“Analytics using Apache Spark”

(Lightening Fast Cluster Computing)

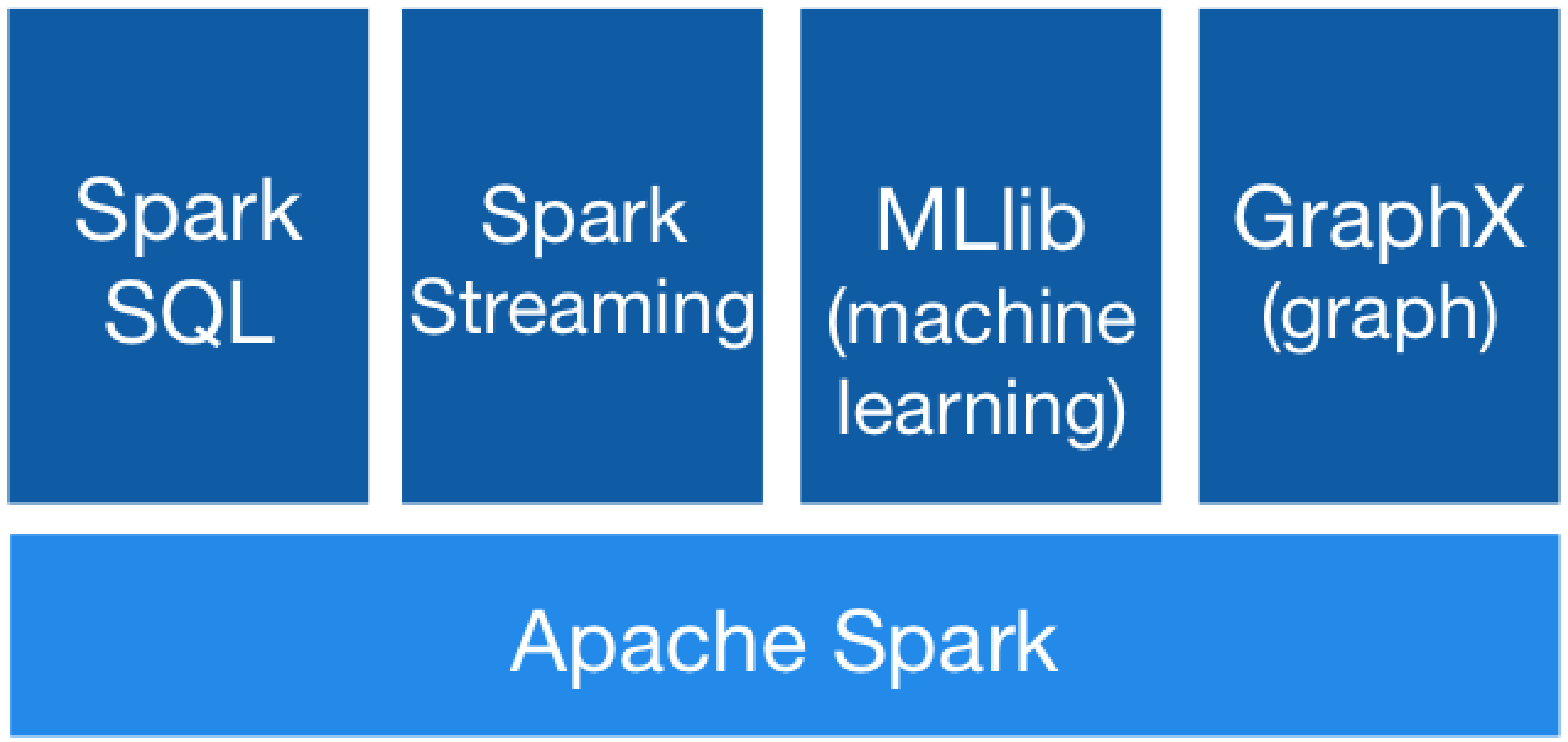
Sukeshini
Big Data Analytics & Machine Learning Team
C-DAC Bengaluru
sukeshini@cdac.in

Spark MLlib

Library of ML algorithms & utilities designed to run in parallel on Spark clusters.

Introduction

- MLlib is **Scalable Machine Learning Library**
- Initial contribution from AMPLab, UC Berkeley
- ***spark.mllib (RDD Based)***
- ***spark.ml – New API (DF based)***
- High-quality algorithms
- 100x faster than MapReduce



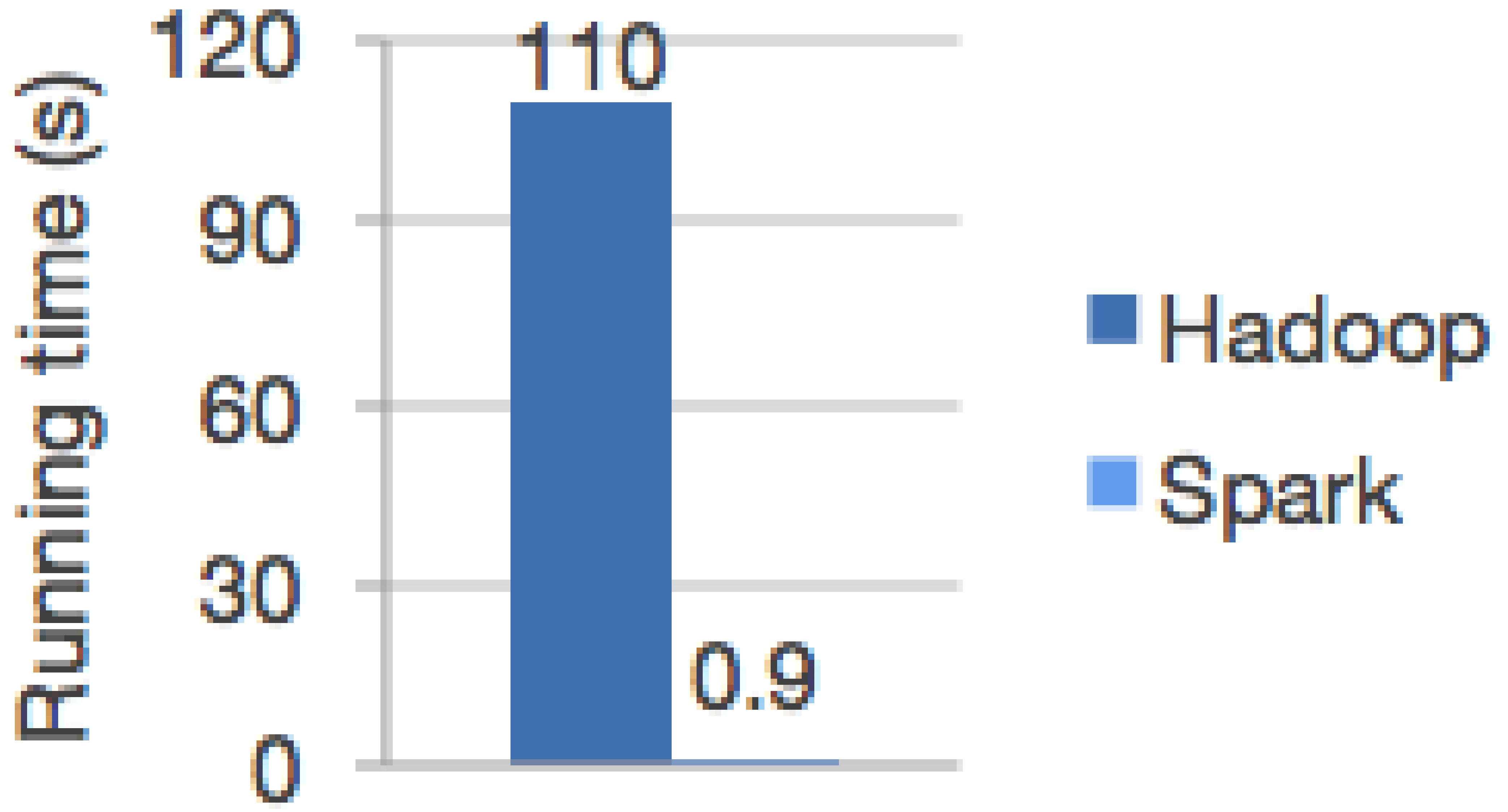


Figure: Logistic regression on Hadoop and Spark

Announcement: DataFrame-based API is primary API

- As of Spark 2.0, the RDD-based APIs in the spark.mllib package have entered maintenance mode.
- Primary ML API for Spark is now the DataFrame-based API in the spark.ml package.
- **Implications?**
    - MLlib will still support the RDD-based API in spark.mllib with bug fixes.
    - MLlib will not add new features to the RDD-based API.
    - In the Spark 2.x releases, MLlib will add features to the DataFrames-based API to reach feature parity with the RDD-based API.
    - After reaching feature parity (roughly estimated for Spark 2.3), the RDD-based API will be deprecated.
    - The RDD-based API is expected to be removed in Spark 3.0.

Why is MLlib switching to the DataFrame-based API?

- DataFrames provide a **more user-friendly** API than RDDs.
- Benefits of DataFrames include Spark Data Sources, SQL/DataFrame queries, **Tungsten and Catalyst optimizations** and uniform APIs across languages.
- The DataFrame-based API for MLlib **provides a uniform API across ML algorithms** and across multiple languages.
- DataFrames facilitate practical **ML Pipelines**, particularly feature transformations.

spark.mllib Features

- Utilities: Linear Algebra, Statistics, etc.
- Features Extraction, Features Transforming, etc.
- Regression
- Classification
- Clustering
- Collaborative Filtering, e.g. Alternating Least Squares
- Dimensionality reduction
- And many more…

http://spark.apache.org/docs/latest/mllib-guide.html

spark.ml Features

”All” spark.mllib features plus

- Pipelines
- Persistence

Model selection and tuning

- Train validation split
- K-folds cross validation

spark.ml Features

1. **Statistics:** Correlation, Hypothesis Testing
2. **Featurization:** Feature Extraction, Transformation, Dimensionality Reduction and Selection
3. **ML Algorithms:** Common Learning Algorithms Such As Classification, Regression, Clustering and Collaborative Filtering
4. **Pipelines:** Tools For Constructing, Evaluating and Tuning ML Pipelines
5. **Persistence:** Saving And Load Algorithms, Models and Pipelines
6. **Utilities:** Linear Algebra, Statistics, Data Handling, Etc.

Statistics

Correlation

- Calculating the correlation between two series of data is a common operation in Statistics.
- MLLib Supports **Pearson's** and **Spearman's** correlation.

`Correlation` computes the correlation matrix for the input Dataset of Vectors using the specified method. The output will be a DataFrame that contains the correlation matrix of the column of vectors.

```
from pyspark.ml.linalg import Vectors
from pyspark.ml.stat import Correlation

data = [(Vectors.sparse(4, [(0, 1.0), (3, -2.0)]),),
        (Vectors.dense([4.0, 5.0, 0.0, 3.0]),),
        (Vectors.dense([6.0, 7.0, 0.0, 8.0]),),
        (Vectors.sparse(4, [(0, 9.0), (3, 1.0)]),)]
df = spark.createDataFrame(data, ["features"])

r1 = Correlation.corr(df, "features").head()
print("Pearson correlation matrix:\n" + str(r1[0]))

r2 = Correlation.corr(df, "features", "spearman").head()
print("Spearman correlation matrix:\n" + str(r2[0]))
```

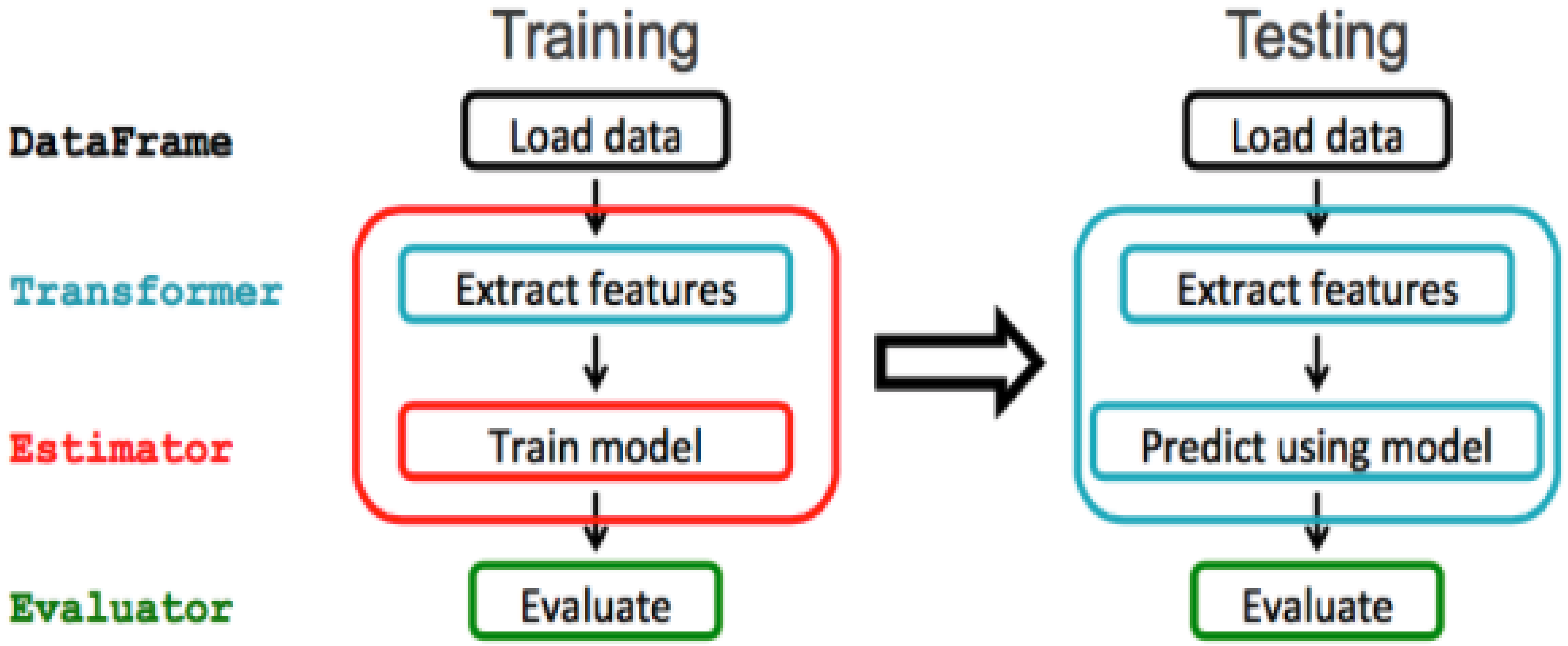
Hypothesis Testing

- **Hypothesis testing** is a powerful tool in statistics **to determine whether a result is statistically significant,** whether this result occurred by chance or not.
- spark.ml currently supports Pearson's Chi-squared ($\chi 2$) tests for independence.
- ChiSquareTest conducts Pearson's independence test for every feature against the label.
- For each feature, the (feature, label) pairs are converted into a contingency matrix for which the Chi-squared statistic is computed.
- All label and feature values must be categorical.

ML Pipeline

Pipeline

- MLlib standardizes APIs for ML algorithms to make it easier to combine multiple algorithms into a single pipeline or workflow.
- Mostly inspired by the scikit-learn project.



Pipelines

A pipeline chains multiple Transformers and Estimators together to specify an ML Workflow.

Components

a) **DataFrame**: This ML API uses DataFrame from Spark SQL.

b) **Transformer**: is an algorithm which can transform one DataFrame into another DataFrame.

   Ex: an ML model is a Transformer which transforms a DataFrame with features into a DataFrame with predictions.

c) **Estimator**: An Estimator is an algorithm which can be fit on a DataFrame to produce a Transformer.

   Ex: a learning algorithm is an Estimator which trains on a DataFrame and produces a model.

d) **Parameter**: All Transformers and Estimators now share a common API for specifying parameters.

a) DataFrames

- Machine Learning can be applied to a wide variety of data types, such as vectors, text, images and structured data. This API adopts the DataFrame from Spark SQL in order to support a variety of data types.
- DataFrame supports many basic and structured types; In addition to the types listed in the Spark SQL, DataFrame can use ML Vector types.
- A DataFrame can be created either implicitly or explicitly from a regular RDD.
- Columns in a DataFrame are named.

b) Transformers

- **A Transformer** is an abstraction that includes feature transformers and learned models.
- Transformer implements a method **transform()**, which **converts one DataFrame into another**, generally by appending one or more columns. Ex:
  - A Feature Transformer might take a DataFrame, read a column (e.g., text), map it into a new column (e.g., feature vectors), and output a new DataFrame with the mapped column appended.
  - A Learning Model might take a DataFrame, read the column containing feature vectors, predict the label for each feature vector and output a new DataFrame with predicted labels appended as a column.

c) Estimators

- An Estimator abstracts the concept of a Learning Algorithm or any algorithm that fits or trains on data.
- Technically, an **Estimator** implements a method **fit()**, which **accepts a DataFrame and produces a Model,** which is a Transformer.
- Ex: A Learning Algorithm such as LogisticRegression is an Estimator and calling fit() trains a LogisticRegressionModel.

d) Parameters

- MLlib Estimators and Transformers use a uniform API for specifying parameters.
- There are two main ways to pass parameters to an algorithm:
  - If lr is an instance of LogisticRegression, one could call lr.setMaxIter(10) to make lr.fit() use at most 10 iterations.
  - Pass a ParamMap to fit() or transform(). Any parameters in the ParamMap will override parameters previously specified via setter methods.
- Ex: If we have two LogisticRegression instances lr1 and lr2, then we can build a ParamMap with both maxIter parameters specified:

  ParamMap(lr1.maxIter -> 10, lr2.maxIter -> 20).

  This is useful if there are two algorithms with the maxIter parameter in a Pipeline.

Featurization

Extracting, Transforming and Selecting Features

a) **Extraction:** Extracting features from “raw” data

b) **Transformation:** Scaling, converting or modifying features

c) **Selection:** Selecting a subset from a larger set of features

Feature Extractors

Extracting features from “raw” data

TF-IDF

- Provide an indication of how important a word is taking into account how often the word appears in the entire dataset.
- Term Frequency **TF(t,d)** is the number of times that the term **t** appears in the document **d**.
- Inverse Document Frequency **IDF(t,D)** is a numerical measure of how important a term is by taking into account how often the term appears across the corpus.

$$IDF(t,D) = \log \frac{|D| + 1}{DF(t,D) + 1}$$

$$TFIDF(t,d,D) = TF(t,d) \cdot IDF(t,D)$$

Feature Transformers

- Tokenizer
- StopWordsRemover
- PCA
- OneHotEncoder
- MinMaxScaler
- ElementwiseProduct
- Imputer

ML Algorithms

Algorithms

a) Classification

b) Regression

c) Clustering

d) Collaborative Filtering

e) Frequent Pattern Mining

Classification

a) Logistic Regression
   - Binomial Logistic Regression
   - Multinomial Logistic Regression

b) Decision Tree Classifier

c) Random Forest Classifier

d) Gradient-boosted Tree Classifier

e) Multilayer Perceptron Classifier

f) Linear Support Vector Machine

g) One-vs-Rest Classifier (a.k.a. One-vs-All)

h) Naive Bayes

Regression

a) Linear regression

b) Decision tree regression

c) Random forest regression

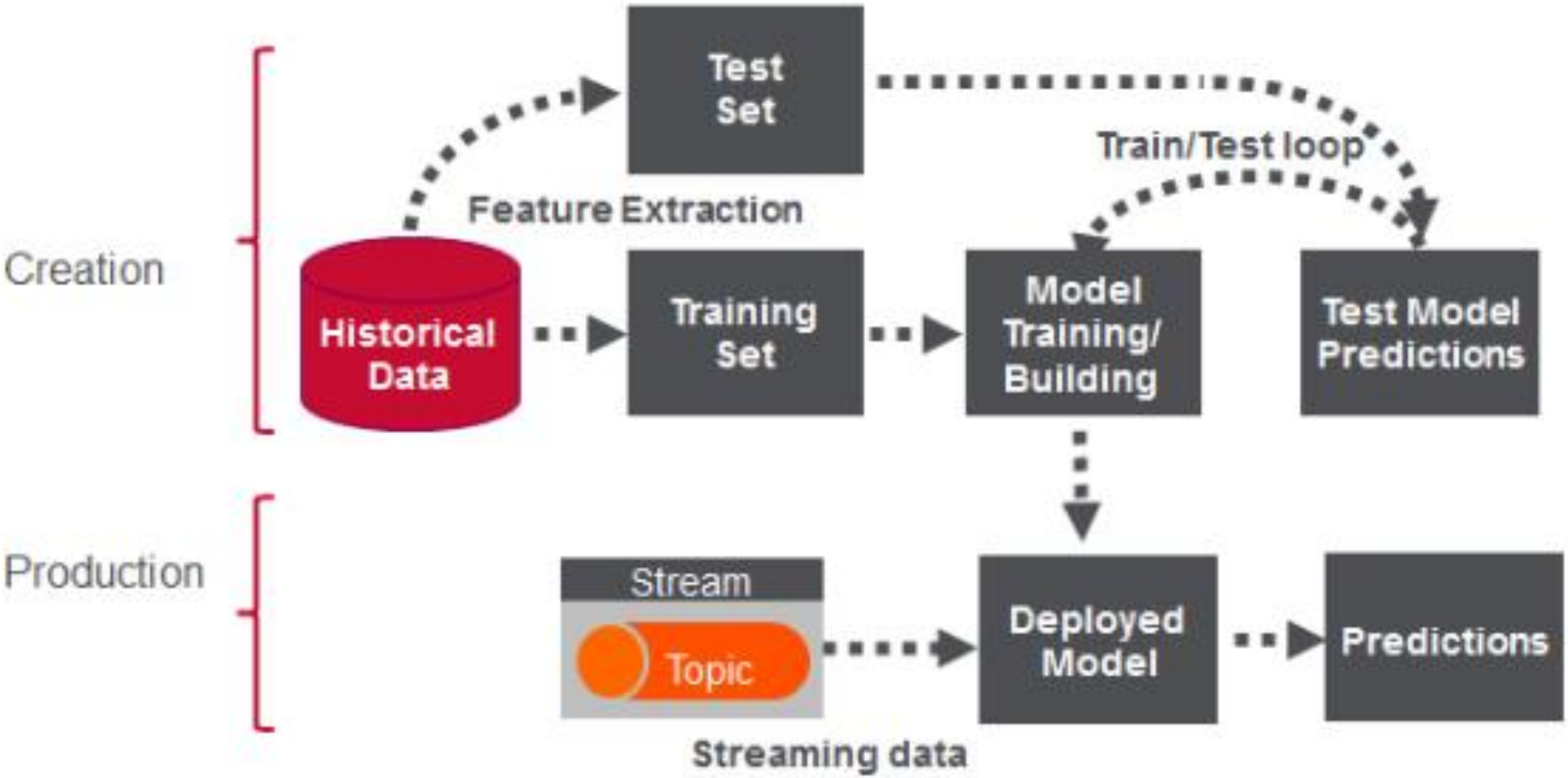
d) Gradient-boosted tree regression

e) Survival regression (For survival analysis)

ML Application

Credit Card Fraud Detection

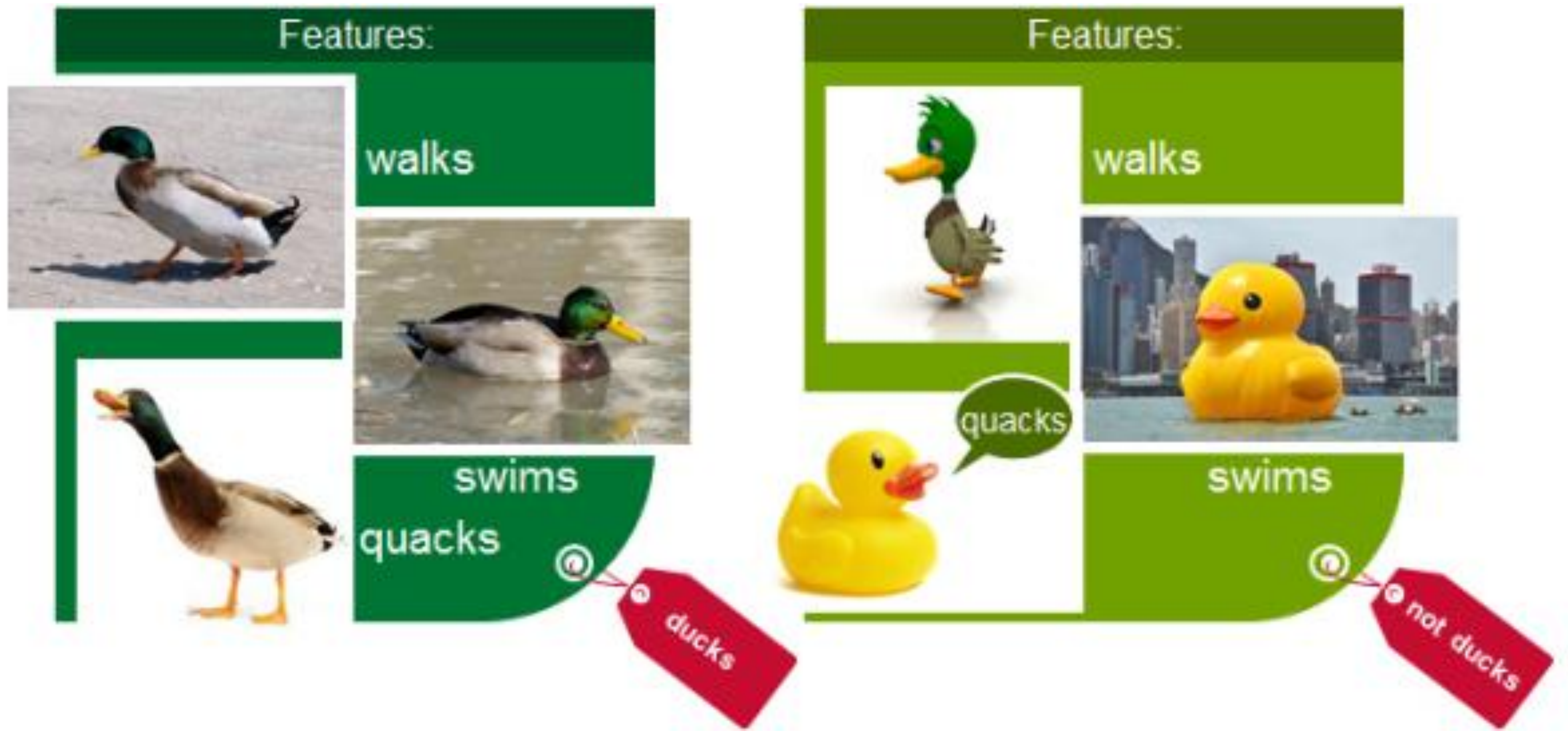
Fraud Detection Phases



How to build the model ?

Use Classification – Identify to which category an item belongs – Fraud or Not-Fraud

- Takes known data with labels
- Features – Can be treated as answers to certain questions

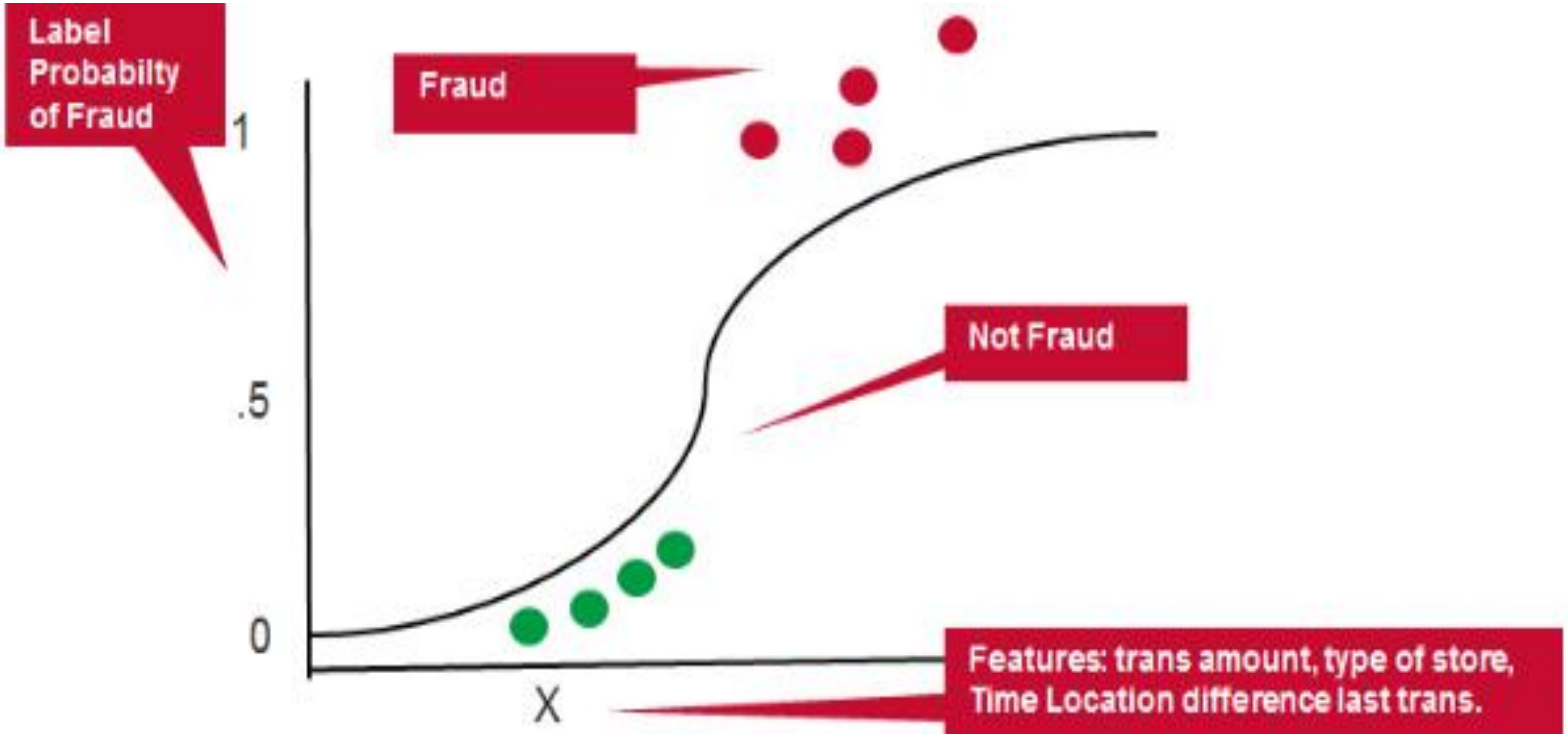


If it Walks, Swims and Quacks like a duck, then the label is "Duck".

How to build the model ?

For credit card transactions…

- **Example Features:** Transaction Amount, Type of Merchant, Distance from and time since last transaction etc.
- **Example Label**: Probability of Fraud



How to build the model ?

For credit card transactions….

Logistic Regression measures the relationship between the Y "Label" and the X "Features" by estimating probabilities using a logistic function.

The model **predicts the Probability of Fraud**, which is used to predict the label class.

How to get the features?

For credit card transactions…

Feature Engineering is the process of **transforming raw data into inputs** for a ML algorithm.

- **Goal:** To ensure if someone using the card other than the cardholder
- **Strategy:** To design features <u>measuring the differences between recent and historical activities.</u>

Features Extraction/ Collection

Credict Card Transaction Features
Card Type
Expiration Date
Home address

Features associated with the Card Holder

Credict Card Transaction Features
POS number
Account number
Date and Time
Transaction amount
Merchant category code

Features associated with the Transaction

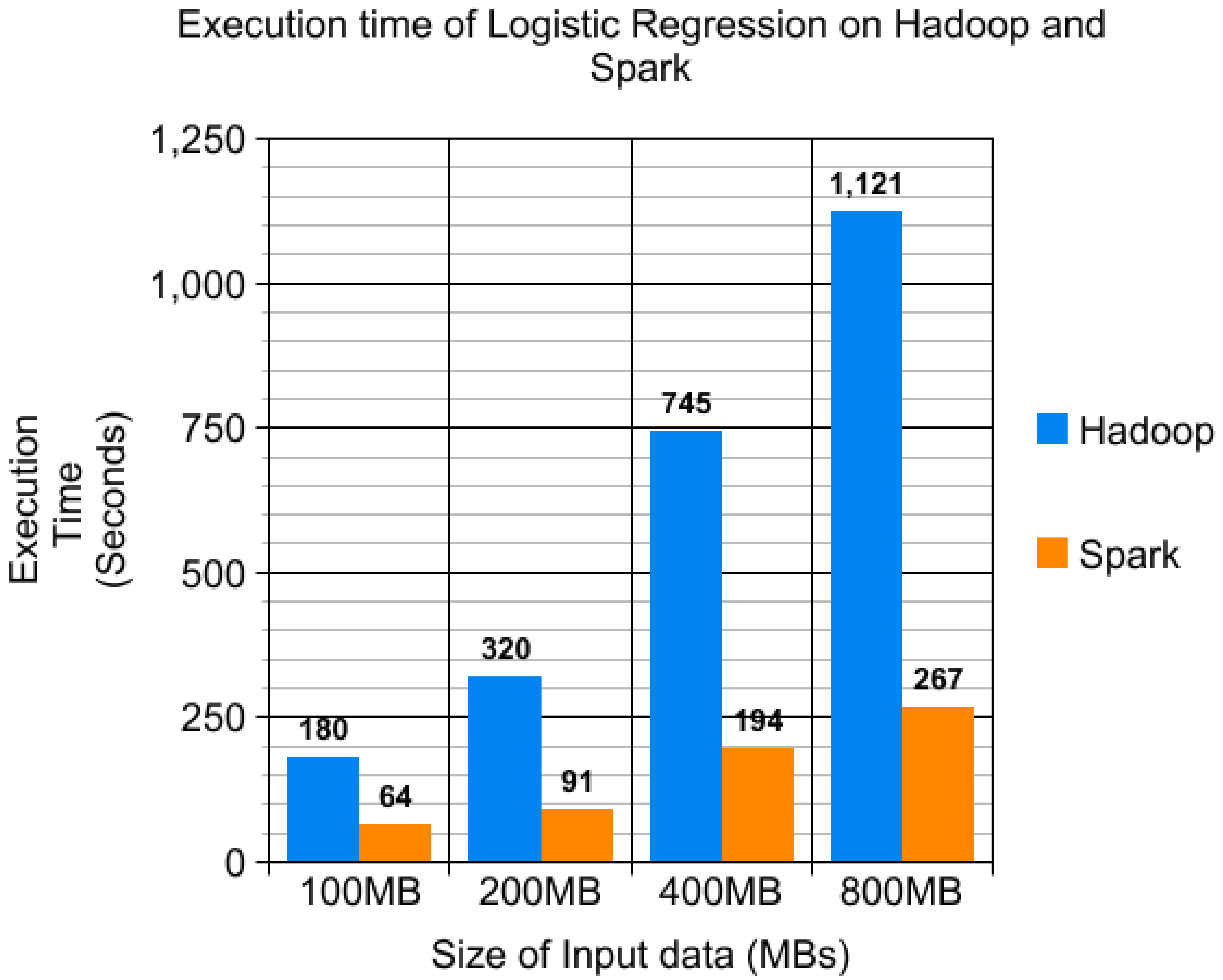
Credict Card Transaction Features
Number of Transactions last 24 hours
Total $ Amount last 24 hours
Average Amount last 24 hours
Average Amount last 24 hours compared to historical use
Location and Time difference since Last Transaction
Average transaction
fraud risk of merchant type
Merchant types for day compared to historical use

Features derived From Transaction History

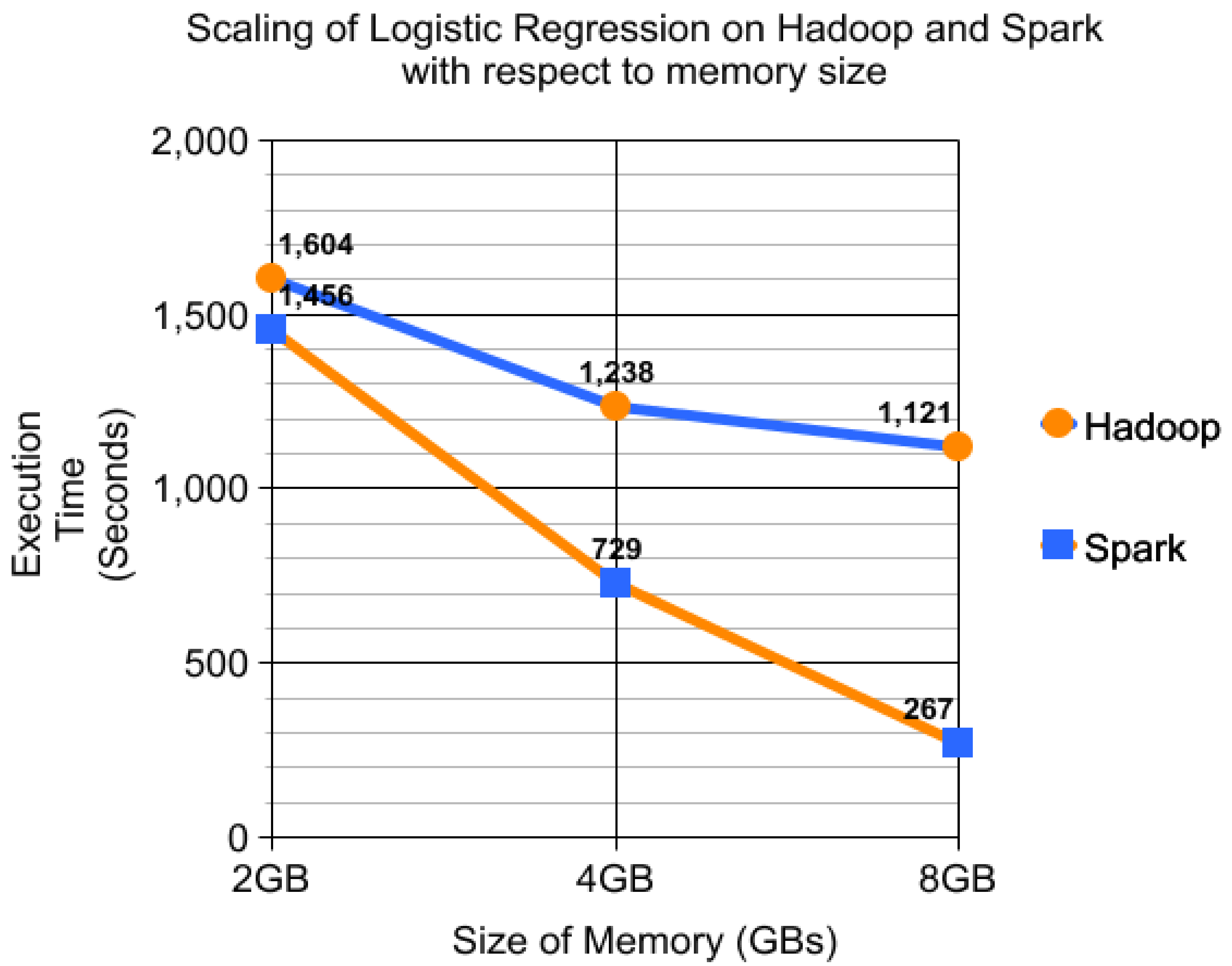
Real Time Fraud Detection using Spark

Hadoop (or) Spark

Hadoop .vs. Spark



Hadoop vs. Spark (w.r.t to RAM size)



Thank You