



PROYEK AKHIR

SISTEM MONITORING PERAWATAN DAN PREDIKSI KERUSAKAN PADA MESIN *CONVEYOR* MENGGUNAKAN *DEEP LEARNING*

Ekv Bintarno Wicaksono
NRP. 2210171023

Dosen Pembimbing 1:
Dwi Kurnia Basuki, S.Si., M.Kom.
NIP. 197404102008011014

Dosen Pembimbing 2:
Bayu Sandi Marta, S.ST., M.T.
NIP. 198903262015041001

Dosen Pembimbing 3:
Sritrusta Sukaridhoto, S.T., Ph.D.
NIP. 197903062002121002

PROGRAM STUDI D4 TEKNIK KOMPUTER
DEPARTEMEN TEKNIK INFORMATIKA DAN KOMPUTER
POLITEKNIK ELEKTRONIKA NEGERI SURABAYA
2021



PROYEK AKHIR

SISTEM MONITORING PERAWATAN DAN PREDIKSI KERUSAKAN PADA MESIN CONVEYOR MENGGUNAKAN *DEEP LEARNING*

Ekv Bintarno Wicaksono
NRP. 2210171023

Dosen Pembimbing 1:
Dwi Kurnia Basuki, S.Si., M.Kom.
NIP. 197404102008011014

Dosen Pembimbing 2:
Bayu Sandi Marta, S.ST., M.T.
NIP. 198903262015041001

Dosen Pembimbing 3:
Sritrusta Sukaridhoto, S.T., Ph.D.
NIP. 197903062002121002

**PROGRAM STUDI D4 TEKNIK KOMPUTER
DEPARTEMEN TEKNIK INFORMATIKA DAN KOMPUTER
POLITEKNIK ELEKTRONIKA NEGERI SURABAYA
2021**

**SISTEM MONITORING PERAWATAN DAN PREDIKSI
KERUSAKAN PADA MESIN CONVEYOR MENGGUNAKAN
DEEP LEARNING**

Oleh :

Ekv Bintarno Wicaksono

NRP. 2210171023

Proyek Akhir ini Digunakan Sebagai Salah Satu Syarat Untuk
Memperoleh Gelar Sarjana Terapan Teknik (S.Tr.T.)
di

Politeknik Elektronika Negeri Surabaya
2021

Disetujui Oleh :

Tim Penguji Proyek Akhir:

Dosen Pembimbing :

1. **Iwan Kurnianto Wibowo, S.ST., M.T.**
NIP. 198712302014041001

1. **Dwi Kurnia Basuki, S.Si., M.Kom.**
NIP. 197404102008011014

2. **Dr. Eng. Bima Sena Bayu Dewantara,**
S.ST., M.T.
NIP. 197612151999032002

2. **Bayu Sandi Marta, S.ST., M.T.**
NIP. 198903262015041001

3. **Reni Soelistijorini, B.Eng., MT.**
NIP. 197104281999032002

3. **Sritrusta Sukaridhoto, S.T., Ph.D.**
NIP. 196605211997022001

Mengetahui

Ketua Program Studi D4 Teknik Komputer
Departemen Teknik Informatika dan Teknik Komputer
Politeknik Elektronika Negeri Surabaya

Rivanto Sigit, S.T., M.Kom., Ph.D.
NIP197008111995121001

-----Halaman ini sengaja dikosongkan-----


LEMBAR PERNYATAAN BEBAS PLAGIARISME

Saya yang bertanda tangan dibawah ini dengan sebenarnya menyatakan bahwa Proyek Akhir ini saya susun tanpa tindakan plagiarisme sesuai dengan peraturan yang berlaku di Politeknik Elektronika Negeri Surabaya (PENS)

Nama : Eky Bintarno Wicaksono
NRP : 2210171023
Program Studi : Teknik Komputer
Departemen : Teknik Informatika dan Komputer

Jika dikemudian hari saya terbukti melakukan tindakan plagiarisme, saya akan bertanggung jawab sepenuhnya dan menerima sanksi yang dijatuhkan oleh PENS kepada saya.

Surabaya, 16 Juli 2021



Eky Bintarno Wicaksono
NRP. 2210171023

-----Halaman ini sengaja dikosongkan-----

KATA PENGANTAR

Segala puji syukur kepada Tuhan Yang Maha Esa, karena atas segala rahmat dan penyertaan-Nya, penulis dapat menyelesaikan proyek akhir yang berjudul:

SISTEM MONITORING PERAWATAN DAN PREDIKSI KERUSAKAN PADA MESIN CONVEYOR MENGGUNAKAN *DEEP LEARNING*

Buku Proyek Akhir ini disusun dengan maksud dan tujuan untuk memenuhi persyaratan guna menyelesaikan studi Diploma IV dan memperoleh gelar Sarjana Terapan Teknik (S.Tr.T.) di Politeknik Elektronika Negeri Surabaya. Buku ini telah dikerjakan dengan mengerahkan seluruh kemampuan dan konsentrasi penulis, serta tidak lepas dari dukungan dosen pembimbing serta pihak-pihak lain yang telah banyak memberikan semangat dan bantuan.

Penulis menyadari bahwa buku ini masih memiliki banyak sekali kekurangan karena Proyek Akhir ini dikerjakan dalam masa Pandemi *COVID-19*, dimana mahasiswa maupun dosen pembimbing diwajibkan bekerja dari rumah (*Work From Home*) sehingga terdapat keterbatasan baik dalam hal pengambilan data, pengujian maupun dalam kegiatan bimbingan. Saya selaku penulis buku proyek akhir ini, sempat terkena *COVID-19* selama 2 minggu. Oleh karena itu hasil dari laporan Proyek Akhir ini masih belum bisa maksimal. Maka dari itu penulis memohon maaf sebesar besarnya atas kekurangan yang ada pada buku ini. Penulis sangat mengharapkan saran dan kritik dari semua pihak, serta berharap agar buku proyek akhir ini dapat bermanfaat khususnya bagi penulis dan pembaca pada umumnya.

Akhir kata, penulis mengucapkan terima kasih kepada seluruh pihak yang telah membantu selama pengerjaan proyek akhir ini.

Surabaya, 16 Juli 2021

Penulis

-----Halaman ini sengaja dikosongkan-----

UCAPAN TERIMA KASIH

Segala puji syukur saya panjatkan kepada Allah SWT, karena berkat rahmat dan hidayah-Nya maka proyek akhir ini dapat diselesaikan. Halaman ini didedikasikan sepenuhnya untuk menyampaikan rasa terima kasih yang tak terhingga kepada seluruh pihak yang senantiasa memberikan dukungan dan bantuan kepada penulis dalam seluruh rangkaian pengerjaan proyek akhir. Penulis menyampaikan terima kasih banyak kepada:

1. **Orang tua** tercinta, Bapak **Basuki Sugiarto** dan Ibu **Endang Sriharwati** yang tiada henti-hentinya mendoakan, selalu menguatkan dan memberikan dukungan kepada saya hingga dapat menyelesaikan studi di PENS.
2. **Keluarga** tercinta, khususnya kakak sepupu saya, yang memberikan dukungan dan semangat hingga proyek akhir ini selesai.
3. Bapak **Ali Ridho Barakbah, S.Kom, Ph.D.** selaku Direktur Politeknik Elektronika Negeri Surabaya beserta jajaran direksi lainnya
4. Bapak **Dwi Kurnia Basuki, S.Si., M.Kom.** selaku pembimbing pertama Proyek Akhir saya yang telah memberi banyak saran dan masukan untuk menyelesaikan proyek akhir saya.
5. Bapak **Bayu Sandi Marta, S.ST., M.T.** selaku pembimbing kedua Proyek Akhir saya yang selalu senantiasa memberikan saya nasihat, saran dan masukan untuk menyelesaikan proyek akhir.
6. Bapak **Sritrusta Sukaridhoto, ST., Ph.D.** selaku pembimbing ketiga yang telah mengizinkan saya untuk bergabung dalam “Lab Duafa Lt 9”, beliau juga senantiasa memberi saran, masukan dan ilmu untuk proyek akhir saya.
7. Bapak **Hary Setyo Wahyudi**, selaku pembimbing dari PT. JAI yang telah memberikan banyak bantuan selama pengerjaan proyek akhir ini hingga selesai.
8. Seluruh Dosen Penguji mulai dari sidang TPPA hingga PA yang memberikan kritik dan saran untuk proyek akhir saya.
9. Seluruh Dosen Teknik Komputer PENS yang telah memberikan bekal ilmu selama 4 tahun perkuliahan.

10. Seluruh teman-teman **Teknik Komputer Angkatan 2017** yang telah membantu, mendampingi, dan berbagi cerita dan kebahagiaan dengan saya selama menjalani perkuliahan di PENS selama 4 tahun.

ABSTRAK

PT. Jatim Autocomp Indonesia atau disebut PT. JAI merupakan perusahaan yang bergerak di bidang produksi *wiring harness*. Untuk menunjang proses produksi, PT. JAI menggunakan berbagai mesin salah satunya *conveyor*. Penggunaan *conveyor* secara terus menerus dapat menyebabkan kerusakan yang berdampak pada proses produksi yang terganggu. Ketika terjadi kerusakan, *Group Leader* sebagai penanggungjawab membuat laporan kerusakan yang terdiri dari jenis dan detail kerusakan untuk dilakukan perbaikan. Tetapi dari laporan tersebut, pihak management PT. JAI tidak dapat memprediksi kerusakan mesin tersebut berdasarkan rekam jejak laporan kerusakan mesin. Pada proyek akhir ini, salah satu solusi dari permasalahan tersebut yaitu menggunakan *Deep Learning* dengan metode *Long Short-Term Memory*. Sistem ini bekerja berdasarkan data laporan kerusakan dan laporan perawatan pada mesin conveyor dari Januari 2016 hingga Juli 2020. Sistem melakukan *Pre-Processing* untuk mengubah kata menjadi angka *array* menggunakan *CountVectorizer* dan *Label Binarizer* untuk memberikan *label* pada kolom *dataset*. Selanjutnya data tersebut dibagi menjadi *data training* dan *data testing*, dilanjutkan dengan proses *Build* dan *Compile* model *Deep Learning* untuk membuat model *Deep Learning* yang digunakan dan melakukan *training dataset* kerusakan dan klasifikasi kerusakan mesin conveyor, yang selanjutnya dilakukan evaluasi akurasi model *Deep Learning* dan melakukan proses prediksi kerusakan mesin conveyor. Pada proyek akhir ini, dilakukan pengujian terhadap 1500 *data training* dan 375 *data testing*. Hasil menunjukkan sistem dapat memprediksi dan mendapatkan akurasi 97,71%, rata-rata nilai akurasi dengan *KFold* 95,66%, akurasi dengan *confusion matrix* 93,33% dan akurasi dengan *ROC* 99,19%.

Kata kunci: *Deep Learning*, Prediksi, Kerusakan, *Long Short-Term Memory*

-----Halaman ini sengaja dikosongkan-----

ABSTRACT

PT. Jatim Autocomp Indonesia or called PT. JAI is a company engaged in the production of wiring harness. To support the production process, PT. JAI uses various machines, one of which is a conveyor. Continuous use of conveyors can cause damage that has an impact on disrupted production processes. When damage occurs, the Group Leader as the person in charge makes a damage report consisting of the type and details of the damage to be repaired. But from the report, the management of PT. JAI cannot predict the engine failure based on the track record of engine failure reports. In this final project, one solution to this problem is to use Deep Learning with the Long Short-Term Memory method. This system works based on data from damage reports and maintenance reports on conveyor machines from January 2016 to July 2020. The system performs Pre-Processing to convert words into array numbers using CountVectorizer and Label Binarizer to label the dataset columns. Furthermore, the data is divided into training data and testing data, followed by the Build and Compile Deep Learning model process to create a Deep Learning model that is used and conduct training on the damage dataset and classification of conveyor machine damage, which is then evaluated for the accuracy of the Deep Learning model and performs the prediction process. conveyor failure. In this final project, 1500 training data and 375 testing data were tested. The results show the system can predict and get 97.71% accuracy, the average accuracy value with KFold 95.66%, accuracy with confusion matrix 93.33% and accuracy with ROC 99.19%.

Keywords: Deep Learning, Prediction, Damage, Long Short-Term Memory

-----Halaman ini sengaja dikosongkan-----

DAFTAR ISI

LEMBAR PERNYATAAN BEBAS PLAGIARISME.....	v
KATA PENGANTAR	vii
UCAPAN TERIMA KASIH.....	ix
ABSTRAK.....	xi
ABSTRACT.....	xiii
DAFTAR ISI.....	xv
DAFTAR GAMBAR	xix
DAFTAR TABEL.....	xxi
DAFTAR <i>SOURCE</i>	xxiii
BAB I PENDAHULUAN	1
1.1 Latar Belakang.....	1
1.2 Rumusan Masalah.....	2
1.3 Batasan Masalah	3
1.4 Tujuan dan Manfaat	3
1.5 Metodologi Penelitian.....	4
BAB II KAJIAN PUSTAKA DAN DASAR TEORI	7
2.1 Kajian Pustaka	7
2.1.1 Prediksi Kerusakan Mesin	7
2.2 Dasar Teori	8
2.2.1 Conveyor.....	8
2.2.2 Kerusakan Conveyor.....	12
2.2.4 <i>Python</i>	15
2.2.5 <i>Keras</i>	15
2.2.6 <i>Pandas</i>	16
2.2.7 <i>Skicit-Learn</i>	16

2.2.8	<i>CountVectorizer</i>	16
2.2.9	<i>MinMaxScaler</i>	16
2.2.10	<i>Label Binarizer</i>	17
2.2.11	<i>Deep Learning</i>	17
2.2.12	<i>Long Short-Term Memory</i>	18
2.2.13	<i>KFold Cross Validation</i>	20
2.2.14	<i>Confusion Matrix</i>	20
2.2.15	<i>Receiver Operating Characteristics</i>	22
BAB III PERANCANGAN DAN IMPLEMENTASI SISTEM		25
3.1	Perancangan Sistem	25
3.2	Pengambilan data di PT. Jatim Autocomp Indonesia	29
3.3	<i>Input data text</i>	31
3.4	<i>Pre-Processing Data Text</i>	31
3.4.1	<i>CountVectorizer</i>	31
3.4.2	<i>MinMaxScaler</i>	32
3.4.2	<i>Label Binarizer</i>	33
3.5	<i>Split Dataset menjadi Data Training dan Data Testing</i>	33
3.6	<i>Build dan compile model Deep Learning</i>	34
3.7	<i>Training model Deep Learning</i>	35
3.8	Membuat Prediksi Kerusakan Mesin <i>Conveyor</i>	37
3.9	Evaluasi model <i>Deep Learning</i>	38
3.9.1	Evaluasi Akurasi model <i>Deep Learning</i>	38
3.9.2	Evaluasi model <i>Deep Learning</i> dengan metode <i>KFold</i>	38
3.9.3	Evaluasi <i>Confusion Matrix</i> dari model <i>Deep Learning</i>	40
3.9.4	Evaluasi <i>ROC</i> dari model <i>Deep Learning</i>	42
3.10	<i>User Interface</i>	44

3.10.1 Perubahan Hasil Prediksi menjadi Tulisan Prediksi dan Rekomendasi Tindakan.....	44
3.10.2 Pembuatan <i>User Interface</i>	45
BAB IV PENGUJIAN DAN ANALISA	49
4.1 Pengujian <i>Input data text</i>	49
4.2 Pengujian <i>Pre-Processing Data Text</i>	49
4.2.1 Pengujian <i>CountVectorizer</i>	50
4.2.2 Pengujian <i>MinMaxScaler</i>	50
4.2.3 Pengujian <i>Label Binarizer</i>	51
4.3 Pengujian <i>Split Dataset</i> menjadi <i>Data Training</i> dan <i>Data Testing</i>	52
4.4 Pengujian <i>Build</i> dan <i>Compile</i> model <i>Deep Learning</i>	55
4.5 Pengujian <i>Training</i> model <i>Deep Learning</i>	56
4.6 Pengujian Membuat Prediksi Kerusakan Mesin <i>Conveyor</i>	58
4.7 Pengujian Evaluasi model <i>Deep Learning</i>	60
4.7.1 Pengujian Evaluasi akurasi model <i>Deep Learning</i>	61
4.7.2 Pengujian Evaluasi akurasi model <i>Deep Learning</i> dengan metode <i>KFold</i>	61
4.7.3 Pengujian Evaluasi <i>Confusion Matrix</i> model <i>Deep Learning</i>	62
4.7.4 Pengujian Evaluasi <i>ROC</i> model <i>Deep Learning</i>	63
4.8 Pengujian <i>User Interface</i>	65
4.8.1 Pengujian Perubahan Hasil Prediksi menjadi Tulisan Prediksi dan Rekomendasi Tindakan.....	65
4.8.2 Pengujian <i>User Interface</i>	66
BAB V PENUTUP.....	69
5.1 Kesimpulan	69
5.2 Saran	69
DAFTAR PUSTAKA	71

DAFTAR RIWAYAT HIDUP	75
----------------------------	----

DAFTAR GAMBAR

Gambar 1.1 Diagram Metodologi Penelitian.....	4
Gambar 2.1 <i>Conveyor</i> di PT. Jatim Autocomp Indonesia	8
Gambar 2.2 Penampakan <i>Gear Box</i> dan <i>Electric Motor</i>	10
Gambar 2.3 Penampakan <i>Roller Chain Sprocket with Transmission Chain</i>	11
Gambar 2.4 Penampakan <i>Conveyor Chain</i>	11
Gambar 2.5 Arsitektur LSTM	18
Gambar 2.6 <i>Confusion Matrix</i>	21
Gambar 2.7 Kurva <i>ROC</i>	23
Gambar 3.1 Diagram Rancangan Sistem.....	25
Gambar 3.2 Diagram Alur Proses <i>Deep Learning</i>	27
Gambar 4.1 Hasil pengujian <i>input</i> data kerusakan dan <i>level</i> kerusakan.....	49
Gambar 4.2 Hasil pengujian <i>CountVectorizer</i>	50
Gambar 4.3 Hasil pengujian <i>MinMaxScaler</i>	51
Gambar 4.4 Hasil pengujian <i>Label Binarizer</i>	52
Gambar 4.5 Hasil pengujian <i>Data Training</i> Variabel X	53
Gambar 4.6 Hasil pengujian <i>Data Training</i> Variabel Y	53
Gambar 4.7 Hasil pengujian <i>Data Testing</i> Variabel X.....	54
Gambar 4.8 Hasil pengujian <i>Data Testing</i> Variabel Y.....	54
Gambar 4.9 Hasil proses <i>Build</i> dan <i>Compile</i> model <i>Deep Learning</i>	55
Gambar 4.10 Hasil pengujian <i>Training</i> model <i>Deep Learning</i>	57
Gambar 4.11 Hasil pengujian <i>Data testing</i> level kerusakan.....	59
Gambar 4.12 Hasil pengujian prediksi kerusakan	59
Gambar 4.13 Grafik hasil pengujian prediksi dengan model <i>Deep Learning</i>	60
Gambar 4.14 Grafik hasil pengujian model <i>Deep Learning</i>	58
Gambar 4.15 Hasil pengujian <i>Confusion Matrix</i>	62
Gambar 4.16 Hasil pengujian <i>ROC</i>	64
Gambar 4.17 Hasil pengujian Perubahan Hasil Prediksi.....	65
Gambar 4.18 Hasil pengujian <i>User Interface</i>	66

-----Halaman ini sengaja dikosongkan-----

DAFTAR TABEL

Tabel 2.1 Kerusakan Mesin Conveyor	12
Tabel 3.1 Jenis Kerusakan.....	29
Tabel 3.2 Jumlah Jenis Kerusakan, Level Kerusakan dan Rekomendasi Tindakan.....	30
Tabel 4.1 Pengujian Evaluasi Akurasi model <i>Deep Learning</i> dengan metode <i>KFold</i>	61

-----Halaman ini sengaja dikosongkan-----

DAFTAR SOURCE

Source 3.1 <i>Input Data Text</i>	31
Source 3.2 <i>CountVectorizer</i>	32
Source 3.3 <i>MinMaxScaler</i>	32
Source 3.4 <i>Label Binarizer</i>	33
Source 3.5 <i>Split Dataset</i> menjadi <i>Data Training</i> dan <i>Data Testing</i>	33
Source 3.6 <i>Build dan Compile model Deep Learning</i>	35
Source 3.7 <i>Training model Deep Learning</i>	35
Source 3.8 <i>Plot Grafik Training Deep Learning</i>	36
Source 3.9 <i>Membuat prediksi kerusakan</i>	37
Source 3.10 <i>Evaluasi akurasi model Deep Learning</i>	38
Source 3.11 <i>Evaluasi dengan metode KFold</i>	40
Source 3.12 <i>Confusion Matrix</i>	41
Source 3.13 <i>Grafik Confusion Matrix</i>	42
Source 3.14 <i>ROC</i>	42
Source 3.15 <i>Grafik Evaluasi ROC</i>	43
Source 3.16 <i>Perubahan Hasil Prediksi menjadi Tulisan Prediksi dan Rekomendasi Tindakan</i>	45
Source 3.17 <i>Inisialisasi Jendela User Interface</i>	46
Source 3.18 <i>Inisialisasi Waktu dan Tanggal</i>	46
Source 3.19 <i>Menampilkan Hasil Prediksi dalam User Interface</i>	48
Source 3.20 <i>Close Button</i>	48

-----Halaman ini sengaja dikosongkan-----

BAB I

PENDAHULUAN

1.1 Latar Belakang

Setiap perusahaan yang didirikan mempunyai tujuan dalam pelaksanaannya. Ada perusahaan yang bergerak di bidang manufaktur, perusahaan yang bergerak di bidang jasa maupun perusahaan lain yang bergerak di bidang tertentu. Perusahaan yang bergerak di bidang manufaktur memiliki mesin produksi, dimana mesin ini akan memproduksi barang yang akan dijual kepada konsumen. Dalam industri manufaktur, mesin dan peralatan merupakan sumber daya penunjang produksi yang dibutuhkan untuk menjadi kekuatan utama perusahaan dalam berlangsungnya proses produksi. Sama halnya seperti manusia, kondisi mesin dan peralatan mengalami pertambahan umur yang menyebabkan terjadinya penurunan kemampuan dalam melaksanakan tugas masing-masing [1]. Selain faktor internal, terdapat faktor eksternal yang dapat mempengaruhi terjadinya penurunan kemampuan mesin dan peralatan dalam bekerja. Antara lain kesalahan dalam pengoperasian mesin, input bahan baku yang tidak sesuai dengan kesalahan instalasi peralatan pendukung ataupun penyebab lainnya yang mengakibatkan mesin tersebut tidak dapat bekerja seperti keadaan normal [2].

Dampak mesin dan peralatan yang mengalami penurunan kemampuan karena faktor internal maupun eksternal, produktivitas dari industri tersebut akan menurun. Produktivitas dalam suatu sistem produksi merupakan masalah yang penting. Produktivitas itu sendiri dipengaruhi oleh banyak faktor, diantaranya yaitu kondisi mesin, tenaga kerja, lingkungan tempat kerja, dan lain-lain. Dengan bertambahnya usia mesin maka nilai kemampuannya cenderung menurun, sehingga pada gilirannya komponen-komponen mesin akan sering mengalami kerusakan. dan hal ini secara tidak langsung akan berakibat pada produktivitas atau kemampuan memproduksi dari mesin juga akan menurun [3].

Salah satu pabrik yang bergerak di bidang produksi barang yaitu PT. Jatim Autocomp Indonesia, atau disebut PT. JAI, dimana perusahaan ini bergerak di bidang manufaktur, yaitu *wiring harness* atau yang dikenal

sebagai kabel kelistrikan mobil. Untuk menunjang proses produksi, PT. JAI menggunakan berbagai mesin salah satunya *conveyor* yang total berjumlah 29 *conveyor*. Penggunaan *conveyor* secara terus menerus tidak lepas dari kerusakan. Ketika terjadi kerusakan, *Group Leader* sebagai penanggungjawab membuat laporan kerusakan yang terdiri dari jenis kerusakan dan detail kerusakan untuk dilakukan perbaikan. Setelah *Group Leader* melaporkan kerusakan yang terjadi, teknisi datang untuk memperbaiki kerusakan dan membuat laporan perawatan mesin tersebut. Tetapi dari laporan tersebut, pihak management PT. JAI memprediksi kerusakan mesin *conveyor* secara rinci sehingga tidak dapat diprediksi kerusakan selanjutnya berdasarkan rekam jejak laporan kerusakan mesin *conveyor* sehingga mempengaruhi kinerja produktivitas mereka.

Berbagai upaya untuk memprediksi kerusakan selanjutnya menggunakan rekam jejak laporan kerusakan telah dilakukan. Salah satunya yaitu memprediksi kerusakan menggunakan data visual sebagai persepsi untuk model *Deep Learning* menggunakan berbagai metode dan *dataset* yang digunakan yaitu gambar visualisasi mesin [4]. Upaya lain yang dilakukan yaitu memprediksi kerusakan pada pabrik manufaktur sepeda motor menggunakan *Deep Learning* [5]. Tetapi upaya tersebut belum dapat menjawab cara memprediksi kerusakan mesin *conveyor* yang akan terjadi di masa depan apabila data yang diberikan berupa data *text*, bukan berupa data gambar atau visual.

Pada proyek akhir ini dibangun sebuah sistem kecerdasan buatan yang dapat memprediksi kerusakan selanjutnya. Sistem ini dibuat dengan menggunakan metode *Deep Learning* untuk memprediksi kerusakan mesin *conveyor* yang akan terjadi di masa depan dilihat dari catatan kerusakan mesin *conveyor* pada waktu sebelumnya.

1.2 Rumusan Masalah

Permasalahan yang dihadapi pada proyek akhir ini adalah:

1. Bagaimana melakukan *pre-processing* pada data kerusakan?
2. Bagaimana menentukan *level* kerusakan berdasarkan data kerusakan mesin *conveyor* yang digunakan?
3. Bagaimana memprediksi kerusakan mesin *conveyor* yang akan terjadi di masa depan?

1.3 Batasan Masalah

Adapun batasan-batasan yang akan diterapkan dalam proyek akhir ini antara lain sebagai berikut :

1. *Conveyor* yang diuji sebanyak satu dari 29 *conveyor*.
2. Skenario simulasi berjalan sesuai masukan dari data laporan kerusakan yang dilakukan dalam bentuk data teks.
3. Skenario yang diujikan sebatas kerusakan yang akan terjadi di masa depan.
4. Pengujian simulasi sistem *monitoring* akan berada pada lingkup industri yaitu PT. Jatim Autocomp Indonesia

1.4 Tujuan dan Manfaat

Adapun tujuan dan manfaat dari proyek akhir ini adalah sebagai berikut:

1.4.1 Tujuan

Tujuan dari proyek akhir ini yaitu:

1. Melakukan *pre-processing* pada data kerusakan sebelum di-*input*-kan pada sistem *Deep Learning*.
2. Menentukan *level* kerusakan mesin *conveyor* berdasarkan data kerusakan mesin *conveyor*.
3. Sistem dapat memprediksi kerusakan yang akan terjadi di masa depan.

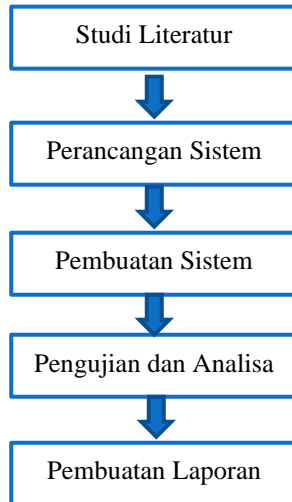
1.4.2 Manfaat

Manfaat dari proyek akhir ini yaitu:

1. Membangun aplikasi Sistem *Monitoring* yang dapat membantu pengguna dalam memprediksi kerusakan pada mesin conveyor yang akan terjadi di masa depan.
2. Memberikan kontribusi terhadap pengembangan sistem pada PT. Jatim Autocomp Indonesia.
3. Mengenalkan *Deep Learning* sebagai salah satu solusi permasalahan yang dihadapi di PT. Jatim Autocomp Indonesia.

1.5 Metodologi Penelitian

Penyelesaian sistem akan dikerjakan dalam beberapa tahap. Metodologi yang digunakan dalam pengerjaan proyek akhir adalah seperti pada Gambar 1.1.



Gambar 1.1 Diagram Metodologi Penelitian

1.5.1 Studi Literatur

Studi literatur dilakukan untuk mendapatkan referensi yang berkaitan dengan keseluruhan sistem penelitian yang terdiri dari studi tentang metode *Deep Learning* untuk memprediksi kerusakan mesin conveyor yang akan terjadi di masa depan. Studi literatur dilakukan dengan membaca buku, *paper*, jurnal ilmiah dengan media internet.

1.5.2 Perancangan Sistem

Tahap perencanaan sistem dimulai dari instalasi *IDE* dan *Compiler* untuk *Deep Learning*, kemudian melakukan *input* dataset yang terdiri dari data kerusakan dan perawatan mesin conveyor, dilanjutkan dengan pembagian *dataset* menjadi *Data Training* dan *Data Testing*, membuat dan melakukan *training* dengan model *Deep Learning*, membuat prediksi kerusakan mesin *conveyor* dan melakukan evaluasi model *Deep Learning* yang digunakan.

1.5.3 Pembuatan Sistem

Pada tahap ini dilakukan pembuatan sistem untuk pengolahan data kerusakan dan perawatan mesin conveyor dan diproses dengan Spyder 3.4 yang telah tersedia *library Keras, Pandas* dan *Skicit-learn*. Berikut proses yang dilakukan pada sistem ini:

1. Melakukan *input* dataset yang telah disiapkan.
2. Melakukan *Pre-Processing* pada *dataset*.
3. Melakukan pembagian *dataset* menjadi *Data Training* dan *Data Testing*.
4. Membuat model *Deep Learning*.
5. Melakukan *Training* dengan model *Deep Learning*.
6. Membuat prediksi kerusakan yang akan terjadi di masa depan.
7. Menghitung akurasi model *Deep Learning* yang digunakan.

1.5.4 Pengujian dan Analisa

Pada tahapan ini dilakukan uji sistem menggunakan model *Deep Learning* yang telah dibuat. Uji sistem ini terdiri dari akurasi model *Deep Learning* yang dibuat dan hasil prediksi kerusakan mesin conveyor yang akan terjadi di masa depan berdasarkan hasil model *Deep Learning* tersebut.

1.5.5 Pembuatan Laporan

Pembuatan laporan ditujukan sebagai dokumentasi dan penjelasan seluruh kegiatan dalam pembuatan proyek akhir. Selain itu juga untuk mengetahui apakah hasil telah sesuai dengan capaian yang diharapkan atau tidak. Pembuatan laporan dilaksanakan pada akhir tahapan pembuatan proyek akhir.

-----Halaman ini sengaja dikosongkan-----

BAB II

KAJIAN PUSTAKA DAN DASAR TEORI

Pada bab ini membahas mengenai teori-teori yang berkaitan dengan penyelesaian proyek akhir yang didapatkan dari berbagai sumber yang terkait.

2.1 Kajian Pustaka

2.1.1 Prediksi Kerusakan Mesin

Penelitian [6] membahas tentang penggunaan *Machine Learning* untuk memprediksi kerusakan pada komputasi kinerja tinggi dan sistem komputer *cloud*. Pendekatan yang digunakan yaitu menggunakan *Machine Learning* dimana pendekatan ini menggunakan berbagai macam metode yang digunakan yaitu *Support Vector Machine (SVM)*, *Random Forest (RF)*, *K-Nearest Neighbors (KNN)*, *Classification and Regression Trees (CART)* dan *Linear Discriminant Analysis (LDA)*. *Dataset* yang digunakan berasal dari riwayat kerusakan dalam rentang waktu 5 tahun terakhir yaitu 2001-2006 sejumlah 383 data dengan 5 sumber kerusakan yang berbeda-beda, yaitu *Hardware*, *Software*, *Human Error*, *Network*, *Undetermined*. Hasil pengujian prediksi kerusakan yang telah dilakukan, dapat dilihat bahwa metode SVM memiliki hasil akurasi yang lebih baik yaitu 90% dan menghasilkan RMSE yang lebih kecil yaitu 0,1718 (Bashir Mohammed dkk. 2019).

Penelitian [7] membahas tentang penggunaan *Machine Learning* untuk memprediksi kerusakan pada sistem pesawat terbang. Pendekatan yang digunakan yaitu *Machine Learning* dimana menggunakan berbagai macam algoritma yaitu *Multilayer Perceptron (MLP)* sebagai *Artificial Neural Network (ANN)*, *Support Vector Regression (SVR)*, dan *Linear Regression (LR)*. *Dataset* yang digunakan berasal dari riwayat kerusakan dalam rentang waktu 2 tahun terakhir sejumlah 585 data yang berasal dari salah satu perusahaan aviasi di Turki dengan 9 variabel *input* dimana variabel tersebut dapat mempengaruhi terjadinya kerusakan. Hasil pengujian yang telah dilakukan dapat dilihat bahwa metode LR lebih *powerful* daripada metode lainnya dalam memprediksi kerusakan. (Kadir Celikmih dkk. 2020).

2.2 Dasar Teori

2.2.1 Conveyor

Conveyor adalah mesin produksi yang ada di PT. Jatim Autocomp Indonesia. Conveyor ini digunakan sebagai salah satu mesin produksi manufaktur *wiring harness* atau yang disebut dengan kabel kelistrikan mobil. Kabel kelistrikan mobil yang diproduksi oleh PT. Jatim Autocomp Indonesia yaitu milik Toyota, Nissan, Subaru, dan lain-lain.



Gambar 2.1 *Conveyor* di PT. Jatim Autocomp Indonesia

Setiap *conveyor* dipimpin oleh *Group Leader (GL)*, yang merupakan penanggungjawab atas berjalannya mesin conveyor di *line* mereka. *GL* ini juga yang melaporkan kerusakan menggunakan *datasheet* laporan kerusakan yang tersedia serta memanggil teknisi *backup* untuk melakukan perbaikan mesin conveyor apabila terjadi kerusakan. Pada mesin conveyor ini, proses *Final Assy* dilakukan. Jumlah *conveyor* yang ada di PT. JAI sebanyak 29 *conveyor*.

Jenis conveyor yang digunakan yaitu Seret / *Chain / Tow*, dimana conveyor ini menggunakan perangkat mekanis yang bergerak, seperti

rantai atau kabel, untuk menarik barang. Jenis *conveyor* ini digunakan untuk memindahkan material curah di tempat sampah, penerbangan, atau yang lain dan memiliki beberapa titik pelepasan atau pemuatan. Parameter lain dari conveyor adalah sebagai berikut:

- Karakteristik barang yang diangkut : bahan besi dan papan kayu
- Geometri conveyor : panjang total 38 meter dan kemiringan kereta yang diseret 60°.
- Kapasitas conveyor : 160 buah kabel / *shift*.
- Kecepatan conveyor : 48 km/menit
- Tipe conveyor : *round*.
- Usia conveyor : lebih dari 10 tahun.
- Penggunaan conveyor : 16 jam per hari.

Conveyor dibentuk dari 4 komponen utama, yaitu:

a. *Electric Motor*

Electric motor disini memiliki spesifikasi sebagai berikut:

- *Type: DC (Direct Current) / AC (Alternating Current)*
- *HP (Horse Power)* [hp]
- *Output range* [kW] - 0.75 KW
- *R.P.M (Revolutions Per Minute)* [r/min] - 1660 r/min
- *Rating voltage* [Volts] - 380 V
- *Frequency* [Hertz] - 50/60 Hz

b. *Gear Box*

Gear box memiliki spesifikasi sebagai berikut:

- Model: EWM 125V600R – LU075 S
- *Type: (worm or spur or helical gears)*
- *Gear Ration: 1:600*

Gear Box disambungkan dengan *electric motor* seperti pada gambar berikut:



Gambar 2.2 Penampakan *Gear Box* dan *Electric Motor*

c. *Roller Chain Sprocket with Transmission Chain*
Roller Chain Sprocket with Transmission Chain memiliki spesifikasi sebagai berikut:

- *Sprocket type* : D = 60mm
Material Stainless Steel
- *Chain Type* : *Pitch size* : 24.4 mm
/ Material: Stainless Steel



Gambar 2.3 Penampakan *Roller Chain Sprocket with Transmission Chain*

- d. *Drive Sprocket + Driven with Conveyor Chain*
Drive Sprocket + Driven with Conveyor Chain memiliki spesifikasi sebagai berikut:
- *Sprocket type* : D = 520 mm / Material: *Stainless Steel*
 - *Chain Type* : *Pitch size* : 101.6 mm / Material: *Stainless Steel*



Gambar 2.4 Penampakan *Conveyor Chain*

Suatu conveyor dapat berjalan dengan baik apabila memenuhi *variable* yang disebut dengan *4M Transition* sebagai berikut:

- a. **MAN**
Man, berarti karyawan yang bekerja di suatu mesin conveyor harus sesuai dengan *mapping plan*. Misal, *Final Assy* harus ada 60 orang dan *Pre-Assy* 40 orang, minimal 97% dari setiap *assy* harus hadir.
- b. **MATERIAL**
Material berarti semua jenis *part number material* dan kualitas harus minimal sama dengan jumlah ordernya.
- c. **METHODE**

Method berarti kelancaran si operator dalam bekerja yaitu cara atau mekanisme yang dipakai operator pada job stationnya (Langkah-langkah standart dalam bekerja) harus ada dan dijalankan.

d. MACHINE

Mesin otomatis sangat menunjang jalannya suatu sistem produksi manufaktur. Maka tidak boleh ada *equipment*/alat maupun mesin yang *trouble*/rusak yang bisa menghentikan jalannya conveyor baik di *pre-assy* maupun *final assy*.

Penggunaan conveyor di PT. Jatim Autocomp Indonesia tidak lepas dari kerusakan. Kerusakan yang paling sering terjadi yaitu *Drive Sprocket* yang sering putus sehingga conveyor tidak dapat berjalan. Sedangkan kerusakan yang terjadi di *Electrical* seperti *Electrical Motor* jarang sekali terjadi kerusakan. Penyebab *conveyor* mengalami berhenti total yaitu *supply circuit* / kabel terhambat supplynya menuju *final assy*.

2.2.2 Kerusakan Conveyor

Kerusakan *conveyor* yang sering terjadi terdiri dari beberapa subbagian dan memiliki kerusakan yang berbeda-beda. Kerusakan *conveyor* yang terjadi di PT. Jatim Autocomp Indonesia dapat dilihat pada tabel 2.1 berikut.

Tabel 2.1 Kerusakan Mesin Conveyor

No	Jenis Kerusakan	Detail Kerusakan	Gejala Kerusakan
1	<i>Control Box Panel</i>	<i>Control Box Panel</i> rusak	<i>Control Box Panel</i> terjadi kerusakan
2	Lampu Conveyor	Lampu Conveyor tidak menyala maupun berputar	Adanya kerusakan pada lampu conveyor
3	Sensor	Sensor depan belakang rusak	Sensor tidak dapat mendeteksi depan

			belakang <i>conveyor</i>
4	Motor	Motor terlalu panas atau berbunyi keras	Motor <i>Conveyor</i> menghasilkan panas yang tidak wajar atau berbunyi tidak wajar
5	Kabel <i>Electrical</i>	Kabel <i>electrical</i> <i>conveyor</i> tidak tertata rapi	Kabel <i>electrical</i> <i>conveyor</i> berantakan karena penggunaan
6	<i>Job Station</i>	Tombol job atau Tali Job station conveyor rusak atau putus	Tombol job atau Tali Job station conveyor rusak atau putus
7	<i>Display Conveyor</i>	<i>Display Conveyor</i> tidak berfungsi	Adanya kerusakan pada <i>display</i> <i>conveyor</i> yang menyebabkan tidak berfungsinya <i>display conveyor</i>
8	<i>Limit Switch</i>	<i>Limit Switch</i> rusak	Ada kerusakan pada <i>limit switch</i>
9	Lampu Andong	Display Lampu Andong tidak menyala atau rusak, lampu andong rusak, kabel <i>electrical</i> andong tidak tertata rapi	Lampu andong terjadi kerusakan sehingga tidak dapat menyala maupun rusak

10	Patek Kereta	Patek kereta tidak berfungsi atau patah	Patek kereta patah atau tidak berfungsi
11	<i>T-Joint</i> Pengait Kereta	<i>T-Joint</i> Pengait kereta tidak berfungsi atau patah	<i>T-Joint</i> pengait kereta patah karena penggunaan setiap hari
12	Mur Pengait	Mur pengait kereta lepas atau tidak berfungsi	Kurangnya pelumasan pada mur pengait atau terjadi kerusakan pada mur pengait kereta
13	Gear Rantai Conveyor	<i>Gear</i> dan Rantai <i>Conveyor</i> tidak berfungsi	Adanya kerusakan pada <i>Gear</i> dan Rantai <i>Conveyor</i> karena penggunaan setiap hari
14	<i>Seiling Lock Jig</i> Penarik	<i>Seiling Lock Jig</i> Penarik tidak berfungsi atau rusak	Ada kerusakan pada <i>Seiling</i> Penarik <i>Lock Jig</i>
15	Kebersihan <i>Conveyor</i>	<i>Conveyor</i> kotor, berdebu, atau oli <i>conveyor</i> berceceran	<i>Conveyor</i> kotor, berdebu, atau oli <i>conveyor</i> berceceran karena penggunaan setiap hari
16	Roda <i>Conveyor</i>	Roda <i>Conveyor</i> tidak berfungsi atau rusak	Adanya kerusakan pada roda <i>conveyor</i> .

2.2.4 *Python*

Python adalah bahasa pemrograman tingkat tinggi yang ditafsirkan, berorientasi objek, dengan semantik dinamis. Pemrograman tingkat tinggi yang dibangun dalam struktur data, dikombinasikan dengan pengetikan dinamis dan pengikatan dinamis, membuatnya sangat menarik untuk pengembangan aplikasi secara cepat, serta digunakan sebagai bahasa scripting untuk menghubungkan komponen yang ada bersama-sama [8]. Sintaksis *Python* yang sederhana dan mudah dipelajari menekankan keterbacaan dan karenanya mengurangi biaya pemeliharaan program.

Python mendukung modul dan paket, yang mendorong modularitas program dan penggunaan kembali kode. Interpreter *Python* dan pustaka standar yang luas tersedia dalam bentuk sumber atau biner tanpa biaya untuk semua platform utama, dan dapat didistribusikan secara bebas. *Python* juga dapat dikolaborasikan dengan beberapa bahasa pemrograman seperti *Java*, *C++*, *Javascript*.

2.2.5 *Keras*

Keras adalah API dengan jaringan saraf tingkat tinggi, membantu jalannya *Deep Learning* dan kecerdasan buatan. *Keras* ditulis dengan bahasa pemrograman *Python* dan mampu dijalankan pada *TensorFlow*, *CNTK*, atau *Theano*. *Keras* merupakan *neural network library* yang mudah digunakan. [9]

Fitur yang menonjol dari *Keras* yaitu:

1. *Keras* merupakan antarmuka tingkat tinggi yang menggunakan *TensorFlow* dan *Theano* sebagai *backend*-nya.
2. *Keras* dapat berjalan lancar di kedua CPU dan GPU.
3. *Keras* mendukung hampir semua model jaringan saraf - sepenuhnya terhubung, konvolusional, *pooling*, *recurrent*, *embedding*, dan lain lain. Selanjutnya, model ini dapat dikombinasikan untuk membangun model yang lebih kompleks.
4. *Keras* adalah kerangka kerja berbasis *Python*, yang membuatnya mudah untuk dideteksi dan dijelajahi atau dipelajari.

2.2.6 *Pandas*

Pandas adalah sebuah *library* di *Python* yang berlisensi *BSD* dan *open source* yang menyediakan struktur data dan analisis data yang mudah digunakan. *Pandas* biasa digunakan untuk membuat tabel, mengubah dimensi data, mengecek data, dan lain sebagainya. Struktur data dasar pada *Pandas* dinamakan *DataFrame*, yang memudahkan kita untuk membaca sebuah file dengan banyak jenis format seperti file *.txt*, *.csv*, dan *.tsv*. Fitur ini akan menjadikannya *table* dan juga dapat mengolah suatu data dengan menggunakan operasi seperti *join*, *distinct*, *group by*, *agregasi*, dan teknik lainnya yang terdapat pada *SQL* [10].

2.2.7 *Skicit-Learn*

Skicit-Learn merupakan *library Machine Learning open source* berbasis *Python* yang bisa digunakan dalam *Data Science*. Kelebihan *Skicit-Learn* adalah penggunaan *API* yang mudah serta kecepatannya saat melakukan tolok ukur yang berbeda dalam *dataset*. *Sklearn* kompatibel dengan *NumPy* dan *SciPy*. [11]

Skicit-Learn memberikan sejumlah fitur untuk keperluan *Data Science* seperti algoritma Regresi, pengelompokan, algoritma *Naive Bayes*, algoritma *Decision Tree*, parameter tuning, data *preprocessing tool*, *export/import model*, *Machine learning pipeline* dan algoritma klasifikasi termasuk *gradien*, *K-means*, mesin dukungan vektor, *DBSCAN*, dan juga mampu beroperasi dengan *SciPy* dan *NumPy*.

2.2.8 *CountVectorizer*

CountVectorizer adalah salah satu algoritma dari *library Skicit-Learn* yang dapat digunakan dalam *Machine Learning*. Algoritma ini dapat mengubah fitur *text* menjadi sebuah representasi vektor. Algoritma ini juga memungkinkan untuk melakukan *pre-processing* sebelum *dataset* menuju *Machine Learning* maupun *Deep Learning* untuk dilakukan *training*. [12]

2.2.9 *MinMaxScaler*

MinMaxScaler digunakan untuk mengurangi nilai minimum dalam fitur dan kemudian membaginya dengan rentang. Rentang disini adalah perbedaan antara maksimum asli dan minimum asli. *MinMaxScaler* mempertahankan bentuk distribusi aslinya dan tidak

secara berarti mengubah informasi yang disematkan dalam data asli [13]. Perhatikan bahwa *MinMaxScaler* tidak mengurangi pentingnya *outlier*. Rentang default untuk nilai yang dihasilkan oleh *MinMaxScaler* adalah 0 hingga 1.

2.2.10 Label Binarizer

Beberapa algoritma regresi dan klasifikasi biner tersedia di *scikit-learn*. Cara sederhana untuk memperluas algoritma ini ke kasus klasifikasi multi-kelas adalah dengan menggunakan apa yang disebut skema satu lawan semua. Saat waktu *learning* hanya terdiri dari mempelajari satu regresi atau pengklasifikasi biner per kelas. Dalam melakukannya, seseorang perlu mengonversi *label* multi-kelas ke label biner (milik atau bukan milik kelas). *Label Binarizer* mempermudah proses ini dengan metode transformasi. Pada waktu prediksi, seseorang menetapkan kelas yang model yang sesuai memberikan kepercayaan terbesar. *LabelBinarizer* membuatnya mudah dengan metode *inverse_transform* [14].

2.2.11 Deep Learning

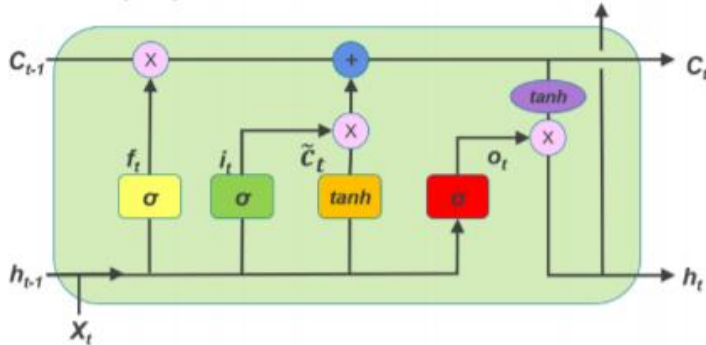
Deep Learning merupakan cabang *Machine Learning* yang didasarkan pada serangkaian algoritma yang memodelkan abstraksi tingkat tinggi dalam data menggunakan beberapa lapisan pemrosesan dengan struktur yang kompleks atau terdiri dari banyak transformasi data non-linear. *Deep Learning* adalah bagian dari *family* yang lebih luas dari metode *Machine Learning* berdasarkan dari representasi data pembelajaran. Observasi seperti gambar dapat direpresentasikan dalam banyak cara, seperti nilai vector intensitas per pixel, atau dengan cara lebih kompleks seperti kumpulan titik tepi, daerah dengan bentuk tertentu, dan lain-lain. Salah satu potensi dari *Deep Learning* yaitu mengganti fitur tangan manusia dengan algoritma yang efisien untuk *unsupervised* atau *semi-supervised* dan ekstraksi fitur hirarkis [15].

Varian *Deep Learning* terdiri dari *Deep Neural Network*, *Convolutional Deep Neural Network*, *Deep Belief Network* dan *Recurrent Neural Network*. Varian tersebut telah diaplikasikan pada berbagai bidang seperti komputer visi, pengenalan suara otomatis, pemrosesan bahasa

alami, pengenalan suara, dan bioinformatika. Atau bisa disebut *Deep Learning* ditandai dengan kata kunci atau rebranding jaringan syaraf. *Deep Learning* dapat dicirikan sebagai kelas algoritma pemrograman mesin yang menggunakan kaskade banyak lapisan unit pemrosesan non-linier untuk ekstraksi fitur dan transformasi [16].

2.2.12 Long Short-Term Memory

Long Short-Term Memory (LSTM) merupakan implementasi dari *Recurrent Neural Network (RNN)*. *LSTM* bertujuan untuk memecahkan masalah *RNN* yaitu *gradient vanishing* dan *exploding*. *Gradient Vanishing* adalah suatu penurunan gradien mendapatkan nilai yang lebih kecil setiap model *Deep Learning* dijalankan hingga mencapai 0. *Exploding Gradient* adalah suatu kenaikan nilai gradien yang tidak terkendali seiring berjalannya model *Deep Learning* [17]. *LSTM* menggantikan vektor tersembunyi *RNN* dengan memori yang dilengkapi dengan gerbang. Mekanisme gerbang *LSTM* mengimplementasikan tiga lapisan: gerbang (1) masukan, (2) *forget* dan (3) keluaran [18].



Gambar 2.5 Arsitektur *LSTM*

Setiap unit *LSTM* (Gambar 2.1) memiliki sel memori, dan status pada waktu t direpresentasikan sebagai C_t . Membaca dan memodifikasi dikendalikan oleh gerbang sigmoid dan berpengaruh pada gerbang masukan i_t , gerbang *forget* f_t dan gerbang keluaran o_t . *LSTM* dihitung sebagai berikut: Pada saat momen t th, model menerima masukan dari dua

sumber eksternal (h_{t-1} dan x_t). Status tersembunyi h_t dihitung oleh vektor masukan x_t yang diterima jaringan pada waktu t dan status tersembunyi sebelumnya h_{t-1} . Pada saat menghitung status simpul lapisan tersembunyi, gerbang masukan, keluaran, *forget* dan x_t akan secara bersamaan mempengaruhi keadaan *node*. Selain itu, setiap gerbang memiliki sumber internal, yaitu, status sel c_{t-1} dari blok selnya. Tautan antara *cell* dan gerbang sendiri dirujuk ke koneksi *peephole*.

Langkah-langkah LSTM dan gerbangnya adalah sebagai berikut:

1. **Gerbang Masukan**

Gerbang ini memutuskan nilai mana yang akan diperbarui dengan nilai transformasi antara 0 dan 1. Gerbang ini memiliki dua bagian. Pertama, lapisan sigmoid yang disebut lapisan gerbang masukan memutuskan nilai mana yang harus diperbarui (Persamaan 1). Kedua, lapisan tanh membuat vektor dari kandidat baru C_t yang dapat ditambah kedalam status (Persamaan 2).

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (1)$$

$$C_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \quad (2)$$

2. **Gerbang Forget**

Pada gerbang ini memutuskan informasi mana yang harus dibuang dan mana yang harus disimpan. Keputusan dibuat oleh lapisan sigmoid yang disebut lapisan *forget* gerbang (Persamaan 3) dimana keluaran angka antara 0 dan 1.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (3)$$

3. **Status Memori**

Pada gerbang ini memungkinkan untuk menjatuhkan nilai di cell state jika dikalikan dengan nilai mendekati 0 (Persamaan 4).

$$C_t = f_t * C_{t-1} + i_t * \tilde{c}_t \quad (4)$$

4. **Gerbang Keluaran**

Pada gerbang ini membuat keputusan apa yang harus dilakukan hidden state selanjutnya. Mengingat bahwa status tersembunyi terdiri dari informasi pada masukan sebelumnya. Pertama, menjalankan lapisan sigmoid yang memutuskan bagian mana dari cell state yang akan menjadi

keluaran (Persamaan 5). Kemudian, menempatkan cell state melalui tanh (Persamaan 6).

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (5)$$

$$h_t = o_t * \tanh(C_t) \quad (6)$$

2.2.13 *KFold Cross Validation*

KFold Cross Validation adalah suatu metode tambahan dari teknik *data mining* yang bertujuan untuk memperoleh hasil akurasi yang maksimal. Metode ini berjalan dimana percobaan sebanyak k kali untuk satu model dengan parameter yang sama. Fungsidari penggunaan metode *kfold cross validation* adalah

1. Untuk mengetahui performa dari suatu model algoritma dengan melakukan percobaan sebanyak k kali
2. Untuk meningkatkan tingkat performa dari model tersebut
3. Untuk mengolah *dataset* dengan kelas yang seimbang

Dalam kasus klasifikasi, ada yang perlu diperhatikan dalam pembagian *dataset* ke sejumlah k partisi, yaitu harus melakukan *stratification* yang artinya kita akan mempartisi atau membagi *dataset* tersebut ke k partisi dengan komposisi kelas yang seimbang disetiap partisinya. Dengan kata lain, distribusi kelas setiap partisi harus sama antar kelas, yang berarti juga sama dengan distribusi kelas di set data originalnya [19].

2.2.14 *Confusion Matrix*

Confusion matrix adalah suatu metode yang biasanya digunakan untuk melakukan perhitungan akurasi pada konsep *data mining* atau Sistem Pendukung Keputusan. Pada pengukuran kinerja menggunakan *confusion matrix*, terdapat 4 (empat) istilah sebagai representasi hasil proses klasifikasi.

		True Values	
		True	False
Prediction	True	TP Correct result	FP Unexpected result
	False	FN Missing result	TN Correct absence of result

Gambar 2.6 *Confusion Matrix*

Keempat istilah tersebut adalah *True Positive (TP)*, *True Negative (TN)*, *False Positive (FP)* dan *False Negative (FN)*. Nilai *True Negative (TN)* merupakan jumlah data negatif yang terdeteksi dengan benar, sedangkan *False Positive (FP)* merupakan data negatif namun terdeteksi sebagai data positif. Sementara itu, *True Positive (TP)* merupakan data positif yang terdeteksi benar. *False Negative (FN)* merupakan kebalikan dari *True Positive*, sehingga data positif, namun terdeteksi sebagai data negative [20].

Cara membaca *confusion matrix* adalah dengan menghitung nilai *accuracy*, *precision*, *recall* dan *F-1 Score* [21].

1. *Accuracy*

Accuracy menggambarkan seberapa akurat model dalam mengklasifikasikan dengan benar. *Accuracy* dapat dihitung dengan rumus berikut

$$Accuracy = \frac{True\ Positive}{Total\ Data\ Testing} \quad (7)$$

2. *Precision*

Precision menggambarkan akurasi antara data yang diminta dengan hasil prediksi yang diberikan oleh model. *Precision* dapat dihitung dengan rumus berikut

$$Precision = \frac{True\ Positive}{(True\ Positive + False\ Positive)} \quad (8)$$

3. *Recall*

Recall menggambarkan keberhasilan model dalam menemukan kembali sebuah informasi. *Recall* dapat dihitung dengan rumus berikut

$$Recall = \frac{True\ Positive}{(True\ Positive + False\ Negative)} \quad (9)$$

4. *F-1 Score*

F-1 Score menggambarkan perbandingan rata-rata precision dan recall yang dibobotkan. *Accuracy* tepat digunakan sebagai acuan performansi algoritma jika *dataset* memiliki jumlah data *False Negatif* dan *False Positif* yang sangat mendekati (*symmetric*). Namun jika jumlahnya tidak mendekati, maka sebaiknya menggunakan F1 Score sebagai acuan. *F-1 Score* dapat dihitung dengan rumus berikut

$$F - 1 \text{ Score} = \frac{(2 \times \text{Recall} \times \text{Precision})}{(\text{Recall} + \text{Precision})} \quad (10)$$

2.2.15 Receiver Operating Characteristics

Receiver Operating Characteristic (ROC) digunakan untuk memverifikasi hasil akurasi model *Deep Learning*. Kurva ROC biasanya menampilkan tingkat positif sejati pada sumbu Y, dan tingkat positif palsu pada sumbu X. Ini berarti bahwa sudut kiri atas plot adalah titik "ideal" - tingkat positif palsu nol, dan tingkat positif benar satu. Ini tidak terlalu realistis, tetapi itu berarti bahwa area yang lebih besar di bawah kurva (AUC) biasanya lebih baik. "Kecuraman" kurva ROC juga penting, karena ideal untuk memaksimalkan tingkat positif sejati sambil meminimalkan tingkat positif palsu.

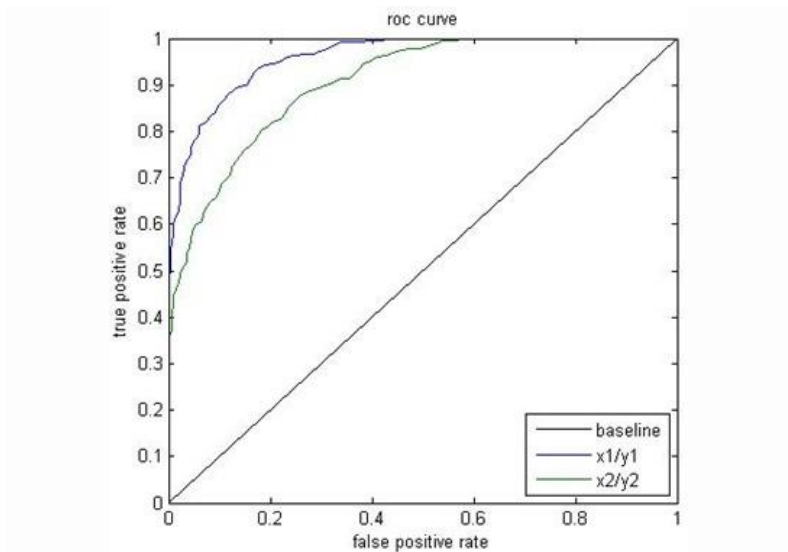
Kurva ROC biasanya digunakan dalam klasifikasi biner untuk mempelajari output dari classifier. Untuk memperluas kurva ROC dan area ROC ke klasifikasi *multi-label*, output perlu dibinerisasi. Satu kurva ROC dapat digambar per label, tetapi kurva ROC juga dapat digambar dengan mempertimbangkan setiap elemen dari matriks indikator label sebagai prediksi biner (*micro-averaging*) [22].

Kurva ROC dibuat berdasarkan nilai telah didapatkan pada perhitungan dengan *confusion matrix*, yaitu antara *False Positive Rate* dengan *True Positive Rate*. Dimana:

$$\text{False Positive Rate (FPR)} = \frac{\text{False Positive}}{(\text{False Positive} + \text{True Negative})} \quad (11)$$

$$\text{True Positive Rate (TPR)} = \frac{\text{True Positive}}{((\text{True Positive} + \text{False Negative}))} \quad (12)$$

Berikut adalah contoh kurva ROC.



Gambar 2.7 Kurva *ROC*

Untuk membaca kurva ini sangat mudah, kinerja algoritma klasifikasi adalah:

1. JELEK, apabila kurva yang dihasilkan mendekati garis *baseline* atau garis yang melintang dari titik koordinat 0,0.
2. BAGUS, jika kurva mendekati titik koordinat 0,1.

Pada contoh di atas dapat dilihat 2 kurva, yaitu kurva dengan warna biru dan kurva dengan warna hijau. Berdasarkan cara membaca di atas, maka dapat disimpulkan kinerja kurva berwarna biru lebih bagus dibandingkan kinerja kurva berwarna hijau [23].

-----Halaman ini sengaja dikosongkan-----

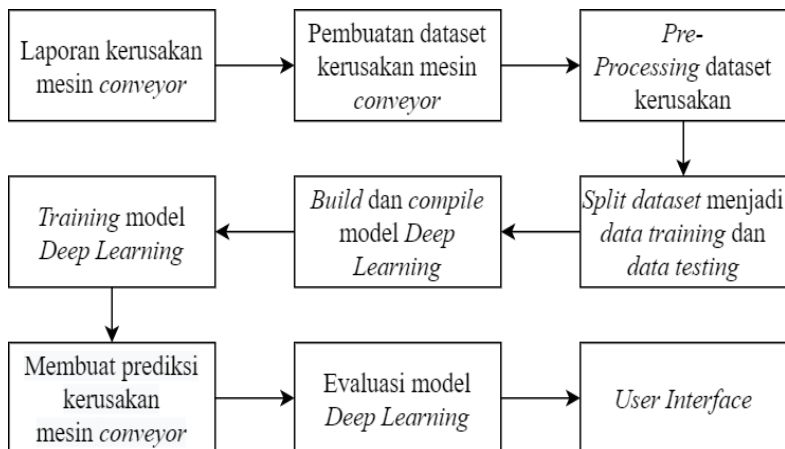
BAB III

PERANCANGAN DAN IMPLEMENTASI SISTEM

Pada bab 3 ini akan dijelaskan sistematika perancangan dan implementasi sistem. Berikut merupakan perancangan sistem beserta implementasinya untuk “Sistem *Monitoring* Perawatan dan Prediksi Kerusakan pada Mesin *Conveyor* menggunakan *Deep Learning*”.

3.1 Perancangan Sistem

Dalam perancangan sistem yang digunakan dalam proyek akhir ini terdapat beberapa tahap yaitu pembagian dataset menjadi data training dan data testing, pre-processing yang terdiri dari *CountVectorizer* , *MinMaxScaler* dan *Label Binarizer*, membagi dataset menjadi *data training* dan *data testing*, membuat dan melakukan *training* menggunakan model *Deep Learning*, membuat prediksi kerusakan yang akan terjadi di masa depan dan melakukan evaluasi akurasi model *Deep Learning* dan yang dapat dilihat pada gambar 3.1.



Gambar 3.1 Diagram Rancangan Sistem

Gambar 3.1 adalah blok diagram sistem yang akan dibangun. Berdasarkan diagram perancangan sistem diatas proses yang dilakukan dapat dijabarkan sebagai berikut:

4. *Input data text*

Memasukkan data *text* ke dalam sistem. *Text* yang digunakan yaitu file data kerusakan dan data perawatan mesin conveyor dalam format *csv*. Input data *text* tersebut berisi jenis kerusakan dan *level* kerusakan.

5. *Pre-Processing*

Proses ini mengubah isi data *text* yang telah di-*input*-kan menjadi kalimat yang dapat dibaca oleh sistem dengan metode *CountVectorizer* dan *MinMaxScaler*.

6. *Split Dataset* menjadi *Data Training* dan *Data Testing*

Proses ini membagi dataset yang telah melalui proses *Pre-Processing* menjadi *Data Training* dan *Data Testing* menggunakan library *Skicit-learn*.

7. *Build dan compile model Deep Learning*.

Proses ini membuat model *Deep Learning* menggunakan library *Keras* dan melakukan *compile* model *Deep Learning* tersebut menggunakan bahasa *python*.

8. *Training model Deep Learning*.

Proses ini melakukan *training* dengan model *Deep Learning* yang telah di-*compile* sebelumnya dengan menggunakan *data training* sebagai data *input* yang akan di-*training*-kan.

9. *Membuat Prediksi Kerusakan*

Proses ini membuat prediksi sesuai yang kita butuhkan berdasarkan hasil dari model *Deep Learning* yang digunakan.

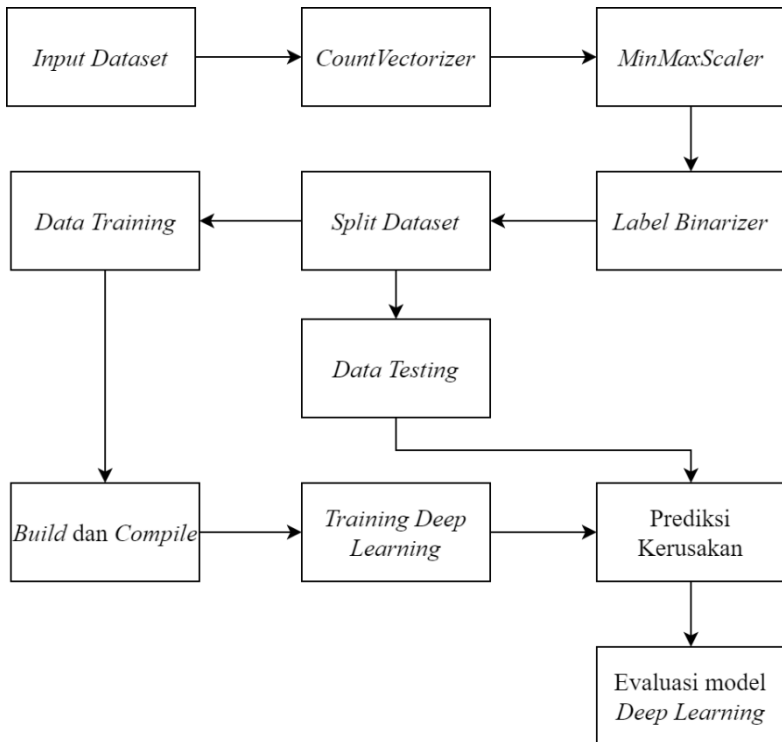
10. *Evaluasi Akurasi model Deep Learning*

Proses ini mengevaluasi akurasi dari model *Deep Learning* yang digunakan. Semakin tinggi akurasi dari model *Deep Learning* yang digunakan, maka prediksi dapat diketahui lebih akurat. Selain akurasi, evaluasi dilakukan dengan metode *KFold*, *Confusion Matrix* dan *ROC* untuk memperkuat hasil evaluasi.

11. *User Interface*

Proses ini membuat dan menampilkan hasil prediksi kerusakan dalam sebuah aplikasi antarmuka yang bertujuan untuk mempermudah *user* dalam melihat hasil prediksi yang telah dilakukan.

Untuk proses *Deep Learning* yang dilakukan, dapat dilihat pada gambar 3.2 berikut.



Gambar 3.2 Diagram Alur Proses *Deep Learning*

Gambar 3.2 adalah blok diagram alur proses *Deep Learning* yang akan dibangun. Berdasarkan diagram alur proses *Deep Learning* diatas proses yang dilakukan dapat dijabarkan sebagai berikut.

1. *Input Dataset*

Input Dataset merupakan tahapan memasukkan *dataset* ke dalam sistem sebelum masuk ke dalam *Deep Learning*.

2. *CountVectorizer*

CountVectorizer adalah proses *Pre-Processing* untuk mengubah teks menjadi nilai vektor yang dapat dikenali oleh sistem.

3. *MinMaxScaler*

MinMaxScaler adalah proses *Pre-Processing* kedua untuk mengubah nilai yang belum normal menjadi nilai dengan rentang 0 hingga 1.

4. *Label Binarizer*

Label Binarizer adalah proses *Pre-Processing* untuk memberikan label pada kolom *dataset* dan digunakan pada proses evaluasi model *deep learning*.

5. *Split Dataset*

Split Dataset adalah proses memisahkan *dataset* menjadi *data training* dan *data testing*.

5.1 *Data Training*

Data Training adalah *dataset* yang di-input-kan ke dalam sistem untuk melakukan *training*.

5.2 *Data Testing*

Data Testing adalah *dataset* yang digunakan untuk menguji hasil *training* yang telah dilakukan. *Data testing* menurut diagram langsung menuju blok Prediksi Kerusakan karena *Data Testing* akan menjalankan perannya.

6. *Build dan compile*

Build dan compile adalah proses membuat dan membentuk model *Deep Learning* sesuai dengan kebutuhan.

7. *Training Deep Learning*

Training model *Deep Learning* adalah proses *Training* sistem dengan menggunakan *Data Training*.

8. Prediksi Kerusakan

Prediksi Kerusakan adalah proses memprediksi kerusakan setelah sistem melakukan *Training data*.

9. Evaluasi model *Deep Learning*

Evaluasi model *Deep Learning* adalah proses evaluasi terhadap model *Deep Learning* yang digunakan.

3.2 Pengambilan data di PT. Jatim Autocomp Indonesia

Data kerusakan dan data perawatan diperoleh dari salah satu perusahaan yang memproduksi *wiring harness* yaitu PT. Jatim Autocomp Indonesia yang berlokasi di Gempol, Pasuruan. Data yang digunakan berasal dari tim *backup* dan *maintenance* dari salah satu divisi yaitu *New Yazaki System* yang bertugas melakukan perbaikan pada *equipment* mesin *conveyor* yang ada di PT. Jatim Autocomp Indonesia. Pengambilan data dilakukan menggunakan *input* teks yang dituliskan di *datasheet maintenance* yang dilakukan. Data tersebut berupa data *hard copy* dan dimasukkan ke sistem melalui *Microsoft Excel* secara *manual*. Data yang digunakan dimulai dari Januari 2016 hingga Juli 2020. Jumlah data kerusakan yang digunakan yaitu 1875 data. Data yang diambil dari PT. JAI adalah jenis kerusakan dan *level* kerusakan dan status data yang digunakan yaitu data simulasi. Variabel dari jenis kerusakan yang diketahui ada 16 variabel jenis kerusakan yang dapat dilihat pada tabel 3.1 berikut.

Tabel 3.1 Jenis Kerusakan

No	Jenis Kerusakan	Label Kerusakan
1	Control Box Panel	CBP
2	Lampu Conveyor	LC
3	Sensor	SNSR
4	Motor	MTR
5	Kabel Electrical	KE
6	Job Station	JS
7	Display Conveyor	DC

8	Limit Switch	LS
9	Lampu Andong	LA
10	Patek Kereta	PK
11	T-Joint Pengait Kereta	TJPK
12	Gear Rantai Conveyor	GRC
13	Mur Pengait	MP
14	Seiling Penarik Lock Jig	SPLJ
15	Kebersihan Conveyor	KC
16	Roda Conveyor	RC

Untuk level kerusakan yang diketahui ada 5, yaitu Normal, Ringan, Menengah, Berat dan Berbahaya. *Level* kerusakan ditentukan dengan banyaknya jenis kerusakan yang terjadi dalam waktu satu hari. *Level* kerusakan yang didapatkan mempengaruhi rekomendasi tindakan yang harus dilakukan untuk menangani kerusakan sesuai dengan *level*-nya. Berikut tabel jumlah jenis kerusakan, *level* kerusakan dan rekomendasi tindakan yang digunakan.

Tabel 3.2 Jumlah Jenis Kerusakan, *Level* Kerusakan dan Rekomendasi Tindakan

No	Jumlah jenis kerusakan	<i>Level</i> kerusakan	Rekomendasi Tindakan
1	0	Normal	Selalu waspada
2	1-3	Ringan	Mulai waspada, lapor ke teknisi
3	4-7	Menengah	Mulai cek conveyor, mulai ekstra hati-hati
4	8-11	Berat	Cek conveyor, ekstra hati-hati
5	12-16	Berbahaya	Hentikan penggunaan conveyor, Cek conveyor keseluruhan

Tabel 3.2 menampilkan jumlah jenis kerusakan yang muncul dan mempengaruhi *level* kerusakan dan rekomendasi tindakan yang harus diambil. Setiap rekomendasi tindakan yang ditampilkan berbeda-beda sesuai dengan *level* kerusakan yang ditampilkan.

3.3 *Input data text*

Tahap *input data text* merupakan tahap memasukkan data kerusakan dan data perawatan yang telah di-sortir dan disimpan dalam bentuk file *csv*. Proses *input data* dilakukan dengan menuliskan file *csv* yang telah disiapkan sebelumnya. Berikut kode program *input data text*:

```
import pandas as pd

data = pd.read_csv('dataset.csv')

X = data.iloc[:,0:16].values

y = data.iloc[:, 16].values
```

Source 3.1 Input Data Text

Source 3.1 merupakan fungsi yang digunakan untuk memanggil nama file *.csv* yang merupakan *dataset* yang digunakan. *Library* yang digunakan yaitu *pandas* yang dapat digunakan untuk *import* file *.csv* yang terdiri dari kalimat/kata-kata dan angka. Kemudian data tersebut dibagi menjadi variabel *X* yang merupakan jenis kerusakan dan *y* yang merupakan *level* kerusakan.

3.4 *Pre-Processing Data Text*

Tahap *Pre-Processing data text* merupakan tahap mengubah data *text* yang telah di-*input* menjadi dataset yang dapat dibaca oleh sistem komputer. Metode *Pre-Processing data text* yang digunakan yaitu Mengubah huruf menjadi angka array menggunakan *CountVectorizer*, *MinMaxScaler* dan *Label Binarizer*.

3.4.1 *CountVectorizer*

Tahap *CountVectorizer* yaitu mengubah huruf yang ada pada dataset menjadi representasi vektor. Misal, “Ringan” menjadi menjadi representasi vektor berupa {0,0,1}. *CountVectorizer* diperlukan untuk mengubah teks menjadi angka vektor dikarenakan *Deep Learning* tidak mengenali *input data* berupa teks. Berikut kode program mengubah huruf menjadi angka vektor.

```
from sklearn.feature_extraction.text import CountVectorizer  
  
cv = CountVectorizer()  
  
y_vector = cv.fit_transform(y).toarray()
```

Source 3.2 CountVectorizer

Source 3.2 diawali dengan mengambil isi variabel kemudian diubah dari huruf menjadi representasi vektor kemudian disimpan dalam variabel *y_vector*. Untuk data variabel *X* tidak perlu dilakukan *CountVectorizer* karena data yang diberikan sudah berupa angka sehingga tidak perlu melalui tahapan ini.

3.4.2 MinMaxScaler

Tahap *MinMaxScaler* yaitu mengubah hasil representasi vektor yang sudah diberikan oleh *CountVectorizer* menjadi angka *decimal* antara 0 hingga 1 yang dapat digunakan untuk memprediksi kerusakan mesin *conveyor* di tahap akhir. Berikut kode program *MinMaxScaler* yang digunakan.

```
from sklearn.preprocessing import MinMaxScaler  
  
min_max_scaler = MinMaxScaler(feature_range=(0, 1))  
  
y_normalized = min_max_scaler.fit_transform(y_vector)
```

Source 3.3 MinMaxScaler

Source 3.3 diawali dengan *import* algoritma *MinMaxScaler* dari library *Skicit-learn*, kemudian untuk data klasifikasi kerusakan yang telah melalui proses *CountVectorizer*, masuk ke dalam program *MinMaxScaler* kemudian hasil dari *MinMaxScaler* tersebut masuk ke dalam variabel baru yang bernama *y_normalized* yang merupakan hasil *Pre-Processing* yaitu *MinMaxScaler*.

3.4.2 *Label Binarizer*

Tahap *Label Binarizer* digunakan untuk memberikan label pada setiap kolom dari dataset yang telah melalui proses *MinMaxScaler*. Selain memberikan label, proses ini diperlukan untuk tahapan *plot* grafik *ROC*. Berikut kode program *Label Binarizer* yang digunakan.

```
from sklearn.preprocessing import label_binarize  
  
y_binarized = label_binarize(y_normalized, classes=[0, 1, 2, 3, 4])  
  
n_classes = y_normalized.shape[1]
```

Source 3.4 Label Binarizer

Source 3.4 diawali dengan *import* algoritma *label_binarize* dari library *sklearn*. Kemudian membuat variabel baru bernama *y_binarized* yang berisi hasil dari proses *label binarizer*. Selanjutnya yaitu variabel *n_classes* yang berisi hasil proses *label binarizer* yang diberikan *shape*. Variabel ini diperlukan untuk tahapan *plot* grafik *ROC*.

3.5 *Split Dataset menjadi Data Training dan Data Testing*

Tahap *Split Dataset* menjadi *Data Training* dan *Data Testing* yaitu proses membagi *dataset* yang telah melalui proses *pre-processing* menjadi *Data Training* dan *Data Testing* sebelum melalui proses *Deep Learning*.

```
from sklearn.model_selection import train_test_split  
  
X_train, X_test, y_train, y_test = train_test_split(X, y_normalized,  
                                                    test_size = 0.2, random_state = 1)
```

Source 3.5 Split Dataset menjadi Data Training dan Data Testing

Source 3.5 diawali dengan memasukkan isi variabel yang akan dibagi menjadi *Data Training* dan *Data Testing*, yaitu variabel *X* dan *y_processed*. *X* disini adalah data kerusakan mesin *conveyor*. *y_binarized* disini adalah data klasifikasi kerusakan berdasarkan data kerusakan yang diberikan. Kemudian dua data tersebut dibagi menggunakan algoritma

train_test_split dari *library Skicit-learn* yang hasilnya disimpan dalam variabel *X_train*, *X_test*, *y_train*, *y_test*. Variabel *X* diisi oleh variabel kerusakan dan variabel *y* diisi oleh variabel *level* kerusakan mesin *conveyor*.

3.6 ***Build dan compile model Deep Learning***

Tahap *build* dan *compile* model *Deep Learning* yaitu melakukan pembuatan dan *compile* model *Deep Learning* yang digunakan. Model *Deep Learning* yang digunakan yaitu *Long Short-Term Memory*. Berikut kode program model *Deep Learning* yang digunakan.

```
from keras.models import Sequential

from keras.layers import Dense

from keras.layers import LSTM

from keras.layers.embeddings import Embedding

from keras.preprocessing import sequence

import numpy as np

input_dim = 16

output_dim = 16

X_train = sequence.pad_sequences(X_train,
maxlen=output_dim)

X_test = sequence.pad_sequences(X_test,
maxlen=output_dim)

embedding_vector_length = 16

model = Sequential()

model.add(Embedding(input_dim, output_dim,
input_length=output_dim))

model.add(LSTM(350))

model.add(Dense(5, activation='sigmoid'))
```

```
model.compile(loss='binary_crossentropy',  
optimizer='adam', metrics=['accuracy'])
```

Source 3.6 Build dan Compile model Deep Learning

Source 3.6 diawali dengan *top_words* dan *max_review_length* yang digunakan untuk membatasi jangkauan program *Deep Learning* terhadap dataset yang diberikan. Kemudian membuat model *Deep Learning* secara *sequential* dan menambahkan LSTM dalam model tersebut dan diaktivasi secara *sigmoid*. Model *loss* yang digunakan pada *model.compile* diatas menggunakan *binary_crossentropy* kemudian *optimizer* yang digunakan menggunakan metode *adam* dan *metrics* atau satuan yang digunakan yaitu *accuracy*.

3.7 Training model Deep Learning

Training model *Deep Learning* yaitu menggabungkan dataset dengan model *Deep Learning* yang telah dibuat untuk dilakukan *training* dataset. Berikut kode program training model *Deep Learning* yang digunakan.

```
history = model.fit(X_train, y_train, epochs=20,  
batch_size=128, verbose=0, shuffle=False,  
validation_data=(X_test, y_test))
```

Source 3.7 Training model Deep Learning

Source 3.7 terdiri dari data *X_train* dan *Y_train* yang telah dibagi pada proses *split dataset* sebelumnya. Kemudian ada *model.fit* yang merupakan penggabungan model *Deep Learning* yang telah dibuat dengan dataset sebelumnya. Kemudian terdapat *epoch*, dimana digunakan untuk melakukan *looping training*, yang dilakukan sebanyak 20 kali. *Epoch* dilakukan sebanyak 20 kali karena jumlah *dataset* yang diberikan sebanyak 1875 *data* sehingga *epoch* yang diberikan cukup 20 kali untuk meringankan beban sistem. Selain itu proses training ini juga melibatkan data *testing* yang terdiri dari *X_test* dan *y_test* untuk memvalidasi hasil *training* yang telah dilakukan. Evaluasi akurasi tersebut dapat digambarkan dalam bentuk grafik dengan Source 3.8 berikut.

```

history_dict = history.history
history_dict.keys()

acc = history_dict['accuracy']
loss = history_dict['loss']
val_acc = history_dict['val_accuracy']
val_loss = history_dict['val_loss']

epochs = range(len(loss))

import matplotlib.pyplot as plt
plt.figure()

plt.plot(epochs, loss, 'r', label='Data Training Loss')
plt.plot(epochs, acc, 'g', label='Data Training Accuracy')
plt.plot(epochs, val_loss, 'b', label='Data Testing Loss')
plt.plot(epochs, val_acc, 'y', label='Data Testing Accuracy')

plt.title("Training dan Accuracy Loss")

plt.legend(loc="right")

plt.show()

```

Source 3.8 Plot Grafik Training Deep Learning

Source 3.8 diawali dengan pengambilan proses *training* yang disimpan dalam variabel *history* yang kemudian mengambil hasil *training* yang telah dilakukan, diantaranya *accuracy*, *loss*, *val_accuracy* dan *val_loss*. *Accuracy* adalah hasil akurasi pada *data training*, *loss* adalah hasil *loss* pada *data training*, *val_accuracy* adalah hasil akurasi pada *data testing* dan *val_loss* adalah hasil *loss* pada *data testing*.

3.8 Membuat Prediksi Kerusakan Mesin Conveyor

Membuat prediksi kerusakan yaitu membuat prediksi kerusakan mesin conveyor berdasarkan model *Deep Learning* yang telah melalui proses *training data*. Variabel yang digunakan untuk memprediksi kerusakan yaitu *X_test* dan hasil prediksi tersebut divalidasi menggunakan variabel *y_test*. Berikut kode program untuk membuat prediksi kerusakan mesin conveyor.

```
data_original = pd.DataFrame(y_test)

data_predictions = pd.DataFrame(model.predict(X_test))

####Normalisasi hasil prediksi####

data_original =
min_max_scaler.fit_transform(data_original)

data_predictions =
min_max_scaler.fit_transform(data_predictions)

####Ubah data jadi nilai bulat####

x_olah = np.round(data_original)

y_olah = np.round(data_predictions)

vector = np.vectorize(np.int)

x_olah = vector(x_olah)

y_olah = vector(y_olah)

original_fix = pd.DataFrame(x_olah)

predicted_fix = pd.DataFrame(y_olah)
```

Source 3.9 Membuat prediksi kerusakan

Source 3.9 menggunakan variabel *y_test* dan *X_test* yang telah melalui proses prediksi kerusakan. Disini *library Pandas* berperan yang dapat dilihat dengan adanya kode program *DataFrame* yang merupakan

struktur data dasar dari *Pandas*, dan menampung hasil prediksi dalam variabel *data_predictions*. Selanjutnya hasil dari prediksi kerusakan tersebut di-normalisasi menggunakan *MinMaxScaler* pada data hasil prediksi kerusakan yang nanti digunakan untuk tahapan selanjutnya dan diubah menjadi angka bulat antara 0 dan 1 menggunakan *Vectorize* dari library *numpy*. *Vectorize* digunakan untuk mempermudah dalam membaca dan membuktikan hasil prediksi dengan nilai dari variabel *y_test*.

3.9 Evaluasi model *Deep Learning*

Evaluasi model *Deep Learning* yaitu mengevaluasi model *Deep Learning* yang digunakan. Evaluasi model *Deep Learning* terdiri dari 3 bagian, yaitu Akurasi, *Confusion Matrix* dan *ROC*.

3.9.1 Evaluasi Akurasi model *Deep Learning*

Model *Deep Learning* dapat dievaluasi menggunakan teknik akurasi model *Deep Learning*. Berikut kode program evaluasi akurasi yang digunakan.

```
#####Menghitung nilai akurasi#####  
  
scores = model.evaluate(X_test, y_test, verbose=0)  
  
print("Accuracy from LSTM: %.2f%%\n" %  
(scores[1]*100))
```

Source 3.10 Evaluasi akurasi model *Deep Learning*

Source 3.10 terdiri dari *model.evaluate* yang digunakan untuk mengevaluasi keakurasian model *Deep Learning* ketika menggunakan dataset yang telah diberikan. Kemudian program dapat menuliskan akurasi model *Deep Learning* dengan format persentase. Nilai persentase yang digunakan yaitu nilai *val_accuracy* dimana variabel tersebut merupakan nilai akurasi dengan metode validasi data menggunakan *data testing*.

3.9.2 Evaluasi model *Deep Learning* dengan metode *KFold*

Evaluasi selanjutnya menggunakan metode *KFold*, suatu metode tambahan dari teknik *data mining* yang bertujuan untuk memperoleh hasil

akurasi yang maksimal. Metode ini berjalan dimana percobaan sebanyak k kali untuk satu model dengan parameter yang sama. Berikut kode program evaluasi dengan metode *KFold*.

```
seed = 7

numpy.random.seed(seed)

from sklearn.model_selection import KFold

kfold      =      KFold(n_splits=10,      shuffle=True,
random_state=seed)

cvscores = []

for train, test in kfold.split(X, y_normalized):

    model = Sequential()

    model.add(Embedding(input_dim,      output_dim,
input_length=output_dim))

    model.add(LSTM(350))

    model.add(Dense(5, activation='sigmoid'))

    # Compile model

    model.compile(loss='binary_crossentropy',
optimizer='adam', metrics=['accuracy'])

    # Fit the model

    model.fit(X[train], y_normalized[train], epochs=20,
batch_size=128, verbose=0,

            shuffle=False, validation_data=(X_test, y_test))

    # evaluate the model

    scores = model.evaluate(X[test], y_normalized[test],
verbose=0)
```

```

        print("%s:  %.2f%%" % (model.metrics_names[1],
scores[1]*100))

        cvscores.append(scores[1] * 100)

        print("%.2f%% (+/- %.2f%%)" % (numpy.mean(cvscores),
numpy.std(cvscores)))

```

Source 3.11 Evaluasi dengan metode *KFold*

Source 3.11 diawali dengan inisialisasi angka acak menggunakan library *numpy* kemudian inisialisasi *KFold* yang digunakan. Sistem pembagian yang digunakan yaitu 1 : 10 dimana 1 adalah *data testing* dan 10 adalah *data training* dan dijalankan sebanyak 10 kali. Selanjutnya yaitu memasukkan program *Deep Learning* yang digunakan ke dalam fungsi *looping* menggunakan variabel *X* dan *y_normalized* yang belum melalui tahapan *split dataset*. Setelah itu program dijalankan secara *looping* sebanyak 10 kali dan menampilkan hasil rata-rata akurasi dengan *Deep Learning* setelah *looping* sebanyak 10 kali.

3.9.3 Evaluasi *Confusion Matrix* dari model *Deep Learning*

Evaluasi selanjutnya dapat dilakukan menggunakan metode *Confusion Matrix*. Berikut kode program *confusion matrix* yang digunakan.

```

from sklearn.metrics import confusion_matrix

import matplotlib.pyplot as plt

data_asli = original_fix.values.tolist()

data_prediksi = predicted_fix.values.tolist()

data_asli=np.argmax(data_asli, axis=1)

data_prediksi=np.argmax(data_prediksi, axis=1)

cm = confusion_matrix(data_asli, data_prediksi)

cm = numpy.flip(cm)

```



```

from sklearn.metrics import accuracy_score,
precision_score, recall_score

Accuracy_confusion = accuracy_score(data_asli,
data_prediksi)

Precision_confusion = precision_score(data_asli,
data_prediksi, average='macro')

Recall_confusion = recall_score(data_asli,data_prediksi,
average='macro')

print("Accuracy of Confusion: %.2f%%" %
(Accuracy_confusion*100))

print("Precision of Confusion: %.2f%%" %
(Precision_confusion*100))

print("Recall of Confusion: %.2f%%\n" %
(Recall_confusion*100))

```

Source 3.12 Confusion Matrix

Source 3.12 diawali dengan memasukkan hasil prediksi yang sudah dilakukan ke dalam *data_asli* dan *data_prediksi* untuk diubah *type data*-nya menjadi *list*, kemudian diolah oleh *library numpy* dan algoritma *argmax* dan disimpan ke dalam variabel yang sama. Selanjutnya dari hasil olah data tersebut masuk ke dalam variabel *cm* yang merupakan variabel untuk memasukkan program *confusion matrix*. Nilai *Accuracy*, *Precision* dan *Recall* juga dapat ditemukan dengan memanggil fungsi *accuracy_score*, *precision_score* dan *recall_score* yang menggunakan variabel *data_asli* dan *data_prediksi*. Selanjutnya hasil dari *confusion matrix* dapat diketahui dalam bentuk grafik tabel dengan *Source 3.13* berikut.

```
f, ax = plt.subplots(figsize=(8,5))
sns.heatmap(cm, annot=True, fmt="1.0f", cmap='Blues')
plt.xlabel("predicted")
plt.ylabel("truth")
plt.show()
```

Source 3.13 Grafik Confusion Matrix

Source 3.13 diawali dengan inisialisasi *plot* grafik menggunakan *plt*, kemudian integrasi hasil *confusion matrix* ke dalam *sns.heatmap* yang merupakan latar dari *plot* grafik yang diberikan. Kemudian diberi *label* untuk *X* adalah *truth* atau hasil yang sebenarnya dan *y* adalah hasil prediksi dengan model *Deep Learning*.

3.9.4 Evaluasi ROC dari model Deep Learning

Bentuk evaluasi lain terhadap model *Deep Learning* yang telah dibuat yaitu menggunakan algoritma *ROS*. *ROS* juga digunakan sebagai metode evaluasi model *Deep Learning* terhadap hasil akurasi dari model *Deep Learning* yang telah dibuat. Berikut kode program dari evaluasi menggunakan algoritma *ROS*.

```
from sklearn.metrics import roc_auc_score

roc_score = roc_auc_score(data_original,
data_predictions, multi_class='ovo', average="macro")

print("ROC Score from LSTM = %.2f%%" %
(roc_score*100))
```

Source 3.14 ROC

Source 3.14 diawali dengan import *roc_auc_score* dari *library sklearn.metrics*. Selanjutnya program menghitung nilai *ROC* tersebut menggunakan variabel *data_original* dan *data_predictions* dimana dua variabel tersebut berisi nilai hasil prediksi dari *training* yang telah dilakukan sebelum dinormalisasikan menggunakan *vectorize*. Yang perlu

diketahui adalah nilai dari dua variabel tersebut harus bertipe data *float*, bukan *integer*.

```
import matplotlib.pyplot as plt

plt.figure()

lw = 2

plt.plot(fpr[2], tpr[2], color='darkorange',
```

```
         lw=lw, label='Predicted (area = %0.2f)' %
roc_auc[2])

plt.plot([0, 1], [0, 1], color='navy', lw=lw, label='Predicted',
linestyle='--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('Prediction')

plt.ylabel('True Data')

plt.title('ROC')

plt.show()
```

Source 3.15 Grafik Evaluasi ROC

Source 3.15 merupakan proses *plot* grafik evaluasi dengan ROC, dimana hasil *roc_auc* digunakan untuk melakukan *plot* grafik evaluasi dengan ROC, selanjutnya menampilkan garis *baseline* dimulai dari koordinat (0,0) sampai di koordinat (1,1). Label X diberikan nama *Prediction* dan label y diberikan nama *true data*.

3.10 User Interface

User Interface adalah antarmuka program yang dapat digunakan untuk menampilkan hasil prediksi kerusakan pada mesin *conveyor* yang lebih mudah dipahami oleh *user* awam.

3.10.1 Perubahan Hasil Prediksi menjadi Tulisan Prediksi dan Rekomendasi Tindakan

Sebelum menampilkan hasil prediksi di *user interface*, perlu ada perubahan data hasil prediksi tersebut yang dari angka menjadi tulisan prediksi dan rekomendasi tindakan agar lebih mudah dipahami oleh *user*. Berikut kode program perubahan hasil prediksi menjadi tulisan serta rekomendasi tindakan yang telah dibuat.

```
pre_prediksi = np.array(y_olah).astype('str')

pre_prediksi =
pd.DataFrame(pre_prediksi, columns=['Berat', 'Berbahaya', 'Menengah', 'Normal', 'Ringan'])

pre_prediksi["level"] = pre_prediksi["Berat"] +
pre_prediksi["Berbahaya"] + pre_prediksi["Menengah"] +
pre_prediksi["Normal"] + pre_prediksi["Ringan"]

pre_prediksi['level'] = pre_prediksi['level'].replace(
['10000', '01000', '00100', '00010', '00001', '00101', '11000', '10100'], ['
BERAT', 'BERBAHAYA', 'MENENGAH', 'NORMAL', 'RINGAN', 'MEN
ENGAH', 'BERBAHAYA', 'BERAT'])

pre_prediksi["perawatan"] = pre_prediksi["Berat"] +
pre_prediksi["Berbahaya"] + pre_prediksi["Menengah"] +
pre_prediksi["Normal"] + pre_prediksi["Ringan"]

pre_prediksi['perawatan'] =
pre_prediksi['perawatan'].replace(['10000', '01000', '00100', '00010'
, '00001', '00101', '11000', '10100'],
```

```

        ['Cek conveyor, ekstra hati-hati',
         'Hentikan penggunaan conveyor, Cek conveyor
keseluruhan',
         'Mulai cek conveyor, mulai ekstra hati-hati',
         'Selalu waspada',
         'Mulai waspada, lapor ke teknisi',
         'Mulai cek conveyor, mulai ekstra hati-hati',
         'Hentikan penggunaan conveyor, Cek conveyor
keseluruhan',
         'Cek conveyor, ekstra hati-hati'])
data_prediksi = pre_prediksi["level"].tolist()
data_rekomendasi = pre_prediksi["perawatan"].tolist()

```

Source 3.16 Perubahan Hasil Prediksi menjadi Tulisan Prediksi dan Rekomendasi Tindakan

Source 3.16 diawali dengan pengambilan data dari variabel *y_olah* yang diubah menjadi list data dengan tipe data *PandaFrame* untuk mempermudah dalam proses perubahan ini dan memberikan label dalam variabel *PandaFrame* kemudian disimpan dalam variabel *pre-prediksi*. Selanjutnya membuat kolom baru yang menampilkan hasil penggabungan *level* kerusakan yang bernama “*level*” dan “*perawatan*” dan hasil penggabungan tersebut diubah menjadi *level* kerusakan dan rekomendasi tindakan yang harus diambil.

3.10.2 Pembuatan User Interface

Pembuatan *User Interface* dilakukan setelah melalui proses perubahan hasil prediksi yang telah dilakukan. *User interface* yang dibangun masih berbasis bahasa *python* dan menggunakan *library* bernama *tkinter*. Berikut kode program *user interface* yang telah dibuat.

```

from tkinter import Label, Button, Tk

####inisiasi User Interface####

root = Tk()

####Ukuran dan judul UI####

root.geometry('640x480')

root.title("Program Prediksi Kerusakan Mesin Conveyor by
PENS")

```

Source 3.17 Inisialisasi Jendela User Interface

Source 3.17 merupakan kode program inisialisasi jendela *user interface* yang diatur dengan ukuran 640x480 dan diberi nama program tersebut yaitu *"Program Prediksi Kerusakan Mesin Conveyor by PENS"*.

```

####Menampilkan hari dan jam secara realtime####

from datetime import datetime

def tick():

    now          =          datetime.now().strftime('%d-%m-%y
%H:%M:%S')

    clock.config(text=now)

    clock.after(200, tick)

clock = Label(root, font='ariel 10', bg="white", fg="black")

clock.grid(row=0, column=0)

clock.place(x=530,y=0)

tick()

```

Source 3.18 Inisialisasi Waktu dan Tanggal

Source 3.18 merupakan kode program untuk inisialisasi fungsi menampilkan waktu dan tanggal secara *real time* kemudian diubah menjadi sebuah teks, kemudian diintegrasikan ke dalam jendela *user interface* dan diletakkan pada koordinat 530,0.

```
#####Hasil Prediksi#####

day = datetime.now().strftime('%d')

a = int(day)

kemarin = a - 1

hari_ini = a

besok = a + 1

Kemarin = Label(root, text="Kemarin ada kerusakan %s di
salah satu conveyor" % data_prediksi[kemarin], fg="black",
font='ariel 12' )

Kemarin.grid(row=10, column=15)

Kemarin.place(x=100, y=200)

HasilKerusakan = Label(root, text="Hari ini diprediksi ada
kerusakan %s di salah satu conveyor" % data_prediksi[hari_ini],
fg="black", font='ariel 12' )

HasilKerusakan.grid(row=12, column=15)

HasilKerusakan.place(x=100, y=250)

RekomendasiAksi = Label(root, text="Rekomendasi: %s" %
data_rekomendasi[hari_ini], fg="black", font='ariel 12' )

RekomendasiAksi.grid(row=14, column=15)

RekomendasiAksi.place(x=100, y=300)
```

```
MasaDepan = Label(root, text="Besok diprediksi ada
kerusakan %s di salah satu conveyer" % data_prediksi[besok],
fg="black", font='ariel 12' )
```

```
MasaDepan.grid(row=16, column=15)
```

```
MasaDepan.place(x=100, y=350)
```

Source 3.19 Menampilkan Hasil Prediksi dalam User Interface

Source 3.19 merupakan proses menampilkan hasil prediksi yang sudah diubah menjadi sebuah teks prediksi sesuai hasil dari *Source 3.16* sebelumnya dan ditampilkan dalam *user interface*. Ada 4 kalimat yang ditampilkan, yaitu kerusakan yang terjadi kemarin, hari ini dan keesokan hari. Untuk rekomendasi tindakan disesuaikan dengan kerusakan yang diprediksi terjadi hari ini. Hasil prediksi tersebut ditampilkan dalam jumlah 365 hari.

```
#####Fungsi Close Button#####
```

```
def clicked():
```

```
    root.destroy()
```

```
    btn = Button(root, text = "Close", fg = "red",
command=clicked)
```

```
    btn.grid(column=2, row=0)
```

```
    btn.place(x=540,y=430)
```

```
#####Jalankan program#####
```

```
root.mainloop()
```

Source 3.20 Close Button

Source 3.20 merupakan kode program untuk menampilkan *close button* yang digunakan untuk menutup *user interface* yang telah dibuat.

BAB IV PENGUJIAN DAN ANALISA

Pada bab pengujian dan analisa ini akan dibahas mengenai tahapan pengujian dan analisa yang dilakukan berdasarkan hasil yang didapatkan. Sehingga dari proses ini dapat diketahui tingkat keberhasilan sistem.

4.1 Pengujian *Input data text*

Pada tahap ini digunakan untuk memasukkan data kerusakan dan data perawatan ke dalam sistem.

Index	CBP	LC	SNSR	MTR	KE	JS	DC	LS	LA	PK	TJPK	MP	GRC	SPLJ	KC	RC	level
0	1	0	1	1	1	1	0	1	1	0	1	0	1	1	1	1	Berbahaya
1	1	1	0	1	1	1	1	0	0	1	0	1	1	1	1	1	Berbahaya
2	0	1	0	0	1	1	0	0	1	0	1	1	1	1	0	1	Berat
3	0	1	1	0	0	1	0	1	1	1	1	0	0	1	1	1	Berat
4	1	0	0	1	0	0	0	1	1	0	1	1	1	0	1	0	Berat
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Normal
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Normal
7	1	0	1	0	0	1	1	1	1	1	1	0	0	1	0	0	Berat
8	0	1	0	0	0	1	1	0	0	0	1	0	1	0	1	1	Menengah
9	1	1	0	1	0	0	1	0	1	1	0	1	0	0	1	0	Berat
10	0	1	0	0	1	1	0	1	1	0	1	0	1	0	1	1	Berat
11	0	1	0	0	1	1	1	0	0	1	1	0	0	1	1	0	Berat

Gambar 4.1 Hasil pengujian *input* data kerusakan dan *level* kerusakan.

Pada gambar 4.1, hasil input data kerusakan dan data perawatan tersebut terdapat dua kolom, yaitu jenis kerusakan dan klasifikasi kerusakan. Setiap kerusakan ditandai dengan angka 1 dan setiap tiada kerusakan diberi angka 0.

4.2 Pengujian *Pre-Processing Data Text*

Tahap *Pre-Processing data text* merupakan tahap mengubah data *text* yang telah di-*input* menjadi dataset yang dapat dibaca oleh sistem komputer. Metode *Pre-Processing data text* yang digunakan yaitu *CountVectorizer*, *MinMaxScaler* dan *Label Binarizer*.

4.2.1 Pengujian *CountVectorizer*

Tahap *CountVectorizer* yaitu mengubah huruf yang ada pada dataset menjadi angka array. Berikut hasil pengujian huruf menjadi angka array pada gambar 4.2.

	0	1	2	3	4
0	0	1	0	0	0
1	0	1	0	0	0
2	1	0	0	0	0
3	1	0	0	0	0
4	1	0	0	0	0
5	0	0	0	1	0
6	0	0	0	1	0
7	1	0	0	0	0
8	0	0	1	0	0
9	1	0	0	0	0
10	1	0	0	0	0
11	1	0	0	0	0

Gambar 4.2 Hasil pengujian *CountVectorizer*

Pada gambar 4.2, diketahui bahwa kalimat yang mengandung kata-kata Normal, Ringan, Menengah, Berat dan berbahaya memiliki nilai vektor yang berbeda. *Level* normal memiliki nilai vektor {0,0,0,1,0}, ringan memiliki nilai vektor {0,0,0,0,1}, menengah memiliki nilai vektor {0,0,1,0,0}, berat memiliki nilai vektor {1,0,0,0,0} dan berbahaya memiliki nilai vektor {0,1,0,0,0}. Nilai vektor tersebut tidak bisaurut seperti urutan *level* pada umumnya karena sistem mengurutkan nilai vektor tersebut berdasarkan abjad dari *level* kerusakan yang diberikan.

4.2.2 Pengujian *MinMaxScaler*

Tahap *MinMaxScaler* yaitu proses mengubah hasil dari *CountVectorizer* menjadi angka dalam skala 0 hingga 1 yang dapat digunakan untuk melakukan prediksi kerusakan mesin conveyor. *MinMaxScaler* masih terdalem proses *Pre-Processing* dimana proses ini

diperlukan sebelum melakukan tahapan selanjutnya. Berikut hasil pengujian *MinMaxScaler* yang telah dilakukan

y_normalized - NumPy object array

	0	1	2	3	4
0	0	1	0	0	0
1	0	1	0	0	0
2	1	0	0	0	0
3	1	0	0	0	0
4	1	0	0	0	0
5	0	0	0	1	0
6	0	0	0	1	0
7	1	0	0	0	0
8	0	0	1	0	0
9	1	0	0	0	0
10	1	0	0	0	0
11	1	0	0	0	0

Gambar 4.3 Hasil pengujian *MinMaxScaler*

Pada gambar 4.3, diketahui bahwa hasil proses *MinMaxScaler* tidak ada perbedaan dengan hasil pada gambar 4.2. Hal ini terjadi karena *input data* yang diberikan sudah berupa angka pasti yaitu 0 dan 1, akan tetapi *type data* yang dihasilkan berbeda antara *CountVectorizer* dengan *MinMaxScaler*, dimana *type data* dari *CountVectorizer* berupa *array of int64*, sedangkan *type data* dari *MinMaxScaler* berupa *array of float64*.

4.2.3 Pengujian *Label Binarizer*

Label Binarizer digunakan untuk memberi label pada setiap kolom sebelum *dataset* di-*split* untuk masuk ke model *Deep Learning*. *Label Binarizer* juga dibutuhkan untuk proses *plot* grafik *ROC* yang akan dilakukan selanjutnya. Berikut hasil dari proses *Label Binarizer*.

y_binarized - NumPy object array					
	0	1	2	3	4
0	0	1	0	0	0
1	0	1	0	0	0
2	1	0	0	0	0
3	1	0	0	0	0
4	1	0	0	0	0
5	0	0	0	1	0
6	0	0	0	1	0
7	1	0	0	0	0
8	0	0	1	0	0
9	1	0	0	0	0
10	1	0	0	0	0
11	1	0	0	0	0

Gambar 4.4 Hasil pengujian *Label Binarizer*

Pada gambar 4.4, dapat dilihat bahwa secara angka *dataset* tidak ada perubahan. Label tidak diberikan di *Label Binarizer* bertujuan untuk mempermudah dalam proses selanjutnya. Hasil *Label Binarizer* sendiri digunakan untuk evaluasi *ROC* model *Deep Learning* pada subbab 4.7.4

4.3 Pengujian *Split Dataset* menjadi *Data Training* dan *Data Testing*

Tahap *split dataset* menjadi *data training* dan *data testing* digunakan untuk membagi dataset menjadi *data training* dan *data testing* sebelum melakukan tahapan selanjutnya yaitu *Deep Learning*. Jumlah data yaitu 1875 dibagi dengan rasio 8:2. Rasio 8 dimiliki oleh *data training* sedangkan rasio 2 dimiliki oleh *data testing*. Berikut hasil pengujian *split dataset* yang berhasil dilakukan.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	1	0	0	1	1	1	0	0	0	1	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	0	0	1	0	0	0	1	0	0	1	0	0	1	0	1
3	0	1	0	0	0	0	1	0	0	1	0	0	1	0	0	0
4	0	0	0	1	0	1	1	0	0	0	0	1	0	0	1	0
5	1	0	1	0	0	1	1	0	0	0	1	0	0	1	1	1
6	1	0	0	1	0	1	0	0	1	0	1	0	1	1	1	0
7	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0
8	1	0	1	0	1	1	0	0	1	0	0	1	0	1	1	1
9	1	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0
10	0	1	1	0	0	1	0	1	1	0	1	1	0	0	1	1
11	1	0	0	1	0	0	0	1	0	0	1	0	0	1	0	1

Gambar 4.5 Hasil pengujian *Data Training* Variabel *X*

	0	1	2	3	4
0	0	0	1	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	0	0
4	0	0	1	0	0
5	1	0	0	0	0
6	1	0	0	0	0
7	0	0	0	0	1
8	1	0	0	0	0
9	0	0	1	0	0
10	1	0	0	0	0
11	0	0	1	0	0

Gambar 4.6 Hasil pengujian *Data Training* Variabel *Y*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
3	0	1	0	0	0	0	0	1	0	0	1	0	0	0	1	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	1	1	1	1	0	1	1	0	1	1	1	1	1	1	1	1
8	0	1	1	0	0	1	0	1	1	0	1	1	0	0	1	1
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	1	1	1	0	1	1	1	1	1	1	1	1	0	1	1	1
11	1	1	1	1	1	1	1	1	1	1	1	0	0	1	0	1

Gambar 4.7 Hasil pengujian *Data Testing* Variabel X

	0	1	2	3	4
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	0	0	1
3	0	0	1	0	0
4	0	0	0	1	0
5	0	0	0	0	1
6	0	0	0	1	0
7	0	1	0	0	0
8	1	0	0	0	0
9	0	0	0	1	0
10	0	1	0	0	0
11	0	1	0	0	0

Gambar 4.8 Hasil pengujian *Data Testing* Variabel Y

Pada gambar 4.5, 4.6, 4.7 dan 4.8, dapat diketahui bahwa terdapat dua variabel yang dibagi menjadi *data training* dan *data testing*, yaitu data perawatan dan data kerusakan. Data perawatan disini yang digunakan yaitu data perawatan yang telah melalui proses *pre-processing* dan disimpan dalam variabel *X_train* dan *X_test*. Data kerusakan yang diambil tidak melalui proses *pre-processing* dan disimpan dalam variabel *y_train* dan *y_test*. Total *data training* yang didapatkan berdasarkan rasio tersebut berjumlah 1500 *data* dan *data testing* yang didapatkan berjumlah 375 *data*.

4.4 Pengujian Build dan Compile model Deep Learning

Tahap *build* dan *compile* model *Deep Learning* digunakan untuk membentuk dan *compile* model *Deep Learning*. Berikut hasil pengujian *build* dan *compile* model *Deep Learning*.

Model: "sequential_16"

Layer (type)	Output Shape	Param #
embedding_16 (Embedding)	(None, 16, 16)	256
lstm_16 (LSTM)	(None, 350)	513800
dense_16 (Dense)	(None, 5)	1755
Total params: 515,811		
Trainable params: 515,811		
Non-trainable params: 0		

Gambar 4.9 Hasil proses *Build* dan *Compile* model *Deep Learning*.

Gambar 4.9 menampilkan salah satu model *Deep Learning* yaitu *Long Short-Term Memory* yang digunakan sebagai model *Deep Learning* utama dalam memprediksi kerusakan yang akan terjadi di masa depan. *Long Short-Term Memory (LSTM)* merupakan implementasi dari *Recurrent Neural Network (RNN)* yang bertujuan untuk memecahkan masalah RNN yaitu *gradient vanishing* dan *gradient exploding* yang mengakibatkan rendahnya akurasi hasil prediksi yang didapatkan. *Shape* yang digunakan dalam *LSTM* disini terdapat 350 *shape* yang dapat meningkatkan tingkat akurasi prediksi kerusakan menggunakan *LSTM*.

4.5 Pengujian *Training model Deep Learning*

Tahap *Training model Deep Learning* digunakan untuk melakukan training model *Deep Learning* berdasarkan model *Deep Learning* yang telah di-build dan *compile* sebelumnya. Berikut hasil pengujian *training model Deep Learning*.

```
Train on 1500 samples, validate on 375 samples
Epoch 1/20
1500/1500 [=====] - 3s 2ms/step - loss: 0.5678 - accuracy:
0.7793 - val_loss: 0.4129 - val_accuracy: 0.8155
Epoch 2/20
1500/1500 [=====] - 2s 1ms/step - loss: 0.4371 - accuracy:
0.7963 - val_loss: 0.4130 - val_accuracy: 0.8155
Epoch 3/20
1500/1500 [=====] - 2s 1ms/step - loss: 0.4266 - accuracy:
0.7939 - val_loss: 0.4062 - val_accuracy: 0.7973
Epoch 4/20
1500/1500 [=====] - 2s 1ms/step - loss: 0.3903 - accuracy:
0.7972 - val_loss: 0.3206 - val_accuracy: 0.8565
Epoch 5/20
1500/1500 [=====] - 2s 1ms/step - loss: 0.2895 - accuracy:
0.8648 - val_loss: 0.2478 - val_accuracy: 0.8843

Epoch 6/20
1500/1500 [=====] - 2s 1ms/step - loss: 0.2877 - accuracy:
0.8665 - val_loss: 0.2762 - val_accuracy: 0.8544
Epoch 7/20
1500/1500 [=====] - 2s 1ms/step - loss: 0.2553 - accuracy:
0.8792 - val_loss: 0.2308 - val_accuracy: 0.8912
Epoch 8/20
1500/1500 [=====] - 2s 1ms/step - loss: 0.2174 - accuracy:
0.8967 - val_loss: 0.1930 - val_accuracy: 0.9163
Epoch 9/20
1500/1500 [=====] - 2s 1ms/step - loss: 0.1846 - accuracy:
0.9148 - val_loss: 0.1719 - val_accuracy: 0.9136
Epoch 10/20
1500/1500 [=====] - 2s 1ms/step - loss: 0.1564 - accuracy:
0.9271 - val_loss: 0.1554 - val_accuracy: 0.9285
Epoch 11/20
1500/1500 [=====] - 2s 1ms/step - loss: 0.1534 - accuracy:
0.9313 - val_loss: 0.1360 - val_accuracy: 0.9413
```



```

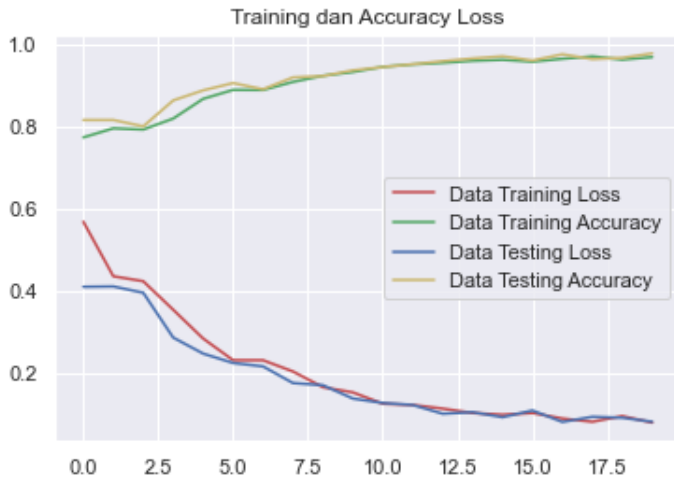
Epoch 12/20
1500/1500 [=====] - 2s 1ms/step - loss: 0.1325 - accuracy:
0.9455 - val_loss: 0.1280 - val_accuracy: 0.9461
Epoch 13/20
1500/1500 [=====] - 2s 1ms/step - loss: 0.1143 - accuracy:
0.9537 - val_loss: 0.1059 - val_accuracy: 0.9579
Epoch 14/20
1500/1500 [=====] - 2s 1ms/step - loss: 0.1392 - accuracy:
0.9443 - val_loss: 0.1342 - val_accuracy: 0.9344
Epoch 15/20
1500/1500 [=====] - 2s 1ms/step - loss: 0.1161 - accuracy:
0.9551 - val_loss: 0.1039 - val_accuracy: 0.9664
Epoch 16/20
1500/1500 [=====] - 2s 1ms/step - loss: 0.0964 - accuracy:
0.9619 - val_loss: 0.0984 - val_accuracy: 0.9611
Epoch 17/20
1500/1500 [=====] - 2s 1ms/step - loss: 0.0938 - accuracy:
0.9627 - val_loss: 0.0957 - val_accuracy: 0.9701

Epoch 18/20
1500/1500 [=====] - 2s 1ms/step - loss: 0.0968 - accuracy:
0.9605 - val_loss: 0.0904 - val_accuracy: 0.9669
Epoch 19/20
1500/1500 [=====] - 2s 1ms/step - loss: 0.0830 - accuracy:
0.9689 - val_loss: 0.0861 - val_accuracy: 0.9627
Epoch 20/20
1500/1500 [=====] - 2s 1ms/step - loss: 0.0903 - accuracy:
0.9659 - val_loss: 0.0889 - val_accuracy: 0.9659

```

Gambar 4.10 Hasil pengujian *Training* model *Deep Learning*

Gambar 4.10 menampilkan proses *training* model *Deep Learning*, yaitu *Long Short-Term Memory*. Bisa dilihat bahwa setiap *epoch* memiliki *loss* yang semakin sedikit tiap *epoch* tersebut bertambah, baik untuk data *training* maupun data *testing*. Untuk data *testing*, nilai akurasi dan *loss*-nya bernama *val_accuracy* dan *val_loss*. *Training* yang dilakukan menghasilkan beberapa variabel baru, diantaranya *accuracy*, *loss*, *val_accuracy* dan *val_loss*. *Accuracy* adalah hasil akurasi pada data *training*, *loss* adalah hasil *loss* pada data *training*, *val_accuracy* adalah hasil akurasi pada data *testing* dan *val_loss* adalah hasil *loss* pada data *testing*. Setiap nilai tersebut dapat dibuat grafiknya pada gambar 4.11 sebagai berikut.



Gambar 4.11 Grafik hasil pengujian model *Deep Learning*

Gambar 4.11 menampilkan grafik hasil pengujian model *Deep Learning* dimana ada 4 variabel yang digrafikkan, diantaranya *Data Training Accuracy* yang diambil dari variabel *accuracy*, *Data Training Loss* yang diambil dari variabel *loss*, *Data Testing Accuracy* yang diambil dari variabel *val_accuracy* dan *Data Testing Loss* yang diambil dari variabel *val_loss*. Dapat dilihat bahwa semakin banyak berjalannya *epoch*, maka nilai akurasi dan loss, baik dari *Data Training* maupun *Data Testing* memiliki nilai yang sama antara satu sama lain.

4.6 Pengujian Membuat Prediksi Kerusakan Mesin Conveyor

Membuat prediksi kerusakan mesin conveyor digunakan untuk membuat prediksi kerusakan berdasarkan hasil model *Deep Learning* yaitu *Long Short-Term Memory* dimana yang diprediksi adalah klasifikasi kerusakan mesin conveyor, apakah kerusakan tersebut tergolong berat, ringan, normal, menengah atau berbahaya. Berikut salah satu prediksi yang berhasil dibuat dengan model *Deep Learning*.

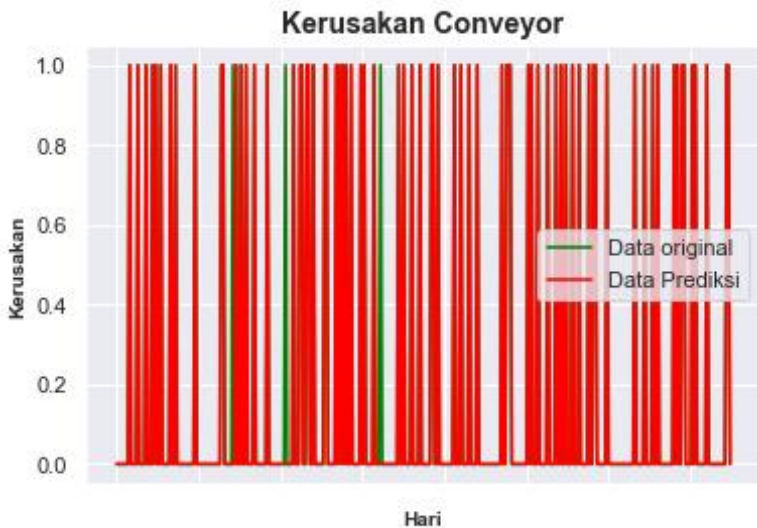
Index	0	1	2	3	4
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	0	0	1
3	0	0	1	0	0
4	0	0	0	1	0
5	0	0	0	0	1
6	0	0	0	1	0
7	0	1	0	0	0
8	1	0	0	0	0
9	0	0	0	1	0
10	0	1	0	0	0
11	0	1	0	0	0

Gambar 4.12 Hasil pengujian *Data testing level* kerusakan

Index	0	1	2	3	4
0	0	0	0	1	0
1	0	0	0	1	0
2	0	0	0	1	0
3	0	0	1	0	0
4	0	0	0	1	0
5	0	0	0	0	1
6	0	0	0	1	0
7	0	1	0	0	0
8	1	0	0	0	0
9	0	0	0	1	0
10	0	1	0	0	0
11	0	1	0	0	0

Gambar 4.13 Hasil pengujian prediksi kerusakan

Berdasarkan gambar 4.11 dan 4.12, dapat dilihat bahwa prediksi dianggap sesuai apabila nilai dari kolom, baik dari data *original* maupun data prediksi, memiliki posisi yang sama. Misal, pada data prediksi ke 1, nilai tertinggi berada pada kolom bernama 2, kemudian kita cocokkan dengan data asli. Pada kasus diatas, posisi nilai tertinggi antara data prediksi dengan data asli berada di kolom sama, maka validasi tersebut dianggap benar, dimana ada kesamaan antara data asli dengan data prediksi. Hasil prediksi tersebut dapat dilihat secara grafik seperti pada gambar berikut.



Gambar 4.14 Grafik hasil pengujian prediksi dengan model *Deep Learning*

Berdasarkan gambar 4.13 diatas, dapat dilihat bahwa hasil prediksi sesuai dengan *data testing* yang diberikan, walau ada beberapa prediksi yang tidak sesuai dengan *data testing* yang diberikan.

4.7 Pengujian Evaluasi model *Deep Learning*

Evaluasi model *Deep Learning* digunakan untuk mengevaluasi model *Deep Learning* yang digunakan, terutama untuk akurasi model *Deep Learning* yang digunakan, yaitu *Long Short-Term Memory*.

4.7.1 Pengujian Evaluasi akurasi model *Deep Learning*

Evaluasi pertama terhadap model *Deep Learning* yaitu keakuratan model *Deep Learning* dalam memprediksi kerusakan mesin *conveyor*. Hasil evaluasi akurasi model *Deep Learning* yang didapatkan yaitu **97,71%**. Hasil akurasi tersebut didapatkan dengan menggunakan *Deep Learning* itu sendiri bersama dengan fungsi *Sequential* yang melibatkan data *testing* baik variabel *X* maupun *y* yang telah melalui proses *split dataset*.

4.7.2 Pengujian Evaluasi akurasi model *Deep Learning* dengan metode *KFold*

Evaluasi kedua model *Deep Learning* yaitu menggunakan metode *KFold* yang bertujuan untuk memperoleh hasil akurasi yang maksimal. Metode ini berjalan dimana percobaan sebanyak *k* kali untuk satu model dengan parameter yang sama. Dalam pengujian ini menggunakan 10 kali untuk satu model dengan parameter yang sama, dimana model *Deep Learning* yang sudah ada, dijalankan sebanyak 10 kali melalui proses *looping* dengan pembagian *dataset* menjadi *data training* dan *data testing* yang berbeda dengan pengujian *split dataset* yang sudah dilakukan dimana rasio perbandingannya adalah 1:10. Berikut tabel hasil pengujian yang didapatkan pada Tabel 4.1.

Tabel 4.1 Pengujian Evaluasi Akurasi model *Deep Learning* dengan metode *KFold*

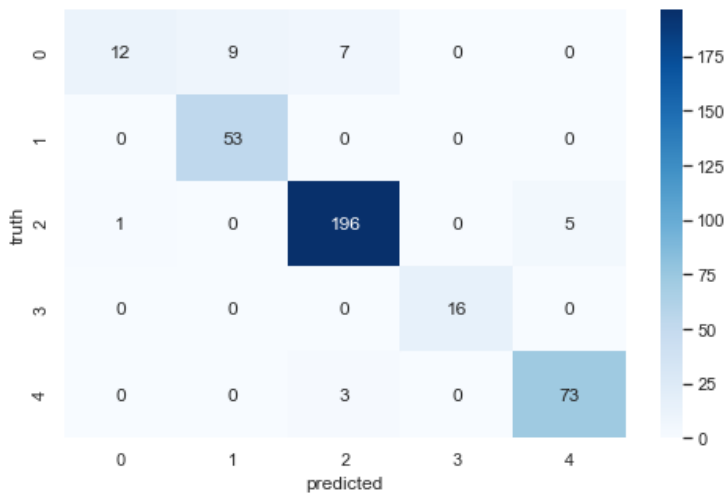
K	Akurasi
1	97,13%
2	96,70%
3	94,79%
4	95,96%
5	95,11%
6	96,90%
7	94,65%
8	97,33%
9	91,76%

10	96,26%
Rata-rata	95,66%

Berdasarkan tabel 4.1, setelah diuji dengan 10 kali *looping*, rata-rata nilai akurasi yang didapatkan yaitu sebesar **95,66%**. Apabila dibandingkan dengan nilai akurasi dari model *Deep Learning* yang telah dilakukan, terdapat perbedaan nilai akurasi $\pm 2\%$ dikarenakan perbedaan pembagian *dataset* antara pengujian evaluasi dengan model *Deep Learning* dan pengujian evaluasi dengan metode *KFold*.

4.7.3 Pengujian Evaluasi *Confusion Matrix* model *Deep Learning*

Evaluasi ketiga terhadap model *Deep Learning* yaitu menggunakan *Confusion Matrix* dari model *Deep Learning*. Berikut hasil *Confusion Matrix* yang didapatkan.



Gambar 4.15 Hasil pengujian *Confusion Matrix*

Berdasarkan gambar 4.15, kita dapat menghitung *confusion matrix* tersebut dengan *Accuracy*, *Precision* dan *Recall* [24].

1. *Accuracy*

Accuracy mennggambarkan seberapa akurat model dalam mengklasifikasikan dengan benar. Rumus dari *Accuracy* sebagai berikut:

$$Accuracy = \frac{True\ Predicted}{Total\ Dataset} \quad (13)$$

2. *Precision*

Precision menggambarkan Data yang diambil berdasarkan informasi yang kurang atau salah atau tidak tepat. Rumus dari *Accuracy* sebagai berikut:

$$Precision = \frac{True\ Predicted}{(True\ Predicted + False\ Predicted)} \quad (14)$$

3. *Recall*

Recall menggambarkan data yang tidak mampu diprediksi dengan benar. Rumus dari *Recall* sebagai berikut:

$$Recall = \frac{True\ Predicted}{(True\ Predicted + False\ Negative)} \quad (15)$$

Dari perhitungan yang sudah dilakukan, berikut nilai hasil perhitungan *Confusion Matrix*:

1. Nilai *Accuracy* : **93,33%**
2. Nilai *Precision* : **93,31%**
3. Nilai *Recall* : **87,19%**

Apabila dibandingkan antara nilai *Accuracy* dari model *Deep Learning* dan *Confusion Matrix* terdapat perbedaan yang cukup signifikan. Hal itu dikarenakan variabel yang digunakan untuk mengevaluasi nilai akurasi tersebut berbeda, dimana nilai *Accuracy* ditentukan dari hasil evaluasi yang menggunakan variabel *X_testing* dan *y_testing*, sedangkan variabel yang digunakan untuk evaluasi akurasi dengan *Confusion Matrix* ditentukan oleh variabel hasil prediksi kerusakan diantaranya variabel *data_asli* dan *data_prediksi* yang berasal dari hasil pembulatan prediksi kerusakan.

4.7.4 Pengujian Evaluasi *ROC* model *Deep Learning*

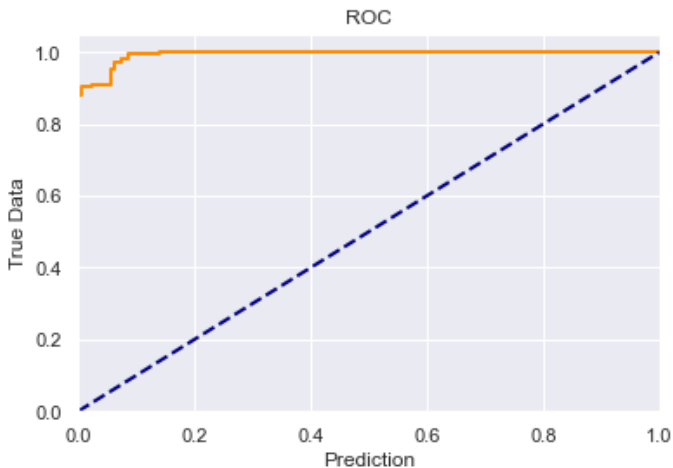
Evaluasi keempat terhadap model *Deep Learning* yaitu menggunakan *ROC* dari model *Deep Learning*. *ROC* digunakan untuk memverifikasi hasil akurasi model *Deep Learning*. Perhitungan nilai *ROC* didapatkan dengan mengetahui perbandingan *False Positive Rate* dengan *True Positive Rate* sebagai fungsi *threshold* dari sebuah model untuk

mengklasifikasikan kelas positif [25], yang dapat diketahui dengan rumus berikut

$$\text{False Positive Rate (FPR)} = \frac{\text{False Positive}}{(\text{False Positive} + \text{True Negative})} \quad (16)$$

$$\text{True Positive Rate (TPR)} = \frac{\text{True Positive}}{((\text{True Positive} + \text{False Negative}))} \quad (17)$$

Berdasarkan rumus yang diketahui, nilai tersebut dimasukkan ke dalam perhitungan di program yang dibuat dan nilai ROC yang didapatkan dari hasil pengujian ini adalah **99,19%**. Berikut hasil plot ROC yang didapatkan.



Gambar 4.16 Hasil pengujian ROC

Gambar 4.16 merupakan hasil pengujian dari ROC. Cara membaca kurva ROC ini relatif mudah, yaitu ada 2 kategori sebagai berikut:

1. Kurva ROC tergolong “Buruk” jika kurva yang dihasilkan mendekati garis *baseline* atau garis yang melintang dari titik 0,0 yang diwakilkan dengan garis berwarna biru putus-putus.

2. Kurva *ROC* tergolong “Baik” jika kurva mendekati 0,1 dan menjauhi garis *baseline*.

Dari hasil *ROC* yang dilakukan, sebagai keterangan, untuk warna oranye mewakili hasil prediksi dan warna biru putus-putus mewakili data *testing*. Berdasarkan panduan membaca kurva *ROC* sebelumnya, dapat dilihat bahwa hasil *ROC* diasumsikan baik, karena nilai kurva dari data prediksi yang diwakilkan dengan warna oranye mendekati titik koordinat 0,1 dan menjauhi garis *baseline*.

4.8 Pengujian User Interface

User Interface digunakan untuk memudahkan *user* awam dalam mengakses dan mengetahui hasil prediksi kerusakan yang diberikan tanpa harus mencocokkan hasil prediksi dengan model *Deep Learning* yang dilakukan sebelumnya.

4.8.1 Pengujian Perubahan Hasil Prediksi menjadi Tulisan Prediksi dan Rekomendasi Tindakan

Sebelum menampilkan hasil prediksi di *user interface*, perlu ada perubahan data hasil prediksi tersebut yang dari angka menjadi tulisan prediksi dan rekomendasi tindakan agar lebih mudah dipahami oleh *user*. Berikut hasil pengujian perubahan hasil prediksi yang telah dilakukan.

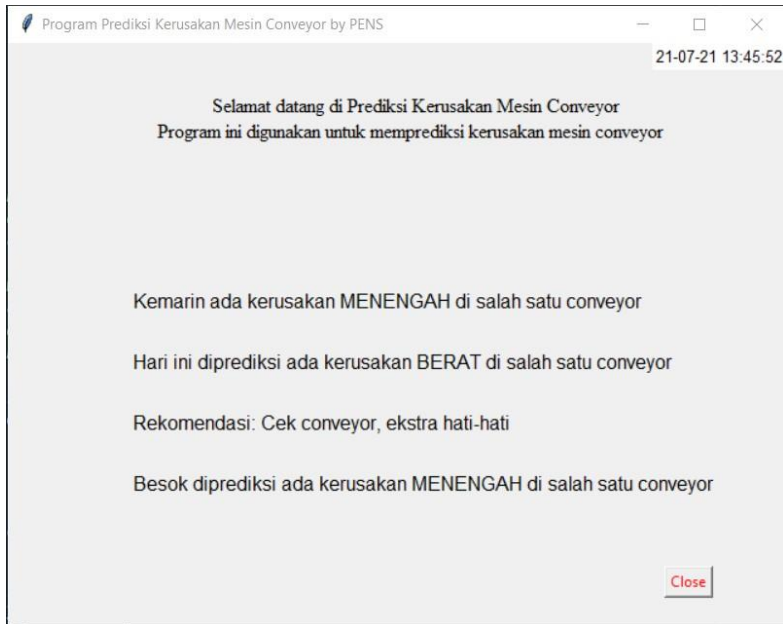
Index	Berat	Berbahaya	Menengah	Normal	Ringan	level	perawatan
0	0	0	0	1	0	NORMAL	Selalu waspada
1	0	0	0	1	0	NORMAL	Selalu waspada
2	0	0	0	0	1	RINGAN	Mulai waspada, lapor ke teknisi
3	0	0	1	0	0	MENENGAH	Mulai cek conveyor, mulai ekstra hati-hati
4	0	0	0	1	0	NORMAL	Selalu waspada
5	0	0	0	0	1	RINGAN	Mulai waspada, lapor ke teknisi
6	0	0	0	1	0	NORMAL	Selalu waspada
7	0	1	0	0	0	BERBAHAYA	Hentikan penggunaan conveyor, Cek conveyor keseluruhan
8	1	0	0	0	0	BERAT	Cek conveyor, ekstra hati-hati
9	0	0	0	1	0	NORMAL	Selalu waspada
10	0	1	0	0	0	BERBAHAYA	Hentikan penggunaan conveyor, Cek conveyor keseluruhan
11	0	1	0	0	0	BERBAHAYA	Hentikan penggunaan conveyor, Cek conveyor keseluruhan

Gambar 4.17 Hasil pengujian Perubahan Hasil Prediksi

Gambar 4.17 merupakan pengujian perubahan hasil prediksi dimana berhasil dilakukan. Pengujian ini mengubah hasil prediksi yang sebelumnya berangka *array* menjadi sebuah tulisan baru yang mudah dibaca oleh *user* dan diteruskan kepada aplikasi *user interface* yang akan dibuat.

4.8.2 Pengujian *User Interface*

Setelah mengubah hasil prediksi menjadi tulisan prediksi dan rekomendasi tindakan, selanjutnya yaitu menampilkan tulisan prediksi tersebut ke dalam *User Interface* yang telah dibuat. *User Interface* diperlukan untuk menampilkan tulisan hasil prediksi yang mudah dibaca oleh orang awam, karena tidak semua orang memahami hasil prediksi yang diberikan dari model *Deep Learning* yang digunakan. Berikut gambar hasil pengujian *User Interface* yang telah dibuat.



Gambar 4.18 Hasil pengujian *User Interface*

Gambar 4.18 merupakan hasil pengujian *user interface* yang dibuat dengan sederhana, dimana terdapat tanggal hari ini, prediksi kerusakan yang terjadi kemarin, prediksi kerusakan yang akan terjadi hari ini, rekomendasi tindakan yang harus dilakukan dan prediksi kerusakan yang akan terjadi keesokan harinya. *Range* waktu untuk prediksinya yaitu 1 sampai 365 hari. Informasi yang ditampilkan yaitu level kerusakan dan rekomendasi tindakan yang harus dilakukan. Kemudian ada *button*

bernama *Close* yang digunakan untuk menutup program tersebut secara sempurna.

-----Halaman ini sengaja dikosongkan-----

BAB V

PENUTUP

5.1 Kesimpulan

Berdasarkan hasil pengujian dan analisa yang telah dilakukan maka dapat diambil kesimpulan sebagai berikut:

1. *Preprocessing* dengan *CountVectorizer* dapat digunakan untuk mengubah dataset dari tulisan menjadi angka *array*, kemudian dilanjutkan dengan *Pre-Processing* dengan *MinMaxScaler* untuk mengubah tipe data dari *CountVectorizer* agar dapat digunakan untuk memprediksi kerusakan mesin *conveyor* dan *Label Binarizer* untuk memberikan label pada setiap kolom *dataset*.
2. *Level* kerusakan mesin *conveyor* ditentukan dari banyaknya kerusakan pada setiap variabel jenis kerusakan.
3. Hasil prediksi kerusakan diperoleh menggunakan metode *LSTM (Long Short-Term Memory)* yaitu salah satu model *Deep Learning* dimana sistem dapat memprediksi kerusakan mesin *conveyor* dan mendapatkan akurasi 97,71%, rata-rata nilai akurasi dengan *KFold* 95,66%, akurasi dengan *confusion matrix* 93,33% dan akurasi dengan *ROC* 99,19%.

5.2 Saran

Berdasarkan penelitian yang telah dilakukan, penulis menyadari bahwa masih banyak kekurangan dalam sistem ini. Hingga saat ini saran yang dapat diberikan adalah:

1. Menguji dengan metode *Deep Learning* lain yang lebih akurat dalam prediksi kerusakan mesin *conveyor*.
2. Melakukan integrasi dengan sistem laporan kerusakan yang sudah ada di PT. JAI.
3. Menambahkan pengujian prediksi untuk lebih dari 1 *conveyor*.

-----Halaman ini sengaja dikosongkan-----

DAFTAR PUSTAKA

- [1] R. Magdalena and D. P. Negara, “Pengukuran Produktivitas dengan Metode Overall Equipment Effectiveness dan OMAX di Lini Produksi TMM1 dan TMM2 PT .,” vol. 20, no. 2019, pp. 131–138, 2020.
- [2] I. A. Samat, H. A. Kamaruddin, S. & Azid, “Maintenance performance measurement: a review,” *J. Int. Pertanika J.Sci& Technol.*, no. 19 (2), 2011.
- [3] E. Kristinawati, “Penentuan Interval Perawatan Mesin Produksi Untuk Meningkatkan Availability Melalui Analisis Keandalan,” *J. Tek. Ind.*, vol. 2, no. 1, p. 36, 2010, doi: 10.22219/jtiumm.vol2.no1.36-46.
- [4] P. Zhou, G. Zhou, H. Wang, D. Wang, and Z. He, “Automatic Detection of Industrial Wire Rope Surface Damage Using Deep Learning-Based Visual Perception Technology,” *IEEE Trans. Instrum. Meas.*, vol. 70, no. c, pp. 1–10, 2021, doi: 10.1109/TIM.2020.3011762.
- [5] S. Y. Shao, W. J. Sun, R. Q. Yan, P. Wang, and R. X. Gao, “A Deep Learning Approach for Fault Diagnosis of Induction Motors in Manufacturing,” *Chinese J. Mech. Eng. (English Ed.)*, vol. 30, no. 6, pp. 1347–1356, 2017, doi: 10.1007/s10033-017-0189-y.
- [6] B. Mohammed, I. Awan, H. Ugail, and M. Younas, “Failure prediction using machine learning in a virtualised HPC system and application,” *Cluster Comput.*, vol. 1, 2019, doi: 10.1007/s10586-019-02917-1.
- [7] K. Celikmih, O. Inan, and H. Uguz, “Failure Prediction of Aircraft Equipment Using Machine Learning with a Hybrid Data Preparation Method,” *Sci. Program.*, vol. 2020, 2020, doi: 10.1155/2020/8616039.
- [8] S. Digital, “Panduan Utama untuk Python Blockchain Bagian 1,” 2019. <https://selembardigital.com/panduan-utama-untuk-python-blockchain-bagian-1/> (accessed Jul. 15, 2021).
- [9] W. Utami, “Praktik Kerja Lapangan/Kerja Praktik BAB II,” pp. 7–16, 2018, [Online]. Available: repository.ittelkom-pwt.ac.id.

- [10] Dqlab, “Belajar Python Mengenal Pandas dan Series untuk Meningkatkan Kompetensi Data,” 2021. <https://www.dqlab.id/belajar-python-mengenal-pandas-dan-series-untuk-meningkatkan-kompetensi-data> (accessed Jul. 16, 2021).
- [11] Dqlab, “Belajar Machine Learning Dengan Library Python : Scikit-Learn,” 2020. <https://www.dqlab.id/belajar-machine-learning-dengan-library-python-scikit-learn> (accessed Jul. 14, 2021).
- [12] KMKLabs, “We’re Doing Machine Learning!,” 2016. <https://blog.kmkonline.co.id/were-doing-machine-learning-9d4075d46cc3> (accessed Jul. 14, 2021).
- [13] J. Hale, “Scale, Standardize, or Normalize with Scikit-Learn,” 2019. <https://towardsdatascience.com/scale-standardize-or-normalize-with-scikit-learn-6ccc7d176a02> (accessed Jul. 12, 2021).
- [14] Skicit-learn, “sklearn.preprocessing.LabelBinarizer,” 2019. <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelBinarizer.html> (accessed Jul. 16, 2021).
- [15] B. B. Benuwa, Y. Zhan, B. Ghansah, D. K. Wornyo, and F. B. Kataka, “A review of deep machine learning,” *Int. J. Eng. Res. Africa*, vol. 24, no. June, pp. 124–136, 2016, doi: 10.4028/www.scientific.net/JERA.24.124.
- [16] A. Shrestha and A. Mahmood, “Review of deep learning algorithms and architectures,” *IEEE Access*, vol. 7, pp. 53040–53065, 2019, doi: 10.1109/ACCESS.2019.2912200.
- [17] B. Yash, “The Challenge of Vanishing/Exploding Gradients in Deep Neural Networks,” 2021. <https://www.analyticsvidhya.com/blog/2021/06/the-challenge-of-vanishing-exploding-gradients-in-deep-neural-networks/> (accessed Jul. 29, 2021).
- [18] Winda Kurnia Sari, D. P. Rini, Reza Firsandaya Malik, and Iman Saladin B. Azhar, “Multilabel Text Classification in News Articles Using Long-Term Memory with Word2Vec,” *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 4, no. 2, pp. 276–285,

2020, doi: 10.29207/resti.v4i2.1655.

- [19] A. M. Khalimi, "Pengujian Data dengan Cross Validation," 2020. <https://www.pengalaman-edukasi.com/2020/04/apa-itu-k-fold-cross-validation.html> (accessed Jul. 28, 2021).
- [20] Kuliahkomputer, "Pengujian Dengan Confusion Matrix," *July, 23rd* 2018, 2018. <http://www.kuliahkomputer.com/2018/07/pengujian-dengan-confusion-matrix.html> (accessed Jul. 15, 2021).
- [21] M. K. DR. MARIA SUSAN ANGGREANY, S.KOM., "Confusion Matrix," 2020. <https://socs.binus.ac.id/2020/11/01/confusion-matrix/> (accessed Jul. 17, 2021).
- [22] Skicit-learn, "Receiver Operating Characteristic (ROC)," 2019. https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html (accessed Jul. 16, 2021).
- [23] Reyza, "Menghitung kinerja algoritma klasifikasi: Pilih ROC Curve atau Precision-Recall Curve?," 2017. <https://www.rezafaisal.net/?p=3068> (accessed Jul. 17, 2021).
- [24] A. M. Khalimi, "Cara Menghitung Confusion Matrix 4 Kelas," 2020. <https://www.pengalaman-edukasi.com/2020/01/confusion-matrix-multi-class-menghitung.html> (accessed Jul. 17, 2021).
- [25] A. Setiaji, "Machine Learning : Accuracy, Recall & Precision," 2018. <https://mragungsetiaji.github.io/python/machine-learning/2018/09/21/machine-learning-accuracy-recall-dan-precision.html> (accessed Jul. 28, 2021).

-----Halaman ini sengaja dikosongkan-----

DAFTAR RIWAYAT HIDUP



Nama	: Eky Bintarno Wicaksono
NRP	: 2210171023
Tempat Lahir	: Sidoarjo
Tanggal Lahir	: 21 Mei 1998
Agama	: Islam
Alamat	: Desa Cemeng Bakalan RT 23 RW
05 Sidoarjo, Jawa Timur	
No Telp	: 081232979931
Email	: ekyk.1998@gmail.com
Riwayat Pendidikan	:
2005-2011	: SDN Cemeng Bakalan 1 Sidoarjo
2011-2014	: SMPN 1 Sidoarjo
2014-2017	: SMAN 3 Sidoarjo
2017-2021	: D4 Teknik Komputer PENS