Prediksi Retensi Customer Berdasarkan *Support Vector Machine* dengan *Preprocessing* Menggunakan *Hadoop*

Giovanni Gabriela Gunadi, Silvia Rostianingsih, Andre Gunawan
Program Studi Informatika Fakultas Teknologi Industri Universitas Kristen Petra
Jl. Siwalankerto 121 – 131 Surabaya 60236
Telp. (031) – 2983455, Fax. (031) – 8417658
gabbygunadi31@gmail.com, silvia@petra.ac.id, andre.gunawan@petra.ac.id

ABSTRAK

Saat ini, banyak sekali perusahaan yang sering memberikan penawaran berupa diskon atau promo kepada *customer*. Tujuannya adalah agar perusahaan dapat menarik *customer* baru. Namun penawaran yang diberikan tidak dapat dilakukan secara acak, melainkan harus tertuju kepada *customer* yang sesuai. Hal ini agar perusahaan tidak perlu mengeluarkan terlalu banyak biaya.
Penelitian ini akan melakukan prediksi retensi *customer* dengan menggunakan metode SVM (*Support Vector Machine*). Data yang digunakan yaitu *Acquire Valued Shopper Challenge* yang diambil dari *website* Kaggle. Sebagaimana program yang digunakan yaitu java dengan *framework* Hadoop untuk proses *preprocessing* dan pembuatan *feature* serta *python* untuk implementasi SVM.

Pengujian dilakukan sebanyak dua kali dengan menggunakan data yang berbeda. Objektif yang ingin dicapai dari pengujian adalah agar model yang dibuat dapat memprediksi customer yang akan melakukan retensi apabila mendapatkan *offer*. Dengan pengujian yang dilakukan, hasil terbaik untuk menjawab objektif adalah menggunakan data pada percobaan kedua dengan rasio data *training* terhadap data *testing* adalah 7:3 dan *gamma* 0.1 dengan akurasi yang diperoleh adalah 58,21%.

Kata Kunci: Retensi Customer, Hadoop, Support Vector Machine

ABSTRACT

Nowadays, many companies often offer a discount to customer. The goal is that the company can attract new customers. However, the offer given cannot be done randomly. Offers must be directed to the right customers so the company doesn't need to spend much money. This study will analyze to determine whether there is an influence of the association rule on customer retention. The data used is the Acquire Valued Shopper Challenge taken from the Kaggle website. As the program used is java with the Hadoop framework that uses the association rule method.
Tests carried out twice using different data. The objective to be achieved from testing is that the model created can predict customers who will retain if they get an offer. With the tests carried out, the best results to answer the objective is to use data in the second experiment with the ratio of training data to testing data is 7: 3 and gamma 0.1 with accuracy obtained is 58.21%.

Keywords: *Customer Retention, Hadoop, Support Vector Machine*

1. INTRODUCTION

Kebanyakan perusahaan sering memberikan penawaran produk berupa diskon atau promo lainnya kepada pembeli. Hal ini berguna agar perusahaan dapat menarik pembeli baru untuk membeli produknya. Namun, tentu saja perusahaan tidak dapat secara acak dalam memberikan penawaran kepada pembeli. Perusahaan harus memperhitungkan calon pembeli mana yang dapat memberikan keuntungan bagi perusahaan. Hal ini agar perusahaan tidak menghabiskan terlalu banyak biaya serta *offer* yang diberikan dapat memberikan dampak. Salah satu hal yang dapat dibilang menjadi penentu utama dari profitabilitas perusahaan adalah loyalitas dari para pembelinya. Loyalitas *customer* dapat diperhitungkan dari retensi *customer* dalam membeli suatu produk. Namun, terdapat beberapa kendala agar dapat mencari pelanggan mana yang loyal dan sebaiknya diberikan *offer* tertentu.

Skripsi ini dibuat untuk melakukan prediksi retensi *customer* dengan menggunakan metode *Support Vector Machine* (SVM). Data yang digunakan adalah data *acquired valued shopper challenge* yang diambil dari *website Kaggle*. Dataset ini berisi data *transaction*, *offer*, dan *history* dimana seluruh *field* dianonim, tipe data *numeric*,dan berukuran kurang lebih 22 GB. Diharapkan dengan adanya skripsi ini dapat membantu perusahaan dalam menentukan *customer* mana yang diberikan penawaran sehingga dapat meberikan keuntungan bagi perusahaan.

2. DASAR TEORI

2.1 Data Preprocessing

Data *raw* yang didapatkan biasanya belum bersih. Masalah yang ada di dalam data adalah :

- Tidak lengkap : terdapat atribut yang tidak memiliki nilai (*NULL*)
- Memiliki *noise* : memiliki *error* atau data yang asing dari data yang lain
- Tidak konsisten : memiliki perbedaan pengkodean di beberapa data

Masalah-masalah ini dapat menimbulkan hasil yang tidak konsisten pada analisis data. Oleh karena itu perlu dilakukan *preprocessing data* dimana data yang raw akan dilakukan data*cleansing* sehingga dapat menyelesaikan masalah-masalah tersebut.

2.2 Support Vector Machine

Support Vector Machine dikembangkan pertama kali oleh Boser, Guyin, Vapnik dan pertama kali dipresentasikan pada tahun 1992. Konsep dasar SVM sebenarnya merupakan kombinasi harmonis dari teori komputasi yang telah ada sebelumnya. Secara sederhana, konsep SVM dapat dijelaskan sebagai usaha mencari *hyperlane* terbaik yang berfungsi sebagai pemisah dua buah *class* pada *input space*. Prinsip dasar SVM adalah *linear classifier* yang kemudian dikembangkan agar dapat bekerja pada problem *non-linear* dengan memasukkan konsep *kernel*. Pada umumnya, dua buah *class* pada *input space* tidak dapat terpisah secar sempurna. Hal ini menyebabkan optimasi tidak dapat dilakukan. Untuk mengatasi masalah ini, SVM dirumuskan ulang dengan memperkenalkan teknik *softmargin*. Nilai C dipilih untuk mengontrol *tradeoff*. Nilai C yang besar akan memberikan *penalty* yang lebih besar terhadap

error klasifikasi. Nilai C yang kecil akan memberikan *margin* yang lebih luas dengan *cost* ada misklasifikasi. Nilai C yang semakin besar akan memberikan *margin* klasifikasi yang sulit dan toleransi nol terhadap pelanggaran. Untuk dapat menemukan nilai C yang tepat, maka dapat menggunakan *grid search* dengan *cross validation* [3]

[5] SVM memiliki beberapa *kernel*, yaitu :

- *Linear kernel* dimana *kernel* ini merupakan *kernel* yang paling sederhana diantara yang lain.
- *Polynomial kernel* dimana *kernel* ini memiliki dua parameter berupa C dan *degree*. *Polynomial* dengan *degree* satu sama dengan *linear kernel*. Semakin tinggi nilai *degree* yang diberikan maka akan semakin kompleks batasnya dan dapat menyebabkan *overfitting*.

RBF *kernel* dimana pada *kernel* ini memiliki parameter gamma. Ketika gamma yang diberikan terlalu kecil, maka model akan berperilaku seperti *linear* SVM. Namun ketika gamma yang diberikan besar, maka akan memberikan dampak *overfitting*.

2.3 Big Data

Big Data merupakan suatu istilah yang menggambarkan voulme data yang besar [6]. *Big Data* menjadi suatu teknologi baru yang digunakan karena data yang digunakan besar dan banyak sehingga tidak dapat dikelola menggunakan teknik konvensional. Konsep *Big Data* dapat dibagi menjadi tiga aspek, yaitu [7] :

- *Volume :* Organisasi mengumpulkan data dari berbagai macam sumber, termasuk transaksi bisnis, sosial media, dan informasi dari sensor atau data dari mesin ke mesin.
- *Velocity :* Berapa lama waktu yang dibutuhkan untuk memproses *Big Data* tersebut. Dalam hal ini, proses yang dilakukan nantinya harus diefisiensikan.
- *Variety :* Tipe data yang digunakan dalam *Big Data* beragam dan dapat berupa data terstruktur maupun tidak terstruktur.

Menurut SAS, terdapat dua dimensi tambahan pada *big data*, yaitu

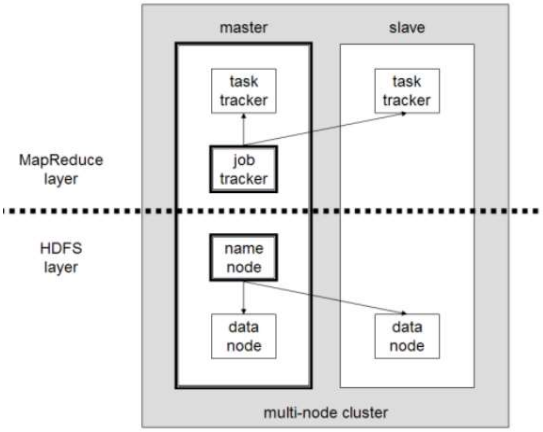
- *Variability :* Selain meningkatnya *velocities* dan *varieties* data, aliran data dapat sangat tidak konsisten dengan puncak periodik. Beban data puncak harian, musiman, dan yang dipicu oleh peristiwa dapat menjadi tantangan untuk dikelola. Terlebih lagi dengan data yang tidak terstruktur.
- *Complexity :* Data dapat berasal dari berbagai sumber sehingga membuat sulit untuk di proses. Namun, penting untuk dapat menghubungkan hubungan, hierarki, dan beberapa tautan data sheingga tidak cepat lepas kendali.

2.4 Apache Hadoop

Hadoop merupakan sebuah *framework* yang memungkinkan untuk melakukan *distributed processing* pada suatu *dataset* yang besar diberbagai *cluster* yang terpisah. Hadoop bekerja dalam *environment* Java. Arsitektur dari Hadoop dapat dilihat pada Gambar 1.

Menurut Maitrey & Jha [9], *Hadoop* dapat menjalankan 3 macam cara, yaitu :

- *Standalone mode* : *default mode* yang disediakan oleh *Hadoop* dimana semua dijalankan melalui sebuah proses *Java*.
- *Pseudo – Distributed mode* : *Hadoop* dikonfigurasi untuk berjalan pada sebuah mesin, akan tetapi dengan beberapa *Hadoop Daemon* yang berjalan sebagai proses *Java* yang berbeda.
- *Fully distributed* atau *cluster mode* : sebuah mesin pada sebuah *cluster* yang dilabel *Namenode* dan yang lain sebagai *JobTracker*. Hanya satu *Namenode* yang diletakan dalam sebuah *cluster*. *Namenode* ini bertugas mengatur *namespace*, metadata *FileSystem*, dan kontrol akses. *SecondaryNamenode* dapat diletakkan untuk proses *handshaking* secara periodik dengan *Namenode* untuk toleransi kegagalan. Semua mesin di dalam *cluster* bekerja sebagai *DataNode* dan *TaskTracker*. *DataNode* menyimpan data sistem. Tiap *DataNode* menyimpan *local storage* masing-masing. *TaskTracker* yang bertugas menjalankan operasi *Map* dan *Reduce*.

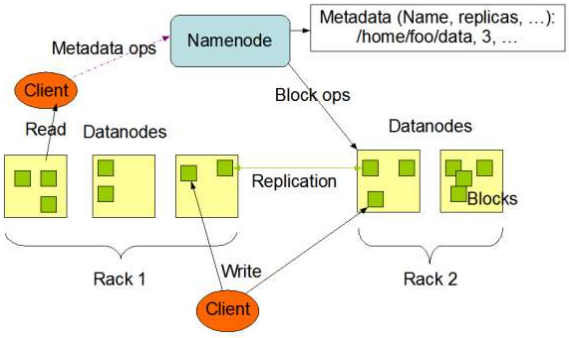


Gambar 1. Hadoop Architecture [4]

Hadoop terdiri dari 2 bagian, yaitu HDFS (*Hadoop Distributed File System*) dan *MapReduce*. HDFS terdiri dari *name node* dan *data node*. *MapReduce* terdiri dari *job tracker* dan *task tracker*.

2.5 Hadoop Distributed File System (HDFS)

Hadoop memiliki sebuah *Distributed FileSystem* yang disebut *Hadoop Distributed FileSystem* dimana dapat menyimpan informasi dalam jumlah yang besar dengan akses data yang berpola. HDFS dimodel berdasarkan GFS (*Google File System*). Arsitektur dari HDFS dapat dilihat pada Gambar 2.



Gambar 2. HDFS Architecture [2]

HDFS memiliki beberapa fitur utama yaitu [8] :

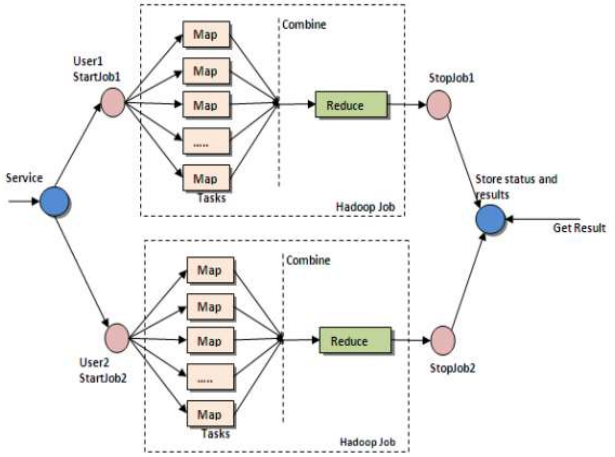
a. Penyimpanan data pada HDFS berbentuk blok yang besarnya secara umum 64 MB yang lebih besar daripada 4-32 kb (pada *file system* umumnya).
b. HDFS dioptimasi dalam hal kecepatan *transfer* data sehingga lebih efisien dalam membaca *request* data yang sangat besar, namun kurang bagus untuk membaca *request* data yang sedikit.
c. Setiap *cluster* HDFS memiliki 2 tipe *node* yang bekerja dalam arsitektur *master-slave*. *Namenode* adalah *node* yang bekerja sebagai *master* dan beberapa *datanode* adalah *node* yang bekerja sebagai *slave*. *Namenode* mengatur *namespace* dari

FileSystem dan bertugas untuk mengelola *tree FileSystem* dan *metadata* dari semua *file* dan direktori dari *tree*. Semua informasi disimpan dalam 2 bentuk *file*, yaitu *namespace image* dan *edit log*. *Namenode* memiliki replika sebagai *backup* yang disebut *Secondary Namenode*. *Datanode* bekerja untuk menyimpan dan mengambil blok ketika diperintah, kemudian melakukan *report* kepada *namenode* tentang *list* blok yang akan disimpan.

HDFS menggunakan replikasi data sebagai data redundan untuk mengatasi kegagalan *disk*. Data direplikasi ke *harddisk* yang ada secara fisik. Tiap blok yang menyimpan *file* disebarkan ke banyak *node* di dalam satu *cluster*. HDFS *Namenode* secara konstan melakukan *monitoring* terhadap *report* yang diberikan oleh *datanode* untuk memastikan bahwa kegagalan yang ada tidak menghilangkan *blok* sampai jumlahnya dibawah faktor replikasi.

2.6 Hadoop MapReduce

MapReduce merupakan sebuah model pemrograman untuk pemrosesan data. *MapReduce* dapat berjalan secara paralel sehingga dapat menyelesaikan analisis data dengan ukuran yang sangat besar. *MapReduce* sendiri terdiri dari 2 bagian, yaitu fase *map* dan fase *reduce* dimana masing-masing fase memiliki *key* dan *value* sebagai *input* dan *output* yang tipenya dapat ditentukan sendiri oleh *programmer*. *Programmer* juga harus menentukan isi dari fungsi *map* dan *reduce*. Arsitektur *MapReduce* pada *Hadoop* dapat dilihat pada Gambar 3.



Gambar 3. MapReduce Architecture [1]

Diantara *Mapper* dan *Reducer* terdapat *combiner* yang merupakan *semi-Reducer*. *Combiner* melakukan proses data dari *output* data *map* dan kemudian diberikan kepada *reducer*. Ketika menjalankan *MapReduce* maka *Hadoop* akan menjalankan *job*. Sebuah data dapat memiliki banyak *job* yang disebarkan ke tiap *cluster*. *Job* akan melakukan *mapping* yang disebut proses *task*. Hasil dari *map* akan berupa *pairing key* dan *value*. Hasil dari banyak *mapping* akan digabungkan kemudian dilakukan *reduce*. Setelah *task reduce* selesai maka *job* dari tiap *cluster* akan disatukan menghasilkan *result*.

3. DESAIN SISTEM

3.1 Desain Arsitektur Sistem

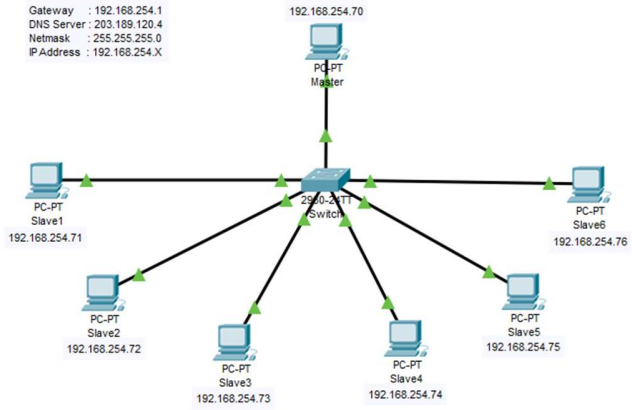
Dalam mengimplementasikan *FP-Growth Tree* pada *Hadoop*, akan digunakan 7 buah komputer dimana pembagiannya adalah 1 komputer sebagai *master* dan 6 komputer sebagai *slave*. Spesifikasi *hardware* dari setiap komputer adalah :

- Processor : Intel Core i5-4570 CPU @ 3.20GHz x 4
- Graphics : Intel Haswell Desktop
- Memory : 7,7 GiB
- Gnome : 3.28.2
- OS type : 64-bit
- Disk : 479,8 GB
- Connection : Fast Ethernet 100 Mbps

Dengan spesifikasi software sebagai berikut :

- Ubuntu 18.04.3 LTS
- Java SE Development Kit 12
- Hadoop 3.2.1
- SSH

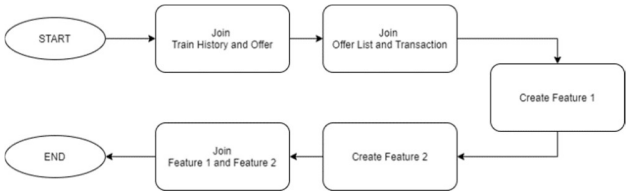
Untuk *network configuration* setiap komputer, komputer *master* memiliki *IP Address* 192.168.254.70. Sedangkan 6 komputer *slave* lainnya memiliki *range IP Address* dari 192.168.254.71-192.168.254.76. Untuk gambaran jelasnya dapat dilihat pada Gambar 4.



Gambar 4. Desain Arsitektur Sistem

3.2 Implementasi Algoritma untuk Proses Preprocessing Data

Pada bagian ini dijelaskan mengenai algoritma dalam proses pengolahan data hingga akhirnya menghasilkan data untuk proses *training* dan *testing*. Algoritma akan dipecah menjadi beberapa modul dimana masing-masing modul memiliki kegunaan tersendiri. Gambar 5 merupakan gambar workflow program secara keseluruhan.



Gambar 5. *Workflow* program

Pada proses dilakukan penggabungan antara *file* trainHistory.csv dan offers.csv. Penggabungan kedua data ini bertujuan untuk mendapatkan data *offer* secara utuh yang akan digunakan untuk pengolahan selanjutnya. Hasil *file* dari proses penggabungan ini adalah offersList.csv. Pada proses kedua akan dilakukan penggabungan antara *file* offersList.csv dengan *file* transactions.csv. Penggabungan ini bertujuan untuk menghasilkan *file* transaksi yang baru dimana hanya berisi *customer* yang pernah mendapatkan penawaran saja. Hasil *file* dari proses penggabungan adalah filterTransactions.csv.

Proses selanjutnya adalah pengolahan data untuk membuat data yang akan digunakan untuk proses *training* dan *testing*. Proses ini dipecah menjadi dua buah proses yaitu fitur satu dan fitur dua. Fitur satu menghasilkan data berupa *customer id, offer id, repeater, repeattrips*, *offervalue*, total pengeluaran *customer*, jumlah transaksi, *amount*, dan *quantity* yang pernah dilakukan oleh *customer* terhadap *brand, company*, dan *category* pada *offer*. Sedangkan fitur dua menghasilkan data berupa *customer id*, jumlah transaksi, *amount*, dan *quantity* yang pernah dilakukan oleh *customer* terhadap *brand, company*, dan *category* pada *offer* yang dihitung selama 30, 60, 90, dan 180 hari sebelum *customer* mendapatkan *offer*. Kemudian seluruh hasil dari fitur satu dan dua digabung menjadi sebuah *file* yang kemudian dilanjutkan dengan proses *training* dan *testing* menggunakan *Google Colaboratory*.

4. PENGUJIAN SISTEM

4.1 Proses Pengujian

Sebelum melakukan proses training dan testing, data harus di proses terlebih dahulu. Proses *preprocessing* dilakukan pada Hadoop dimana menggunakan 7 worker dengan masing-masing worker memiliki block size 300 GB. Proses preprocessing ini digunakan untuk menghilangkan data-data yang tidak terpakai dimana customer yang tidak terdapat dalam data train history akan dihilangkan. Hasil dari preprocessing ini menghasilkan data dengan ukuran 10.61 GB dari sebelumnya 19.81 GB. Kemudian data hasil preprocessing digunakan untuk membuat feature data sehingga dapat digunakan untuk proses training dan testing.

4.2 Hasil Pengujian

Pada penelitian ini, objektif yang ingin dicapai adalah agar dapat melakukan prediksi *customer* mana yang apabila mendapatkan *offer* tertentu akan melakukan retensi. Berdasarkan hasil pengujian dengan menggunakan *grid search*, akurasi tertinggi baik pada percobaan satu dan dua adalah ketika *gamma* sama dengan 10 dan data *test* sebersar 30% dengan akurasi 72,86%**.** Namun apabila diperhatikan pada *true* positif dan *false* positif dimana bernilai 0. Sedangkan untuk *true negative* dan *false negative* bernilai 34986 dan 13032. Hal ini berarti model yang telah terbentuk dan diujikan tidak pernah menghasilkan prediksi sama dengan *true* atau *retention*. Dengan demikian maka objektif yang ingin dicapai menjadi tidak terwujud karena ketidakmampuan model dalam memprediksi *customer* yang akan melakukan retensi. Oleh karena itu agar dapat menjawab objektif yang diinginkan, maka akan dilihat berdasarkan *true* positif yang tertinggi. Pada data *test* sebesar 30%, *true* positif yang tertinggi adalah pada *gamma* sama dengan 0,1.

Tabel 1 Hasil pengujian pada data test 30% dan gamma 0.1

Percobaan	True Positive	False Positive	False Negative	True Negative
1	2798	9874	10234	25112
2	2951	9985	10081	25001

Berdasarkan Tabel 1 hasil *true* positif tertinggi adalah pada percobaan dua. Dengan demikian dapat disimpulkan bahwa hasil terbaik dalam menjawab objektif yang ingin dicapai adalah menggunakan data *test* percobaan kedua sebesar 30% dan *gamma* sama dengan 0.1 dengan akurasi 58,21%.

5. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil pengujian yang dilakukan pada sistem, maka dapat disimpulkan bahwa :

- *Features data* yang digunakan sangat mempengaruhi hasil prediksi. Hal ini dapat dilihat bahwa pada percobaan satu yang merupakan *feature* satu dan dua yang merupakan gabungan dari *feature* satu dan dua memiliki hasil akurasi yang berbeda dimana pada percobaan dua akurasi lebih tinggi daripada percobaan pertama.
- Hasil prediksi paling baik untuk menjawab objektif adalah menggunakan data feature satu dan dua. *Feature* satu berisikan perhitungan jumlah transaksi, barang yang dibeli, serta uang yang digunakan terhadap kombinasi *category*, *brand*, dan *company*. Sedangkan *feature* dua berisikan *customer id*, jumlah transaksi, *amount*, dan *quantity* yang pernah dilakukan oleh *customer* terhadap *brand, company,* dan *category* pada *offer* yang dihitung selama 30, 60, 90, dan 180 hari sebelum *customer* mendapatkan *offer.* Dengan menggunakan rasio data training terhadap data testing adalah 7:3 dan gamma 0.1 menghasilkan akurasi 58,21%.

5.2 Saran

Saran yang dapat diberikan untuk menyempurnakan dan mengembangkan program ini lebih lanjut antara lain:

- Integrasi dengan infrastruktur *Big Data* seperti *Apache Spark* karena sudah dapat memproses data tidak hanya *Big Data* saja tetapi juga *machine learning*. Sehingga tidak diperlukan lagi pihak ketiga apabila memerlukan pemrosesan menggunakan *machine learning*.
- Untuk kedepan dapat dilakukan analisa *customer retention* dengan menggunakan metode klasifikasi lain seperti Extra Tree Classifier atau Random Forest.

6. DAFTAR PUSTAKA

[1] Apache. 2015. Apache Hadoop 2.7.1 – MapReduce Tutorial. URI=https://hadoop.apache.org/docs/r2.7.1/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html.

[2] Apache. 2015. Apache Hadoop 2.7.1 – HDFS Architecture. URI=https://hadoop.apache.org/docs/r2.7.1/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html.

[3] Awad, M., & Khanna, R. 2015. *Efficient Learning Machines.* Springer Science.

[4] Bhosale, H.S. & Gadekar, P. D. 2014. A Review Paper on Big Data and Hadoop. *International Journal of Scientific and Research Publications, Volume 4, Issue 10, 1-6*.

[5] Kowalczyk, A. 2017. Support Vector Machines Succinctly. Syncfusion.

[6] Permana, Y. 2016. Mengenal Big Data. URI=https://www.codepolitan.com/mengenal-big-data.

[7] SAS. 2019. Big Data: What it is and why it matters. URI=https://www.sas.com/en_id/insights/big-data/what-is-big-data.html

[8] Turkington, G. 2013. Hadoop Beginner's Guide. Packt Publishing Ldt.

[9] Maitrey, S. & Jha, C.K. 2015. MapReduce:Simplified Data Analysis of Big Data. *Procedia Computer Science 57*, 563 – 571.