534|
+--------------------+--------------------+
only showing top 20 rows

areaUnderROC: 0.7016539333842758
```

- Figure 2: The `batchTraining` running results.

As we can see, our model accuracy reached to 80 percent and the AUC was 0.70. Therefore, the model can be used to achieve streaming prediction.

After we started Kafka instance, we started `Kafka Producer` process and `Streaming Predictor` process. The `Kafka Producer` running results are given in Figure 3.

```
(base) xuanqiwang@p230-n146 Project % python kafkaProducer.py -i ./data/streamingPredictionData.csv -c 10 -s 5
Send b'{"id": 20000, "loanAmnt": 11200.0, "term": 5, "interestRate": 14.08, "installment": 261.07, "grade": "C", "subGrade": "C3", "employmentTitle": 55.0, "employmentLength": "1 year", "homeOwnership
": 1, "annualIncome": 72000.0, "verificationStatus": 2, "issueDate": "2017-07-01", "isDefault": 1, "purpose": 3, "postCode": 116.0, "regionCode": 14, "dti": 31.07, "delinquency_2years": 0.0, "ficoRang
eLow": 750.0, "ficoRangeHigh": 754.0, "openAcc": 17.0, "pubRec": 0.0, "pubRecBankruptcies": 0.0, "revolBal": 5656.0, "revolUtil": 7.4, "totalAcc": 61.0, "initialListStatus": 0, "applicationType": 0, "
earliesCreditLine": "Sep-2000", "title": 3.0, "policyCode": 1.0, "n0": 0.0, "n1": 4.0, "n2": 5.0, "n3": 5.0, "n4": 12.0, "n5": 16.0, "n6": 40.0, "n7": 15.0, "n8": 21.0, "n9": 5.0, "n10": 17.0, "n11":
0.0, "n12": 0.0, "n13": 0.0, "n14": 10.0}' to Kafka

Send b'{"id": 20001, "loanAmnt": 19200.0, "term": 5, "interestRate": 14.65, "installment": 453.25, "grade": "C", "subGrade": "C5", "employmentTitle": 96430.0, "employmentLength": "9 years", "homeOwner
ship": 0, "annualIncome": 40000.0, "verificationStatus": 1, "issueDate": "2015-03-01", "isDefault": 1, "purpose": 4, "postCode": 91.0, "regionCode": 7, "dti": 32.28, "delinquency_2years": 0.0, "ficoRa
ngeLow": 695.0, "ficoRangeHigh": 699.0, "openAcc": 11.0, "pubRec": 0.0, "pubRecBankruptcies": 0.0, "revolBal": 36625.0, "revolUtil": 55.7, "totalAcc": 20.0, "initialListStatus": 0, "applicationType":
0, "earliesCreditLine": "Sep-1999", "title": 4.0, "policyCode": 1.0, "n0": 0.0, "n1": 3.0, "n2": 7.0, "n3": 7.0, "n4": 4.0, "n5": 8.0, "n6": 0.0, "n7": 10.0, "n8": 17.0, "n9": 7.0, "n10": 11.0, "n11":
 0.0, "n12": 0.0, "n13": 0.0, "n14": 4.0}' to Kafka

Send b'{"id": 20002, "loanAmnt": 15000.0, "term": 3, "interestRate": 21.85, "installment": 571.7, "grade": "D", "subGrade": "D5", "employmentTitle": 34.0, "employmentLength": "1 year", "homeOwnership"
: 2, "annualIncome": 125000.0, "verificationStatus": 0, "issueDate": "2018-03-01", "isDefault": 1, "purpose": 0, "postCode": 143.0, "regionCode": 8, "dti": 16.21, "delinquency_2years": 2.0, "ficoRange
Low": 660.0, "ficoRangeHigh": 664.0, "openAcc": 20.0, "pubRec": 0.0, "pubRecBankruptcies": 0.0, "revolBal": 21218.0, "revolUtil": 68.7, "totalAcc": 28.0, "initialListStatus": 0, "applicationType": 0,
"earliesCreditLine": "Aug-2002", "title": 0.0, "policyCode": 1.0, "n0": 1.0, "n1": 9.0, "n2": 16.0, "n3": 16.0, "n4": 9.0, "n5": 9.0, "n6": 4.0, "n7": 17.0, "n8": 18.0, "n9": 16.0, "n10": 20.0, "n11":
 0.0, "n12": 0.0, "n13": 1.0, "n14": 1.0}' to Kafka

Send b'{"id": 20003, "loanAmnt": 18750.0, "term": 5, "interestRate": 24.11, "installment": 540.6, "grade": "F", "subGrade": "F2", "employmentTitle": 150532.0, "employmentLength": "6 years", "homeOwner
ship": 1, "annualIncome": 38000.0, "verificationStatus": 2, "issueDate": "2016-03-01", "isDefault": 1, "purpose": 0, "postCode": 25.0, "regionCode": 13, "dti": 15.76, "delinquency_2years": 0.0, "ficoR
angeLow": 705.0, "ficoRangeHigh": 709.0, "openAcc": 8.0, "pubRec": 0.0, "pubRecBankruptcies": 0.0, "revolBal": 18156.0, "revolUtil": 54.4, "totalAcc": 14.0, "initialListStatus": 0, "applicationType":
1, "earliesCreditLine": "Jun-2003", "title": 0.0, "policyCode": 1.0, "n0": 0.0, "n1": 4.0, "n2": 6.0, "n3": 6.0, "n4": 5.0, "n5": 6.0, "n6": 2.0, "n7": 8.0, "n8": 12.0, "n9": 6.0, "n10": 8.0, "n11": 0
.0, "n12": 0.0, "n13": 0.0, "n14": 0.0}' to Kafka

Send b'{"id": 20004, "loanAmnt": 20000.0, "term": 3, "interestRate": 12.69, "installment": 670.9, "grade": "C", "subGrade": "C2", "employmentTitle": 19.0, "employmentLength": "10+ years", "homeOwnersh
ip": 0, "annualIncome": 100000.0, "verificationStatus": 2, "issueDate": "2015-03-01", "isDefault": 0, "purpose": 0, "postCode": 361.0, "regionCode": 8, "dti": 29.91, "delinquency_2years": 0.0, "ficoRa
ngeLow": 660.0, "ficoRangeHigh": 664.0, "openAcc": 12.0, "pubRec": 0.0, "pubRecBankruptcies": 0.0, "revolBal": 11197.0, "revolUtil": 63.3, "totalAcc": 30.0, "initialListStatus": 1, "applicationType":
0, "earliesCreditLine": "Nov-1999", "title": 0.0, "policyCode": 1.0, "n0": 0.0, "n1": 2.0, "n2": 4.0, "n3": 4.0, "n4": 2.0, "n5": 3.0, "n6": 12.0, "n7": 6.0, "n8": 13.0, "n9": 4.0, "n10": 12.0, "n11":
 0.0, "n12": 0.0, "n13": 0.0, "n14": 1.0}' to Kafka

Send b'{"id": 20005, "loanAmnt": 28000.0, "term": 5, "interestRate": 13.99, "installment": 651.37, "grade": "C", "subGrade": "C4", "employmentTitle": 46579.0, "employmentLength": "8 years", "homeOwner
ship": 0, "annualIncome": 83000.0, "verificationStatus": 0, "issueDate": "2015-09-01", "isDefault": 1, "purpose": 4, "postCode": 203.0, "regionCode": 30, "dti": 23.13, "delinquency_2years": 0.0, "fico
RangeLow": 675.0, "ficoRangeHigh": 679.0, "openAcc": 17.0, "pubRec": 0.0, "pubRecBankruptcies": 0.0, "revolBal": 47037.0, "revolUtil": 80.0, "totalAcc": 26.0, "initialListStatus": 0, "applicationType"
: 0, "earliesCreditLine": "Jul-1990", "title": 4.0, "policyCode": 1.0, "n0": 0.0, "n1": 8.0, "n2": 14.0, "n3": 14.0, "n4": 8.0, "n5": 10.0, "n6": 2.0, "n7": 15.0, "n8": 20.0, "n9": 14.0, "n10": 17.0,
"n11": 0.0, "n12": 0.0, "n13": 0.0, "n14": 3.0}' to Kafka

^CTraceback (most recent call last):
  File "kafkaProducer.py", line 52, in <module>
    time.sleep(sleeptime)
KeyboardInterrupt

(base) xuanqiwang@p230-n146 Project %
```

- Figure 3: The `Kafka Producer` running results.

In Figure 3, the `-c 10` means the chunksize is 10 when this process read the original data, i.e., `StreamingPredictionData.csv`. In this scenario, the process read the 10 lines at one time.

The `-s 5` means the process read the data every five seconds. In this way, the producer send one line(one user) every five second so that we can read the results of `Streaming Predictor` more clearly.

The `Streaming Predictor` running results are given in Figure 4-6.

```
(base) xuanqiwang@p230-n146 Project % spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.12:3.3.0 streamingPredictor.py
:: loading settings :: url = jar:file:/Users/xuanqiwang/opt/anaconda3/lib/python3.8/site-packages/pyspark/jars/ivy-2.5.0.jar!/org/apache/ivy/core/settings/ivysettings.xml
Ivy Default Cache set to: /Users/xuanqiwang/.ivy2/cache
The jars for the packages stored in: /Users/xuanqiwang/.ivy2/jars
org.apache.spark#spark-sql-kafka-0-10_2.12 added as a dependency
:: resolving dependencies :: org.apache.spark#spark-submit-parent-436fe3a8-9f86-427d-9eac-b106c987c3c9;1.0
        confs: [default]
        found org.apache.spark#spark-sql-kafka-0-10_2.12;3.3.0 in central
        found org.apache.spark#spark-token-provider-kafka-0-10_2.12;3.3.0 in central
        found org.apache.kafka#kafka-clients;2.8.1 in central
        found org.lz4#lz4-java;1.8.0 in central
        found org.xerial.snappy#snappy-java;1.1.8.4 in central
        found org.slf4j#slf4j-api;1.7.32 in central
        found org.apache.hadoop#hadoop-client-runtime;3.3.2 in central
        found org.spark-project.spark#unused;1.0.0 in central
        found org.apache.hadoop#hadoop-client-api;3.3.2 in central
        found commons-logging#commons-logging;1.1.3 in central
        found com.google.code.findbugs#jsr305;3.0.0 in central
        found org.apache.commons#commons-pool2;2.11.1 in central
:: resolution report :: resolve 279ms :: artifacts dl 7ms
        :: modules in use:
        com.google.code.findbugs#jsr305;3.0.0 from central in [default]
        commons-logging#commons-logging;1.1.3 from central in [default]
        org.apache.commons#commons-pool2;2.11.1 from central in [default]
        org.apache.hadoop#hadoop-client-api;3.3.2 from central in [default]
        org.apache.hadoop#hadoop-client-runtime;3.3.2 from central in [default]
        org.apache.kafka#kafka-clients;2.8.1 from central in [default]
        org.apache.spark#spark-sql-kafka-0-10_2.12;3.3.0 from central in [default]
        org.apache.spark#spark-token-provider-kafka-0-10_2.12;3.3.0 from central in [default]
        org.lz4#lz4-java;1.8.0 from central in [default]
        org.slf4j#slf4j-api;1.7.32 from central in [default]
        org.spark-project.spark#unused;1.0.0 from central in [default]
        org.xerial.snappy#snappy-java;1.1.8.4 from central in [default]
        ---------------------------------------------------------------------
        |                  |            modules            ||   artifacts   |
        |       conf       | number| search|dwnlded|evicted|| number|dwnlded|
        ---------------------------------------------------------------------
        |      default     |   12  |   0   |   0   |   0   ||   12  |   0   |
        ---------------------------------------------------------------------
:: retrieving :: org.apache.spark#spark-submit-parent-436fe3a8-9f86-427d-9eac-b106c987c3c9
        confs: [default]
        0 artifacts copied, 12 already retrieved (0kB/5ms)
22/10/15 20:05:36 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
22/10/15 20:05:38 INFO SparkContext: Running Spark version 3.3.0
22/10/15 20:05:38 INFO ResourceUtils: ==============================================================
22/10/15 20:05:38 INFO ResourceUtils: No custom resources configured for spark.driver.
22/10/15 20:05:38 INFO ResourceUtils: ==============================================================
22/10/15 20:05:38 INFO SparkContext: Submitted application: streamingPredictor.py
22/10/15 20:05:38 INFO ResourceProfile: Default ResourceProfile created, executor resources: Map(cores -> name: cores, amount: 1, script: , vendor: , memory -> name: memory, amount: 1024, script: , vendor: , offHeap -> name: offHeap, amount: 0, script: , vendor: ), task resources: Map(cpus -> name: cpus, amount: 1.0)
22/10/15 20:05:38 INFO ResourceProfile: Limiting resource is cpu
22/10/15 20:05:38 INFO ResourceProfileManager: Added ResourceProfile id: 0
22/10/15 20:05:38 INFO SecurityManager: Changing view acls to: xuanqiwang
22/10/15 20:05:38 INFO SecurityManager: Changing modify acls to: xuanqiwang
22/10/15 20:05:38 INFO SecurityManager: Changing view acls groups to:
22/10/15 20:05:38 INFO SecurityManager: Changing modify acls groups to:
22/10/15 20:05:38 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users  with view permissions: Set(xuanqiwang); groups with view permissions: Set(); users  with modify permissions: Set(xuanqiwang); groups with modify permissions: Set()
22/10/15 20:05:38 INFO Utils: Successfully started service 'sparkDriver' on port 61117.
22/10/15 20:05:38 INFO SparkEnv: Registering MapOutputTracker
22/10/15 20:05:38 INFO SparkEnv: Registering BlockManagerMaster
22/10/15 20:05:38 INFO BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopologyMapper for getting topology information
22/10/15 20:05:38 INFO BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
22/10/15 20:05:38 INFO SparkEnv: Registering BlockManagerMasterHeartbeat
22/10/15 20:05:38 INFO DiskBlockManager: Created local directory at /private/var/folders/p_/mhcmqf_j0hq2cx954d5d7l5c0000gn/T/blockmgr-a843e8c1-1042-40c5-b172-05337b6fde68
22/10/15 20:05:38 INFO MemoryStore: MemoryStore started with capacity 366.3 MiB
```

- Figure 4: The first part `Streaming Predictor` running results.

In Figure 4, we started the process using

> spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.12:3.3.0 streamingPredictor.py.

```
 Starting receving streaming data and prediction:

22/10/15 20:05:40 WARN package: Truncated the string representation of a plan since it was too large. This behavior can be adjusted by setting 'spark.sql.debug.maxToStringFields'.
All features of received data are given as follows:

root
 |-- id: integer (nullable = true)
 |-- loanAmnt: double (nullable = true)
 |-- term: integer (nullable = true)
 |-- interestRate: double (nullable = true)
 |-- installment: double (nullable = true)
 |-- grade: string (nullable = true)
 |-- subGrade: string (nullable = true)
 |-- employmentTitle: double (nullable = true)
 |-- employmentLength: string (nullable = true)
 |-- homeOwnership: integer (nullable = true)
 |-- annualIncome: double (nullable = true)
 |-- verificationStatus: integer (nullable = true)
 |-- issueDate: timestamp (nullable = true)
 |-- isDefault: integer (nullable = true)
 |-- purpose: integer (nullable = true)
 |-- postCode: double (nullable = true)
 |-- regionCode: integer (nullable = true)
 |-- dti: double (nullable = true)
 |-- delinquency_2years: double (nullable = true)
 |-- ficoRangeLow: double (nullable = true)
 |-- ficoRangeHigh: double (nullable = true)
 |-- openAcc: double (nullable = true)
 |-- pubRec: double (nullable = true)
 |-- pubRecBankruptcies: double (nullable = true)
 |-- revolBal: double (nullable = true)
 |-- revolUtil: double (nullable = true)
 |-- totalAcc: double (nullable = true)
 |-- initialListStatus: integer (nullable = true)
 |-- applicationType: integer (nullable = true)
 |-- earliesCreditLine: string (nullable = true)
 |-- title: double (nullable = true)
 |-- policyCode: double (nullable = true)
 |-- n0: double (nullable = true)
 |-- n1: double (nullable = true)
 |-- n2: double (nullable = true)
 |-- n3: double (nullable = true)
 |-- n4: double (nullable = true)
 |-- n5: double (nullable = true)
 |-- n6: double (nullable = true)
 |-- n7: double (nullable = true)
 |-- n8: double (nullable = true)
 |-- n9: double (nullable = true)
 |-- n10: double (nullable = true)
 |-- n11: double (nullable = true)
 |-- n12: double (nullable = true)
 |-- n13: double (nullable = true)
 |-- n14: double (nullable = true)

After features filtering, use the following features to predict the probability:

root
 |-- id: integer (nullable = true)
 |-- isDefault: integer (nullable = true)
 |-- loanAmnt: double (nullable = true)
 |-- term: integer (nullable = true)
 |-- interestRate: double (nullable = true)
 |-- installment: double (nullable = true)
 |-- homeOwnership: integer (nullable = true)
 |-- annualIncome: double (nullable = true)
 |-- verificationStatus: integer (nullable = true)
 |-- dti: double (nullable = true)
 |-- delinquency_2years: double (nullable = true)
 |-- ficoRangeLow: double (nullable = true)
 |-- ficoRangeHigh: double (nullable = true)
 |-- openAcc: double (nullable = true)
 |-- pubRec: double (nullable = true)
 |-- revolBal: double (nullable = true)
 |-- totalAcc: double (nullable = true)
 |-- grade: string (nullable = true)
```

- Figure 5: The second part `Streaming Predictor` running results.

In Figure 5 above, we showed the schema of received data and data after feature filtering.

```
-------------------------------------------
Batch: 0
-------------------------------------------
| id|isDefault|loanAmnt|term|interestRate|installment|homeOwnership|annualIncome|verificationStatus|dti|delinquency_2years|ficoRangeLow|ficoRangeHigh|openAcc|pubRec|revolBal|totalAcc|grade|gradeIndex|gradeIndexVect|num_features_vec|scaled_num_features_vec|features|rawPrediction|probability|prediction|

-------------------------------------------
Batch: 1
-------------------------------------------
|   id|isDefault|loanAmnt|term|interestRate|installment|homeOwnership|annualIncome|verificationStatus|  dti|delinquency_2years|ficoRangeLow|ficoRangeHigh|openAcc|pubRec|revolBal|totalAcc|grade|gradeIndex|gradeIndexVect|    num_features_vec|scaled_num_features_vec|            features|       rawPrediction|         probability|prediction|
|20000|        1| 11200.0|   5|       14.08|     261.07|            1|     72000.0|                 2|31.07|               0.0|       750.0|        754.0|   17.0|   0.0|  5656.0|    61.0|    C|       1.0| (6,[1],[1.0])|[11200.0,5.0,14.0...|   [1.28196976854495...|[1.28196976854495...|[1.28657442289906...|[0.78356681035850...|       0.0|

-------------------------------------------
Batch: 2
-------------------------------------------
|   id|isDefault|loanAmnt|term|interestRate|installment|homeOwnership|annualIncome|verificationStatus|  dti|delinquency_2years|ficoRangeLow|ficoRangeHigh|openAcc|pubRec|revolBal|totalAcc|grade|gradeIndex|gradeIndexVect|    num_features_vec|scaled_num_features_vec|            features|       rawPrediction|         probability|prediction|
|20001|        1| 19200.0|   5|       14.65|     453.25|            0|     40000.0|                 1|32.28|               0.0|       695.0|        699.0|   11.0|   0.0| 36625.0|    20.0|    C|       1.0| (6,[1],[1.0])|[19200.0,5.0,14.6...|   [2.19766246036277...|[2.19766246036277...|[1.07973574210507...|[0.74644397172700...|       0.0|

-------------------------------------------
Batch: 3
-------------------------------------------
|   id|isDefault|loanAmnt|term|interestRate|installment|homeOwnership|annualIncome|verificationStatus|  dti|delinquency_2years|ficoRangeLow|ficoRangeHigh|openAcc|pubRec|revolBal|totalAcc|grade|gradeIndex|gradeIndexVect|    num_features_vec|scaled_num_features_vec|            features|       rawPrediction|         probability|prediction|
|20002|        1| 15000.0|   3|       21.85|      571.7|            2|    125000.0|                 0|16.21|               2.0|       660.0|        664.0|   20.0|   0.0| 21218.0|    28.0|    D|       3.0| (6,[3],[1.0])|[15000.0,3.0,21.8...|   [1.71692379715841...|[1.71692379715841...|[0.92692634014940...|[0.71645129069712...|       0.0|

-------------------------------------------
Batch: 4
-------------------------------------------
|   id|isDefault|loanAmnt|term|interestRate|installment|homeOwnership|annualIncome|verificationStatus|  dti|delinquency_2years|ficoRangeLow|ficoRangeHigh|openAcc|pubRec|revolBal|totalAcc|grade|gradeIndex|gradeIndexVect|    num_features_vec|scaled_num_features_vec|            features|       rawPrediction|         probability|prediction|
|20003|        1| 18750.0|   5|       24.11|      540.6|            1|     38000.0|                 2|15.76|               0.0|       705.0|        709.0|    8.0|   0.0| 18156.0|    14.0|    F|       5.0| (6,[5],[1.0])|[18750.0,5.0,24.1...|   [2.14615474644802...|[2.14615474644802...|[0.56254633757053...|[0.63704150870409...|       0.0|

-------------------------------------------
Batch: 5
-------------------------------------------
|   id|isDefault|loanAmnt|term|interestRate|installment|homeOwnership|annualIncome|verificationStatus|  dti|delinquency_2years|ficoRangeLow|ficoRangeHigh|openAcc|pubRec|revolBal|totalAcc|grade|gradeIndex|gradeIndexVect|    num_features_vec|scaled_num_features_vec|            features|       rawPrediction|         probability|prediction|
|20004|        0| 20000.0|   3|       12.69|      670.9|            0|    100000.0|                 2|29.91|               0.0|       660.0|        664.0|   12.0|   0.0| 11197.0|    30.0|    C|       1.0| (6,[1],[1.0])|[20000.0,3.0,12.6...|   [2.28923172954455...|[2.28923172954455...|[1.21620133956347...|[0.77139436496770...|       0.0|

-------------------------------------------
Batch: 6
-------------------------------------------
|   id|isDefault|loanAmnt|term|interestRate|installment|homeOwnership|annualIncome|verificationStatus|  dti|delinquency_2years|ficoRangeLow|ficoRangeHigh|openAcc|pubRec|revolBal|totalAcc|grade|gradeIndex|gradeIndexVect|    num_features_vec|scaled_num_features_vec|            features|       rawPrediction|         probability|prediction|
|20005|        1| 28000.0|   5|       13.99|     651.37|            0|     83000.0|                 0|23.13|               0.0|       675.0|        679.0|   17.0|   0.0| 47037.0|    26.0|    C|       1.0| (6,[1],[1.0])|[28000.0,5.0,13.9...|   [3.20492442136237...|(21,[0,1,2,3,5,7,...|[1.10559659161862...|[0.75130726889861...|       0.0|
```

- Figure 6: The second part `Streaming Predictor` running results.

In Figure 6, the micro batch 0 has no results since the process was waiting for data. Then, in micro batch 1, the data whose `id` is 20000 arrived and the predictor get the probability and prediction in the final two column. In the following micro batches, the data whose `id` was from 20001 to 20005 arrived and got their prediction results.

Part 2: Deployment

We deploy the following components step by step:

1. **`Batch Training Application`**,
2. **`Kafka Server`**,
3. **`Streaming Predictor`**,
4. **`Kafka Producer`**.

You may need root access to install components and run docker. All the docker images are by default pulled from `docker-hub`.

1. `Batch Training Application`:

Run the following command:

```
python batchTraining.py
```

2. `Kafka Server`:

Run the following command:

```
docker-compose up -d
```

To stop the Kafka instance, use the following to exit.

```
docker-compose stop
```

3. `Streaming Predictor`:

1. run the following command.

```
spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.12:3.3.0
streamingPredictor.py
```

4. `Kafka Producer`:

To clearly show the predict results, run the following command:

```
python kafkaProducer.py -i ./data/streamingPredictionData.csv -c 10 -s 5
```

To simulate real-time streaming data, run the following command:

```
python kafkaProducer.py -i ./data/streamingPredictionData.csv -c 10 -s 0
```