Analisis Kinerja Algoritma *Distributed Support Vector Machine* Menggunakan Spark Untuk Memprediksi Penundaan Penerbangan

Performance Analysis of Distributed Support Vector Machine Algorithm For Predicting Flight Delays

Husnul Khotimah[1], Ari Hernawan, S.Kom., M.Sc.[1], Gibran Satya Nugraha, S.Kom., M.Eng.[2]
[1]Dept Informatics Engineering, Mataram University
Jl. Majapahit 62, Mataram, Lombok NTB, INDONESIA
Email: husnulhk31@gmail.com, arihernawan@unram.ac.id, gibransn@unram.ac.id

Abstract In big data analysis, it requires powerful machine learning frameworks, strategies, and environments to analyze data at scale. Therefore, Apache Spark is used as a cluster computing framework to process big data in parallel and can run on multiple clusters. In this study, the Support Vector Machine (SVM) algorithm is used as a classification method to predict whether a flight will experience a delayed arrival or not. This study also aims to analyze the performance of the distributed SVM algorithm using the Apache Spark framework in classifying delayed flight arrivals. Running time evaluation plays an important role in proving how fast the data processing is done using Apache Spark. In addition, there is a test to prove the effect of using the SVM algorithm with a distributed system on the results of the classification accuracy of delayed flight arrivals. Distributed SVM performance testing is carried out using variations in data size and the number of worker nodes in the built cluster. From the test results, it was found that the most effective number of worker nodes used in the flight delay classification process were 4 worker nodes with the lowest running time results from the experiment of 4 variations in the number of worker nodes. In terms of accuracy, adding the number of worker nodes does not affect the accuracy of the program. The difference in accuracy results is caused by the random oversampling process on the data performed on each system test. The use of the SVM algorithm with Spark is sufficient to provide good performance in the classification process with the highest accuracy result in the test being 93.98%..

Key words*: *Big data*, Apache Spark, *Support Vector Machine, Distributed Computing, Flight Delay Classification

I. INTRODUCTION

Perkembangan teknologi digital di berbagai bidang menghasilkan data dalam jumlah besar atau biasa disebut *big data*. Penyimpanan dan pengolahan data dalam skala besar yang baik dan cepat juga memerlukan teknologi komputer yang memiliki performa yang tinggi atau biasa disebut super komputer. Namun untuk menciptakan suatu sistem super komputer memerlukan biaya yang cukup mahal. Oleh karena itu sistem terdistribusi dapat menjadi solusi yang bisa digunakan. Salah satu jenis sistem terdistribusi adalah *Distributed Computing System*. *Distributed Computing System* adalah salah satu teknik untuk melakukan komputasi secara bersamaan dengan menggunakan beberapa komputer yang saling terhubung pada pemrosesan data dalam jumlah yang besar [1]. Sistem yang menggabungan beberapa beberapa komputer dalam suatu jaringan disebut dengan *computing*. Setiap komputer yang merupakan bagian dari *cluster* diberi nama *worker node* [2] Salah satu *framework* yang dapat digunakan untuk mengolah *big data* secara paralel adalah Apache Spark

Apache Spark merupakan salah satu *framework cluster computing* yang dapat digunakan untuk analisis *big data* dengan cepat. Apache Spark sangat cocok digunakan untuk aplikasi iteratif dan *machine learning* karena dapat menyimpan data, hasil, dan bekerja di dalam memori [1]. Apache Spark memiliki komponen *library open-source* yang digunakan untuk melakukan analisis *big data*, yaitu *Machine Learning library* (MLlib). MLlib Apache Spark dilengkapi dengan serangkaian fitur untuk *machine learning* seperti klasifikasi [3]

Salah satu tujuan big data adalah mampu memprediksi suatu kejadian di masa yang akan datang. Dalam dunia penerbangan, keterlambatan jadwal menjadi sesuatu yang sering terjadi. Hal ini mengakibatkan dampak negatif terhadap banyak hal, seperti ketepatan jadwal, keamanan penerbangan, dan kepuasan pelanggan [4]. Untuk memprediksi keterlambatan keberangkatan penerbangan dapat juga dilakukan dengan memprediksi keterlambatan dari kedatangan penerbangan. Oleh karena itu, dalam penelitian ini akan dilakukan prediksi apakah suatu kedatangan penerbangan akan mengalami keterlambatan atau tidak. Dari hasil prediksi tersebut pihak maskapai dapat mempersiapkan langkah-langkah untuk menghindari keterlambatan.

Dalam prediksi keterlambatan dari kedatangan penerbangan ini digunakan metode klasifikasi dengan menggunakan algoritma *Support Vector Machine* (SVM). Beberapa metode yang dapat digunakan pada klasifikasi yaitu Logistic Regression, Naive bayes, SVM dan lain-lain.

Pada penelitian [4] diantara beberapa algoritma tersebut SVM merupakan salah satu algoritma yang berhasil diterapkan pada kasus klasifikasi dengan tingkat akurasi paling tinggi dibandingkan algoritma lainnya yaitu sebesar 97%. SVM merupakan algoritma klasifikasi *machine learning* yang cocok digunakan karena mampu menangani data yang kompleks seperti data penerbangan. Algoritma SVM juga memiliki kemampuan untuk mengatasi masalah *overfitting* dan *underfitting* karena kemampuannya dalam menemukan *hyperplane* terbaik yang memisahkan data dengan margin yang maksimal [5]. Dengan menggunakan SVM, dapat ditentukan *hyperplane* terbaik yang memisahkan data kedatangan penerbangan yang tertunda dan tidak tertunda.

Spark menggunakan pemrosesan *in-memory* untuk mempercepat komputasi. Hal ini menghasilkan kinerja yang tinggi dan waktu eksekusi yang lebih cepat saat melatih model SVM pada dataset besar. Dalam Spark, model SVM dapat dilatih dan dievaluasi secara terdistribusi di seluruh *cluster* Spark. Ini memungkinkan pemanfaatan sumber daya yang tersedia dalam *cluster* dan mempercepat proses pelatihan model dengan memanfaatkan komputasi paralel.

Penelitian ini dilakukan untuk menganalisis kinerja algoritma SVM menggunakan *distributed computing* terutama dari sisi kecepatan dalam melakukan klasifikasi kedatangan penerbangan yang tertunda. Hal ini dilakukan karena belum adanya hasil evaluasi spesifik yang memberikan hasil kinerja algoritma SVM yang digunakan dalam melakukan klasifikasi menggunakan framework Apache Spark. Kinerja algoritma SVM terdistribusi dapat dilihat dari waktu *running time* yang dibutuhkan dalam melakukan pemrosesan data. Oleh karena itu, pada penelitian ini dilakukan pengujian terhadap kinerja penggunaan SVM terdistribusi dengan *framework* Apache Spark pada proses klasifikasi dengan melakukan perbandingan pada penggunaan jumlah *worker node* yang berbeda guna mengetahui jumlah *node worker* yang paling efektif digunakan pada proses klasifikasi. Evaluasi *running time* yang didapatkan dari program sangat berperan penting untuk membuktikan seberapa cepat proses pengolahan data dilakukan dengan menggunakan Apache Spark yang dijalankan pada berbagai ukuran *cluster*.

II. TINJAUAN PUSTAKA

A. Penelitian Terkait

Pertama, penelitian [5] ini menganalisis berbagai jurnal mengenai big data dan algoritma *Machine Learning* . Pada jurnal ini akan meringkas berbagai aplikasi dari Machine Learning yang digunakan pada *Big Data Analytic*. Hasil dari penelitian ini adalah Spark merupakan *framework* yang paling banyak digunakan oleh para peneliti di bidang *Big Data Analytic* yaitu sebesar (34,88%). Selain itu didapatkan bahwa SVM merupakan penggunaan algoritma pembelajaran mesin yang paling banyak digunakan dalam *Big Data Analytic* karena performanya yang bagus.

Kedua, penelitian [6] ini bertujuan untuk mengidentifikasi kelebihan dan kekurangan dari penggunaan Apache Spark dan Hadoop *MapReduce* dalam klasifikasi kumpulan data besar. Dataset yang digunakan pada penelitian ini terdiri dari beberapa kumpulan data dengan ukuran yang berbeda dan keseluruhan data tersebut merupakan kumpulan data klasifikasi biner. Hasil yang diperoleh dari penelitian ini yaitu penggunaan *framework* Spark sekitar 5 kali lebih cepat daripada Hadoop dalam hal eksekusi pada fase training.

Ketiga, penelitian [7] membahas mengenai penggunaan *Big Data Analytic* untuk mendukung analisis prediksi dan peningkatan pengambilan keputusan di perguruan tinggi. *Big Data Analytic* dalam penelitian ini menggunakan library pada Apache Spark untuk dapat mengolah big data. Dan hasil dari penelitian ini berhasil mengelompokkan kinerja mahasiswa di perguruan tinggi.

Keempat, pada penelitian [8] ini diketahui bahwa SVM merupakan algoritma klasifikasi yang memiliki jangkauan yang luas. Pada penelitian ini diusulkan penggunaan SVM yang dijalankan secara paralel di Spark guna mempercepat proses training data. Hasil dari penelitian ini menunjukkan bahwa *Multiple Submodels Paralel* (MSM-SVM) menghabiskan *running time* yang lebih sedikit untuk melakukan proses klasifikasi.

Berdasarkan dari tinjauan pustaka, penelitian ini akan menganalisis kinerja algoritma SVM dengan menggunakan sistem terdistribusi Apache Spark untuk klasifikasi jadwal kedatangan penerbangan yang tertunda. Kinerja sistem akan diuji dengan menggunakan beberapa jumlah *worker* pada Spark *cluster*.

B. Teori Penunjang

Big Data Analytic dapat didefinisikan sebagai proses mengumpulkan, mensistematisasikan, dan meneliti *big data* untuk mengetahui dan menampilkan pola, menemukan pengetahuan dan informasi di dalam *big data*. *Big Data Analytic* terdiri dari tiga jenis yaitu *Big Data Descriptive Analytic*, *Big Data Predictive Analytic* dan *Big Data Prescriptive Analytic* [5].

Apache Spark merupakan *framework* terdistribusi yang digunakan pada *Big Data Analytic* untuk pemrosesan data berskala besar karena menyediakan antarmuka untuk *cluster* pemrograman dengan paralelisme data. Terdapat beberapa fitur yang ada pada Apache Spark antara lain [3]:

1. RDD, menyediakan API yang dapat melakukan operasi transformasi seperti *map*, *filter*, dan *reduce* serta operasi aksi seperti *count*, *collect*, dan *save* pada data [9].
2. DataFrame menyediakan antarmuka berbasis kolom yang terstruktur yang digunakan untuk pemrosesan data.
3. Spark *Core* merupakan mesin dasar yang digunakan pada pemrosesan data paralel dan terdistribusi.
4. Spark SQL merupakan modul Spark yang dirancang untuk memproses data secara structural.
5. *MLlib* Apache Spark merupakan *library* yang terdiri dari beragam Algoritma *Machine Learning*.
6. *GraphX* merupakan sebuah paralel komputasi yang menggunakan API pada pemrosesan data.

7. Spark *Streaming* merupakan *framework stream* yang berjalan di Apache Spark yang dapat digunakan untuk memproses data secara *real time.*
8. Spark *Cluster Managers* merupakan komponen *Spark* yang membantu mengelola sumber daya/*cluster*.

PySpark merupakan salah satu modul Python yang berfungsi sebagai antarmuka atau API untuk Apache Spark,. PySpark dapat dimanfaatkan dalam pengembangan aplikasi Spark menggunakan bahasa pemrograman Python. PySpark menggunakan bahasa pemrograman python dalam mengimplementasikan *machine learning* yang digunakan untuk pengklasifikasian. Penggunaan PySpark dapat memanfaatkan fitur-fitur Apache Spark yang terintegrasi dengan kode Python [10].

Support Vector Machine (SVM) adalah sistem pembelajaran yang menggunakan ruang hipotesis berupa fungsi linier dalam sebuah ruang fitur yang berdimensi tinggi. SVM berfungsi untuk menemukan *hyperplane* terbaik yang digunakan sebagai garis pemisah dua kelas yang berbeda. Proses penentuan jarak terjauh dilakukan berulang kali hingga menemukan *hyperplane* terbaik. SVM mencari *hyperplane* berdasarkan *support vectors* dan margin. *Support vectors* merupakan seluruh vektor data yang memiliki jarak terdekat dengan *hyperplane*, sedangkan margin adalah besarnya lebar dari pemisah *hyperplane* [11]. Berikut adalah algoritma SVM paralel di Spark:

1. Dataset dipecah menjadi beberapa subset yang akan diproses secara paralel dan dapat diakses oleh Spark.
2. Spark menerapkan algoritma SVM ke setiap subset secara terpisah. SVM diimplementasikan pada setiap subset menggunakan library MLlib Spark.
3. Setelah pelatihan SVM selesai, parameter model SVM, seperti bobot, bias, dan nilai support vector dikirim ke Spark driver untuk digabungkan dan menghasilkan model SVM yang terlatih.
4. Model SVM yang telah dilatih dievaluasi untuk mengetahui kinerja dari model.
5. Setelah model dilatih dan dievaluasi, model tersebut dapat digunakan untuk melakukan prediksi pada data baru.

SVM pada Spark dapat menggunakan algoritma *Batch Gradient Descent* untuk mempercepat pelatihan dan memungkinkan pelatihan SVM dilakukan pada beberapa mesin secara paralel. Berikut beberapa Persamaan yang digunakan pada penggunaan algoritma SVM [8] :

1. Fungsi keputusan :
$$f(x) = w^T x + b \quad (1)$$
2. Fungsi Objektif :
$$\min f(w) = \frac{1}{n}\sum_{i=1}^{n} \max(0, 1 - y_i(w^T x_i + b)) + \frac{\lambda}{2}||w||^2 \quad (2)$$
3. Gradient Descent:
$$\mathrm{L}(w^T \mathrm{xy}) = \frac{1}{n}\sum_{i=1}^{n} \begin{cases} -y_i x_i & y_i(w^T x_i + b) \le 1 \\ 0 & otherwise \end{cases} \quad (3)$$
4. Perbarui Bobot :
$$w' = w - a \,.\, \mathrm{L}(w^T \mathrm{xy}) \quad (4)$$

Keterangan :

- w : Vektor bobot SVM.
- x_i : Vektor fitur dari data pelatihan ke-i.
- y_i : Label target (1 atau -1) dari data pelatihan ke-i.
- b : Konstanta bias.
- n : Jumlah total data pelatihan.
- λ : Parameter regulasi
- a : *Learning rate*.
- $\mathrm{L}(w^T \mathrm{xy})$: Gradien fungsi objektif terhadap vektor bobot SVM.

Ketika menggunakan beberapa *worker* pada pelatihan SVM, algoritma SVM akan dipecah menjadi beberapa subset yang akan diproses secara paralel pada setiap *worker*. Setiap *worker* akan menghitung gradien dari *subset* yang diberikan dan mengirimkannya ke *driver* untuk diperbarui. Setelah *driver* menerima seluruh gradien, bobot SVM akan diperbarui dan proses ini akan diulangi untuk beberapa iterasi yang telah ditentukan. Berikut beberapa Persamaan yang digunakan pada penggunaan algoritma SVM dengan menggunakan beberapa *worker* [8] :

1. Fungsi Objektif :
$$\min f(w) = \frac{1}{m}\sum_{j=1}^{m}\left[\frac{1}{|S_j|}\sum_{i \in S_j} \max(0, 1 - y_i(w^T x_i + b)) + \frac{\lambda}{2}||w||^2\right] \quad (5)$$
2. Perbarui Bobot Menggunakan *Map Reduce Batch Gradient Descent* :
$$w' = w - a \,.\, \mathrm{L}(w^T \mathrm{xy})$$
$$w' = w - a \,.\, \frac{1}{n}\sum_{i=1}^{n} \begin{cases} -y_i x_i & y_i(w^T x_i + b) \le 1 \\ 0 & otherwise \end{cases} \quad (6)$$

Contoh :

m =100, n =400

Machine 1 : Use $(x_1, y_1), \ldots, (x_{100}, y_{100})$

$$temp^{(1)} = \sum_{i=1}^{100} \begin{cases} -y_i x_i & y_i(w^T x_i + b) \le 1 \\ 0 & otherwise \end{cases}$$

Machine 2 : Use $(x_{101}, y_{101}), \ldots, (x_{200}, y_{200})$

$$temp^{(2)} = \sum_{i=101}^{200} \begin{cases} -y_i x_i & y_i(w^T x_i + b) \le 1 \\ 0 & otherwise \end{cases}$$

Machine 3 : Use $(x_{201}, y_{201}), \ldots, (x_{300}, y_{300})$

$$temp^{(3)} = \sum_{i=201}^{300} \begin{cases} -y_i x_i & y_i(w^T x_i + b) \le 1 \\ 0 & otherwise \end{cases}$$

Machine 4 : Use $(x_{301}, y_{301}), \ldots, (x_{400}, y_{400})$

$$temp^{(4)} = \sum_{i=301}^{400} \begin{cases} -y_i x_i & y_i(w^T x_i + b) \le 1 \\ 0 & otherwise \end{cases}$$

Keterangan :

- w : Vektor bobot SVM.
- x_i : Vektor fitur dari data pelatihan ke-i.
- y_i : Label target (1 atau -1) dari data pelatihan ke-i.
- b : Konstanta bias.
- m : Jumlah *batch* (subset) data pelatihan yang akan dibagi ke dalam beberapa *worker*.
- S_j : Subset data pelatihan ke-j yang digunakan pada *worker* ke-j.
- n : Jumlah total data pelatihan.
- λ : Parameter regulasi

- $L(w^T xy)$: Gradien fungsi objektif terhadap vektor bobot SVM.
- a : *Learning rate*.

Pada SVM dengan *batch gradient descent*, *subset* dari data *training* akan dibagi ke dalam beberapa *batch.* Setiap *batch* akan dilakukan perhitungan gradien secara independen oleh setiap *worker* pada *cluster*. Setelah itu, gradien dari setiap *batch* akan dijumlahkan dan dihitung rata-ratanya untuk mendapatkan gradien yang akan digunakan untuk memperbarui bobot [8].

Metode *Random oversampling* (ROS) digunakan dalam pemrosesan data klasifikasi untuk mengatasi ketidakseimbangan jumlah sampel antara kelas mayoritas dan kelas minoritas. ROS merupakan proses resampling yang dilakukan dengan cara memilih data pada kelas minoritas secara acak, lalu pada data yang terpilih, dilakukan duplikasi dan ditambahkan pada data [12]. Dalam penggunaan ROS, terlebih dahulu menghitung selisih antara jumlah sampel kelas mayoritas dan kelas minoritas untuk mengetahui tingkat ketidakseimbangan data. Selanjutnya, dilakukan perulangan sesuai selisih tersebut, di mana setiap iterasi perulangan akan mengambil data secara acak dari kelas minoritas dan menambahkannya ke dalam data training. Proses ini dilakukan hingga jumlah sampel kelas minoritas setara dengan kelas mayoritas. Dengan menerapkan ROS, distribusi sampel antara kelas-kelas menjadi lebih seimbang dan memungkinkan algoritma pembelajaran mesin untuk mempelajari pola dari kelas minoritas dengan lebih baik [13].

Metode *Feature Selection* dapat digunakan untuk mengurangi dimensi data dari atribut yang berlebihan atau tidak relevan. Seleksi fitur bekerja dengan mengurangi jumlah fitur dan memilih hanya fitur-fitur yang memberikan manfaat secara signifikan. Pengurangan jumlah fitur ini bertujuan untuk mengatasi masalah overfitting, yang dapat terjadi ketika model terlalu kompleks [14]. Penggunaan optimasi *feature selection* memungkinkan untuk melakukan pemilihan subset dari fitur yang relevan yang digunakan pada konstruksi model sehingga mampu mengatasi masalah imbalencing data. Pemilihan fitur berbasis korelasi merupakan teknik yang efektif untuk meningkatkan performa klasifikasi pada data yang tidak seimbang [15].

Normalisasi adalah proses mengubah ulang data agar nilai-nilai ambigu dapat dieliminasi [16]. *MinMax Scaler* merupakan salah satu jenis normalisasi yang dilakukan pada tahap preprocessing data. Normalisasi data ini bekerja dengan cara menyesuaikan nilai data dalam rentang tertentu yaitu antara 0 sampai 1 [17]. Berikut adalah persamaan dari *MinMax Scaler* :

$$x' : \frac{x - x_{min}}{x_{max} - x_{min}} \quad (7)$$

Keterangan:

- x : Nilai asli data
- $xmin$: Nilai minimal dari keseluruhan data

$xmax$: Nilai maksmimal dari keseluruhan data

III. METODE PENELITIAN

A. Alat dan Bahan

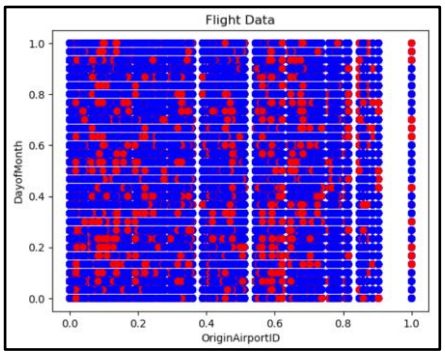
Alat dan bahan yang dibutuhkan pada penelitian ini berupa *software*, *hardware* serta data dan informasi pendukung selama dilakukannya penelitian.

A.1. Alat

1. Laptop dengan sistem operasi Windows 10 digunakan sebagai perangkat untuk dokumentasi penelitian.
2. 5 *Virtual Private Server* (VPS) digunakan untuk membuat Spark *cluster.*
3. Bahasa Pemrograman *Python* digunakan sebagai bahasa pemrograman dalam pemrograman Spark.
4. Jupyter *Notebook* digunakan untuk membuat, menjalankan, dan memantau kode *Python*.
5. Spark *Framework* menyediakan fitur dan fungsi untuk pemrosesan *big data*.

A.2. Bahan

Bahan yang digunakan pada penelitian ini diambil dari literatur-literatur dari jurnal, buku, penelitian sebelumnnya. Selain itu bahan lain yang diperlukan yaitu dataset *Flight Status Prediction*. Dataset *Flight Status Prediction* yang digunakan di dalam penelitian ini merupakan informasi penerbangan di Amerika Serikat pada tahun 2018 hingga 2022 dengan ukuran 10,19 GB yang bersumber dari Kaggle [18]. Dataset ini terdiri dari beberapa file CSV yang berisi informasi tentang penerbangan dari berbagai maskapai penerbangan di Amerika Serikat.



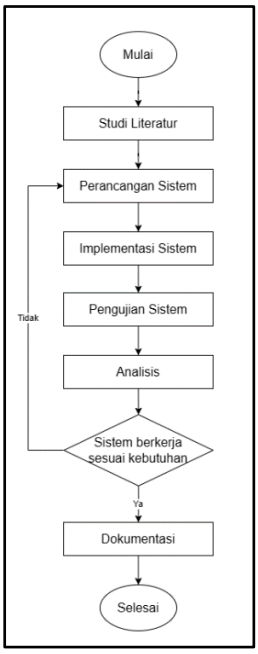
Gambar. 1. Dataset Flight Status Prediction

Pada Gambar 1 merupakan visualisasi dari dataset Flight Status Prediction. Dataset ini berisi 119 fitur yang dikategorikan ke dalam dua kelompok yaitu *delay* dan *on time* pada data keterlambatan kedatangan penerbangan. Pada penelitian ini, digunakan tiga ukuran dataset yang berbeda yang diambil dari dataset Flight Status Prediction, yaitu 1 GB dengan jumlah *record* 2.726.277, 1,5 GB dengan jumlah *record* 3.763.588, dan 2 GB dengan jumlah *record* 5.054.186. Beberapa fitur yang digunakan pada penelitian ini karena relevan dan memiliki kontribusi tinggi terhadap keterlambatan kedatangan penerbangan antara lain [19]:

1. *Origin Airport Id*: ID Bandara keberangkatan.
2. *Day of Month* : Angka yang menunjukkan hari dalam sebulan
3. *Day of Week* : Angka yang menunjukkan hari dalam seminggu
4. *Taxi-in Time* : Waktu yang dibutuhkan pesawat setelah mendarat untuk bergerak dari landasan pacu menuju gerbang atau tempat parkir di bandara.
5. *Air Time* : Waktu yang diperlukan pesawat dalam perjalanan di udara
6. *Carrier Delay* : Waktu keterlambatan penerbangan yang disebabkan oleh maskapai
7. *Weather Delay* : Waktu keterlambatan penerbangan yang disebabkan oleh kondisi cuaca
8. *NAS Delay* : Waktu keterlambatan penerbangan yang disebabkan oleh gangguan dalam Sistem Ruang Udara Nasional.

B. Alur Penelitian

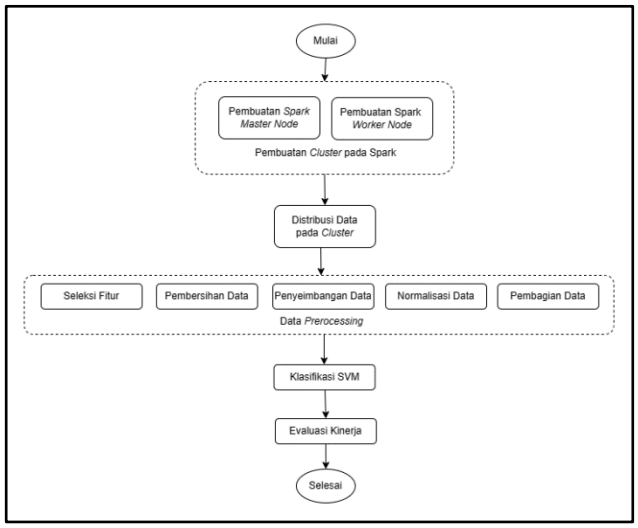
Penelitian ini dilakukan melalui beberapa tahapan yaitu studi literatur dan persiapan alat dan bahan, perancangan sistem, implementasi sistem, pengujian sistem, dan dokumentasi. Alur langkah penelitian yang dilakukan dapat dilihat pada Gambar 3.1.



Gambar. 2. Alur Penelitian

1. Pada tahap studi literatur dan persiapan alat dan bahan dilakukan analisis terhadap kebutuhan dalam pengembangan sistem dengan mengumpulkan beberapa referensi yang berkaitan dengan topik penelitian yang dilakukan.
2. Pada tahap perancangan sistem akan dilakukan perancangan konsep pembuatan sistem.
3. Pada tahap implementasi dilakukan pembuatan sistem berdasarkan pada konsep yang telah dirancang. Diawali dengan pembuatan Spark *cluster*, klasifikasi menggunakan SVM, proses pengujian, dan evaluasi.
4. Pada tahap pengujian akan dilakukan pengujian untuk mengetahui pegaruh jumlah worker terhadap *running time* yang dijalankan oleh program. Pengujian ini dilakukan dengan menggunakan 4 *node worker* pada *cluster* Spark.
5. Pada tahap analisis, dilakukan analisis terhadap hasil yang didapatkan dari proses pengujian.
6. Pada tahap dokumentasi dilakukan dokumentasi berupa laporan dari proses analisis kinerja algoritma *Distributed* SVM untuk memprediksi jadwal kedatangan penerbangan yang tertunda.

C. Perancangan Sistem



Gambar. 3. Perancangan Sistem

1. Pembuatan Spark *Cluster*, pada tahap ini dilakukan penginstalan Spark *Cluster* yang bertugas mengkoordinasi tugas-tugas dalam *cluster*.
2. Pembuatan Spark Worker Node, pada tahap ini dilakukan penginstalan 4 Spark Worker Node yang akan terhubung ke Spark *Cluster*
3. Distribusi Data, pada tahap ini terdapat proses pendistribusian data pada *cluster* Spark.
4. Seleksi fitur, pada tahap ini dilakukan pemilihan fitur yang akan digunakan pada klasifikasi. Seleksi fitur dilakukan dengan berbasis korelasi.
5. Pembersihan data, pada tahap ini dilakukan pembersihan data dari nilai kosong.
6. Penyeimbangan data, pada tahap ini menggunakan metode *random oversampling* untuk menangani dataset penerbangan yang tidak seimbang.
7. Normalisasi data, pada tahap ini dilakukan normalisasi pada data yang digunakan.
8. Pembagian data, pada tahap ini dilakukan pembagian data menjadi dua bagian, yaitu data training dan data testing dan data uji. Data training digunakan untuk melatih model klasifikasi, sedangkan data testing digunakan untuk menguji kinerja model. Dataset ini akan dibagi menjadi 70% sebagai data *training* serta 30% sebagai data *testing*.

9. Klasifikasi SVM, pada tahap ini data akan diklasifikasikan menggunakan algoritma SVM. Pada SVM terdistribusi dengan penggunaan beberapa worker, data training akan dibagi menjadi beberapa subset. Setelah itu subset tersebut akan diproses secara paralel oleh beberapa worker yang berbeda pada *cluster*. Hasil dari setiap worker tersebut akan digabungkan untuk menghasilkan bobot SVM akhir
10. Pengujian, pada data yang telah diklasifikasikan akan dilakukan proses pengujian untuk menentukan kategori dari jadwal penerbangan. Pengujian juga dilakukan dengan melihat *running time* dan akurasi dari proses klasifikasi dengan menggunkan jumlah *worker node* yang berbeda.
11. Evaluasi, pada tahap ini digunakan sebagai evaluator untuk mengukur efektivitas klasifikasi yang dihasilkan. Evaluasi terhadap *running time* dan akurasi juga dilakukan pada setiap penggunaan jumlah *worker* yang berbeda pada Spark *cluster* yang digunakan untuk pemrosesan *machine learning*.

D. Metode

Penelitian ini menggunakan metode SVM untuk melakukan klasifikasi dataset penerbangan guna mengklasifikasikan kedatangan penerbangan yang tertunda. SVM merupakan metode klasifikasi yang bertujuan untuk menemukan *hyperplane* margin maksimum yang memisahkan kelompok vektor fitur antara dua kelas, yaitu kelas 0 dan 1. SVM yang digunakan pada penelitian ini adalah SVM yang tersedia pada Spark *Machine Learning*. SVM berfungsi untuk mencari hyperplane simetris yang memisahkan titik data dalam set pelatihan antara kelas yang berbeda. *Hyperplane* ini berfungsi sebagai batas keputusan yang memisahkan ruang menjadi dua bagian, di mana satu bagian digunakan untuk kelas dengan penerbangan yang tidak tertunda (kelas 0) dan bagian lainnya digunakan untuk kelas dengan penerbangan yang tertunda (kelas 1).

Pada SVM terdistribusi dengan penggunaan beberapa *worker*, data *training* akan dibagi menjadi beberapa *subset*. Setelah itu *subset* tersebut akan diproses secara paralel oleh beberapa *worker* yang berbeda pada *cluster*. Hasil dari setiap worker tersebut akan digabungkan untuk menghasilkan bobot SVM akhir.

E. Pengujian

Bentuk pengujian yang dilakukan adalah melakukan perbandingan hasil penggunaan jumlah *worker* yang berbeda terhadap kinerja algoritma SVM terdistribusi pada proses klasifikasi jadwal penerbangan tertunda menggunakan Apache Spark

E.1. Metrik Pengujian

1. Akurasi, digunakan untuk mengukur kinerja model dalam mengklasifikasikan data dengan benar secara keseluruhan.
2. Running time, yaitu mengukur waktu yang diperlukan oleh jalannya program hingga selesai. Hal ini dilakukan untuk membandingkan kecepatan eksekusi dengan penggunaan jumlah *worker node* yang berbeda.

E.2. Alur Pengujian

1. Melakukan konfigurasi *cluster* Apache Spark menggunakan Spark *cluster* sebagai infrastruktur yang terdiri dari 4 *node worker*. Setiap *node worker* harus terhubung dengan Spark master dan memiliki akses ke data yang sama di dalam penyimpanan lokal.
2. Menjalankan *cluster* dengan penggunaan 1 *node worker*.
3. Memuat data ke dalam DataFrame pada setiap *node worker* yang aktif.
4. Melakukan pra-pemrosesan data termasuk menyeimbangkan data.
5. Menetapkan parameter SVM dengan tipe kernel linear dan model dikonfigurasi.
6. Melatih model SVM menggunakan data yang telah diproses pada setiap *node worker*.
7. Model dievaluasi secara terpisah pada setiap *node worker*. Hasil evaluasi dari setiap *node worker* digabungkan untuk mendapatkan hasil keseluruhan.
8. Menganalisis waktu pelatihan, evaluasi model, dan penggunaan sumber daya.
9. Mengaktifkan tambahan 1 *node worker* dari *cluster* yang telah dibuat dan mengulangi Langkah 2-8 hingga penggunaan 1 *node worker*.
10. Menganalisis hasil evaluasi dari penggunaan 1 *node worker*, 2 *node worker*, 3 *node worker*, dan 4 *node worker* serta membandingkan kinerja model SVM dari setiap pengujian dengan penggunaan jumlah *node worker* yang berbeda. Penelitian dikatakan berhasil ketika seluruh alur pengujian selesai dilakukan

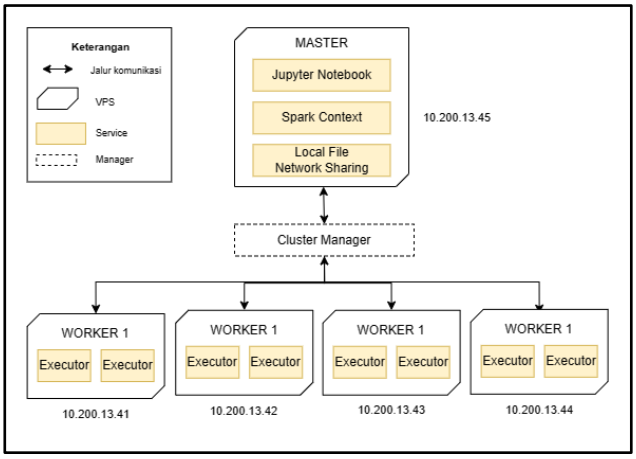
F. Cara Analisis

Analisis dilakukan dengan membandingkan hasil dari kinerja model SVM dengan jumlah *node worker* yang berbeda. Analisis terdiri dari waktu pelatihan, evaluasi model, dan penggunaan sumber daya saat menggunakan jumlah *node worker* yang berbeda. Berdasarkan analisis hasil tersebut dibuat kesimpulan tentang penggunaan jumlah *node worker* yang optimal dan memberikan kinerja serta hasil evaluasi yang lebih baik dalam klasifikasi penerbangan tertunda menggunakan algoritma SVM terdistribusi

IV. HASIL DAN PEMBAHASAN

A. Pembuatan Spark Cluster

Pada penelitian ini, *cluster* spark yang dibangun terdiri dari satu *master node* dan 4 *worker node*. Pemilihan 4 *worker node* sebagai jumlah maksimal *worker node* dari *cluster* didasarkan pada ketersedian jumlah VPS. Dengan menggunakan 4 *worker node*, dapat dilakukan pengujian terhadap kinerja *cluster Spark* dari skenario dengan 1 *worker node* hingga 4 *worker node*. *Master node* digunakan sebagai koordinator utama dalam pengelolaan sumber daya dan tugas-tugas di dalam *cluster*. *Worker node* digunakan untuk menjalankan tugas-tugas pemrosesan..



Gambar. 4. Topologi *Cluster* Spark

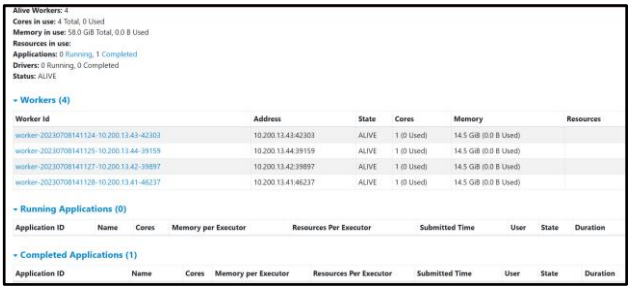
Pada Gambar 4 terdapat topologi dari *cluster* spark yang terdiri dari susunan node yang digunakan pada proses komputasi metode SVM secara terdistirbusi di spark. Pada setiap pergantian jumlah worker node, terdapat perubahan dari konfigurasi yang tertulis pada file 'spark-defaults.conf' serta 'slaves' yang terletak pada direktori konfigurasi spark yaitu '/usr/local/spark/conf'. Keseluruhan perubahan konfigurasi ini diterapkan ke seluruh *node* yang terlibat pada *cluster*. Pada file 'slaves' dapat disesuaikan dengan menambahkan node yang berperan sebagai *worker node* pada *cluster*. Pada file 'spark-defaults.conf' untuk mengoptimalkan kinerja spark, terdapat perubahan nilai pada parameter 'spark-defaults-parallelism' dan 'spark-executor instances'. Berikut merupakan rumus perhitungan pada kedua parameter tersebut :

- spark.executor.instances = jumlah *node* * (jumlah *core* per node - 1) – *core* untuk *application manager*
- spark-defaults-parallelism= jumlah parallelism per core * spark.executor.instances

Misalkan pada penggunaan 1 *worker node* :

- spark.executor.instances = 1 * (4-1) -1 = 2
- spark-defaults-parallelism = 2 * 2 = 4

Pada *master node*, terdapat Jupyter Notebook yang digunakan untuk menjalankan kode pemrograman klasifikasi menggunakan SVM. Jupyter Notebook dapat berinteraksi dengan spark *cluster* menggunakan PySpark. PySpark merupakan API Python untuk Spark yang mampu mengakses dan mengoperasikan Spark melalui bahasa pemrograman Python. Jupyter Notebook ini dijalankan pada port 8888 di *master node*. Di dalam Jupyter Notebook, terlebih dahulu dilakukan pembuatan sesi Spark untuk berinteraksi dengan *cluster* Spark. Pembuatan sesi ini dapat dilakukan dengan menggunakan objek *SparkSession* yang tersedia pada PySpark..



Gambar. 5. *Interface* Spark Master

Pada Gambar 5 merupakan interface pada cluster manager yang dapat diakses melalui port 8080. Interface ini memberikan informasi mengenai *cluster* spark seperti worker yang aktif. Selain itu interface ini juga menampilkan seluruh aktivitas saat sebuah proses sistem dijalankan mulai dari awal waktu saat dieksekusi hingga akhir. Dalam penelitian ini, hasil running time dari proses klasifikasi yang dilakukan oleh system dapat dilihat pada *interface* spark *cluster*.

Dalam proses *training* SVM, data training didistribusikan oleh *master node* ke 4 *worker node* untuk dilakukan pemrosesan paralel. Selanjutnya pada Apache Spark digunakan library *machine learning* yaitu Spark MLlib yang menyediakan implementasi SVM yang dapat dijalankan di Spark *Cluster*. MLlib tersebut yang melatih model dan menerapkan SVM. Setiap *worker node* menjalankan algoritma pelatihan SVM pada bagian data yang diterima. Hasil dari proses tersebut dikumpulkan kembali oleh *master node* untuk digabungkan menjadi satu model SVM.

B. Distribusi Data pada Node

Data awal yang akan diproses di Spark *Cluster* didistribusikan dan disimpan pada penyimpanan lokal di setiap *node* yang ada pada *cluster*. Hal ini dilakukan agar setiap node dapat membaca data yang akan diproses oleh spark. Data tersebut terdiri dari tiga ukuran data yaitu 1gb, 1,5gb, dan 2gb. Pada penelitian ini untuk data 1gb disimpan pada file data2.csv, data 1,5gb disimpan pada file data3.csv, dan data 2gb disimpan pada file data4.csv. Setiap terjadi pergantian pengujian data, pada pembacaan file juga terdapat perubahan pembacaan file csv sesuai dengan file yang akan diuji di dalam program. Dalam mengatasi keterbatasan penyimpanan lokal dan memastikan data dapat diakses oleh setiap *node* dalam *cluster* dapat menggunakan penyimpanan terdistribusi seperti HDFS (*Hadoop Distributed File System*).

C. PreProcessing Data

C.1. Seleksi Fitur

Dalam seleksi fitur, pemelihan fitur didasarkan pada fitur yang memiliki kontribusi tinggi terhadap keterlambatan kedatangan penerbangan yaitu [19], *Origin Airport Id, Day of Month, Day of Week, Taxi-in Time, Air Time, Carrier Delay, Weather Delay, NAS Delay*. Untuk label dari klasifikasi ini terdapat pada kolom *ArrDel15*, dengan nilai 0 sebagai data dengan kategori *not delay* dan

nilai 1 sebagai kategori delay. Contoh data *not delay* terdapat pada Tabel I dan contoh data *delay* terdapat pada Tabel II

TABLE I. CONTOH DATA NOT DELAY

ArrDel 15	Origin Airport Id	Day Of Month	Day Of Week	Taxi-In Time	Air Time	Carrier Delay	Weither Delay	NAS Delay
0.0	10146	23	2	38	7	NULL	NULL	NULL
0.0	10146	24	3	36	12	NULL	NULL	NULL

TABLE II. CONTOH DATA DELAY

ArrDel 15	Origin Airport Id	Day Of Month	Day Of Week	Taxi-In Time	Air Time	Carrier Delay	Weither Delay	NAS Delay
1.0	10146	28	7	37	13	0	0	22
1.0	10397	3	3	32	3	22	0	0

C.2. Pembersihan Data

Pada tahap pembersihan data, dilakukan penghapusan nilai null dalam dataset. Namun sebelumnya, untuk fitur *Carrier Delay*, *Weather Delay*, *NAS Delay*, nilai null diganti dengan nilai 0 dalam data Hal ini dilakukan untuk menjaga konsistensi data dan memungkinkan analisis terhadap faktor-faktor penyebab penundaan penerbangan lainnya. Berikut merupakan hasil dari pembersihan data pada Tabel III.

TABLE III. CONTOH HASIL PEMBERSIHAN DATA

ArrDel 15	Origin Airport Id	Day Of Month	Day Of Week	Taxi-In Time	Air Time	Carrier Delay	Weither Delay	NAS Delay
0.0	10146	23	2	38	7	0	0	0
0.0	10146	24	3	36	12	0	0	0
1.0	10146	28	7	37	13	0	0	22
1.0	10397	3	3	32	3	22	0	0

C.3. Penyeimbangan Data

Pada tahap penyeimbangan data ini , terlebih dahulu dilakukan perhitungan jumlah data pada kelas mayoritas yaitu kelas *not delay* dan kelas minoritas yaitu kelas *delay*. Setelah mendapatkan hasil jumlah data dari kedua kelas, selanjutnya menghitung ratio antara jumlah data kelas mayoritas dan minoritas. Rasio tersebut digunakan sebagai acuan untuk melakukan *oversampling*. Tahapan *random oversampling* yang dilakukan setiap pengujian sehingga mempengaruhi hasil akurasi akhir dari jalannya program. Berikut merupakan hasil perubahan jumlah data dari proses *oversampling* pada Tabel IV.

TABLE IV. TABEL JUMLAH DATA SEBELUM DAN SESUDAH OVERSAMPLING

Ukuran Data	Jumlah Data			
	Sebelum		Sesudah	
	Not Delay	Delay	Not Delay	Delay
1 GB	2117931	543853	2117931	1630369
1,5 GB	2944900	738890	2944900	2214522
2 GB	4009352	949683	4009352	37997959

C.4. Normalisasi Data

Pada tahap normalisasi data, data hasil *oversampling* dinormalisasikan menggunakann *MinMaxScaler*. Rentan hasil dari normalisasi data berada pada skala 0 hingga 1. Berikut merupakan contoh hasil dari normalisasi data pda Tabel V.

TABLE V. CONTOH HASIL NORMALISASI DATA

ArrDel 15	Origin Airport Id	Day Of Month	Day Of Week	Taxi-In Time	Air Time	Carrier Delay	Weither Delay	NAS Delay
0.0	0.002	0.733	0.167	0.661	0.03	0	0	0
0.0	0.002	0.767	0.333	0.659	0.046	0	0	0
1.0	0.002	0.9	1	0.66	0.05	0	0	0.014
1.0	0.043	0.067	0.333	0.657	0.012	0.008	0	0

C.5. Pembagian Data

Dataset hasil *oversampling* akan dibagi menjadi data *training* dan *testing* dengan rasio 70% untuk data *training* dan 30% untuk data *testing*. Proses pembagian data dilakukan menggunakan metode *.randomSplit()*. Selain itu, pemberian nilai *seed* 42 dilakukan agar menghasilkan pembagian yang konsisten.

D. Hasil Pengujian

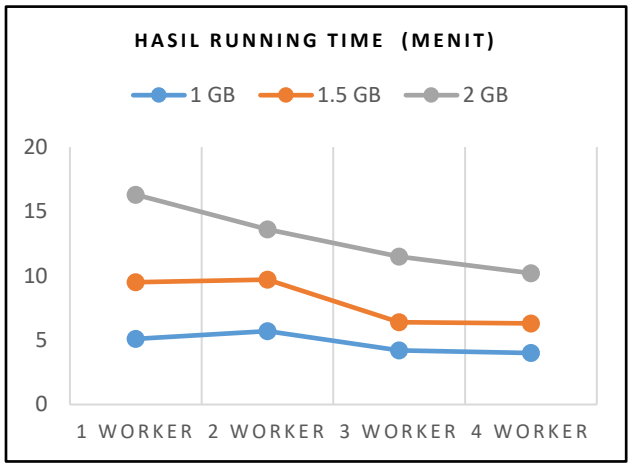
Dalam membangun model SVM menggunakan Spark, model diuji dengan mengubah penggunaan ukuran dataset dan jumlah *node* pada *cluster* spark yang telah dibangun. Dari pengujian tersebut didapatkan hasil evaluasi akurasi dan *running time* program. Pengujian jumlah *worker node* dilakukan secara berurutan yang terdiri dari 1 hingga 4 *worker node*. Sedangkan penentuan ukuran dataset berada pada nilai 1gb hingga 2gb dengan penambahan 500mb pada setiap pergantian nilai.

TABLE VI. HASIL PENGUJIAN RUNNING TIME

Ukuran Dataset	*Running time* (min)			
	1 Worker	2 Worker	3 Worker	4 Worker
1 GB	5.1	5.7	4.2	**34.0**
1,5 GB	9.5	9.7	6.4	**6.3**
2 GB	16.3	13.6	11.5	**10.2**

Pada Tabel VI menampilkan evaluasi rata-rata hasil *running time* dari 3 kali pengujian yang dihasilkan oleh sistem yang menjalankan klasifikasi kedatangan penerbangan tertunda dengan metode SVM terdistribusi apache spark dalam berbagai jumlah *worker node*. Hasil *running time* dipilih dari pengujian yang mampu menjalankan program dari pembuatan *session* spark hingga penghentian *session*. Terdapat beberapa pengujian dimana program tidak selesai dijalankan hingga penghentian *session* spark. Pengujian ini tidak dimasukkan dalam perhitungan rata-rata hasil *running time*. Kegagalan ini disebabkan oleh terputusnya koneksi kernel jupyter sehingga mengakibatkan program tidak dapat menyelesaikan eksekusinya. Hasil *running time* pada Tabel VI tersebut menunjukkan performa yang paling optimal

pada penggunaan 4 *worker node*. Penggunaan 4 *worker node* mengalami peningkatan kapasitas pemrosesan dan pengoptimalan alokasi sumber daya yang tersedia sehingga menghasilkan *running time* yang lebih efisien dari lainnya.



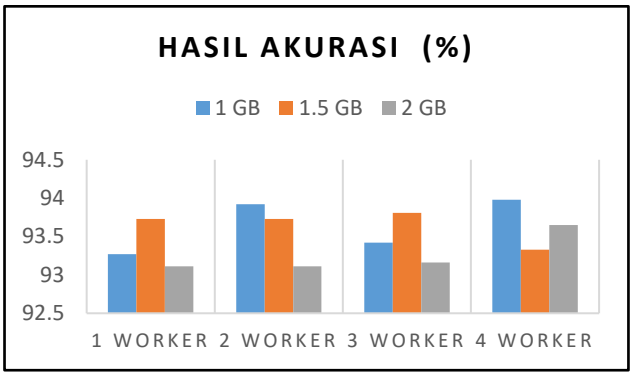
Gambar. 6. Hasil Pengujian Running Time

Pada Gambar 6 terdapat grafik hasil evaluasi *running time* dari klasifikasi penundaan kedatangan penerbangan menggunakan algoritma SVM. Berdasarkan grafik tersebut didapatkan bahwa *running time* terendah didapatkan dengan penggunaan 4 *worker node*. Peningkatan jumlah *node* pada pengujian menunjukkan peningkatan terhadap kinerja Spark hingga 4 *worker node* pada data berukuran 2gb dengan hasil *running time* yang semakin kecil setiap penambahan jumlah *worker node*. Namun pada data berukuran 1 gb dan 1,5 gb, penggunaan 2 *worker node* terdapat sedikit perbedaan dan peningkatan hasil rata-rata *running time* dari penggunaan 1 *worker node*. Hal ini berarti bahwa kinerja spark pada penggunaan 2 *worker node* tidak mengalami peningkatan. Penggunaan 2 *worker node* memiliki kinerja yang sama dengan 1 *worker node* dengan hasil *running time* yang dihasilkan berada pada kisaran yang sama dengan *running time* pada penggunaan 1 *worker node*. Namun disaat penggunaan 3 *worker node* dan 4 *worker node* hasil *running time* mengalami penurunan yang berarti kinerja Spark mengalami peningkatan. Berdasarkan keseluruhan pengujian didapatkan bahwa penggunaan 4 *worker node* merupakan penggunaan yang paling optimal pada klasifikasi penundaan kedatangan penerbangan menggunakan SVM terdistribusi pada Spark.

TABLE VII. HASIL PENGUJIAN AKURASI

Ukuran Dataset	Akurasi (%)			
	1 Worker	2 Worker	3 Worker	4 Worker
1 GB	93.27	93.92	93.42	93.98
1,5 GB	93.73	93.73	93.81	93.33
2 GB	93.11	93.11	93.16	93.65

Pada Tabel VII tersebut menampilkan evaluasi akurasi yang dihasilkan oleh sistem yang menjalankan klasifikasi penundaan penerbangan dengan metode SVM terdistribusi Apache Spark dalam berbagai jumlah *worker node*. Penggunaan jumlah *worker node* yang berbeda tidak mempengaruhi hasil akurasi dari model yang telah dibuat. Namun hasil akurasi tidak mengikuti pola perbandingan yang konsisten di berbagai penggunaan ukuran *worker node*. Perbedaan hasil akurasi tidak terkait dengan jumlah *worker node* dalam Spark, tetapi disebabkan oleh variasi dalam pemilihan dan penambahan sampel secara acak saat menjalankan *random oversampling* pada data. Proses *oversampling* ini dilakukan setiap kali program dijalankan pada pengujian. Hal tersebut mengakibatkan perbedaan akurasi yang didapatkan program pada pengujian.



Gambar. 7. Hasil Pengujian Akurasi

Pada Gambar 7 tersebut tersebut menampilkan grafik dari hasil akurasi model dalam melakukan klasifikasi penundaan penerbangan dengan metode SVM. Hasil akurasi tersebut menunjukkan bahwa akurasi yang dihasilkan oleh program tidak memiliki nilai yang sama. Perbedaan hasil akurasi terjadi dikarenakan proses *random oversampling* yang menambahkan sampel secara acak pada data, dan proses ini dilakukan pada setiap pengujian. Namun pada seluruh penggunaan variasi *worker node*, performa model tidak mengalami perubahan yang signifikan. Dengan demikian metode SVM terdistribusi menggunakan Apache Spark memiliki kemampuan yang baik dalam memanfaatkan paralelisasi dalam komputasi, sehingga meningkatkan efisiensi dan konsistensi pada hasil akurasi.

Berdasarkan hal tersebut penggunaan 4 *worker node* merupakan jumlah *worker node* yang paling efisian pada implementasi SVM pada Spark dalam melakukan klasifikasi penerbangan tertunda. Model yang telah dibuat berhasil melakukan klasifikasi pada kedatangan penerbangan tertunda dengan tingkat akurasi tertinggi pada pengujian yaitu sebesar 93.98%. Hal ini menunjukkan bahwa penggunaan metode SVM terdistribusi pada Spark dapat digunakan dalam melakukan klasifikasi kedatangan penerbangan tertunda.

V. KESIMPULAN DAN SARAN

A. Kesimpulan

Berdasarkan hasil penelitian penggunaan Spark pada klasifikasi penundaan kedatangan penerbangan dengan menggunakan SVM didapatkan beberapa kesimpulan antara lain :

1. Pada klasifikasi penerbangan, Spark dengan *library* MLlib dapat digunakan untuk membangun model SVM yang dapat memprediksi penundaan kedatangan penerbangan. Penggunaan algoritma SVM dengan Spark cukup memberikan performa yang baik dalam proses klasifikasi. Model yang telah dibuat berhasil melakukan prediksi pada penundaan kedatangan penerbangan dengan tingkat akurasi tertinggi pada pengujian sebesar 93,98%.
2. Penambahan jumlah *worker node* dapat mempengaruhi hasil *running time* dari jalannya program. Jumlah dataset yang besar juga membutuhkan sumber daya komputasi yang lebih besar. Performa dan kecepatan *running time* dapat dipengaruhi oleh banyak faktor yaitu, algoritma yang digunakan, ukuran data dan ketersediaan sumber daya. Peningkatan atau penurunan jumlah *worker node* dapat disesuaikan dengan beberapa faktor tersebut. Namun dalam hal akurasi, penambahan jumlah *worker node* tidak berpengaruh terhadap akurasi dari program. Pada proses klasifikasi penerbangan tertunda menggunakan SVM, penggunaan jumlah *worker node* yang optimal adalah 4 *worker node*.

B. Saran

Berdasarkan implementasi penggunaan spark pada klasifikasi penundaan kedatangan penerbangan dengan menggunakan SVM terdistribusi, terdapat beberapa saran yang dapat membantu pengembangan sistem.

1. Penyimpanan dataset dapat menggunakan HDFS. Penyimpanan ini dapat meningkatkan efisiensi dan efektivitas dalam penyebaran data pada *cluster*.
2. Peningkatan jumlah *worker node* dapat dilakukan untuk mengetahui titik jenuh penggunaan *worker node* pada *cluster* Spark saat menggunakan SVM terdistribusi dalam klasifikasi penundaan kedatangan penerbangan.

DAFTAR PUSTAKA

[1] Y. K. Gupta and S. Kumari, "A study of big data analytics using apache spark with python and scala," in *Proceedings of the 3rd International Conference on Intelligent Sustainable Systems, ICISS 2020*, Institute of Electrical and Electronics Engineers Inc., Dec. 2020, pp. 471–478. doi: 10.1109/ICISS49785.2020.9315863.

[2] J. Resti and F. Selva Jumeilah, "Penerapan Support Vector Machine (SVM) untuk Pengkategorian Penelitian," *JURNAL RESTI*, vol. 1, no. 1, pp. 2580–0760, 2017, [Online]. Available: http://jurnal.iaii.or.id

[3] A. A. Kurniawan, M. Mustikasari, and P. Korespondensi, "Evaluasi Kinerja Mllib Apache Spark Pada Klasifikasi Berita Palsu Dalam Bahasa Indonesia," vol. 9, no. 3, 2022, doi: 10.25126/jtiik.202293538.

[4] M. I. Putri and I. Kharisudin, "Penerapan Synthetic Minority Oversampling Technique (SMOTE) Terhadap Analisis Sentimen Data Review Pengguna Aplikasi Marketplace Tokopedia," *PRISMA, Prosiding Seminar Nasional Matematika*, vol. 5, pp. 759–766, 2022, [Online]. Available: https://journal.unnes.ac.id/sju/index.php/prisma/

[5] I. K. Nti, J. A. Quarcoo, J. Aning, and G. K. Fosu, "A mini-review of machine learning in big data analytics: Applications, challenges, and prospects," *Big Data Mining and Analytics*, vol. 5, no. 2. Tsinghua University Press, pp. 81–97, Jun. 01, 2022. doi: 10.26599/BDMA.2021.9020028.

[6] T. Tekdogan and A. Cakmak, "Benchmarking Apache Spark and Hadoop MapReduce on Big Data Classification," 2021. doi: https://doi.org/10.1145/3481646.3481649.

[7] R. Purnomo, W. Priatna, and T. D. Putra, "Implementasi Big Data Analytical Untuk Perguruan Tinggi Menggunakan Machine Learning," *Journal of Information and Information Security (JIFORTY)*, vol. 2, no. 1, p. 77, 2021, [Online]. Available: https://archive.ics.uci.edu

[8] Chang Liu, Bin Wu, Yi Yang, and Zhihong Guo, "Multiple Submodels Parallel Support Vector Machine on Spark," *IEEE International Conference on Big Data*, 2016.

[9] A. M. Ryanto, A. A. Ilham, and M. Niswar, "Analisis Kinerja Framework Big Data Pada Cluster Tervirtualisasi: Hadoop Mapreduce Dan Apache Spark," 2017.

[10] M. Hariadi, M. Sc, P. D. Ir Mauridhi Hery Purnomo, M. Eng, and P. D. Program Magister Bidang Keahlian Jaringan Cerdas Multimedia Jurusan Teknik Elektro Fakultas, "KLASIFIKASI AIR SUNGAI BERBASIS KOMBINASI TEKNOLOGI IOT-BIG DATA MENGGUNAKAN SVM RIZQI PUTRI NOURMA BUDIARTI NRP 2214205202 DOSEN PEMBIMBING," 2017.

[11] Z. H. You, J. Z. Yu, L. Zhu, S. Li, and Z. K. Wen, "A MapReduce based parallel SVM for large-scale predicting protein-protein interactions," *Neurocomputing*, vol. 145, pp. 37–43, Dec. 2014, doi: 10.1016/j.neucom.2014.05.072.

[12] R. Arisandi, "PERBANDINGAN MODEL KLASIFIKASI RANDOM FOREST DENGAN RESAMPLING DAN TANPA RESAMPLING PADA PASIEN PENDERITA GAGAL JANTUNG," *Jurnal Gaussian*, vol. 12, no. 1, pp. 136–145, May 2023, doi: 10.14710/j.gauss.12.1.136-145.

[13] S. Diantika, “PENERAPAN TEKNIK RANDOM OVERSAMPLING UNTUK MENGATASI IMBALANCE CLASS DALAM KLASIFIKASI WEBSITE PHISHING MENGGUNAKAN ALGORITMA LIGHTGBM,” 2023.

[14] A. Riski Indra Pratama *et al.*, “OPTIMASI KLASIFIKASI CURAH HUJAN MENGGUNAKAN SUPPORT VECTOR MACHINE (SVM) DAN RECURSIVE FEATURE ELIMINATION (RFE).”

[15] D. Ibrahim, “Analisis Hubungan antar Faktor dan Komparasi Algoritma Klasifikasi pada Penentuan Penundaan Penerbangan,” 2017. [Online]. Available: http://conference.poltektegal.ac.id/index.php/senit2017

[16] H. Rezkian, A. Dama, A. A. Supianto, and N. Y. Setiawan, “Analisis Penggunaan Model Regresi untuk Prediksi Penjualan Spare Part pada AHASS Nur Andhita Grogol,” 2021. [Online]. Available: http://j-ptiik.ub.ac.id

[17] J. I. Matematika and S. Adi, “MATHunesa Tahun 2022 KOMPARASI METODE SUPPORT VECTOR MACHINE (SVM), K-NEAREST NEIGHBORS (KNN), DAN RANDOM FOREST (RF) UNTUK PREDIKSI PENYAKIT GAGAL JANTUNG Atik Wintarti”.

[18] R. Mulla, “Flight Status Prediction,” 2018. https://www.kaggle.com/datasets/robikscube/flight-delay-dataset-20182022 (accessed Mar. 30, 2023).

[19] D. B. Bisandu, I. Moulitsas, and S. Filippone, “Social ski driver conditional autoregressive-based deep learning classifier for flight delay prediction,” *Neural Comput Appl*, vol. 34, no. 11, pp. 8777–8802, Jun. 2022, doi: 10.1007/s00521-022-06898-y.