SWE404/DMT413
BIG DATA ANALYTICS

Lecture 7: Machine Learning and MLlib

Lecturer: Dr. Yang Lu

Email: luyang@xmu.edu.my

Office: A1-432

Office hour: 2pm-4pm Mon & Thur

Outline

- Introduction of Machine Learning
- Spark MLlib

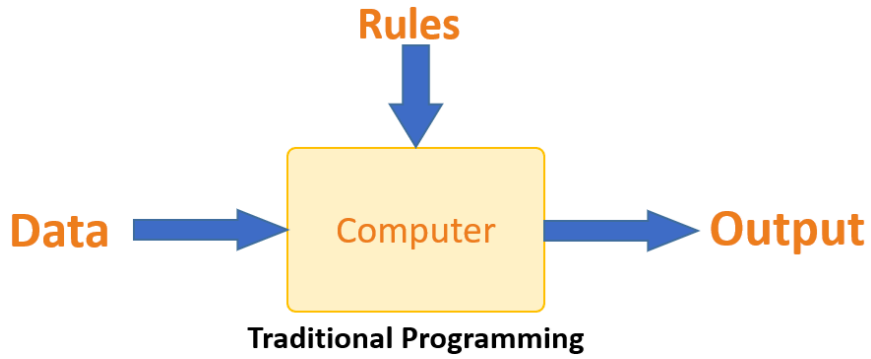
INTRODUCTION OF MACHINE LEARNING

What is Machine Learning?

- Machine Learning is a system that can learn from example to produce accurate results through self-improvement and without being explicitly coded by programmer.
- The goal of machine learning: *do prediction by learning from data*.
  - The prediction output is then used to makes actionable insights.

Machine Learning vs. Traditional Programming

- In traditional programming, a programmer code all the rules in consultation with an expert in the area.
- Each rule is based on a logical foundation. The machine will execute an output following the logical statement.
- Disadvantages:
  - When the system grows complex, more rules need to be written. It can quickly become unsustainable to maintain.
  - The expert's knowledge may not be comprehensive enough to provide accurate rules.



Traditional Programming

Image source: https://www.guru99.com/machine-learning-tutorial.html

Example of Traditional Programming

- Take recommandation as an example.
- We want to predict if a customer is willing to buy a lipstick. We build the following rules:
  - Check if the customer is female.
    - Yes, check if the customer's age is above 18.
      - Yes, return true.
      - No, return false.
    - No, return false.
  - (Female, 20) -> True / (Male, 25) -> False.

Machine Learning vs. Traditional Programming

- The machine learns how the input and output data are correlated and it writes a rule.
- The programmers write an algorithm to generate rules rather than write rules.
- The algorithms may learn implicit rules that experts are not able to know.

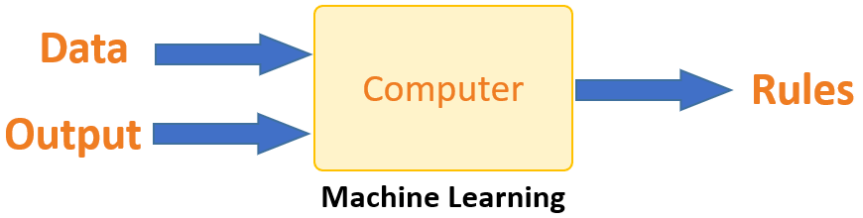


Image source: https://www.guru99.com/machine-learning-tutorial.html

Example of Machine Learning

(Female, 20, True)

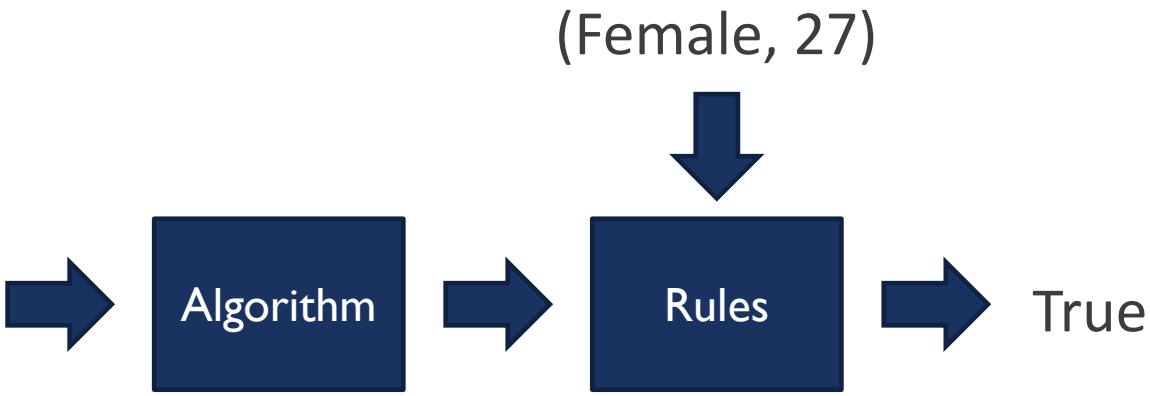
(Male, 20, False)

(Female, 15, True)

(Female, 50, False)

(Male, 25, True)

(Male, 18, False)



Machine Learning Terminology

- The list of attributes used to solve a problem is called a *feature vector*.
  - E.g. gender and age.
- The target we want to predict is called *label*.
  - E.g. Willing to buy or not (True / False).
- The rules used to predict is called *model*.
- The program that is used to generate the model is called *algorithm*.

Machine Learning Terminology

- The process to generate a model a called *training* or *learning*, and the set of data used in this process is called *training data*.
- The process to use a trained model to predict is called *testing* or *inference*, and the data used in this process is called *test data*.
- A single data instance is called a *sample*.

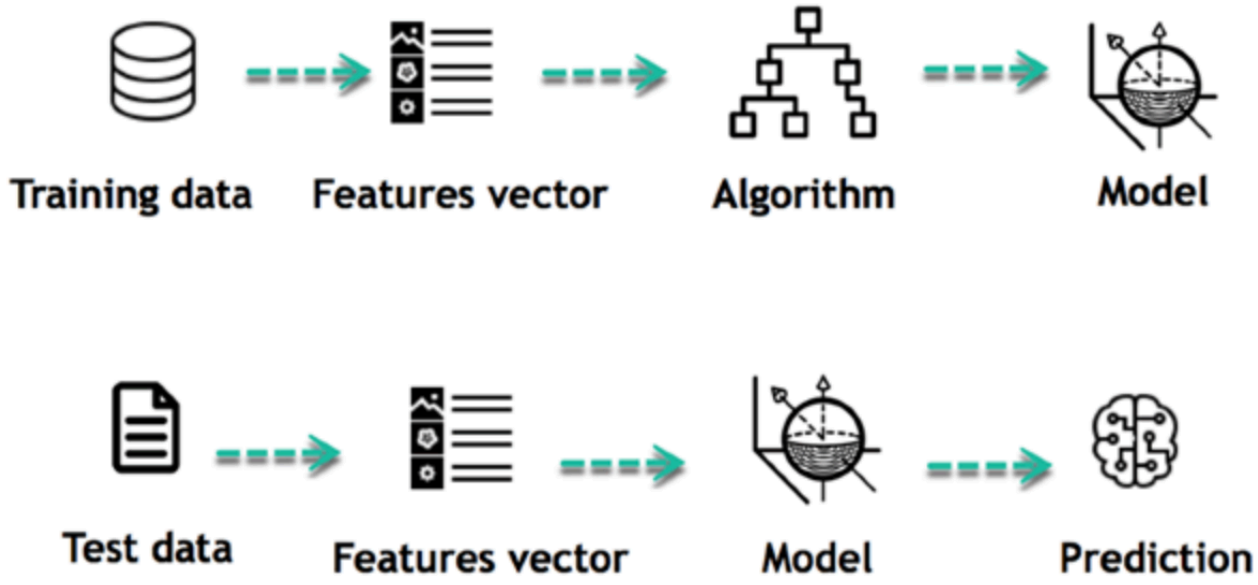
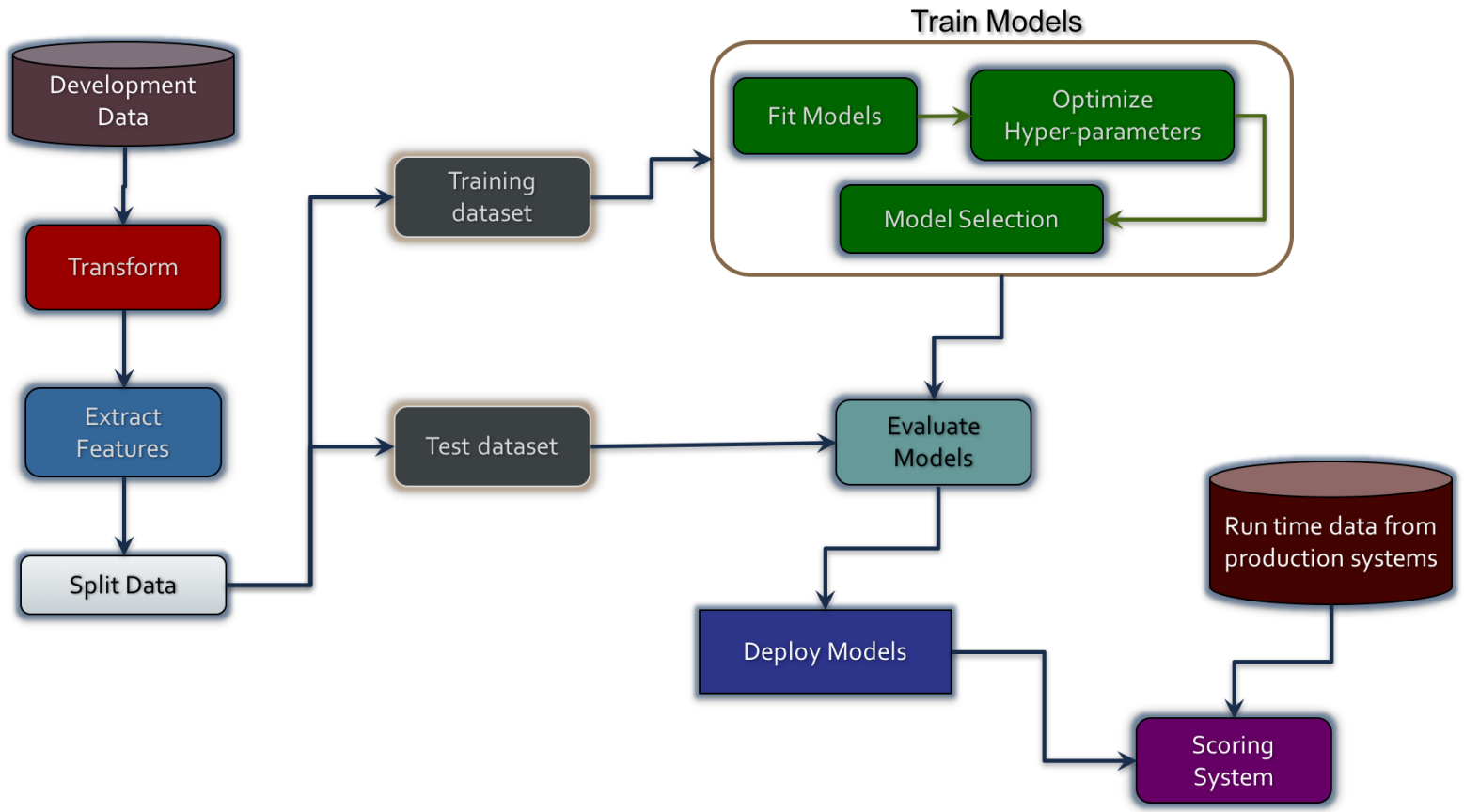


Image source: https://www.guru99.com/machine-learning-tutorial.html

How does Machine learning work?

- Machine learning is the brain where all the learning takes place.
- The way the machine learns is similar to the human being.
  - Humans learn from experience.
  - The more we know, the more easily we can predict.
  - By analogy, when we face an unknown situation, the likelihood of success is lower than the known situation.
- Machines are trained the same.
  - To make an accurate prediction, the machine sees an example.
  - When we give the machine a similar example, it can figure out the outcome.
  - However, like a human, if its feed a previously unseen example, the machine has difficulties to predict.

A Typical Machine Learning Process



A Typical Machine Learning Process

- The first step is to get the raw data.
- Once we have the raw data, we'll transform it.
  - For example, we can convert a comma-separated CSV format into a DataFrame consisting of strings and numbers.
- Then we extract the features and labels that can be used to train our machine learning models, such as
  - separating lines into words;
  - normalizing words, such as deleting special characters and converting words to lowercase;
  - turning columns into categories, for example, Yes/No to 1/0 or Survived/Dead to 1/0.

A Typical Machine Learning Process

- Once we have the features and labels, the next step is to split them into training set, testing set and validation set.
- For example, the 8-1-1 random split:
  - Training: train the model with 80 percent.
  - Validation: select the model with the best hyperparameters with 10 percent.
  - Testing: evaluate the model with 10 percent.
- Notice that using data in validation set and training set for training will lead to data leakage and less generalized model.

A Typical Machine Learning Process

- The training part has multiple steps:
  - First we try different models and fit the training data into the algorithms to develop the models.
  - Different algorithms have different hyperparameters. So even while developing the models, we need to tune the hyper-parameters.
  - Finally, we might try different algorithms and choose the best one that fits the problem.
- Once the model is selected with the optimized parameters on validation set, the test data is run with the model, which is used to measure the performance of the model.
- The model is then deployed into production where it uses the actual runtime data for predictions.

Machine Learning Tasks

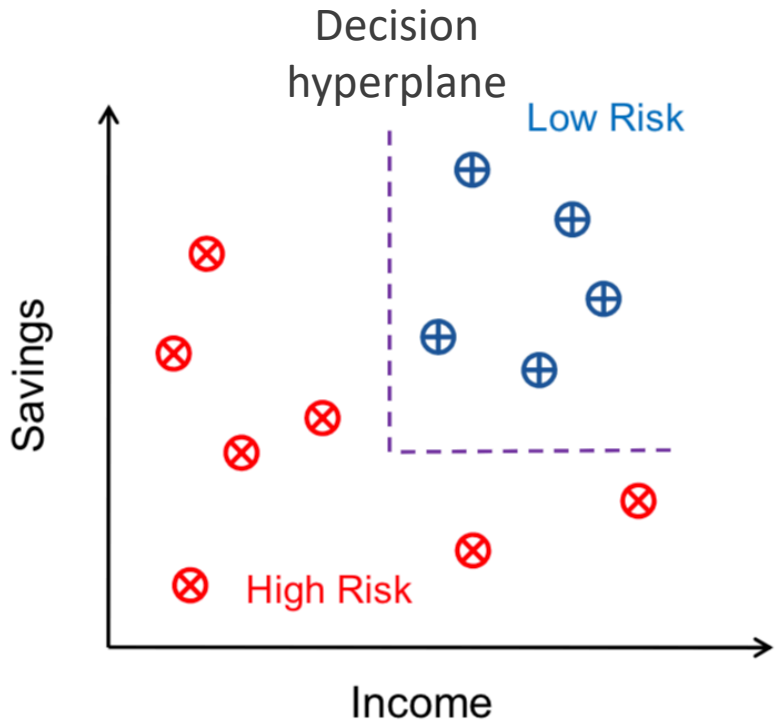
- Machine learning can be grouped into two broad learning tasks:
  - Supervised learning: the label information is available during training.
  - Unsupervised learning: the label information is unavailable during training.

Supervised Learning

- We have the label information such that we are able to know if a sample is correctly predicted or not during training.
  - E.g. the market knows if a customer buys the lipstick or not and use it in training.
- The label in the training data is also called *ground truth*.
- There are two categories of supervised learning:
  - Classification
  - Regression

Classification

- The label of classification problem is discreate.
- The discreate value that the label can take is called *class*.
  - If the label only has two classes, it is called *binary classification*.
  - If the label only has more than two classes, it is called *multiclass classification*.



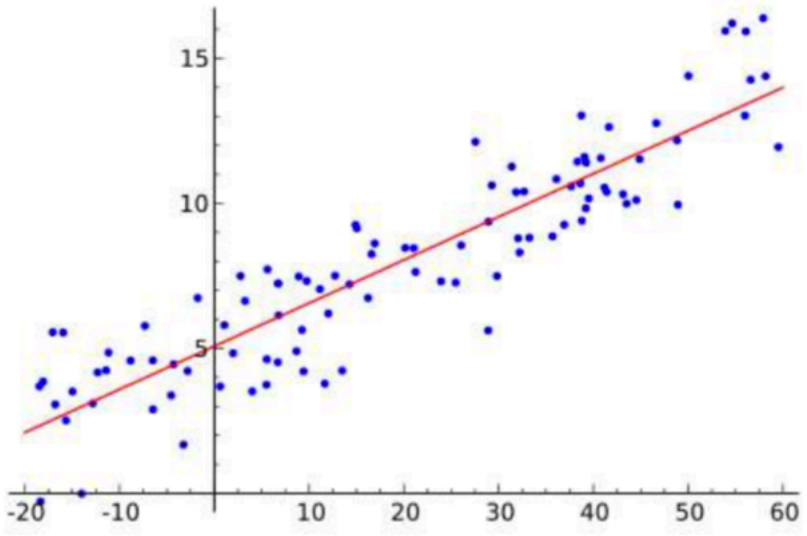
Binary classification with 2-d feature space

Classification Examples

- Binary classification:
  - Face verification (True, False)
  - Sentiment analysis (positive, negative)
  - Spam filtering (True, False)
  - Cancer diagonosis (benign, malignant)
- Multiclass classification:
  - Face identification (Alice, Bob, Charles, …)
  - Object recognition (flower, ball, cup, dog, …)
  - Weather prediction (sunny, rainy, foggy, …)
  - Behavior recognition (walking, running, dancing, …)

Regression

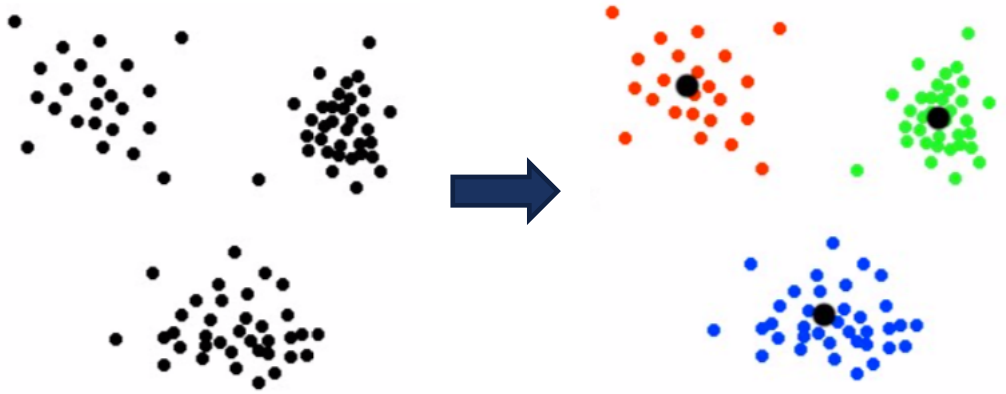
- The label of classification problem is continuous.
- Example:
  - Stock price prediction.
  - Temperature prediction.
  - Salary estimation.
  - House price prediction.



Linear regression

Unsupervised Learning

- In unsupervised learning, a model is trained to learn previously undetected patterns in a data set with no pre-existing labels.
  - You can use it when you do not know how to classify the data, and you want the algorithm to find patterns and classify the data for you.
- Example:
  - Social network user grouping.
  - Image segmentation.
  - Anomaly detection.



Clustering

Machine Learning Based on Big Data

- Traditional machine learning algorithms can only be used on a small amount of data due to the limitations of technology and single machine storage, relying on sampled data.
- The emergence of big data technology can support machine learning on full data.
- Machine learning algorithms involve a large number of iterative computations:
  - Disk-based MapReduce is not suitable for a large number of iterative computations.
  - Memory-based Spark is more suitable for a large number of iterative computations.

MLLIB

What is Spark MLlib?

- Spark provides a machine learning library based on big data to simplify the engineering practice of machine learning tasks.
- It provides a *distributed implementation* of common machine learning algorithms.
  - Some classic machine learning algorithms are not included because they cannot be executed in parallel.
- Developers only need to know:
  - Basics of Spark.
  - Principles of machine learning algorithms.
  - Meaning of algorithm-related parameters.

What is Spark MLlib?

At a high level, MLlib provides tools such as:

- **ML Algorithms**: common learning algorithms such as classification, regression, clustering, and collaborative filtering.
- **Featurization**: feature extraction, transformation, dimensionality reduction, and selection.
- **Pipelines**: tools for constructing, evaluating, and tuning ML Pipelines.
- **Persistence**: save and load algorithms, models, and pipelines.
- **Utilities**: linear algebra, statistics, data handling, etc.

Spark DataFrame

- DataFrame is a more specific data type in Spark that is built on top of RDD.
- Unlike an RDD, data in DataFrame is organized into named columns, like a table in a relational database.
- Designed to make large data sets processing even easier, DataFrame
  - allows developers to impose a structure onto a distributed collection of data;
  - allows higher-level abstraction;
  - provides a domain specific language API to manipulate your distributed data;
  - makes Spark accessible to a wider audience, beyond specialized data engineers.
- DataFrame operations is silimar to RDD operations:
  - https://spark.apache.org/docs/latest/api/python/pyspark.sql.html#pyspark.sql.DataFrame

DataFrame-based API

- As of Spark 2.0, the RDD-based APIs in the `spark.mllib` package have entered maintenance mode.
- The primary Machine Learning API for Spark is now the DataFrame-based API in the `spark.ml` package.
- Why is MLlib switching to the DataFrame-based API?
  - DataFrames provide a more user-friendly API than RDDs. The many benefits of DataFrames include Spark Datasources, 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.

ML Pipelines

- ML Pipelines provide a uniform set of high-level APIs built on top of DataFrames that help users create and tune practical machine learning pipelines.
- There are several pipeline components:
  - DataFrame: This ML API uses DataFrame from Spark SQL as an ML dataset, which can hold a variety of data types.
  - Transformer: Transform one DataFrame into another DataFrame.
    - E.g., an ML model is a Transformer which transforms a DataFrame with features into a DataFrame with predictions.
  - Estimator: An Estimator is an algorithm which can be fit on a DataFrame to produce a Transformer.
    - E.g., a learning algorithm is an Estimator which trains on a DataFrame and produces a model.
  - Pipeline: A Pipeline chains multiple Transformers and Estimators together to specify an ML workflow.
  - Parameter: All Transformers and Estimators now share a common API for specifying parameters.

Transformers

- A Transformer is an abstraction that includes:
  - *Feature transformers:* DataFrame (feature) -> DataFrame (feature).
  - *Learned models:* DataFrame (feature) -> DataFrame (label).
- Technically, a Transformer implements a method `transform()`, which converts one DataFrame into another, generally by appending one or more columns.
- For example:
  - 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.

Estimators

- An Estimator abstracts the concept of a learning algorithm or any algorithm that fits or trains on data.
  - DataFrame (training data) -> Transformer (model).
- Technically, an Estimator implements a method `fit()`, which accepts a DataFrame and produces a Model, which is a Transformer.
- For example, a learning algorithm such as LogisticRegression is an Estimator, and calling `fit()` trains a LogisticRegressionModel, which is a Model and hence a Transformer.

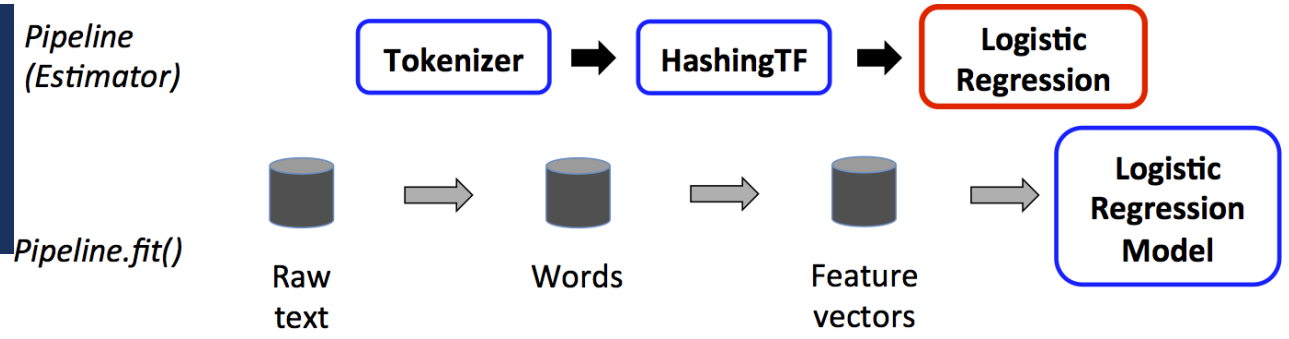
Pipeline

- In machine learning, it is common to run a sequence of algorithms to process and learn from data.
- For example, a simple text document processing workflow might include several stages:
  - Split each document's text into words.
  - Convert each document's words into a numerical feature vector.
  - Learn a prediction model using the feature vectors and labels.

Pipeline

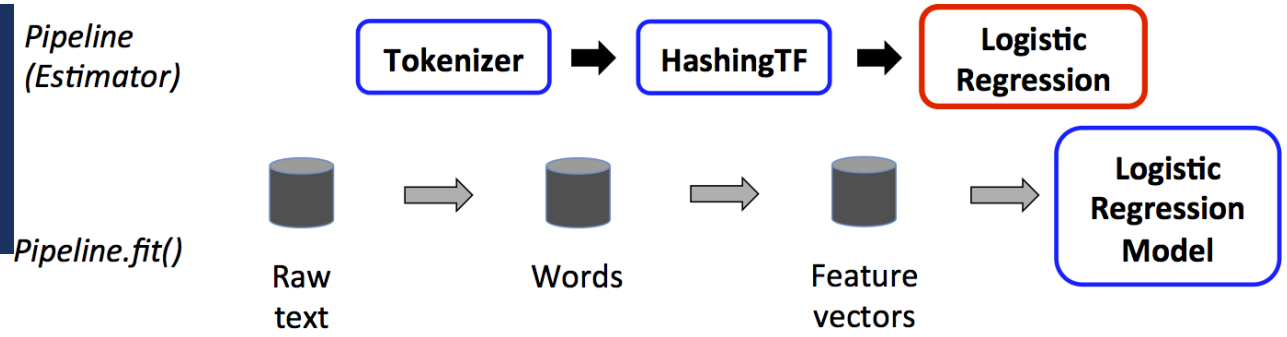
- A Pipeline is specified as a sequence of stages, and each stage is either a Transformer or an Estimator.
- These stages are run in order, and the input DataFrame is transformed as it passes through each stage.
  - For Transformer stages, the `transform()` method is called on the DataFrame to produce another DataFrame.
  - For Estimator stages, the `fit()` method is called on the DataFrame to produce a Transformer.

Training Pipeline



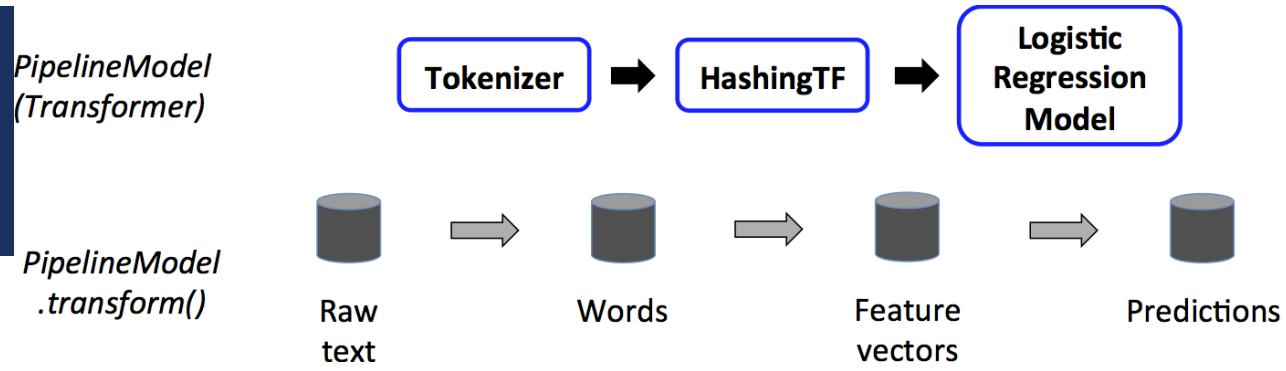
- We illustrate this for the simple text document workflow. The figure is for the *training time* usage of a Pipeline.
- The top row represents a Pipeline with three stages. The first two (Tokenizer and HashingTF) are Transformers (blue), and the third (LogisticRegression) is an Estimator (red).
- The bottom row represents data flowing through the pipeline, where cylinders indicate DataFrames.

Training Pipeline



- The `Pipeline.fit()` method is called on the original DataFrame, which has raw text documents and labels.
  - The whole pipeline is an Estimator to produce a transformer (model) from a DataFrame (raw data).
- The `Tokenizer.transform()` method splits the raw text documents into words, adding a new column with words to the DataFrame.
- The `HashingTF.transform()` method converts the words column into feature vectors, adding a new column with those vectors to the DataFrame.
- Since `LogisticRegression` is an Estimator, the Pipeline first calls `LogisticRegression.fit()` to produce a `LogisticRegressionModel`.

Testing Pipeline



- After a `Pipeline`'s `fit()` method runs, it produces a `PipelineModel`, which is a Transformer used at *test time*.
- The `PipelineModel` has the same number of stages as the original Pipeline but with all Transformers.
  - The last one is changed from Estimator to Transformer.
- When the `PipelineModel`'s `transform()` method is called on a test dataset, the data are passed through the fitted pipeline in order.
- Each stage's `transform()` method updates the dataset and passes it to the next stage.

DAG Pipelines

- A Pipeline's stages are specified as an ordered array.
- The previous example is a linear Pipeline, i.e., Pipeline in which each stage uses data produced by the previous stage.
- It is possible to create non-linear Pipelines as long as the data flow graph forms a Directed Acyclic Graph (DAG).
- If the Pipeline forms a DAG, then the stages must be specified in topological order.

Runtime Checking

- Since Pipelines can operate on DataFrames with varied types, they cannot use compile-time type checking.
  - The DataFrame is unknown while building the pipeline.
- Pipelines and PipelineModels instead do runtime checking before actually running the Pipeline.
- This type checking is done using the DataFrame *schema*, a description of the data types of columns in the DataFrame.

Unique Pipeline Stages

- A Pipeline's stages should be unique instances.
- The same instance myHashingTF should not be inserted into the Pipeline twice since Pipeline stages must have unique IDs.
- However, different instances myHashingTF1 and myHashingTF2 (both of type HashingTF) can be put into the same Pipeline since different instances will be created with different IDs.

Parameters

- There are two main ways to pass parameters to an algorithm:
  - Set parameters for an instance. E.g., 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.
- Parameters belong to specific instances of Estimators and Transformers.
  - For example, if we have two `LogisticRegression` instances `lr1` and `lr2`, then we can build a `ParamMap` with both `maxIter` parameters specified.
  - This is useful if there are two algorithms with the `maxIter` parameter in a Pipeline.

Example

- Build training data.

```
from pyspark.ml import Pipeline
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.feature import HashingTF, Tokenizer

# Prepare training documents from a list of (id, text, label) tuples.
training = spark.createDataFrame([
    (0, "a b c d e spark", 1.0),
    (1, "b d", 0.0),
    (2, "spark f g h", 1.0),
    (3, "hadoop mapreduce", 0.0)
], ["id", "text", "label"])
```

```
training.show()
```

```
+---+----------------+-----+
| id|            text|label|
+---+----------------+-----+
|  0| a b c d e spark|  1.0|
|  1|             b d|  0.0|
|  2|     spark f g h|  1.0|
|  3|hadoop mapreduce|  0.0|
+---+----------------+-----+
```

Example

- Define PipelineStage in Pipeline, including Transformers and Estimators (tokenizer, hashingTF and lr).
- Each stage has a unique id.

```
# Configure an ML pipeline, which consists of three stages: tokenizer, hashingTF, and lr.
tokenizer = Tokenizer(inputCol="text", outputCol="words")
hashingTF = HashingTF(inputCol=tokenizer.getOutputCol(), outputCol="features")
lr = LogisticRegression(maxIter=10, regParam=0.001)
pipeline = Pipeline(stages=[tokenizer, hashingTF, lr])
```

```
pipeline.getStages()
```

```
[Tokenizer_4921b3bf7320,
 HashingTF_b60580768399,
 LogisticRegression_08b856f5aa49]
```

Example

- Now the Pipeline just built is essentially an Estimator. After its fit() method is run using the training data, it will generate a PipelineModel, which is a Transformer.

```
# Fit the pipeline to training documents.
model = pipeline.fit(training)
```

```
model.stages
```

```
[Tokenizer_4921b3bf7320,
 HashingTF_b60580768399,
 LogisticRegressionModel: uid = LogisticRegression_08b856f5aa49, numClasses = 2, numFeatures = 262144]
```

Example

- Build test data.

```
# Prepare test documents, which are unlabeled (id, text) tuples.
test = spark.createDataFrame([
    (4, "spark i j k"),
    (5, "l m n"),
    (6, "spark hadoop spark"),
    (7, "apache hadoop")
], ["id", "text"])
```

```
test.show()
```

```
+---+----------------+
| id|            text|
+---+----------------+
|  4|       spark i j k|
|  5|             l m n|
|  6|spark hadoop spark|
|  7|     apache hadoop|
+---+----------------+
```

Example

- Make predictions on test sets.

```
# Make predictions on test documents and print columns of interest.
prediction = model.transform(test)
selected = prediction.select("id", "text", "probability", "prediction")
for row in selected.collect():
    rid, text, prob, prediction = row
    print("(%d, %s) --> prob=%s, prediction=%f" % (rid, text, str(prob), prediction))
```

```
(4, spark i j k) --> prob=[0.1596407738787475,0.8403592261212525], prediction=1.000000
(5, l m n) --> prob=[0.8378325685476744,0.16216743145232562], prediction=0.000000
(6, spark hadoop spark) --> prob=[0.06926633132976037,0.9307336686702395], prediction=1.000000
(7, apache hadoop) --> prob=[0.9821575333444218,0.017842466655578 08], prediction=0.000000
```

Conclusion

After this lecture, you should know:

- What is machine learning.
- What is the difference between supervised and unsupervised learning.
- What is a typical machine learning process.
- What is the pipeline in MLlib.

Course Project

- Course Project is released. The deadline is **18:00, 20th July**.

Thank you!

Reference:

- Machine Learning Tutorial for Beginners: https://www.guru99.com/machine-learning-tutorial.html
- The official guide: https://spark.apache.org/docs/latest/ml-guide.html

Acknowledgement: Thankfully acknowledge slide contents shared by Dr. Ye Luo