AI as a Service

Build Shared AI Service Platforms Based on Deep Learning Technologies

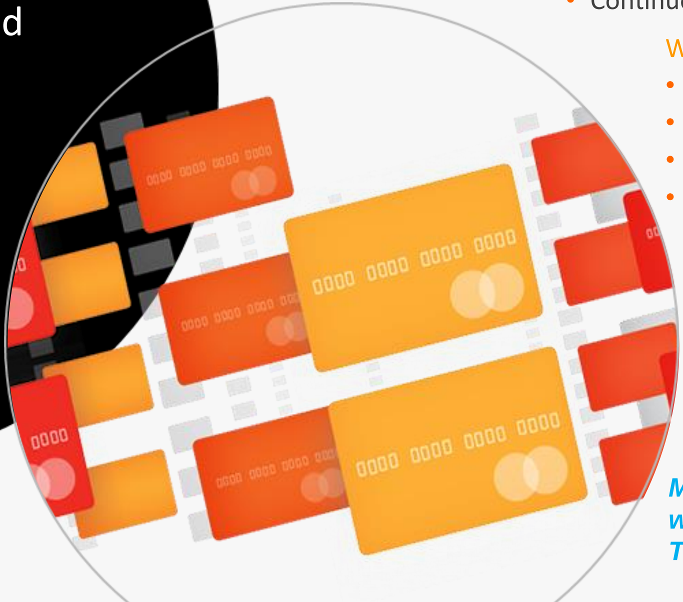
Suqiang Song, Director, Chapter Leader of Data Engineering & AI
Mastercard

#AI1SAIS

Mastercard Big Data & AI Expertise

Differentiation starts with consumer insights from a massive worldwide payments network and our experience in data cleansing, analytics and modeling

What can
2.4 BILLION
Global Cards and
56 BILLION
Transactions/
Year mean
to you?



MULTI-SOURCED

- **38MM+** merchant locations
- **22,000** issuers

CLEANSED, AGGREGATD, ANONYMOUS, AUGMENTED

- **1.5MM** automated rules
- Continuously tested

WAREHOUSED

- **10** petabytes
- **5+** year historic global view
- Rapid retrieval
- Above-and-beyond privacy protection and security

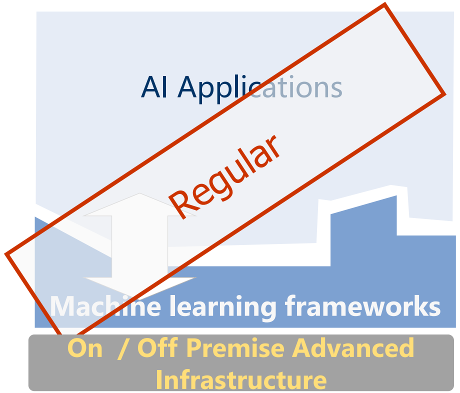
TRANSFORMED INTO ACTIONABLE INSIGHTS

- Reports, indexes, benchmarks
- Behavioral variables
- Models, scores, forecasting
- Econometrics

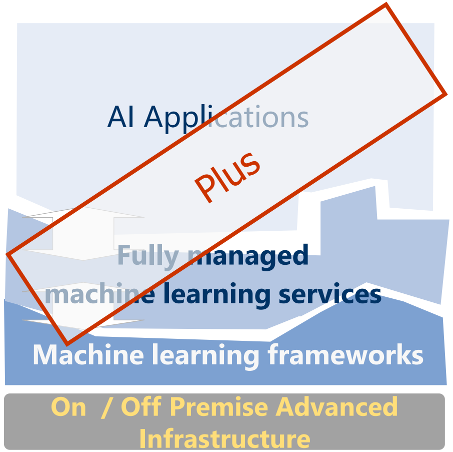
Mastercard Enhanced Artificial Intelligence Capability with the Acquisitions of Applied Predictive Technologies(2015) and Brighterion (2017)

What is the AI as a Service ?

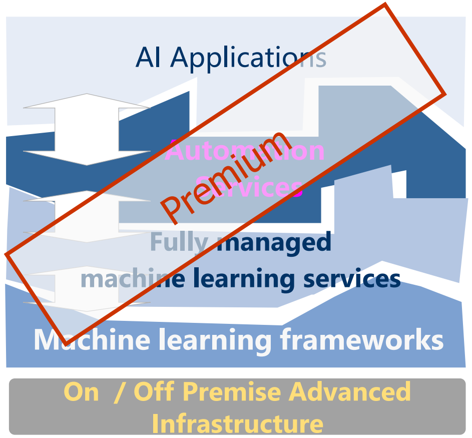
Three modes of AI as a Services



- Machine learning frameworks: Provide stable and secure environments and consolidate integrated wrappers on top of variable technologies for regular machine learning works
- Applications build silos from scratch



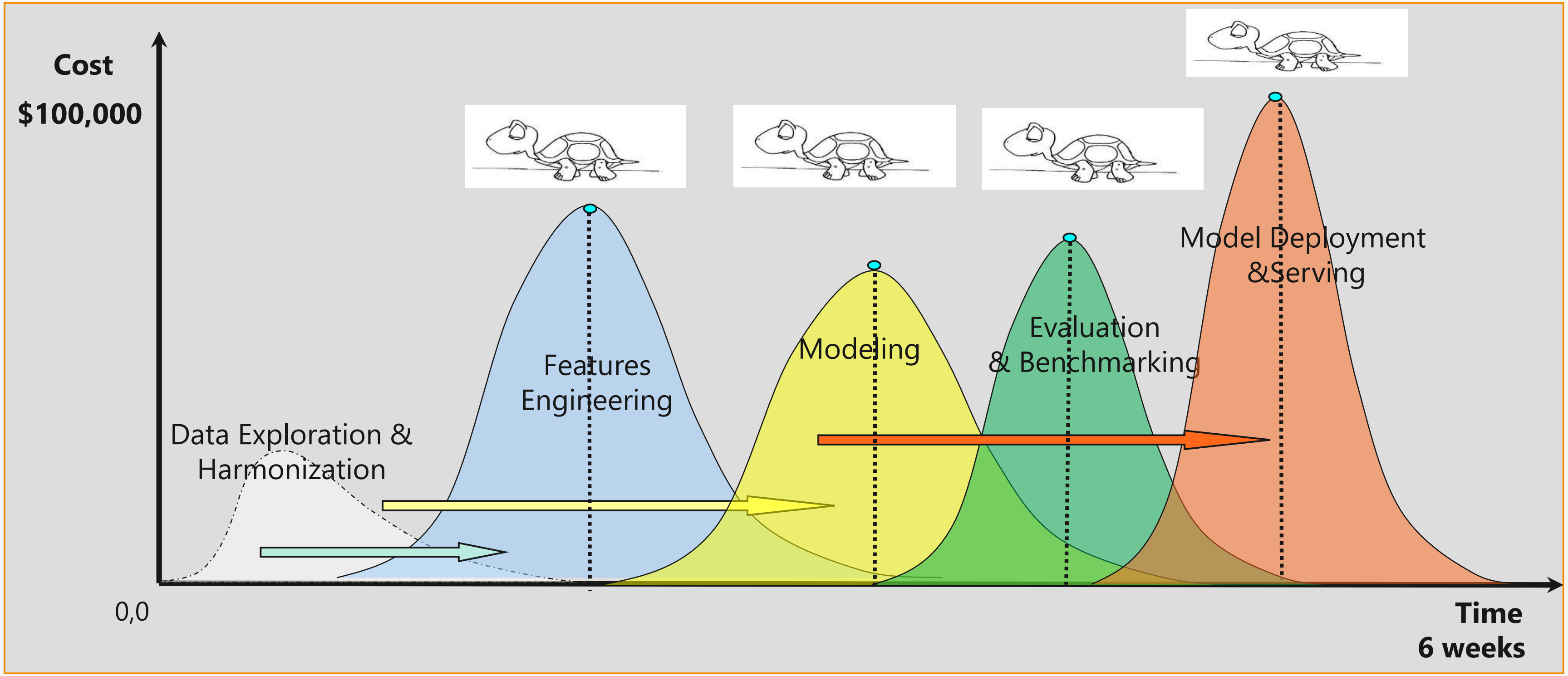
- Fully managed machine learning servic es use templates, pre-built models and drag-and-drop development tools to si mplify and expedite the process of usin g a machine learning framework
- Applications share templates and pre-built models , assembly and infer them into pipelines or business context



- Automation Services, tasks like explora tory data analysis, pre-processing of da ta, hyper-parameter tuning, model sele ction and putting models into producti on can be automated
- “God's Return to God, Satan's Return to Satan , Math’s Return AI, Business’s Return Biz”

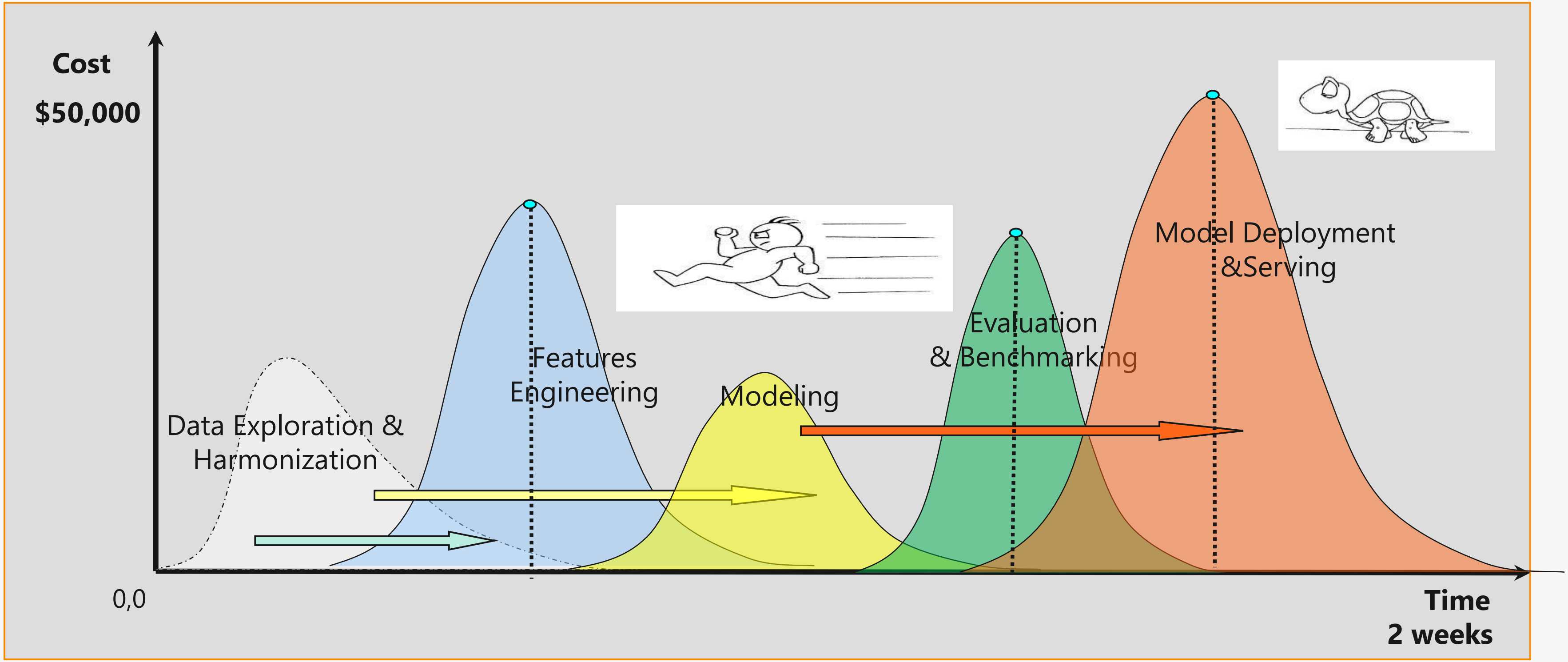
Regular Mode :Machine learning frameworks

Example : Machine Learning Sandbox



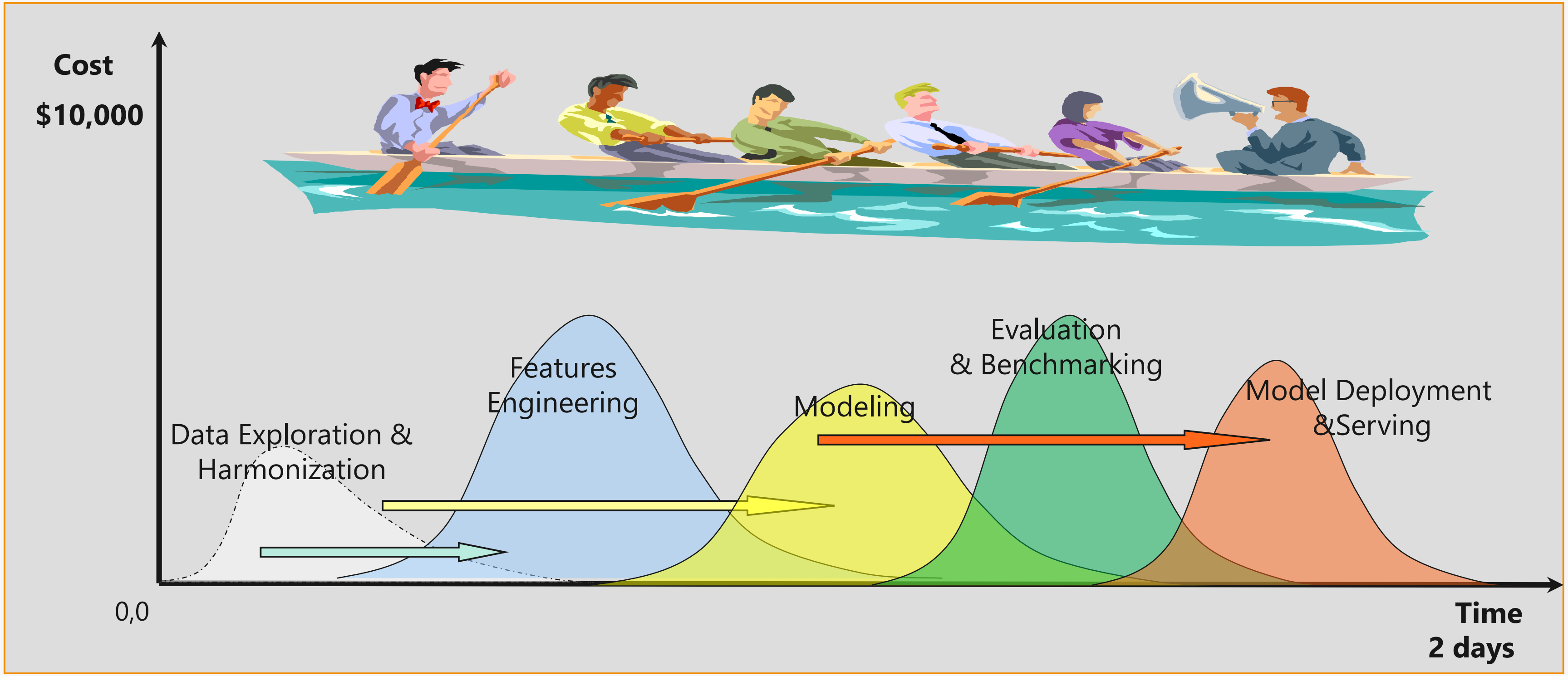
Plus Mode : Fully managed machine learning services

Example : Data Science Workbench



Premium Mode: Automation Services

Example : Amazon SageMaker ?



Challenges to achieve Premium Automation AI Service

Learning Automation

1 **Feature engineering bottlenecks**

Pre-calculate hundreds or thousands Long Term Variables take lots of resources and times

2 **Model scalability limitations**

Trade-off between automation in parallel and scaling machine learning to ever larger datasets and ever more complicated models

3 **Heavily relies on human machine learning experts**

Relies on human to perform the most of tasks

Serving Automation

4 **Less integration with end to end data pipelines, fill in the loop**

Gap to bring machine learning process into the existing enterprise data pipelines , including batch , streaming and real-time

5 **Model Serving to multiple contexts**

Gap to connect to existing business pipelines , offline ,streaming and real-time

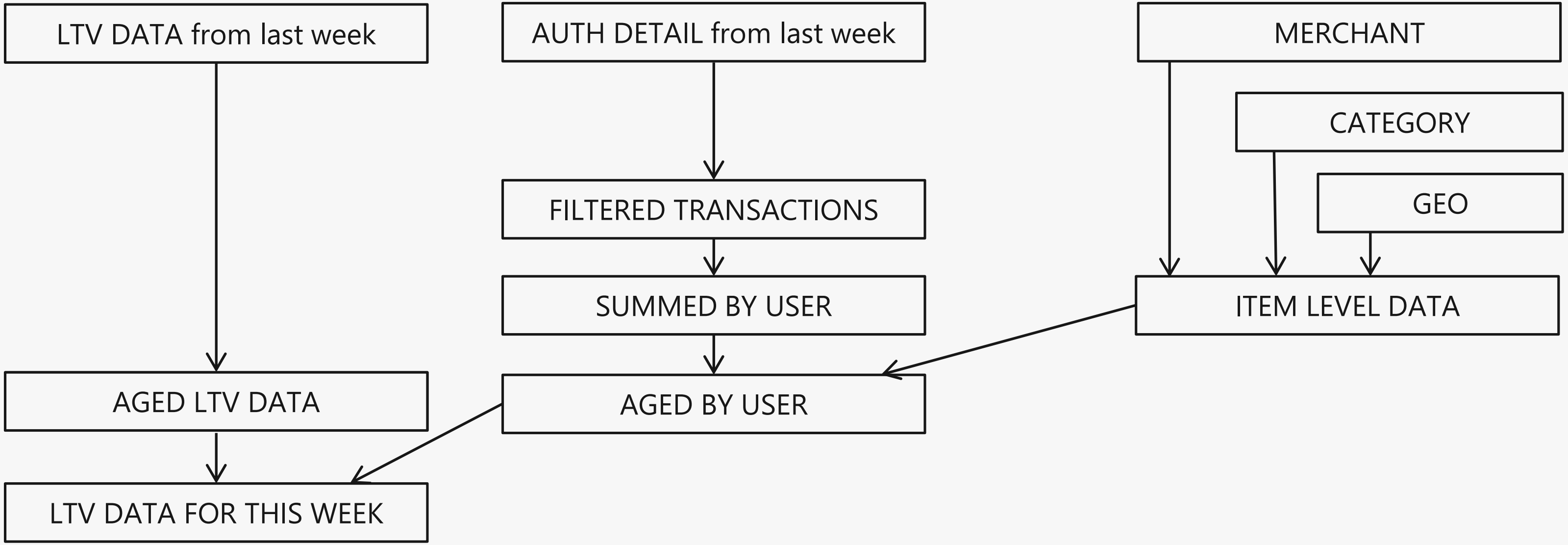
6 **API Enablement and automate deployment**

Low productivity to create more models with low level raw APIs

Isolated promotions and operation readness with automate deployment

What Deep Learning can help ?

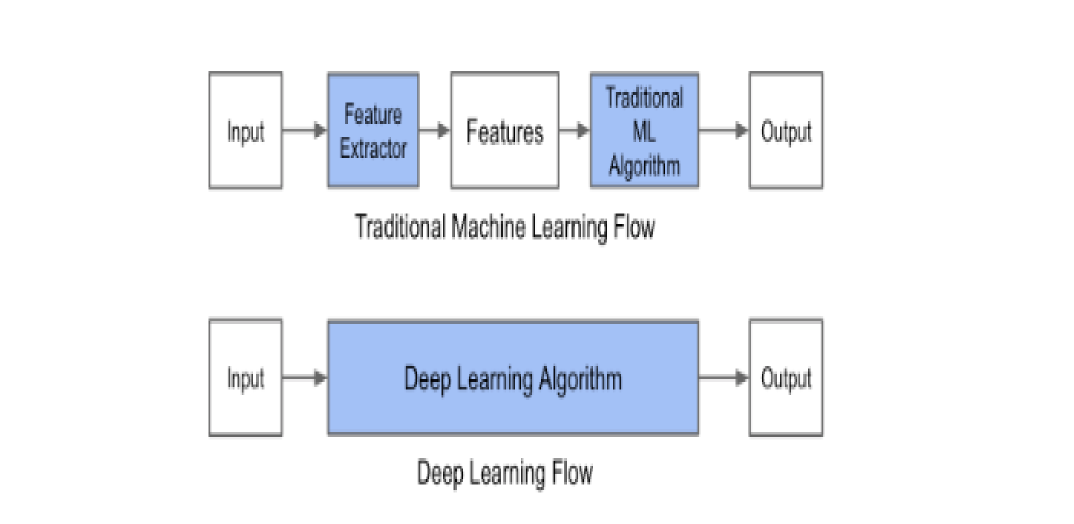
Challenges with Traditional ML : Feature engineering bottlenecks



Bottlenecks

- Need to pre-calculate hundreds or thousands Long Term Variables for each user, such as total spends /visits for merchants list, category list divided by week, months and years
- The computation time for LTV features took > 70% of the data processing time for the whole lifecycle and occupied lots of resources which had huge impact to other critical workloads.
- Miss the feature selection optimizations which could save the data engineering efforts a lot

With Deep Learning : Remove lots of LTV workloads and simply the feature engineering



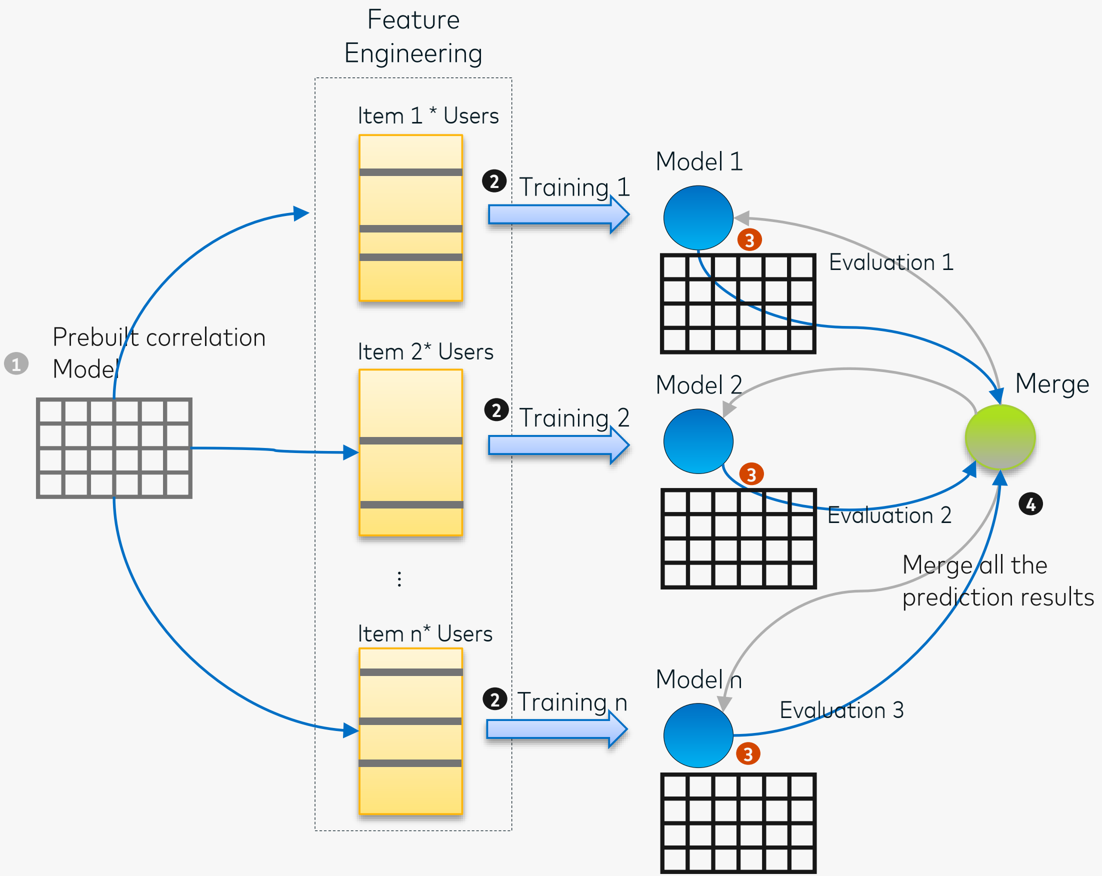
Improvements

- When build model , only focus on few pre-defined sliding features and custom overlap features (Users only need to identify the columns names from data source)

- Remove most of the LTV pre-calculations works, saved hours time and lots of resources

- Deep learning algorithm generates exponential growth of hidden embedding features ,do the internal features selections and optimization automatically when it does cross validation at training stage

Challenges with Traditional ML : Model scalability



Limitations

- All the pipelines separated by items and generate one model for each item
- Have to pre-calculate the correlation matrix between items
- Lots of redundant duplications and computations at feature engineering ,training and testing process
- Run items in parallel and occupied most of cluster resources when executed
- Bad metrics for items with few transactions
- It is very hard to scale more items , from hundreds to millions ?

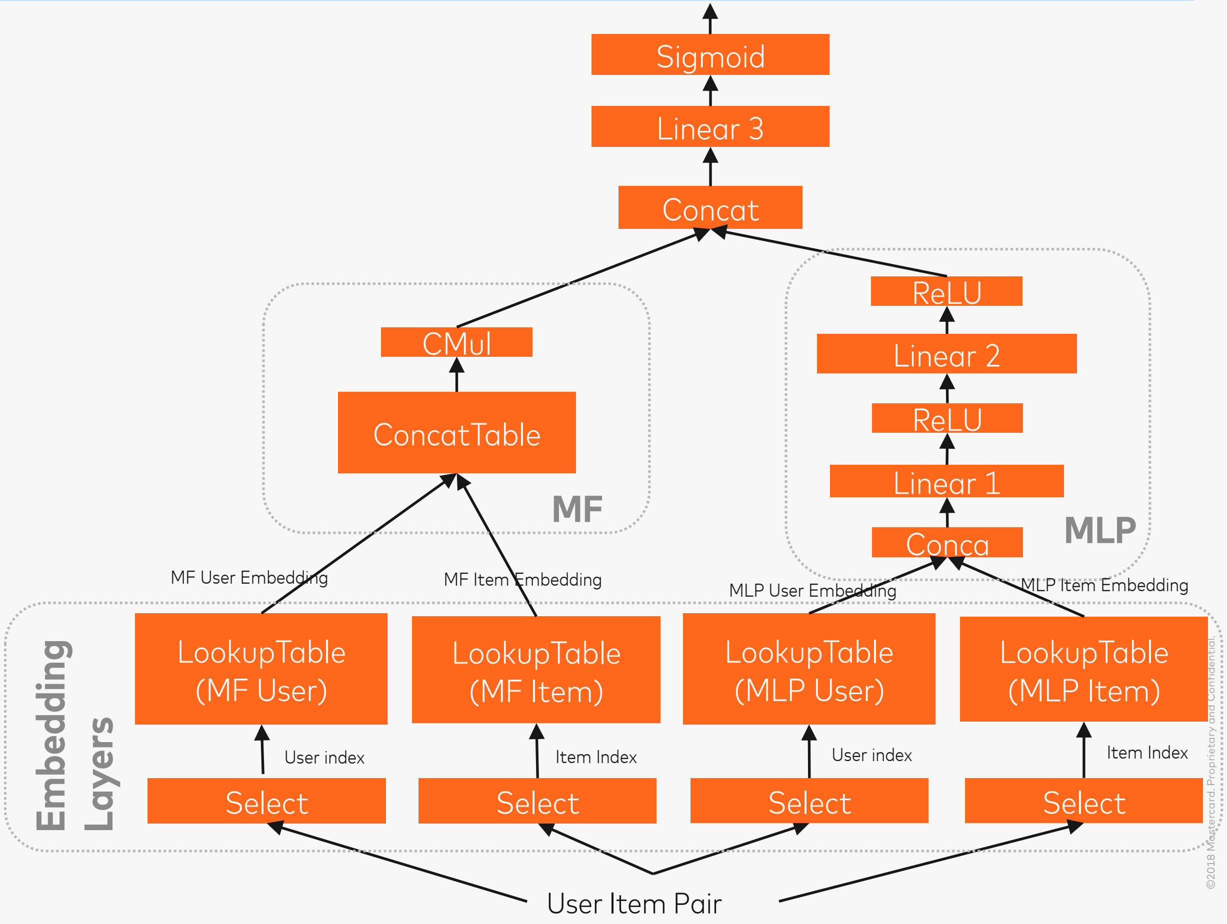
With Deep Learning : Scale models in deeper and wider without decreasing metrics

- **NCF**
- Scenario： Neural Collaborative Filtering ,recommend products to customers (priority is to recommend to active users) according to customers’ past history activities.
- https://www.comp.nus.edu.sg/~xiangnan/papers/ncf.pdf

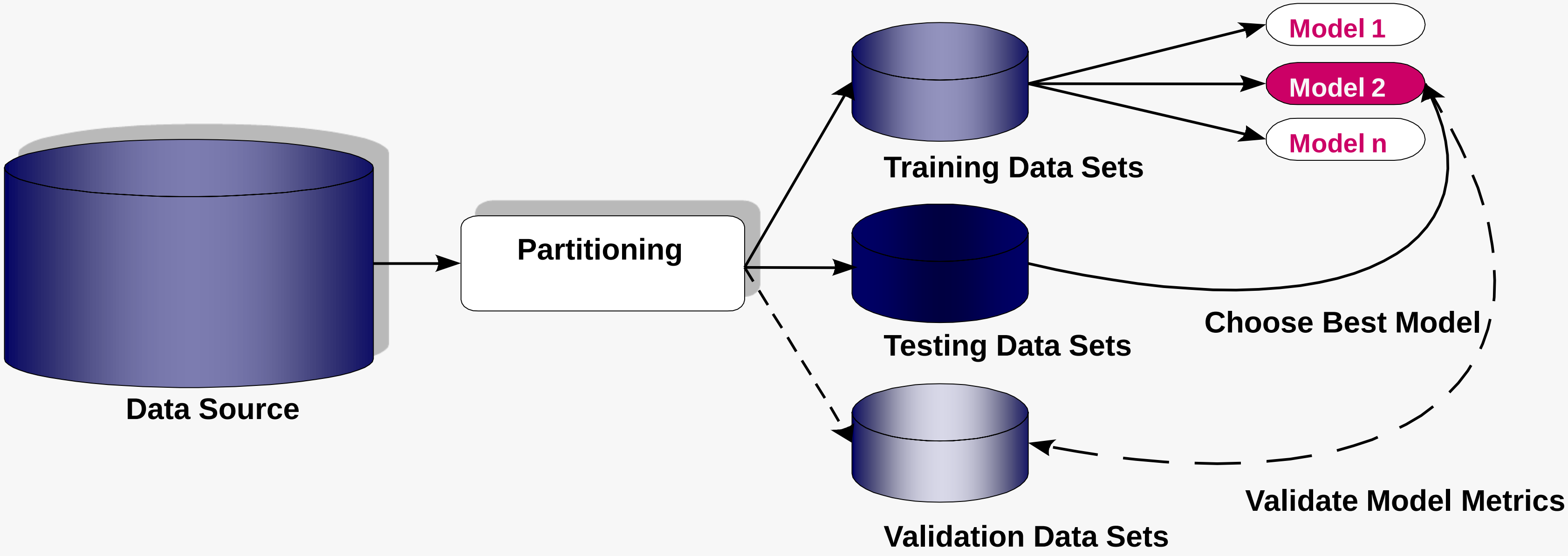
- **Wide & Deep learning**

- Scenario: jointly trained wide linear models and deep neural networks---to combine the benefits of memorization and generalization for recommender systems.

- https://pdfs.semanticscholar.org/aa9d/39e938c84a867ddf2a8cabc575ffba27b721.pdf



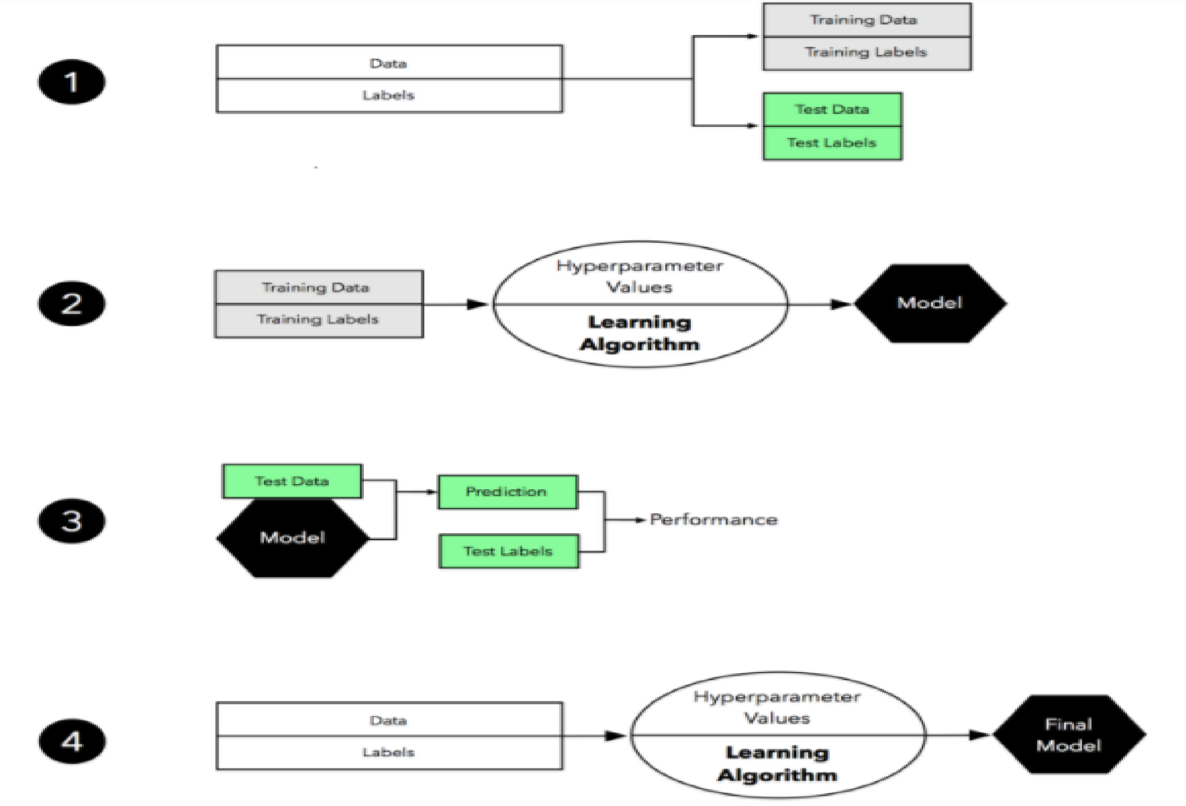
Challenges with Traditional ML : Heavily relies on human machine learning experts



Relies on human to perform the following tasks:

- Select and construct appropriate features.
- Select an appropriate model family.
- Optimize model hyper parameters.
- Post process machine learning models.
- Critically analyze the results obtained.

With Deep Learning : Gives more options for finding an optimally performing robust configuration



Improvements

- Common neural network "tricks", including initialization, L2 and dropout regularization, Batch normalization, gradient checking

- A variety of optimization algorithms, such as mini-batch gradient descent, Momentum, RMSprop and Adam

- Provides optimization-as-a-service using an ensemble of optimization strategies, allowing practitioners to efficiently optimize models faster and cheaper than standard approaches.

Our Explore & Evaluation Journey

Enterprise requirements for Deep Learning

Collocated with mass data storage

- Analyze a large amount of data on the same Big Data clusters where the data are stored (HDFS, HBase, Hive, etc.) rather than ***move or duplicate data***

Seamless integration with Products Internal & External

- Add deep learning capabilities to existing Analytic Applications and/or machine learning workflows rather than ***rebuild all of them***

Data governance with restricted Processing

- Follow data privacy, regulation and compliance (such as PCI/PII compliance and GDPR rather than ***operate data in unsecured zones***

Shared infrastructure with Multi-tenant isolated resources

- Leverage existing Big Data clusters and deep learning workloads should be managed and monitored with other workloads (ETL, data warehouse, traditional ML etc..) rather than ***run DL workloads standalone in separate clusters***

Challenges and limitations to Production considering some “Super Stars” ….

- Claimed that the GPU computing are better than CPU which requires new hardware infrastructure (very long timeline normally)
- Success requires many engineer-hours (Impossible to Install a Tensor Flow Cluster at STAGE ...)
- Low level APIs with steep learning curve (Where is your PHD degree ?)
- Not well integrated with other enterprise tools and need data movements (couldn't leverage the existing ETL, data warehousing and other analytic relevant data pipelines, technologies and tool sets. And it is also a big challenge to make duplicate data pipelines and data copy to the capacity and performance.)
- Tedious and fragile to distribute computations (less monitoring)
- The concerns of Enterprise Maturity and InfoSec (use GPU cluster with Tensor Flow from Google Cloud)

…………..

Maybe not your story , but we have ….

What does Spark offer?

Integrations with existing DL libraries

- Deep Learning Pipelines (from Databricks)
- Caffe (CaffeOnSpark)
- Keras (Elephas)
- mxnet
- Paddle
- TensorFlow (TensorFlow on Spark, TensorFrames)
- CNTK (mmlspark)

Implementations of DL on Spark

- BigDL
- DeepDist
- DeepLearning4J
- SparkCL
- SparkNet

Need more break down …..

	Programming interface	Contributors	commits
BigDL	Scala & Python	50	2221
TensorflowOnSpark	Python	9	257
Databricks/tensor	Python	9	185
Databricks/spark-deep-learning	Python	8	51

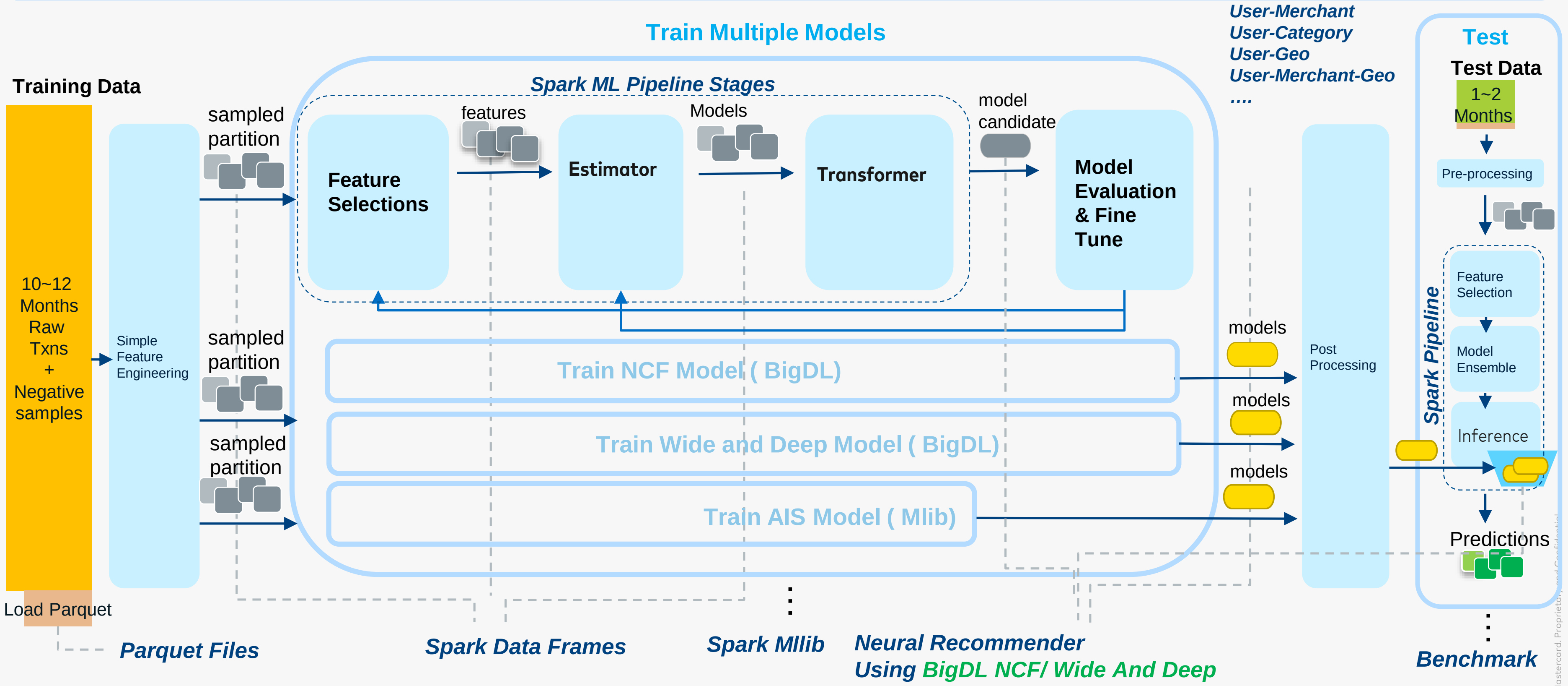
Statistics collected on Mar 5th, 2018

Tensor Flow-on-Spark (or Caffe-on-Spark) uses Spark executors (tasks) to launch Tensor Flow/Caffe instances in the cluster; however, the distributed deep learning (e.g., training, tuning and prediction) are performed outside of Spark (across multiple Tensor Flow or Caffe instances).

(1) As a results, Tensor Flow/Caffe still runs on specialized HW (such as GPU servers interconnected by InfiniBand), and the Open MP implementations in Tensor Flow/Caffe conflicts with the JVM threading in Spark (resulting in lower performance).

(2) In addition, in this case Tensor Flow/Caffe can only interact the rest of the analytics pipelines in a very coarse-grained fashion (running as standalone jobs outside of the pipeline, and using HDFS files as job input and output).

POC: Benchmark BigDL & Spark Mllib



Benchmark results (> 100 rounds)

Mllib AIS

AUROC: A
AUPRCs: B
recall: C
precision: D
20 precision: E

Parameters :
MaxIter(100)
RegParam(0.01)
Rank(200)
Alpha(0.01)

BigDL NCF

AUROC: A+23%
AUPRCs: B+31%
recall: C+18%
precision: D+47%
20 precision: E+51%

Parameters :
MaxEpoch(10)
learningRate(3e-2)
learningRateDecay(3e-7)
uOutput(100)
mOutput(200)
batchSize(1.6 M)

BigDL WAD

AUROC: A+20% (3 % down)
AUPRCs: B+30% (1% down)
recall: C+12% (4 % down)
precision: D+49% (2 % up)
20 precision: E+54% (3% up)

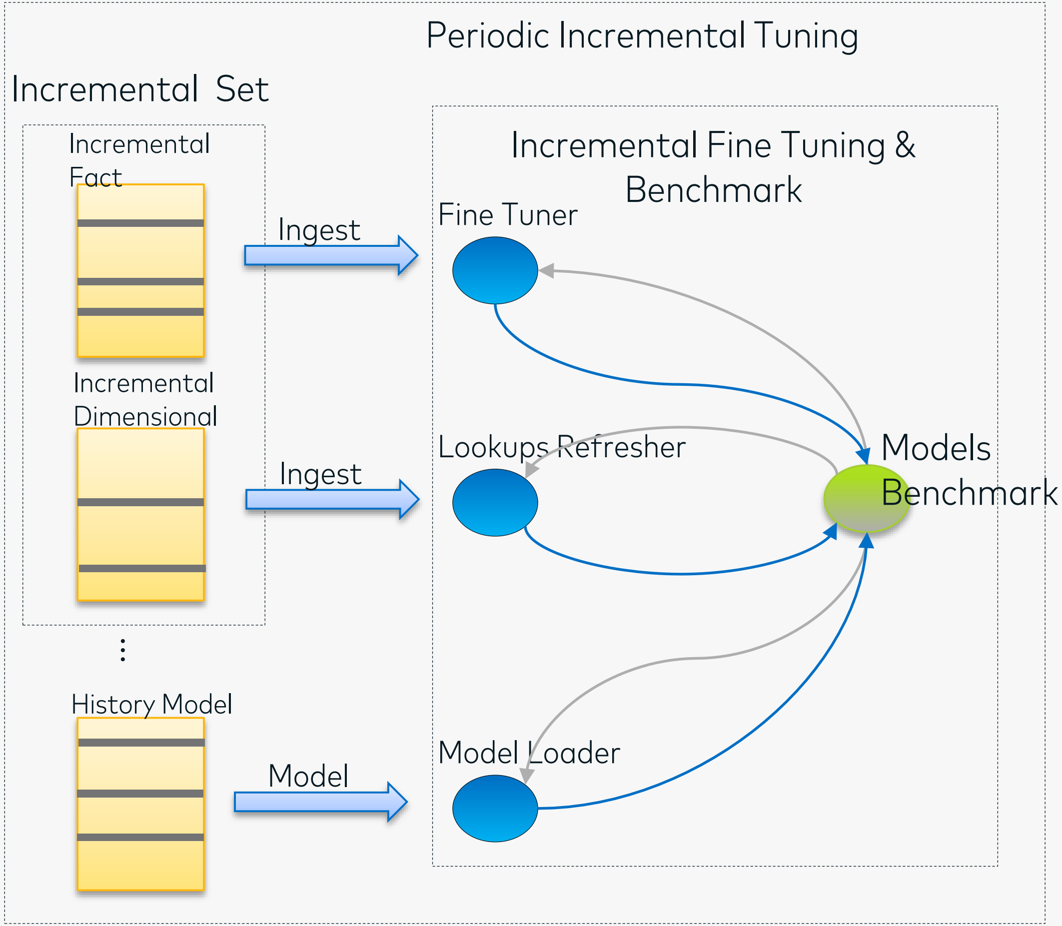
Parameters :
MaxEpoch(10)
learningRate(1e-2)
learningRateDecay(1e-7)
uOutput(100)
mOutput(200)
batchSize(0.6 M)

Beyond Deep Learning library , we need more automated platform capabilities to fit PROD adoption gaps

Gap 1 : Incremental Tuning

Incremental Tuning (only re-run the whole pipeline with incremental changed datasets such as daily changed transactions and benchmark the models)

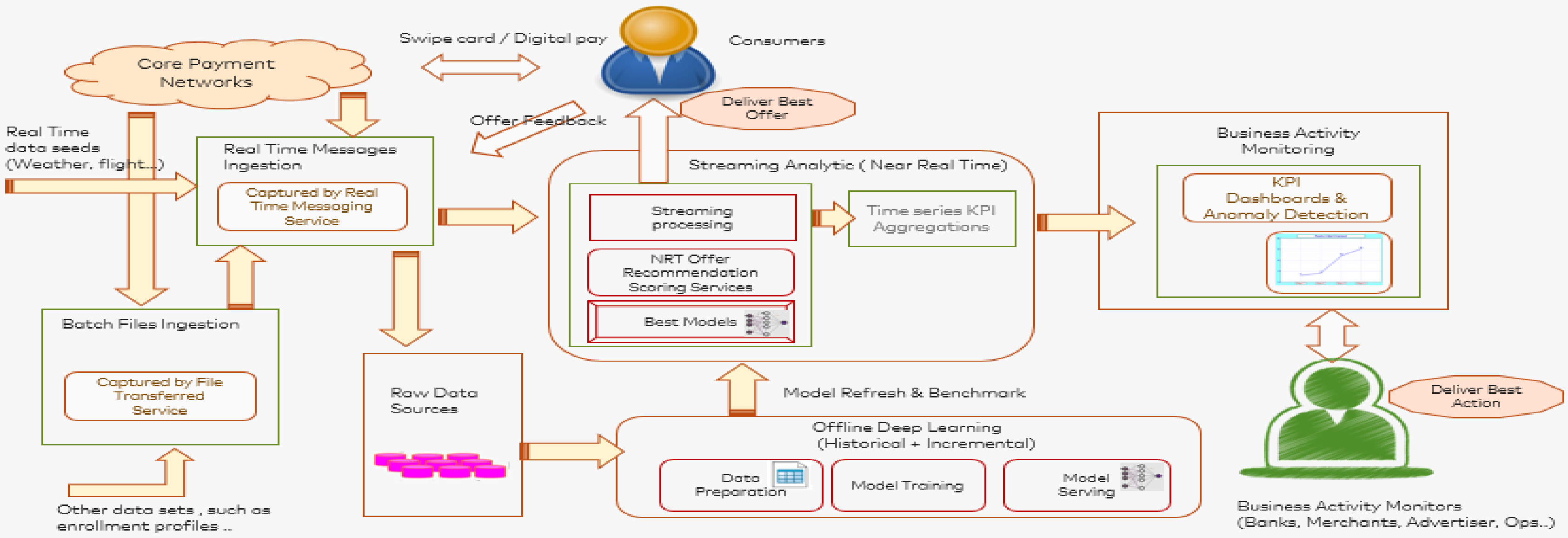
- Refresh the dimensional datasets (such as adding new users , items …)
- Load the history model to the context and update incremental parts of model based on the incremental data sets
- Periodic Re-training with a batch algorithm and time-series prediction
- Benchmark the history model and update model and on-board the better ones.



Gap 2 : Model Serving to multiple contexts

***Model Serving** (Connect to existing business pipelines , offline ,streaming and real-time)*

• Build the model serving capability by exporting model to scoring/prediction/recommendation services and integration points
• Integrate the model serving services inside the business pipelines , such as embed them into Spark jobs for offline, Spark Streaming jobs for streaming , the real-time "dialogue" with Kafka messaging ...



Gap 3 : Build user friendly high level pipeline APIs

High level pipeline APIs

- Abstract and purify high level data and learning pipeline APIs on top of BigDL lib to simply the deep learning model assembly process and increase productivity

```
val mode = param.mode
mode match {
  case ParamUtils.Mode_Data =>
    dataPipeline.genUDFMDF(spark, param, rawDF)
    dataPipeline.genData(spark, param, rawDF)
  case ParamUtils.Mode_Sliding_Data =>
    dataPipeline.genUDFMDF(spark, param, rawDF)
    dataPipeline.genData(spark, param, rawDF)
  case ParamUtils.Mode_DataAndTrain =>
    dataPipeline.genUDFMDF(spark, param, rawDF)
    dataPipeline.genData(spark, param, rawDF)
    learningPipeline.train(spark, param)
  case ParamUtils.Mode_Sliding =>
    dataPipeline.genUDFMDF(spark, param, rawDF)
    dataPipeline.genData(spark, param, rawDF)
    learningPipeline.train(spark, param)
  case ParamUtils.Mode_Train =>
    learningPipeline.train(spark, param)
  case ParamUtils.Mode_Sliding_Train =>
    learningPipeline.train(spark, param)
  case ParamUtils.Mode_Incremental =>
    dataPipeline.incrementalData(spark, param, rawDF)
    learningPipeline.incrementalTrain(spark, param)
}
```

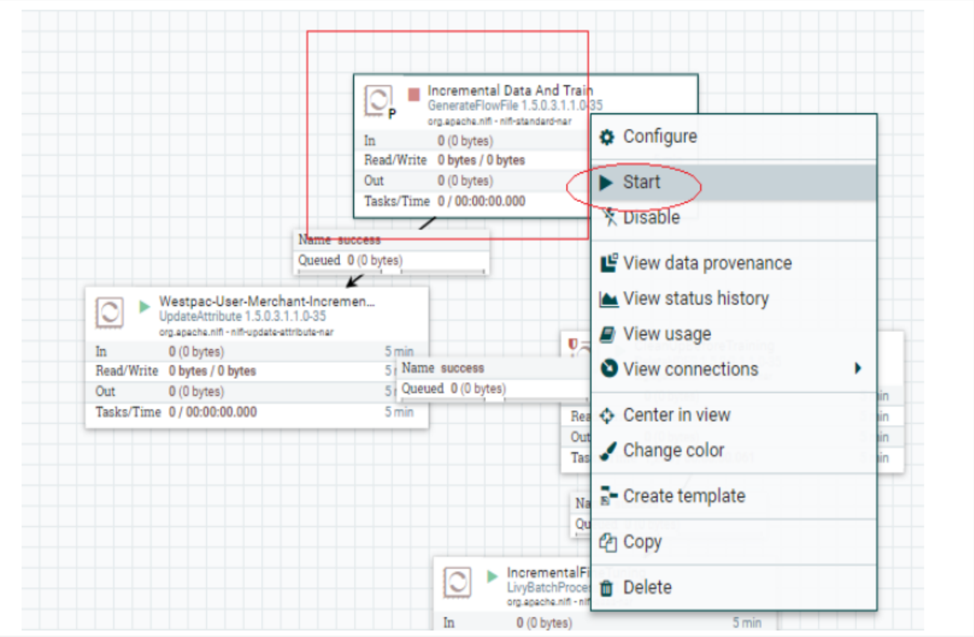
- DataPipeline
  - genUDFMDF(spark: SQLContext, params: AppParams, rawDF: DataFr
  - genData(spark: SQLContext, params: AppParams, rawDF: DataFrame
  - genTrainData(spark: SQLContext, params: AppParams, rawDF: DataF
  - genValidateData(spark: SQLContext, params: AppParams, rawDF: Da
  - genData4Period(spark: SQLContext, params: AppParams, dFwithMo
  - genTAndVDFs(months: Array[Int], allPosIDDF: DataFrame, tDFwithM
  - genSlidingDFs(months: Array[Int], allPosIDDF: DataFrame, tDFwithM
  - genNegativeIDDF(allPosIDDF: DataFrame, uDF: DataFrame, mDF: Da
  - randomNegativeSamples(allPosIDDF: DataFrame, uDF: DataFrame, r
  - InvertNegativeSamples(allPosIDDF: DataFrame, uDF: DataFrame, mD
  - compositeNegativeSamples(allPosIDDF: DataFrame, uDF: DataFram
  - randomAndFilterNegativeSamples(allPosIDDF: DataFrame, uDF: Dat
  - genFixedFeatureDF(IDDF: DataFrame, tDFwithMonth: DataFrame, uD
  - genSlidingFeatureDF(IDDF: DataFrame, tDFwithMonth: DataFrame, u
  - norm(df: DataFrame, params: AppParams): DataFrame
  - incrementalData(spark: SQLContext, param: AppParams, rawDF: Dat

- LearningPipeline
  - spark: SQLContext
  - param: AppParams
  - DeepModel
  - WideModel
  - alsModel
  - kmeanModel
  - ComposeModel
  - incrementalTrain(spark: SQLContext, param: AppParams): Uni
  - fineTune(spark: SQLContext, param: AppParams, trainingDF: D
  - train(spark: SQLContext, param: AppParams): Unit
  - saveModel(spark: SQLContext, composeModel: ComposeMod
  - trainModel(spark: SQLContext, param: AppParams, trainingDF
  - loadModel(param: AppParams): ComposeModel
  - savePredictions(spark: SQLContext, param: AppParams, predic
  - getOptimMethod(optimMethodType: String): OptimMethod[
  - evaluateModel(spark: SQLContext, param: AppParams, uDF: D
  - fineTuningModel(spark: SQLContext, trainingDF: DataFrame,

Gap 4 : Integrated with end to end data pipelines, fill in the loop

Embedded the deep learning process into existing enterprise data pipelines

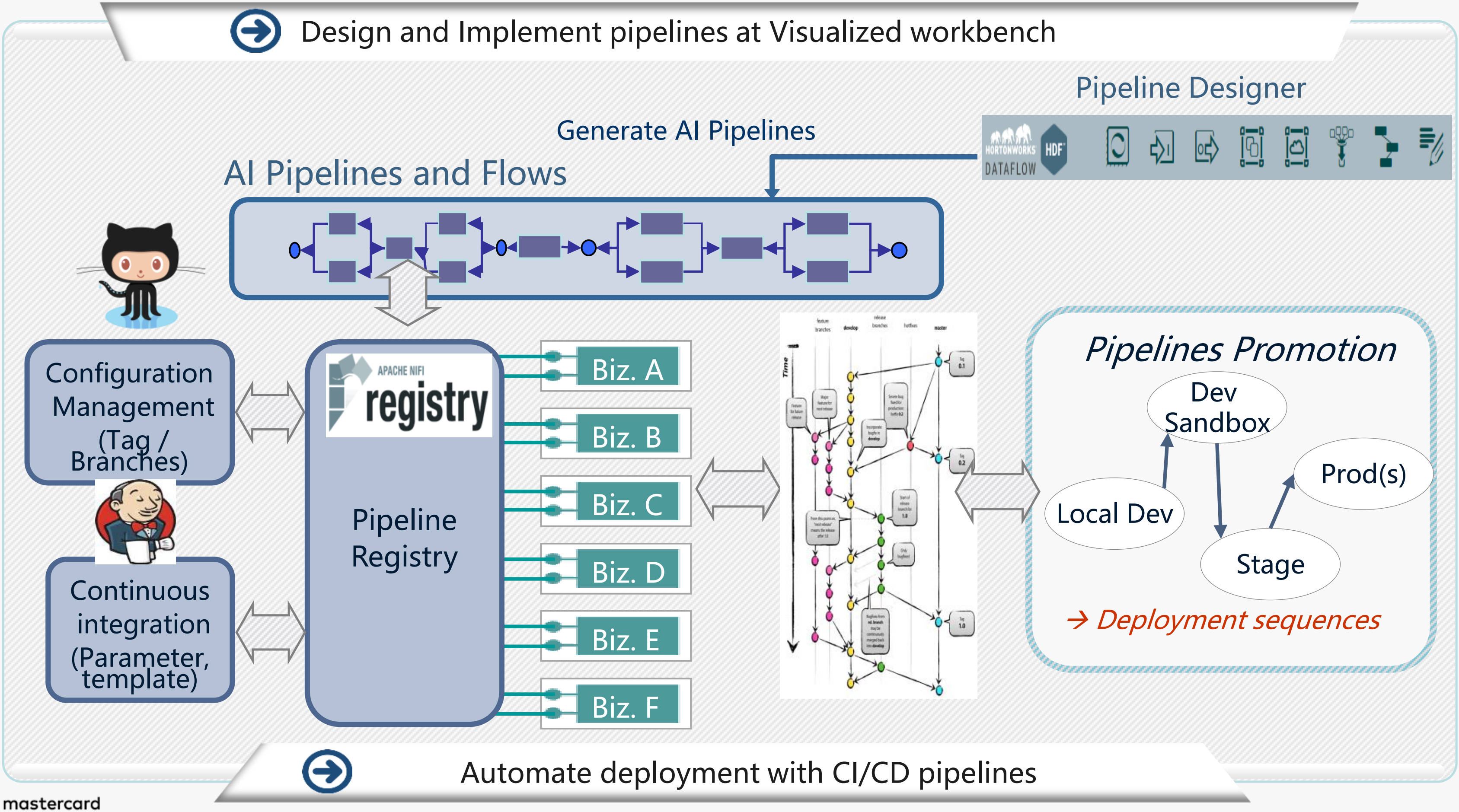
• Build pre-defined templates and customized processors to bring deep learning process into the existing enterprise data pipelines , including batch , streaming and real-time



Property	Value
batchSize	180000
dataPrepareSql	select external_userid as u,cast(amount*(-1) as double...
dataSamplingParams	random,10,randomAndFilter,20,0.8,true
debug	false
defaultPartition	100
fi	westpac
learningRate	1e-3
learningRateDecay	1e-7
maxEpoch	5
overlapFeatureParams	f1,f2,f3,f4
recommendationMode	mix
slidingHistoryLengh	3
target	user-merchant-deep-livy

mastercard

Gap 5 : AI Pipelines promotion with automated CI/CD deployment



Community improvements : Analytics Zoo -> Unified Analytics + AI Platform for Spark and BigDL

Easier to build end-to-end analytics + AI applications

- Reference use cases
  - Anomaly detection, sentiment analysis, fraud detection, chatbot, sequence prediction, etc.

- Predefined models
  - Object detection, image classification, text classification, recommendations, GAN, etc.

- Feature engineering & transformations
  - Image, text, speech, 3D imaging, time-series, etc.

- High level pipeline APIs
  - Dataframes, ML Pipelines, autograd, transfer learning, Keras/Keras2, etc.

https://github.com/intel-analytics/analytics-zoo

Thanks
Q & A