W05 – W08
SPARK MACHINE LEARNING

2024

The Hoang

LEARNING OBJECTIVES

Understanding following:

- What is Spark MLlib?
- Key Features of Spark MLlib
- ML Pipelines
- Common ML Algorithms in MLlib
- Data Types in Spark ML
- Transformer and Estimator
- Feature Engineering
- Model Evaluation
- Example: Building a Machine Learning Pipeline
- Tips for Working with Spark MLlib

WHAT IS SPARK MLLIB?

Definition

Spark MLlib is the machine learning module of Apache Spark which provides several algorithms for classification, regression, clustering, collaborative filtering, dimensionality reduction, and more. It is optimized for distributed computing, allowing you to scale machine learning tasks across clusters of machines.

Spark MLlib is divided into two APIs:

- RDD-based API (MLlib): The original API that works with RDDs and now is in *maintenance mode*.
- DataFrame-based API (Spark ML): A newer API introduced in Spark 2.0, which is more user-friendly and supports DataFrames. This is *primary API*.

The DataFrame-based API is the **recommended** approach for most tasks.

KEY FEATURES OF SPARK MLLIB?

Scalability: MLlib is designed to work with Big Data and can easily scale to clusters of machines.

Distributed Computing: Spark MLlib uses Spark's distributed architecture, so it can distribute computations across multiple nodes.

High-Level API: The DataFrame-based API lets you use familiar SQL-like operations for machine learning tasks.

Support for Multiple Languages: You can use Spark MLlib with Scala, Java, Python (PySpark), and R.

Integration with Pipelines: Spark MLlib allows you to build complex machine learning workflows using pipelines.

WHAT IS A MACHINE LEARNING WORKFLOW?

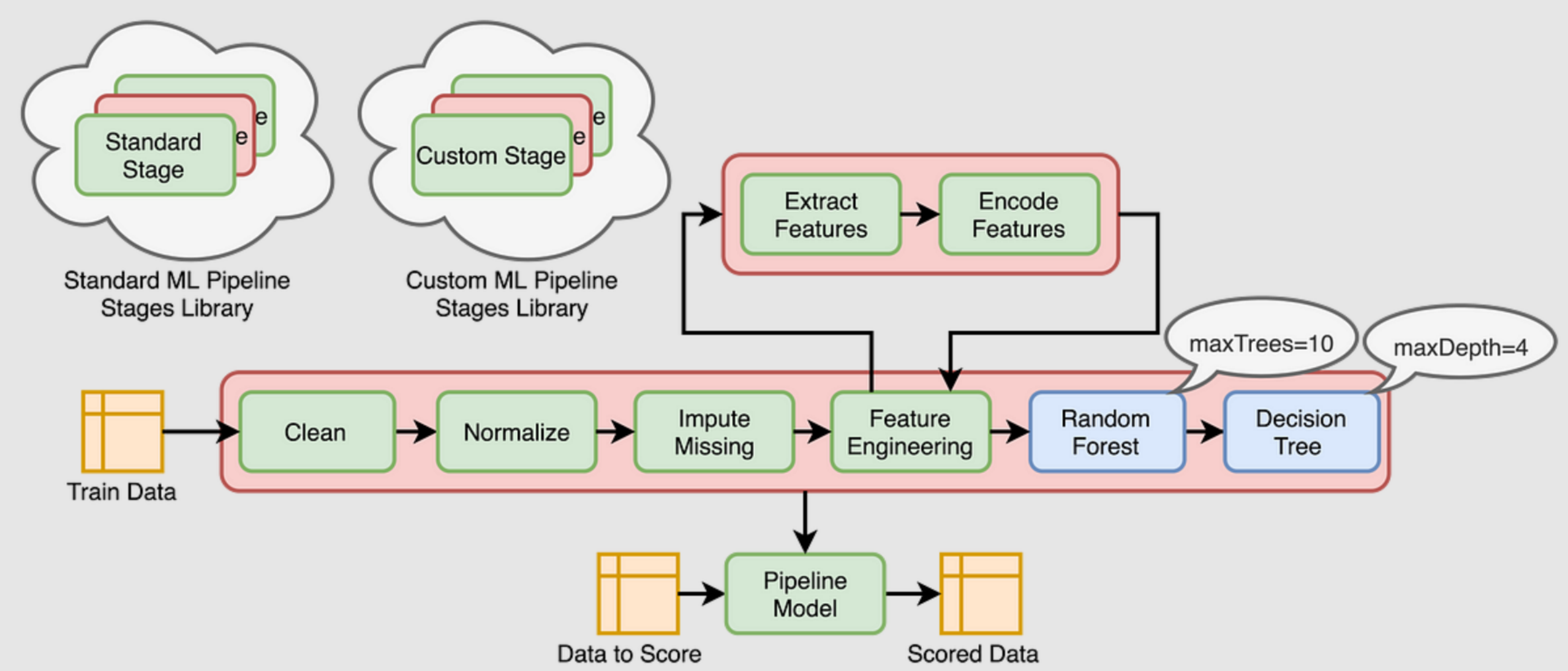
A machine learning workflow is a *structured process* that guides the *development* and *deployment* of *machine learning models*. It typically consists of several key stages:

1. **Problem Definition**: Clearly define the problem you want to solve and the objectives of your machine learning project.
2. **Data Collection**: Gather relevant data from various sources. This data can be structured (like databases) or unstructured (like text or images).
3. **Data Preprocessing**: Clean and prepare the data for analysis. This includes handling missing values, normalizing data, and encoding categorical variables.
4. **Exploratory Data Analysis (EDA)**: Analyze the data to understand its characteristics, identify patterns, and visualize relationships between variables.
5. **Feature Engineering**: Create new features or modify existing ones to improve model performance. This can involve selecting the most relevant features or transforming features into a more useful format.

WHAT IS A MACHINE LEARNING WORKFLOW?

6. **Model Selection**: Choose appropriate machine learning algorithms based on the problem type (e.g., classification, regression) and the nature of the data.
7. **Model Training**: Train the selected models using the prepared dataset. This involves feeding the data into the model and allowing it to learn from the data.
8. **Model Evaluation**: Assess the model's performance using metrics such as accuracy, precision, recall, and F1 score. This step often involves using a separate validation dataset.
9. **Hyperparameter Tuning**: Optimize the model's hyperparameters to improve performance. This can be done using techniques like grid search or random search.
10. **Deployment**: Implement the model in a production environment where it can make predictions on new data.
11. **Monitoring and Maintenance**: Continuously monitor the model's performance and update it as necessary to ensure it remains accurate over time.

ML PIPELINES



SPARK ML PIPELINES

A Pipeline in Spark MLlib is a sequence of stages where each stage is either a Transformer or an Estimator.

Pipelines help automate and streamline machine learning workflows by chaining together multiple steps, such as data processing, feature engineering, and model training.

Pipeline Components:

- **DataFrame**: The input data for the pipeline, typically structured as rows and columns.
- **Transformer**: A processing step that transforms the input DataFrame into another DataFrame (e.g., scaling or encoding features).
- **Estimator**: A learning algorithm that fits a model to the data and produces a Transformer (e.g., a classifier or regressor).
- **Pipeline**: Combines Transformers and Estimators into a sequence to be applied to the data.
- **Parameter**: a common API to specify parameters fo all Transformers and Estimators.

SPARK ML PIPELINES - ESTIMATOR

Estimator:

An Estimator implements a fit() method that takes a DataFrame and produces a Transformer (usually a fitted model).

Examples:

- LogisticRegression: Fits a logistic regression model to the input data.
- DecisionTreeClassifier: Fits a decision tree model to the input data.

```
# Learn a LogisticRegression model. This uses the parameters stored in lr.
model1 = lr.fit(training)
```

✓ 3.2s

SPARK ML PIPELINES - TRANSFORMER

Transformer:

A Transformer is an abstraction that implements a transform() method, which converts one DataFrame into another.

Examples:

- StandardScaler: Scales features to have zero mean and unit variance.
- VectorAssembler: Combines multiple columns into a single vector column.

```
# Make predictions on test data using the Transformer.transform() method.
# LogisticRegression.transform will only use the 'features' column.
# Note that model2.transform() outputs a "myProbability" column instead of the usual
# 'probability' column since we renamed the lr.probabilityCol parameter previously.
prediction = model2.transform(test)
```

✓ 0.1s

SPARK ML PIPELINES – FEATURE ENGINEERING

Feature engineering involves transforming raw data into features that can be used by machine learning algorithms. Spark MLlib provides various tools for feature engineering:

- **VectorAssembler**: Combines multiple columns into a single vector column.
- **StringIndexer**: Converts categorical string labels into numerical indices.
- **OneHotEncoder**: Converts indexed categorical features into a one-hot encoded vector.
- **StandardScaler**: Scales features to have zero mean and unit variance.
- **MinMaxScaler**: Scales features to a specified range, usually [0, 1].
- **PCA**: Reduces the dimensionality of the input data.

DATA TYPES IN SPARK ML

2. Vectors:

MLlib uses vectors to represent features for machine learning algorithms. There are two types of vectors:

- Dense Vectors: Store all entries explicitly.
- Sparse Vectors: Store only non-zero elements, useful for high-dimensional data.

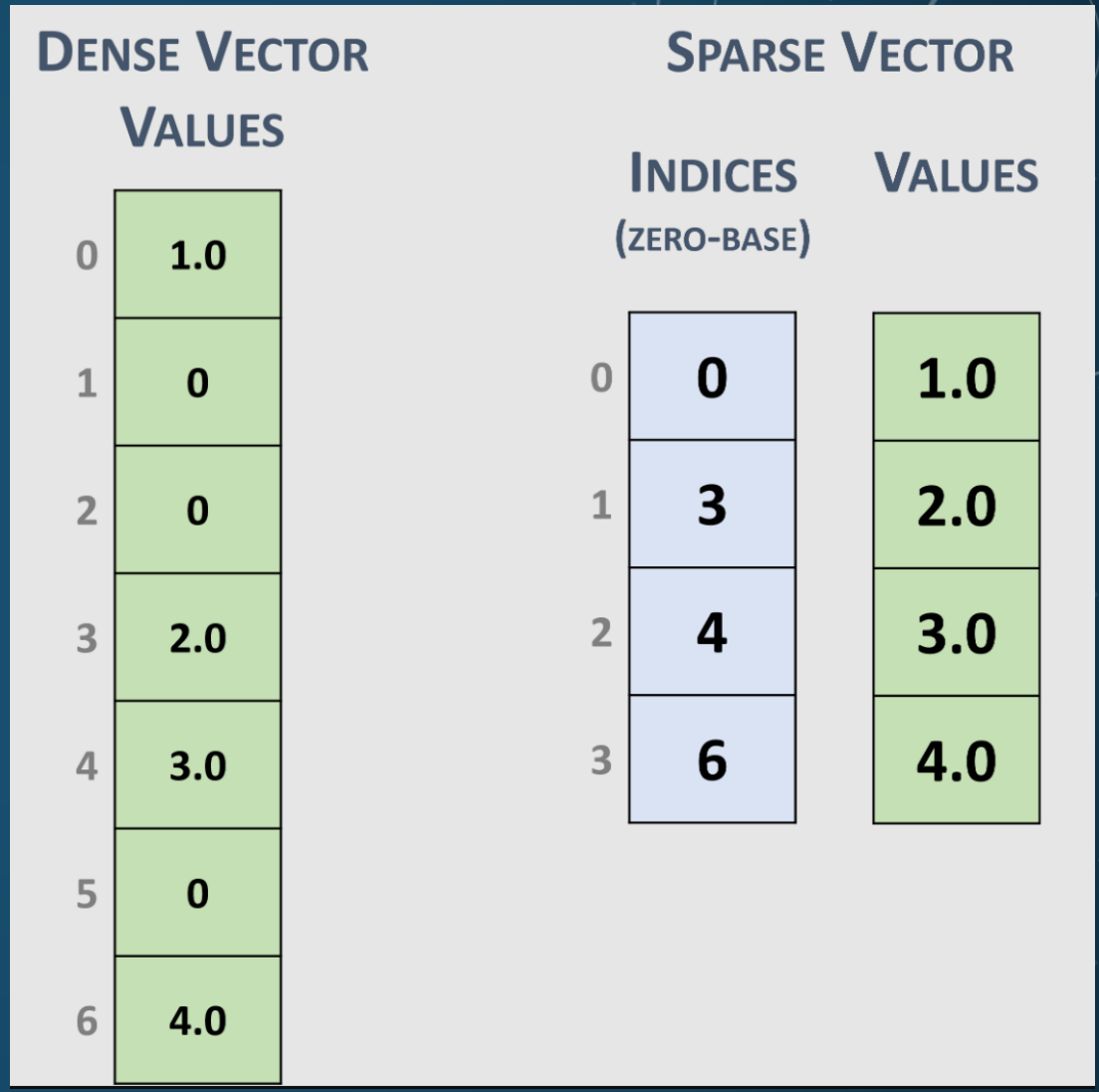
Structure of Sparse Vectors

A Sparse Vector is represented using two parallel arrays:

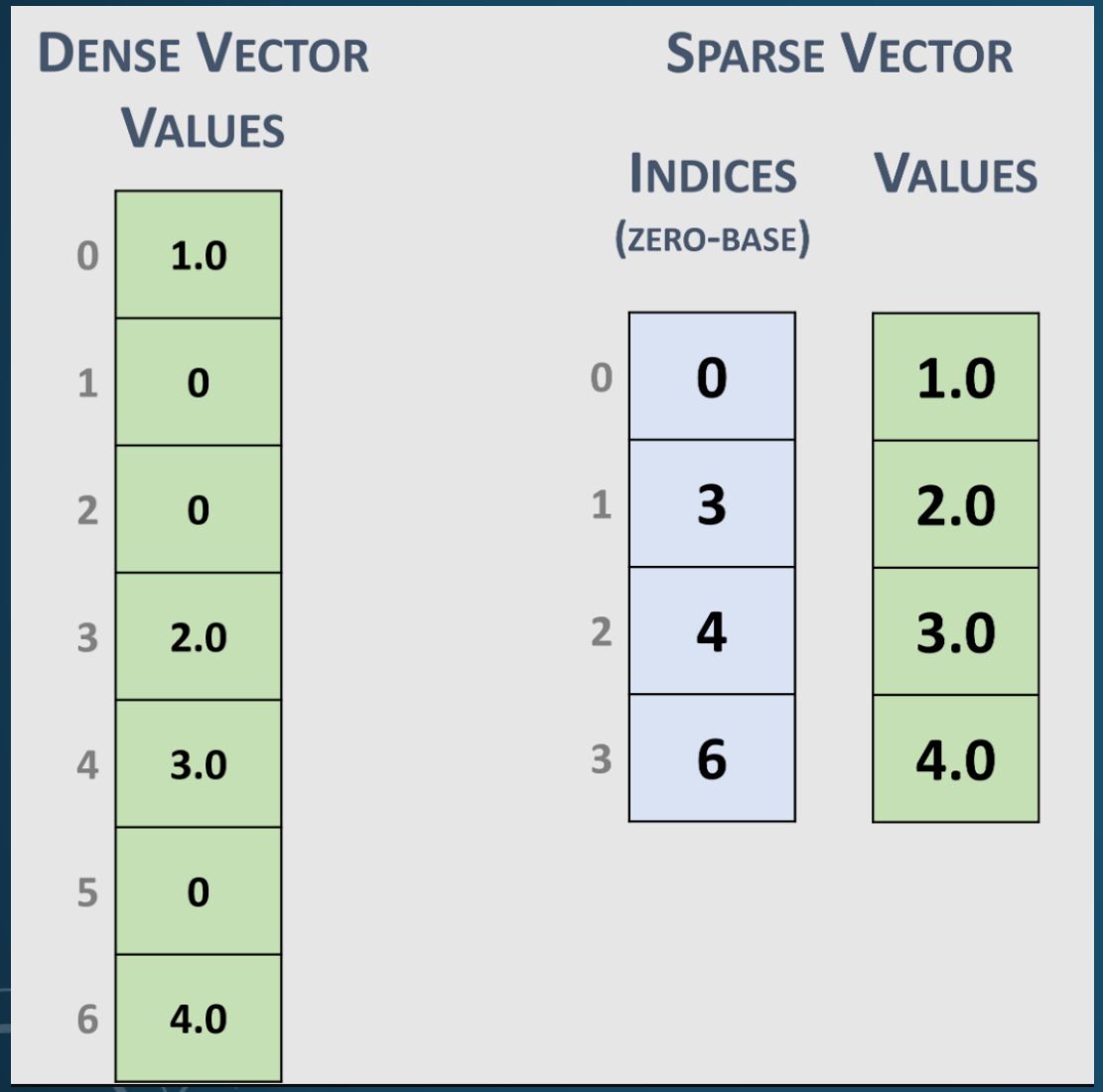
- Indices: An array that contains the indices of the non-zero elements.
- Values: An array that contains the corresponding non-zero values.

Examples

- Dense vector: [1.0, 0.0, 3.0]
- Sparse vector: (3, [0, 2], [1.0, 3.0]) represents a vector of length 3 with values 1.0 at index 0 and 3.0 at index 2.



DATA TYPES IN SPARK ML




```
[41]  vector_sparse = Vectors.sparse(7, [0,3,4,6],[1,2,3,4])
      ✓ 0.0s

[42]  vector_sparse
      ✓ 0.0s
...   SparseVector(7, {0: 1.0, 3: 2.0, 4: 3.0, 6: 4.0})

[43]  vector_sparse.values
      ✓ 0.0s
...   array([1., 2., 3., 4.])

[44]  vector_sparse.toArray()
      ✓ 0.0s
...   array([1., 0., 0., 2., 3., 0., 4.])
```

LAB PRACTICES

- W06/LAB01
- W06/LAB02
- W06/LAB03

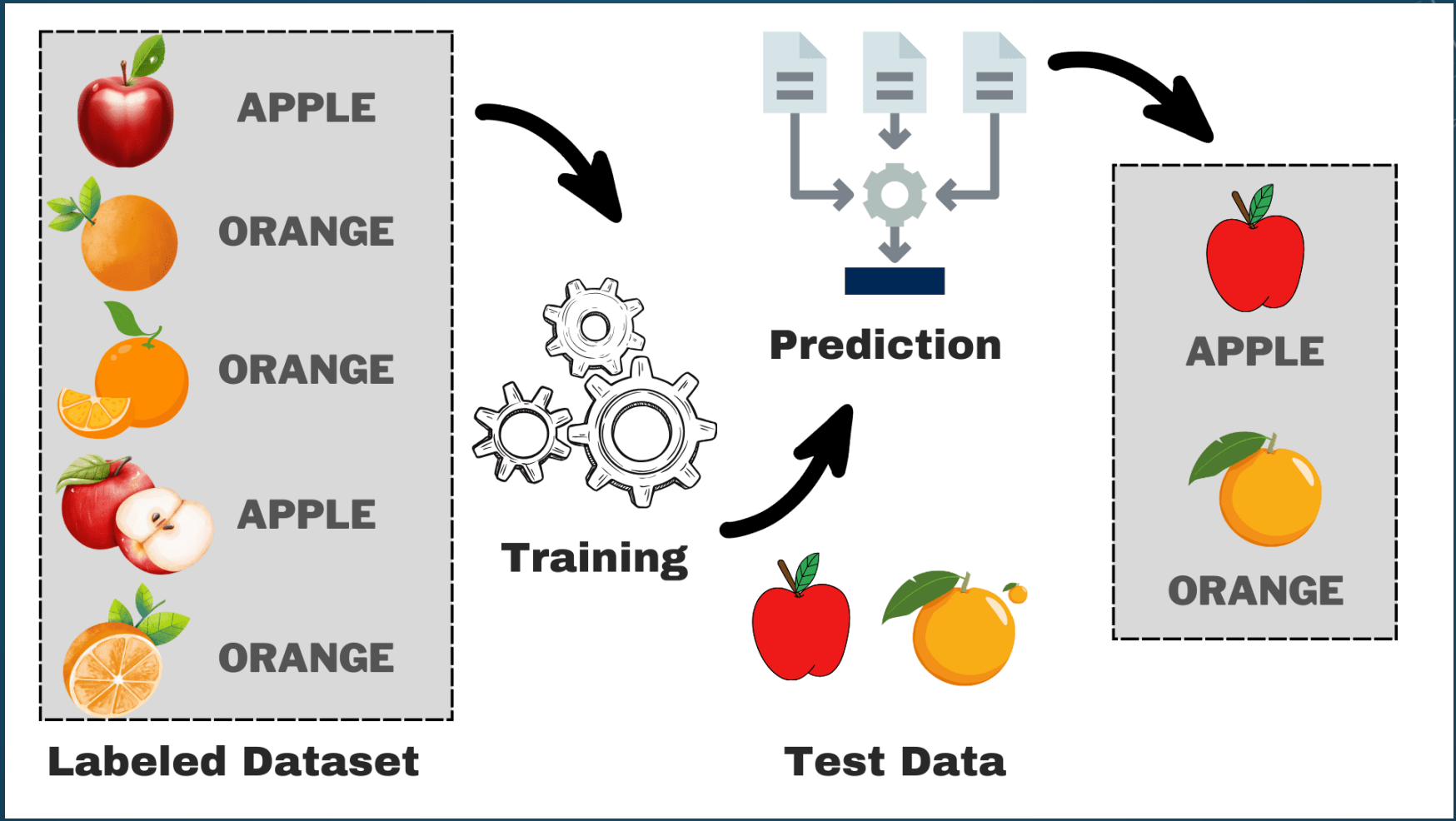
ML ALGORITHMS

Machine Learning algorithms can generally be grouped into several broad categories based on their learning paradigms, problem types, and tasks they are designed to solve. The most common types of machine learning algorithms include:

1. **Supervised Learning**
    1. Classification
    2. Regression
2. **Unsupervised Learning**
    1. Clustering
    2. Dimensionality Reduction
3. **Semi-Supervised Learning (not as commonly used in Spark MLlib)**
4. **Reinforcement Learning (not directly supported by Spark MLlib)**
5. **Recommendation Systems**
6. **Ensemble Methods (can be part of supervised learning)**

APACHE SPARK'S MLLIB – SUPERVISED LEARNING

Supervised learning involves training a model on labeled data. For each input, the corresponding output is known, and the model learns to predict the output for unseen data.



APACHE SPARK'S MLLIB – SUPERVISED LEARNING

Classification Algorithms - algorithms are used when the output is a discrete label (e.g., spam vs. not spam, fraud vs. not fraud).

1. Logistic Regression: org.apache.spark.ml.classification.LogisticRegression
2. Decision Trees (Classification): org.apache.spark.ml.classification.DecisionTreeClassifier
3. Random Forest (Classification): org.apache.spark.ml.classification.RandomForestClassifier
4. Gradient-Boosted Trees (Classification): org.apache.spark.ml.classification.GBTClassifier
5. Support Vector Machines (SVM): org.apache.spark.ml.classification.LinearSVC
6. Naive Bayes: org.apache.spark.ml.classification.NaiveBayes
7. Multilayer Perceptron (Neural Networks): org.apache.spark.ml.classification.MultilayerPerceptronClassifier

Use Cases:

- Email Spam Detection
- Fraud Detection
- Image Classification

APACHE SPARK'S MLLIB – SUPERVISED LEARNING

Regression Algorithms - algorithms are used when the output is a continuous value (e.g., predicting house prices, stock prices).

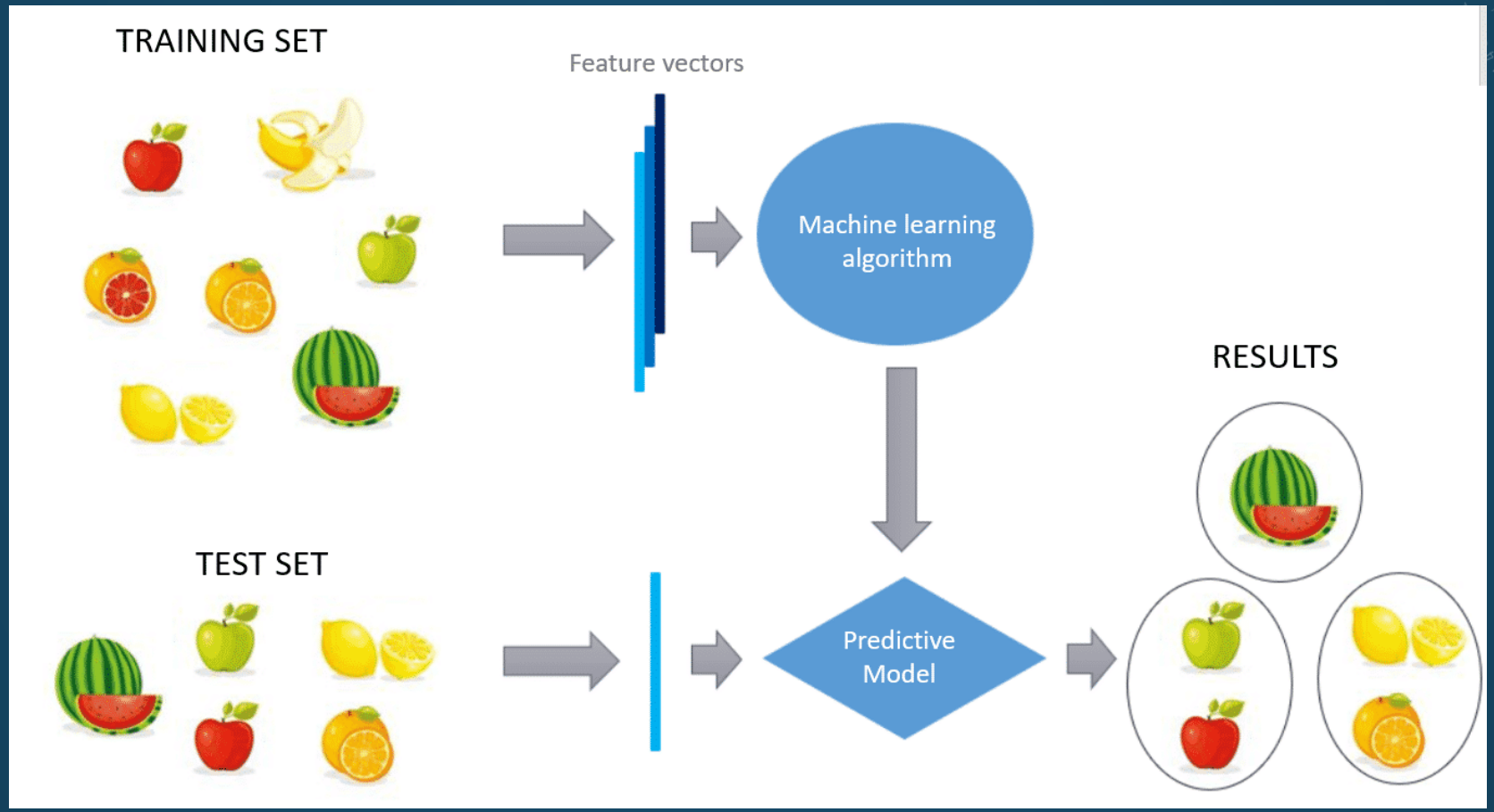
1. Linear Regression: org.apache.spark.ml.regression.LinearRegression
2. Decision Trees (Regression): org.apache.spark.ml.regression.DecisionTreeRegressor
3. Random Forest (Regression): org.apache.spark.ml.regression.RandomForestRegressor
4. Gradient-Boosted Trees (Regression): org.apache.spark.ml.regression.GBTRegressor
5. Generalized Linear Regression: org.apache.spark.ml.regression.GeneralizedLinearRegression
6. AFT Survival Regression (Accelerated Failure Time Model): org.apache.spark.ml.regression.AFTSurvivalRegression
7. Isotonic Regression: org.apache.spark.ml.regression.IsotonicRegression

Use Cases:

- Predicting house prices
- Forecasting sales
- Predicting stock prices

APACHE SPARK'S MLLIB – UNSUPERVISED LEARNING

Unsupervised Learning deals with unlabeled data, where the goal is to uncover hidden patterns, structures, or relationships within the data.



APACHE SPARK'S MLLIB – UNSUPERVISED LEARNING

Clustering Algorithms group the data into clusters based on similarity, without any predefined labels.

1. K-means: org.apache.spark.ml.clustering.KMeans
2. Bisecting K-means: org.apache.spark.ml.clustering.BisectingKMeans
3. Gaussian Mixture Models (GMM): org.apache.spark.ml.clustering.GaussianMixture
4. Latent Dirichlet Allocation (LDA): org.apache.spark.ml.clustering.LDA

Use Cases:

Customer segmentation

Document clustering (e.g., topic modeling)

Anomaly detection

APACHE SPARK'S MLLIB – UNSUPERVISED LEARNING

Dimensionality Reduction Algorithms reduce the number of features in the dataset while preserving its essential structure.

1. Principal Component Analysis (PCA): org.apache.spark.ml.feature.PCA
2. Singular Value Decomposition (SVD): (implemented as part of matrix factorization in Spark MLlib)
3. Truncated Singular Value Decomposition (SVD): org.apache.spark.ml.feature.TruncatedSVD

Use Cases:

- Data compression
- Noise reduction
- Visualization of high-dimensional data

APACHE SPARK'S MLLIB – RECOMMENDATION SYSTEMS

Recommendation systems provide personalized recommendations by learning user preferences from historical data. Spark MLlib supports collaborative filtering for building recommendation engines.

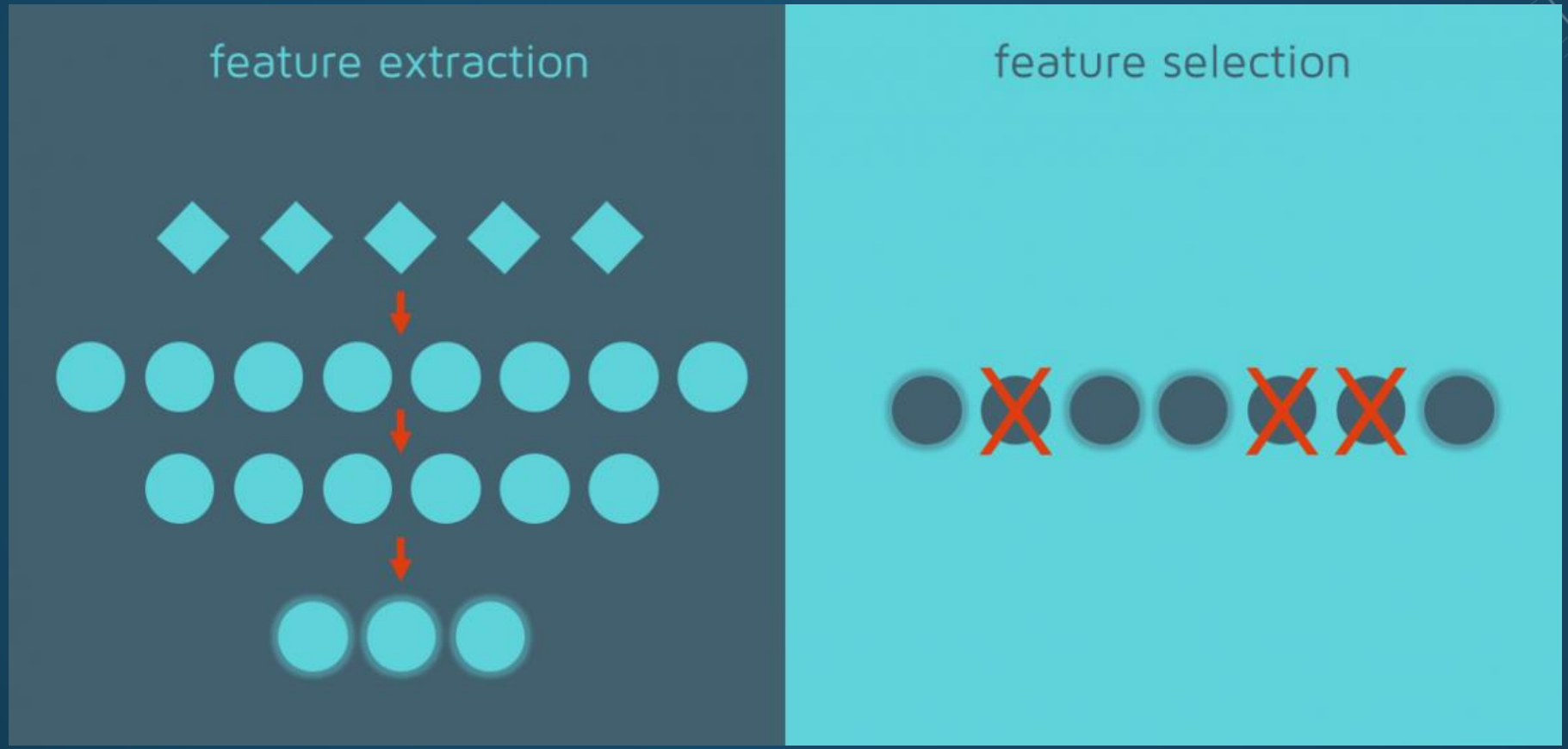
- Alternating Least Squares (ALS): org.apache.spark.ml.recommendation.ALS

Use Cases:

- Movie recommendations (e.g., Netflix)
- Product recommendations (e.g., Amazon)
- Music recommendations (e.g., Spotify)

APACHE SPARK'S MLLIB – FEATURE EXTRACTION AND TRANSFORMATION

Though not strictly "learning algorithms" feature extraction and transformation methods are essential for preparing data for machine learning models. Spark MLlib supports various techniques for transforming raw data into features that can be used in models.



APACHE SPARK'S MLLIB – FEATURE EXTRACTION AND TRANSFORMATION

Feature Scaling and Normalization

- StandardScaler: org.apache.spark.ml.feature.StandardScaler
- MinMaxScaler: org.apache.spark.ml.feature.MinMaxScaler
- MaxAbsScaler: org.apache.spark.ml.feature.MaxAbsScaler
- Normalizer: org.apache.spark.ml.feature.Normalizer

APACHE SPARK'S MLLIB – FEATURE EXTRACTION AND TRANSFORMATION

Categorical Feature Encoding

- One-Hot Encoding: org.apache.spark.ml.feature.OneHotEncoder
- StringIndexer: org.apache.spark.ml.feature.StringIndexer
- IndexToString: org.apache.spark.ml.feature.IndexToString

APACHE SPARK'S MLLIB – FEATURE EXTRACTION AND TRANSFORMATION

Text Feature Extraction

- TF-IDF (Term Frequency-Inverse Document Frequency): org.apache.spark.ml.feature.IDF
- Word2Vec: org.apache.spark.ml.feature.Word2Vec
- CountVectorizer: org.apache.spark.ml.feature.CountVectorizer
- HashingTF: org.apache.spark.ml.feature.HashingTF

APACHE SPARK'S MLLIB – FEATURE EXTRACTION AND TRANSFORMATION

Dimensionality Reduction

- PCA (Principal Component Analysis): org.apache.spark.ml.feature.PCA
- SVD (Singular Value Decomposition): org.apache.spark.ml.feature.TruncatedSVD

APACHE SPARK'S MLLIB – FEATURE EXTRACTION AND TRANSFORMATION

Use Cases:

- Text mining and sentiment analysis
- Preparing categorical and numerical data for machine learning models
- Reducing the dimensionality of data for visualization

APACHE SPARK'S MLLIB – ENSEMBLE METHODS

Ensemble methods combine multiple models to improve performance. Spark MLlib supports several ensemble methods, particularly for supervised learning.

- Random Forest (Classification and Regression): org.apache.spark.ml.classification.RandomForestClassifier and org.apache.spark.ml.regression.RandomForestRegressor
- Gradient-Boosted Trees (Classification and Regression): org.apache.spark.ml.classification.GBTClassifier and org.apache.spark.ml.regression.GBTRegressor

Use Cases:

- Predictive modeling in structured data
- Improving the accuracy and robustness of models
- Handling noisy datasets

APACHE SPARK'S MLLIB – MODEL EVALUATION AND TUNING

Model Selection and Hyperparameter Tuning

Spark MLlib provides tools for hyperparameter tuning and model selection using cross-validation and grid search.

- CrossValidator: org.apache.spark.ml.tuning.CrossValidator
- TrainValidationSplit: org.apache.spark.ml.tuning.TrainValidationSplit

APACHE SPARK'S MLLIB – MODEL EVALUATION AND TUNING

Model Evaluation Metrics

- Binary Classification: org.apache.spark.ml.evaluation.BinaryClassificationEvaluator
- Multiclass Classification: org.apache.spark.ml.evaluation.MulticlassClassificationEvaluator
- Regression: org.apache.spark.ml.evaluation.RegressionEvaluator
- Clustering: org.apache.spark.ml.evaluation.ClusteringEvaluator

APACHE SPARK'S MLLIB – SUMMARY

Category	Algorithms in Spark MLlib
Supervised Learning	Logistic Regression, Decision Trees, Random Forests, GBT, Linear Regression, Naive Bayes, etc.
Classification	Logistic Regression, Decision Trees, Random Forests, GBT, SVM, Naive Bayes, MLP
Regression	Linear Regression, Decision Trees, Random Forests, GBT, Generalized Linear Models, AFT
Unsupervised Learning	K-means, GMM, Bisecting K-means, LDA, PCA, SVD
Clustering	K-means, GMM, Bisecting K-means, LDA
Dimensionality Reduction	PCA, SVD
Recommendation Systems	ALS (Alternating Least Squares)
Ensemble Methods	Random Forests, Gradient-Boosted Trees
Feature Transformation	StandardScaler, MinMaxScaler, StringIndexer, OneHotEncoder, TF-IDF, Word2Vec, etc.
Evaluation & Tuning	CrossValidator, TrainValidationSplit, BinaryClassificationEvaluator, RegressionEvaluator, etc.

APACHE SPARK'S MLLIB – SUMMARY

Category	Algorithms in Spark MLlib
Supervised Learning	Logistic Regression, Decision Trees, Random Forests, GBT, Linear Regression, Naive Bayes, etc.
Classification	Logistic Regression, Decision Trees, Random Forests, GBT, SVM, Naive Bayes, MLP
Regression	Linear Regression, Decision Trees, Random Forests, GBT, Generalized Linear Models, AFT
Unsupervised Learning	K-means, GMM, Bisecting K-means, LDA, PCA, SVD
Clustering	K-means, GMM, Bisecting K-means, LDA
Dimensionality Reduction	PCA, SVD
Recommendation Systems	ALS (Alternating Least Squares)
Ensemble Methods	Random Forests, Gradient-Boosted Trees
Feature Transformation	StandardScaler, MinMaxScaler, StringIndexer, OneHotEncoder, TF-IDF, Word2Vec, etc.
Evaluation & Tuning	CrossValidator, TrainValidationSplit, BinaryClassificationEvaluator, RegressionEvaluator, etc.

APACHE SPARK'S MLLIB – TIPS

Tips for Working with Spark MLlib

- **Use DataFrames**: Always prefer the DataFrame-based API over the deprecated RDD-based API.
- **Caching**: Cache your data (using .cache()) if you are performing multiple operations or iterations on the same dataset to improve performance.
- **Parameter Tuning**: Use CrossValidator or TrainValidationSplit for hyperparameter tuning.
- **Model Persistence**: Save and load models with the save() and load() methods to reuse them across different sessions.
- **Scaling Data**: Many algorithms perform better when features are scaled. Always consider using StandardScaler or MinMaxScaler.