

SKRIPSI

TEKNIK KLASIFIKASI INKREMENTAL BIG DATA PADA SISTEM TERSEBAR SPARK



Cindia Winarta

NPM: 2016730015

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN

«tahun»

UNDERGRADUATE THESIS

«JUDUL INGGRIIS»



Cindia Winarta

NPM: 2016730015

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY AND SCIENCES
PARAHYANGAN CATHOLIC UNIVERSITY

«tahun»

LEMBAR PENGESAHAN

TEKNIK KLASIFIKASI INKREMENTAL BIG DATA PADA SISTEM TERSEBAR SPARK

Cindia Winarta

NPM: 2016730015

Bandung, «tanggal» «bulan» «tahun»

Menyetujui,

Pembimbing Utama

Pembimbing Pendamping

Dr. Veronica Sri Moertini

Ketua Tim Penguji

Anggota Tim Penguji

«penguji 1»

«penguji 2»

Mengetahui,

Ketua Program Studi

Mariskha Tri Adithia, P.D.Eng

PERNYATAAN

Dengan ini saya yang bertandatangan di bawah ini menyatakan bahwa skripsi dengan judul:

TEKNIK KLASIFIKASI INKREMENTAL BIG DATA PADA SISTEM TERSEBAR SPARK

adalah benar-benar karya saya sendiri, dan saya tidak melakukan penjiplakan atau pengutipan dengan cara-cara yang tidak sesuai dengan etika keilmuan yang berlaku dalam masyarakat keilmuan.

Atas pernyataan ini, saya siap menanggung segala risiko dan sanksi yang dijatuhkan kepada saya, apabila di kemudian hari ditemukan adanya pelanggaran terhadap etika keilmuan dalam karya saya, atau jika ada tuntutan formal atau non-formal dari pihak lain berkaitan dengan keaslian karya saya ini.

Dinyatakan di Bandung,
Tanggal «tanggal» «bulan» «tahun»

Meterai Rp. 6000

Cindia Winarta
NPM: 2016730015

ABSTRAK

«Tuliskan abstrak anda di sini, dalam bahasa Indonesia»

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Kata-kata kunci: «Tuliskan di sini kata-kata kunci yang anda gunakan, dalam bahasa Indonesia»

ABSTRACT

«Tuliskan abstrak anda di sini, dalam bahasa Inggris»

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Keywords: «Tuliskan di sini kata-kata kunci yang anda gunakan, dalam bahasa Inggris»

«kepada siapa anda mempersembahkan skripsi ini...?»

KATA PENGANTAR

«Tuliskan kata pengantar dari anda di sini ...»

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Bandung, «bulan» «tahun»

Penulis

DAFTAR ISI

KATA PENGANTAR	xv
DAFTAR ISI	xvii
DAFTAR GAMBAR	xix
DAFTAR TABEL	xxi
1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Tujuan	2
1.4 Batasan Masalah	2
1.5 Metodologi	2
1.6 Sistematika Pembahasan	3
2 LANDASAN TEORI	5
2.1 Apache Spark	5
2.1.1 Fitur-fitur Apache Spark	5
2.1.2 Komponen-komponen Spark	5
2.1.3 Resilient Distributed Dataset	6
2.1.4 Arsitektur Spark	9
2.2 Scala	10
2.2.1 Pemrograman Dasar Scala	10
2.3 Klasifikasi	11
2.3.1 Teknik Klasifikasi di Spark MLlib	12
A KODE PROGRAM	17
B HASIL EKSPERIMEN	19

DAFTAR GAMBAR

B.1 Hasil 1	19
B.2 Hasil 2	19
B.3 Hasil 3	19
B.4 Hasil 4	19

DAFTAR TABEL

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Pada era sekarang ini, data merupakan sesuatu yang sangat penting dan sudah melekat pada kehidupan sehari-hari. Data adalah suatu fakta yang sudah diolah sedemikian rupa sehingga dapat dengan mudah digunakan dan dibaca oleh komputer. Hampir setiap kegiatan yang dilakukan pada era sekarang ini menghasilkan data. Contohnya adalah semua kegiatan yang dilakukan dengan teknologi sekarang terutama handphone. Kegiatan-kegiatan tersebut antara lain seperti *chatting*, telepon, deteksi gerak tubuh dengan handphone (sedang jalan, lari, sepeda, naik mobil, dan lain-lain), deteksi lokasi, dan masih banyak lagi. Selain itu contoh yang lainnya adalah kegiatan transaksi seperti membeli barang, mengambil uang, transfer uang, dan lainnya. Data-data tersebut selalu datang dan dikumpulkan secara terus-menerus. Data yang terkumpul dalam jumlah yang sangat besar dapat juga disebut *Big data*. *Big data* dapat berukuran sangat besar mulai dari ratusan *Gigabyte*, *Terabyte*, hingga *Petabyte*. *Big data* tersebut perlu diolah terlebih dahulu untuk menghasilkan informasi-informasi dan prediksi yang dibutuhkan. Salah satu proses pengolahan data yang digunakan untuk menghasilkan prediksi adalah proses klasifikasi.

Proses klasifikasi adalah proses untuk memprediksi label atribut kelas dari suatu data. Proses ini nantinya akan menghasilkan model klasifikasi yang dapat digunakan untuk memprediksi data-data baru. Data-data baru disini maksudnya adalah data yang belum diketahui label dari atribut kelasnya atau dengan kata lain belum diketahui suatu data ini masuk ke dalam kelompok kelas apa. Model klasifikasi data yang dilakukan harus diperbarui seiring dengan bertambahnya data agar tetap mendapatkan informasi dan prediksi yang tetap valid dan memiliki akurasi yang diharapkan. Untuk mendapatkan hasil prediksi dengan akurasi yang tinggi, digunakan teknik yang namanya Ensemble Method. *Ensemble method* merupakan teknik untuk meningkatkan akurasi pada teknik klasifikasi. Pada teknik ini, *dataset training* akan dipecah secara acak, tiap bagian digunakan untuk membuat model klasifikasi. *Dataset training* dipecah menggunakan teknik *bagging*. *Bagging* (*bootstrap aggregating*) merupakan teknik untuk membagi-bagi dataset dengan menerapkan ‘*sampling with replacement*’ yang dapat dimanfaatkan pada *ensemble method*. Pada tahap prediksi, kasus baru akan diumpungkan ke semua model. Hasil prediksi kelas akan ditentukan dari kelas dengan jumlah voting terbanyak.

Karena *big data* terus-menerus datang dan bertambah maka proses klasifikasi big data tentu sangat menghabiskan waktu dan *memory* yang tentu saja tidak dapat dilakukan seperti proses klasifikasi data biasa di dalam satu komputer. Oleh karena itu, proses pengolahan data dilakukan secara *cluster* yakni proses pengolahan data dilakukan menggunakan beberapa komputer sekaligus dan komputasi dilakukan secara paralel agar dapat dihasilkan hasil komputasi secara cepat. Maka dari itu, pada skripsi ini akan digunakan *Apache Spark*. *Apache Spark* adalah teknologi komputasi *cluster* yang dirancang untuk komputasi cepat yang berdasar pada *Hadoop MapReduce* dan model *MapReduce*. Algoritma klasifikasi paralel untuk klasifikasi *big data* pada sistem tersebar *Spark* sudah tersedia pada *Spark Machine Learning Library* (*Spark MLLib*). Akan tetapi algoritma tersebut masih perlu dikembangkan agar dapat menangani pembuatan model klasifikasi secara inkremental.

1.2 Rumusan Masalah

Berdasarkan penjelasan latar belakang pada bagian 1.1, rumusan masalah yang akan diselesaikan dalam skripsi ini adalah:

1. Bagaimana cara mengimplementasikan *ensemble method classifier* dengan *bagging* dan voting ke dalam algoritma klasifikasi spark MLlib?
2. Bagaimana menguji model klasifikasi dengan batch-batch data secara inkremental?

1.3 Tujuan

Berdasarkan rumusan masalah yang ada pada bagian 1.2, berikut adalah tujuan dari pembuatan skripsi ini:

1. Mengembangkan algoritma klasifikasi pada *Spark MLlib* menjadi *ensemble method classifier* dengan *bagging*.
2. Melakukan eksperimen dengan model klasifikasi untuk melihat apakah *batch-batch* data dapat diproses secara inkremental.

1.4 Batasan Masalah

Batasan masalah pada skripsi ini antara lain:

1. Skripsi ini hanya menggunakan algoritma klasifikasi yang ada pada *Spark Machine Learning Library* saja.
2. Skripsi ini menggunakan data yang sudah ada, jadi tidak ada proses pengumpulan data secara manual.

1.5 Metodologi

Metodologi yang digunakan dalam pembuatan skripsi ini adalah:

1. Melakukan studi literatur tentang Spark yang meliputi arsitektur Spark dan operasi-operasi *RDD(Resilient Distributed Dataset)*.
2. Melakukan studi literatur tentang penggunaan bahasa Scala.
3. Melakukan studi tentang teknik-teknik klasifikasi baik secara literatur maupun pada kelas pengantar data mining.
4. Melakukan eksperimen dengan membuat program *word count* untuk memahami bagaimana spark bekerja dengan menggunakan bahasa scala.
5. Mengimplementasikan algoritma-algoritma klasifikasi yang ada pada Spark MLlib dengan data kecil untuk memahami performa dan kinerja masing-masing algoritma.
6. Memilih salah satu algoritma klasifikasi dan merancang algoritma tersebut menjadi *ensemble method* dengan *bagging*.
7. Mencari batch-batch data yang dapat digunakan untuk menguji algoritma klasifikasi.
8. Mengimplementasikan algoritma klasifikasi yang sudah dirancang.

9. Menguji algoritma klasifikasi yang sudah dibuat dengan menggunakan batch-batch data secara inkremental.
10. Membuat dokumen skripsi.

1.6 Sistematika Pembahasan

Rencananya Bab 2 akan berisi petunjuk penggunaan template dan dasar-dasar L^AT_EX. Mungkin bab 3,4,5 dapt diisi oleh ketiga jurusan, misalnya peraturan dasar skripsi atau pedoman penulisan, tentu jika berkenan. Bab 6 akan diisi dengan kesimpulan, bahwa membuat template ini ternyata sungguh menghabiskan banyak waktu.

1. Bab 1 Pendahuluan

Bab ini menjelaskan tentang latar belakang, rumusah masalah, tujuan, batasan masalah, metodologi, dan sistematika pembahasan yang akan diselesaikan dan menjadi petunjuk dalam melakukan penelitian serta penyusunan dokumen skripsi ini.

2. Bab 2 Landasan Teori

Bab ini membahas mengenai teori-teori yang digunakan dalam penyusunan dokumen skripsi. Teori tersebut antara lain adalah penjelasan mengenai Spark, bahasa pemrograman Scala, dan juga algoritma-algoritma klasifikasi yang ada pada Spark MLlib.

3. Bab 3 Studi Eksplorasi Spark Bab ini berisikan mengenai eksplorasi yang dilakukan menggunakan Spark dengan bahasa Scala. Eksplorasi tersebut adalah dengan membuat program *word count* dengan menggunakan spark dan juga mengimplementasikan algoritma-algoritma klasifikasi yang ada pada Spark MLlib.

BAB 2

LANDASAN TEORI

2.1 Apache Spark

Rencananya akan diisi dengan penjelasan umum mengenai buku skripsi.

Apache spark adalah komputasi cluster cepat yang dirancang untuk komputasi cepat. Apache spark dibangun di atas Hadoop MapReduce dan meningkatkan model MapReduce agar dapat menggunakan lebih banyak jenis perhitungan secara efisien. Fitur utama spark adalah komputasi cluster di dalam memori yang meningkatkan kecepatan pemrosesan aplikasi. Spark dirancang untuk mencakup berbagai data yang besar dan berat seperti aplikasi batch, algoritma iteratif, query yang interaktif dan streaming.

2.1.1 Fitur-fitur Apache Spark

Apache Spark digunakan dikarenakan banyaknya kelebihan-kelebihan yang dimilikinya. Kelebihan tersebut antara lain.

1. Kecepatan
Spark membantu menjalankan aplikasi di cluster Hadoop, hingga 100 kali lebih cepat dalam memori, dan 10 kali lebih cepat saat dijalankan pada disk. Hal tersebut dapat terjadi dengan cara mengurangi jumlah operasi baca / tulis ke disk dan cara mengurangnya adalah dengan menyimpan pemrosesan data di dalam memori.
2. Kemampuan cache yang kuat.
3. Dapat dipasang di berbagai platform misal Mesos, Hadoop via Yarn atau cluster manager milik spark sendiri.
4. Menyediakan komputasi real-time.
5. Mendukung banyak bahasa (Java, Scala, dan Python).
6. Spark dapat menangani penambahan beban misalnya penambahan volume data atau jumlah pengguna tanpa penurunan kinerja yang berarti.

2.1.2 Komponen-komponen Spark

Komponen-komponen yang ada pada spark antara lain.

1. Apache Spark Core
Spark Core adalah komponen yang mendasari semua platform spark dengan semua fungsinya. Menyediakan komputasi dalam memory dan mereferensikan dataset-dataset di dalam sistem penyimpanan eksternal.

2. Spark SQL

Spark SQL adalah komponen diatas SparkCore yang menyediakan abstraksi data yang disebut SchemaRDD. SchemaRDD menyediakan bantuan untuk kueri-kueri data terstruktur dan data semi-terstruktur.

3. Spark Streaming

Spark Streaming memanfaatkan kemampuan penjadwalan cepat dari Spark Core untuk melakukan analisis streaming. Analisis streaming mengambil data dalam batch-batch kecil dan menjalankan transformasi RDD (Resilient Distributed Datasets) pada data di dalam batch-batch kecil tersebut.

4. MLlib(Machine Learning Library)

MLlib adalah framework machine learning terdistribusi diatas Spark dikarenakan arsitektur Spark yang berbasis memori terdistribusi. Spark MLlib menyediakan algoritma-algoritma yang dapat digunakan untuk analisis data seperti regresi, klasifikasi, dan lain-lain.

5. GraphX

GraphX adalah framework untuk pemrosesan grafik terdistribusi di atas spark. GraphX menyediakan API untuk membuat perhitungan grafik yang dapat memodelkan grafik yang ditentukan user.

6. SparkR

SparkR adalah package milik R yang menyediakan implementasi data frame terdistribusi. SparkR juga mendukung operasi seperti seleksi, filter, agregasi untuk dataset dataset yang ukurannya besar.

2.1.3 Resilient Distributed Dataset

RDD adalah bagian inti dari semua aplikasi Spark. Secara umum, RDD bersifat immutable. Immutable artinya objek tersebut tidak dapat diubah setelah dibuat, akan tetapi dapat ditransformasi. RDD artinya adalah

- Resilient : Toleran terhadap kesalahan dan mampu mengembalikan data yang gagal
- Distributed : Data terdistribusi ke semua node-node dalam sebuah cluster
- Dataset : Kumpulan partisi data dengan value

Jadi, dapat disimpulkan bahwa RDD adalah dataset yang tersebar ke semua node di dalam sebuah cluster dan data tersebut tahan/toleran terhadap kesalahan serta dapat mengembalikan data yang gagal.

Cara kerja RDD adalah data di dalam RDD dibagi menjadi beberapa bagian berdasarkan suatu key. RDD sangat resilient (highly resilient) karena RDD dapat pulih dengan cepat dari kesalahan-kesalahan. Hal tersebut dapat terjadi karena potongan data yang sama direplikasi di beberapa node pelaksana (executor nodes). Jadi, jika satu node pelaksana gagal maka yang lain masih akan memproses data. Oleh karena itu, RDD memungkinkan untuk melakukan perhitungan fungsi terhadap dataset dengan sangat cepat dengan memanfaatkan beberapa node.

Setiap dataset di RDD dibagi menjadi partisi yang logis, yang artinya dapat di komputasi/ dijalankan di node yang berbeda dalam sebuah cluster. Karena sifatnya ini, kita dapat melakukan transformasi atau aksi ke seluruh data secara paralel. Distribusinya dilakukan secara otomatis oleh Spark.

Ada 2 cara membuat RDD, cara pertama adalah dengan memparalelkan koleksi data yang ada di dalam driver program dan cara yang kedua adalah dengan mereferensi dataset di luar storage system seperti shared file system, HDFS, HBase, dan lain-lain.

RDD bisa melakukan 2 operasi sebagai berikut:

- Transformations (transformasi) : operasi yang dipakai untuk membuat RDD baru.
- Actions (aksi) : aksi dipakai di dalam RDD untuk memberitahukan Apache Spark untuk menggunakan komputasi dan memberikan hasilnya kembali ke driver.

Transformasi RDD

RDD dapat melakukan berbagai macam operasi transformasi, berikut adalah operasi-operasi transformasi yang dapat dilakukan oleh RDD.

No.	Transformasi	Arti
1	map(func)	Mengembalikan RDD baru, terbentuk dari memasukkan setiap elemen melalui fungsi func.
2	filter(func)	Mengembalikan dataset baru, terbentuk dari memilih elemen yang bernilai <i>true</i> dari fungsi func.
3	flatMap(func)	Mirip dengan map, tapi setiap <i>input item</i> dapat di map ke 0 atau lebih <i>output item</i> (jadi fungsi func harus mengembalikan sebuah <i>Sequence</i> daripada hanya sebuah <i>item</i>).
4	mapPartitions(func)	Mirip dengan map, tapi berjalan secara terpisah di setiap partisi(blok) dari RDD, jadi fungsi func harus bertipe $\text{Iterator}<T> \Rightarrow \text{Iterator}<U>$ saat berjalan dalam sebuah RDD tipe T.
5	mapPartitionsWithIndex(func)	Mirip dengan mapPartitions, akan tetapi juga menyediakan fungsi func dengan sebuah nilai integer representasi dari index dari partisi, jadi fungsi func harus bertipe $(\text{Int}, \text{Iterator}<T>) \Rightarrow \text{Iterator}<U>$ saat berjalan di sebuah RDD dengan tipe T.
6	sample(withReplacement, fraction, seed)	Membuat sampel dari sebagian (<i>fraction</i>) data, dengan atau tidak replacement, menggunakan <i>random number generator seed</i> yang diberikan.
7	union(otherDataset)	Mengembalikan dataset baru yang berisi gabungan (union) elemen dari dataset dan dataset dari parameter.
8	intersection(otherDataset)	Mengembalikan RDD baru yang berisi irisan (intersection) antara elemen dari dataset dan dataset dari parameter.
9	distinct([numTasks])	Mengembalikan dataset baru yang berisi elemen yang berbeda-beda dari dataset.
10	groupByKey([numTasks])	Saat dipanggil pada dataset dari pasangan (K,V), mengembalikan dataset dari pasangan (K, Iterable<V>).
11	reduceByKey(func, [numTasks])	Saat dipanggil pada dataset dari pasangan (K,V), mengembalikan dataset dari pasangan (K,V) dimana values dari setiap key diagregasi menggunakan fungsi reduce func yang diberikan di parameter, dimana harus bertipe $(V,V) \Rightarrow V$. Seperti pada groupByKey, jumlah dari tugas reduce dapat dikonfigurasi melalui parameter kedua yang opsional.

12	<code>aggregateByKey(zeroValue)(seqOp, combOp, [numTasks])</code>	Saat dipanggil pada dataset dari pasangan (K,V), mengembalikan dataset dari pasangan (K,U) dimana values dari setiap key diagregasi menggunakan fungsi penggabungan yang diberikan (combOP) dan sebuah value "nol" netral. Memungkinkan sebuah tipe value yang diagregasi yang berbeda dari tipe value input, sambil menghindari alokasi yang tidak perlu. Seperti pada <code>groupByKey</code> , jumlah dari tugas reduce dapat dikonfigurasi melalui parameter kedua yang opsional.
13	<code>sortByKey([ascending],[numTasks])</code>	Saat dipanggil pada dataset dari pasangan (K,V) dimana K mengimplementasikan Ordered, mengembalikan dataset dari pasangan (K,V) terurut berdasarkan keys secara menaik atau menurun, sesuai parameter Boolean <code>ascending</code> .
14	<code>join(otherDataset, [numTasks])</code>	Saat dipanggil pada dataset dari tipe (K,V) dan (K,W), mengembalikan dataset dari pasangan (K,(V,W)) dengan semua pasangan dari elemen tiap key. Outer joins didukung dengan <code>leftOuterJoin</code> , <code>rightOuterJoin</code> , dan <code>fullOuterJoin</code> .
15	<code>cogroup(otherDataset,[numTasks])</code>	Saat dipanggil pada dataset dari tipe (K,V) dan (K,W), mengembalikan sebuah dataset dari tuple (K,Iterable<V>,Iterable<W>)). Operasi ini juga disebut 'group With'.
16	<code>cartesian(otherDataset)</code>	Saat dipanggil pada dataset dari tipe T dan U, mengembalikan sebuah dataset dari pasangan (T,U) (semua pasangan elemen)
17	<code>pipe(command, [envVars])</code>	Pipe setiap partisi dari RDD melalui shell command, contohnya Perl atau bash script. Elemen RDD ditulis ke proses stdin dan baris output ke stdout dikembalikan sebagai RDD strings
18	<code>coalesce(numPartitions)</code>	Mengurangi jumlah partisi di dalam RDD hingga numPartitions. Berguna untuk menjalankan operasi secara lebih efisien setelah memfilter dataset berukuran besar
19	<code>repartition(numPartitions)</code>	Mengacak ulang data di dalam RDD secara acak untuk menciptakan antara lebih sedikit partisi dan menyeimbangkannya. Method ini selalu mengacak semua data di jaringan.
20	<code>repartitionAndSortWithinPartitions(numPartitions)</code>	Mengacak ulang RDD berdasarkan partitioner yang diberikan dan, dalam setiap partisi yang dihasilkan, record diurutkan berdasarkan key. Hal ini lebih efisien daripada memanggil <code>repartition</code> dan kemudian menurutkannya pada setiap partisi.

Aksi RDD

RDD juga dapat melakukan berbagai operasi aksi. Berikut adalah operasi-operasi aksi yang dapat dilakukan oleh RDD.

No	Action	Arti
1	reduce(func)	Mengagregasi elemen dataset menggunakan fungsi func (dimana func menerima dua argumen/parameter dan mengembalikan satu). Fungsi harus komutatif dan asosiatif supaya dapat dikomputasi secara paralel dengan benar.
2	collect()	Mengembalikan semua elemen dari dataset sebagai sebuah array dalam driver program. Aksi ini biasanya berguna setelah melakukan filter atau operasi lain yang mengembalikan subset kecil dari data.
3	count()	Mengembalikan jumlah elemen di dalam dataset.
4	first()	Mengembalikan elemen pertama dari dataset.
5	take(n)	Mengembalikan sebuah array dengan n elemen pertama dari dataset
6	takeSample (withReplacement,num, [seed])	Mengembalikan sebuah array dengan sampel acak dari banyak elemen dataset, dengan atau tidak replacement, menentukan random number generator seed secara optional.
7	takeOrdered(n, [ordering])	Mengembalikan n elemen pertama dari RDD menggunakan antara natural ordernya atau sebuah comparator buatan sendiri.
8	saveAsTextFile(path)	Menulis elemen dari dataset sebagai text file di dalam directory yang diberikan (pada parameter) di dalam file sistem lokal, dalam HDFS atau file sistem Hadoop yang lain. Spark memanggil toString pada setiap elemen untuk mengubah elemen dataset menjadi baris text di dalam file.
9	saveAsSequenceFile(path)	Menulis elemen dari dataset sebagai Hadoop SequenceFile di dalam path yang diberikan (pada parameter) dalam file sistem lokal, HDFS atau file sistem Hadoop yang lainnya. Aksi ini tersedia dalam pasangan key-value dari RDD yang mengimplementasikan Writable Interface milik Hadoop. Di Scala, aksi ini juga tersedia dalam tipe yang secara implisit dapat diubah ke Writable (Spark menyediakan konversi untuk tipe dasar seperti Int, Double, String, dll).
10	saveAsObjectFile(path)	Menulis elemen dari dataset dalam format sederhana menggunakan serialisasi Java, dimana nantinya bisa dimuat menggunakan SparkContext.objectFile().
11	countByKey()	Hanya tersedia pada RDD dari tipe (K,V). Mengembalikan sebuah hashmap dari pasangan (K, Int) dengan jumlah dari setiap key.
12	foreach(func)	Menjalankan fungsi func dalam setiap elemen di dataset. Hal ini biasanya dilakukan untuk efek samping dari suatu hal seperti mengupdate Accumulator atau berinteraksi dengan sistem penyimpanan eksternal.

2.1.4 Arsitektur Spark

Dalam spark, terdapat yang namanya master node dan juga worker node. Master node bertugas untuk manajemen pembagian RDD dan juga pembagian tugas-tugas ke semua worker node yang

ada. Master node memiliki sebuah driver program. Tugas dari driver program ini adalah untuk menjalankan aplikasi yang telah dibuat. Kode yang dibuat bertindak sebagai driver program atau jika menggunakan interactive shell, shell yang bertindak sebagai driver program.

Di dalam driver program, hal pertama yang harus dilakukan adalah membuat Spark Context. Spark Context merupakan gerbang ke semua fungsi-fungsi milik spark. Semua fungsi yang akan dilakukan pada driver program akan melewati Spark Context. Driver program dan spark context bertugas menangani eksekusi pekerjaan di dalam sebuah cluster. Sebuah pekerjaan dibagi-bagi menjadi beberapa tugas-tugas kecil yang nantinya didistribusikan ke worker node. RDD juga dibuat di dalam spark context dan juga didistribusikan ke berbagai worker node. Kemudian RDD akan dicache di dalam worker node tersebut.

Worker node adalah node pekerja yang pekerjaannya adalah pada dasarnya mengerjakan tugas-tugas yang diberikan kepadanya. Tugas-tugas tersebut dieksekusi di dalam node tersebut. Tugas tersebut adalah tugas di dalam RDD yang sudah dipartisi. Kemudian, worker node akan mengembalikan hasilnya kembali ke Spark Context.

Jadi secara garis besar, Spark Context mengambil pekerjaan, memecah pekerjaan dalam tugas-tugas/tasks dan mendistribusikannya ke worker node. Tugas-tugas tersebut bekerja pada RDD yang dipartisi, melakukan operasi, mengumpulkan hasil dan mengembalikannya ke Spark Context. Jika node worker ditambah, maka pekerjaan akan menjadi lebih cepat selesai. Hal ini terjadi karena pekerjaan dapat dipecah ke lebih banyak partisi dan dapat mengeksekusi pekerjaan tersebut secara paralel dalam banyak sistem yang berbeda. Besar memori juga akan bertambah yang berefek pada meningkatnya kekuatan mencache pekerjaan. Oleh karena itu, pekerjaan dapat dieksekusi dengan lebih cepat.

2.2 Scala

Mengapa menggunakan L^AT_EX untuk buku skripsi dan apa keunggulan/kerugiannya bagi mahasiswa dan pembuat template.

Scala adalah bahasa pemrograman yang berbasis Java Development Kit. Bahasa Scala menyediakan optimasi untuk kompleksitas kode dan concise notation (notasi singkat). Concise Notation artinya suatu kode memiliki banyak informasi dalam simbol atau tulisan yang sedikit yang hasilnya adalah kode yang pendek namun berbobot. Bahasa Scala juga kompatibel dengan bahasa Java. Oleh karena itu, Scala juga mendukung pemrograman berbasis objek (object-oriented) sama seperti Java. Selain itu, kompatibel dengan Java juga memungkinkan Scala untuk menggunakan keuntungan dari Java Virtual Machine (JVM) dan juga menggunakan library dari Java.

2.2.1 Pemrograman Dasar Scala

Bahasa Scala mirip dengan bahasa pemrograman Java yang digabungkan dengan bahasa pemrograman Python. Kemiripan dengan bahasa Java terletak pada inisialisasi kelas, library yang dipakai, adanya kelas main, mendukung object oriented, dan lain-lain. Sedangkan kemiripan dengan bahasa Python antara lain saat inisialisasi variabel, inisialisasi fungsi, tidak perlu semicolon(;) untuk pindah baris, dan definisi tipe data saat membuat variabel atau kembalian fungsi itu opsional. Berikut adalah sintaks-sintaks dasar dalam bahasa Scala.

No	Kode	Fungsi
1	<code>class NamaKelas(constructor:TipeData) { ... }</code>	Menginisialisasi kelas. Constructor dan TipeData opsional.
2	<code>new NamaKelas()</code>	Menginstansiasi kelas
3	<code>val namavalue: TipeData</code>	Menginisialisasi value (TipeData opsional). Value adalah atribut yang tidak dapat diubah (immutable).
4	<code>var namavariabel: TipeData</code>	Menginisialisasi variabel (TipeData opsional). Variabel adalah atribut yang dapat diubah (mutable).
5	<code>def namadef(param: TipeData): TipeKembalian { ... }</code>	Menginisialisasi fungsi. Parameter, TipeData, dan TipeKembalian adalah opsional.
6	<code>object NamaObject { ... }</code>	Menginisialisasi objek
7	<code>def main(args: Array[String]):Unit { ... }</code>	Meninisialisasi fungsi main
8	<code>if(ekspresiboolean1) { ... }else if(ekspresiboolean2) { ... } else { ... }</code>	Branching
9	<code>while(ekspresiboolean) { ... }</code>	Looping dengan while
10	<code>for(i<-0 until max by inc) { ... }</code>	Looping dengan for dari 0 hingga max. inc adalah increment dari i.

2.3 Klasifikasi

Di dalam KBBI, klasifikasi adalah penyusunan bersistem dalam kelompok atau golongan menurut kaidah atau standar yang ditetapkan. Klasifikasi adalah pembagian sesuatu menurut kelas-kelas. Klasifikasi dilakukan untuk memprediksi suatu objek dengan atribut-atribut tertentu berada di dalam kelompok kelas yang mana. Cara klasifikasi ada bermacam-macam dan dengan akurasi yang bermacam-macam pula. Karena akurasi yang bermacam-macam, maka ada metode-metode yang tujuannya adalah meningkatkan akurasi dari klasifikasi. Salah satu caranya adalah dengan metode ensemble.

Ensemble Method atau metode ensemble adalah algoritma dalam machine learning dimana algoritma ini adalah algoritma pencarian solusi prediksi terbaik dibandingkan dengan algoritma yang lain. Metode ensemble ini menggunakan beberapa algoritma pembelajaran untuk pencapaian solusi prediksi yang lebih baik daripada algoritma yang bisa diperoleh dari salah satu pembelajaran algoritma saja. Cara kerja dari metode ini adalah dengan memilih sebagian data dari data train secara acak dan membuat model dengan data train yang dipilih tersebut. Proses membuat model tersebut dilakukan berkali-kali dengan terus memilih data train secara acak. Biasanya proses tersebut dilakukan 500 sampai 1000 kali hingga terbentuk 500 hingga 1000 model klasifikasi. Kemudian, record yang ingin diklasifikasi dites disemua model. Proses ini menggunakan teknik bagging.

Bootstrap Aggregating atau bagging merupakan metode yang dapat memperbaiki hasil dari algoritma klasifikasi machine learning dengan menggabungkan klasifikasi prediksi dari beberapa model. Hasil klasifikasi dari beberapa model tersebut kemudian dihitung secara voting. Hasil voting kelas terbanyak akan menjadi label dari record yang ingin diprediksi. Tujuan dari penggabungan hasil prediksi dari beberapa model adalah untuk mengatasi ketidakstabilan pada model yang kompleks. Bagging adalah salah satu algoritma berbasis ensemble yang paling awal dan paling sederhana, namun efektif.

2.3.1 Teknik Klasifikasi di Spark MLlib

Akan dipaparkan bagaimana menggunakan template ini, termasuk petunjuk singkat membuat referensi, gambar dan tabel. Juga hal-hal lain yang belum terpikir sampai saat ini. Ada banyak algoritma untuk mengimplementasikan proses klasifikasi. Diantara banyak algoritma tersebut, algoritma yang terdapat pada Spark Machine Learning Library hanya ada 5 dan 2 diantaranya sudah mengaplikasikan metode ensemble dari 1 algoritma dasar yakni decision tree dengan metode ensemblenya yakni random forest dan gradient boosted tree. Hal ini meninggalkan hanya 2 algoritma yang dapat dipilih untuk diaplikasikan metode ensemble yakni algoritma logistic regression dan naive bayes.

Logistic Regression

Decision Tree

Naive Bayes

Random Forest

Gradient Boosted Tree

Hasil Eksperimen Mandiri Instalasi Scala Instalasi Hadoop Instalasi Spark Instalasi IntelliJ
Contoh program HelloWorld dengan Scala Contoh program MapReduce dengan Scala

Analisis

Kesimpulan

LAMPIRAN A

KODE PROGRAM

Listing A.1: MyCode.c

```
1 // This does not make algorithmic sense,
2 // but it shows off significant programming characters.
3
4 #include<stdio.h>
5
6 void myFunction( int input, float* output ) {
7     switch ( array[i] ) {
8         case 1: // This is silly code
9             if ( a >= 0 || b <= 3 && c != x )
10                 *output += 0.005 + 20050;
11             char = 'g';
12             b = 2^n + ~right_size - leftSize * MAX_SIZE;
13             c = (--aaa + &daa) / (bbb++ - ccc % 2 );
14             strcpy(a,"hello_$@?");
15         }
16         count = ~mask | 0x00FF00AA;
17     }
18 }
19
20 // Fonts for Displaying Program Code in LATEX
21 // Adrian P. Robson, nepsweb.co.uk
22 // 8 October 2012
23 // http://nepsweb.co.uk/docs/progfonts.pdf
```

Listing A.2: MyCode.java

```
1 import java.util.ArrayList;
2 import java.util.Collections;
3 import java.util.HashSet;
4
5 //class for set of vertices close to furthest edge
6 public class MyFurSet {
7     protected int id; //id of the set
8     protected MyEdge FurthestEdge; //the furthest edge
9     protected HashSet<MyVertex> set; //set of vertices close to furthest edge
10    protected ArrayList<ArrayList<Integer>> ordered; //list of all vertices in the set for each trajectory
11    protected ArrayList<Integer> closeID; //store the ID of all vertices
12    protected ArrayList<Double> closeDist; //store the distance of all vertices
13    protected int totaltrj; //total trajectories in the set
14
15    /*
16     * Constructor
17     * @param id : id of the set
18     * @param totaltrj : total number of trajectories in the set
19     * @param FurthestEdge : the furthest edge
20     */
21    public MyFurSet(int id,int totaltrj,MyEdge FurthestEdge) {
22        this.id = id;
23        this.totaltrj = totaltrj;
24        this.FurthestEdge = FurthestEdge;
25        set = new HashSet<MyVertex>();
26        ordered = new ArrayList<ArrayList<Integer>>();
27        for (int i=0;i<totaltrj;i++) ordered.add(new ArrayList<Integer>());
28        closeID = new ArrayList<Integer>(totaltrj);
29        closeDist = new ArrayList<Double>(totaltrj);
30        for (int i = 0;i <totaltrj;i++) {
31            closeID.add(-1);
32            closeDist.add(Double.MAX_VALUE);
33        }
34    }
35
36 }
```


LAMPIRAN B

HASIL EKSPERIMEN

Hasil eksperimen berikut dibuat dengan menggunakan TIKZPICTURE (bukan hasil excel yg diubah ke file bitmap). Sangat berguna jika ingin menampilkan tabel (yang kuantitasnya sangat banyak) yang datanya dihasilkan dari program komputer.



Gambar B.1: Hasil 1



Gambar B.2: Hasil 2



Gambar B.3: Hasil 3



Gambar B.4: Hasil 4