



SPARK

Spark



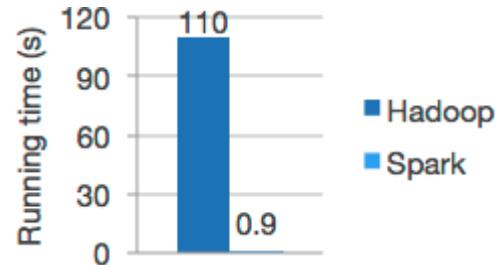
- **Spark** adalah *engine* analitik umum (*general engine*) yang cepat dalam pemrosesan *large-scale* Big Data.
- Salah satu project Apache, free dan open-source
- Spark merupakan *general purpose cluster engine* yang mendukung konsep sistem terdistribusi dengan *application programming interface* (APIs)
- Bisa digunakan **Java**, **Scala**, **Python**, dan **R** serta beberapa *library* untuk *streaming*, *graph* dan juga *machine learning* (mesin pembelajaran yang merupakan sekumpulan dari banyak algoritme di dalamnya)

Spark

- Spark disebut juga dengan “Lightning Fast Cluster Computing”.
- Spark 100x lebih cepat dari Hadoop MapReduce pada memory, dan 10x lebih cepat pada disk
- Spark dapat dijalankan standalone, di Hadoop, Mesos, atau di *cloud*.
- Spark dapat mengakses beragam sumber data termasuk HDFS, Cassandra, HBase, dan S3.

Key Features

- Speed
 - Run workloads 100x faster



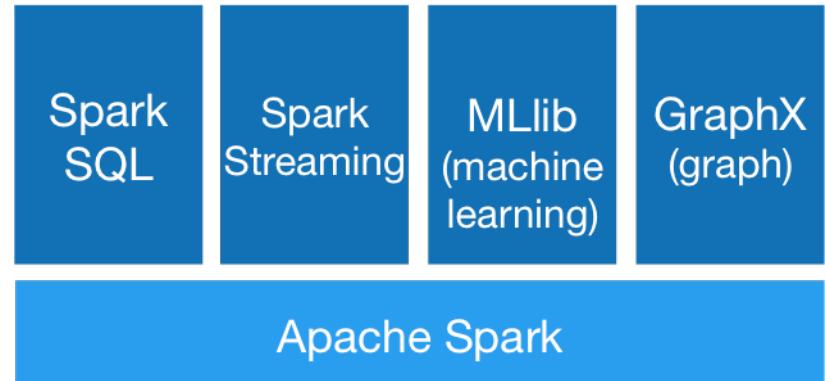
- Ease of Use
 - Write applications quickly in Java, Scala, Python, R, and SQL.

```
df = spark.read.json("logs.json")df.where("age >  
21") .select("name.first").show()
```

Spark's Python DataFrame API
Read JSON files with automatic schema inference

Key Features

- Generality
 - Combine SQL, streaming, and complex analytics.



- Runs Everywhere
 - Spark runs on Hadoop, Apache Mesos, Kubernetes, standalone, or in the cloud. It can access diverse data sources.



Resilient Distributed Datasets (RDDs)

- Spark menawarkan suatu fungsional dari pemrograman API untuk memanipulasi **Resilient Distributed Datasets (RDDs)**.
- **RDDs** merepresentasikan suatu *logical plan* untuk melakukan komputasi suatu *dataset*.
- **RDDs** adalah kumpulan record yang *resilient* dan terdistribusi, tersebar di satu atau banyak partisi

Resilient Distributed Datasets (RDDs)

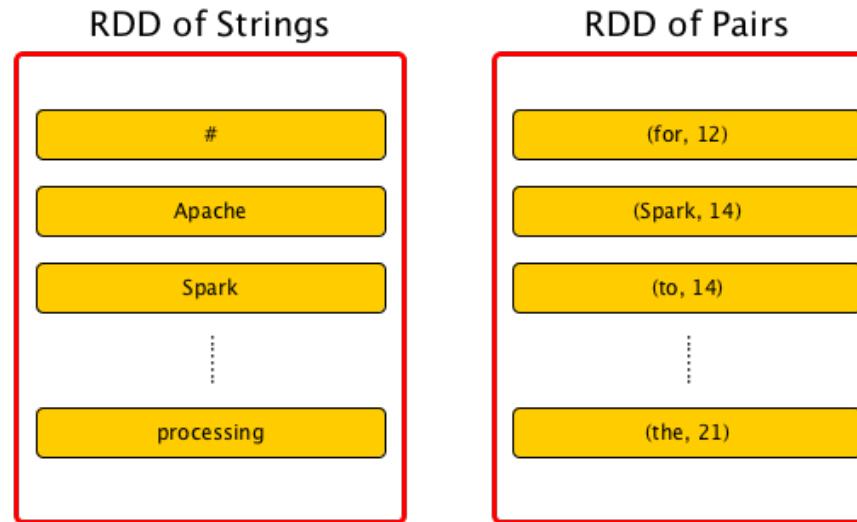
- **Resilient:** mendukung toleransi kesalahan (*fault-tolerant*), sehingga sistem dapat me-recover data yang hilang (*lost*) atau gagal saat diproses menggunakan ***lineage graph*** RDDs
 - (dengan me-running kembali operasi pada *input* data untuk me-rebuild kembali partisi yang hilang).
- **Distributed:** data berada pada beberapa node di suatu cluster
- **Datasets:** kumpulan data yang terpartisi (*partitioned data*)

Operasi pada RDDs

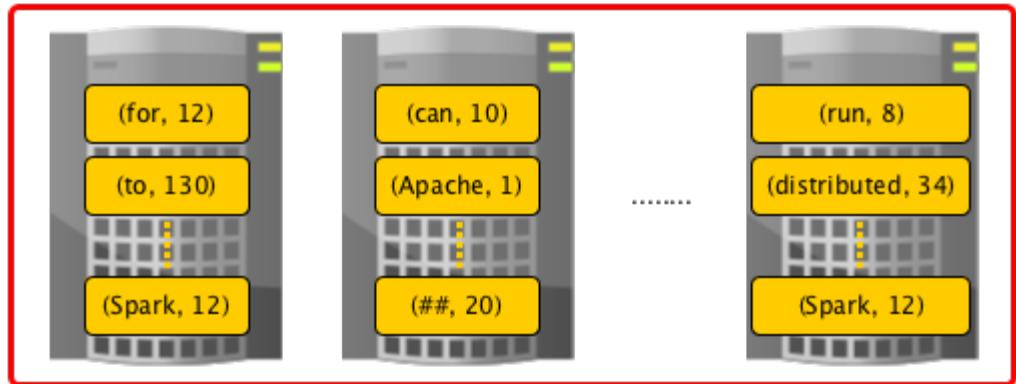
- RDDs memiliki 2 tipe operasi:
 - **Transformation:** mengonstruksi/membuat RDD baru dari satu atau lebih dari yang ada sebelumnya.
 - Contoh: map(), filter(), flatmap()
 - **Actions:** menghitung hasil dari suatu komputasi berdasarkan RDD dan memberikan *return/kembalian* kepada *program driver* atau simpan ke penyimpanan eksternal.
 - Contoh: reduce(), collect(), count(), first(), take()
- Semua transformasi di Spark bersifat *lazy* → tidak langsung menghitung hasil, tapi mengingat transformasi yang diberikan pada dataset
- Transformasi hanya akan dihitung setelah suatu action meminta hasil untuk diberikan ke program driver

Ilustrasi RDDs

- RDD bisa berupa *primitive value*, *value of value* (*tuples*, atau *object* lain)



distributed and partitioned RDD



Pembuatan RDDs

- Ada dua cara membuat RDDs:
 - **Parallelizing Collections**
 - Melakukan *parallelizing* dari data yang dimiliki
 - **External Dataset**
 - Mereferensi ke *dataset* di penyimpanan eksternal, misalnya *shared filesystem*, HDFS, Hbase, atau *data source* yang mendukung Hadoop InputFormat

Transformation

Transformation	Meaning
map(func)	Return a new distributed dataset formed by passing each element of the source through a function func.
filter(func)	Return a new dataset formed by selecting those elements of the source on which func returns true.
flatMap(func)	Similar to map, but each input item can be mapped to 0 or more output items (so func should return a Seq rather than a single item).
mapPartitions(func)	Similar to map, but runs separately on each partition (block) of the RDD, so func must be of type Iterator<T> => Iterator<U> when running on an RDD of type T.
mapPartitionsWithIndex(func)	Similar to mapPartitions, but also provides func with an integer value representing the index of the partition, so func must be of type (Int, Iterator<T>) => Iterator<U> when running on an RDD of type T.
sample(withReplacement, fraction , seed)	Sample a fraction fraction of the data, with or without replacement, using a given random number generator seed.
union(otherDataset)	Return a new dataset that contains the union of the elements in the source dataset and the argument.
intersection(otherDataset)	Return a new RDD that contains the intersection of elements in the source dataset and the argument.
distinct([numPartitions]))	Return a new dataset that contains the distinct elements of the source dataset.

Transformation

Transformation	Meaning
groupByKey([numPartition s])	When called on a dataset of (K, V) pairs, returns a dataset of (K, Iterable<V>) pairs.
reduceByKey(func, [numPartitions])	When called on a dataset of (K, V) pairs, returns a dataset of (K, V) pairs where the values for each key are aggregated using the given reduce function <i>func</i> , which must be of type (V,V) => V. Like in groupByKey, the number of reduce tasks is configurable through an optional second argument.
aggregateByKey(zeroValue)(seqOp, combOp, [numPartitions])	When called on a dataset of (K, V) pairs, returns a dataset of (K, U) pairs where the values for each key are aggregated using the given combine functions and a neutral "zero" value. Allows an aggregated value type that is different than the input value type, while avoiding unnecessary allocations. Like in groupByKey, the number of reduce tasks is configurable through an optional second argument.

Transformation

Transformation	Meaning
sortByKey([ascending], [numPartitions])	When called on a dataset of (K, V) pairs where K implements Ordered, returns a dataset of (K, V) pairs sorted by keys in ascending or descending order, as specified in the boolean ascending argument.
join(otherDataset, [numPartitions])	When called on datasets of type (K, V) and (K, W), returns a dataset of (K, (V, W)) pairs with all pairs of elements for each key. Outer joins are supported through leftOuterJoin, rightOuterJoin, and fullOuterJoin.
cogroup(otherDataset, [numPartitions])	When called on datasets of type (K, V) and (K, W), returns a dataset of (K, (Iterable<V>, Iterable<W>)) tuples. This operation is also called groupWith.
cartesian(otherDataset)	When called on datasets of types T and U, returns a dataset of (T, U) pairs (all pairs of elements).

Transformation

Transformation	Meaning
pipe(<i>command, [envVars]</i>)	Pipe each partition of the RDD through a shell command, e.g. a Perl or bash script. RDD elements are written to the process's stdin and lines output to its stdout are returned as an RDD of strings.
coalesce(<i>numPartitions</i>)	Decrease the number of partitions in the RDD to numPartitions. Useful for running operations more efficiently after filtering down a large dataset.
repartition(<i>numPartitions</i>)	Reshuffle the data in the RDD randomly to create either more or fewer partitions and balance it across them. This always shuffles all data over the network.
repartitionAndSortWithinPartitions(<i>partitioner</i>)	Repartition the RDD according to the given partitioner and, within each resulting partition, sort records by their keys. This is more efficient than calling repartition and then sorting within each partition because it can push the sorting down into the shuffle machinery.

Action

Action	Meaning
reduce(func)	Aggregate the elements of the dataset using a function <i>func</i> (which takes two arguments and returns one). The function should be commutative and associative so that it can be computed correctly in parallel.
collect()	Return all the elements of the dataset as an array at the driver program. This is usually useful after a filter or other operation that returns a sufficiently small subset of the data.
count()	Return the number of elements in the dataset.
first()	Return the first element of the dataset (similar to <code>take(1)</code>).
take(<i>n</i>)	Return an array with the first <i>n</i> elements of the dataset.

Action

Action	Meaning
takeSample(<i>withReplacement, num, [seed]</i>)	Return an array with a random sample of <i>num</i> elements of the dataset, with or without replacement, optionally pre-specifying a random number generator seed.
takeOrdered(<i>n, [ordering]</i>)	Return the first <i>n</i> elements of the RDD using either their natural order or a custom comparator.
saveAsTextFile(<i>path</i>)	Write the elements of the dataset as a text file (or set of text files) in a given directory in the local filesystem, HDFS or any other Hadoop-supported file system. Spark will call <code>toString</code> on each element to convert it to a line of text in the file.
saveAsSequenceFile(<i>path</i>) (Java and Scala)	Write the elements of the dataset as a Hadoop SequenceFile in a given path in the local filesystem, HDFS or any other Hadoop-supported file system.

Action

Action	Meaning
takeSample(<i>withReplacement, num, [seed]</i>)	Return an array with a random sample of <i>num</i> elements of the dataset, with or without replacement, optionally pre-specifying a random number generator seed.
saveAsObjectFile(<i>path</i>) (Java and Scala)	Write the elements of the dataset in a simple format using Java serialization, which can then be loaded using <code>SparkContext.objectFile()</code> .
countByKey()	Only available on RDDs of type (K, V). Returns a hashmap of (K, Int) pairs with the count of each key.
foreach(<i>func</i>)	Run a function <i>func</i> on each element of the dataset. This is usually done for side effects such as updating an Accumulator or interacting with external storage systems.

Latihan

Diberikan: $a=[(1,2),(3,4),(5,6)]$

1. Buat suatu kode dengan suatu ekspresi untuk mendapatkan hanya element kedua dari tiap tuple dari a .
2. Buat suatu kode untuk menghitung hasil penjumlahan elemen kedua.
3. Buat suatu kode untuk menghitung hasil penjumlahan elemen pertama yang bernilai ganjil.

DataFrame API

- API yang digunakan dalam MLlib ada dua:
 - RDD-based API
 - DataFrame-based API
- Mulai Spark 2.0 RDD-based API tidak lagi menjadi API utama
 - Masuk ke maintenance mode (tidak akan dimutakhirkan atau mendapat fitur baru)
- API **utama** yang digunakan dalam ML API di Spark adalah **DataFrame**

Spark SQL, DataFrames, Dataset

- **Spark SQL:** module Spark untuk pemrosesan data terstruktur
- Tidak seperti RDD API, interface yang disediakan Spark SQL memiliki lebih banyak informasi tentang struktur data dan komputasi yang dilakukan
- Secara internal, informasi tsb. digunakan untuk optimasi
- DataFrame API tersedia untuk Scala, Java, Python, R

Dataset dan DataFrame

- **Dataset (Spark)**: kumpulan data yang didistribusikan/tersebar
 - Memiliki kelebihan yang dimiliki RDD dan optimasi Spark SQL
- Dataset dapat dibuat melalui object JVM yang dimanipulasi melalui functional transformation
 - map, flatmap, filter, dll.
- Dataset API tersedia untuk Scala, Java
- Dataset API tidak tersedia untuk Python
 - Tapi banyak kelebihan di Python yang menyediakan fungsi layaknya Dataset API (begitu juga di R)

Dataset dan DataFrame

- **DataFrame (Spark)**: Dataset yang diatur ke dalam bentuk kolom bernama
- Secara konsep mirip dengan tabel di dalam RDBMS atau data frame di R/Python
- DataFrame dapat dibuat melalui banyak sumber:
 - File terstruktur (CSV, JSON, Parquet), table di Hive, external database, atau RDDs

Membaca File melalui DataFrame Spark

- Contoh file txt dengan delimiter “\t”
- Baris pertama tidak berisi nama kolom (header)

Bill Gates	M	32	C++
Guido van Rossum	M	26	Python
Linus Torvalds	M	72	Scala
Marissa Ann Mayer	F	25	Python
Ginni Rometty	F	24	C++

- Sebaiknya baris pertama berisi nama kolom

Membaca File melalui DataFrame Spark

```
8 from pyspark.sql import SparkSession
9
10 spark = SparkSession.builder \
11     .master("local") \
12     .appName("Test Spark") \
13     .getOrCreate()
14
15 df = spark.read.format("csv") \
16     .option("sep", "\t") \
17     .option("header", "false") \
18     .load("/home/hduser/data/spark/people.txt")
19
20 df = df.withColumnRenamed("_c0", "Name") \
21     .withColumnRenamed("_c1", "Sex") \
22     .withColumnRenamed("_c2", "Age") \
23     .withColumnRenamed("_c3", "Lang")
24 df.show()
25
26 #hentikan Spark Session
27 spark.stop()
```

- Baris 18: bernilai false bila tidak ada header pada baris 1 text
- Baris 19: memuat dari file lokal, bisa juga dengan protocol hdfs:// dan lainnya
- Baris 21-25: memberi nama kolom dengan cara rename dari default _cX bila tidak ada nama kolom
- Baris 27: menghentikan Spark Session untuk membebaskan memori terpakai

Hasil df.show()

Name	Sex	Age	Lang
Bill Gates	M	32	C++
Guido van Rossum	M	26	Python
Linus Torvalds	M	72	Scala
Marissa Ann Mayer	F	25	Python
Ginni Rometty	F	24	C++

Membaca File melalui DataFrame Spark

- Spark juga menyediakan API untuk memuat data ke DataFrame dengan perintah SQL
- Contohnya
 - File CSV

```
df = spark.sql("SELECT * FROM csv.`csv/file/path/in/hdfs`")
```
 - File JSON

```
df = spark.sql("SELECT * FROM json.`json/file/path/in/hdfs`")
```
- Bisa juga menambahkan predicate WHERE untuk filtering layaknya SQL pada RDBMS

Membaca File melalui DataFrame Spark: Filtering data

- Contoh filtering DataFrame
 - Age > 30

```
In [8]: x = df.filter(df.Age > 30)
```

```
In [9]: x.show()
```

```
+-----+-----+-----+
|       Name|Sex|Age| Lang|
+-----+-----+-----+
|    Bill Gates| M| 32| C++|
|Linus Torvalds| M| 72|Scala|
+-----+-----+-----+
```

- Lang = “Python”

```
In [10]: x = df.filter(df.Lang == "Python")
```

```
In [11]: x.show()
```

```
+-----+-----+-----+
|       Name|Sex|Age| Lang|
+-----+-----+-----+
| Guido van Rossum| M| 26|Python|
|Marissa Ann Mayer| F| 25|Python|
+-----+-----+-----+
```

Membaca File melalui DataFrame Spark: Filtering data

- Contoh filtering DataFrame
 - Age > 30

```
In [8]: x = df.filter(df.Age > 30)
```

```
In [9]: x.show()
```

```
+-----+-----+-----+
|       Name|Sex|Age| Lang|
+-----+-----+-----+
|    Bill Gates| M| 32| C++|
|Linus Torvalds| M| 72|Scala|
+-----+-----+-----+
```

- Lang = “Python”

```
In [10]: x = df.filter(df.Lang == "Python")
```

```
In [11]: x.show()
```

```
+-----+-----+-----+
|       Name|Sex|Age| Lang|
+-----+-----+-----+
| Guido van Rossum| M| 26|Python|
|Marissa Ann Mayer| F| 25|Python|
+-----+-----+-----+
```

DataFrame Spark: Filtering data

- Contoh filtering DataFrame
 - Lang = Python dan Sex = M

```
In [22]: x = df.filter((df.Lang == "Python") & (df.Sex == "M"))
```

```
In [23]: x.show()
```

```
+-----+-----+-----+
|       Name|Sex|Age|  Lang|
+-----+-----+-----+
|Guido van Rossum|  M| 26|Python|
+-----+-----+-----+
```

- Bisa juga dengan string SQL

```
In [26]: x = df.filter("Lang == 'Python' AND Sex == 'M'")
```

```
In [27]: x.show()
```

```
+-----+-----+-----+
|       Name|Sex|Age|  Lang|
+-----+-----+-----+
|Guido van Rossum|  M| 26|Python|
+-----+-----+-----+
```

Membaca File melalui DataFrame Spark: Aggregation

- Contoh fungsi **avg()** untuk menghitung rata-rata umur
 - Ubah dulu tipe data dari DataFrame kolom Age ke integer dengan fungsi **cast()**

```
In [30]: df = df.withColumn("Age", df["Age"].cast("int"))
```

```
In [31]: df.show()
```

	Name	Sex	Age	Lang
1	Bill Gates	M	32	C++
2	Guido van Rossum	M	26	Python
3	Linus Torvalds	M	72	Scala
4	Marissa Ann Mayer	F	25	Python
5	Ginni Rometty	F	24	C++

```
In [32]: df.groupBy().avg("Age").collect()  
Out[32]: [Row(avg(Age)=35.8)]
```

DataFrame Spark: Aggregation

- Contoh fungsi **groupBy()** dan **count()**
- Menghitung banyaknya data berdasarkan kolom Lang
- Menghitung banyaknya data berdasarkan kolom Lang dan Sex

```
In [37]: x = df.groupBy("Lang").count()
```

```
In [38]: x.show()
+---+---+
| Lang|count|
+---+---+
| C++|    2|
| Scala|   1|
| Python|   2|
+---+---+
```

```
In [39]: x = df.groupBy("Lang", "Sex").count()
```

```
In [40]: x.show()
+---+---+---+
| Lang|Sex|count|
+---+---+---+
| Python| F|    1|
| C++| F|    1|
| C++| M|    1|
| Scala| M|    1|
| Python| M|    1|
+---+---+---+
```

-
- Contoh yang lain dapat dicek melalui cheat sheet untuk PySpark DataFrame yang sudah diunggah
 - <https://www.datacamp.com/community/blog/pyspark-cheat-sheet-python>

RDD-based API

RDD-based API: Classification, Regression

- **Classification dan Regression:** proses pembelajaran supervised
- Algoritme yang didukung:
 - Linear model, logistic regression, SVM
 - Naïve Bayes
 - Decision trees
 - Ensemble of trees: Random Forest, Gradient-Boosted Trees
 - Isotonic Regression

Problem Type	Supported Methods
Binary Classification	linear SVMs, logistic regression, decision trees, random forests, gradient-boosted trees, naive Bayes
Multiclass Classification	logistic regression, decision trees, random forests, naive Bayes
Regression	linear least squares, Lasso, ridge regression, decision trees, random forests, gradient-boosted trees, isotonic regression

RDD-based API: Clustering

- **Clustering:** proses pembelajaran unsupervised yang mengelompokkan data berdasarkan kemiripan
- Algoritme yang didukung:
 - K-means
 - Gaussian mixture
 - Power iteration clustering (PIC)
 - Latent Dirichlet Allocation (LDA)
 - Bisecting k-means
 - Streaming k-means

RDD-based API: Dimensionality reduction

- **Dimensionality reduction:** merupakan proses mereduksi/mengurangi banyaknya variable yang akan digunakan sebagai fitur
- Algoritme yang didukung:
 - Singular Value Decomposition (SVD)
 - Principal Component Analysis (PCA)

RDD-based API: Extraction, Transformation

- **Feature Extraction dan Transformation:** berfungsi mengekstrasi fitur dari raw data dan juga mentransformasi fitur:
- Algoritme yang didukung:
 - TF-IDF
 - Word2Vec
 - StandardScaler, menstandarkan fitur dengan penskalaan berdasarkan ragam (variance) dan menghilangkan mean
 - Normalizer, menskalakan tiap unit sampel ke bentuk L^p norm
 - ChiSqSelector, menggunakan Chi-Squared feature selection
 - ElementwiseProduct
 - PCA

RDD-based API: Frequent Pattern Mining

- **Frequent Pattern Mining:** melakukan perlombongan data (*data mining*) untuk mengetahui pola data
- Algoritme yang didukung:
 - FP-growth
 - Association Rule
 - PrefixSpan

RDD-based API: Evaluation Metrics

- **Evaluation Metrics:** metrik untuk mengukur hasil pembelajaran model (evaluasi)
- Setiap algoritme memiliki metrik yang berbeda
- **Binary classification:** 1 dokumen 1 label → label {0,1}
 - Precision (Positive Predictive Value)
 - Recall (True Positive Rate)
 - F-measure
 - Receiver Operating Characteristic (ROC)
 - Area Under ROC Curve
 - Area Under Precision-Recall Curve

RDD-based API: Evaluation Metrics

- **Multiclass classification:** 1 dokumen 1 label → label {1..n}
 - Confusion Matrix
 - Accuracy
 - Precision by label
 - Recall by label
 - F-measure by label
 - Weighted precision
 - Weighted recall
 - Weighted F-measure

RDD-based API: Evaluation Metrics

- **Multilabel classification:** 1 dokumen m label → label {1...n}
 - Precision
 - Recall
 - Accuracy
 - Precision by label
 - Recall by label
 - F1-measure by label
 - Hamming Loss
 - Subset Accuracy
 - F1 Measure
 - Micro precision
 - Micro recall
 - Micro F1 Measure

RDD-based API: Evaluation Metrics

- Ranking system/sistem pemeringkatan
 - Precision at k ($P@k$)
 - Mean Average Precision
 - Normalized Discounted Cumulative Gain

DataFrame-based API

DataFrame-based API: Extracting, Transforming, Selection Features

- **Feature Extractors**

- [TF-IDF](#)
- [Word2Vec](#)
- [CountVectorizer](#)
- [FeatureHasher](#)

- **Feature Transformers**

- [Tokenizer](#)
- [StopWordsRemover](#)
- [nn-gram](#)
- [Binarizer](#)
- [PCA](#)
- [PolynomialExpansion](#)
- [Discrete Cosine Transform \(DCT\)](#)
- [StringIndexer](#)
- [IndexToString](#)
- [OneHotEncoder \(Deprecated since 2.3.0\)](#)
- [OneHotEncoderEstimator](#)

- **Feature Transformers**

- [VectorIndexer](#)
- [Interaction](#)
- [Normalizer](#)
- [StandardScaler](#)
- [MinMaxScaler](#)
- [MaxAbsScaler](#)
- [Bucketizer](#)
- [ElementwiseProduct](#)
- [SQLTransformer](#)
- [VectorAssembler](#)
- [VectorSizeHint](#)
- [QuantileDiscretizer](#)
- [Imputer](#)

DataFrame-based API: Extracting, Transforming, Selection Features

- Feature Selectors
 - VectorSlicer
 - RFormula
 - ChiSqSelector
- Locality Sensitive Hashing

DataFrame-based API: Classification and regression

- Classification
 - Logistic regression
 - Binomial logistic regression
 - Multinomial logistic regression
 - Decision tree classifier
 - Random forest classifier
 - Gradient-boosted tree classifier
 - Multilayer perceptron classifier
 - Linear Support Vector Machine
 - One-vs-Rest classifier (a.k.a. One-vs-All)
 - Naive Bayes

DataFrame-based API: Classification and regression

- Regression
 - Linear regression
 - Generalized linear regression
 - Available families
 - Decision tree regression
 - Random forest regression
 - Gradient-boosted tree regression
 - Survival regression
 - Isotonic regression

DataFrame-based API: Classification and regression

- Linear methods
- Decision trees
- Tree Ensembles
 - Random Forests
 - Gradient-Boosted Trees (GBTs)

DataFrame-based API: Clustering

- K-means
- Latent Dirichlet allocation (LDA)
- Bisecting k-means
- Gaussian Mixture Model (GMM)

Contoh Mllib: DataFrame-based API

Contoh kode untuk Klasifikasi dengan Naïve Bayes

```
from pyspark.ml.classification import NaiveBayes
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

# Load training data
data = spark.read.format("libsvm") \
    .load("data/mllib/sample_libsvm_data.txt")

# Split the data into train and test
splits = data.randomSplit([0.6, 0.4], 1234)
train = splits[0]
test = splits[1]

# create the trainer and set its parameters
nb = NaiveBayes(smoothing=1.0, modelType="multinomial")

# train the model
model = nb.fit(train)

# select example rows to display.
predictions = model.transform(test)
predictions.show()

# compute accuracy on the test set
evaluator = MulticlassClassificationEvaluator(labelCol="label", \
    predictionCol="prediction", metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print("Test set accuracy = " + str(accuracy))
```

Hasil Klasifikasi dengan Naïve Bayes

```
hduser@master:~  
>>> nb = NaiveBayes(smoothing=1.0, modelType="multinomial")  
>>> model = nb.fit(train)  
>>> predictions = model.transform(test)  
>>> predictions.show()  
+-----+-----+-----+-----+  
|label|    features| rawPrediction|probability|prediction|  
+-----+-----+-----+-----+  
| 0.0|(692,[95,96,97,12...|[-174115.98587057...|[1.0,0.0]| 0.0|  
| 0.0|(692,[98,99,100,1...|[-178402.52307196...|[1.0,0.0]| 0.0|  
| 0.0|(692,[100,101,102...|[-100905.88974016...|[1.0,0.0]| 0.0|  
| 0.0|(692,[123,124,125...|[-244784.29791241...|[1.0,0.0]| 0.0|  
| 0.0|(692,[123,124,125...|[-196900.88506109...|[1.0,0.0]| 0.0|  
| 0.0|(692,[124,125,126...|[-238164.45338794...|[1.0,0.0]| 0.0|  
| 0.0|(692,[124,125,126...|[-184206.87833381...|[1.0,0.0]| 0.0|  
| 0.0|(692,[127,128,129...|[-214174.52863813...|[1.0,0.0]| 0.0|  
| 0.0|(692,[127,128,129...|[-182844.62193963...|[1.0,0.0]| 0.0|  
| 0.0|(692,[128,129,130...|[-246557.10990301...|[1.0,0.0]| 0.0|  
| 0.0|(692,[152,153,154...|[-208282.08496711...|[1.0,0.0]| 0.0|  
| 0.0|(692,[152,153,154...|[-243457.69885665...|[1.0,0.0]| 0.0|  
| 0.0|(692,[153,154,155...|[-260933.50931276...|[1.0,0.0]| 0.0|  
| 0.0|(692,[154,155,156...|[-220274.72552901...|[1.0,0.0]| 0.0|  
| 0.0|(692,[181,182,183...|[-154830.07125175...|[1.0,0.0]| 0.0|  
| 1.0|(692,[99,100,101,...|[-145978.24563975...|[0.0,1.0]| 1.0|  
| 1.0|(692,[100,101,102...|[-147916.32657832...|[0.0,1.0]| 1.0|  
| 1.0|(692,[123,124,125...|[-139663.27471685...|[0.0,1.0]| 1.0|  
| 1.0|(692,[124,125,126...|[-129013.44238751...|[0.0,1.0]| 1.0|  
| 1.0|(692,[125,126,127...|[-81829.799906049...|[0.0,1.0]| 1.0|  
+-----+-----+-----+-----+  
only showing top 20 rows  
  
>>> evaluator = MulticlassClassificationEvaluator(labelCol="label", predictionCo  
l="prediction", metricName="accuracy")  
>>> accuracy = evaluator.evaluate(predictions)  
>>> print("Test set accuracy = " + str(accuracy))  
Test set accuracy = 1.0  
>>>
```

Acknowledgement

1. Spark & Python: Working with RDDs (I): Jose A Dianes
2. Analisis Big Data: Putra Pandu A, Imam Cholisoddin
3. PySpark Dataframe Tutorial – PySpark Programming with Dataframes: Kurt
4. Running KMeans clustering on Spark: Don