

**APLIKASI *CV MATCHER* UNTUK MELIHAT KECOCOKAN
DAFTAR RIWAYAT HIDUP DENGAN LOWONGAN
PEKERJAAN MENGGUNAKAN *MACHINE*
LEARNING DAN METODE *COSINE*
SIMILARITY BERBASIS WEB**

SKRIPSI

**Karya Tulis sebagai syarat untuk memperoleh
Gelar Sarjana Komputer dari Fakultas Teknologi Informasi
Universitas Bale Bandung**

Disusun oleh :

**WILDAN ADHITYA GERALDINE
NPM. 301180016**



**PROGRAM STRATA I
PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS BALE BANDUNG
BANDUNG
2022**

LEMBAR PERSETUJUAN PEMBIMBING

**APLIKASI *CV MATCHER* UNTUK MELIHAT KECOCOKAN
DAFTAR RIWAYAT HIDUP DENGAN LOWONGAN
PEKERJAAN MENGGUNAKAN *MACHINE
LEARNING* DAN METODE *COSINE
SIMILARITY* BERBASIS WEB**

Disusun oleh :

**WILDAN ADHITYA GERALDINE
NPM. 301180016**

Telah diterima dan disetujui untuk persyaratan mencapai gelar
SARJANA KOMPUTER

Pada

**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS BALE BANDUNG**

Baleendah, Agustus 2022

Disetujui oleh:

Pembimbing Utama

Pembimbing Pendamping

Yudi Herdiana S. T, M. T.
NIK. 04104808008

Yusuf Muharam M. Kom.
NIK. 04104820003

LEMBAR PERSETUJUAN PENGUJI

**APLIKASI *CV MATCHER* UNTUK MELIHAT KECOCOKAN
DAFTAR RIWAYAT HIDUP DENGAN LOWONGAN
PEKERJAAN MENGGUNAKAN *MACHINE*
LEARNING DAN METODE *COSINE*
SIMILARITY BERBASIS WEB**

Disusun oleh :

WILDAN ADHITYA GERALDINE
NPM. 301180016

Telah diterima dan disetujui untuk persyaratan mencapai gelar
SARJANA KOMPUTER

Pada

**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS BALE BANDUNG**

Baleendah, Agustus 2022

Disetujui oleh:

Penguji 1

Penguji 2

Dr. Rustiyana, S. T, M. T.
NIK. 04104808015

Yaya Suharya, S. Kom, M. T.
NIK. 01043170007

LEMBAR PENGESAHAN PROGRAM STUDI

**APLIKASI *CV MATCHER* UNTUK MELIHAT KECOCOKAN
DAFTAR RIWAYAT HIDUP DENGAN LOWONGAN
PEKERJAAN MENGGUNAKAN *MACHINE
LEARNING* DAN METODE *COSINE
SIMILARITY* BERBASIS WEB**

Disusun oleh :

**WILDAN ADHITYA GERALDINE
NPM. 301180016**

Telah diterima dan disetujui untuk persyaratan mencapai gelar
SARJANA KOMPUTER

Pada

**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS BALE BANDUNG**

Baleendah, Agustus 2022

Mengetahui,
Dekan,

Yudi Herdiana S. T, M, T.
NIK. 04104808008

Mengesahkan,
Ketua Program Studi,

Yusuf Muharam M, Kom.
NIK. 04104820003

LEMBAR PERNYATAAN

Yang bertanda tangan dibawah ini, saya :

Nama : Wildan Adhitya Geraldine
NPM : 301180016
Jurusan : Teknik Informatika
Fakultas : Teknologi Informasi
Judul : Aplikasi *CV Matcher* Untuk Melihat Kecocokan
Daftar Riwayat Hidup Dengan Lowongan
Pekerjaan Menggunakan *Machine Learning* Dan
Metode *Cosine Similarity* Berbasis Web.

Dengan ini saya menyatakan bahwa skripsi ini benar-benar karya saya sendiri. Sepengetahuan saya tidak terdapat karya yang ditulis atau diterbitkan orang lain, kecuali sebagai acuan atau kutipan dengan mengikuti tata penulisan karya ilmiah yang lazim.

Bandung, Agustus 2022

Yang menyatakan,

Wildan Adhitya Geraldine
NPM. 301180016

ABSTRAK

Tugas akhir skripsi merupakan salah satu syarat dalam menyelesaikan program strata I (satu). Sebelum memulai pengerjaan skripsi ini, dilakukan terlebih dahulu pengajuan proposal skripsi, yang mana akan menjadi sebuah dasar dalam pengerjaan tugas akhir skripsi. Berdasarkan laporan Badan Pusat Statistik (BPS), jumlah pengangguran di Indonesia pada Agustus 2021 adalah sebesar 9,10 juta penduduk. Jumlah itu menurun dibanding jumlah pengangguran setahun sebelumnya yang mencapai 9,77 juta orang. Permasalahan yang saat ini banyak dijumpai adalah ketatnya persaingan dalam memperoleh pekerjaan sehingga para pelamar kerja harus mempersiapkan lamaran kerja dengan matang, salah satunya yaitu berkas CV yang dapat menunjukkan keunggulan pelamar kerja.

Tujuan dari penelitian ini adalah membuat aplikasi CV Matcher yang dapat membantu pelamar kerja dalam mengecek kesesuaian Curriculum Vitae yang mereka buat dengan kriteria kandidat yang sudah ditentukan oleh perusahaan yang dituju. Penelitian ini menggunakan metode kualitatif yaitu dengan melakukan studi literatur. Selanjutnya dilanjutkan dengan tahapan perancangan sebagai gambaran umum dari aplikasi yang dibangun. Pada penelitian ini, model pengembangan pada perancangan sistem yang digunakan oleh penulis adalah model AI Project Cycle.

Ada 6 tahapan pada model AI Project Cycle diantaranya adalah Problem Scoping, Data Acquisition, Data Exploration, Modelling, Evaluation, Deployment. Kemudian secara detail mengenai rancangan aplikasi pada tahap Deployment AI Project Cycle yaitu dengan menggunakan UML (Unified Modelling Language) yang meliputi use case diagram, dan activity diagram. Kemudian untuk beberapa Software yang digunakan dalam melakukan penelitian ini adalah Balsamiq Mockup untuk merancang desain tampilan aplikasi. Google Colaboratory digunakan untuk merancang model Machine Learning yang akan digunakan. Selanjutnya untuk bahasa pemrograman yang digunakan adalah python 2.7. Untuk perancangan aplikasi menggunakan Unified Model Language (UML) dan HTML serta CSS Bootstrap untuk implementasi tampilannya. Hasil dari penelitian ini adalah dengan adanya aplikasi CV Matcher yang dapat mencocokkan CV pelamar kerja dengan deskripsi lowongan pekerjaan yang akan dilamar.

Kata Kunci: Artificial Intelligence, CV, Lowongan Pekerjaan, Pelamar Kerja

ABSTRACT

The final thesis is one of the requirements in completing the strata I (one) program. Before starting to work on this thesis, it is necessary to submit a thesis proposal first, which will be the basis for working on the final thesis. Based on a report from the Central Statistics Agency (BPS), the number of unemployed in Indonesia in August 2021 was 9.10 million people. That number decreased compared to the number of unemployed a year earlier which reached 9.77 million people. The problem that is currently encountered is the intense competition in getting a job, so job applicants must prepare job applications carefully, one of which is a CV file that can show the advantages of job applicants.

The purpose of this study is to Create a CV Matcher application that can assist job applicants in checking the suitability of the Curriculum Vitae they have made with the candidate criteria that have been determined by the intended company. This study uses a qualitative method, namely by conducting a literature study. Then proceed with the design stage as a general description of the application being built. In this study, the development model in system design used by the author is the AI Project Cycle model.

There are 6 stages in the AI Project Cycle model including Problem Scoping, Data Acquisition, Data Exploration, Modelling, Evaluation, Deployment. Then in detail about the application design at the AI Project Cycle Deployment stage, namely by using UML (Unified Modelling Language) which includes Use case diagrams, and activity diagrams. Then for some of the Software used in doing this design is the Balsamiq Mockup to design the display design of the application. Google Collaboratory is used to design the Machine Learning model that will be used. Next for the programming language used is python 2.7. For application design using Unified Model Language (UML) and HTML and CSS Bootstrap for the implementation of the appearance. The result of this research is that there is a CV Matcher application that can match the CV of job applicants with a description of the job vacancies to be applied for.

Keywords: Artificial Intelligence, CV, Job Vacancies, Job applicants

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa, karena dengan rahmat dan karunia-Nya, penulis dapat menyelesaikan Proposal Skripsi. Adapun maksud dari proposal ini adalah sebagai dasar untuk menyusun Laporan Tugas Akhir Skripsi, dalam rangka menyelesaikan Program Strata 1 (satu) Teknik Informatika Universitas Bale Bandung.

Penulis mengucapkan terimakasih sebesar – besarnya kepada:

1. Allah SWT. Yang telah memberi kelancaran dalam pengerjaan skripsi ini.
2. Orang tua saya tercinta, Bapak Asep Sopiyan dan Ibu Susanti Yulianti yang selalu mendukung saya.
3. Yudi Herdiana, S.T, M.T, selaku Dekan Fakultas Teknologi Informasi Universitas Bale Bandung, dan selaku pembimbing utama skripsi.
4. Yusuf Muharam, M.Kom, selaku Kaprodi Teknik Informatika Fakultas Teknologi Informasi Universitas Bale Bandung, dan selaku pembimbing pendamping.
5. Dosen-dosen Universitas Bale Bandung yang senantiasa membimbing saya.
6. Teman-teman saya yang selalu memberi semangat kepada saya.
7. Seluruh pihak yang telah mendukung atas kelancaran pengerjaan skripsi ini.

Akhir kata, penulis berharap semoga Proposal Skripsi ini dapat bermanfaat bagi semua pihak yang membacanya.

Bandung, Agustus 2022

Penulis

DAFTAR ISI

ABSTRAK	vi
ABSTRACT	vii
KATA PENGANTAR.....	viii
DAFTAR ISI.....	ix
DAFTAR GAMBAR.....	xii
DAFTAR TABEL	xiv
DAFTAR LAMPIRAN	xv
BAB I PENDAHULUAN.....	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Batasan Masalah.....	3
1.4 Tujuan.....	3
1.5 Metodologi Penelitian	4
1.6 Sistematika Penulisan.....	4
BAB II TINJAUAN PUSTAKA.....	5
2.1 Landasan Teori	5
2.1.1 <i>Multilabel Text Classification</i> Menggunakan SVM dan <i>Doc2Vec Classification</i> pada Dokumen Berita Bahasa Indonesia	12
2.1.2 Penerapan Metode Klasifikasi <i>Support Vector Machine</i> (Svm) Pada Data Akreditasi Sekolah Dasar (Sd) Di Kabupaten Magelang.....	13
2.1.3 Penerapan Algoritma <i>Support Vector Machine</i> (SVM) Dengan TF-IDF N-Gram Untuk <i>Text Classification</i>	14
2.1.4 Perbandingan Algoritma <i>Random Forest Classifier</i> , <i>Support Vector Machine</i> dan <i>Logistic Regression Classifier</i> Pada Masalah <i>High Dimension</i> (Studi Kasus: Klasifikasi <i>Fake News</i>)	14
2.1.5 <i>Text Summarization using Clustering Technique</i>	16
2.1.6 Peringkasan Sentimen Esktraktif di Twitter Menggunakan <i>Hybrid TF-IDF</i> dan <i>Cosine Similarity</i>	17
2.2 Dasar Teori	18
2.2.1 Kecerdasan Buatan	18

2.2.2 <i>Machine Learning</i>	19
2.2.3 <i>Natural Processing Language (NLP)</i>	21
2.2.4 <i>Support Vector Machine (SVM)</i>	24
2.2.5 <i>Cosine Similarity</i>	25
2.2.5 <i>AI Project Cycle</i>	26
2.2.6 <i>(Unified Model Language) UML</i>	37
2.2.7 <i>Balsamiq Mockup 3</i>	39
BAB III METODOLOGI PENELITIAN	40
3.1 Kerangka Pikir.....	40
3.2 Deskripsi.....	41
3.2.1 Identifikasi Masalah.....	41
3.2.2 Pengumpulan Data.....	41
3.2.3 Analisis Kebutuhan Sistem.....	41
3.2.4 Perancangan	42
3.2.5 Pengujian	44
3.2.6 Implementasi.....	44
3.2.7 Pelaporan	44
BAB IV ANALISIS DAN PERANCANGAN	45
4.1 Analisis.....	45
4.1.1 Analisis Masalah.....	45
4.1.2 Analisis Perangkat	45
4.1.3 Analisis Pengguna.....	47
4.1.4 <i>User Interface</i>	47
4.1.5 Fitur - Fitur.....	48
4.1.6 Analisa Data.....	49
4.1.6 Analisa Biaya.....	49
4.2 Perancangan.....	50
4.2.1 Deskripsi <i>CV Matcher</i>	50
4.2.2 Perancangan <i>AI Project Cycle</i>	51
BAB V IMPLEMENTASI DAN PENGUJIAN	71
5.1 Implementasi	71

5.1.1 Listing Program	71
5.1.2 Impelementasi Sistem	85
5.1.3 Spesifikasi Sistem	85
5.2 Pengujian	88
BAB VI KESIMPULAN	90
5.1 Kesimpulan.....	90
5.2 Saran	90
DAFTAR PUSTAKA	91

DAFTAR GAMBAR

Gambar 2.1 Hasil penelitian Anjali.....	16
Gambar 2.2 Persamaan matematis SRM.....	25
Gambar 2.3 Persamaan matematis <i>Cosine Similarity</i>	26
Gambar 2.4 <i>AI Project Cycle</i>	26
Gambar 2.5 Ilustrasi pemodelan.	32
Gambar 2.6 Persamaan <i>Accuration</i>	35
Gambar 2.7 Persamaan <i>Precision</i>	36
Gambar 2.8 Persamaan <i>Recall</i>	36
Gambar 2.9 Representasi Akurasi, Presisi, dan <i>Recall</i>	36
Gambar 2.10 <i>Model Selection</i>	37
Gambar 2.11 Keterangan <i>Use case diagram</i>	38
Gambar 2.12 Keterangan <i>Activity diagram</i>	38
Gambar 3.1 Kerangka Pikir.....	40
Gambar 4.1 Alur Program <i>CV Matcher</i>	50
Gambar 4.2 Tampilan Tabel Metadata <i>Dataset</i> (file .csv).....	54
Gambar 4.3 Hasil EDA terhadap Kategori CV pada <i>Dataset</i>	54
Gambar 4.4 Rumus <i>Cosine Similarity</i>	61
Gambar 4.5 Ilustrasi <i>Cosine Similarity</i>	62
Gambar 4.6 <i>Use case diagram</i>	65
Gambar 4.7 <i>Activity diagram</i> Pengecekan	67
Gambar 4.8 <i>Activity diagram</i> Beranda.....	68
Gambar 4.9 <i>Activity diagram</i> Pengembang	68
Gambar 4.10 <i>Activity diagram</i> Informasi.....	68
Gambar 4.11 Desain Tampilan Beranda	69
Gambar 4.12 Desain Tampilan Pengembang	69
Gambar 4.13 Desain Tampilan Informasi	70
Gambar 4.14 Desain Tampilan Pengecekan	70
Gambar 4.15 Desain Tampilan Hasil Pengecekan	70
Gambar 5.1 Tampilan Beranda Aplikasi <i>CV Matcher</i>	86

Gambar 5.2 Tampilan Developer Team.....	87
Gambar 5.3 Tampilan Informasi Program <i>CV Matcher</i>	87
Gambar 5.4 Tampilan Pengecekan CV dengan Lowongan Pekerjaan	87
Gambar 5.5 Tampilan Hasil Pengecekan CV dengan Lowongan Pekerjaan	88

DAFTAR TABEL

Tabel 2.1 Referensi Jurnal.....	5
Tabel 4.1 Daftar Perangkat Keras	45
Tabel 4.2 Daftar Kebutuhan Perangkat Lunak.....	46
Tabel 4.3 Daftar Kebutuhan Biaya.....	49
Tabel 4.3 Hasil Evaluasi Model Klasifikasi Kategori CV	63
Tabel 4.4 Deskripsi Aktor	65
Tabel 4.5 Deskripsi <i>Use case</i> pada Aplikasi	66
Tabel 5.1 Spesifikasi Minimum <i>Hardware</i> Program <i>CV Matcher</i>	85
Tabel 5.2 Spesifikasi Minimum <i>Software</i> Program <i>CV Matcher</i>	85
Tabel 5.3 Tabel Pengujian Aplikasi <i>CV Matcher</i>	88

DAFTAR LAMPIRAN

Lampiran A-1. Daftar Riwayat Hidup	94
--	----

BAB I

PENDAHULUAN

1.1 Latar Belakang

Berdasarkan laporan Badan Pusat Statistik (BPS), jumlah pengangguran di Indonesia pada Agustus 2021 adalah sebesar 9,10 juta penduduk. Jumlah itu menurun dibanding jumlah pengangguran setahun sebelumnya yang mencapai 9,77 juta orang (*BPS: Tingkat Pengangguran Terbuka Pada Agustus 2021 Turun 0,58 Persen Dibanding Agustus 2020 - Data Tempo.Co*, 2021). Hal ini menunjukkan bahwa masih banyak nya jumlah pengangguran. Hal ini tentunya menjadi salah satu faktor kegagalan para pelamar kerja dalam proses seleksi penerimaan karyawan di suatu perusahaan..

Livecareer.com merupakan sebuah website penyedia informasi mengenai lowongan pekerjaan yang mencakup dari beberapa perusahaan di seluruh dunia seperti *Amazon*, *Google*, *AWS*, dan perusahaan lainnya. Selain itu livecareer.com juga merupakan website penyedia informasi tentang pembuatan daftar riwayat hidup bagi para pelamar kerja. Livecareer.com juga menyediakan berbagai artikel informasi yang dapat berguna bagi para pelamar kerja untuk melamar suatu pekerjaan. Hal ini juga didukung oleh berbagai fitur yang disediakan seperti fitur pembuatan daftar riwayat hidup dengan ribuan desain menarik yang bisa digunakan.

Disamping berbagai fitur menarik yang telah disediakan oleh website livecareer.com, namun dalam penelusuran penulis pada website tersebut masih ada beberapa fitur yang kurang bagi pengguna terutama pelamar kerja untuk menyiapkan daftar riwayat hidup atau berkas lamaran yang dimilikinya. Salah satunya adalah penulis menemukan bahwa pada website livecareer.com masih belum memiliki fitur untuk melakukan pengecekan kecocokan berkas lamaran pelamar kerja dengan lowongan pekerjaan yang akan dilamar. Sehingga hal tersebut dirasa kurang meyakinkan pelamar kerja untuk memiliki peluang diterima terhadap suatu lowongan pekerjaan tersebut.

Berdasarkan jurnal yang diterbitkan oleh Jurnal Aghinya Stiesnu, (Handoko, 2020) menyatakan bahwa rekrutmen adalah proses mencari, menemukan dan menarik para pelamar yang kapabel untuk dipekerjakan dalam suatu organisasi. Selain itu, rekrutmen juga dapat diartikan sebagai proses untuk mendapatkan sejumlah SDM atau karyawan yang berkualitas untuk menduduki suatu jabatan atau pekerjaan disebuah perusahaan. Pada sebuah proses rekrutmen, akan diadakan tahap seleksi. Seleksi adalah proses memilih pelamar sampai dengan memutuskan pelamar mana yang diterima atau yang ditolak untuk dijadikan pegawai. Salah satu tahapan seleksi pada lowongan kerja adalah tahapan seleksi surat - surat lamaran. Maka dari itu, berkas lamaran seperti daftar riwayat hidup atau surat lamaran harus disiapkan sebaik-baiknya oleh pelamar kerja, agar dapat sesuai dengan lowongan pekerjaan yang akan dilamar.

Maka dari itu, penulis mengangkat topik penelitian ini dengan judul “Aplikasi *CV Matcher* Untuk Melihat Kecocokan Daftar Riwayat Hidup Dengan Lowongan Pekerjaan Menggunakan *Machine Learning* Dan Metode *Cosine Similarity* Berbasis Web”. Sehingga diharapkan aplikasi ini dapat ikut membantu para pelamar kerja untuk mengetahui kecocokan daftar riwayat hidup yang dimiliki dengan deskripsi lowongan pekerjaan yang akan dilamarnya.

1.2 Rumusan Masalah

Berdasarkan latar belakang masalah tersebut, maka perumusan masalah adalah :

- a. Bagaimana membuat sebuah model untuk mengklasifikasikan jenis daftar riwayat hidup.
- b. Bagaimana membuat sebuah perhitungan untuk membandingkan kesesuaian isi dari daftar riwayat hidup dengan deskripsi lowongan kerja.
- c. Bagaimana cara menerapkan model dan perhitungan untuk mendapatkan skor kesesuaian isi daftar riwayat hidup dengan lowongan kerja ke dalam sebuah aplikasi yang siap digunakan dan diakses oleh masyarakat dengan mudah.

1.3 Batasan Masalah

Batasan masalah terhadap penelitian ini adalah sebagai berikut :

- a. Kecerdasan buatan yang digunakan adalah *Machine Learning* dengan model :
 - 1) *SVM (Support Vector Machine)*,
 - 2) *KNN Classifier*,
 - 3) *Multinomial Naive Bayes*,
 - 4) *Logistic Regression*,
 - 5) *Decission Tree Classifier*,
 - 6) *Random Forest Classifier*.
- b. Perhitungan yang digunakan untuk mendapatkan skor kesesuaian isi daftar riwayat hidup dengan lowongan kerja adalah menggunakan metode *Cosine Similarity*.
- c. Bahasa pemrograman yang digunakan adalah *python* versi 3.x.
- d. *Framework* web yang digunakan adalah *Flask*.
- e. *Deployment* aplikasi menggunakan hosting *herokuapp* versi gratis.
- f. *Dataset* yang digunakan berasal dari platform www.kaggle.com. *Dataset* tersebut hasil dari metode *Web Scrapping* dari website livecareer.com yang dilakukan oleh Snehaan Bhawal.

1.4 Tujuan

Tujuan yang ingin dicapai dalam penelitian ini adalah sebagai berikut.

- a. Membuat aplikasi *CV Matcher* yang dapat membantu pelamar kerja dalam mengecek kesesuaian *Curriculum Vitae* yang mereka buat dengan kriteria kandidat yang sudah ditentukan oleh perusahaan yang dituju.
- b. Sebagai salah satu cara untuk mencapai tujuan dalam *Sustainable Development Goals* (SDGs) yaitu pekerjaan layak dan pertumbuhan ekonomi.

1.5 Metodologi Penelitian

Dalam penelitian ini, penulis menggunakan metode kualitatif yaitu dengan melakukan studi literatur. Selanjutnya dilanjutkan dengan tahapan perancangan sebagai gambaran umum dari aplikasi yang dibangun. Pada tahap ini, model pengembangan pada perancangan sistem yang digunakan oleh penulis adalah model *AI Project Cycle*. Ada 6 tahapan pada model *AI Project Cycle* diantaranya adalah *Problem Scoping*, *Data Acquisition*, *Data Exploration*, *Modelling*, *Evaluation*, *Deployment*. Kemudian secara detail mengenai rancangan aplikasi pada tahap *Deployment AI Project Cycle* yaitu dengan menggunakan UML (*Unified Modelling Language*) yang meliputi *use case diagram*, dan *activity diagram*.

1.6 Sistematika Penulisan

Dalam menyusun laporan skripsi ini diatur dan disusun dalam lima bab, yang masing-masing terdiri dari beberapa sub bab. Adapun urutannya adalah sebagai berikut:

BAB I : PENDAHULUAN

Bagian pendahuluan berisi mengenai latar belakang masalah, rumusan masalah, batasan masalah, tujuan penelitian, metodologi penelitian dan sistematika penulisan.

BAB II : TINJAUAN PUSTAKA

Bagian ini berisi mengenai landasan teori dan dasar teori.

BAB III : METODOLOGI PENELITIAN

Bagian ini berisi mengenai kerangka pikir dan deskripsi.

BAB IV : ANALISIS, PERANCANGAN DAN HASIL

Bagian ini membahas mengenai Analisis Masalah, Analisis *Software*, Analisis Pengguna, *User Interface*, Fitur – fitur, Analisis Data, Analisis Biaya dan Perancangan.

BAB V : IMPLEMENTASI DAN PENGUJIAN

Bagian ini membahas mengenai Implementasi dan Pengujian.

BAB VI : KESIMPULAN DAN SARAN

Bagian ini membahas mengenai kesimpulan dan saran.

BAB II

TINJAUAN PUSTAKA

2.1 Landasan Teori

Landasan teori ini berisi mengenai informasi-informasi penelitian yang berkaitan dengan topik penelitian ini yang pernah dilakukan sebelumnya. Hal ini dimaksudkan untuk melakukan perbandingan mengenai kelebihan dan kekurangan yang sudah ada. Berbagai informasi landasan teori ini didapatkan dari berbagai jurnal-jurnal penelitian yang ada.

Tabel 2.1 Referensi Jurnal

No	Judul	Masalah	Metode	Solusi
1	<i>Multilabel Text Classification</i> Menggunakan SVM dan <i>Doc2Vec</i> pada Dokumen Berita Bahasa Indonesia	Penelitian yang dilatar belakangi kesulitan untuk mengenali jenis kategori dari informasi tersebut satu persatu.	Sistem dibuat dengan menggunakan <i>Python</i> , memanfaatkan <i>Doc2Vec</i> untuk mengambil fitur <i>dataset</i> , dan SVM untuk melakukan klasifikasi terhadap banyak kelas.	Berupa model <i>Machine Learning</i> yang bertujuan untuk dapat mengidentifikasi dan mengklasifikasikan dokumen-dokumen berita dalam bahasa Indonesia ke dalam beberapa kategori sekaligus, maka dibuatlah sebuah penelitian berupa sistem untuk menangani

				<p>klasifikasi dokumen teks dalam bahasa Indonesia. Sistem tersebut akan memproses berita-berita yang diberikan, dan kemudian akan memberikan 2 kategori yang paling mendekati terhadap isi dari berita tersebut.</p>
2	<p>Penerapan metode klasifikasi <i>Support Vector Machine</i> (SVM) pada data akreditasi sekolah dasar (SD) di Kabupaten Magelang</p>	<p>Penulisan jurnal tersebut dilatar belakangi oleh program akreditasi sekolah dasar yang dilaksanakan di Kabupaten Magelang.</p>	<p>Dalam penelitian ini akan dilakukan sebuah penerapan metode klasifikasi SVM (<i>Support Vector Machine</i>) pada program akreditasi di Kabupaten Magelang tersebut.</p>	<p>Hasil penelitian tersebut berupa model SVM yang menunjukkan bahwa model SVM yang menggunakan kernel <i>Gaussian Radial Basic Function</i> (RBF) memiliki tingkat akurasi klasifikasi sebesar 100%. Sedangkan menggunakan fungsi kernel <i>Polynomial</i></p>

				akurasi klasifikasinya sebesar 98,810%.
3	Penerapan algoritma <i>Support Vector Machine</i> (Svm) dengan Tf-Idf N-Gram untuk <i>Text Classification</i>	Penelitian ini dilakukan untuk mengklasifikasi artikel ilmiah ke dalam kategori sesuai dengan fokus dan ruang lingkup yang terdapat pada laman <i>Syntax</i> Jurnal Informatika secara otomatis dengan memanfaatkan proses <i>Text Mining</i> .	Metodologi penelitian yang digunakan adalah <i>Knowledge Discovery Database</i> (KDD) dengan tahapan <i>data selection</i> , <i>preProcessing</i> , <i>transformation</i> , <i>modelling</i> dan <i>Evaluation</i> . Penelitian ini akan membandingkan klasifikasi berdasarkan judul pada artikel. Adapun algoritma yang digunakan adalah <i>Support Vector Machine</i> (SVM	Hasil penelitian setelah dilakukan pengujian terhadap model diukur dengan nilai <i>Accuracy</i> , <i>Precision</i> , <i>Recall</i> dan <i>F-Measure</i> . Hasil terbaik adalah <i>Accuracy</i> sebesar 70%, <i>Precision</i> sebesar 75%, <i>Recall</i> sebesar 69% dan <i>F-Measure</i> sebesar 71% pada skenario perbandingan 90:10 dan kernel <i>linear</i> .

) dengan menggunakan empat kernel SVM, diantaranya adalah kernel <i>linear</i> , kernel <i>polynomial</i> , kernel <i>sigmoid</i> dan kernel RBF.	
4	Perbandingan Algoritma <i>Random Forest Classifier</i> , <i>Support Vector Machine</i> dan <i>Logistic Regression Classifier</i> Pada Masalah <i>High Dimension</i> (Studi Kasus: Klasifikasi <i>Fake News</i>).	Penelitian ini di latar belakang oleh <i>Fake News</i> yang merupakan informasi palsu yang menyerupai seakan-akan itu adalah benar. Berita dapat juga dikatakan sebagai senjata politik yang kebenarannya tidak bisa dipertanggungjawabkan yang disebarkan secara sengaja untuk mencapai suatu tujuan tertentu.	Metode yang digunakan dalam penelitian dengan menggunakan <i>Random Forest Classifier</i> (RF C), <i>Support Vector Machine</i> (SVM) dan <i>Logistic Regression</i> (LR) dengan <i>high dimension</i> dan hasil penelitian ini untuk mendapatkan perbandingan nilai akurasi	Hasil dari percobaan pada <i>high dimension</i> klasifikasi SVM untuk <i>dataset</i> . Dari percobaan, nilai <i>Training Accuracy</i> dengan nilai persentase sebesar 99,78%. Untuk <i>Testing Accuracy</i> , dengan nilai persentase sebesar 99,98%. Hasil percobaan, nilai LR pada <i>high dimension Training Accuracy</i> dengan nilai persentase

			<p>pada masing-masing metode yang digunakan.</p>	<p>sebesar 99,47%. Untuk <i>Testing Accuracy</i>, nilai tertinggi dimiliki oleh skenario dengan nilai persentase sebesar 99,20%. Nilai performa terbaik dengan data <i>high dimension</i> dari semua percobaan terdapat pada SVM untuk nilai akurasi <i>dataset</i> dengan <i>Training set Accuracy Test</i> persentase 99,97%. <i>Test set Accuracy</i> persentasenya 99,77 %. Untuk hasil <i>Performance Measurements</i> dari percobaan yang memiliki hasil tertinggi dari klasifikasi SVM untuk <i>dataset</i>, untuk</p>
--	--	--	--	---

				<p><i>Sensitivity</i> dengan nilai persentase sebesar 99,75%. Untuk <i>Specificity</i> nilai persentase sebesar 99,80%. Untuk <i>Precision</i> nilai persentase sebesar 99,80%. Untuk <i>F1-Score</i> nilai persentase sebesar 99,80%. Untuk <i>Accuracy</i> nilai persentase sebesar 99,78%. Untuk <i>Error</i> nilai persentase sebesar 0,23%. Nilai performa <i>Sensitivity</i> terbaik dari semua skenario percobaan LRC dengan persentase 99,96 % dibandingkan dengan yang lain.</p>
5	<i>Text Summarization using</i>	Penelitian ini di latar belakang bagaimana caranya untuk menemukan sebuah	Dalam melakukan penelitian ini menggunakan	Hasil pada penelitian ini adalah berupa perbandingan

	<i>Clustering Technique</i>	informasi yang diinginkan dalam sebuah <i>Text</i> dengan kosakata yang cukup banyak. Salah satu cara yang paling umum adalah dengan mereduksi kata yang tidak berhubungan dengan kata kunci yang kita berikan.	metode <i>Cosine Similarity</i> yang digunakan untuk mengukur kemiripan 2 buah <i>Text</i> .	metode <i>Clustering</i> yang digunakan dengan melihat evaluasi <i>confusion matrix</i> pada gambar grafik penjelasan jurnal.
6	Peringkasan Sentimen Esktraktif di Twitter Menggunakan <i>Hybrid TF-IDF</i> dan <i>Cosine Similarity</i>	Penelitian Peringkasan Sentimen Esktraktif di Twitter Menggunakan <i>Hybrid TF-IDF</i> dan <i>Cosine Similarity</i> , dilatar belakangi oleh antusias masyarakat yang memberikan perhatian lebih terhadap akun resmi selebriti di Twitter memunculkan tren penggunaan Twitter sebagai upaya manajemen kesan. Penggalan reaksi masyarakat di media sosial merupakan upaya strategis untuk	Metode <i>Senti Strength</i> digunakan untuk mendapatkan skor kekuatan sentimen dan mengklasifikasi tweet ke dalam kelas positif, negatif dan netral. Tweet bersentimen positif dan negatif diringkaskan dengan cara pemeringkatan tweet menggunakan	Hasil pengujian memperlihatkan bahwa kombinasi <i>Senti Strength</i> , <i>Hybrid TF-IDF</i> , dan <i>Cosine Similarity</i> mampu menghasilkan ringkasan sentimen dengan akurasi yang lebih baik dibandingkan menggunakan <i>Hybrid TF-IDF</i> saja, dengan perolehan akurasi rata-rata sebesar 60% dan <i>F-Measure</i> sebesar

		memperoleh umpan balik, namun tidak mudah dilakukan. Pengguna membutuhkan waktu yang lama untuk membaca ribuan tweet sekaligus memilah sentimennya, sehingga dibutuhkan peringkasan sentimen ekstraktif secara otomatis.	<i>Hybrid</i> TF-IDF yang dikombinasi dengan skor kekuatan sentimen, kemudian menghilangkan tweet yang mirip menggunakan <i>Cosine Similarity</i> .	62%. Hal ini disebabkan karena penambahan kekuatan sentimen sebagai bobot peringkasan.
--	--	--	---	--

2.1.1 *Multilabel Text Classification Menggunakan SVM dan Doc2Vec Classification* pada Dokumen Berita Bahasa Indonesia

Jurnal yang ditulis oleh Kristian Indradiarta Gunawan dan Joan Santoso, mahasiswa Teknik Informatika, Institut Sains dan Teknologi Terpadu Surabaya dengan judul “*Multilabel Text Classification Menggunakan SVM dan Doc2Vec Classification* pada Dokumen Berita Bahasa Indonesia”, tahun 2021 menjelaskan tentang penelitian yang dilatar belakangi kesulitan untuk mengenali jenis dari informasi tersebut satu persatu. Mengutip dari jurnal tersebut hasil penelitiannya adalah berupa model *Machine Learning* yang bertujuan untuk dapat mengidentifikasi dan mengklasifikasikan dokumen-dokumen berita dalam bahasa Indonesia ke dalam beberapa kategori sekaligus, maka dibuatlah sebuah penelitian berupa sistem untuk menangani klasifikasi dokumen teks dalam bahasa Indonesia. Sistem tersebut akan memproses berita-berita yang diberikan, dan kemudian akan memberikan 2 kategori yang paling mendekati terhadap isi dari berita tersebut. Sistem dibuat dengan menggunakan *Python*, memanfaatkan *Doc2Vec* untuk mengambil fitur *dataset*, dan *SVM* untuk melakukan klasifikasi

terhadap banyak kelas. *Dataset* yang digunakan adalah kumpulan dokumen berupa berita-berita yang diperoleh dari CNN Indonesia tahun 2016-2017, dan terbagi dalam 5 kategori berita utama, yaitu: Politik, Ekonomi, Teknologi, Olahraga, dan Hiburan. Dikarenakan sedikitnya literatur untuk klasifikasi *Text* dalam bahasa Indonesia, maka pada penelitian ini hanya menargetkan akurasi sebesar 70% saja. Namun dari hasil ujicoba, akurasi yang diperoleh melebihi 90%. Hasil prediksi untuk kelas dokumen pun memiliki tingkat keberhasilan yang tinggi. Dengan penggunaan *dataset* dan penanganan *preProcessing* yang tepat untuk dokumen bahasa Indonesia, maka hasil yang dicapai bisa lebih bagus dan akurat (Gunawan & Santoso, 2021).

2.1.2 Penerapan Metode Klasifikasi *Support Vector Machine* (Svm) Pada Data Akreditasi Sekolah Dasar (Sd) Di Kabupaten Magelang

Jurnal yang ditulis oleh Puspita Anna Octaviani yang merupakan mahasiswa jurusan statistika PSM UNDIP dan Yuciana Wilandari, Dwi Ispriyanti yang merupakan Staf Pengajar Jurusan Statistika PSM UNDIP menjelaskan tentang Penerapan Metode Klasifikasi *Support Vector Machine* (Svm) Pada Data Akreditasi Sekolah Dasar (Sd) Di Kabupaten Magelang. Penulisan jurnal tersebut dilatar belakangi oleh program akreditasi sekolah dasar yang dilaksanakan di Kabupaten Magelang. Dalam penelitian ini akan dilakukan sebuah penerapan metode klasifikasi SVM (*Support Vector Machine*) pada program akreditasi di Kabupaten Magelang tersebut.

Pada proses klasifikasi tersebut, terdiri dari 3 kategori, yaitu :

- a. 1, untuk sekolah dasar yang memiliki akreditasi A
- b. 2, untuk sekolah dasar yang memiliki akreditasi B
- c. 3, unuk sekolah dasar yang memiliki akreditasi C

Hasil penelitian tersebut menunjukkan bahwa model SVM yang menggunakan kernel *Gaussian Radial Basic Function* (RBF) memiliki tingkat akurasi klasifikasi sebesar 100%. Sedangkan menggunakan fungsi kernel *Polynomial* akurasi klasifikasinya sebesar 98,810% (Octaviani et al., 2014).

2.1.3 Penerapan Algoritma *Support Vector Machine* (SVM) Dengan TF-IDF N-Gram Untuk *Text Classification*

Sebuah jurnal ilmiah yang ditulis oleh Nur Arifin, Ultach Enri, Nina Sulistiyowati Sitanggang menjelaskan mengenai Penerapan Algoritma *Support Vector Machine* (SVM) Dengan Tf-Idf N-Gram Untuk *Text Classification*. Mengutip dari abstraksi jurnal tersebut, penelitian ini dilakukan untuk mengklasifikasi artikel ilmiah ke dalam kategori sesuai dengan fokus dan ruang lingkup yang terdapat pada laman *Syntax* Jurnal Informatika secara otomatis dengan memanfaatkan proses *Text Mining*.

Text Mining merupakan proses yang bertujuan untuk mendapatkan informasi penting dari teks. Metodologi penelitian yang digunakan adalah *Knowledge Discovery Database* (KDD) dengan tahapan *data selection*, *preProcessing*, *transformation*, *modelling* dan *Evaluation*. Penelitian ini akan membandingkan klasifikasi berdasarkan judul pada artikel. Adapun algoritma yang digunakan adalah *Support Vector Machine* (SVM) dengan menggunakan empat kernel SVM, diantaranya adalah kernel *linear*, kernel *polynomial*, kernel *sigmoid* dan kernel RBF. Pembagian data menggunakan *trainTestsplit* dibagi menjadi empat skenario yaitu 60:40, 70:30, 80:30 dan 90:10. Hasil penelitian setelah dilakukan pengujian terhadap model diukur dengan nilai *Accuracy*, *Precision*, *Recall* dan *F-Measure*. Hasil terbaik adalah *Accuracy* sebesar 70%, *Precision* sebesar 75%, *Recall* sebesar 69% dan *F-Measure* sebesar 71% pada skenario perbandingan 90:10 dan kernel *linear* (Arifin et al., 2021).

2.1.4 Perbandingan Algoritma *Random Forest Classifier*, *Support Vector Machine* dan *Logistic Regression Classifier* Pada Masalah *High Dimension* (Studi Kasus: Klasifikasi *Fake News*)

Jurnal yang ditulis oleh Willy, Dian Palupi Rini, Samsuryadi, menjelaskan mengenai penelitian Perbandingan Algoritma *Random Forest Classifier*, *Support Vector Machine* dan *Logistic Regression Classifier* Pada Masalah *High Dimension* (Studi Kasus: Klasifikasi *Fake News*). Penelitian ini di latar belakang oleh *Fake News* yang merupakan informasi palsu yang menyerupai seakan-akan itu adalah

benar. Berita dapat juga dikatakan sebagai senjata politik yang kebenarannya tidak bisa dipertanggungjawabkan yang disebarluaskan secara sengaja untuk mencapai suatu tujuan tertentu.

Dataset yang digunakan pada penelitian ini berjumlah 20000 dan 17 atribut. Metode yang digunakan dalam penelitian dengan menggunakan *Random Forest Classifier*(RFC), *Support Vector Machine*(SVM) dan *Logistic Regression*(LR) dengan *high dimension* dan hasil penelitian ini untuk mendapatkan perbandingan nilai akurasi pada masing-masing metode yang digunakan.

Mengutip dari jurnal ini, berdasarkan hasil dari keseluruhan proses yang dilakukan pada penelitian ini tentang klasifikasi berita palsu yang terdapat di web dengan menggunakan pendekatan dan metode yang diajukan, maka didapatkan kesimpulan *Dataset Fake News high dimension* diambil pada situs <https://www.kaggle.com/> sebanyak 32000 data dan 17 atribut. Sistem identifikasi berita palsu dapat dibuat menggunakan metode *Machine Learning* dengan *high dimension* melalui tahapan *Text PreProcessing* yang meliputi, Case Folding, Punctuation Removal, Number Removal, Removing Word <N Character, *Stemming* dan *Lemmatization*, kemudian masuk ke tahapan Extraction and Selection *Feature* dan terakhir masuk ke tahapan klasifikasi dengan menggunakan metode *Random Forest Classifier*, *Support Vector Machine*, *Logistic Regression Classifier*. Nilai *Training* dan *Test* pada *high dimension* RFC menjelaskan detail hasil *Training Accuracy* 99,76%. Untuk *Testing Accuracy* 99,73%. Dari hasil yang di dapat, *Training Accuracy* hasil akurasi tertinggi.

Hasil dari percobaan pada *high dimension* klasifikasi SVM untuk *dataset*. Dari percobaan, nilai *Training Accuracy* dengan nilai persentase sebesar 99,78%. Untuk *Testing Accuracy*, dengan nilai persentase sebesar 99,98%. Hasil percobaan, nilai LR pada *high dimension Training Accuracy* dengan nilai persentase sebesar 99,47%. Untuk *Testing Accuracy*, nilai tertinggi dimiliki oleh skenario dengan nilai persentase sebesar 99,20%. Nilai performa terbaik dengan data *high dimension* dari semua percobaan terdapat pada SVM untuk nilai akurasi *dataset* dengan *Training set Accuracy Test* persentase 99,97%. *Test set Accuracy* persentasenya 99,77 %. Untuk hasil *Performance Measurements* dari percobaan yang memiliki hasil tertinggi dari klasifikasi SVM untuk *dataset*, untuk *Sensitivity* dengan nilai

persentase sebesar 99,75%. Untuk *Specificity* nilai persentase sebesar 99,80%. Untuk *Precision* nilai persentase sebesar 99,80%. Untuk *F1-Score* nilai persentase sebesar 99,80%. Untuk *Accuracy* nilai persentase sebesar 99,78%. Untuk *Error* nilai persentase sebesar 0,23%. Nilai performa *Sensitivity* terbaik dari semua skenario percobaan LRC dengan persentase 99,96 % dibandingkan dengan yang lain (Willy et al., 2021).

2.1.5 Text Summarization using Clustering Technique

Jurnal yang ditulis oleh Anjali R. Deshpande, Lobo L. M. R. J., menjelaskan mengenai penelitian *Text Summarization using Clustering Technique*. Penelitian ini di latar belakang bagaimana caranya untuk menemukan sebuah informasi yang diinginkan dalam sebuah *Text* dengan kosakata yang cukup banyak. Salah satu cara yang paling umum adalah dengan mereduksi kata yang tidak berhubungan dengan kata kunci yang kita berikan. Dalam melakukan penelitian ini menggunakan metode *Cosine Similarity* yang digunakan untuk mengukur kemiripan 2 buah *Text*. Berikut adalah fungsi matematis dari metode *Cosine Similarity* (Deshpande & L.M.R.J, 2015).

Hasil dari penelitian tersebut adalah sebagai berikut :

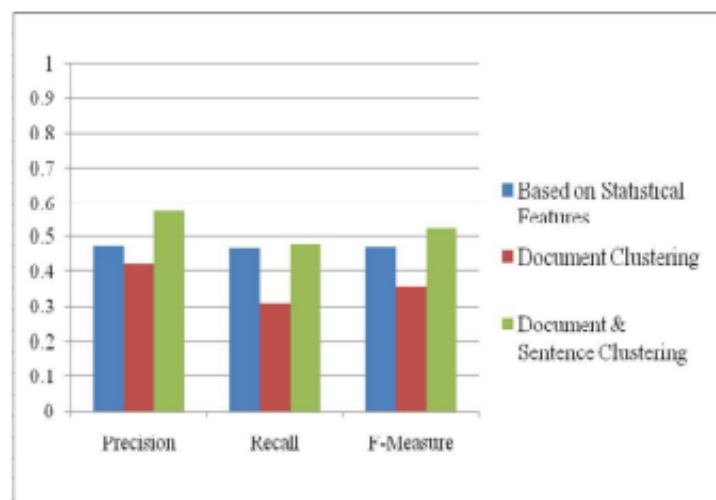


Fig. 2 Summarization performed on Document Collection D1.

Gambar 2.1 Hasil penelitian Anjali

2.1.6 Peringkasan Sentimen Esktraktif di Twitter Menggunakan *Hybrid TF-IDF* dan *Cosine Similarity*

Mengutip dari jurnal yang ditulis oleh Devid Haryalesmana Wahid, Azhari SN mengenai penelitian Peringkasan Sentimen Esktraktif di Twitter Menggunakan *Hybrid TF-IDF* dan *Cosine Similarity*, dilatar belakangi oleh antusias masyarakat yang memberikan perhatian lebih terhadap akun resmi selebriti di Twitter memunculkan tren penggunaan Twitter sebagai upaya manajemen kesan. Penggalan reaksi masyarakat di media sosial merupakan upaya strategis untuk memperoleh umpan balik, namun tidak mudah dilakukan. Pengguna membutuhkan waktu yang lama untuk membaca ribuan tweet sekaligus memilah sentimennya, sehingga dibutuhkan peringkasan sentimen ekstraktif secara otomatis.

Penelitian terdahulu umumnya tidak memasukkan informasi sentimen yang terkandung pada sebuah tweet sebagai bobot peringkat kalimat, sehingga hasil ringkasan masih berupa topik umum yang dibicarakan masyarakat. Penelitian ini bertujuan mengkombinasikan metode *Senti Strength*, *Hybrid TF-IDF* dan *Cosine Similarity* untuk mengekstraksi ringkasan sentimen positif dan negatif masyarakat terhadap topik selebriti di Twitter secara otomatis, dengan artis Agnes Monica sebagai studi kasus. Metode *Senti Strength* digunakan untuk mendapatkan skor kekuatan sentimen dan mengklasifikasi tweet ke dalam kelas positif, negatif dan netral. Tweet bersentimen positif dan negatif diringkas dengan cara pemeringkatan tweet menggunakan *Hybrid TF-IDF* yang dikombinasi dengan skor kekuatan sentimen, kemudian menghilangkan tweet yang mirip menggunakan *Cosine Similarity*.

Hasil pengujian memperlihatkan bahwa kombinasi *Senti Strength*, *Hybrid TF-IDF*, dan *Cosine Similarity* mampu menghasilkan ringkasan sentimen dengan akurasi yang lebih baik dibandingkan menggunakan *Hybrid TF-IDF* saja, dengan perolehan akurasi rata-rata sebesar 60% dan *F-Measure* sebesar 62%. Hal ini disebabkan karena penambahan kekuatan sentimen sebagai bobot peringkasan (Wahid & SN, 2016).

2.2 Dasar Teori

Dalam perancangan *CV Matcher*, pastinya memiliki metode-metode atau teori-teori dasar yang terdapat dan digunakan dalam aplikasi keuangan tersebut antara lain:

2.2.1 Kecerdasan Buatan

Mengutip dari kumpulanpengertian.com, menurut Rich dan Knight (1991, p3) kecerdasan buatan adalah ilmu yang mempelajari bagaimana membuat sebuah komputer dapat mengerjakan sesuatu yang masih lebih baik dikerjakan manusia. Kemudian menurut Rolston (1998, p15) Kecerdasan buatan merupakan solusi berbasis komputer terhadap masalah yang ada, yang menggunakan aplikasi yang mirip dengan proses berpikir menurut manusia. Adapun Menurut John McCarthy (1956), kecerdasan buatan adalah suatu sistem komputer yang terbentuk untuk mengetahui dan memodelkan proses-proses berpikir manusia dan mendesain mesin agar dapat menirukan perilaku manusia. Sedangkan menurut setiawan (1993, p1) : Kecerdasan buatan dapat didefinisikan sebagai cabang ilmu komputer yang mempelajari otomasi tingkah laku cerdas.

Sehingga berdasarkan pengertian-pengertian tersebut dapat disimpulkan bahwa kecerdasan buatan adalah sebuah cabang ilmu komputer yang mempelajari tentang penerapan kecerdasan tingkah laku maupun proses-proses berpikir manusia terhadap sebuah komputer atau mesin. Sehingga mesin tersebut dapat mengerjakan pekerjaan maupun berpikir seperti manusia walaupun terkadang masih lebih baik dikerjakan oleh manusia itu sendiri.

Dalam membuat suatu aplikasi kecerdasan buatan, dibutuhkan 2 hal yang sangat penting, yaitu :

- a. Basis Pengetahuan (*Knowledge Base*) : berisi fakta-fakta, data-data, atau informasi yang berkaitan satu sama lain sebagai acuan kecerdasan buatan dalam mengambil keputusan.
- b. Motor Inferensi (*Inference Engine*) : merupakan kemampuan komputer atau mesin dalam mengambil sebuah keputusan terhadap suatu kasus berdasarkan basis pengetahuan yang diberikan.

Ada tiga tujuan kecerdasan buatan, yaitu: membuat komputer lebih cerdas, mengerti tentang kecerdasan, dan membuat mesin lebih berguna. Yang di maksud kecerdasan adalah kemampuan untuk belajar atau mengerti dari pengalaman, memahami pesan yang kontradiktif dan ambigu, menanggapi dengan cepat dan baik atas situasi yang baru, menggunakan penalaran dalam memecahkan masalah serta menyelesaikannya dengan efektif (Winston dan Prender gast, 1994).

Dalam proses penerapannya, Kecerdasan buatan dapat diaplikasikan di hampir semua bidang. Contohnya seperti dibidang pendidikan, kesehatan, pemerintahan, militer, ekonomi, sosial, dan bidang-bidang lainnya. Dalam bidang ekonomi, kita mengenal adanya berbagai aplikasi market place e-commerce. Ketika kita menggunakan atau membuka salah satu aplikasi e-commerce, terkadang aplikasi tersebut merekomendasikan barang-barang yang kebetulan sedang kita inginkan atau perlukan saat itu. Hal tersebut ternyata bukan hanya kebetulan semata. Namun dibalik itu semua ada sebuah sistem kecerdasan buatan yang memantau kita dalam melakukan kegiatan di dunia digital khususnya media sosial. Sistem tersebut melacak dan memantau kita berdasarkan history, percakapan, cookies, maupun kegiatan kita lainnya di media sosial yang kemudian data-data tersebut diolah oleh bantuan kecerdasan buatan sehingga sistem tersebut dapat merekomendasikan barang-barang yang kita butuhkan atau inginkan saat itu.

Contoh lainnya adalah dibidang pertanian. Sekarang sudah cukup banyak aplikasi yang mampu mendeteksi penyakit tanaman hanya melalui sebuah kamera smartphone. Dengan adanya teknologi tersebut, memungkinkan para petani dapat mengetahui kesehatan tanamannya dengan cepat.

2.2.2 *Machine Learning*

Menurut Arthur Samuel *Machine Learning* adalah sebuah pertanyaan “how can computers learn to solve problems without being explicitly programmed?” yaitu bagaimana agar komputer dapat berjalan untuk memecahkan masalah sendiri tanpa harus diprogram secara eksplisit (Samuel, 1988). Sedangkan menurut Mehryar Mohri “*Machine Learning can be broadly defined as computational methods using experience to improve Performance or to make accurate predictions.*” Bersama kawan-kawannya Mohri mendefinisikan pengertian *Machine Learning* sebagai

metode komputasi yang memanfaatkan experience untuk meningkatkan akurasi prediksi (Mehryar Mohri -- *Foundations of Machine Learning - Book*, 2022). Adapun menurut Ian Goodfellow dalam bukunya mendefinisikan sebagai berikut: *Machine Learning is essentially a form of applied statistics with increased emphasis on the use of computers to statistically estimate complicated Functions* (Deep Learning, 2022).

Sehingga berdasarkan penjelasan ketiga para ahli tersebut, dapat disimpulkan bahwa *Machine Learning* adalah suatu bidang ilmu komputer yang mempelajari tentang bagaimana sebuah komputer atau mesin dapat memiliki pola perilaku belajar terhadap sebuah kasus. Pola perilaku belajar tersebut melalui sebuah model algoritma yang dibangun dengan *dataset* sebagai basis pengetahuan untuk proses pembelajaran tersebut. Sehingga komputer atau mesin tersebut dapat berpikir dan memiliki nalar dalam memecahkan sebuah kasus permasalahan.

Melansir dari situs www.dicoding.com, ada beberapa teknik yang dimiliki oleh *Machine Learning*, namun secara luas ML memiliki dua teknik dasar belajar, yaitu *supervised* dan *unsupervised*.

a. *Supervised Learning*

Teknik *supervised learning* merupakan teknik yang bisa kita terapkan pada pembelajaran mesin yang bisa menerima informasi yang sudah ada pada data dengan memberikan label tertentu. Diharapkan teknik ini bisa memberikan target terhadap *output* yang dilakukan dengan membandingkan pengalaman belajar di masa lalu.

Misalkan kita mempunyai sejumlah film yang sudah kita beri label dengan kategori tertentu. Kita juga memiliki film dengan kategori komedi meliputi film 21 Jump Street dan Jumanji. Selain itu kita juga punya kategori lain misalkan kategori film *horror* seperti *The Conjuring* dan *It*. Ketika kita membeli film baru, maka kita akan mengidentifikasi genre dan isi dari film tersebut. setelah film teridentifikasi barulah kita akan menyimpan film tersebut pada kategori yang sesuai.

b. *Unsupervised Learning*

Teknik *unsupervised learning* merupakan teknik yang bisa kita terapkan pada *Machine Learning* yang digunakan pada data yang tidak memiliki informasi yang bisa diterapkan secara langsung.

Sedikit berbeda dengan *supervised learning*, kita tidak memiliki data apapun yang akan dijadikan acuan sebelumnya. Misalkan kita belum pernah sekalipun membeli film sama sekali, akan tetapi pada suatu waktu, kita membeli sejumlah film dan ingin membaginya ke dalam beberapa kategori agar mudah untuk ditemukan.

Tentunya kita akan mengidentifikasi film-film mana saja yang mirip. Dalam hal ini misalkan kita mengidentifikasi berdasarkan dari genre film. Misalnya, kita mempunyai film *The Conjuring*, maka kita akan menyimpan film *The Conjuring* tersebut pada kategori film *horror* (*Apa Itu Machine Learning? Beserta Pengertian Dan Cara Kerjanya - Dicoding Blog*, 2018).

2.2.3 *Natural Processing Language (NLP)*

Natural Processing Language (NLP) adalah cabang ilmu komputer, linguistik, dan kecerdasan buatan yang mengkaji interaksi antara komputer dan bahasa (alami) manusia, khususnya cara memprogram komputer untuk mengolah data bahasa alami dalam jumlah besar. Hasilnya adalah komputer mampu "memahami" isi dokumen, termasuk nuansa bahasa di dalamnya. Dengan ini, komputer dapat dengan akurat mengambil informasi dan wawasan dari dokumen sekaligus mengelompokkan dan menata dokumen-dokumen itu sendiri (*Pengolahan Bahasa Alami - Wikipedia Bahasa Indonesia, Ensiklopedia Bebas*, 2022).

Pustejovsky dan Stubbs (2012) menjelaskan bahwa ada beberapa area utama penelitian pada field NLP, diantaranya:

- a. *Question Answering Systems (QAS)*. Kemampuan komputer untuk menjawab pertanyaan yang diberikan oleh *user*. Daripada memasukkan *keyword* ke dalam *browser* pencarian, dengan QAS, *user* bisa langsung bertanya dalam bahasa natural yang digunakannya, baik itu Inggris, Mandarin, ataupun Indonesia.

- b. *Summarization*. Pembuatan ringkasan dari sekumpulan konten dokumen atau email. Dengan menggunakan aplikasi ini, *user* bisa dibantu untuk mengkonversikan dokumen teks yang besar ke dalam bentuk slide presentasi.
- c. *Machine Translation*. Produk yang dihasilkan adalah aplikasi yang dapat memahami bahasa manusia dan menterjemahkannya ke dalam bahasa lain. Termasuk di dalamnya adalah *Google Translate* yang apabila dicermati semakin membaik dalam penterjemahan bahasa. Contoh lain lagi adalah *BabelFish* yang menterjemahkan bahasa pada *real time*.
- d. *Speech Recognition*. Field ini merupakan cabang ilmu NLP yang cukup sulit. Proses pembangunan model untuk digunakan telpon/komputer dalam mengenali bahasa yang diucapkan sudah banyak dikerjakan. Bahasa yang sering digunakan adalah berupa pertanyaan dan perintah.
- e. *Document Classification*. Sedangkan aplikasi ini adalah merupakan area penelitian NLP Yang paling sukses. Pekerjaan yang dilakukan aplikasi ini adalah menentukan dimana tempat terbaik dokumen yang baru diinputkan ke dalam sistem. Hal ini sangat berguna pada aplikasi spam filtering, *News article Classification*, dan *movie review*.

Perkembangan NLP menghasilkan kemungkinan dari *interface* bahasa natural menjadi *Knowledge base* dan penterjemahan bahasa natural. Poole dan Mackworth (2010) menjelaskan bahwa ada 3 (tiga) aspek utama pada teori pemahaman mengenai *Natural language*:

- a. *Syntax*: menjelaskan bentuk dari bahasa. *Syntax* biasa dispesifikasikan oleh sebuah *grammar*. *Natural language* jauh lebih daripada formal *language* yang digunakan untuk logika kecerdasan buatan dan program komputer
- b. *Semantics*: menjelaskan arti dari kalimat dalam satu bahasa. Meskipun teori *semantics* secara umum sudah ada, ketika membangun sistem *Natural language understanding* untuk aplikasi tertentu, akan digunakan representasi yang paling sederhana.
- c. *Pragmatics*: menjelaskan bagaimana pernyataan yang ada berhubungan dengan dunia. Untuk memahami bahasa, agen harus mempertimbangan lebih dari hanya sekedar kalimat. Agen harus melihat lebih ke dalam

konteks kalimat, keadaan dunia, tujuan dari *speaker* dan *listener*, konvensi khusus, dan sejenisnya.

Information Retrieval (IR) adalah pekerjaan untuk menemukan dokumen yang relevan dengan kebutuhan informasi yang dibutuhkan oleh *user*. Contoh sistem IR yang paling populer adalah search engine pada *World Wide Web*. Seorang pengguna Web bisa menginputkan *query* berupa kata apapun ke dalam sebuah search engine dan melihat hasil dari pencarian yang relevan. Karakteristik dari sebuah sistem IR (Russel & Norvig, 2010) diantaranya adalah:

- a. *A corpus of Documents*. setiap sistem harus memutuskan dokumen yang akan diperlakukan sebagai apa. Bisa sebagai sebuah paragraf, halaman, atau teks multipage.
- b. *Queries posed in a query language*. Sebuah *query* menjelaskan tentang apa yang *user* ingin peroleh. *Query language* dapat berupa list dari kata-kata, atau bisa juga menspesifikasikan sebuah frase dari kata-kata yang harus berdekatan
- c. *A result set*. Ini adalah bagian dari dokumen yang dinilai oleh sistem IR sebagai yang relevan dengan *query*.
- d. *A presentation of the result set*. Maksud dari bagian ini adalah tampilan list judul dokumen yang sudah di *ranking*.

Kemudian ada yang disebut dengan *Stemming & Lemmatization*. *Stemming* merupakan sebuah proses yang bertujuan untuk mereduksi jumlah variasi dalam representasi dari sebuah kata (Kowalski, 2011). Resiko dari proses *Stemming* adalah hilangnya informasi dari kata yang di-stem. Hal ini menghasilkan menurunnya akurasi atau presisi. Sedangkan untuk keuntungannya adalah, proses *Stemming* bisa meningkatkan kemampuan untuk melakukan *Recall*. Tujuan dari *Stemming* sebenarnya adalah untuk meningkatkan performace dan mengurangi penggunaan *resource* dari sistem dengan mengurangi jumlah *unique word* yang harus diakomodasikan oleh sistem. Jadi, secara umum, algoritma *Stemming* mengerjakan transformasi dari sebuah kata menjadi sebuah standar representasi morfologi (yang dikenal sebagai stem).

Ingason dkk. (2008) mengemukakan bahwa *Lemmatization* adalah sebuah proses untuk menemukan bentuk dasar dari sebuah kata. Nirenburg (2009)

mendukung teori ini dengan kalimatnya yang menjelaskan bahwa *Lemmatization* adalah proses yang bertujuan untuk melakukan *Normalisasi* pada teks/kata dengan berdasarkan pada bentuk dasar yang merupakan bentuk lemma-nya. Lemma adalah bentuk dasar dari sebuah kata yang memiliki arti tertentu berdasar pada katas (*Natural language Processing*, 2022).

2.2.4 Support Vector Machine (SVM)

Support Vector Machine (SVM) adalah sistem pembelajaran yang menggunakan ruang hipotesis berupa fungsi-fungsi linier. Fungsi-fungsi itu ada dalam sebuah ruang fitur (*feature space*) berdimensi tinggi, dilatih dengan algoritma pembelajaran yang didasarkan pada teori optimasi dengan mengimplementasikan *learning* bias yang berasal dari teori pembelajaran statistik (Cristianini & Shawe-Taylor, 2000).

Teori yang mendasari SVM sendiri sudah berkembang sejak 1960-an, tetapi baru diperkenalkan oleh Vapnik, Boser dan Guyon pada tahun 1992 dan sejak itu SVM berkembang dengan pesat. SVM adalah salah satu teknik yang relatif baru dibandingkan dengan teknik lain, tetapi memiliki performansi yang lebih baik di berbagai bidang aplikasi seperti bioinformatics, pengenalan tulisan tangan, klasifikasi teks dan lain sebagainya (Chang & Lin, 2011).

SVM bertujuan untuk melakukan proses klasifikasi terhadap sebuah data-data yang diberikan. Proses klasifikasi tersebut adalah sebuah hipotesis berupa bidang pemisah terbaik yang juga memiliki generalisasi yang baik. Generalisasi adalah sebuah kemampuan berupa hipotesis yang berguna untuk mengklasifikasikan data yang tidak ada dalam data pelatihan dengan benar. Dalam menjamin proses generalisasi ini, SVM bekerja menggunakan prinsip SRM.

SRM (Structural Risk Minimization) adalah prinsip menemukan subset dari ruang hipotesis yang dipilih sehingga batas atas *actual risk* dengan menggunakan subset tersebut diminimumkan. SRM bertujuan untuk meminimumkan *actual risk* dengan cara meminimumkan kesalahan pada data pelatihan dan juga VC confidence. Namun, implementasi SRM tidak dilakukan dengan meminimumkan persamaan (SVM-001) karena dimensi VC dari ruang hipotesis $f(\alpha)$ sulit untuk

dihitung. Dan juga hal tersebut hanya terdapat sedikit model hipotesis yang diketahui bagaimana cara menghitung dimensi VC-nya (Ladwani, 2018).

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\left(\frac{h(\log \frac{2l}{h} + 1 - \log(\frac{\eta}{4}))}{l} \right)}$$

Gambar 2.2 Persamaan matematis SRM

SRM bertujuan untuk menjamin batas atas dari generalisasi pada data pengujian dengan cara mengontrol "kapasitas" (fleksibilitas) dari hipotesis hasil pembelajaran. Untuk mengukur kapasitas ini digunakan dimensi Vapnik-Chervonenkis (VC) yang merupakan properti dari ruang hipotesis $f(\alpha)$. Nilai dari dimensi VC ini, berdasarkan teori pembelajaran statistik akan menentukan besarnya nilai kesalahan hipotesis pada data pengujian. Lebih jelasnya, besar kesalahan pada data pengujian/*actual risk* $R(\alpha)$ dengan probabilitas sebesar $1-\eta$, $0 \leq \eta \leq 1$, pada *dataset* yang terdiri dari n data dapat dilihat pada persamaan (SVM-001). $R_{emp}(\alpha)$ adalah kesalahan pada data pelatihan dan h adalah dimensi VC. Nilai VC confidence (nilai elemen kedua pada ruas kanan (SVM-001)), ditentukan oleh hipotesis/ fungsi hasil pembelajaran (Burges & Burges, 1998).

2.2.5 Cosine Similarity

Berdasarkan artikel dari hendroprasetyo.com, *Cosine Similarity* adalah “ukuran kesamaan”, salah satu implementasinya adalah pada kasus mencari tingkat kemiripan teks pada teks itu sendiri atau *sentence*/kalimat . Kemiripan teks bisa kita gunakan untuk membuat steganografi ataupun steganalysis linguistik.

Rumus *Cosine Similarity* :

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}},$$

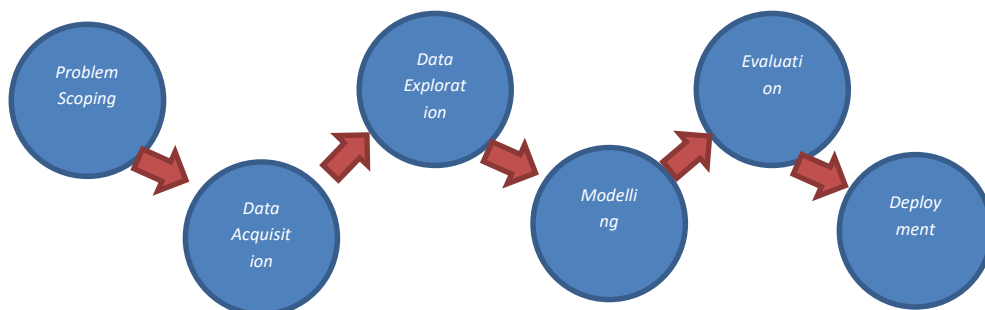
Gambar 2.3 Persamaan matematis *Cosine Similarity*

Penggunaan *Cosine Similarity* ini meliputi pada pencarian data, mengukur kemiripan 2 buah kalimat, dll (*Perbedaan Cosine Similarity Dan Cosine Distance - Hendro Prasetyo, 2022*).

2.2.5 *AI Project Cycle*

Pada proyek ini akan menggunakan metode *AI Project Cycle*, yaitu merupakan sebuah kerangka kerja yang dapat digunakan untuk melakukan development sebuah proyek *Artificial Intelligence* (Handbook, n.d.). Proses dari *AI Project Cycle* tersebut terdiri atas 6 tahap, yaitu sebagai berikut :

- a. *Problem Scoping*
- b. *Data Acquisition*
- c. *Data Exploration*
- d. *Modelling*
- e. *Evaluation*
- f. *Deployment*



Gambar 2.4 *AI Project Cycle*

Berikut adalah penjelasan dari masing-masing proses tersebut, dimulai dari proses *Problem Scoping* hingga *Deployment*.

a. *Problem Scoping*

Problem Scoping merupakan sebuah proses yang dilakukan untuk melakukan analisis terhadap suatu permasalahan yang akan kita coba pecahkan menggunakan sebuah solusi dengan *Artificial Intelligence* (AI). Dalam proses analisis permasalahan tersebut, kita dapat menggunakan sebuah teknik *4W's* yang merupakan kepanjangan dari *Who*, *What*, *Where*, dan *Why*.

1) *Who*

Merujuk pada siapa yang menghadapi masalah dan siapa pemangku kepentingan dari masalah tersebut. Kalimat “Siapa pemangku kepentingan” disini bisa merujuk pada individu atau kelompok yang memiliki kewenangan dalam keterlibatan masalah tersebut. Sedangkan kalimat “Siapa yang menghadapi masalah” dapat merujuk pada individu atau kelompok yang mendapatkan masalah dan menanggung masalah tersebut.

2) *What*

Merujuk pada apa masalahnya dan bagaimana anda tahu tentang masalahnya. Permasalahan yang ada harus dijelaskan secara singkat dan langsung pada intinya.

3) *Where*

Hal ini mengacu pada dimana lokasi permasalahan tersebut, maupun mengacu pada terkait konteks atau situasi permasalahan itu terjadi. Sehingga kita dapat mengetahui lebih jauh situasi dan kondisi masalah yang kita hadapi tersebut.

4) *Why*

Mengacu pada mengapa kita perlu memecahkan masalah dan apa manfaat bagi para pemangku kepentingan setelah menyelesaikan masalah. Hal ini penting untuk kita analisis supaya kita dapat mengetahui seberapa berat masalah yang kita hadapi, dan seberapa

besar dampak yang akan didapat jika kita berhasil memecahkan masalah tersebut.

Dalam melakukan tahap *Problem Scoping* ini, keempat hal tersebut sangat berpengaruh terhadap pemahaman kita kepada masalah tersebut. Dengan melakukan analisis masalah menggunakan *Problem Scoping* ini, Kita dapat mengetahui pokok dari permasalahan yang kita hadapi, seberapa penting masalah ini untuk diselesaikan, seberapa sulit masalah ini dipecahkan, dan seberapa besar dampak kebermanfaatan yang akan terjadi jika kita berhasil menyelesaikan masalah ini.

b. *Data Acquisition*

Data Acquisition merupakan sebuah tahapan dimana kita mendapatkan data-data atau *dataset* yang diperlukan untuk diolah sebagai bahan model untuk melakukan prediksi. Bagian tahapan ini sangat penting dilakukan karena *dataset* yang digunakan akan sangat berpengaruh terhadap hasil prediksi model. Semakin bagus *dataset* yang kita miliki, maka akan sangat membantu sekali ketika kita melakukan tahapan *Data Exploration*. Tahapan tersebut berguna untuk mengetahui pola dan insight yang ada pada *dataset* tersebut. Ketika kita sudah mengetahui pola pada *dataset* tersebut, kita dapat mengetahui dan memilih model apa yang tepat untuk digunakan dalam melakukan prediksi.

Pada tahapan *Data Acquisition* ini, ada beberapa sumber yang dapat digunakan sehingga kita bisa mendapatkan sebuah *dataset*, yaitu sebagai berikut:

1) Survey

Survey merupakan sebuah sumber yang dapat digunakan untuk mengumpulkan *dataset* yang diperlukan dalam tahap data acquisition. Biasanya survey dilakukan menggunakan sebuah kuisisioner yang diberikan atau ditujukan kepada responden individu atau kelompok yang berkaitan dengan permasalahan. Dalam melakukan pembagian kuisisioner tersebut, biasanya media yang digunakan adalah media digital.

Saat ini sudah banyak sekali aplikasi form kuisioner yang dapat digunakan untuk mendapatkan data-data dari para responden contohnya adalah aplikasi *Google Form*. Media digital digunakan untuk membantu mempermudah dan mempersingkat proses *Data Acquisition* tersebut.

2) *Web Scrapping*

Web Scrapping merupakan sebuah teknik pengambilan data yang biasa digunakan pada situs website artikel, maupun media sosial. Tidak seperti kuisioner, metode *scrapping* ini digunakan biasanya dilakukan oleh profesional yang mengerti dengan pengolahan data. Metode *Web Scrapping* ini bekerja mirip seperti copy paste, dengan cara mengambil data-data tertentu yang ada pada sebuah situs web kemudian mengumpulkannya menjadi satu kesatuan *dataset*.

3) Sensor

Dataset yang diperoleh bisa berasal dari sensor-sensor. Data-data tersebut terkadang cenderung bersifat kurang terstruktur dan terlihat berantakan. Sehingga data-data tersebut perlu dilakukan proses *data cleaning* yang cukup sulit dilakukan. Contoh data-data sensor bisa berupa data cuaca, data suhu ruangan, data tingkat keasaman PH, dan lain-lain.

4) Kamera

Sumber data dari kamera tentunya akan berupa sebuah data image atau gambar. *Dataset* yang berbentuk gambar ini biasanya memiliki ukuran yang cukup besar. Dalam melakukan tahap pemodelan pun, model yang digunakan untuk mengolah *dataset* gambar cukup rumit dan membutuhkan *resource* yang cukup besar untuk melakukan proses komputasinya.

5) Observasi

Observasi hampir mirip dengan survey. Namun biasanya observasi dilakukan dengan cara kita harus turun langsung lokasi permasalahan untuk mendapatkan data-data tersebut. Hal ini dilakukan ketika kita memerlukan sebuah *dataset* yang dimiliki oleh suatu objek

tertentu. Hal ini bisa juga dilakukan ketika *dataset* tersebut harus dilakukan dikumpulkan dengan pengamatan dan proses secara langsung. Umumnya data yang didapat dari hasil observasi adalah sebuah *dataset* baru yang belum pernah ada sebelumnya, sehingga kita harus benar-benar mengumpulkannya sendiri secara langsung.

6) *API (application Programming Interface)*

API adalah singkatan dari *application Programming Interface* atau dapat diartikan sebagai perangkat lunak yang berfungsi menghubungkan suatu aplikasi dengan aplikasi lainnya. Pengaplikasian API pada tahapan *Data Acquisition* adalah biasanya menggunakan teknologi JSON.

c. *Data Exploration*

Data Exploration merupakan tahapan *AI Project Cycle* yang berguna untuk mengetahui pola dan insight dalam sebuah *dataset*. Pada tahap ini juga digunakan untuk mengetahui seberapa bagus data yang kita miliki sehingga nantinya *dataset* tersebut bisa digunakan oleh model untuk *Training* dengan baik. Ada beberapa *tools* yang dapat digunakan untuk melakukan proses *Data Exploration* tersebut diantaranya:

- 1) *Google Charts*
- 2) *Tableau*
- 3) *Fusion Charts*
- 4) *High Charts*

Namun pada faktanya, tidak selamanya data yang peroleh itu bersih dan langsung dapat digunakan. Maka dari itu pada tahap ini juga perlu yang dinamakannya sebuah proses *data cleaning*. *Data cleaning* merupakan sebuah proses pengecekan *dataset*, apakah *dataset* sudah sesuai untuk digunakan pada tahap *modelling* atau tidak. Masalah-masalah yang biasanya ditemukan dan perlu diselesaikan oleh *data cleaning* adalah sebagai berikut:

1) *Imbalance Data*

Permasalahan ini muncul biasanya jika kasus nya adalah klasifikasi. Dalam klasifikasi terdapat beberapa label yang mengindikasikan *output* klasifikasi tersebut. Terkadang jumlah data

pada setiap label itu berbeda-beda dan tidak sama. Jika perbedaan jumlah data setiap label itu cukup besar maka hal tersebut dapat menyebabkan masalah *Imbalance Data*. Misalkan kita memiliki sebuah *dataset* klasifikasi air yang terdiri atas label air bersih dan air kotor. Data yang berlabel air bersih adalah sebanyak 3000 *record*. Sedangkan data yang berlabel air kotor adalah sebanyak 1500 *record*. Bisa kita simpulkan dari kasus tersebut bahwa *dataset* tersebut memiliki permasalahan *Imbalance Data* dimana terjadi ketidakseimbangan data antar setiap labelnya.

Untuk mengatasi hal tersebut, kita bisa menggunakan teknik SMOTE. Teknik ini bekerja dengan melakukan manipulasi data sehingga data yang lebih sedikit bisa menggandakan diri untuk menyamai jumlah data yang lebih banyak. Sehingga setiap label memiliki jumlah data yang sama.

2) *Outlier Point*

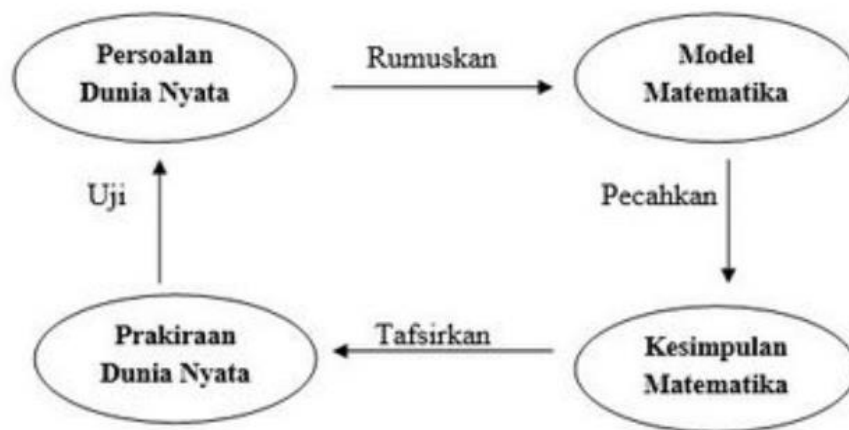
Permasalahan ini muncul ketika ada beberapa data yang terletak diluar penyebaran *dataset*. Hal ini terjadi ketika ada beberapa nilai data yang berada di luar range nilai kebanyakan data. Permasalahan ini tentu sangat berpengaruh sekali terhadap hasil proses *Training model* karena dapat menimbulkan bias pada model. Contohnya adalah ketika kita punya data usia penduduk, range usia antara 0 – 60 tahun adalah sebanyak 250 data. Namun ada seorang penduduk yang berusia 90 tahun. Dari contoh tersebut bisa kita lihat bahwa data penduduk yang berusia 90 tahun itu adalah sebuah *Outlier Point*. Hal itu tentu dapat mengganggu pada saat proses *Training* nanti, terutama pada hasil akurasi model setelah proses *Training* dilakukan.

Bisa kita lihat bahwa akan lebih mudah menemukan sebuah *Outlier Point* jika kita merepresentasikan *dataset* nya kedalam sebuah grafik. Karena nantinya akan terlihat titik penyebaran datanya seperti apa. Dengan representasi seperti itu juga, kita dapat mengetahui tren dari penyebaran *dataset*nya.

Untuk mengatasi permasalahan tersebut, dapat dilakukan dengan cara menghapus data *outlier* tersebut, baik secara manual maupun dengan *tools* atau code secara otomatis. Dengan menghapus *outlier* tersebut, dapat mencegah bias atau *Error* pada model. Sehingga hasil dari proses *Training* pada model menjadi lebih baik dan optimal.

d. *Modelling*

Model dalam istilah teknologi adalah representasi suatu masalah dalam bentuk yang lebih sederhana sehingga lebih jelas dan mudah dikerjakan. Model yang baik cukup mengandung bagian-bagian yang perlu saja. Dalam *AI Project Cycle*, *Modelling* adalah sebuah proses pelatihan model menggunakan *dataset* yang telah kita siapkan sebelumnya. Sehingga model tersebut mampu memprediksi suatu *output* berdasarkan *input* yang kita berikan kepada model tersebut.



Gambar 2.5 Ilustrasi pemodelan.

Pada gambar tersebut, kita bisa melihat sebuah representasi dari sebuah pemodelan. Dimulai dari mengidentifikasi persoalan dunia nyata, kemudian persoalan tersebut di rumuskan kedalam model matematika sehingga dapat diperhitungkan. Selanjutnya setelah didapat model

matematikanya kita dapat langsung menghitung dan memecahkan persoalan tersebut, dan mendapatkan sebuah hasil kesimpulan matematika. Kemudian kesimpulan matematika tersebut dapat kita tafsirkan sebagai prediksi keputusan dalam dunia nyata.

Pada tahap ini kita akan melakukan beberapa proses sebagai berikut:

1) Menentukan model

Pada proses ini adalah proses menentukan model *Machine Learning* yang akan kita gunakan untuk melakukan *Training* atau pelatihan menggunakan *dataset* yang kita miliki. Dalam menentukan model yang akan kita gunakan, kita bisa melihat dari jenis tahap *Problem Scoping* hingga *Data Exploration* yang telah kita lakukan. Kita harus mampu menentukan model yang sesuai dengan jenis *dataset* kita. Apakah jenis datanya berupa data *Text*, gambar, ataupun suara. Kemudian *output* yang kita harapkan apakah berupa jenis klasifikasi, *Clustering*, atau *forecasting*.

2) Menyiapkan data

Pada proses ini, kita harus menyiapkan *dataset* yang akan kita gunakan untuk melakukan *Training* pada model yang telah kita tentukan. Persiapan tersebut dilakukan dengan membagi *dataset* yang kita miliki ini kedalam beberapa bagian, yaitu *Training Data*, *Validation Data*, dan *Testing Data*.

Training Data digunakan untuk proses *trainig* pada model. Biasanya untuk jumlah *Training Data* adalah 70% dari total keseluruhan jumlah data. Kemudian untuk *Validation Data* digunakan untuk melakukan proses evaluasi terhadap model yang telah di *Training*. Sehingga dengan proses evaluasi tersebut, kita dapat mengetahui performa dari model yang kita latih.

Untuk jumlah *Validation Data* yang digunakan adalah sebesar 20% dari total jumlah keseluruhan data. Sedangkan *Testing Data* ini digunakan untuk melakukan proses tes terhadap model yang telah dievaluasi. Untuk jumlah *Testing Data* yang digunakan adalah sebesar 10% dari total jumlah keseluruhan data.

Secara umum berdasarkan pola pembelajarannya terhadap *dataset*, model dibagi dalam 3 kelompok yaitu adalah sebagai berikut:

1) *Supervised Learning*

Supervised Learning umumnya digunakan untuk menemukan pola dalam data masukan yang diberi label sehingga memungkinkan kita menghasilkan data keluaran yang benar secara efektif. Ciri-ciri dari model *supervised learning* ini adalah sebagai berikut:

- *Training Data* telah diberi label
- Algoritma memprediksi *output* dari *input*, contoh algoritma:
 - ✓ Klasifikasi (memetakan masukan ke label keluaran)
 - ✓ Regresi (memetakan masukan ke keluaran berkelanjutan)

2) *Unsupervised Learning*

Unsupervised Learning umumnya digunakan untuk mempelajari struktur karakteristik data kita tanpa menggunakan label yang disediakan secara eksplisit. Pada kasus tertentu, permasalahan yang sering terjadi pada *dataset* untuk model *unsupervised* ini adalah *Outlier Point*. Hal ini perlu diperhatikan mengingat data yang kita dapatkan tidak selamanya dapat langsung digunakan dan diterapkan pada model. Ciri-ciri dari model *unsupervised learning* adalah sebagai berikut:

- *Training Data* tidak berlabel
- Algoritma mempelajari struktur karakteristik dari data masukan, contoh algoritma:
 - *Clustering* (mempelajari hubungan antara fitur individu)
 - *Dimensional Reduction* (metode untuk mengurangi fitur)

3) *Reinforcement Learning*

Reinforcement learning merupakan jenis model yang bekerja dengan cara melihat pola penghargaan dan hukuman pada suatu proses. Jadi jika model memprediksi data dengan benar, maka model tersebut akan mendapatkan suatu point positif dan begitupun sebaliknya.

Dari penjelasan di atas, kita bisa menyimpulkan bahwa proses menentukan model sangat penting dilakukan. Karena setiap *dataset*

memiliki karakteristik dan jenis nya tersendiri terhadap suatu model. Adapun perbedaan diantara ketiga kelompok model tersebut yaitu:

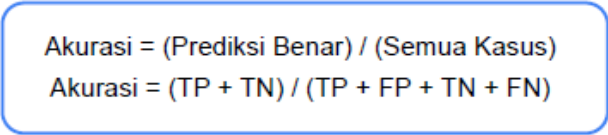
- Model *supervised learning* untuk *dataset* yang memiliki label.
- Model *unsupervised learning* untuk *dataset* yang tidak memiliki label.
- Model *Reinforcement Learning* bekerja dengan pola penghargaan dan hukuman.

e. *Evaluation*

Pada tahapan *Evaluation* ini, kita akan mengukur seberapa baik dan optimal model yang kita telah *Training* dengan *dataset* yang kita punya sebelumnya. Pada proyek skripsi ini, kita akan menggunakan teknik *Evaluation confusion matrix* yang mana memiliki 3 paramater acuan yang sering digunakan, yaitu:

1) *Accuration*

Accuracy atau akurasi merupakan peresentase prediksi yang benar dari semua pengamatan. Terkadang akurasi merupakan sebuah evaluasi utama yang digunakan untuk mengukur performa dari suatu model yang dilatih. Berikut adalah persamaan matematika untuk mengukur *Accuration*.



$$\text{Akurasi} = (\text{Prediksi Benar}) / (\text{Semua Kasus})$$

$$\text{Akurasi} = (TP + TN) / (TP + FP + TN + FN)$$

Gambar 2.6 Persamaan *Accuration*.

2) *Precision*

Precision atau presisi merupakan sebuah persentase kasus yang diprediksi positif (PB + PP) yang ternyata positif (PB). Persamaan matematis presisi ini dapat dilihat pada gambar berikut.

$$\text{Presisi} = (\text{True Positive}) / (\text{Total Prediksi Positif})$$

$$\text{Presisi} = (TP) / (TP + FP)$$

Gambar 2.7 Persamaan *Precision*.

3) *Recall*

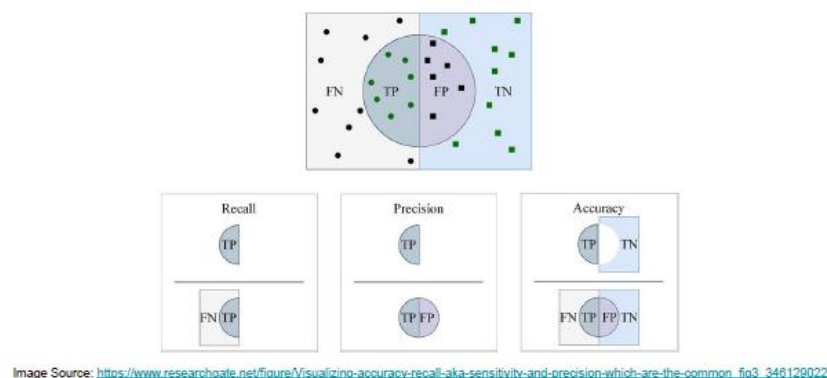
Recall merupakan teknik untuk mengukur pecahan kasus positif (TP + FN) yang diidentifikasi dengan benar (TP). Persamaannya adalah sebagai berikut.

$$\text{Recall} = (\text{True Positive}) / (\text{Total aktual positif})$$

$$\text{Recall} = (TP) / (TP + FN)$$

Gambar 2.8 Persamaan *Recall*.

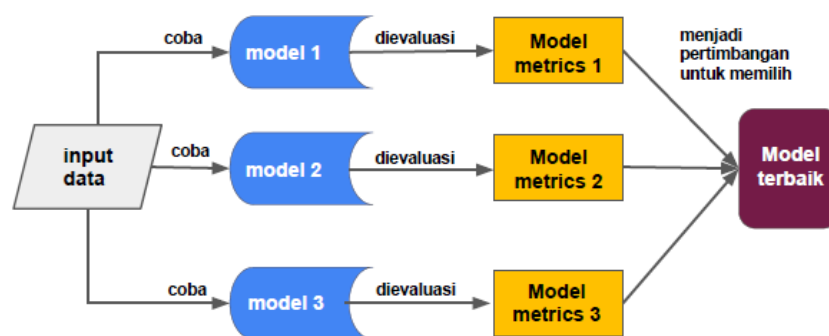
Untuk lebih memahami terhadap *confusion matrix* ini, kita lihat representasi dari akurasi, presisi, dan *Recall* adalah sebagai berikut:



Gambar 2.9 Representasi Akurasi, Presisi, dan *Recall*.

Tahapan evaluasi ini bertujuan untuk menentukan model terbaik mana yang akan kita gunakan. Karena dengan hasil evaluasi ini, kita dapat mengetahui performa dari setiap model yang akan kita evaluasi. Jadi untuk melakukan evaluasi ini biasanya akan menggunakan beberapa model untuk diseleksi.

Untuk lebih jelas dalam memahami pemilihan model pada tahap evaluasi ini, mari kita lihat bagan penentuan model sebagai berikut:



Gambar 2.10 Model Selection.








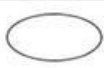


f. *Deployment*

Deployment merupakan tahap terakhir pada rangkaian tahapan *AI Project Cycle*. Pada tahap ini, kita akan menerapkan model *Machine Learning* kita terhadap aplikasi agar model *Machine Learning* kita dapat digunakan oleh *user* dengan mudah.







2.2.6 (*Unified Model Language*) UML

Unified *Modelling* Language (UML) digunakan agar memudahkan pengembang sistem untuk menghasilkan sebuah sistem informasi yang memiliki paradigma berbasis objek (Waruwu & Nasution, 2018). UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan system (Hardiyanto et al., 2019).

a. *Use case diagram*

SIMBOL	NAMA	KETERANGAN
	<i>Actor</i>	Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan <i>use case</i> .
	<i>Dependency</i>	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (<i>independent</i>) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri.
	<i>Generalization</i>	Hubungan dimana objek anak (<i>descendent</i>) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (<i>ancestor</i>).
	<i>Include</i>	Menspesifikasikan bahwa <i>use case</i> sumber secara eksplisit.
	<i>Extend</i>	Menspesifikasikan bahwa <i>use case</i> target memperluas perilaku dari <i>use case</i> sumber pada suatu titik yang diberikan.
	<i>Association</i>	Apa yang menghubungkan antara objek satu dengan objek lainnya.
	<i>System</i>	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
	<i>Use Case</i>	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.
	<i>Collaboration</i>	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen-elemennya (<i>sinergi</i>).
	<i>Note</i>	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.

Gambar 2.11 Keterangan *Use case diagram*b. *Activity diagram*

NO	GAMBAR	NAMA	KETERANGAN
1		<i>Activity</i>	Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain
2		<i>Action</i>	State dari sistem yang mencerminkan eksekusi dari suatu aksi
3		<i>Initial Node</i>	Bagaimana objek dibentuk atau diawali.
4		<i>Activity Final Node</i>	Bagaimana objek dibentuk dan diakhiri
5		<i>Decision</i>	Digunakan untuk menggambarkan suatu keputusan / tindakan yang harus diambil pada kondisi tertentu
6		<i>Line Connector</i>	Digunakan untuk menghubungkan satu simbol dengan simbol lainnya

Gambar 2.12 Keterangan *Activity diagram*

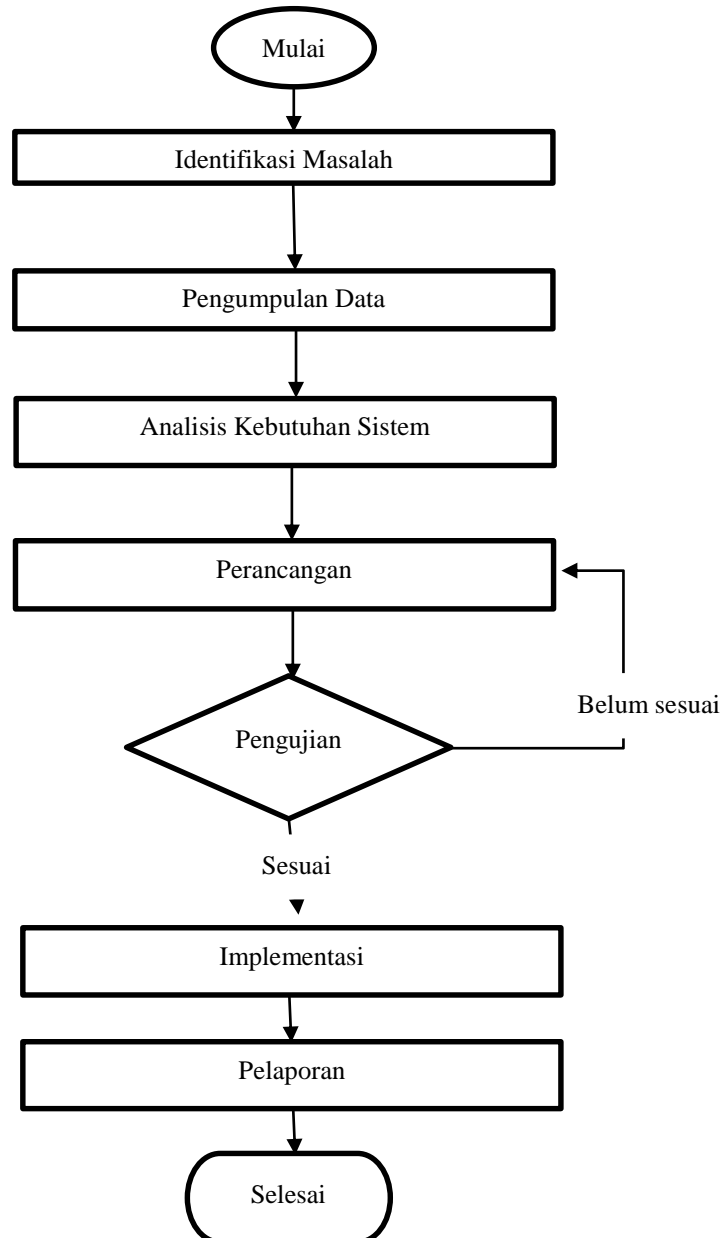
2.2.7 Balsamiq Mockup 3

Balsamiq merupakan aplikasi yang digunakan untuk menggambar rancangan *user interface* sebuah aplikasi yang biasa dan umum digunakan oleh para perancang aplikasi. Aplikasi ini merupakan aplikasi berbayar, namun dianggap sebagai aplikasi yang mudah digunakan dan tidak membutuhkan kode untuk bisa mengoperasikannya. Pengguna hanya cukup melakukan *drag and drop* terhadap icon-icon yang akan digunakan ke dalam sebuah lembar kerja. Selain itu, *Balsamiq Mockup* juga memiliki fitur yang digunakan untuk membuat rancangan tampilan aplikasi yang interaktif.

BAB III

METODOLOGI PENELITIAN

3.1 Kerangka Pikir



Gambar 3.1 Kerangka Pikir

3.2 Deskripsi

3.2.1 Identifikasi Masalah

Pada tahap awal penelitian ini sesuai dengan kerangka pikir yaitu penulis mencoba untuk mencari permasalahan atau menentukan rumusan masalah. Dalam tahap ini, penulis melakukan penelusuran terhadap website *livecareer.com* untuk mengetahui tentang permasalahan atau kurangnya fitur yang ada di website *livecareer.com* sehingga penulis bisa membuat aplikasi *CV Matcher* ini.

3.2.2 Pengumpulan Data

a. Studi Literatur

Tahapan studi literatur ini dilakukan dengan mengumpulkan serta membaca jurnal dan buku-buku yang relevan dan berkaitan dengan topik *Machine Learning* khususnya pada domain *Natural Language Processing* (NLP). Dalam hal ini, penulis memakai 6 jurnal yang terdiri dari:

- 1) *Multilabel Text Classification* Menggunakan SVM dan *Doc2Vec Classification* pada Dokumen Berita Bahasa Indonesia.
- 2) Penerapan metode klasifikasi *Support Vector Machine* (SVM) pada data akreditasi sekolah dasar (SD) di Kabupaten Magelang.
- 3) Penerapan algoritma *Support Vector Machine*(Svm) dengan Tf-Idf N-Gram untuk *Text Classification*.
- 4) Perbandingan Algoritma *Random Forest Classifier*, *Support Vector Machine* dan *Logistic Regression Classifier* Pada Masalah *High Dimension* (Studi Kasus: Klasifikasi *Fake News*).
- 5) *Text Summarization using Clustering Technique*.
- 6) Peringkasan Sentimen Esktraktif di Twitter Menggunakan *Hybrid TF-IDF* dan *Cosine Similarity*.

3.2.3 Analisis Kebutuhan Sistem

Tahap selanjutnya adalah dalam penelitian ini yaitu menganalisis kebutuhan sistem yang akan dibuat meliputi analisis kebutuhan perangkat lunak, analisis

kebutuhan perangkat keras, hingga analisis biaya. Sehingga tahap ini menjadi tolak ukur dalam melakukan perancangan aplikasi *CV Matcher*.

3.2.4 Perancangan

Pada tahap perancangan ini, penulis menggunakan model *AI Project Cycle* yang terdiri dari 6 tahap, yaitu:

a. *Problem Scoping*

Pada tahap ini penulis mencoba untuk memetakan permasalahan yang akan dipecahkan. Penulis menggunakan metode *4W's* yaitu *Who*, *What*, *Where*, dan *Why*. Tahap ini berguna untuk lebih memahami kembali permasalahan dan acuan bentuk aplikasi *CV Matcher* yang akan dibuat.

b. *Data Acquisition*

Pada tahap ini, dilakukan pemilihan dan pencarian data yang akan digunakan dalam membuat aplikasi *CV Matcher*. Penulis mengambil sebuah *dataset* yang ditemukan pada forum Kaggle yang merupakan hasil metode *Web Scrapping* dari sumber website *livecareer.com* yang dilakukan oleh Snehaan Bhawal. *Dataset* tersebut berupa data daftar riwayat hidup pelamar kerja yang nanti akan digunakan untuk melakukan pelatihan klasifikasi jenis daftar riwayat hidup terhadap model *Machine Learning*.

c. *Data Exploration*

Dalam melakukan tahap ini, penulis mencoba untuk mendapatkan informasi lebih lagi terhadap *dataset* yang akan digunakan yaitu dengan pendekatan statistik dalam bentuk grafik. Hal ini digunakan untuk memastikan *dataset* ini siap untuk digunakan.

d. *Modelling*

Pada tahap ini, penulis akan melakukan pelatihan terhadap model *Machine Learning* yang akan digunakan untuk melakukan klasifikasi jenis daftar riwayat hidup pelamar kerja. Model *Machine Learning* yang digunakan yaitu *Logistic Regression*, *Decision tree Classifier*, *Random Forest Classifier*, *Multinomial Naïve Bayes*, *KNN Classifier*, dan *SVM*. Kemudian selanjutnya yaitu membandingkan data daftar riwayat hidup

pelamar kerja dengan data deskripsi lowongan pekerjaan menggunakan metode *Cosine Similarity*.

e. *Evaluation*

Pada tahap ini, penulis melakukan evaluasi terhadap model *Machine Learning* yang telah dilatih menggunakan pendekatan statistik *Confusion Matrix* untuk mengetahui *Accuracy*, *Recall*, *Precision*, dan *F1-Score*. Sehingga dapat dilakukan pemilihan model terbaik yang akan digunakan untuk diterapkan pada aplikasi *CV Matcher*.

f. *Deployment*

1) Perancangan Diagram

Pada tahap perancangan ini adalah membuat desain menggunakan bahasa pemodelan *Unified Model Language* (UML) yang pada penelitian ini terdiri dari:

- *Use case diagram*, digunakan untuk mengetahui gambaran umum keterhubungan Aktor dan *use case* serta fungsi apa saja yang ada di dalam sistem aplikasi. Aktor dan *use case* yang terlibat pada penelitian ini adalah Pengguna aplikasi sebagai aktor serta 5 *use case* berupa beranda, pengembang, informasi, pengecekan, dan hasil pengecekan.
- *Activity diagram* digunakan untuk memodelkan aktivitas yang ada dalam suatu sistem meliputi gambaran keseluruhan aktivitas yang dilakukan oleh pengguna dengan aplikasi.

2) Perancangan Antarmuka

Tahapan perancangan antarmuka ini adalah membuat desain antarmuka aplikasi yang dibuat sederhana menggunakan *Balsamiq Mockup*. Walaupun sederhana, akan tetapi dapat cukup mengilustrasikan aplikasi yang akan dibangun dan tentunya memerhatikan interaksi yang akan terjadi dalam aplikasi antara *user* dan sistem.

3.2.5 Pengujian

Pada tahapan pengujian ini dilakukan sebuah pengujian aplikasi yang telah dibuat untuk memastikan apakah hasil aplikasi sudah sesuai dengan rancangan yang diharapkan atau belum. Apabila masih ada kekurangan maka kembali ke tahap perancangan untuk diperbaiki sampai benar – benar sesuai dengan rancangan.

3.2.6 Implementasi

Tahap selanjutnya adalah melakukan implementasi perancangan yang telah dibuat dengan cara menerjemahkan kedalam bentuk pengkodean secara nyata memanfaatkan bahasa pemrograman *Python*. Kemudian untuk tampilannya menggunakan bahasa markup HTML, serta *Framework* CSS yaitu *Bootstrap*.

3.2.7 Pelaporan

Tahapan selanjutnya adalah tahap pelaporan yaitu merupakan pembuatan laporan skripsi sebagai salah satu syarat kelulusan yang disusun sesuai dengan ketentuan yang tercantum dalam Pedoman Penulisan Skripsi Fakultas Teknologi Informasi Universitas Bale Bandung.

BAB IV

ANALISIS DAN PERANCANGAN

4.1 Analisis

4.1.1 Analisis Masalah

Langkah pertama yaitu menganalisis website *lifecareer.com* yang sedang berjalan dengan tujuan untuk mengetahui lebih jelas bagaimana cara kerja website tersebut dan kekurangan fitur-fitur yang ada. Sehingga dapat menjadi dasar dalam melakukan perancangan sistem yang akan dibuat.

Berdasarkan analisis langsung yang dilakukan penulis terhadap website *lifecareer.com* bahwa ditemukan bahwa pada website tersebut belum adanya fitur untuk menentukan klasifikasi jenis berkas lamaran pelamar kerja dan fitur untuk melihat kecocokan antara berkas lamaran kerja dengan deskripsi lowongan pekerjaan yang akan dilamar. Pada website *lifecareer.com* hanya memiliki fitur-fitur diantaranya sebagai berikut:

- a. Artikel mengenai pembuatan berkas lamaran.
- b. Fitur desain berkas lamaran dengan *template* yang sudah tersedia.
- c. Fitur pencarian lowongan pekerjaan.
- d. Fitur pengecekan sistematika berkas lamaran kerja.

4.1.2 Analisis Perangkat

Pada proyek skripsi ini, berikut adalah daftar alat dan bahan yang diperlukan untuk penelitian ini.

Tabel 4.1 Daftar Perangkat Keras

No.	Perangkat Keras (<i>Hardware</i>)	Spesifikasi
1	Laptop	<i>Asus X451MA</i>
2	<i>Processor</i>	<i>Intel dual core N2840</i>

3	<i>Installed Memory (RAM)</i>	2 GB
4	<i>Harddisk</i>	500 GB
5	Grafis	<i>Intel HD Graphic</i>

Tabel 4.2 Daftar Kebutuhan Perangkat Lunak

No.	Perangkat Lunak (<i>Software</i>)	Keterangan
1	<i>Windows 10 Pro 64-bit</i>	Sistem Operasi
2	<i>Python 3.9</i>	Bahasa Pemrograman
3	<i>Google Colaboratory</i>	<i>Tools</i> untuk membangun sebuah model
4	<i>Anaconda</i>	<i>Management Tools</i>
5	<i>Tensorflow</i>	<i>Intel HD Graphic</i>
6	<i>Matplotlib</i>	<i>Library Python</i>
7	<i>Keras</i>	<i>Library Python</i>
8	<i>Numpy</i>	<i>Library Python</i>
9	<i>Scikit-Learn</i>	<i>Library Python</i>
10	<i>Zip File</i>	<i>Library Python</i>
11	<i>Heroku</i>	<i>Cloud Hosting</i>
12	<i>Google Drive</i>	<i>Cloud Storage</i>
13	<i>Flask</i>	<i>Framework</i>
14	<i>Microsoft Word</i>	Dokumentasi penyusunan Tugas Akhir
15	<i>Microsoft Vision</i>	<i>Tools</i> perancangan <i>diagram</i>
16	<i>Bootstrap</i>	<i>Framework</i>
17	<i>Github</i>	<i>Management Version</i>
18	<i>Balsamiq Mockup</i>	Desain Antarmuka

4.1.3 Analisis Pengguna

Penganalisaan pengguna adalah yang berkaitan dengan yang akan menggunakan aplikasi *CV Matcher* ini. Pengguna aplikasi ini adalah masyarakat khususnya yang sedang mencari pekerjaan. Supaya nantinya aplikasi ini dapat membantu pelamar tersebut dalam menganalisa kecocokan *Curriculum Vitae* nya dengan lowongan pekerjaan yang akan dilamar. Aplikasi tersebut dapat memberikan hasil nilai seberapa cocok *Curriculum Vitae* pelamar dengan suatu lowongan pekerjaan.

4.1.4 User Interface

Pada aplikasi *CV Matcher* ini, akan terdapat 5 *User Interface* yang ditampilkan sebagai berikut.

a. *User Interface* Beranda

User Interface ini merupakan tampilan dimana pertama kali saat aplikasi dijalankan. Pada tampilan ini, rencananya akan ditampilkan informasi secara umum mengenai aplikasi *CV Matcher* ini. Sehingga pengguna dapat langsung mengetahui bahwa pengguna sudah berada pada aplikasi *CV Matcher*. Tampilan ini sebagai identitas aplikasi sehingga pengguna dapat langsung mengetahui bahwa aplikasi *CV Matcher* telah siap digunakan.

b. *User Interface* Pengembang Aplikasi

Tampilan ini berguna untuk menyajikan informasi identitas pengembang atau pembuat aplikasi *CV Matcher*. Informasi tersebut terdiri dari nama, jurusan program studi, dan nama kampus asal.

c. *User Interface* Informasi Aplikasi

Pada tampilan ini terdapat informasi detail mengenai aplikasi *CV Matcher*. Informasi tersebut meliputi tujuan aplikasi dibuat, dan informasi lainnya tentang aplikasi *CV Matcher*.

d. *User Interface* Pengecekan *Curriculum Vitae*

Pada tampilan ini, terdapat sebuah form yang digunakan untuk melakukan pengecekan suatu *Curriculum Vitae* pelamar dengan suatu

deskripsi lowongan pekerjaan. Pengguna dapat langsung mengunggah *file Curriculum Vitae* nya dalam bentuk ekstensi *.pdf, dan untuk *file* lowongan pekerjaannya menggunakan *file* dengan ekstensi *.jpg/png/jpeg (*file* gambar).

setelah pengguna mengunggah *file-file* tersebut, pengguna dapat langsung menekan tombol Check untuk melakukan pengecekan kecocokan *Curriculum Vitae* pelamar dengan deskripsi lowongan pekerjaan tersebut. Selanjutnya untuk hasil pengecekan tersebut dapat dilihat pada tampilan hasil pengecekan *Curriculum Vitae*.

e. *User Interface* Hasil Pengecekan *Curriculum Vitae*

Pada tampilan akhir ini, terdapat informasi hasil pengecekan *Curriculum Vitae* pelamar kerja dengan deksripsi lowongan pekerjaan. Tampilan ini berguna untuk menampilkan informasi nilai kecocokan antara *Curriculum Vitae* pelamar kerja dengan deskripsi lowongan pekerjaan yang akan dilamar.

4.1.5 Fitur - Fitur

Pada aplikasi *CV Matcher* ini, fitur utama yang ditawarkan adalah pengecekan menggunakan kecerdasan buatan terhadap *Curriculum Vitae* pelamar kerja dengan deskripsi lowongan pekerjaan yang akan dilamar. Pengecekan menggunakan teknologi kecerdasan buatan tersebut berguna untuk melihat nilai kecocokan antara *Curriculum Vitae* dengan deksripsi lowongan pekerjaan tersebut. Selain itu ada beberapa fitur pendukung tambahan yang berguna untuk mempermudah proses pengecekan *Curriculum Vitae* tersebut, diantaranya adalah sebagai berikut.

a. Fitur Unggah *Curriculum Vitae*

Fitur ini berguna supaya pengguna dapat mengunggah *file Curriculum Vitae* nya untuk dilakukan pengecekan. *File* yang dapat pada aplikasi *CV Matcher* ini adalah *file* dengan jenis ekstensi *.pdf.

b. Fitur Unggah Deskripsi Lowongan Pekerjaan

Fitur ini digunakan untuk mengunggah *file* deskripsi lowongan pekerjaan yang akan dilamar oleh pelamar kerja. Deskripsi lowongan

pekerjaan tersebut biasa dapat ditemukan di berbagai iklan digital. Jenis *file* yang dapat diunggah oleh pengguna adalah *file* dengan ekstensi *.jpg, *.png, *.jpeg (*file* gambar).

4.1.6 Analisa Data

Penganalisaan data merupakan sebuah proses untuk mengetahui data yang akan digunakan pada proyek skripsi ini. Data yang akan digunakan pada proyek ini adalah data dari forum Kaggle yang merupakan hasil metode *Web Scrapping* dari sumber website livecareer.com yang dilakukan oleh Snehaan Bhawal. Data tersebut digunakan untuk melatih model kecerdasan buatan *Machine Learning* sehingga model mampu menganalisa data *Curriculum Vitae* pelamar baru dikemudian hari. Data yang diperlukan adalah sebagai berikut.

- a. Data Masukan :
 - 1) *Curriculum Vitae* pelamar kerja.
 - 2) Deskripsi lowongan pekerjaan.
- b. Data Keluaran:
 - 1) Informasi jenis klasifikasi *Curriculum Vitae* pelamar kerja.
 - 2) *Score* kecocokan antara *Curriculum Vitae* pelamar kerja dengan deskripsi lowongan pekerjaan.

4.1.6 Analisa Biaya

Berikut adalah daftar biaya yang diperlukan pada proyek skripsi ini adalah sebagai berikut.

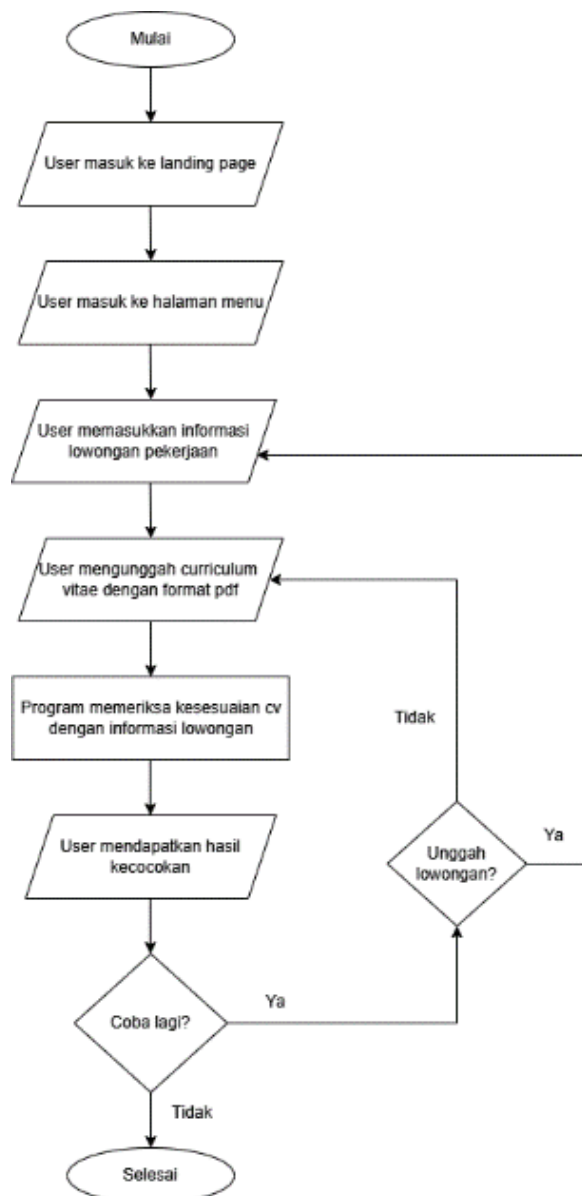
Tabel 4.3 Daftar Kebutuhan Biaya

No.	Nama Barang/Jasa	Jumlah/Satuan	Harga Perkiraan
1	Laptop ASUS X451MA	1 unit	Rp. 2.000.000,00
2	Smartphone Advan G3	1 unit	Rp. 1.200.000,00
3	Kuota Internet	10 GB	Rp. 100.000,00
4	ATK (Alat Tulis Kantor)	1 Paket	Rp. 750.000,00

4.2 Perancangan

4.2.1 Deskripsi *CV Matcher*

CV Matcher adalah sebuah aplikasi yang dapat digunakan oleh pelamar kerja untuk mengecek kecocokan *Curriculum Vitae* mereka dengan kriteria pada deskripsi lowongan pekerjaan yang akan mereka lamar. Alur program *CV Matcher* secara lebih detail ditunjukkan pada Gambar 4.1.



Gambar 4.1 Alur Program *CV Matcher*.

CV Checker diharapkan dapat meningkatkan peluang pelamar pekerjaan untuk mendapatkan pekerjaan yang sesuai dengan kemampuan diri mereka yang sudah dituliskan dalam *Curriculum Vitae*. Hal ini sesuai dengan salah tujuan dalam *Sustainable Development Goals* (SDGs) yaitu pekerjaan layak dan pertumbuhan ekonomi. *Sustainable Development Goals* (SDGs) merupakan suatu rencana aksi global yang disepakati oleh para pemimpin dunia, termasuk Indonesia, guna mengakhiri kemiskinan, mengurangi kesenjangan dan melindungi lingkungan.

4.2.2 Perancangan *AI Project Cycle*

a. *Problem Scoping*

Pendefinisian ruang lingkup masalah yang akan diselesaikan dilakukan dengan pendekatan 4W (*Who, What, Where, Why*). Berikut adalah *Problem Scoping* sebelum dilakukannya pengembangan aplikasi CV *Matcher*.

1) *Who* - Siapa yang memiliki masalah?

Pelamar kerja

2) *What* - Apa permasalahan sebenarnya?

Para pelamar kerja kini menjadi semakin sulit untuk lolos proses rekrutmen yang disebabkan oleh meningkatnya persaingan dalam mendapatkan pekerjaan, utamanya di masa pandemi COVID-19. Sebagian besar dari pelamar tersebut tidak lolos proses rekrutmen pada tahap seleksi berkas *Curriculum Vitae* (CV), dimana rata-rata hanya 10% pelamar yang berhasil lolos ke tahap interview.

3) *Where* - Dimana/pada saat apa permasalahan ini muncul?

Saat di tahap seleksi berkas dalam proses rekrutmen di perusahaan yang menggunakan sistem ATS, dimana salah satu berkas yang menjadi bahan pertimbangan pihak employer adalah CV.

4) *Why* - Mengapa masalah ini sangat penting untuk dibahas?

Masalah scoring CV dengan sistem ATS menjadi hal yang penting untuk dibahas, mengingat 63% employer menginginkan CV yang disesuaikan dengan *Job Description* pada lowongan kerja.

Apabila pelamar kerja gagal dalam proses rekrutmen hanya karena kurang mampu menonjolkan keunggulan dan potensi dirinya dalam CV, maka tentunya hal ini merugikan pelamar kerja maupun employer. Dengan memberi kemudahan kepada para pelamar kerja agar dapat memprediksi skor CV berdasarkan kecocokannya dengan *Job Description*, maka pelamar kerja akan lebih mudah dalam menyesuaikan dan menata kembali CV-nya. Melalui CV tersebut, pelamar kerja akhirnya dapat menunjukkan pengalaman dan skill yang lebih spesifik terhadap posisi yang dilamar. Dengan demikian, hal tersebut diharapkan mampu menyukseskan para pelamar kerja dalam memperoleh pekerjaan serta turut andil dalam menurunkan tingkat tunakarya di Indonesia. Hal ini mengingat adanya peningkatan angka tunakarya di Indonesia semenjak masa pandemi COVID-19, yakni dari sekitar 5,3% pada tahun 2018 kemudian meningkat hingga berada di angka 6,4% per Agustus 2021. Pengentasan tunakarya ini merupakan salah satu isu global yang juga masih menjadi permasalahan di berbagai negara. Dalam *Sustainable Development Goals* (SDGs) yang merupakan rencana aksi global PBB untuk mengakhiri kemiskinan, mengurangi kesenjangan, dan melindungi lingkungan, permasalahan tunakarya harus diselesaikan dengan tujuan untuk ketersediaan pekerjaan yang layak dan peningkatan pertumbuhan ekonomi.

b. Data Acquisition

Pada aplikasi *CV Matcher* ini, dibutuhkan data berupa kumpulan CV atau daftar riwayat hidup yang diperoleh dari Daftar riwayat hidup *Dataset* Kaggle (hasil *Web Scrapping* contoh-contoh daftar riwayat hidup pada situs www.livecareer.com oleh Snehaan Bhawal). Kumpulan CV tersebut akan digunakan untuk melatih dan mengembangkan model AI yang dapat merekognisi tingkat kecocokan CV terhadap *Job Description* dari suatu lowongan kerja.

c. *Data Exploration*

Dari *dataset* tersebut, terdapat sekitar 2400 CV atau resume dalam bentuk PDF serta dalam bentuk string (hasil ekstrak teks dari PDF). Metadata dari tiap-tiap resume telah dicatatkan dalam bentuk *file* .csv dengan rincian kolomnya sebagai berikut.

- 1) ID : *Unique identifier* dari masing-masing *file* daftar riwayat hidup
- 2) resume_str : Teks daftar riwayat hidup dalam bentuk string
- 3) resume_html : Daftar riwayat hidup dalam format HTML dari hasil *Web Scrapping*
- 4) Category : Kategori pekerjaan berdasarkan konten daftar riwayat hidup seperti di bawah ini.
 - HR
 - *Designer*
 - *Information-Technology*
 - *Teacher*
 - *Advocate*
 - *Business-Development*
 - *Healthcare*
 - *Fitness*
 - *Agriculture*
 - BPO
 - *Sales*
 - *Consultant*
 - *Digital-Media*
 - *Automobile*
 - *Chef*
 - *Finance*
 - *apparel*
 - *Engineering*
 - *Accountant*

- *Construction*
- *Public-Relations*
- *Banking*
- *Arts*
- *Aviation*

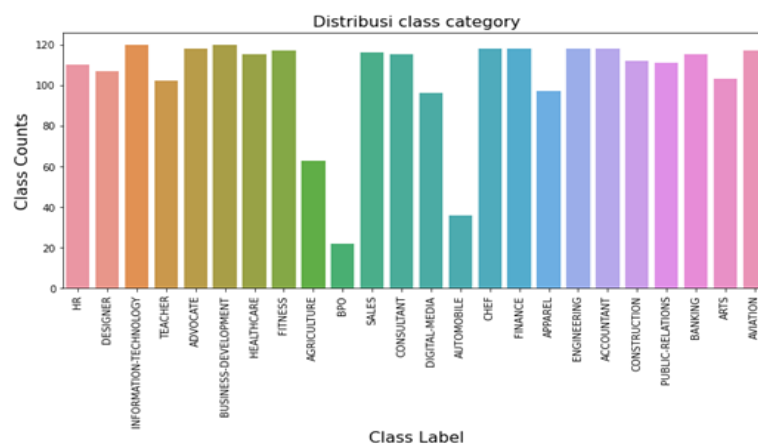
Tampilan tabel dalam *file .csv* tersebut ditunjukkan pada Gambar 4.2, dengan nama kolom 'resume_str' diubah menjadi 'resume' agar lebih mudah dipahami.

	ID		resume	Resume_html	category
0	16852973	HR ADMINISTRATOR/MARKETING ASSOCIATE\...	<div class="fontsize fontface vmargins hmargin...		HR
1	22323967	HR SPECIALIST, US HR OPERATIONS ...	<div class="fontsize fontface vmargins hmargin...		HR
2	33176873	HR DIRECTOR Summary Over 2...	<div class="fontsize fontface vmargins hmargin...		HR
3	27018550	HR SPECIALIST Summary Dedic...	<div class="fontsize fontface vmargins hmargin...		HR
4	17812897	HR MANAGER Skill Highlights ...	<div class="fontsize fontface vmargins hmargin...		HR
...
2479	99416532	RANK: SGT/E-5 NON- COMMISSIONED OFFIC...	<div class="fontsize fontface vmargins hmargin...		AVIATION
2480	24589765	GOVERNMENT RELATIONS, COMMUNICATIONS ...	<div class="fontsize fontface vmargins hmargin...		AVIATION
2481	31605080	GEEK SQUAD AGENT Professional...	<div class="fontsize fontface vmargins hmargin...		AVIATION
2482	21190805	PROGRAM DIRECTOR / OFFICE MANAGER ...	<div class="fontsize fontface vmargins hmargin...		AVIATION
2483	37473139	STOREKEEPER II Professional Sum...	<div class="fontsize fontface vmargins hmargin...		AVIATION

2484 rows x 4 columns

Gambar 4.2 Tampilan Tabel Metadata *Dataset (file .csv)*

Sementara itu, hasil EDA (Exploratory Data Analysis) pada kategori-kategori resume adalah seperti pada Gambar 4.3.



Gambar 4.3 Hasil EDA terhadap Kategori CV pada *Dataset*

d. *Modelling*

Pada tahap *Modelling* atau pemodelan, terdapat 3 langkah kunci yang diterapkan untuk mengembangkan model *CV Matcher* sebagai berikut.

1) Mengonversi *File* CV menjadi Tipe Data Teks

Langkah pertama adalah mengunggah *file* CV dalam format .pdf kemudian mengonversi *file* tersebut menjadi gambar (image) dengan format .jpg. Selanjutnya, seluruh teks dalam image akan diekstrak ke dalam bentuk string agar dapat diproses lebih lanjut untuk *Normalisasi* teks (diubah menjadi huruf non-kapital dan menghilangkan tanda baca ataupun *special character*), tokenisasi kata, menghilangkan *stopwords*, lematisasi, hingga vektorisasi kata. Dengan demikian, langkah ini sama dengan melakukan *optical character recognition* (OCR) terhadap *file* CV yang telah diunggah.

2) Mengklasifikasikan Kategori CV

Langkah mengklasifikasikan kategori CV ini akan memprediksi apakah suatu CV termasuk dalam kategori CV yang berkaitan dengan lowongan kerja di bidang HR, ataukah *Design*, *Banking*, dan seterusnya. Sebelum melakukan klasifikasi, terlebih dahulu dilakukan persiapan terhadap *Training dataset* atau disebut sebagai data *preProcessing*. Data *preProcessing* ini meliputi *label encoding* terhadap kolom *Category* (sebagai kolom target), *Text preProcessing* dengan menghilangkan tanda baca dan memvektorisasi kata, hingga *SMOTE oversampling* untuk mengatasi data *imbalance*. setelah data *preProcessing*, maka *Training dataset* siap digunakan untuk *Training model* klasifikasi. Pengklasifikasian kategori CV ini menggunakan sebuah model klasifikasi *multiclass*. Oleh sebab itu, dilakukan eksperimen terhadap beberapa jenis model yang dapat menangani klasifikasi *multiclass* untuk kemudian diterapkan *Hyperparameter Tuning*.

- *Logistic Regression*

Hyperparameter Tuning:

solver = 'lbfgs', 'Newton-cg', 'liblinear'

$penalty$ = 'l2'
 C = 0.0001, 0.001, 0.01, 0.1, 1.0
 $multi_class$ = 'auto'

Solver merupakan algoritma-algoritma dalam *Logistic Regression* yang ditujukan untuk menemukan parameter *weights* dengan *Error (cost/lost Function)* seminimal mungkin sehingga dihasilkan *output* prediksi terbaik, atau dalam kata lain, *solver* adalah algoritma untuk mengatasi permasalahan optimasi. Dalam eksperimen ini, diujikan *solver* berupa 'lbfgs' (*Default*) yang cocok digunakan untuk *dataset* dalam jumlah kecil serta dapat menangani klasifikasi *binary* maupun *multiclass*, 'Newton-cg' yang menangani klasifikasi *multiclass* dengan *multinomial*, dan 'liblinear' yang melakukan klasifikasi *binary* maupun *multiclass* dengan cara *one-versus-rest*.

Kemudian, penalti yang dimaksud pada *Hyperparameter* tersebut adalah jenis penalti yang akan digunakan dalam regularisasi model. Regularisasi adalah teknik yang digunakan untuk melakukan modifikasi pada model yang bertujuan untuk mengurangi *generalization Error*, bukan mengurangi *Training Error* seperti peran *loss Function*. *Generalization Error* yang tinggi akan menghasilkan model yang *overfitting* sehingga perlu diatasi, salah satunya dengan memberikan batasan/*constraint* atau penambahan penalti pada parameter yang digunakan. *Norm* penalti ini dapat menggunakan regularisasi L2 (Ridge regression), regularisasi L1 (Lasso regression), L1 dan L2, atau tanpa penalti. Pada eksperimen ini, hanya akan menerapkan regularisasi L2, dimana *Norm* penalti ini dapat didukung oleh *solver* 'lbfgs', 'Newton-cg', maupun 'liblinear'.

Nilai C , seperti pada *Support Vector Machine* (SVM), merupakan invers dari kekuatan regularisasi, dimana semakin kecil nilai C , maka semakin ketat/kuat regularisasi terhadap

model. *Default* nilai *C* ini adalah 1.0. Namun, pada eksperimen ini akan dicoba beberapa nilai *C*, yaitu 0.0001, 0.001, 0.01, 0.1, dan 1.0.

Multi_class mempunyai 3 opsi, yaitu ‘auto’, ‘ovr’, dan ‘multinomial’. Pada eksperimen ini, akan digunakan ‘auto’ yang akan menyesuaikan data *multiclass* untuk menerapkan ‘ovr’ (*one-versus-rest*) saat digunakan *solver* ‘liblinear’ dan akan menjadi ‘multinomial’ saat menggunakan *solver* selain ‘liblinear’.

- *Decision tree Classifier*

Hyperparameter Tuning:

criterion = ‘gini’, ‘Entropy’

splitter = ‘random’, ‘best’

max_depth = 10, 15, 20, 50, 100, 150, *None*

Criterion merupakan *measurement* untuk mengukur tingkat *impurity* pada *decision tree*. *Criterion* dengan ‘gini’ (*Default*) akan diukur dengan *Gini index*, yakni mengukur tingkat probabilitas *misclassified* di suatu node. Sedangkan, *Entropy* mengukur dari tingkat uncertainty (ketidakpastian) dari *features* terhadap target klasifikasi di suatu node dengan fungsi logaritma. Penggunaan logaritma pada *Entropy* ini membuat waktu komputasi menjadi lebih lama, akibatnya *Gini index* lebih sering digunakan karena lebih cepat dan efisien. Pada eksperimen ini, akan diuji dengan kedua *criterion* tersebut.

Splitter adalah strategi untuk memilih split pada tiap node. Apabila menggunakan ‘best’ (*Default*), maka akan dipilih split dengan *impurity* terendah, sedangkan ‘random’ akan memilih split secara acak. Tentunya, penggunaan *splitter* ‘best’ akan menghasilkan model yang lebih akurat, namun di sisi lain, *splitter* ‘random’ dapat mempercepat dan mengefisienkan komputasi. Oleh karena itu, pada eksperimen ini, *Decision tree Classifier* akan diuji dengan *splitter* ‘best’ maupun ‘random’.

Max_depth ini adalah *Hyperparameter* yang menentukan tingkat kedalaman dari sebuah *decision tree*, dimana hal ini terkait dengan pruning. Pruning adalah membatasi kedalaman *decision tree* dengan tujuan menghindari *overfitting*. Pada scikit-learn, secara *Default*, kedalaman *decision tree* diatur sebagai *None* yang artinya tanpa pruning. Dengan demikian, pada eksperimen pemodelan ini, *Decision tree Classifier* akan diuji dengan *max_depth* atau kedalaman tree sebesar 10, 15, 20, 50, 100, 150, dan *None* (tanpa pruning).

- *Multinomial Naïve Bayes*

Hyperparameter Tuning:

alpha = 0.01, 0.1, 1.0

Pada eksperimen ini, diujikan klasifikasi dengan *Naïve Bayes* menggunakan pendekatan *multinomial*. Berdasarkan dokumentasi scikit-learn, *Multinomial Naïve Bayes* ini cocok digunakan untuk klasifikasi dengan *features* yang diskrit, seperti contohnya melalui perhitungan kata untuk klasifikasi teks. Kemudian, dilakukan *Hyperparameter Tuning* terhadap *alpha*. *Alpha* merupakan angka yang menentukan tingkat smoothing, yakni seberapa smoothing yang diperlukan dalam mencegah terjadinya komputasi *Naïve Bayes* yang menghasilkan probabilitas nol saat ditemukan *features* yang tidak ada pada *Training dataset*. Apabila *alpha* = 1 (*Default*), maka dinamakan *Laplace smoothing*, sedangkan jika *alpha* < 1, disebut sebagai *Lidstone smoothing*, dan *alpha* = 0 menandakan tanpa smoothing. Eksperimen ini akan menguji model *Multinomial Naïve Bayes* dengan *alpha* sebesar 0.01 dan 0.1 untuk *Lidstone smoothing* serta *alpha* sebesar 1.0 untuk *Laplace smoothing*.

- *K-Nearest Neighbor (KNN) Classifier*

Hyperparameter Tuning:

n_neighbors = 3, 5, 7, 9, 10, 12, 15, 20

weights = 'uniform', 'distance'

$p = 1, 2$

Pada *KNN Classifier*, jumlah *neighbors* terdekat yang menentukan prediksi klasifikasi diatur melalui *n_neighbors*. Penentuan jumlah *neighbors* ini diusahakan untuk dapat menghasilkan *Error rate* seminimal mungkin. Perbandingan *Error rate* dari hasil tuning terhadap *n_neighbors* kemudian dapat divisualisasikan sehingga *n_neighbors* yang paling optimal dipilih menggunakan *Elbow method*.

Weights merupakan fungsi pembobotan yang digunakan dalam prediksi menggunakan *KNN Classifier*. Dalam eksperimen ini, digunakan *weights* berupa ‘*uniform*’ dan ‘*distance*’. Pembobotan secara *uniform* ialah apabila seluruh titik neighbor mempunyai bobot yang sama besar. Sedangkan, pembobotan secara *distance* yakni semakin dekat titik neighbor, maka semakin besar bobotnya sehingga mempunyai pengaruh yang lebih besar dalam menentukan hasil prediksi klasifikasi.

P mengatur jumlah pangkat dalam *Minkowski metric* yang digunakan untuk mengukur jarak (*distance*) suatu titik terhadap *neighbors*. Secara *Default*, $p = 2$ yang artinya menerapkan *Minkowski metric* dengan pangkat 2 lalu diakar pangkat 2 sehingga sama saja dengan *Euclidean distance*. Kemudian, apabila $p = 1$, artinya menerapkan *Minkowski metric* dengan pangkat 1 lalu diakar pangkat 1 sehingga sama saja dengan *Manhattan distance*. Jika selain p tersebut, maka diterapkan *Minkowski distance* dengan pangkat p lalu diakar pangkat p . Pada eksperimen ini, hanya diujikan dengan $p = 1$ (*Manhattan distance*) dan $p = 2$ (*Euclidean distance*).

- *Support Vector Machine (SVM)*

Hyperparameter Tuning:

kernel	= ‘rbf’, ‘linear’, ‘poly’ (degree=3), ‘sigmoid’
C	= 0.1, 1, 10, 100
gamma	= ‘scale’, ‘auto’

decision_Function_shape = 'ovo'

Kernel adalah fungsi yang digunakan untuk memisahkan (mengklasifikasikan) data melalui sebuah hyperplane dalam ruang yang *multi-dimensional*, dimana hyperplane tersebut dapat berupa *linear* maupun *non-linear*. Pada eksperimen ini, dilakukan percobaan dengan kernel yaitu RBF (*Radial basis Function*), *linear*, *polinomial*, dan *sigmoid*. Khusus untuk *polinomial*, degree (derajat/pangkat tertinggi *polinomial*) yang digunakan mengikuti *Default*, yakni $\text{degree} = 3$ *C* (regularisation) digunakan untuk mengatur batas *Error rate* yang dapat diterima. Jika $C = 0$, maka disebut *hard margin*, dimana tidak boleh ada *Error*, namun hal ini justru dapat menyebabkan *overfitting*. Dengan demikian, *C* umumnya diatur agar $C > 0$ (soft margin) untuk menghindari *overfitting*.

Gamma menentukan kurva/garis pembatas antar kelas dalam penggunaan kernel RBF, *Polinomial*, dan *Sigmoid*. Jika *gamma* bernilai tinggi, maka kernel akan semakin menyesuaikan bentuk sebaran data pada masing-masing kelas. Oleh sebab itu, nilai *gamma* yang terlalu tinggi akan menyebabkan *overfitting*. Akan tetapi, *gamma* yang terlalu rendah pun dapat menyebabkan *underfitting*. Pada eksperimen ini, akan dicoba menggunakan *gamma* secara 'scale' = $1 / (\text{n_features} * \text{X.var}())$ dan 'auto' = $1/\text{n_features}$.

Penggunaan SVM untuk klasifikasi *multiclass*, hanya dapat dilakukan dengan skema one-versus-one. Oleh sebab itu, perlu mengganti *Default* dari *decision_Function_shape* yang berupa 'ovr' (*one-versus-rest*) menjadi 'ovo' (*one-versus-one*).

3) Scoring CV dengan Membandingkan CV terhadap *Job Description* Lowongan Kerja

setelah klasifikasi kategori CV, maka akan dinilai kemiripan (*Similarity*) antara CV dengan teks deskripsi lowongan pekerjaan

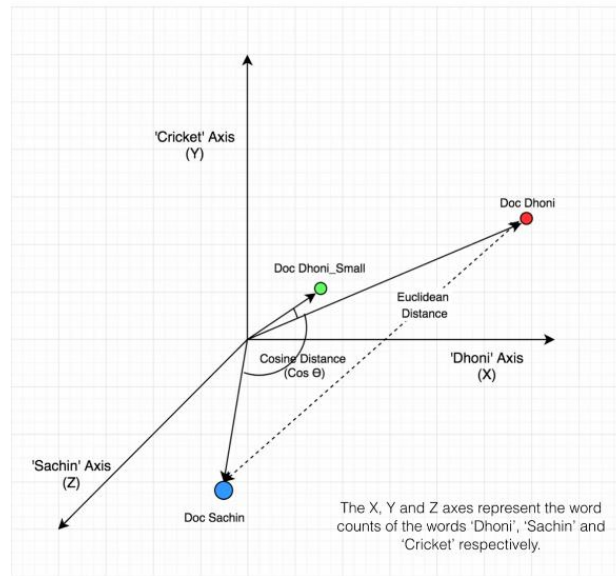
menggunakan *metric* berupa *Cosine Similarity* yang ditunjukkan pada Gambar 4.4.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}},$$

Gambar 4.4 Rumus *Cosine Similarity*

Cosine Similarity mengukur tingkat kemiripan teks antar dokumen, dimana kalkulasinya hanya mempertimbangkan arah vektor (*vector direction*), tanpa memperhitungkan jarak vektor (*vector magnitude*). Vektor yang dimaksud adalah vektor dari kata-kata dalam teks yang direpresentasikan dalam ruang n-dimensi. Dengan begitu, terlepas dari sejauh apapun jarak (*magnitude*) antara 2 vektor kata, apabila arah dari kedua vektor kata menunjukkan perbedaan derajat yang mendekati atau sama dengan 0o, maka semakin tinggi kemiripan kata-kata tersebut sehingga nilai *Cosine Similarity* mendekati atau sama dengan 1. Sebaliknya, apabila derajat antara 2 vektor mendekati atau sama dengan 90o, maka semakin rendah kemiripan katanya dan nilai *Cosine Similarity* akan mendekati atau sama dengan 0. Ilustrasi perbedaan *Cosine Similarity* yang hanya mempertimbangkan *vector direction* dengan *metric Euclidean distance* yang hanya mempertimbangkan *vector magnitude* ditunjukkan pada Gambar 4.5.

Projection of Documents in 3D Space



Gambar 4.5 Ilustrasi *Cosine Similarity*

Dalam proyek akhir ini, *Cosine Similarity* dinilai lebih tepat digunakan daripada *Euclidean distance*. Utamanya, saat terdapat kasus membandingkan 2 dokumen dengan selisih jumlah kata (*word counts*) yang berbeda jauh, namun sebenarnya konteks antara kedua dokumen tersebut sangat mirip. Kemungkinan besar hasil dari *Euclidean distance* akan menyatakan bahwa 2 dokumen mempunyai jarak kemiripan yang jauh, sedangkan hasil *Cosine Similarity* akan dapat menilai bahwa kedua dokumen tersebut masih mempunyai kemiripan konteks yang tinggi. Hal ini dikarenakan *Euclidean distance* hanya mempertimbangkan jarak vektor (*vector magnitude*) sehingga kalkulasi akan sangat dipengaruhi oleh banyaknya jumlah kata yang sama antara 2 dokumen.

e. *Evaluation*

Hasil evaluasi dari model klasifikasi *multiclass* yang digunakan untuk mengklasifikasikan kategori CV ditunjukkan pada Tabel 4.3. Dengan mempertimbangkan *metric* evaluasi model berupa akurasi, *Recall*, *Precision*, dan *F1-Score* serta mempertimbangkan kecepatan waktu

Training dan waktu prediksi, maka diperoleh model *Random Forest Classifier* dengan performa terbaik. Dengan demikian, model yang digunakan untuk klasifikasi kategori CV adalah menggunakan *Random Forest Classifier*.

Tabel 4.3 Hasil Evaluasi Model Klasifikasi Kategori CV

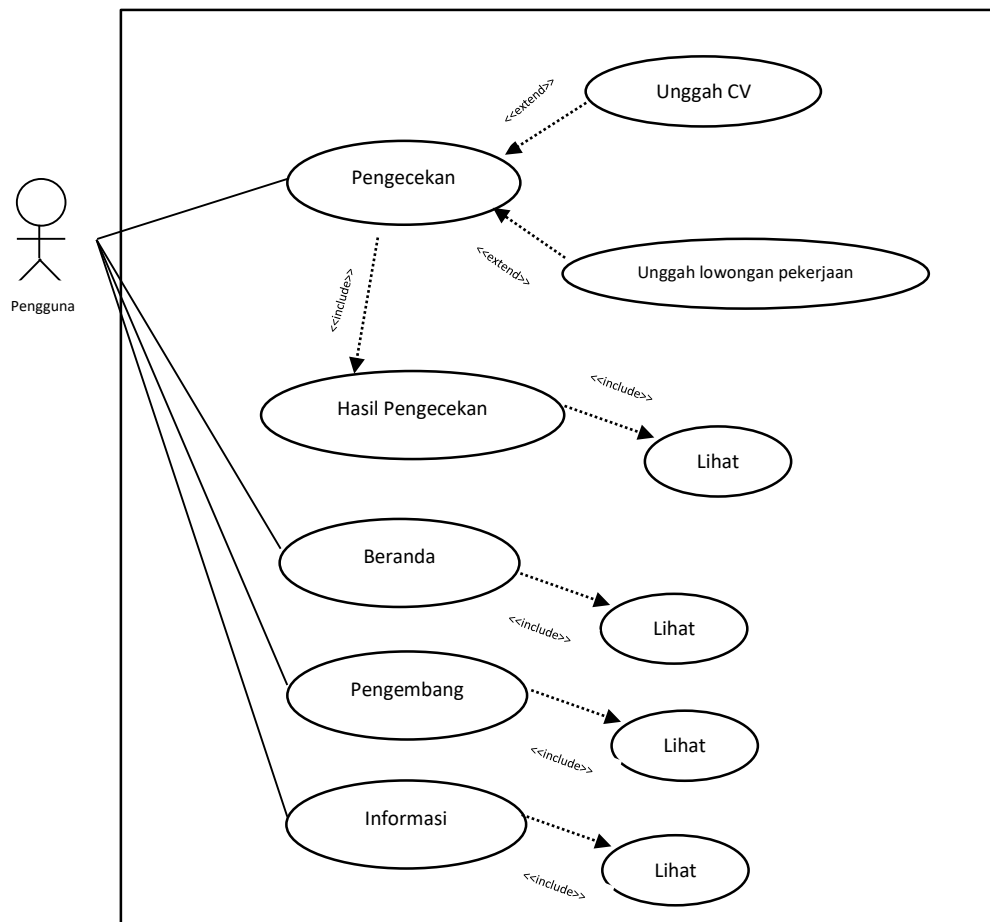
No.	Model	Hyperparameter	Accuracy	Recall	Precision	F1-Score	Waktu Training	Waktu Prediksi
1.	<i>Logistic Regression</i>	<i>solver='liblinear', penalty='l2', C=1, multi_class='auto', random_state=0</i>	72.05%	72.05%	72.84%	71.15 %	5.29 s	0.08 s
2.	<i>Decision tree Classifier</i>	<i>criterion='gini', splitter='random', max_depth=100, random_state=0</i>	66.84%	66.84%	67.84%	66.68 %	10.87 s	0.03 s
3.	<i>Random Forest Classifier</i>	<i>criterion='gini', n_estimators=100, max_depth=150, random_state=0</i>	79.69%	79.69%	80.74%	79.01 %	10.4 s	0.13 s

4.	<i>Multinomial Naïve Bayes</i>	<i>alpha=0.01</i>	67.36%	67.36%	68.18%	66.75 %	0.34 s	0.05 s
5.	<i>KNN Classifier</i>	<i>n_neighbors=5, weights='distance', p=2, metric='minkowski'</i>	41.32%	41.32%	64.26%	40.22 %	0.04 s	1.77 s
6.	<i>SVM</i>	<i>kernel='linear', gamma='scale', C=10, decision_function_shape='ovo', random_state=0</i>	77.95%	77.95%	78.87%	77.97 %	174.03 s	27.51 s

f. *Deployment*

Pada tahap *Deployment* ini dilakukan untuk membuat sebuah aplikasi CV Matcher berbasis web. Sehingga dengan mengimplementasikan model ini terhadap sebuah aplikasi akan mempermudah pengguna untuk menggunakannya. Pada tahap awal *Deployment* ini, hal pertama yang akan dilakukan adalah perancangan terlebih dahulu menggunakan model *Unified Model Language* (UML) untuk sebagai acuan aplikasi yang akan dibuat. Perancangan ini dimulai dengan membuat *Use case diagram* terlebih dahulu kemudian dilanjutkan dengan pembuatan *activity diagram* nya. *Use case diagram* berguna untuk memetakan hal-hal apa saja yang akan terdapat pada aplikasi *CV Matcher*. Sedangkan *activity diagram* berguna untuk mengetahui alur proses apa saja yang terdapat pada aplikasi *CV Matcher* tersebut. Berikut adalah rancangan *use case diagram* dan *activity diagram* nya.

- *Use case diagram*



Gambar 4.6 *Use case diagram*

Berikut adalah penjelasan dari *Use case diagram* tersebut dalam beberapa tabel di bawah ini. Dimulai dari pengguna sebagai aktor hingga 5 *use case* yang terdiri atas Pengecekan, Hasil Pengecekan, Beranda, Pengembang, dan Informasi.

Tabel 4.4 Deskripsi Aktor

Aktor	Deskripsi
Pengguna	Pengguna merupakan seseorang yang dapat menggunakan aplikasi ini untuk

	mengunggah berkas CV dan berkas lowongan pekerjaan. Selain itu pengguna juga dapat melihat hasil dari pengecekan
--	--

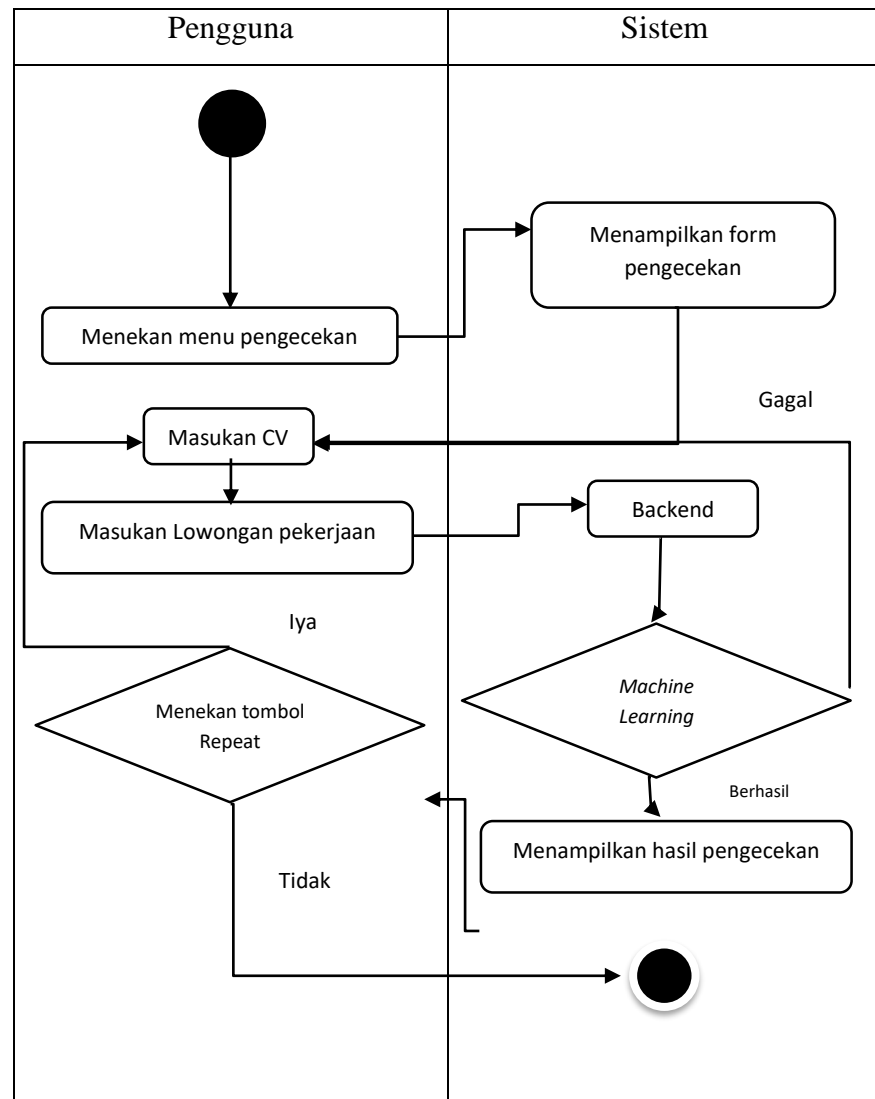
Tabel 4.5 Deskripsi *Use case* pada Aplikasi

<i>Use case</i>	Deskripsi
Pengecekan	Proses pengecekan merupakan proses dimana terjadinya pengecekan berkas CV dan berkas lowongan pekerjaan yang telah diunggah.
Beranda	Merupakan proses yang dilakukan untuk menampilkan tampilan awal aplikasi.
Pengembang	Merupakan sebuah proses yang dilakukan untuk menampilkan tampilan informasi pengembang.
Informasi	Merupakan sebuah proses yang dilakukan untuk menampilkan tampilan informasi terkait aplikasi.
Hasil Pengecekan	Merupakan proses yang dilakukan untuk melihat hasil pengecekan antara berkas CV dengan berkas lowongan pekerjaan.

- *Activity diagram*

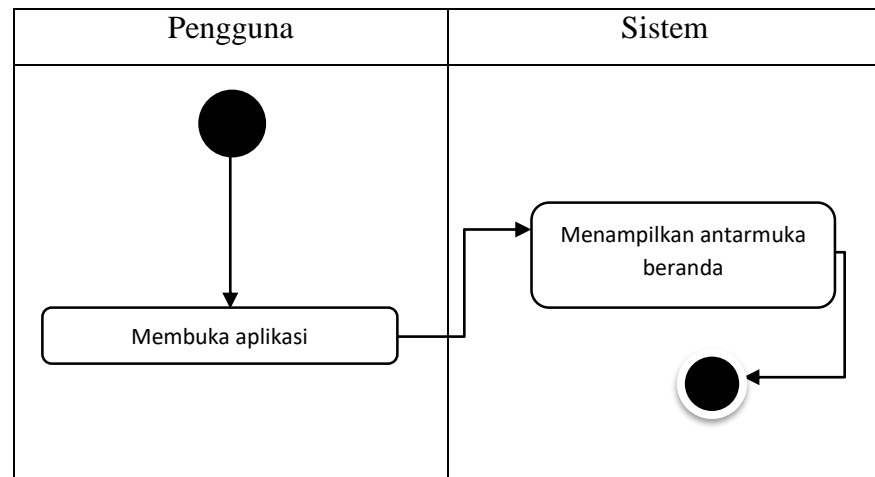
Activity diagram menggambarkan sebuah urutan aktifitas proses yang dilakukan oleh sebuah sistem. Karena *activity diagram* ini akan sangat mempengaruhi alur program aplikasi *CV Matcher* ini. Semakin detail *activity diagram* yang dibuat, maka semakin jelas alur aplikasi yang akan dibuat.

Berikut adalah *activity diagram* yang ada pada aplikasi ini terdiri dari *activity diagram* beranda, *activity diagram* pengecekan, *activity diagram* pengembang, *activity diagram* informasi.

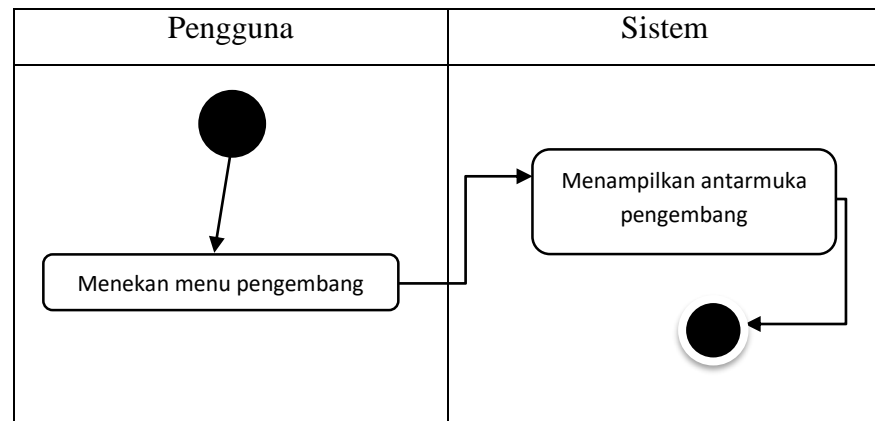


Gambar 4.7 Activity diagram Pengecekan

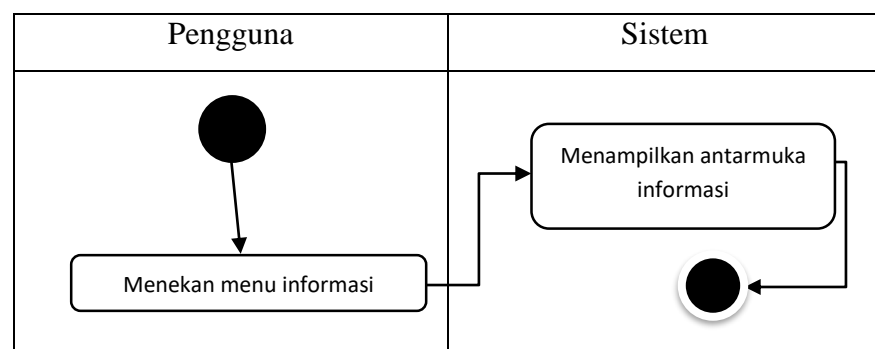
Pada gambar tersebut terlihat bahwa pertama pengguna menekan menu pengecekan, kemudian sistem menampilkan form pengecekan. Selanjutnya pengguna mengunggah CV dan lowongan pekerjaan yang akan diolah oleh backend sistem dan model *Machine Learning*. Jika *Machine Learning* berhasil melakukan pengecekan maka sistem akan menampilkan hasil pengecekan. Jika tidak berhasil maka akan kembali mengunggah berkas.



Gambar 4.8 Activity diagram Beranda

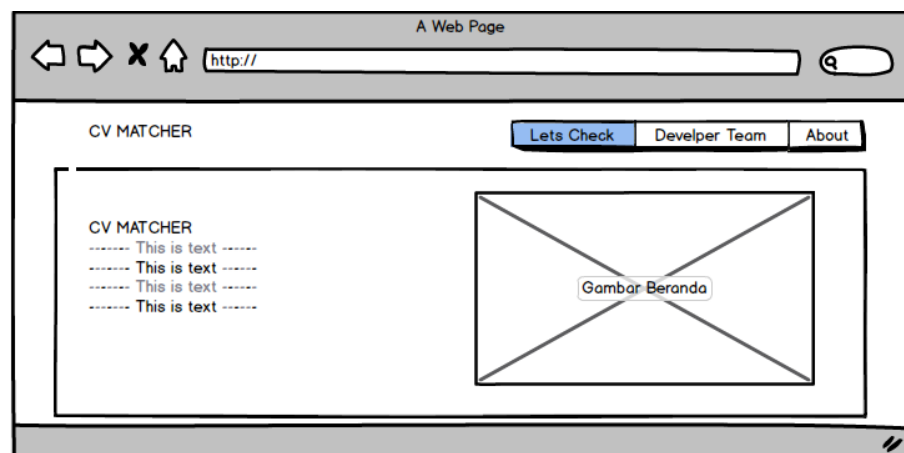


Gambar 4.9 Activity diagram Pengembang

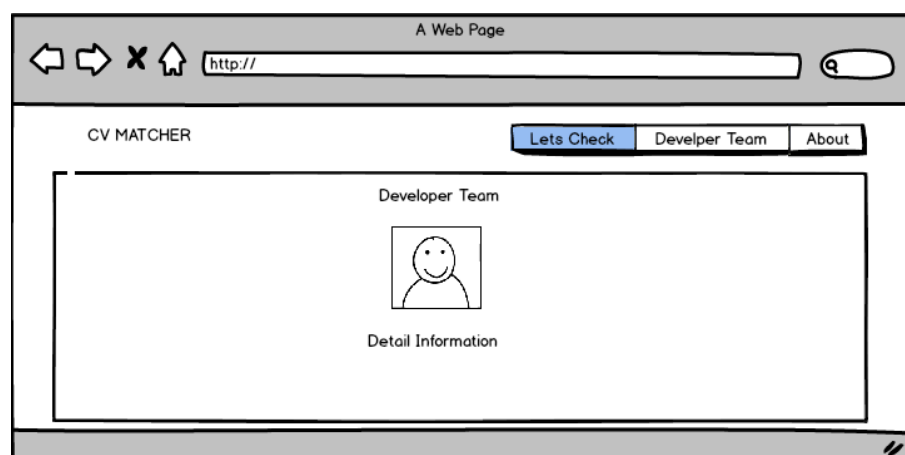


Gambar 4.10 Activity diagram Informasi

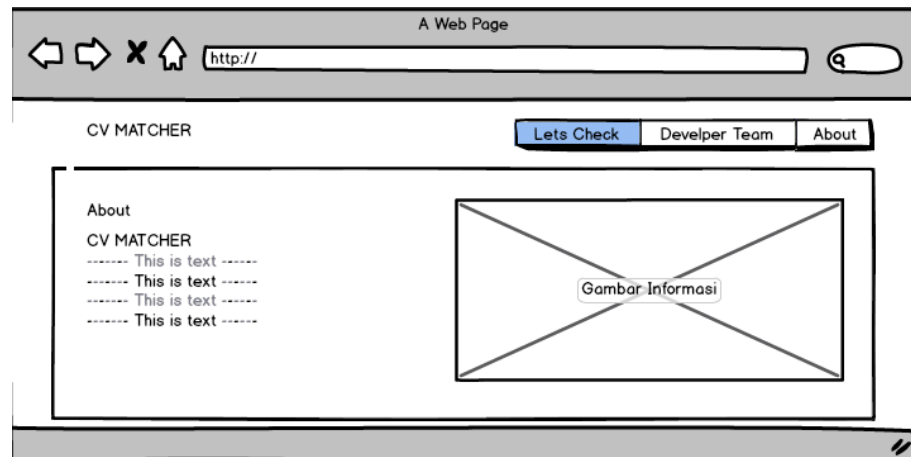
- Struktur Berkas Data
 - Berkas CV
 - ✓ Tipe : PDF (Portable *Document* Format).
 - ✓ Ukuran : Maksimal 10 MB.
 - Berkas Lowongan Pekerjaan
 - ✓ Tipe : JPG, JPEG, PNG.
 - ✓ Ukuran : Maksimal 10 MB
- Desain Sistem



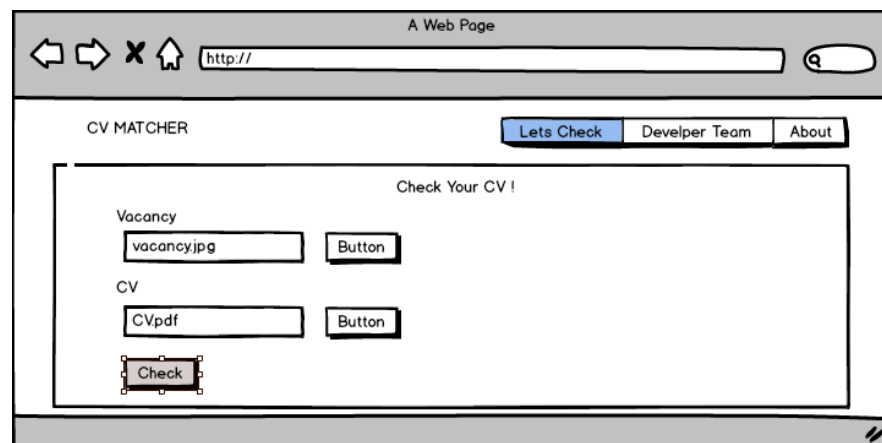
Gambar 4.11 Desain Tampilan Beranda



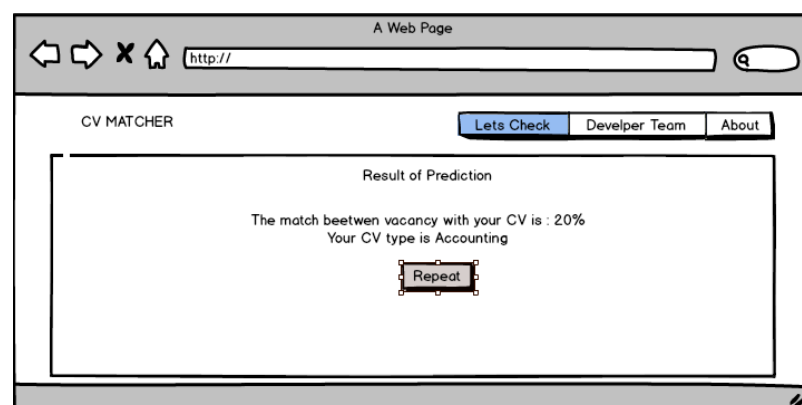
Gambar 4.12 Desain Tampilan Pengembang



Gambar 4.13 Desain Tampilan Informasi



Gambar 4.14 Desain Tampilan Pengecekan



Gambar 4.15 Desain Tampilan Hasil Pengecekan

BAB V

IMPLEMENTASI DAN PENGUJIAN

5.1 Implementasi

5.1.1 Listing Program

Berikut adalah kode-kode program yang dibuat untuk membuat proyek aplikasi *CV Matcher* ini.

a. Kode Program Untuk Mengubah *File* CV PDF Menjadi Tipe Data Teks

```
!sudo apt install tesseract-ocr
!pip install pytesseract
print("Loading...")
!apt-get install poppler-utils &> /dev/null
!pip install pdf2image &> /dev/null
import pytesseract
import shutil
import os
import random
from pdf2image import convert_from_path
try:
    from PIL import Image
except ImportError:
    import Image
from Google.colab import drive
drive.mount('/content/drive')
pages = convert_from_path('/content/drive/MyDrive/Kampus
Merdeka/Final Project SI-Orbit/classfication
Text/data/data/ACCOUNTANT/12338274.pdf', 500)
num_pages = 0
#Saving pages in jpeg format
for page in pages:
```

```

        page.save('/content/daftar riwayat hidup_'+str(num_pages)+'.jpg',
'JPEG')

        num_pages += 1
num_pages = 0
extractedInformation = ""
for page in pages:
    image_path_in_colab = ('/content/daftar riwayat
hidup_'+str(num_pages)+'.jpg')
    Text =
pytesseract.image_to_string(Image.open(image_path_in_colab))
    extractedInformation += Text
    num_pages += 1
print(extractedInformation)

```

b. Kode Program Untuk Membuat Model Machine Learning

Klasifikasi CV

```

import numpy as np
import pandas as pd
import re
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
import seaborn as sns
import matplotlib.pyplot as plt
data = pd.read_csv("/content/drive/MyDrive/Kampus Merdeka/Final
Project SI-Orbit/classfication Text/daftar riwayat hidup/daftar riwayat
hidup.csv")
data = data.rename(columns={'Category': 'category', 'Daftar riwayat
hidup_str': 'daftar riwayat hidup'})
plt.figure(figsize=(12,5))
sns.countplot(x='category', data=data)
plt.title('Distribusi class category', fontsize=16)
plt.ylabel('Class Counts', fontsize=16)

```

```

plt.xlabel('Class Label', fontsize=16)
plt.xticks(rotation='vertical');
data['category'].unique()

from sklearn.preprocessing import LabelEncoder

X = data['daftar riwayat hidup']
le = LabelEncoder()
le.fit(['HR', 'DESIGNER', 'INFORMATION-TECHNOLOGY',
'TEACHER', 'ADVOCATE',
'BUSINESS-DEVELOPMENT', 'HEALTHCARE',
'FITNESS', 'AGRICULTURE',
'BPO', 'SALES', 'CONSULTANT', 'DIGITAL-MEDIA',
'AUTOMOBILE',
'CHEF', 'FINANCE', 'APPAREL', 'ENGINEERING',
'ACCOUNTANT',
'CONSTRUCTION', 'PUBLIC-RELATIONS',
'BANKING', 'ARTS', 'AVIATION'])
print(list(le.classes_))
y = le.transform(data['category'])
#y = tweets.iloc[:, 1].values
print(X.shape)
print(X[0])
print(y.shape)
print(y[0])

import pickle
output = open('encoder.pkl', 'wb')
pickle.dump(le, output)
output.close()
# Membuat empty List
processed_category = []

for daftar riwayat hidup in range(0, len(X)):

```



```

# Hapus semua special characters
processed_daftar riwayat hidup = re.sub(r'\W', ' ', str(X[daftar
riwayat hidup]))

# Hapus semua single characters
processed_daftar riwayat hidup = re.sub(r'\s+[a-zA-Z]\s+', ' ',
processed_daftar riwayat hidup)

# Hapus single characters dari awal
processed_daftar riwayat hidup = re.sub(r'^[a-zA-Z]\s+', ' ',
processed_daftar riwayat hidup)

# Substitusi multiple spaces dengan single space
processed_daftar riwayat hidup = re.sub(r'\s+', ' ',
processed_daftar riwayat hidup, flags=re.I)

# Hapus prefixed 'b'
processed_daftar riwayat hidup = re.sub(r'^b\s+', '',
processed_daftar riwayat hidup)

# Ubah menjadi Lowercase
processed_daftar riwayat hidup = processed_daftar riwayat
hidup.lower()

# Masukkan ke list kosong yang telah dibuat sebelumnya
processed_category.append(processed_daftar riwayat hidup)

# Cek sebelum cleaning data
print(str(X[:5]))

print()

# Cek setelah cleaning data
processed_category[:5]

from sklearn.feature_extraction.Text import TfidfVectorizer
tfidfconverter = TfidfVectorizer(max_features=20000, min_df=5,
max_df=0.7,
stop_words=stopwords.words('english'),ngram_range=(1,3))
X1 = tfidfconverter.fit_transform(processed_category).toarray()
import pickle

```

```

pickle.dump(tfidfconverter, open("vectorizer.pickle", "wb"))
from collections import Counter
counter = Counter(y)
print(counter)
from imblearn.over_sampling import SMOTE
oversample = SMOTE(k_neighbors=5)
X_smote, Y_smote = oversample.fit_resample(X1, y)
from collections import Counter
counter = Counter(Y_smote)
print(counter)
from sklearn.model_selection import train_Test_split
X_train, X_Test, y_train, y_Test = train_Test_split(X_smote, Y_smote,
Test_size=0.2, random_state=2)
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import Classification_report, confusion_matrix,
Accuracy_score, Recall_score, Precision_score, f1_score,
roc_auc_score
import itertools
# Hyperparameter Tuning for Random Forest Classifier
criterion = ["gini", "Entropy"]
max_depth = [10,15,20,50,100,150,None]
n_estimators = [10,50,100]
result_randomforest = pd.DataFrame({'criterion':[], 'max_depth':[],
'n_estimators':[], 'Accuracy_score':[]})
for criterion, max_depth, n_estimators in itertools.product(criterion,
max_depth, n_estimators):

```

```

randomforest = RandomForestClassifier(criterion=criterion,
max_depth=max_depth, n_estimators=n_estimators, random_state=0)
randomforest.fit(X_train, y_train)
pred_randomforest = randomforest.predict(X_Test)
result_randomforest =
result_randomforest.append({'criterion':criterion,
'max_depth':max_depth, 'n_estimators':n_estimators,
'Accuracy_score':Accuracy_score(y_Test, pred_randomforest)},
ignore_index=True)
result_randomforest.sort_values('Accuracy_score',ascending=False).head()

# Hyperparameter Tuning for Decision tree Classifier
criterion = ["gini", "Entropy"]
splitter = ['random', 'best']
max_depth = [10,15,20,50,100,150,None]
result_dectree = pd.DataFrame({'criterion':[], 'splitter':[],
'max_depth':[], 'Accuracy_score':[]})
for criterion, splitter, max_depth in itertools.product(criterion, splitter,
max_depth):
    dectree = DecisionTreeClassifier(criterion=criterion, splitter=splitter,
max_depth=max_depth, random_state=0)
    dectree.fit(X_train, y_train)
    pred_dectree = dectree.predict(X_Test)
    result_dectree = result_dectree.append({'criterion':criterion,
'splitter':splitter, 'max_depth':max_depth,
'Accuracy_score':Accuracy_score(y_Test, pred_dectree)},
ignore_index=True)
result_dectree.sort_values('Accuracy_score',ascending=False).head()

# Hyperparameter Tuning for Logistic Regression
solver = ['lbfgs', 'Newton-cg', 'liblinear']
penalty = ['l2']
C = [0.0001, 0.001, 0.01, 0.1, 1.0]

```

```

result_logreg2 = pd.DataFrame({'solver':[], 'penalty':[], 'C':[],
                              'Accuracy_score':[]})
for solver, penalty, C in itertools.product(solver, penalty, C):
    logreg = LogisticRegression(solver=solver, penalty=penalty, C=C,
                                multi_class='auto', random_state=0)
    logreg.fit(X_train, y_train)
    pred_logreg = logreg.predict(X_Test)
    result_logreg2 = result_logreg2.append({'solver':solver,
                                           'penalty':penalty, 'C':C,
                                           'Accuracy_score':Accuracy_score(y_Test,
                                           pred_logreg)}, ignore_index=True)
result_logreg2.sort_values('Accuracy_score',ascending=False).head()
# Hyperparameter Tuning for SVM

kernel = ['rbf','linear','poly','sigmoid']
C = [0.1, 1, 10, 100]
gamma = ['scale','auto']
result_SVM = pd.DataFrame({'kernel':[], 'C':[], 'gamma':[],
                            'Accuracy_score':[]})

for kernel, C, gamma in itertools.product(kernel, C, gamma):
    svmClassifier = SVC(kernel=kernel, C=C, gamma=gamma,
                        decision_function_shape='ovo', random_state=0)
    svmClassifier.fit(X_train, y_train)
    pred_SVM = svmClassifier.predict(X_Test)
    result_SVM = result_SVM.append({'kernel':kernel, 'C':C,
                                    'gamma':gamma,
                                    'Accuracy_score':Accuracy_score(y_Test,
                                    pred_SVM)}, ignore_index=True)
result_SVM.sort_values('Accuracy_score',ascending=False).head()
# Hyperparameter Tuning for Multinomial Naive Bayes
alpha = [1, 0.1, 0.01]
result_multiNB = pd.DataFrame({'alpha':[], 'Accuracy_score':[]})
for alpha in alpha:

```

```

multiNB = MultinomialNB(alpha=alpha)
multiNB.fit(X_train, y_train)
pred_multiNB = multiNB.predict(X_Test)
result_multiNB = result_multiNB.append({'alpha':alpha,
'Accuracy_score':Accuracy_score(y_Test, pred_multiNB)},
ignore_index=True)

result_multiNB.sort_values('Accuracy_score',ascending=False).head()
# Hyperparameter Tuning for KNN Classifier
n_neighbors = [3,5,7,9,10,12,15,20]
weights = ['uniform','distance']
p = [1,2]
result_knn = pd.DataFrame({'n_neighbors':[], 'weights':[], 'p':[],
'Accuracy_score':[]})
for n_neighbors, weights, p in itertools.product(n_neighbors, weights,
p):
    knn = KNeighborsClassifier(n_neighbors=n_neighbors,
weights=weights, p=p, metric='minkowski')
    knn.fit(X_train, y_train)
    pred_knn = knn.predict(X_Test)
    result_knn = result_knn.append({'n_neighbors':n_neighbors,
'weights':weights, 'p':p,
'Accuracy_score':Accuracy_score(y_Test,
pred_knn)}, ignore_index=True)
result_knn.sort_values('Accuracy_score',ascending=False).head()
import time
Classifier = {'Random Forest': RandomForestClassifier(criterion='gini',
n_estimators=100, max_depth=150, random_state=0),
'Decision tree Classifier':
DecisionTreeClassifier(criterion='gini',splitter='random',
max_depth=100, random_state=0),

```

```

        'Logistic Regression': LogisticRegression(solver='liblinear',
penalty='l2', C=1, multi_class='auto', random_state=0),
        'SVM': SVC(kernel='linear', gamma='scale', C=10,
decision_function_shape='ovo', random_state=0),
        'Naive Bayes': MultinomialNB(alpha=0.01),
        'KNN': KNeighborsClassifier(n_neighbors=5,
weights='distance', p=2, metric='minkowski')
    }

```

```
for key, value in Classifier.items():
```

```
    Text_Classifier_en = value
```

```
    t0_en = time.time()
```

```
    Text_Classifier_en.fit(X_train, y_train)
```

```
    t1_en = time.time()
```

```
    predictions_en = Text_Classifier_en.predict(X_Test)
```

```
    t2_en = time.time()
```

```
    time_linear_train_en = t1_en-t0_en
```

```
    time_linear_predict_en = t2_en-t1_en
```

```
    print(key, "\nEN Training time: %fs; Prediction time: %fs" %
(time_linear_train_en, time_linear_predict_en))
```

```
    print("\nAccuracy = ', round(Accuracy_score(y_Test,
predictions_en)*100,2),'%')
```

```
    print('Recall = ', round(Recall_score(y_Test, predictions_en,
average='weighted')*100,2),'%')
```

```
    print('Precision = ', round(Precision_score(y_Test, predictions_en,
average='weighted')*100,2),'%')
```

```
    print('F1-Score = ', round(f1_score(y_Test, predictions_en,
average='weighted')*100,2),'%')
```

```
    print("-----")
```

```
import time
```

```
from sklearn.ensemble import RandomForestClassifier
```

```
Text_Classifier_en = RandomForestClassifier(n_estimators=100,
random_state=0)
```

```

t0_en = time.time()
Text_Classifier_en.fit(X_train, y_train)
t1_en = time.time()
predictions_en = Text_Classifier_en.predict(X_Test)
t2_en = time.time()
time_linear_train_en = t1_en-t0_en
time_linear_predict_en = t2_en-t1_en

# results
print("EN Training time: %fs; Prediction time: %fs" %
      (time_linear_train_en, time_linear_predict_en))

from sklearn.metrics import Classification_report, confusion_matrix,
Accuracy_score, Recall_score, Precision_score, f1_score,
roc_auc_score
print("Random Forest")
print('Accuracy = ', round(Accuracy_score(y_Test,
predictions_en)*100,2),'%')
print('Recall = ', round(Recall_score(y_Test, predictions_en,
average='weighted')*100,2),'%')
print('Precision = ', round(Precision_score(y_Test, predictions_en,
average='weighted')*100,2),'%')
print('F1-Score = ', round(f1_score(y_Test, predictions_en,
average='weighted')*100,2),'%')
print("")

```

c. Kode Program Untuk Membandingkan CV Dengan Lowongan Pekerjaan

```

import re
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
nltk.download('punkt')

```

```

# Import library Spacy, library untuk melakukan proses yang ada di
dalam domain nlp
from spacy.lang.en.stop_words import STOP_WORDS
from string import punctuation
Texts = extractedInformation
Texts = extractedInformation
from nltk.stem import WordNetLemmatizer
import nltk
nltk.download('wordnet')
# Memasukkan daftar stopword ke dalam variabel stopwords
stopwords = list(STOP_WORDS)
print(stopwords)
# Punctuation = karakter khusus, karakter ini akan dihilangkan dari teks
punctuation = punctuation + '\n'
punctuation
# Membuat token dari teks
doc = nlp(Texts)
tokens = [token.Text for token in doc]
print(tokens)
# Membuat dictionary bag of word
word_frequencies = { }
# Mengisi word_frequencies tanpa stopwords dan karakter khusus
for word in doc:
    if word.Text.lower() not in stopwords:
        if word.Text.lower() not in punctuation:
            if word.Text not in word_frequencies.keys():
                word_frequencies[word.Text] = 1
            else:
                word_frequencies[word.Text] += 1
print(word_frequencies)
# a denotes adjective in "pos"
lemmatizer = WordNetLemmatizer()

```



```

doc1 = list(word_frequencies.keys())
doc1 = [lemmatizer.lemmatize(word, pos='a') for word in doc1]
doc1 = [lemmatizer.lemmatize(word, pos='v') for word in doc1]
doc1 = [lemmatizer.lemmatize(word, pos='n') for word in doc1]
doc1
def listToString(s):
    # initialize an empty string
    str1 = ""
    # traverse in the string
    for ele in s:
        str1 += " "+ele
    # return string
    return str1
doc1 = listToString(doc1)
print(doc1)
processed_tweet = re.sub(r'\W', ' ', doc1)
# Hapus semua single characters
processed_tweet = re.sub(r'\s+[a-zA-Z]\s+', ' ', processed_tweet)
# Hapus single characters dari awal
processed_tweet = re.sub(r'^[a-zA-Z]\s+', ' ', processed_tweet)
# Substitusi multiple spaces dengan single space
processed_tweet = re.sub(r'\s+', ' ', processed_tweet, flags=re.I)
# Hapus prefixed 'b'
processed_tweet = re.sub(r'^b\s+', '', processed_tweet)
processed_tweet = re.sub(r'\d', '', processed_tweet)
# Ubah menjadi Lowercase
processed_tweet1 = processed_tweet.lower()
print(processed_tweet1)
# Membuat token dari teks
doc2 = nlp(keyword)
tokens = [token.Text for token in doc2]
print(tokens)

```

```

# Membuat dictionary bag of word
word_frequencies2 = { }
# Mengisi word_frequencies tanpa stopword dan karakter khusus
for word in doc2:
    if word.Text.lower() not in stopwords:
        if word.Text.lower() not in punctuation:
            if word.Text not in word_frequencies2.keys():
                word_frequencies2[word.Text] = 1
            else:
                word_frequencies2[word.Text] += 1
print(word_frequencies2)
# a denotes adjective in "pos"
doc2 = list(word_frequencies2.keys())
doc2 = [lemmatizer.lemmatize(word, pos='a') for word in doc2]
doc2 = [lemmatizer.lemmatize(word, pos='v') for word in doc2]
doc2 = [lemmatizer.lemmatize(word, pos='n') for word in doc2]
doc2
def listToString(s):
    # initialize an empty string
    str1 = ""
    # traverse in the string
    for ele in s:
        str1 += " "+ele
    # return string
    return str1
doc2 = listToString(doc2)
print(doc2)
processed_tweet = re.sub(r'\W', ' ', doc2)
# Hapus semua single characters
processed_tweet = re.sub(r'\s+[a-zA-Z]\s+', ' ', processed_tweet)
# Hapus single characters dari awal
processed_tweet = re.sub(r'^[a-zA-Z]\s+', ' ', processed_tweet)

```

```

# Substitusi multiple spaces dengan single space
processed_tweet= re.sub(r'\s+', ' ', processed_tweet, flags=re.I)
# Hapus prefixed 'b'
processed_tweet = re.sub(r'^b\s+', '', processed_tweet)
processed_tweet = re.sub(r'\d', '', processed_tweet)
# Ubah menjadi Lowercase
processed_tweet2 = processed_tweet.lower()
print(processed_tweet2)
# Program to measure the Similarity between
# two sentences using Cosine Similarity.
# X = input("Enter first string: ").lower()
# Y = input("Enter second string: ").lower()
from nltk.corpus import stopwords
# tokenization
X_list = word_tokenize(processed_tweet1)
Y_list = word_tokenize(processed_tweet2)
# sw contains the list of stopwords
sw = stopwords.words('english')
l1 = []; l2 = []
# remove stop words from the string
X_set = {w for w in X_list if not w in sw}
Y_set = {w for w in Y_list if not w in sw}
# form a set containing keywords of both strings
rvector = X_set.union(Y_set)
for w in rvector:
    if w in X_set: l1.append(1) # Create a vector
    else: l1.append(0)
    if w in Y_set: l2.append(1)
    else: l2.append(0)
c = 0
# Cosine formula
for i in range(len(rvector)):

```

```

c+= l1[i]*l2[i]
Cosine = round(c / float((sum(l1)*sum(l2))*0.5)*100, 2)
print("score kemiripan : ", Cosine, '%')
avg_score = round(((score+Cosine) / 2),2)
print('Kecocokan daftar riwayat hidup dengan job vacancy :',
avg_score,'%')

```

5.1.2 Implementasi Sistem

- a. Tempat : Bandung, Jawa Barat - Indonesia
- b. Waktu Implementasi : 25 Maret 2022

5.1.3 Spesifikasi Sistem

a. Spesifikasi Minimum *Hardware*

Berikut adalah spesifikasi minimum *Hardware* komputer untuk menjalankan program aplikasi *CV Matcher*.

Tabel 5.1 Spesifikasi Minimum *Hardware* Program *CV Matcher*

No.	Nama Perangkat
1	Processor Intel Pentium 4 atau terbaru
2	Kapasitas Ruang Kosong Harddisk 10GB
3	Kapasitas RAM 2GB

b. Spesifikasi Minimum *Software*

Tabel 5.2 Spesifikasi Minimum *Software* Program *CV Matcher*

No.	Nama <i>Software</i>
1	Sistem operasi Windows, Linux, Mac OS atau sistem operasi yang mendukung Web <i>Browser Google</i> Chrome Versi 103.0.5060 atau versi terbaru
2	<i>Google</i> Chrome versi 103.0.5060

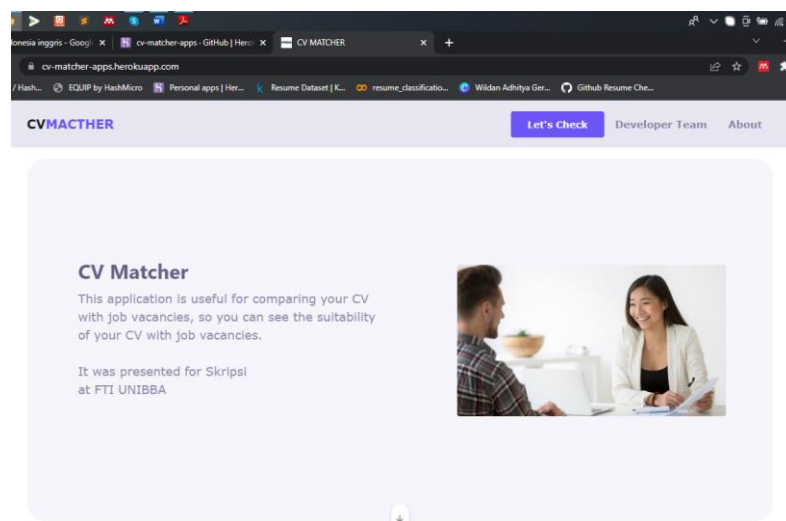
c. Instalasi Sistem

Berikut adalah langkah-langkah untuk menginstall sistem program aplikasi *CV Matcher*.

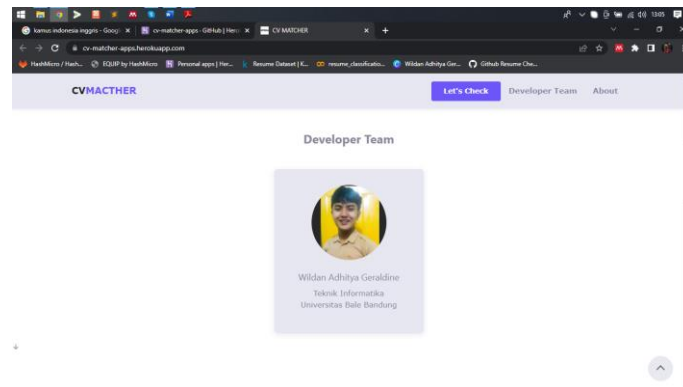
- 1) Upload kode program melalui *Github*
- 2) Kemudian buka website <https://dashboard.heroku.com/apps>
- 3) Pilih tombol 'New' dan pilih 'New app'
- 4) Isi data *app* name dan lainnya kemudian klik tombol *Create app*
- 5) setelah itu maka pada opsi '*Deployment method*' pilih *Github*, dan hubungkan heroku ini dengan repository *Github* kode program yang sudah kita upload
- 6) Kemudian setelah itu tekan tombol '*Deploy*'. Maka tunggu hingga program telah selesai di *deploy*.

d. Menjalankan Program

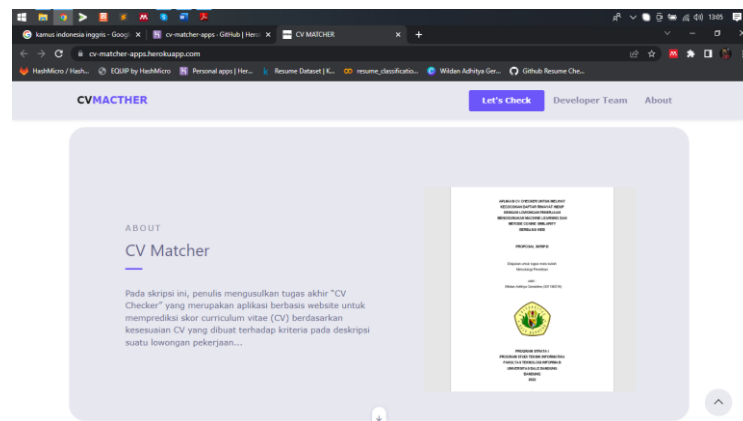
Untuk menjalankan program aplikasi *CV Matcher*, silahkan ketik alamat <https://www.cv-matcher-apps.herokuapp.com>. Kemudian maka akan tampilan Beranda sebagai berikut.



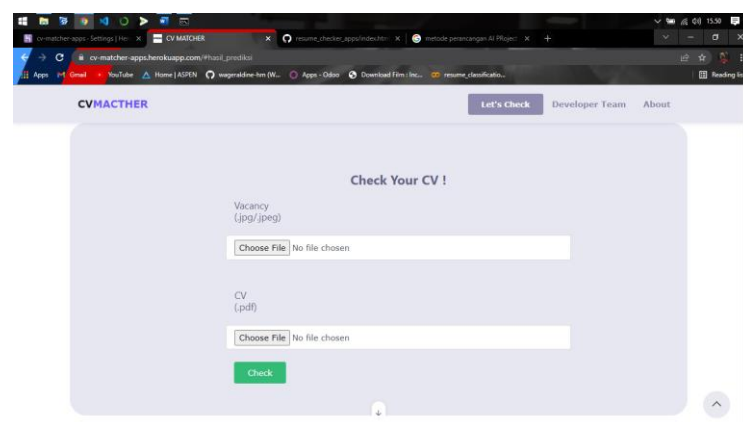
Gambar 5.1 Tampilan Beranda Aplikasi *CV Matcher*



Gambar 5.2 Tampilan *Developer Team*

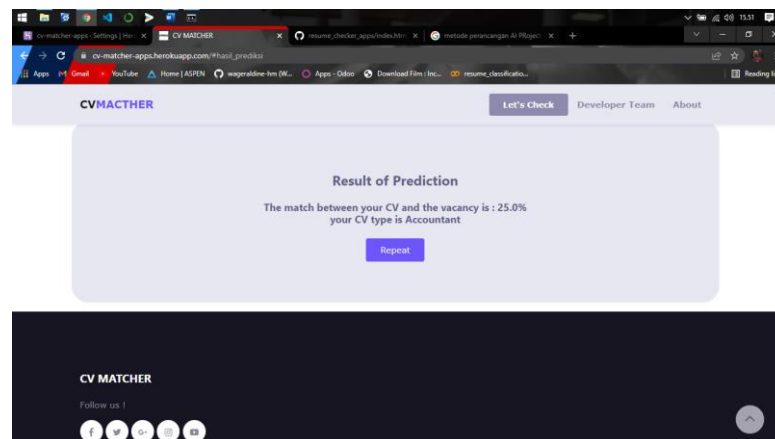


Gambar 5.3 Tampilan Informasi Program *CV Matcher*



Gambar 5.4 Tampilan Pengecekan CV dengan Lowongan Pekerjaan

Bisa dilihat pada Gambar 5.4 merupakan tampilan pengecekan CV dengan lowongan pekerjaan. Pengguna dapat langsung mengunggah *file* CV nya yang berjenis *file* *.pdf pada kolom CV. Kemudian pengguna juga dapat memasukan *file* deksripsi lowongan pekerjaan berjenis *.jpg, *.jpeg, *.png, atau jenis *file* gambar lainnya pada kolom *Vacancy*.



Gambar 5.5 Tampilan Hasil Pengecekan CV dengan Lowongan Pekerjaan

Kemudian setelah pengguna menekan tombol Check, maka pengguna dapat langsung melihat nilai kecocokan antara CV nya dengan lowongan pekerjaan tersebut. Selain itu pengguna juga dapat melihat jenis kategori CV nya yang telah dianalisa oleh kecerdasan buatan *Machine Learning*.

5.2 Pengujian

Tabel 5.3 Tabel Pengujian Aplikasi *CV Matcher*

No	Item Uji	Skenario Pengujian	Hasil yang diharapkan	Hasil pengujian	Kesimpulan
1	Halaman Beranda	Masuk ke aplikasi	Pengguna dapat melihat	Sesuai harapan	Valid

			tampilan awal aplikasi		
2	Halaman Pengembang	Klik menu halaman pengembang	Pengguna dapat melihat tampilan pengembang aplikasi	Sesuai harapan	Valid
3	Halaman Informasi	Klik halaman informasi	Pengguna dapat melihat tampilan informasi aplikasi	Sesuai harapan	Valid
4	Halaman Pengecekan	Mengunggah berkas CV dan Lowongan Pekerjaan	Pengguna dapat mengecek berkas CV dengan Lowongan Pekerjaan	Sesuai harapan	Valid
5	Halaman Hasil Pengecekan	Klik tombol Check pada halaman pengecekan	Pengguna dapat melihat hasil pengecekan berkas CV dengan Lowongan Pekerjaan	Sesuai harapan	Valid

BAB VI

KESIMPULAN

5.1 Kesimpulan

Berikut adalah kesimpulan dari proyek skripsi ini adalah sebagai berikut.

- a. *CV Matcher* adalah aplikasi yang dapat digunakan oleh pelamar kerja untuk mengecek skor *Curriculum Vitae*, yakni apakah *Curriculum Vitae* yang telah dibuat sudah sesuai dengan kriteria pada deskripsi lowongan pekerjaan yang ditentukan oleh employer.
- b. Pengembangan aplikasi *CV Matcher* dilaksanakan dengan pendekatan *AI Project Cycle*. Hal ini dimulai dari *Problem Scoping* dengan 4Ws; *Data Acquisition* dengan sumber data dari Kaggle; *Data Exploration* untuk memahami *dataset* serta mengecek ketidakseimbangan data; *Modelling* untuk konversi *file* CV menjadi teks, klasifikasi kategori CV dengan model klasifikasi *multiclass*, dan scoring CV dengan *Cosine Similarity*; *Evaluation* untuk memilih *Random Forest Classifier* sebagai model klasifikasi *multiclass* terbaik; hingga *Deployment* untuk memasukkan model *Machine Learning* ke aplikasi berbasis website agar bisa digunakan.
- c. Aplikasi *CV Matcher* ini dikembangkan dengan *Python* untuk pemodelan *Machine Learning*. Sementara itu, terkait *Deployment*, digunakan HTML, dan CSS *Bootstrap* sebagai *front-end*; *Flask* sebagai back-end; *tools* untuk repositori berupa *Github*; serta *tools* untuk hosting berupa Heroku versi 18.

5.2 Saran

Saran untuk proyek akhir *CV Matcher* ini yaitu agar kedepannya dapat meningkatkan akurasi serta performa model *Machine Learning*, baik model untuk memprediksi kategori CV, ataupun model untuk memprediksi tingkat kesesuaian CV terhadap kriteria lowongan pekerjaan.

DAFTAR PUSTAKA

- Apa itu Machine Learning? Beserta Pengertian dan Cara Kerjanya - Dicoding Blog.* (2018). Retrieved August 20, 2022, from <https://www.dicoding.com/blog/Machine-learning-adalah/>
- Arifin, N., Enri, U., & Sulistiyowati, N. (2021). Penerapan Algoritma *Support Vector Machine* (SVM) dengan TF-IDF N-Gram untuk *Text Classification*. *STRING (Satuan Tulisan Riset Dan Inovasi Teknologi)*, 6(2), 129. <https://doi.org/10.30998/string.v6i2.10133>
- Badan Pusat Statistik.* (2022). Retrieved August 20, 2022, from <https://www.bps.go.id/pressrelease/2021/11/05/1816/agustus-2021--tingkat-pengangguran-terbuka--tpt--sebesar-6-49-persen.html>
- Burges, C. J. C., & Burges, C. J. C. (1998). A Tutorial on *Support Vector Machines* for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2, 121–167. <https://www.microsoft.com/en-us/research/publication/a-tutorial-on-support-vector-Machines-for-pattern-recognition/>
- Chang, C. C., & Lin, C. J. (2011). LIBSVM. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3). <https://doi.org/10.1145/1961189.1961199>
- Cristianini, N., & Shawe-Taylor, J. (2000). An Introduction to *Support Vector Machines* and Other Kernel-based *Learning Methods*. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. <https://doi.org/10.1017/CBO9780511801389>
- Deep Learning.* (2022). Retrieved August 20, 2022, from <https://www.deeplearningbook.org/>
- Deshpande, A. R., & L.M.R.J, L. (2015). *Text Summarization using Clustering Technique and SVM Technique*. *International Journal of applied Engineering Research*, 10(10), 25511–25519.
- Gunawan, K. I., & Santoso, J. (2021). *Multilabel Text Classification Menggunakan SVM dan Doc2Vec Classification Pada Dokumen Berita Bahasa Indonesia*. *Journal of Information System, Graphics, Hospitality and Technology*, 3(01), 29–38. <https://doi.org/10.37823/insight.v3i01.126>

Handbook, F. (n.d.). *CLASS 10 Curated with support from Intel® AcKnowledgegements*.

Handoko. (2020). Analisis Proses Rekrutmen, Seleksi dan Penempatan Karyawan di PT MRT Jakarta. *Jurnal Aghniya*, 3(1), 137–138. <https://ejournal.stiesnu-bengkulu.ac.id/index.php/aghniya/article/view/41>

Hardiyanto, H. (Hardiyanto), Abdussomad, A. (Abdussomad), Haryadi, E. (Eko), Sopandi, R. (Robi), & Asep, A. (Asep). (2019). Penerapan Model Waterfall dan Uml dalam Rancang Bangun Program Pembelian Barangberorientasi Objek pada PT. FUJITA INDONESIA. *Jurnal Interkom*, 13(4), 4–11. <https://doi.org/10.35969/INTERKOM.V13I4.37>

Ladwani, V. M. (2018). *Support Vector Machines and applications. Computer Vision: Concepts, Methodologies, Tools, and applications, February*, 1381–1390. <https://doi.org/10.4018/978-1-5225-5204-8.ch057>

Mehryar Mohri -- *Foundations of Machine Learning - Book*. (n.d.). Retrieved August 20, 2022, from <https://cs.nyu.edu/~mohri/mlbook/>

Natural language Processing. (2022). Retrieved August 20, 2022, from <https://socs.binus.ac.id/2013/06/22/natural-language-Processing/>

Octaviani, P. A., Wilandari, Y., & Ispriyanti, D. (2014). PENERAPAN METODE KLASIFIKASI *SUPPORT VECTOR MACHINE* (SVM) PADA DATA AKREDITASI SEKOLAH DASAR (SD) DI KABUPATEN MAGELANG. *Undefined*.

Pengolahan bahasa alami - Wikipedia bahasa Indonesia, ensiklopedia bebas. (2022). Retrieved August 20, 2022, from https://id.wikipedia.org/wiki/Pengolahan_bahasa_alami

Perbedaan Cosine Similarity dan Cosine Distance - Hendro Prasetyo. (2022). Retrieved August 20, 2022, from <https://hendroprasetyo.com/perbedaan-Cosine-Similarity-dan-Cosine-distance/#.YwBbwnZBzDd>

Samuel, A. L. (1988). Some Studies in *Machine Learning Using the Game of Checkers*. II—Recent Progress. *Computer Games I*, 366–400. https://doi.org/10.1007/978-1-4613-8716-9_15

- Wahid, D. H., & SN, A. (2016). Peringkasan Sentimen Esktraktif di Twitter Menggunakan *Hybrid TF-IDF* dan *Cosine Similarity*. *IJCCS (Indonesian Journal of Computing and Cybernetics Systems)*, 10(2), 207–218. <https://doi.org/10.22146/IJCCS.16625>
- Waruwu, T. S., & Nasution, S. (2018). Pengembangan Keamanan Web Login Portal Dosen Menggunakan Unified *Modelling Languange* (UML). *Jurnal Mahajana Informasi*, 3(1), 34–40.
- Willy, W., Rini, D. P., & Samsuryadi, S. (2021). Perbandingan Algoritma *Random Forest Classifier*, *Support Vector Machine* dan *Logistic Regression Classifier* Pada Masalah *High Dimension* (Studi Kasus: Klasifikasi *Fake News*). *Jurnal Media Informatika Budidarma*, 5(4), 1720. <https://doi.org/10.30865/mib.v5i4.3177>

LAMPIRAN

Lampiran A-1. Daftar Riwayat Hidup



Wildan Adhitya Geraldine

Bandung, Jawa Barat - Indonesia

Whatsapp : 082315038357

Email : wgeraldine03@gmail.com

Web : <https://tipskampus.wordpress.com>

github : <https://github.com/wgeraldine>

Linkedin : <https://id.linkedin.com/in/wildan-adhitya-geraldine>

ABOUT ME

I am a software engineer who understands several fields of computer engineering such as machine learning, desktop applications, and computer networking. I currently have my own computer repair business.

EXPERIENCES

Microcredential Associate Data Scientist
Ministry of Education and Culture
(November 2021)

Make an application that is able to predict food prices in the Province of D.I Yogyakarta.

Studi Independent - Kampus Merdeka
Orbit Future Academy - AI Gen Y
(August 2021 - January 2022)

Create applications that are able to predict the suitability of job applications with job vacancies.

Second Best Student 2021
Universitas Bale Bandung
(June 2021)

make a covid 19 detection application through x-ray images of the lungs and diagnose the symptoms of covid-19.

Internship UNIBBA
PT Perkebunan Nusantara VIII
(November 2020)

Create a machine learning model that is able to analyze the quality of the tea shoots harvest.

CERTIFICATES OF EXPERTISE

- BNSP SKKNI Junior Web Programmer (HTML, CSS, PHP).
- BNSP SKKNI Junior Network Administrator.
- BNSP SKKNI Junior Computer Network Technician.
- Machine Learning For Beginner Dicoding.
- Kaggle Course Certifications.

WORK HISTORY

Computer Technician (Freelance)
Helium Computer
(2017 to Present)

Perform computer repairs including hardware and software repairs by providing home services.

Procurement Staff
CV Bina Mandiri

(November 2017 to February 2020)

Create bid files for auctions with other companies. Make an auction offer agreement with other companies.

HARD SKILLS & PROFICIENCIES

- Programing (Python, Java)
- Artificial Intelligence | Data Science | Data Engineering | Machine Learning | Deep Learning
- Microsoft Office
- Computer Networks
- Computer Repairs
- Linux Operating System

SOFT SKILLS

- Time Management
- Work Team
- Social Networking
- Creative and Critical Thinking
- Emotional Intelligence
- Communication
- Adaptability and Flexibility

EDUCATIONAL BACKGROUND

SMK TRIBAKTI PANGALENGAN
Computer and Network Engineering
(2015 - 2018)

Universitas Bale Bandung (UNIBBA)
Bachelor's Degree in Informatic Engineering
(2018 - 2022)