Teknik Klasifikasi Inkremental Big Data pada Sistem Tersebar Spark

Cindia Winarta–2016730015

1 Data Skripsi

Pembimbing utama/tunggal: **Veronica Sri Moertini**
Pembimbing pendamping: -
Kode Topik : **VSM4701**
Topik ini sudah dikerjakan selama : **1** semester
Pengambilan pertama kali topik ini pada : Semester **47 - Ganjil 19/20**
Pengambilan pertama kali topik ini di kuliah : **Skripsi 1**
Tipe Laporan : **B -** Dokumen untuk reviewer pada presentasi dan **review Skripsi 1**

2 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 semedikian rupa sehingga dapat dengan mudah digunakan dan dibaca oleh komputer. Hampir setiap kegiatan yang dilakukan pada era sekarang ini menghasilkan data. Contohnya adalah kegiatan transaksi seperti membeli barang, mengambil uang, transfer uang, dan lain-lain. 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. Selain ukurannya yang sangat besar, kecepatan pertambahan datanya juga sangat cepat sehingga *Big data* perlu diolah terlebih dahulu secara khusus untuk menghasilkan informasi-informasi dan prediksi yang bernilai. Salah satu proses pengolahan data yang dapat digunakan untuk menghasilkan prediksi adalah proses klasifikasi.

Proses klasifikasi adalah proses untuk memprediksi label atibut kelas dari suatu data. Proses ini nantinya akan menghasilkan model klasifikasi yang dapat digunakan untuk memprediksi data-data baru. Dikarenakan data-data baru yang terus-menerus datang, maka model klasifikasi harus terus-menerus diperbarui agar tetap mendapatkan informasi dan prediksi yang tetap valid dan memiliki akurasi yang diharapkan. Untuk mendapatkan hasil prediksi dari data yang terus-menerus datang tersebut atau secara inkremental, dapat 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 diumpankan ke semua model. Hasil prediksi kelas akan ditentukan dari kelas dengan jumlah voting terbanyak.

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*. Spark digunakan dalam penyusunan skripsi ini karena sifatnya yang melakukan pemrosesan di dalam memori. Sifat ini sangat dibutuhkan oleh algoritma klasifikasi inkremental yang memiliki operasi-operasi iteratif. 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.

3 Rumusan Masalah

Berdasarkan penjelasan latar belakang pada bagian 2, rumusan masalah yang akan diselesaikan dalam skripsi ini adalah:

1. Bagaimana cara mengimplementasikan *ensemble method classifier* dengan *bagging* voting untuk pembuatan algoritma klasifikasi inkremental?

2. Bagaimana performa model klasifikasi yang telah dibuat dengan data yang inkremental?

4 Tujuan

Berdasarkan rumusan masalah yang ada pada bagian 3, berikut adalah tujuan dari pembuatan skripsi ini:

1. Mengembangkan algoritma klasifikasi pada *Spark MLLib* menjadi *ensemble method classifier* supaya dapat digunakan secara inkremental.

2. Melakukan eksperimen dengan model klasifikasi yang telah dibuat dengan mengklasifikasikan data yang inkremental.

5 Detail Perkembangan Pengerjaan Skripsi

Detail bagian pekerjaan skripsi sesuai dengan rencan kerja/laporan perkembangan terkahir :

1. **Melakukan studi literatur Spark dan cara menggunakannya.**
   Status : Ada sejak rencana kerja skripsi.
   Hasil : Apache spark adalah komputasi cluster cepat yang dirancang untuk komputasi cepat. Komputasi cepat disini dikarenakan Apache spark dibangun di atas Hadoop MapReduce dan Spark meningkatkan model MapReduce yang sudah ada agar dapat menggunakan lebih banyak jenis perhitungan secara efisien. Sama seperti Hadoop, Spark dirancang untuk memproses Big Data. Spark dapat memproses berbagai data yang besar dan berat seperti aplikasi batch, algoritma iteratif, query yang interaktif dan streaming. Yang membedakan Spark dan Hadoop adalah fitur utama Spark dimana Spark menjalankan komputasi cluster di dalam memori sehingga dapat meningkatkan kecepatan pemrosesan aplikasi.

   Spark memiliki banyak komponen-komponen yang mendukung pemrosesan *big data*. Satu komponen inti dari Spark yang harus ada dalam setiap program Spark adalah Spark Core. Komponen-komponen yang lain hanya komponen pendukung Spark yang sifatnya opsional(dapat dipakai maupun tidak), akan tetapi tentu saja juga mempermudah proses komputasi big data. Berikut adalah penjelasan singkat mengenai komponen-komponen yang ada pada Spark.

   (a) Apache Spark Core
   Spark Core adalah komponen yang mendasari semua platform spark dengan semua fungsi-fungsinya. Dengan kata lain, Spark Core adalah komponen yang harus ada jika ingin menggunakan fungsi-fungsi yang ada pada Spark. Spark Core menyediakan komputasi dalam memory dan mereferensikan dataset-dataset di dalam sistem penyimpanan eksternal.

   (b) 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.

(c) 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.

(d) 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.

(e) 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.

(f) 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.

Reselient Distributed Dataset.
RDD adalah bagian inti dari semua aplikasi Spark. Dataset-dataset yang ingin diproses menggunakan Spark akan dibaca melalui Spark Context sebagai objek RDD. RDD adalah dataset yang tersebar ke semua node di dalam sebuah cluster dan data-data tersebut tahan atau toleran terhadap kesalahan dan kerusakan serta dapat mengembalikan data yang gagal diproses. Secara umum, RDD bersifat immutable. Immutable artinya objek tersebut tidak dapat diubah setelah dibuat, akan tetapi objek tersebut dapat ditransformasi serta melakukan aksi.

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. Cara kerja RDD adalah dataset di dalam RDD dibagi menjadi beberapa bagian berdasarkan suatu key. Bagian-bagian data tersebut kemudian akan dibagikan ke seluruh node-node pekerja. Karena itu, RDD memungkinkan untuk melakukan perhitungan fungsi terhadap dataset dengan sangat cepat dengan memanfaatkan beberapa node pekerja. Satu node pekerja dapat menyimpan lebih dari satu bagian data. Selain itu, bagian data yang sama dapat direplikasi di beberapa node pekerja. Jadi, jika satu node pelaksana gagal atau mengalami kesalahan (error) maka yang node yang lain masih bisa memproses data. Oleh karena itulah, RDD disebut dan bersifat sangat *resilient* dimana RDD kebal terhadap kegagalan atau kesalahan dan dapat pulih dengan cepat dari kesalahan.

Setiap dataset di dalam RDD dibagi menjadi partisi yang logis, yang artinya dapat di komputasi atau 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. RDD bisa melakukan 2 operasi yakni transformasi dan aksi. Transformasi merupakan operasi RDD yang digunakan untuk membuat RDD baru dari RDD yang sudah ada. Contohnya adalah operasi map, filter, dan masih banyak lagi. Operasi yang kedua adalah aksi. Aksi dipakai dalam RDD untuk memberitahukan Spark untuk menggunakan komputasi dan memberikan hasilnya kembali ke driver. Contoh dari operasi ini adalah menuliskan RDD hasil transformasi ke dalam file text(.txt) ke driver komputer atau mengembalikan hanya sebuah value.

Transformasi RDD

RDD dapat melakukan berbagai macam operasi transformasi. Semua operasi transformasi yang dilakukan oleh objek RDD akan menghasilkan RDD baru. Berikut adalah operasi-operasi tranformasi yang dapat dilakukan oleh objek 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 *true* dari fungsi func.
3	flatMap(func)	Mirip dengan map, tapi setiap *input item* dapat di map ke 0 atau lebih *output item* (jadi fungsi func harus mengembalikan sebuah *Sequence* daripada hanya sebuah *item*.
4	mapPartitions(func)	Mirip dengan map, tapi berjalan secara terpisah di setiap partisi(blok) dari RDD, jadi fungsi func harus bertipe Iterator<T> => 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 (Int, Iterator<T>) => Iterator<U> saat berjalan di sebuah RDD dengan tipe T.
6	sample(withReplacement, fraction, seed)	Membuat sampel dari sebagian (*fraction*) data, dengan atau tidak replacement, menggunakan *random number generator seed* 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)=> V. Seperti pada groupByKey, jumlah dari tugas reduce dapat dikonfigurasi melalui parameter kedua yang opsional.
12	aggregateByKey(zeroValue) (seqOp, combOp, [numTasks])	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 groupByKey, jumlah dari tugas reduce dapat dikonfigurasi melalui parameter kedua yang opsional.

13	sortByKey([ascending], [numTasks])	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 ascending.
14	join(otherDataset,[numTasks])	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 leftOuterJoin, rightOuterJoin, dan fullOuterJoin.
15	cogroup(otherDataset, [numTasks])	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	cartesian(otherDataset)	Saat dipanggil pada dataset dari tipe T dan U, mengembalikan sebuah dataset dari pasangan (T,U) (semua pasangan elemen)
17	pipe(command, [envVars])	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	coalesce(numPartitions)	Mengurangi jumlah partisi di dalam RDD hingga numPartitions. Berguna untuk menjalankan operasi secara lebih efisien setelah memfilter dataset berukuran besar
19	repartition(numPartitions)	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	repartitionAndSortWithin Partitions(partitioner)	Mempartisi ulang RDD berdasarkan partitioner yang diberikan dan, dalam setiap partisi yang dihasilkan, rekord diurutkan berdasarkan key. Hal ini lebih efisien daripada memanggil repartition dan kemudian menurutkannya pada setiap partisi.

Aksi RDD

Selain transformasi, RDD juga dapat melakukan berbagai operasi aksi. Semua aksi RDD akan mengembalikan value bukan RDD. 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 mengunakan 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 sepeti mengupdate Accumulator atau berinteraksi dengan sistem penyimpanan eksternal.

Arsitektur Spark

Dalam spark, terdapat yang namanya master node dan juga worker node. Master node bertugas untuk memanajemen 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. **Mempelajari bahasa pemrograman Scala.**
   Status : Ada sejak rencana kerja skripsi.
   Hasil :

   Scala atau singkatan dari Scalable Language 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, berkompatibel dengan Java juga memungkinkan Scala untuk menggunakan keuntungan dari Java Virtual Machine (JVM) dan juga menggunakan library dari Java. Oleh karena itu, 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 yang optional.

3. **Mempelajari algoritma klasifikasi paralel pada Spark yang akan dikembangkan.**
   Status : Ada sejak rencana kerja skripsi.
   Hasil :

   Klasifikasi merupakan salah satu pembelajaran Machine Learning selain regresi dan clustering. Di dalam KBBI, klasifikasi adalah penyusunan bersistem dalam kelompok atau golongan menurut kaidah atau standar yang ditetapkan. Secara umum, klasifikasi adalah proses pembagian data menurut label kelas-kelas tertentu. Klasifikasi menyelesaikan permasalahan prediksi label kelas yang sifatnya diskret berbeda dengan proses regresi yang memprediksi label yang sifatnya kontinu. Teknik klasifikasi ada bermacam-macam dan dengan akurasi yang bermacam-macam pula. Dalam Spark, terdapat Spark Machine Learning Library yang sudah menyediakan pembelajaran Machine Learning secara terdistribusi. Teknik klasifikasi multinomial yang ada dalam Spark MLLib ada 4 yakni Logistic Regression, Decision Tree, Naive Bayes, dan Random Forest.

   (a) Logistic Regression
   Logistic Regression atau regresi logistik adalah teknik regresi yang berbeda dengan teknik regresi yang lainnya karena teknik ini digunakan untuk proses klasifikasi. Model dari regresi logistik terbentuk dari persamaan regresi yang ditambahkan dengan fungsi sigmoid sehingga memungkinkan untuk melakukan klasifikasi. Cara prediksi dengan menggunakan model ini adalah rekord baru dimasukan ke model ini. Kemudian dilihat batas keputusan untuk memutuskan bahwa rekord baru masuk ke kelas apa.

   (b) Decision Tree Classifier
   Decision Tree atau pohon keputusan adalah teknik klasifikasi yang menggunakan pohon keputusan sebagai modelnya. Pembuatan model pohon keputusan memerlukan information gain untuk

menentukan fitur apa yang cocok menjadi node-node internal. Semakin besar Information Gain dari suatu fitur maka semakin banyak pula informasi yang bisa didapat dari fitur tersebut yang artinya fitur tersebut cocok untuk menjadi node internal. Jadi, pemilihan fitur suatu model pohon keputusan dipilih berdasarkan fitur yang information gainnya maksimal.

(c) Naive Bayes
Naive Bayes adalah teknik klasifikasi yang berbasis konsep probabilitas dan Teorema Bayesian dengan asumsi bahwa setiap nilai variable/atribut (prediktor) bersifat bebas (independence). Spark MLlib hanya mendukung 2 klasifikasi Naive Bayes yakni Bernoulli Naive Bayes dan Multinomial Naive Bayes. Bernoulli Naive Bayes dipakai apabila dataset memiliki fitur yang sifatnya biner yakni hanya memiliki nilai 0 atau 1. Multinomial Naive Bayes dipakai apabila fitur dataset memiliki sifat diskrit contohnya rating film 1-5 bintang.

(d) Random Forest
Random Forest adalah teknik klasifikasi ensemble dari Decision Tree. Algoritma Random Forest membuat model pohon keputusan sebanyak batasan yang diberikan. Disebut random karena algoritma Random Forest memilih data untuk dijadikan model pohon keputusan secara acak dari data train yang ada. Rekord baru diprediksi dengan menumpankan rekord baru tersebut ke semua model pohon keputusan yang ada. Kemudian, hasil klasifikasi dari masing-masing model pohon divoting dan kelas dengan jumlah voting terbanyak yang akan menjadi kelas dari rekord baru tersebut.

Algoritma-algoritma tersebut masih perlu dikembangkan agar dapat menangani pembuatan model klasifikasi secara inkremental. Salah satu cara untuk mengembangkan algoritma klasifikasi secara inkremental adalah dengan menggunakan Ensemble Method. 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, rekord 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 rekord 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.

4. **Mempelajari source code dari algoritma klasifikasi yang dipilih.**
Status : Ada sejak rencana kerja skripsi.
Hasil :

Berikut adalah source code beserta penjelasan dari algoritma klasifikasi. Untuk bagian ini, hanya dijelaskan teknik klasifikasi dengan menggunakan algoritma decision tree dan random forest beserta algoritma Naive Bayes.

5. **Merancang pengembangan algoritma pada yang dipilih agar menjadi *ensemble method classifier* yang inkremental, dengan menambahkan teknik *bagging* dan voting.**

Status : Ada sejak rencana kerja skripsi.
Hasil : Belum diimplementasikan

6. **Menulis dokumen skripsi**
   Status : Ada sejak rencana kerja skripsi.
   Hasil : Bab 1 dan 2

6 Pencapaian Rencana Kerja

Langkah-langkah kerja yang berhasil diselesaikan dalam Skripsi 1 ini adalah sebagai berikut:

1. Melakukan studi literatur *Spark* dan cara menggunakannya.
2. Mempelajari bahasa pemrograman *Scala*.
3. Mempelajari algoritma klasifikasi pada *Spark*.
4. Mempelajari *source code* dari algoritma klasifikasi yang dipilih.

7 Kendala yang Dihadapi

Kendala - kendala yang dihadapi selama mengerjakan skripsi :

- Adanya kesalah pahaman mengenai materi yang harusnya dipelajari.
- Terlalu banyak godaan berupa hiburan (game, film, dll).
- Skripsi diambil bersamaan dengan kuliah-kuliah lain yang memberikan banyak tugas.

Bandung, 19/11/2019

Cindia Winarta

Menyetujui,

Nama: Veronica Sri Moertini
Pembimbing Tunggal