MANCHINE LEARNING KLASIFIKASI PENDUDUK MISKIN WILAYAH DESA TARAJAU KABUPATEN TASIKMALAYA DENGAN MENGGUNAKAN SUPPORT VEKTOR MACHINE

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh kelulusan Jenjang Strata Satu (S1) Pada Program Studi Teknik Informatika

Oleh

Sumpena

361762011

`

SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN KOMPUTER INDONESIA MANDIRI

2021

LEMBAR PENGESAHAN

MANCHINE LEARNING KLASIFIKASI PENDUDUK MISKIN WILAYAH DESA TARAJAU KABUPATEN TASIKMALAYA DENGAN MENGGUNAKAN SUPPORT VEKTOR MACHINE

Oleh
Sumpena
361762011

Skripsi ini telah diterima dan disahkan untuk
memenuhi persyaratan mencapai gelar

SARJANA TEKNIK INFORMATIKA

Pada
PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TINGGI MANAJEMEN INFORMATIKA DAN
KOMPUTER INDONESIA MANDIRI

Bandung, 17 September 2021
Disahkan oleh

Ketua Program Studi,	Dosen Pembimbing,
Chalifa Chazar, S.T., M.T.	Chalifa Chazar, S.T., M.T.
NIDN. 0421098704	NIDN. 0421098704

LEMBAR PERSETUJUAN REVISI

MANCHINE LEARNING KLASIFIKASI PENDUDUK MISKIN WILAYAH DESA TARAJAU KABUPATEN TASIKMALAYA DENGAN MENGGUNAKAN SUPPORT VEKTOR MACHINE

Oleh
Sumpena
361762011

Telah melakukan sidang skripsi dan telah melakukan revisi sesuai dengan perubahan dan perbaikan yang diminta pada saat sidang skripsi.

Bandung, 17 September 2021
Menyetujui

No	Nama Dosen	Keterangan	Tanda Tangan
1.	Chalifa Chazar, S.T., M.T.	Pembimbing	
2.	Chairuddin, Ir., M.T., M.M., Dr.	Penguji 1	
3.	Patah Herwanto, S.T., M.Kom.	Penguji 2	

Mengetahui
Ketua Program Studi Teknik Informatika

Chalifa Chazar, S.T., M.T.
NIDN. 0421098704

SURAT PERNYATAAN

Dengan ini saya menyatakan bahwa:

(1) Naskah Skripsi ini adalah asli dan belum pernah diajukan untuk mendapatkan gelar akademik, baik di Sekolah Tinggi Manajemen Informatika dan Komputer Indonesia Mandiri maupun perguruan tinggi lainnya.

(2) Skripsi ini murni merupakan karya penelitian saya sendiri dan tidak menjiplak karya pihak lain. Dalam hal ada bantuan atau arahan dari pihak lain maka telah saya sebutkan identitas dan jenis bantuannya di dalam lembar ucapan terima kasih.

(3) Seandainya ada karya pihak lain yang ternyata memiliki kemiripan dengan karya saya ini, maka hal ini adalah di luar pengetahuan saya dan terjadi tanpa kesengajaan dari pihak saya.

Pernyataan ini saya buat dengan sesungguhnya dan apabila di kemudian hari terbukti adanya kebohongan dalam pernyataan ini, maka saya bersedia menerima sanksi akademik sesuai norma yang berlaku di Sekolah Tinggi Manajemen Informatika dan Komputer Indonesia Mandiri.

Bandung, 11 Juni 2021
Yang membuat pernyataan

Sumpena
361762011

ABSTRAK

MANCHINE LEARNING KLASIFIKASI PENDUDUK MISKIN WILAYAH DESA TARAJAU KABUPATEN TASIKMALAYA DENGAN MENGGUNAKAN SUPPORT VEKTOR MACHINE

Oleh
Sumpena
361762011

Kemiskinan merupakan salah satu status sosial atau keadaan pada penduduk atau sebuah keluarga di masyarakat yang ditinjau dari segi ekonomi, pendapatan, pekerjaan, tingkat Pendidikan, kepemilikan barang dan tempat tinggal. Penduduk miskin memiliki rata-rata pengeluaran perkapita perbulan di bawah garis kemiskinan yang merupakan suatu representasi dari jumlah rupiah minimum yang dibutuhkan untuk memenuhi kebutuhan pokok minimum makanan dan kebutuhan pokok non makanan. Pada setiap Desa khususnya Desa Taraju Kecamatan Taraju Kabupaten Tasikmalaya pasti ada penduduknya yang termasuk pada kategori penduduk miskin. Untuk memperhatikan pemerataan kesejahteraan penduduknya diperlukan data penduduk miskin yang terbaru, akurat dan obyektif supaya segala bentuk bantuan dari pemerintah baik dari pemerintah daerah maupun pusat dapat sampai ke yang berhak menerimanya. Dengan adanya penelitian ini diharapkan dapat membantu Pemerintah Desa dalam pengadaan data yang diperlukan dengan pembangunan aplikasi *machine learning* untuk mengklasifikasi penduduk miskin. Metode *Support Vektor Machine* (SVM) digunakan dalam penelitian ini yang mana vaiabel-variabel yang diperlukan didapat dari observasi dan diskusi langsung dengan Perangkat Desa setempat. Untuk data masukkan diperoleh dari data penduduk yang sudah ada berupa buku induk penduduk serta data-data yang didapat secara observasi langsung dengan bantuan Perangkat Desa. Data dibagi untuk Data Training dan Data Testing. Hasil yang didapatkan dari penelitian ini berupa aplikasi machine learning menggunakan metoda *Support Vektor Machine* (SVM) yang dapat mengklasifikasi penduduk miskin dengan cepat dan akurat sehingga dapat membantu Pemerintah Desa dalam pengadaan data penduduk miskin. Semakin banyak data penduduk yang dimasukan ke aplikasi semakin tinggi pula tingkat keakuratan hasilnya.

Kata kunci : Penduduk, Miskin , SVM

ABSTRACT

MANCHINE LEARNING CLASSIFICATION OF THE POOR IN TARAJAU VILLAGE AREA, TASIKMALAYA REGENCY USING SUPPORT VECTOR MACHINE

By

Sumpena

361762011

Poverty is one of the social status or condition of the population or a family in the community in terms of economy, income, employment, level of education, ownership of goods and place of residence. The poor have an average monthly expenditure per capita below the poverty line which is a representation of the minimum amount of rupiah needed to meet the minimum basic needs of food and basic non-food needs. In every village, especially Taraju Village, Taraju District, Tasikmalaya Regency, there must be residents who are included in the category of poor people. To pay attention to the distribution of the welfare of the population, it is necessary to have data on the poor who are up to date, accurate and objective so that all forms of assistance from the government, both from the regional and central government, can reach those who are entitled to receive it. With this research, it is hoped that it can assist the Village Government in procuring the necessary data by developing a machine learning application to classify the poor. The Support Vector Machine (SVM) method was used in this study where the necessary variables were obtained from direct observation and discussion with the local village apparatus. For input data obtained from existing population data in the form of a population master book as well as data obtained by direct observation with the help of Village Officials. Data is shared for Data Training and Data Testing. The results obtained from this study are in the form of a machine learning application using the Support Vector Machine (SVM) method which can classify the poor quickly and accurately so that it can assist the Village Government in procuring data on the poor. The more population data that is entered into the application, the higher the level of accuracy of the results.

Keyword : Population, Poor, SVM

KATA PENGANTAR

Segala puji serta syukur penulis panjatkan kehadirat Allah SWT yang telah memberikan rahmat, hidayah serta kekuatan sehingga penulis dapat menyelesaikan penelitian dan pembuatan skripsi dengan judul " Machine Learning Klasifikasi Penduduk Miskin Wilayah Desa Taraju Kabupaten Tasikmalaya dengan menggunakan Support Vektor Machine ".

Dalam menyusun skripsi ini penulis menyadari bahwa tidak dapat terlaksana dengan baik tanpa bantuan dan bimbingan dari semua pihak baik ilmu, tenaga, maupun pemikiran kepada penulis. Dengan segala kerendahan hati penulis mengucapkan banyak terima kasih kepada semua pihak-pihak yang membantu dalam menyelesaikan skripsi ini.

Penulis menyadari skripsi ini masih terdapat kekurangan dan jauh dari kesempurnaan dikarenakan kurangnya kemampuan dan pengetahuan penulis. Oleh karena itu, penulis mengharapkan kritik dan saran yang membangun.

Semoga skripsi ini dapat memberikan manfaat dan wawasan khususnya bagi penulis sendiri dan berguna bagi masyarakat pada umumnya.

Bandung, 11 Juni 2021

Penulis,

Sumpena
361762011

UCAPAN TERIMA KASIH

Penulis sangat berterima kasih yang sebesar-besarnya kepada rekan-rekan, pembimbing serta pihak yang ikut serta membantu penulis dalam penyusunan skripsi ini, yaitu kepada :

1. Ibu Chalifa Chazar, S.T., M.T. selaku Dosen Pembimbing yang telah memberikan ilmu dan bimbingannya dalam penyusunan skripsi.
2. Bapak Kepala Desa Taraju beserta Jajarannya yang telah memberikan kesempatan penulis untuk melakukan kegiatan penelitian di wilayah pemerintahannya guna mendukung kelancaran penyusunan skripsi.
3. Bapak Dr. Chairuddin, Ir., M.M., M.T. selaku Ketua STMIK-IM Bandung yang telah memberikan ilmu kepada penulis.
4. Kedua orang tua dan saudara-saudara yang telah banyak memberikan dukungan penuh baik secara moral maupun materil.
5. Istri dan kedua anakku tercinta yang telah memberikan semangat, bantuan dan selalu mendukung selama proses penyusunan skripsi.
6. Bapak & Ibu Dosen yang telah memberikan penulis ilmu selama penulis melakukan kegiatan pembelajaran di STMIK-IM Bandung.
7. Rekan dan sahabat Mahasiswa / Mahasiswi di STMIK-IM Bandung.
8. Kepala dan seluruh staff Administrasi, BAAK, BAUK dan staff Perpustakaan STMIK-IM Bandung.
9. Seluruh pihak yang tidak dapat penulis sebutkan.

Demikian ucapan terima kasih penulis, semoga semua yang telah diberikan kepada penulis dalam penyusunan skripsi ini menjadi amal kebaikan dan dibalas Alloh SWT dengan pahala yang berlipat, aamiin.

Bandung, 11 Juni 2021
Penulis,

Sumpena
361762011

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I

PENDAHULUAN

1.1 Latar Belakang

Kemiskinan merupakan salah satu status sosial pada penduduk atau sebuah keluarga di masyarakat yang ditinjau dari segi ekonomi, pendapatan, pekerjaaan, tingkat pendidikan, kepemilikan barang dan tempat tinggal.

Penduduk miskin adalah penduduk atau keluarga yang memiliki kekurangan dari sisi ekonomi baik sandang, pangan maupun papan.

Badan Pusat Statistik (BPS) menggunakan konsep kemiskinan berdasarkan dari kemampuan memenuhi kebutuhan dasar (*basic needs approach*). Dengan pendekatan ini, kemiskinan dipandang sebagai ketidakmampuan dari segi ekonomi untuk memenuhi kebutuhan dasar makanan dan non makanan yang diukur dari sisi pengeluaran. Penduduk miskin memiliki rata-rata pengeluaran perkapita perbulan di bawah Garis Kemiskinan yang merupakan suatu representasi dari jumlah rupiah minimum yang dibutuhkan untuk memenuhi kebutuhan pokok minimum makanan dan kebutuhan pokok non makanan (Pratiwi, 2019).

Desa Taraju merupakan salah satu Pemerintahan Desa yang berada di wilayah Kecamatan Taraju Kabupaten Tasikmalaya dan terdiri dari 21 RT terbagi dalam 5 Dusun. Dengan banyaknya RT di Desa Taraju, Pemerintahan Desa harus memperhatikan kesejahteraan penduduknya pada setiap keluarga, terutama Penduduk kategori miskin.

Pada saat ini pendataan dan klasifikasi penduduk miskin di Desa Taraju masih dilakukan secara manual, sehingga prosesnya lamban, kurang efektif dan tepat. Proses pendataan atau klasifikasi saat ini hanya dilakukan ketika diperlukan saja, misalnya ketika ada penyaluran program bantuan dari pemerintah.

Berhubung data klasifikasi penduduk miskin masih kurang efektif dan tidak tepat maka bila ada program bantuan dari pemerintah tidak tepat sasaran dan kadang-kadang dapat menimbulkan kecemburuan sosial di masyarakat, misalnya ada keluarga yang mampu masuk klasifikasi penduduk miskin dan mendapatkan program bantuan.

Untuk klasifikasi penduduk miskin yang akurat dan efisien diperlukan suatu metode baku dan lebih baik dari metode konvensional. Salah satu metode klasifikasi *machine learning* yang cukup terkenal paling kuat dan akurat adalah metode *Support Vektor Machine* (SVM).

Support Vektor Machine (SVM) merupakan suatu teknik untuk melakukan klasifikasi maupun regresi yang sangat popular belakangan ini. SVM berada dalam satu kelas dengan *Artificial Neural Network* (ANN) dalam hal fungsi dan kondisi permasalahan yang bisa diselesaikan dan masuk dalam kelas *supervised learning* (Ritonga dan Purwaningsih, 2018).

Beberapa penelitian telah dilakukan dengan memilih metode klasifikasi terbaik, yaitu Ana Mariyam Puspitasari, dkk., pada tahun 2018 terhadap penyakit gigi dan mulut dengan menggunakan metode *Support Vektor Machine*. Hasil klasifikasi yang diperolehnya mencapai akurasi yang baik yaitu mempunyai rata-rata nilai akurasi sebesar 94.442%. Kemudian Ritonga, dkk., tahun 2018 melakukan

penelitian terhadap kualitas pengelasan SMAW (*Shield Metal Arc Welding*) dengan menggunakan metode *Support Vektor Machine*. Hasil pengujian model dengan menggunakan kernel fungsi quadratic menunjukan hasil akurasi 96,2%, dan pegujian menggunakan data uji menunjukan hasil akurasi 98%.

Dengan menerapkan metode *Support Vektor Machine* (SVM) diharapkan penelitian ini dapat membantu penyelesaian masalah ketersediaan data klasifikasi penduduk miskin di wilayah Desa Taraju Kabupaten Tasikmalaya dengan cepat dan tepat, sehingga penyaluran program-program bantuan pemerintah untuk program pengentasan kemiskinan tepat sasaran dan juga pemerataan kesejahteraan penduduk wilayah Desa Taraju dapat tercapai.

1.2 Identifikasi Masalah

Berdasarkan latar belakang penelitian yang telah diuraikan sebelumnya, maka dapat dibuat identifikasi masalah sebagai berikut :

1. Bagaimana menentukan klasifikasi penduduk miskin di wilayah Desa Taraju Kabupaten Tasikmalaya dengan efisien dan tepat.
2. Bagaimana cara mendapatkan hasil akurasi dalam menentukan klasifikasi penduduk miskin.

1.3 Tujuan Penelitian

Sesuai dengan identifikasi masalah yang telah di uraikan sebelumnya, maka tujuan penelitian ini adalah:

1. Membangun aplikasi machine learning yang dapat memudahkan Pemerintahan

Desa Taraju untuk mendapatkan data penduduk miskin di wilayahnya dengan cepat dan tepat.

2. Mengetahui cara menerapkan metode *Support Vektor Machine* (SVM) untuk klasifikasi penduduk miskin diharapkan hasilnya akurat.

1.4 Batasan Masalah

Di dalam melakukan suatu penelitian diperlukan adanya batasan suatu masalah agar penelitian tersebut lebih terarah dan memudahkan dalam pembahasan sehingga tujuan penelitian akan tercapai. Adapun batasan masalah dalam penelitian ini adalah :

1. Penelitian ini hanya dilakukan di wilayah Desa Taraju Kabupaten Tasikmalaya.
2. Sumber data yang digunakan adalah data penduduk yang ada di Pemerintahan Desa Taraju.
3. Penelitian ini terbatas hingga pada tahap *Contruction (Code & Test)*.

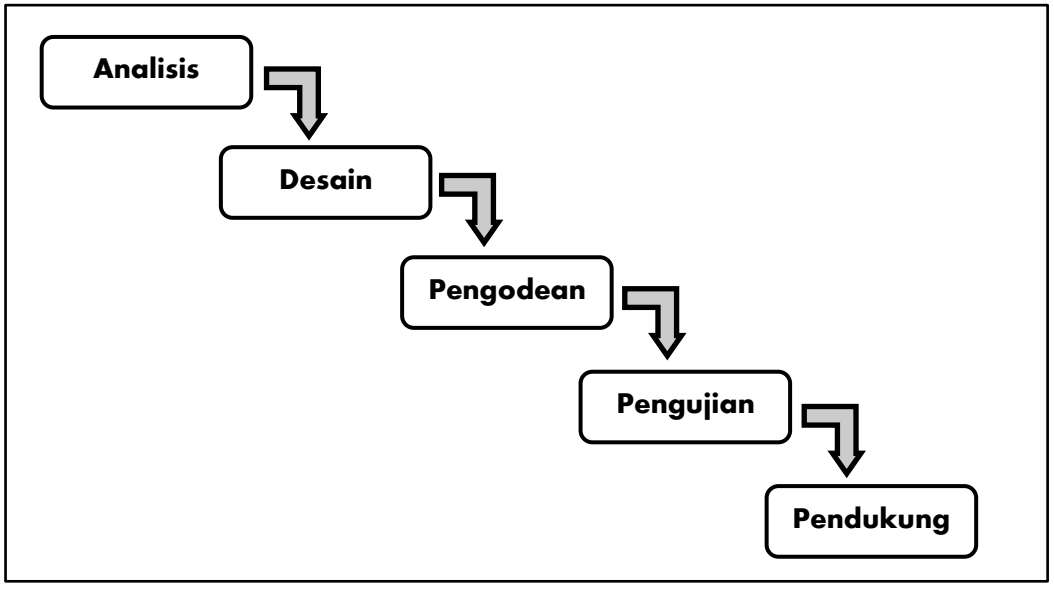
1.5 Metode Penelitian

Adapun metode-metode yang digunakan dalam penelitian adalah sebagai berikut :

1. Metode Pengumpulan Data, yaitu melakukan pengumpulan data-data dan referensi yang dibutuhkan dalam pelaksanaan penelitian baik data utama maupun data pendukung.
2. Studi Literatur, yaitu mengumpulkan bahan-bahan materi dari berbagai sumber

misalnya Buku, Jurnal, artikel, dokumen penelitian-penelitian terdahulu dan sumber lainnya yang berkaitan dengan penelitian ini. Kemudian menghimpun dan menganalisanya.

3. Metode Pengembangan Sistem menggunakan metode *Software Development Life Cycle* (SDLC) *Waterfall* yang memiliki tahapan yang runut. Gambaran urutan tahapan metode *waterfall* dapat dilihat pada Gambar 1.1.



GAMBAR: 1.1. Tahapan model *waterfall* (Rosa dan Shalahuddin, 2019:29)

Berikut ini adalah penjelasan dari tahapan-tahapan yang dilakukan dalam Model air terjun (*Waterfall):*

1. Analisis

   Analisis kebutuhan perangkat lunak merupakan proses pengumpulan kebutuhan dilakukan secara intensif untuk menspesifikasikan kebutuhan perangkat lunak agar dapat dipahami perangkat lunak seperti apa yang dibutuhkan oleh *user*. Spesifikasi kebutuhan perangkat lunak pada tahap ini perlu didokumentasikan.

2. Desain

   Desain perangkat lunak adalah proses multi langkah yang focus pada desain pembuatan program perangkat lunak termasuk struktur data, arsitektur perangkat lunak, representasi antarmuka, dan prosedur pengodean. Tahap ini mentranslasi kebutuhan perangkat lunak dari tahap analisis kebutuhan ke representasi desain agar dapat diimplementasikan menjadi program pada tahap selanjutnya. Desain ini juga perlu didokumentasikan.

3. Pengodean

   Pada tahapan ini adalah pembuatan kode program yang mana desain harus ditranslasikan ke dalam program perangkat lunak. Hasil dari tahap ini adalah program computer sesuai dengan desain yang telah dibuat pada tahap desain.

4. Pengujian

   Pengujian focus pada perangkat lunak secara dari segi lojik dan fungsional dan memastikan bahwa semua bagian sudah diuji. Hal ini dilakukan untuk meminimalisir kesalahan (*error*) dan memastikan keluaran yang dihasilkan sesuai dengan yang diinginkan.

5. Pendukung (*Support*)

   Tahap pendukung (*support*) atau pemeliharaan (*maintenance*) ini diperlukan karena tidak menutup kemungkinan sebuah perangkat lunak mengalami perubahan ketika sudah dikirimkan ke *user*. Perubahan bisa terjadi karena adanya kesalahan yang muncul dan tidak terdeteksi saat pengujian atau perangkat lunak harus beradaptasi dengan lingkungan baru. Tahap pendukung atau pemeliharaan dapat mengulangi proses pengembangan mulai dari analisis

spesifikasi untuk perubahan perangkat lunak yang sudah ada, tapi tidak untuk membuat perangkat lunak baru (Rosa A dan Shalahuddin, 2019).

1.6 Sistematika Penulisan

Sistematika penulisan skripsi ini disusun untuk memberikan gambaran umum tentang penelitian yang dilakukan. Sistematika penulisan skripsi ini adalah sebagai berikut:

BAB I PENDAHULUAN

Bab ini menguraikan tentang latar belakang melakukan penelitian, identifikasi masalah, menentukan tujuan dan kegunaan penelitian, pembatasan masalah dan metode yang digunakan dalam penelitian serta sistematika penulisan.

BAB II LANDASAN TEORI

Berisi referensi terkait dengan topik penelitian yang ditujukan untuk menunjang penulisan skripsi. Bab ini membahas berbagai konsep dasar dan teori-teori yang berkaitan dengan topik penelitian yang dilakukan.

BAB III ANALISA MASALAH DAN PERANCANGAN PROGRAM

Bab ini berisi tentang proses perancangan program berdasarkan analisa permasalahan sesuai dengan metoda pengembangan / metoda perancangan / metoda penelitian yang dipilih dalam penelitian yang dilakukan.

BAB IV IMPLEMENTASI DAN UJI COBA

Berisi tentang penjelasan dalam pengoperasian program secara bertahap. Tuliskan juga hasil evaluasi implementasi program, termasuk kelebihan dan

kekurangannya. Untuk evaluasi dapat menggunakan metode kualitatif, kuantitatif, atau metode evaluasi lain yang sesuai.

BAB V PENUTUP

Bab ini berisi kesimpulan yang didapatkan dari hasil penelitian yang telah dilakukan juga saran yang bertujuan untuk peningkatan topik skripsi yang dapat digunakan di masa mendatang.

BAB II

LANDASAN TEORI

2.1. Klasifikasi

Klasifikasi adalah proses dari pembangunan terhadap suatu model yang mengklasifikasikan suatu objek sesuai dengan atribut-atributnya. Klasifikasi merupakan suatu proses yang bertujuan untuk menentukan suatu obyek atau data kedalam suatu atau beberapa kategori atau kelas, dengan aturan dan kategori atau kelas yang sudah ditentukan sebelumnya (Mahmudah dkk, 2016).

Klasifikasi akan membangun model dari data latih kedalam kategori atau kelas yang sesuai, kemudian pada pengujian (*testing set)* model tersebut digunakan untuk mengklasifikai data yang kategori atau kelasnya belum diketahui sebelumnya. Klasifikasi data ataupun dokumen juga dpat dimualai dari membangun aturan klasifikasi tertentu yang menggunakan data training yang disebut sebagai tahapan pembelajaran dan pengujian digunakan sebagai data testing (Puspitasari dkk, 2018).

2.2. Penduduk Miskin

Badan Pusat Statistik (BPS) menggunakan konsep kemiskinan berdasarkan dari kemampuan memenuhi kebutuhan dasar (*basic needs approach*). Dengan pendekatan ini, kemiskinan dipandang sebagai ketidakmampuan dari segi ekonomi untuk memenuhi kebutuhan dasar makanan dan non makanan yang diukur dari sisi pengeluaran. Jadi Penduduk miskin adalah penduduk memiliki rata-rata

pengeluaran perkapita perbulan di bawah garis kemiskinan yang merupakan suatu representasi dari jumlah rupiah minimum yang dibutuhkan untuk memenuhi kebutuhan pokok minimum makanan dan kebutuhan pokok non makanan (Pratiwi, 2019).

Menurut Nasution (2018) pengukuran kemiskinan dilakukan melalui usaha-usaha penetapan garis kemiskinan dengan menggunakan kriteria tertentu ditetapkan garis kemiskinan yang selanjutnya proporsi penduduk dibawah garis ini digolongkan penduduk miskin. Secara umum ada 2 macam ukuran kemiskinan yang bisa digunakan, yaitu:

1. Kemiskinan Absolut

   Kemiskinan absolut adalah kemiskinan yang berkaitan dengan tingkat pendapatan dan kebutuhan yang terbatas pada kebutuhan pokok sehingga orang tersebut dapat disebut hidup dengan layak. Kemiskinan absolut diukur dengan membandingkan tingkat pendapatan seseorang dengan tingkat pendapatan yang diperlukan untuk mendapatkan kebutuhan dasarnya tersebut dengan tujuan kelangsungan hidupnya. Dengan demikian, seseorang dikatakan miskin absolut apabila pendapatan yang diperolehnya kurang dari garis kemiskinan dan tidak mampu mencukupi kebutuhan pokoknya.

2. Kemiskinan Relatif

   Kemiskinan relative dilihat dari aspek ketimpangan sosial. Apabila seseorang sudah mampu memenuhi kebutuhan dasar minimumnya, namun masih jauh lebih rendah bila dibandingkan dengan masyarakat di sekitarnya, maka orang tersebut termasuk kategori miskin relatif. Semakin tinggi kesenjangan tingkat pendapatan

antara golongan atas dengan golongan bawah maka akan semakin tinggi pula jumlah penduduk miskin. Dengan demikian, kemiskinan relatif berhubungan erat dengan distribusi pendapatan (Nasution, 2018).

Kemiskinan sering dipahami dengan rendahnya tingkat kesejahteraan semata, tetapi kemiskinan merupakan masalah yang bersifat kompleks dan multidimensi. Artinya, kemiskinan berkaitan satu sama lain dengan berbagai macam dimensi kebutuhan manusia. Hal tersebut dikarenakan kebutuhan manusia bermacam-macam, maka kemiskinan memiliki banyak aspek. Kemiskinan meliputi aspek primer yang berupa miskin akan aset, organisasi politik, pengetahuan serta keterampilan. Aspek sekunder yang berupa miskin akan jaringan sosial, sumber-sumber keuangan dan informasi. Dimensi-dimensi kemiskinan tersebut termanifestasikan dalam bentuk kekurangan gizi, air, perumahan yang sehat, perawatan kesehatan yang kurang baik dan tingkat pendidikan yang rendah (Nasution, 2018).

2.3. *Machine Learning*

Machine Learning (ML) merupakan salah satu varian dari sistem kecerdasan buatan yang memungkinkan komputer dapat belajar tanpa diprogram secara ekspisit. Secara umum, pekerjaan *Machine Learning* (ML) yang seringkali digunakan adalah untuk mengklasifikasikan satu permasalahan menjadi beberapa kelompok. Dalam kehidupan sehari-hari, objek dapat diidentifikai dengan mudah oleh manusia, namun belum tentu dapat dijelaskan secara spesifik. Maka diperlukan *Machine Learning* dalam mengenali, mengidentifikasi, ataupun memprediksi data

tertentu dengan mempelajari histori (Nurhayati dkk, 2019).

Menurut Suyanto (2018) pada awalnya komputer memang dibangun hanya untuk menghitung. Namun, sejak tahun 1960-an, para ahli sudah mulai memikirkan fungsi-fungsi komputer yang lain, yang lebih luas yaitu Belajar. Para ahli berusaha membangun konsep yang membuat komputer bisa belajar dari sejumlah pengalaman sehingga menjadi pintar.

Tom M. Mitchell pada buku (Mitchell 1997) menyatakan bahwa suatu program komputer dikatakan belajar dari pengalaman *E* yang berhubungan dengan beberapa tugas *T* dan ukuran performasi *P*, jika performasinya pada tugas-tugas *T*, sebagaimana diukur menggunakan *P*, meningkat dengan pengalaman *E* (Mitchell, 2017).

Pada era 1940-an yaitu saat pertama kali mesin komputer dibuat, para ahli hanya berfikir bagaimana melakukan komputasi secepat mungkin. Berbagai teknis dan metode komputasi dikembangkan. Tetapi belum ada pemikiran untuk membuat program komputer yang membuat komputer bisa belajar seperti manusia. Program komputer yang secara otomatis bisa semakin pintar dengan pengalaman yang didapatkannya (Suyanto, 2018).

Sejak tahun 1980-an hingga saat ini, berbagai program komputer yang memiliki kemampuan belajar telah banayak diperkenalkan. Beberapa diantaranya adalah: ALVINN (*Autonomous Land Vechile in Neural Networks*), sebuah kendaraan yang mampu mempelajari tingkah laku sopir (manusia). Setelah beberapa menit belajar menggunakan metode *Artifical Neural Networks* (ANN), ALVINN mampu berjalan secara otomatis (tanpa sopir manusia) dengan kecepatan

hingga 80 km/jam (Pomerleau 1989*). ImageNet*, sebuah basisdata citra (*image*) yang berisi jutaan citra terkelompok kedalam ribuan kelas yang digunakan untuk pembelajaran mesin berskala besar sehingga mampu melakukan klasifikasi citra dengan akurasi tinggi. Sejak 2010, *ImageNet* telah digunakan untuk menghasilkan ratusan program komputer yang mampu mempelajari karakteristik jutaan citra tersebut dan mengklasifikasikannya ke dalam ribuan kelas (VisionLab 2017). *Dragon Speak*, sebuah program komputer yang mampu belajar mengenali sinyal ucapan manusia dengan akurasi sangat tinggi (Nuance 2017). Di masa depan, komputer-komputer dengan kemampuan belajar diprediksi akan berkembang semakin pesat dengan dukungan teknologi perangkat keras komputer dan *internet of things* (IoT) yang kuat.

Teori-teori yang mendasari *machine learning* sudah muncul sejak sebelum era 1980-an. Perkembangan *machine learning* tergolong cepat, hanya dalam periode 10 tahunan (dasawarsa). Era machine learning dapat dikelompokkan ke dalam tiga era, sebagai berikut:

1. Era sebelum 1980, pada era ini, hamper semua metode *learning* melakukan pembelajaran untuk menghasilkan *linier decision surfaces*. Metode-metode pembeljaran linier ini sudah memiliki pijakan teori kuat.
2. Era 1980-an, *Decision trees* dan ANN menjadi pelopor dalam pembelajaran nonlinier, namun pijakan teorinya masih lemah. Kedua metode juga sering terjebak pada optimum lokal.
3. Era 1990 sampai sekarang, pada era ini telah dikembangkan metode-metode *learning* nonlinier yang efisien berbasis *computational learning theory*. Metode-

metode pembelajaran nonlinier ini memiliki pijakan teori yang sudah mapan.

Secara umum berdasarkan dampak yang diharapkan *user*, algoritma pembelajaran dapat dikelompokkan menjadi enam kelompok, yaitu:

1. *Supervised Learning*

   Algoritma ini membangkitkan suatu fungsi yang memetakan *input* ke *output* yang diinginkan. Kualitas hasil pembelajaran sangat tergantung pada kesesuaian *input* dan *output* yang diberikan. Dengan demikian, *user* sangat berperan dalam memvalidasi *input* dan *output* tersebut. Oleh karena itu, algoritma jenis ini disebut pembelajaran terawasi (*supervised learning*). Algoritma ini biasanya digunakan untuk menyelesaikan masalah klasifikasi maupun regresi. Masalah klasifikasi adalah bagaimana belajar (melakukan aproksimasi) tingkah laku dari suatu fungsi yang memetakan suatu vektor berdimensi N (misalnya [X_1, X_2, ... X_N]) ke dalam satu dari beberapa kelas (yang diinginkan *user*), hanya dengan melihat beberapa contoh (sampel) pasangan *input-output* dari funsi tersebut.

2. *Unsupervised Learning*

   Algoritma ini memodelkan sekumpulan *input* secara otomatis tanpa ada panduan (yang berupa *output* yang diinginkan). Artinya data-data yang dipelajari hanya berupa *input* tanpa label kelas. Algoritma ini biasanya digunakan untuk masalah klasterisasi (*clustering*), yaitu diberikan sebuah himpunan data masukan, kelompokkan data tersebut ke dalam sebuah klaster berdasarkan kriteria tertentu. Jika diberikan sekumpulan data masukan, algoritma ini mampu secara otomatis membagi data data tersebut ke dalam sejumlah klaster, berdasarkan misalnya tingkat kemiripan dalam suatu kelas.

3. *Semi- Supervised Learning*

   Algoritma ini mengkobinasikan kedua algoritma diatas, dimana sampel-sampel *input* yang diberikan ada yang berlabel dan ada yang tidak berlabel. Algoritma ini membangkitkan suatu fungsi atau pengklasifikai yang tepat berdasarkan semua sampel *input* yang diberikan.

4. *Reinforcement Learning*

   Algoritma ini mempelajari suatu kebijakan bagaimana melakukan aksi berdasarkan hasil pengamatan terhadap lingkungan yang ada. Setiap aksi menghasilkan akibat bagi lingkungan tersebut, dan lingkungan memberikan umpan balik (*feedback*) untuk memandu algoritma tersebut.

5. *Transduction*

   Algoritma ini mirip dengan *supervised learning*, tetapi tidak secara eksplisit membangun suatu fungsi. Algoritma ini justru berlatih memprediksi *output* baru berdasarkan *training inputs, training outputs*, dan *testing inputs* yang tersedia selama proses pembelajaran (pelatihan).

6. *Learning to learn*

   Algoritma ini mempelajari bias induktifnya sendiri berdasarkan pengalam-penglaman sebelumnya.

Berdasarkan *input* dan *output*-nya, algoritma pembelajaran dapat dikelompokkan ke dalam dua kategori, yaitu: diskrit dan kontinu. Sejumlah algoritma pembelajaran menerima *input* yang diskrit dan menghasilkan model yang mengeluarkan *output* diskrit juga. Sebagian yang lain menerima *input* yang kontinu dan menghasilkan model yang mengeluarkan *output* diskrit maupun kontinu.

Sebagai contoh, algoritma pembelajaran berbasis *decision tree learning* (DTL) termasuk algoritma pembelajaran diskrit. Sementara itu, *artificial neural network* (ANN) termasuk pembelajaran kontinu.

Beberapa algoritma, misalnya *support vector machine* (SVM), menerima *input* yang kontinu dan menghasilkan model yang mengeluarkan *output* diskrit yang hanya dua kelas. Jika dilihat dari *input*-nya saja, maka SVM termasuk algoritma pembelajaran kontinu. Jika memiliki himpunan data diskrit dan model yang diinginkan adalah yang mengeluarkan *output* diskrit, maka bisa menggunakan algoritma pembelajaran diskrit. Namun, jika himpunan data adalah kontinu dan model yang diinginkan adalah yang mengeluarkan *output* diskrit maupun kontinu, maka bisa menggunakan algoritma pembelajaran kontinu (Suyanto, 2018).

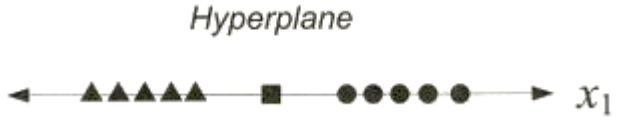
2.4. *Support Vektor Machine* (SVM)

Dalam *Machine Learning*, salah satu metode yang lebih baik teknik klasifikasinya adalah metode *Support Vektor Machine* (SVM) dibandingkan teknik klasifikasi lainnya. SVM dalam hal proses belajarnya (*learning*) termasuk dalam kelompok *Supervised learning* atau pembelajaran terawasi, dimana algoritma ini membangkitkan suatu fungsi yang memetakan *input* ke *output* yang diinginkan dan biasanya digunakan untuk menyelesaikan masalah klasifikasi maupun regresi dengan linier maupun non linier. Disebut pembelajaran terawasi karena untuk mencapai hasil dengan kualitas baik sangat bergantung kepada kesesuaian *input* dan *output* yang diberikan, maka diperlukan validasi atau pengawasan sebelumnya oleh pengguna (*user*).

Konsep *Support Vektor Machine* (SVM) dapat dijelaskan secara sederhana sebagai usaha mencari *hyperplane* terbaik yang berfungsi sebagai pemisah dua buah kelas pada input space. Pattern yang merupakan anggota dari dua buah kelas: +1 dan -1, dan berbagai alternative garis pemisah (*discrimination boundaries*). Margin adalah jarak antara *hyperplane* tersebut dengan pattern terdekat dari masing-masing kelas. Pattern yang paling dekat ini disebut sebagai *support vektor*. Usaha untuk mencari lokasi *hyperplane* ini merupakan inti dari proses pembelajaran pada SVM (Munawarah dkk, 2016).

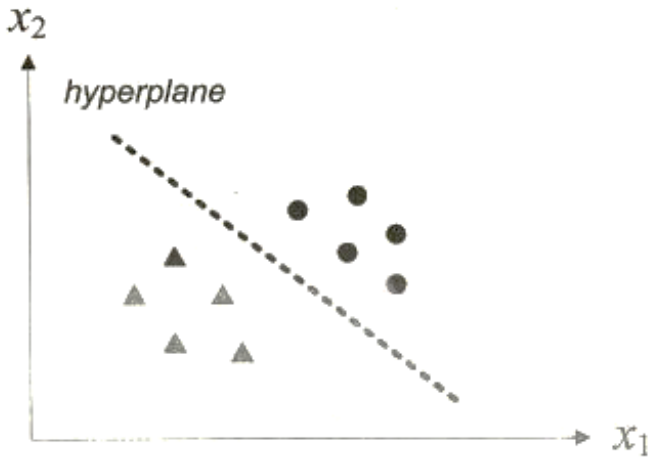
Suyanto (2018) menjelaskan bahwa dengan menggunakan algoritma pembelajaran propagasi balik yang berbasis *greedy search*, ANN tidak menjamin dihasilkannya sebuah *hyperplane* yang paling optimum. Untuk mengatasi masalah tersebut, Vapnik mengusulkan *Support Vektor Machine* (SVM) sekitar tahun 1992. Pada awalnya, SVM digunakan untuk klasifikasi data ke dalam dua kelas, dengan konsep yang secara matematis lebih matang disbanding ANN.

Pada dasarnya konsep SVM mirip dengan *perceptron* pada ANN, yaitu menemukan *hyperplane* yang memisahkan himpunan data ke dalam dua kelas secara linier. *Hyperplane* di sini adalah istilah yang sengaja dibuat general untuk semua dimensi. Untuk himpunan data berdimensi satu *hyperplane* pemisah dua kelas tersebut adalah sebuah titik, seperti diilustrasikan pada Gambar 2.1.



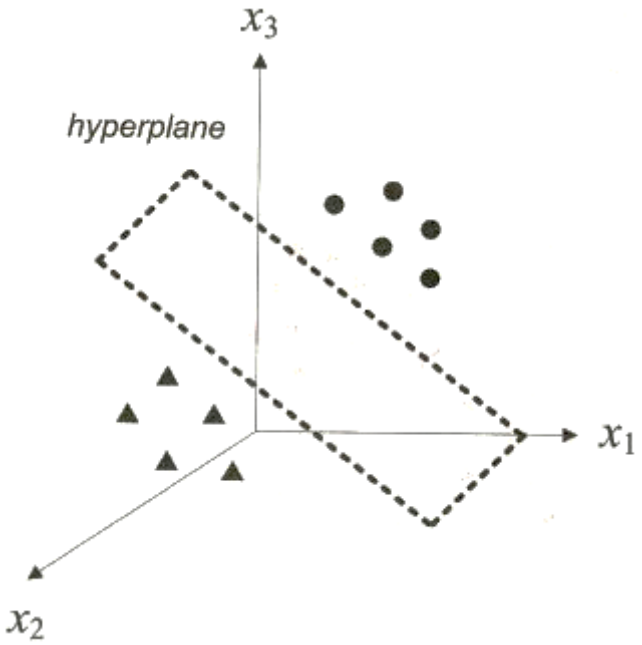
GAMBAR: 2.1. Ilustrasi *hyperplane* pemisah berupa titik (kotak) pada himpunan data satu dimensi (Suyanto, 2018:100)

Untuk himpunan data berdimensi dua, *hyperplane* pemisah adalah sebuah garis lurus, seperti diilustrasikan pada Gambar 2,2.



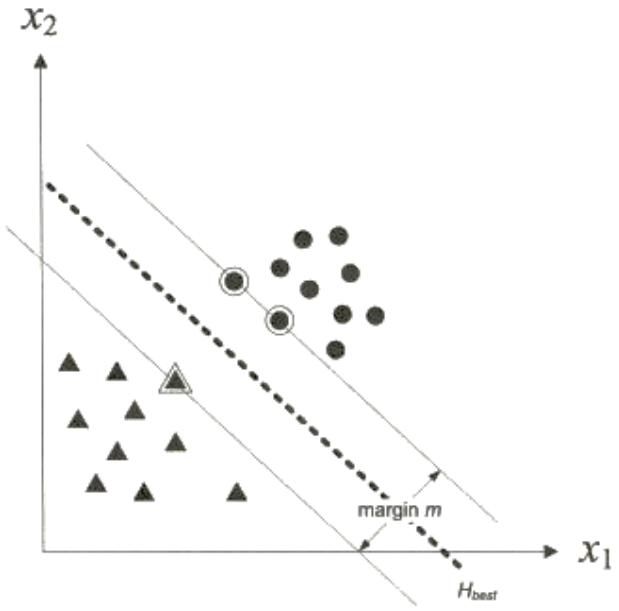
GAMBAR: 2.2. Ilustrasi *Hyperplane* pemisah berupa garis lurus pada himpunan data dua dimensi (Suyanto, 2018:100)

Untuk himpunan data berdimensi tiga, *hyperplane* pemisah adalah sebuah bidang datar, seperti diilustrasikan pada Gambar 2.3.



GAMBAR: 2.3. Ilustrasi *hyperplane* pemisah berupa bidang datar pada himpunan data tiga dimensi (Suyanto, 2018:101)

Untuk himpunan data berdimensi empat, *hyperplane* pemisah adalah sebuah bidang tiga dimensi. Untuk himpunan data berdimensi lima, *hyperplane* pemisah adalah sebuah bidang empat dimensi. Demikian seterusnya. Jadi istilah *hyperplane* bisa mengacu pada titik, garis lurus, bidang datar dua dimensi, maupun bidang-bidang lain yang berdimensi tinggi (tiga atau lebih). Berbeda dengan ANN yang tidak menjamin *hyperplane* paling optimum, SVM berusaha menemukan *hyperplane* paling optimum, yaitu sebuah *hyperplane* yang berada tepat di tengah-tengah kedua kelas sehingga memiliki jarak paling jauh ke data-data terluar di kedua kelas. Dengan kata lain, *hyperplane* tersebut memiliki margin maksimum, seperti diilustrasikan pada Gambar 2.4, sebuah *hyperplane* optimum ***H****best* terbaik yang menghasilkan margin ***m*** maksimum, dimana segitiga dan lingkaran yang diberi tanda tepian adalah data-data terluar yang berada di perbatasan kedua kelas tersebut.



GAMBAR: 2. 4. Ilustrasi *hyperplane* optimum ***Hbest*** terbaik yang menghasilkan margin ***m*** maksimum (Suyanto, 2018:102)

Berdasarkan teori pembelajaran komputasional (*computational learning theory*), sebuah *hyperplane* dikatakan optimum atau memiliki tingkat generalisasi data terbaik jika memiliki margin terbesar.

Data pada suatu dataset diberikan variabel x_i, sedangkan untuk kelas pada dataset diberikan variabel y_i. Metose SVM membagi dataset menjadi 2 kelas. Kelas pertama yang dipisah oleh hyperplane bernilai 1, sedangkan lainnya bernilai -1.

$$x_i \,.\, w + b \geq 1 \qquad untuk \quad y_i = 1 \tag{1}$$

$$x_i \,.\, w + b \leq -1 \qquad untuk \quad y_i = -1 \tag{2}$$

Keterangan :

x_i = data ke - *i*

w = nilai bobot support vector yang tegak lurus dengan *hyperplane*

b = nilai bias

y_i = kelas data ke - *i*

Bobot vector (*w*) adalah garis vektor yang tegak lurus antara titik pusat koordinat dengan garis *hyperplane*. Bias (*b*) merupakan koordinat garis relative terhadap titik koordinat. Persamaan (3) merupakan persamaan untuk menghitung nilai *b*, sedang persamaan (4) merupakan persamaan untuk mencari nilai *w*.

$$b = -\frac{1}{2}\,(w.x^{+} + w.x^{-}) \tag{3}$$

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i \tag{4}$$

Keterangan:

b = nilai bias

$w.x^{+}$ = nilai bobot untuk kelas data positif

$w.x^{-}$ = nilai bobot untuk kelas data negatif

W = bobot vektor

α_i = nilai bobot data ke - *i*

y_i = kelas data ke - *i*

x_i = data ke - *i*

H_1 adalah hyperplane pendukung dari kelas +1 yang memiliki fungsi $w\,.x1 + b = +1$.

Margin = $|dH_1 - dH_2| = \frac{2}{\|w\|}$ (5)

Keterangan:

dH_1 = jarak Hyperplane pendukung kelas +1

dH_2 = jarak Hyperplane pendukung kelas -1

Kemudian untuk menentukan *hyperplane* optimum kedua kelas menggunakan persamaan berikut:

$$\min_{w} \tau\,(w) = \frac{1}{2}\|w\|^2 \quad (6)$$

dengan batasan

$$y_i(x_i\,.w + b) - 1 \geq 0, 1 = 1, \ldots, n \quad (7)$$

Adapun slgoritma *Sequential Training* adalah sebagai berikut:

1. Inisialisasi α_i = 0 setelah itu hitung matrik *Hessian*. Matrik *Hessian* adalah perkalian antara kernel polynomial dengan nilai *y*. Nilai *y* disini yaitu nilai berupa vektor yang berisi nilai 1 dan -1. α_i digunakan untuk mencari nilai *support vector*. Untuk setiap data dari ***i*** sampai ***j***, dihitung menggunakan persamaan matrik Hessian dengan persamaan berikut:

   $$D_{ij} = y_i y_j\left(K\left(x_i, x_j\right) + \lambda^2\right) \quad (8)$$

   Untuk *i, j* = 1,…, *n*

Keterangan:

x_i = data ke - *i*

x_j = data ke - *j*

y_i = kelas data ke - *i*

y_j = kelas data ke - *j*

n = jumlah data

$K\left(x_i, x_j\right)$ = fungsi kernel yang digunakan

2. Lakukan tiga langkah di bawah ini

$$E_i = \sum_{j=1}^{i} \alpha_j D_{ij} \quad (9)$$

$$\delta\alpha_i = \min\{max[\gamma\ (1 - E_i), -\alpha_i], C - \alpha_I\} \quad (10)$$

$$\alpha_i = \alpha_i + \delta\alpha_i \quad (11)$$

Keterangan:

α_j = alfa ke -j

D_{ij} = Matriks *Hessian*

E_i = *Error rate*

γ = Kontanta *Gamma*

C = Konstanta C

$\delta\alpha_i$ = delta alfa ke-*i*

3. Kembali ke langkah 2 sampai nilai α_i mencapai konvergen (tidak ada perubahan signifikan).

Kemudian dilakukan proses testing untuk mendapatkan keputusan dimana fungsi keputusan dapat dihitung dengan persamaan (12).

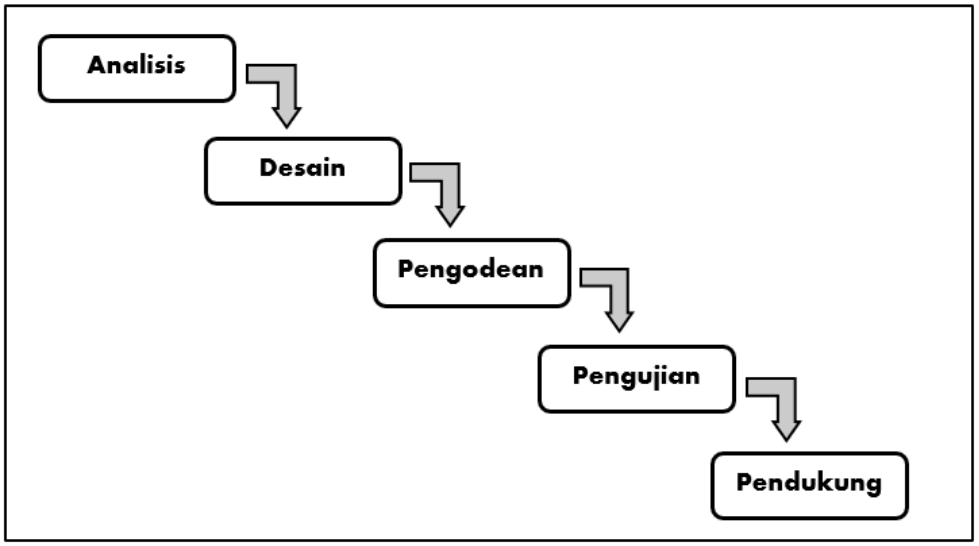
$$f(x) = \sum_{i=1}^{m} \alpha_i y_i\, K(x, x_i) + b \quad (12)$$

2.5. Metode Air Terjun *(Waterfall)*

Rosa dan Shalahuddin (2019) menerangkan, pada awal pengembangan perangkat lunak para pembuat program (*programmer*) langsung melakukan pengodean perangkat lunak tanpa menggunakan prosedur atau tahapan pengembangan perangkat lunak. Dan ditemuilah kendala-kendala seiring dengan perkembangan skala sistem-sistem perangkat yang semakin besar. *Software Development Life Cycle* atau *System Development Life Cycle* (SDLC) dimulai dari tahun 1960-an, untuk mengembangkan sistem skala besar secara fungsional untuk para konglomerat pada jaman itu. Sistem-sistem yang dibangun mengelola informasi kegiatan dan rutinitas dari perusahaan-perusahaan yang berpotensi memiliki data yang besar dalam perkembangannya.

SDLC adalah proses mengembangkan atau mengubah suatu sistem perangkat lunak dengan menggunakan model-model dan metodologi yang digunakan orang untuk mengembangkan sistem-sistem perangkat lunak sebelumnya (berdasarkan *best practice* atau cara-cara yang sudah teruji baik).

Salah satu model yang dimiliki SDLC adalah Model SDLC air terjun (*Waterfall*) yang sering disebut juga model sekuensial linier (*sequential linier*) atau alur hidup klasik (*Classic life cycle*). Model *waterfall* menyediakan pendekatan alur hidup perangkat lunak secara sekuensial atau terurut dimulai dari analisis, desain, pengodean, pengujian, dan tahap pendukung (*support*). Tahapan model air terjun (*waterfall*) dapat dilihat pada Gambar 2.5.



GAMBAR: 2.5. Tahapan model *waterfall* (Rosa dan Shalahuddin, 2019:29)

Berikut ini adalah penjelasan dari tahapan-tahapan yang dilakukan dalam Model air terjun (*Waterfall):*

1. Analisis

   Analisis kebutuhan perangkat lunak merupakan proses pengumpulan kebutuhan dilakukan secara intensif untuk menspesifikasikan kebutuhan perangkat lunak agar dapat dipahami perangkat lunak seperti apa yang dibutuhkan oleh *user*. Spesifikasi kebutuhan perangkat lunak pada tahap ini perlu didokumentasikan.

2. Desain

   Desain perangkat lunak adalah proses multi langkah yang focus pada desain pembuatan program perangkat lunak termasuk struktur data, arsitektur perangkat lunak, representasi antarmuka, dan prosedur pengodean. Tahap ini mentranslasi kebutuhan perangkat lunak dari tahap analisis kebutuhan ke representasi desain agar dapat diimplementasikan menjadi program pada tahap selanjutnya. Desain ini juga perlu didokumentasikan.

3. Pengodean

   Pada tahapan ini adalah pembuatan kode program yang mana desain harus ditranslasikan ke dalam program perangkat lunak. Hasil dari tahap ini adalah program computer sesuai dengan desain yang telah dibuat pada tahap desain.

4. Pengujian

   Pengujian focus pada perangkat lunak secara dari segi lojik dan fungsional dan memastikan bahwa semua bagian sudah diuji. Hal ini dilakukan untuk meminimalisir kesalahan (*error*) dan memastikan keluaran yang dihasilkan sesuai dengan yang diinginkan.

5. Pendukung (*Support*)

   Tahap pendukung (*support*) atau pemeliharaan (*maintenance*) ini diperlukan karena tidak menutup kemungkinan sebuah perangkat lunak mengalami perubahan ketika sudah dikirimkan ke *user*. Perubahan bisa terjadi karena adanya kesalahan yang muncul dan tidak terdeteksi saat pengujian atau perangkat lunak harus beradaptasi dengan lingkungan baru. Tahap pendukung atau pemeliharaan dapat mengulangi proses pengembangan mulai dari analisis spesifikasi untuk perubahan perangkat lunak yang sudah ada, tapi tidak untuk membuat perangkat lunak baru.

Dari kenyataan yang terjadi sangat jarang model air terjun dapat dilakukan sesuai alurnya karena sebab berikut:

a. Perubahan spesifikasi perangkat lunak terjadi di tengah alur pembangunan.
b. Sangat sulit bagi pelanggan untuk mendefinisikan semua di awal alur pembangunan. Pelanggan sering kali butuh contoh (*prototype*) untuk

menjabarkan spesifikasi kebutuhan sistem lebih lanjut.

c. Pelanggan tidak mungkin bersabar mengakomodasi perubahan yang diperlukan di akhir alur pengembangan.

Model air terjun sangat cocok digunakan kebutuhan pelanggan sudah sangat dipahami dan kemungkinan terjadinya perubahan kebutuhan selama pengembangan perangkat lunak kecil. Hal positif dari model air terjun adalah struktur tahap pengembangan sistem jelas, dokumentasi dihasilkan di setiap tahap pengembangan, dan sebuah tahap dijalankan setelah tahap sebelumnya selesai dijalankan sehingga tidak ada tumpeng tindih pelaksanaan tahapan (Rosa dan Shalahuddin, 2019) .

2.6. *Unified Modeling Language* (UML)

Rosa dan Shalahuddin (2019) dalam bukunya menerangkan bahwa pada perkembangan teknologi perangkat lunak, diperlukan adanya Bahasa yang digunakan untuk memodelkan perangkat lunak yang akan dibuat dan perlu adanya standarisasi agar orang di berbagai negara dapat mengerti pemodelan perangkat lunak. Untuk menceritakan sebuah ide dengan tujuan dan memahami hal yang sama tidaklah mudah, oleh karena itu diperlukan sebuah bahasa pemodelan perangkat lunak yang dapat dimengerti oleh banyak orang.

Banyak orang yang telah membuat bahasa pemodelan pembangunan perangkat lunak sesuai dengan teknologi pemrograman yang berkembang saat ini, misalnya yang sempat berkembang dan digunakan oleh banyak pihak adalah *Data Flow Diagram* (DFD) untuk memodelkan perangkat lunak yang menggunakan pemrograman prosedural atau struktural, kemudian juga ada *State Transition*

Diagram (STD) yang digunakan untuk memodelkan sistem *real time* (waktu nyata).

Pada perkembangan teknik pemrograman berorientasi objek, munculah sebuah standarisasi bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun dengan menggunakan teknik pemrograman berorientasi objek, yaitu *Unified Modeling Language* (UML). UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak. UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung. UML hanya berfungsi untuk melakukan pemodelan, jadi penggunaannya tidak terbatas pada metodologi tertentu, meskipun pada kenyataannya UML paling banyak digunakan pada metodologi berorientasi objek.

2.6.1. Sejarah UML

Bahasa pemrograman berorientasi objek yang pertama dikembangkan dikenal dengan nama Simula-67 yang dikembangkan pada tahun 1967. Bahasa pemrograman ini kurang berkembang dan dikembangkan lebih lanjut, namun dengan kemunculannya telah memberikan sumbangan yang besar pada developer pengembang bahasa pemrograman berorientasi objek selanjutnya.

Perkembangan aktif dari pemrograman berorientasi objek mulai menggeliat ketika berkembangnya bahasa pemrograman Smalltalk pada awal 1980-an yang kemudian diikuti dengan perkembangan bahasa pemrograman berorientasi objek yang lainnya seperti C objek, C++, Eiffel, dan CLOS. Secara actual, penggunaan bahasa pemrograman berorientasi objek pada saat itu masih terbatas, namun telah banyak menarik perhatian disaat itu. Sekitar lima tahun setelah Smalltalk

berkembang, maka berkembang pula metode pengembangan berorientasi objek. Metode yang pertama diperkenalkan oleh Sally Shlaer dan Stephen Mellor (Shlaer-Mellor, 1988) dan Peter Coad dan Edward Yourdon (Coad-Yourdon, 1991), diikuti oleh Grady Booch (Booch, 1991), James R. Rumbaugh, Michael R. Blaha, William Lorensen, Frederick Eddy, William Premerlani (Rumbaugh-Blaha-Premerlani-Eddy-Lorensen, 1991) dan banyak lagi.

Buku terkenal yang juga berkembang selanjutnya adalah karangan Ivar Jacobson (Jacobson, 1992) yang menerangkan perbedaan pendekatan yang focus pada *use case* dan proses pengembangan. Karena banyaknya metodologi-metodologi yang berkembang pesat saat itu, maka munculah ide untuk membuat sebuah bahasa yang dapat dimengerti semua orang. Usaha penyatuan ini banyak mengambil dari metodologi-metodologi yang berkembang saat itu. Maka dibuat bahasa yang merupakan gabungan dari beberapa konsep seperti konsep *Object Modelling Technique* (OMT) dari Rumbaugh dan Booch (1991), konsep *The Classes, Responsibilities, Collaborators* (CRC) dari Rebecca Wirfs-Brock (1990), konsep pemikiran Ivar Jacobson, dan beberapa konsep lainnya dimana James R. Rumbaigh, Grady Booch, dan Ivar Jacobson bergabung dalam sebuah perusahaan yang bernamana Rational Software Corporation menghasilkan bahasa yang disebut dengan *Unified Modeling Language* (UML).

Pada 1996, *Object Management Group* (OMG) mengajukan proposal agar adanya standarisasi pemodelan berorientasi objek dan pada bulan September 1997 UML diakomodasi oleh OMG sehingga sampai saat ini UML telah memberikan kontribusinya yang cukup besar di dalam metodologi berorientasi objek dan hal-hal

yang terkait di dalamnya.

Secara fisik, UML adalah sekumpulan spesifikasi yang dikeluarkan oleh OMG. UML terbaru adalah UML 2.3 yang terdiri dari 4 macam spesifikasi, yaitu Diagram Interchange Specification, UML Infrastructure, UML Superstructure, dan *Object Constraint Language* (OCL) (Rosa dan Shalahuddin, 2019).

2.6.2. Use Case Diagram

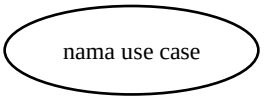
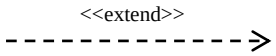
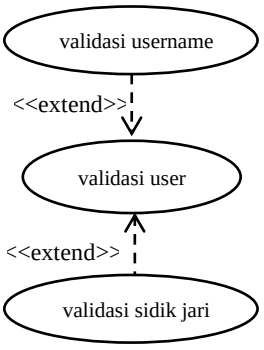
Use case atau diagram *use case* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu.

Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu :

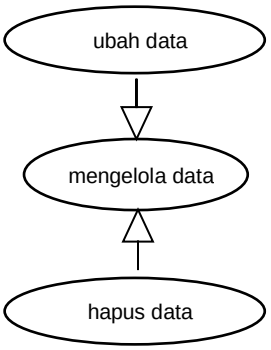
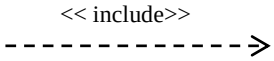
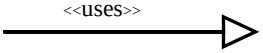
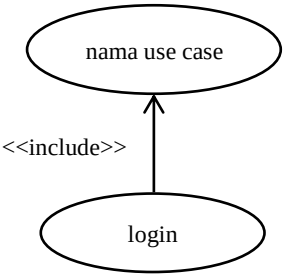
1. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat itu sendiri, jadi walaupun symbol dari actor adalah gambar orang, tapi aktor belum tentu merupakan orang.
2. Use case merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

Simbol-simbol yang ada pada *use case* diagram terdapat pada Tabel 2.1 di bawah ini.

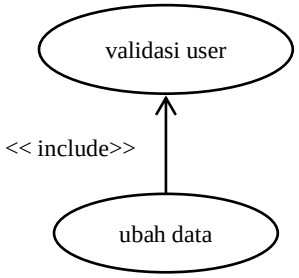
TABEL: 2.1. Simbol-simbol use case diagram (Rosa dan Shalahuddin, 2019:156)

Simbol	Deskripsi
Use Case 	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal di awal frase *nama use case*
Aktor/ *actor* 	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang , tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal frase nama aktor
Asosiasi/ *association* 	Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor
Extensi/ *Extend* 	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu; mirip dengan prinsip inheritance pada pemrograman berorientasi objek; biasanya *use case* tambahan memiliki nama depan yang sama dengan *use case* yang ditambahkan, misalnya:  arah panah mengarah pada *use case* yang menjadi extend-nya merupakan jenis yang sama dengan *use case* yang menjadi induknya

TABEL: 2.1. Simbol-simbol use case diagram (Lanjutan)

Simbol	Deskripsi
Generalisasi/ *generalization* 	Hubungan generalisasi dan spesialisasi (umum – khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lannya, misalnya:  arah panah mengarah pada *use case* yang menjadi generalisasinya (umum)
Menggunakan/ *include*/ *uses*  	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan memerlukan use case ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini. Ada dua sudut pandang yang cukup besar mengenai include di use case: 1. Include berarti use case yang ditambahkan akan selalu dipanggil saat use case tambahan dijalankan, misal pada kasus berikut: 

TABEL: 2.1. Simbol-simbol use case diagram (Lanjutan)

Simbol	Deskripsi
	2. Include berarti use case yang tambahan akan selalu melakukan pengecekan apakah use case yang ditambahkan telah dijalankan sebelum use case tambahan dijalankan, misal pada kasus berikut:  Kedua interpretasi di atas dapat dianut salah satu atau keduanya tergantung pada pertimbangan dan interpretasi yang dibutuhkan.

Use case nantinya akan menjadi kelas proses pada diagram kelas (*class diagram*) sehingga perlu dipertimbangkan penamaan yang dilakukan apakah sudah layak menjadi kelas atau belum sesuai dengan aturan pendefinisian kelas yang baik.

Setiap *use case* dilengkapi dengan skenario. Skenario *use case* adalah alur jalannya proses *use case* dari sisi aktor dan sistem. Berikut adalah format tabel skenario *use case* :

TABEL: 2.2. Format tabel skenario *use case* (Rosa dan Shalahuddin, 2019:161)

Aksi Aktor	Reaksi Sistem
Skenario Normal	
Skenario Alternatif	

Skenario *use case* dibuat per *use case* terkecil, misalkan untuk generalisasi maka skenario yang dibuat adalah *use case* yang lebih khusus. Skenario normal

adalah skenario bila sistem berjalan normal tanpa terjadi kesalahan atau *error*. Sedangkan skenario alternatif adalah skenario bila sistem tidak berjalan normal, atau mengalami *error*. Skenario normal dan skenario alternative dapat lebih dari satu. Alur dari skenario inilah yang nantinya menjadi dasar pembuatan diagram sekuen (*squence diagram*).

2.6.3. Activity Diagram

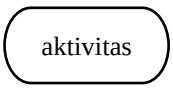
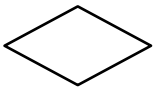
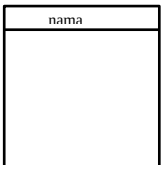
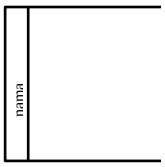
Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem.

Dalam aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut:

1. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
2. Urutan atau pengelompokan tampilan dari sistem/ user interface dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.
3. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.
4. Rancangan menu yang ditampilkan pada perangkat lunak.

Simbol-simbol yang ada pada *activity diagram* diagram terdapat pada Tabel.2.3 di bawah ini.

TABEL: 2.3. Simbol-simbol activity diagram (Rosa dan Shalahuddin, 2019:162)

Simbol	Deskripsi
Status awal 	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal
Aktivitas 	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja
Percabangan/ decision 	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu
Penggabungan	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu
Status akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir
Swimlane  atau 	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

2.6.4. Sequence Diagram

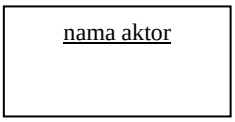
Diagram sekuen atau *sequence diagram* menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan message yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah use case beserta metode-metode yang memiliki kelas yang diinstansiasi menjadi objek itu. Membuat diagram sekuen juga dibutuhkan untuk melihat skenario yang ada

pada *use case*.

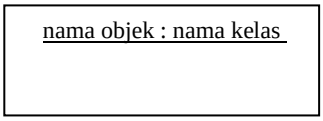
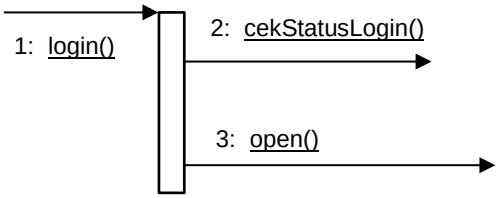
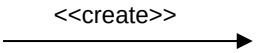
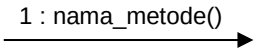
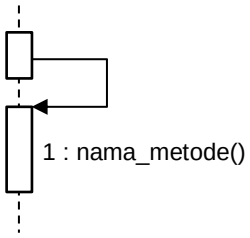
Banyaknya diagram sekuen yang harus digambar adalah minimal sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua use case yang telah didefinisikan interaksi jalnnya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak *use case* yang didefinisikan maka diagram sekuen yang harus dibuat juga semakin banyak.

Adapun simbol-simbol diagram sekuen dapat dilihat pada Tabel.2.4 di bawah ini.

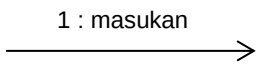
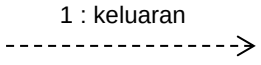
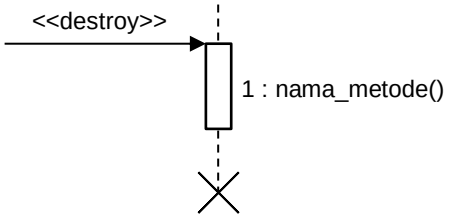
TABEL: 2.4. Simbol-simbol sequence diagram (Rosa dan Shalahuddin, 2019:165)

Simbol	Deskripsi
Aktor  nama aktor atau  tanpa waktu aktif	orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal frase nama aktor
Garis hidup/ *lifeline* 	menyatakan kehidupan suatu objek

TABEL: 2.4. Simbol-simbol sequence diagram (Lanjutan)

Simbol	Deskripsi
Objek 	menyatakan objek yang berinteraksi pesan
Waktu aktif 	menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan di dalamnya, misalnya :  maka cekStatusLogin() dan open() dilakukan didalam metode login() Aktor tidak memiliki waktu aktif
Pesan tipe create 	menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat
Pesan tipe call 	menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri,  arah panah mengarah pada objek yang memiliki operasi/metode, karena ini memanggil operasi/metode maka operasi/metode yang dipanggil harus ada pada diagram kelas sesuai dengan kelas objek yang berinteraksi

TABEL: 2.4. Simbol-simbol sequence diagram (Lanjutan)

Simbol	Deskripsi
Pesan tipe send 	menyatakan bahwa suatu objek mengirimkan data/masukan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi
Pesan tipe return 	menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian
Pesan tipe destroy 	menyatakan suatu objek mengakhiri hidup objek lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada create maka ada destroy

Penomoran pesan berdasarkan urutan interaksi pesan. Penggambaran letak pesan harus berurutan, pesan yang lebih atas dari lainnya adalah pesan yang berjalan terlebih dahulu. Semua metode di dalam kelas harus ada di dalam diagram kolaborasi atau sekuen, jika tidak ada berarti perancangan metode di dalam kelas itu kurang baik. Hal ini dikarenakan ada metode yang tidak dapat dipertanggungjawabkan kegunaannya.

2.6.5. Class Diagram

Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas. Sedangkan operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

Diagram kelas dibuat agar pembuat program atau programmer membuat kelas-kelas sesuai rancangan di dalam diagram kelas agar antara dokumentasi perancangan dan perangkat lunak sinkron. Banyak berbagai kasus, perancangan kelas yang dibuat tidak sesuai dengan kelas-kelas yang dibuat pada perangkat lunak, sehingga tidaklah ada gunanya lagi sebuah perancangan karena apa yang dirancang dan hasil jadinya tidak sesuai.

Kelas-kelas yang ada pada struktur sistem harus dapat melakukan fungsi-fungsi sesuai dengan kebutuhan sistem sehingga pembuat perangkat lunak atau programmer dapat membuat kelas-kelas di dalam program perangkat lunak sesuai dengan perancangan diagram kelas. Susunan struktur kelas yang baik pada diagram kelas sebaiknya memiliki jenis-jenis kelas berikut:

1. Kelas main

   Kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.

2. Kelas yang menangani tampilan (*view*)

   Kelas yang mendefinisikan dan mengatur tampilan ke pemakai.

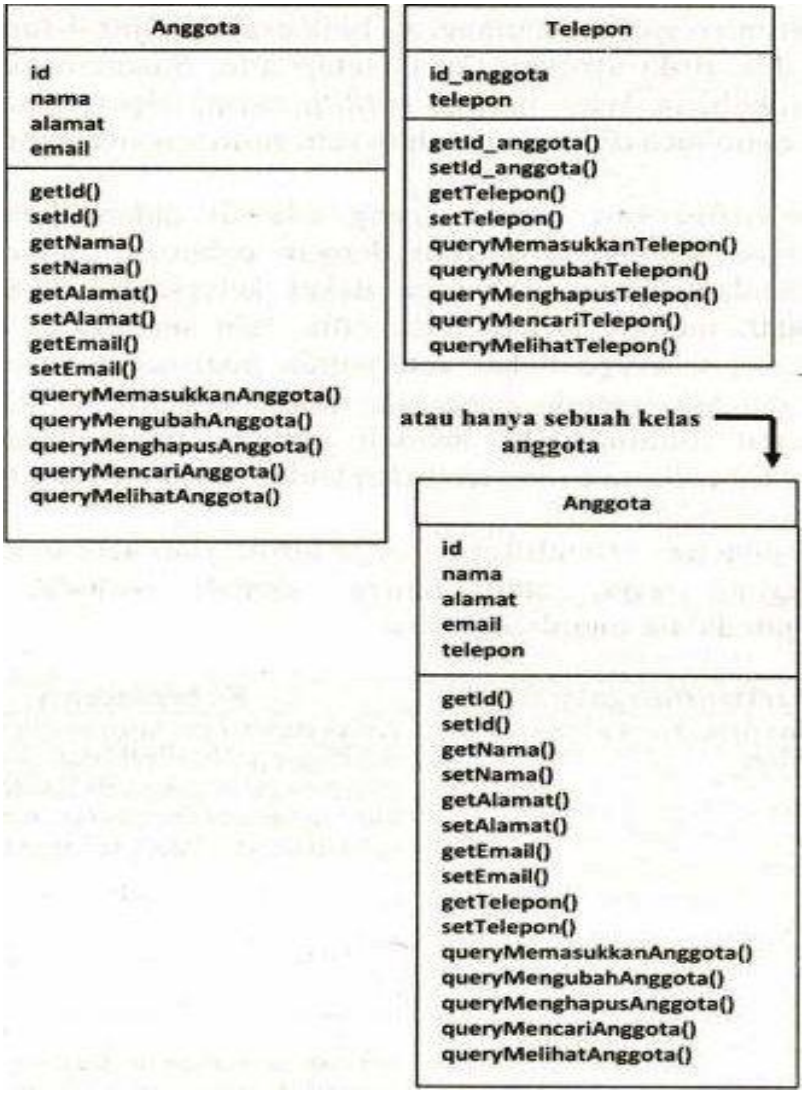
3. Kelas yang diambil dari pendefinisian *use case* (*controller*)

   Kelas yang menangani fungsi-fungsi yang harus ada diambil dari pendefinisian use case, kelas ini biasanya disebut dengan kelas proses yang menangani proses bisnis pada perangkat lunak.

4. Kelas yang diambil dari pendefinisian data (*model*)

   Kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data. Semua tabel yang dibuat di basis data dapat dijadikan kelas, namun untuk tabel dari hasil relasi

atau atribut *multivalue* pada ERD dapat dijadikan kelas tersendiri dapat juga tidak asalkan pengaksesannya dapat dipertanggungjawabkan atau tetap ada dalam perancangan kelas. Misalkan dalam tabel TTelepon dan TAnggota pada studi kasus maka perancangan kelas dapat mengandung kelas Telepon dan Anggota dimana di dalamnya ada sebuah atribut berupa larik (*array*) bertipe *string* dengan nama telepon. Sebagai ilustrasinya dapat dilihat pada Gambar.2.6.



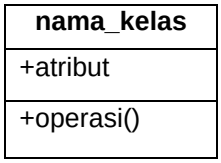
GAMBAR: 2.6. Perancangan kelas data untuk tabel dan atribut *multivalue* (Rosa dan Shalahuddin, 2019:143)

Hal terpenting adalah kolom telepon tetap dapat diakses dari perancangan kelas. Kelas data biasanya adalah kelas yang terkait dengan pengaksesan tabel pada basis data sesuai dengan nama kelasnya, misalnya kelas Anggota maka akan digunakan untuk mengakses tabel yang menyimpan data anggota.

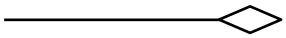
Dalam mendefinisikan metode yang ada di dalam kelas perlu memperhatikan apa yang disebut dengan *cohesion* dan *coupling*. *Cohesion* adalah ukuran seberapa dekat keterkaitan instruksi di dalam sebuah metode terkait satu sama lain sedangkan *coupling* adalah ukuran seberapa dekat keterkaitan instruksi antara metode yang satu dengan metode yang lain dalam sebuah kelas. Sebagai aturan secara umum maka sebuah metode yang dibuat harus memiliki kadar *cohesion* yang kuat dan kadar *coupling* yang lemah.

Adapun untuk simbol-simbol diagram kelas terdapat pada Tabel 2.5 dibawah ini.

TABEL: 2.5. Simbol-simbol class diagram (Rosa dan Shalahuddin, 2019:146)

Simbol	Deskripsi
Kelas 	kelas pada struktur sistem
Antarmuka / *interface*  **Nama_interface**	sama dengan konsep *interface* dalam pemrograman berorientasi objek
Asosiasi / *association* 	relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*

TABEL: 2.5. Simbol-simbol class diagram (Lanjutan)

Simbol	Deskripsi
Asosiasi berarah / *directed association* 	relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*
generalisasi 	relasi antar kelas dengan makna generalisasi-spesialisasi (umum-khusus)
kebergantungan / *dependency* 	relasi antar kelas dengan makna kebergantungan antar kelas
agregasi / *aggregation* 	relasi antar kelas dengan makna semua-bagian (*whole-part*)

2.7. *Entity Relationship Diagram* (ERD)

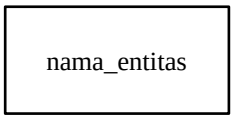
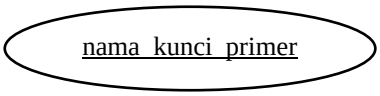
Menurut Rosa dan Shalahuddin (2019) Sistem basis data adalah sistem terkomputerisasi yang tujuan utamanya adalah memelihara data yang sudah diolah atau informasi dan membuat informasi tersedia saat dibutuhkan. Pada intinya basis data adalah media untuk menyimpan data agar dapat diakses dengan mudah dan cepat. Sistem informasi tidak dapat dipisahkan dengan kebutuhan akan basis data apapun bentuknya, bisa berupa file teks ataupun *Database Management System* (DBMS).

Kebutuhan basis data dalam sistem informasi meliputi memasukkan, menyimpan, dan mengambil data serta membuat laporan berdasarkan data yang telah tersimpan. Tujuan dari dibuatnya tabel-tabel adalah untuk menyimpan data ke

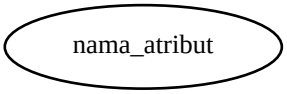
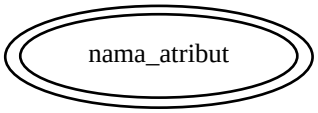
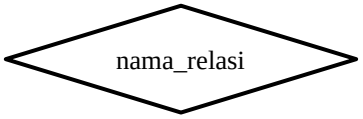
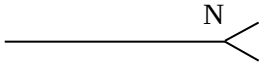
dalam tabel-tabel agar mudah diakses. Oleh karena itu untuk merancang tabel-tabel yang akan dibuat maka dibutuhkan pola pikir penyimpanan data nantinya jika dalam bentuk baris-baris data (*record*) dimana setiap baris terdiri dari beberapa kolom.

Entity Relationship Diagram (ERD) adalah pemodelan awal basis data yang paling banyak digunakan dan dikembangkan berdasarkan teori himpunan dalam bidang matematik. ERD digunakan untuk pemodelan basis data relasional. Sehingga jika penyimpanan basis data menggunakan *Object Oriented* DBMS (OODBMS) maka perancangan tidak perlu menggunakan ERD. ERD memiliki beberapa aliran notasi seperti notasi Chen (dikembangkan oleh Peter Chen), Barker (dikembangkan oleh Richard Barker, Ian Palmer, Harry Ellis), notasi Crow's Foot, dan beberapa notasi lain. Namun yang banyak digunakan adalah notasi dari Chen, Berikut adalah simbol-simbol yang digunakan pada ERD dengan notasi Chen dapat dilihat pada Tabel 2.6.

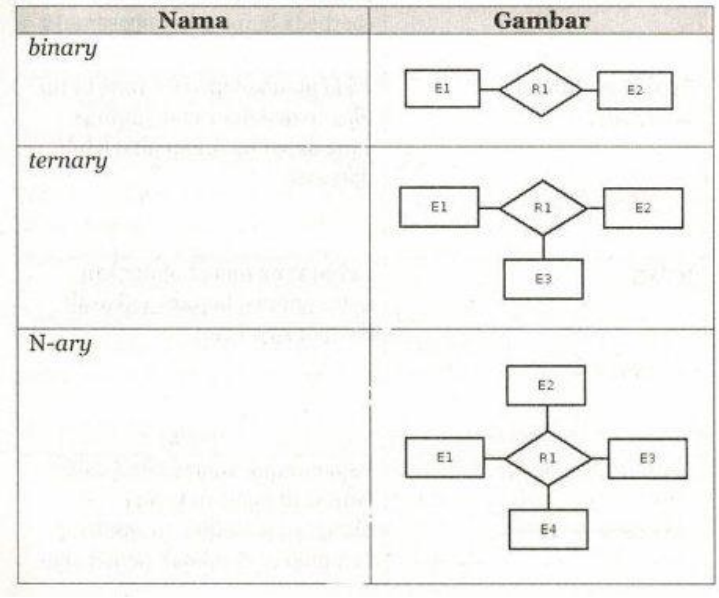
TABEL: 2.6. Simbol-simbol ERD notasi Chen (Rosa dan Shalahuddin, 2019:50)

Simbol	Deskripsi
Entitas/ *entity* 	Entitas merupakan data inti yang akan disimpan; bakal tabel pada basis data; benda yang memiliki data dan harus disimpan datanya agar dapat diakses oleh aplikasi komputer; penamaan entitas biasanya lebih ke kata benda dan belum merupakan nama tabel
Atribut kunci primer 	*Field* atau kolom data yang butuh disimpan dalam suatu entitas dan digunakan sebagai kunci akses *record* yang diinginkan; biasanya berupa id; kunci primer dapat lebih dari satu kolom, asalkan kombinasi dari beberapa kolom tersebut dapat bersifat unik (berbeda tanpa ada yang sama)

TABEL: 2.6. Simbol-simbol ERD notasi Chen (Lanjutan)

Simbol	Deskripsi
Atribut 	*Field* atau kolom data yang butuh disimpan dalam suatu entitas
Atribut multinilai/ *multivalue* 	*Field* atau kolom data yang butuh disimpan dalam suatu entitas yang dapat memiliki nilai lebih dari satu
Relasi 	Relasi yang menghubungkan antar entitas; biasanya diawali dengan kata kerja
Asosiasi/ *association* 	Penghubung antara relasi dan entitas dimana di kedua ujungnya memiliki *multiplicity* kemungkinan jumlah pemakaian. Kemungkinan jumlah maksimum keterhubungan antara entitas satu dengan entitas yang lain disebut dengan kardinalitas. Misalkan ada kardinalitas 1 ke N atau sering disebut dengan one to many menghubungkan entitas A dan entitas B.

ERD biasanya memiliki hubungan *binary* (satu relasi menghubungkan dua buah entitas). Beberapa metode perancangan ERD menoleransi hubungan relasi *ternary* (satu relasi menghubungkan tiga buah relasi) atau *N-ary* (satu relasi menghubungkan banyak entitas), tapi banyak metode perancangan ERD tidak mengizinkan hubungan *ternary* atau *N-ary*. Contoh dari bentuk hubungan relasi dalam ERD dapat dilihat pada Gambar.2.7 dibawah ini.



GAMBAR: 2.7. Bentuk hubungan relasi ERD (Rosa dan Shalahuddin, 2019:52)

2.8. Website

Website merupakan halaman-halaman situs dalam sebuah domain atau subdomain yang berada pada WWW (*Word Wide Web*). Halaman web merupakan dokumen yang ditulis dalam format HTML (*Hyper Text Markup Language*) yang dapat diakses melalui HTTP (Hypertext Transfer Protocol), yaitu sebuah protokol yang menyampaikan informasi dari server website untuk ditampilkan melalui web browser melalui jaringan internet.

Menurut Hidayatullah dan Kawistara (2017) WWW (*Word Wide Web*) adalah suatu program yang ditemukan oleh Tim Berners-Lee pada tahun 1991. Awalnya Berners-Lee ingin menemukan cara unruk menyusun arsip-arsip risetnya. Untuk itu beliau mengembangkan suatu sistem untuk keperluan pribadi. Sistem itu adalah program peranti lunak yang diberi nama Enquire. Dengan program itu Berners-Lee berhasil menciptakan jaringan yang menautkan berbagai arsip

sehingga memudahkan pencarian informasi yang dibutuhkan. Inilah yang kelak menjadi dasar dari sebuah perkembangan pesat yang dikenal sebagai WWW. Pada tahun 1989 Berners-Lee membuat pengajuan untuk proyek pembuatan hipertaks global, kemudian pada bulan Oktober 1990, *'Waring Wera Wanua'* sudah dapat dijalankan dilingkungan CERN (Pusat Penelitian Fisika Partikel Eropa). Pada musim panas tahun 1991, WWW secara resmi digunakan secara luas pada jaringan internet.

WWW (*Word Wide Web*) bekerja berdasarkan pada tiga mekanisme, sebagai berikut:

1. Informasi disimpan di dalam dokumen yang sering kita sebut *we*b. Halaman *web* adalah *file-file* yang disimpan dalam computer. Komputer tersebut dikenal dengan istilah *web server*.
2. Komputer yang menakses isi dari halaman *web* disebut dengan *web clients*.
3. *Web clients* menampilkan halaman *web* dengan program yang dikenal dengan nama *Web browser* seperti Chrome, Firefox dan Internet Explorer.

2.9. *Black Box Testing*

Pada sebuah perangkat lunak sering mengandung kesalahan (*error*) untuk proses-proses tertentu pada saat perangkat lunak sudah berada atau sudah diserahkan kepada *user*. Kesalahan-kesalahan (error) pada perangkat lunak ini sering disebut "*bug*". Untuk menghindari banyaknya *bug* maka diperlukan adanya pengujian perangkat lunak sebelum perangkat lunak diberikan ke *user* atau selama perangkat lunak masih dikembangkan (Rosa A dan Shalahuddin, 2019).

Adanya bug adalah suatu yang biasa karena buatan manusia tidak aka ada yang sempurna. Bahkan pada sebuah perangkat lunak yang sudah besar dan terkenal pun biasanya masih ada *bug*. Untuk meminimalisir *bug* maka pengembang perangkat lunak harus melakukan pengujian. Kelakuan perangkat lunak yang tidak sesuai dengan spesifikasi yang dibutuhkan bisa dianggap sebagai *bug*. Pengujian diperlukan tidak hanya untuk meminimalisasi kesalahan secara teknis tapi juga kesalahan non teknis (misalnya pengujian pesan kesalahan sehingga user tidak bingung atau tidak mengerti dengan pesan kesalahan yang muncul, atau juga jika masukan dan keluaran yang diperlukan berkapasitas sangat besar).

Pengujian adalah satu set aktifitas yang direncanakan dan sistematis untuk menguji atau mengevaluasi kebenaran yang diinginkan. Aktifitas pengujian terdiri dari satu set atau sekumpulan langkah dimana dapat menempatkan desain kasus uji yang spesifik dan metode pengujian. Secara umum pola pengujian pada perangkat lunak adalah sebagai berikut:

1. Pengujian dimulai dari level komponen hingga integrasi antar komponen menjadi sebuah sistem.
2. Teknik pengujian berbeda-beda sesuai dengan berbagai sisi atau unit uji dalam waktu yang berbeda-beda pula bergantung pada pengujian pada bagian mana yang dibutuhkan.
3. Pengujian dilakukan oleh pengembang perangkat lunak, dan jika untuk proyek besar pengujian bisa dilakukan oleh tim uji yang terkait dengan tim pengembang perangkat lunak (*independent test group* (ITG)).
4. Pengujian dan penirkutuan (*debugging*) merupakan aktifitas berbeda, tetapi

harus diakomodasi pada bergai strategi pengujian. Pengujian lebih focus untuk mencari adanya kesalahan (*error*) baik dari sudut pandang orang secara umum atau dari sudut pandang pengembang tanpa harus menemukan lokasi kesalahan pada kode program. Penirkutuan (*debugging*) adalah proses mencari lokasi kesalahan (*error*) pada kode program sehingga dapat segera diperbaiki oleh pembuat program (*programmer*).

Salah satu metode atau pendekatan pengujian yang sering digunakan yaitu dengan *Black-Box Testing* (pengujian kotak hitam). *Black-Box Testing* adalah merupakan pengujian perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dank ode program. Pengujian dimaksudkan untuk mengetahui apakah fungsi-fungsi, masukan (*input*), dan keluaran (*output*) dari perangkat lunak sesuai dengan spesifaki yang dibutuhkan.

Pengujian *Black-Box* dilakukan dengan membuat kasus uji yang bersifat mencoba semua fungsi dengan memakai perangkat lunak apakah sesuai dengan spesifikasi yang dibutuhkan. Kasus uji yang dibuat untuk melakukan pengujian *Black-Box* harus dibuat dengan kasus benar dan kasus salah, misalkan untuk kasus login maka kasus uji yang dibuat adalah:

1. Jika *user* memasukkan nama pemakai (*username*) data kanta sandi (*password*) yang benar.
2. Jika user memasukkan nama pemakai (*username*) dan kata sandi (*password*) yang salah, misalnya nama pemakai benar tapi kata sandi salah, atau sebaliknya, atau keduanya salah (Rosa A dan Shalahuddin, 2019).

BAB III

ANALISA MASALAH DAN PERANCANGAN PROGRAM

Pada bab ini akan diuraikan metode pengumpulan data, studi literatur dan tahapan perancangan program sesuai dengan metode pengembangan yang digunakan yaitu SDLC (*System Development Life Cycle*) model *Waterfall* (Rosa A dan Shalahuddin, 2019). Pada penelitian ini dibatasi hanya menggunakan 4 (empat) tahapan yang terdiri dari: analisis, desain (perancangan program), pengodean (implementasi) dan pengujian (uji coba).

3.1. Pengumpulan Data

3.1.1. Metode Pengumpulan Data

Pengumpulan data bertujuan untuk mendapatkan data-data dan faktor-faktor yang mempengaruhi ketepatan atau keakuratan hasil klasifikasi penduduk miskin guna keberhasilan penelitian. Pada penelitian ini dilakukan dengan cara observasi lansung dan diskusi langsung dengan narasumber yang berkaitan langsung dengan penelitian ini. Diskusi yang dilaksanakan dengan narasumber yang berkaitan langsung dengan penelitian ini adalah untuk mendefinisikan factor-faktor apa saja sebagai variable yang mempengaruhi hasil klasifikasi penduduk miskin berikut data-data yang diperlukan. Selain itu juga dilakukan pengumpulan data-data penduduk wilayah Desa Taraju Kecamatan Taraju Kabupaten Tasikmalaya yang berada di kantor desa baik *hardcopy* maupun *softcopy*.

3.1.2. Hasil Analisis Pengumpulan Data

Hasil dari observasi dan diskusi langsung akan mendapatkan data dan factor-faktor atau variable yang diperlukan yang mempengaruhi ketepatan atau keakuratan hasil klasifikasi penduduk miskin guna keberhasilan penelitian ini.

Penulis menyimpukan populasi dalam penelitian ini adalah Penduduk Desa Taraju yang akan diklasifikasi kriteria atau kelas penduduk miskinnya. Adapun sampel data yang digunakan adalah data penduduk beberapa RT dari data penduduk 2015 dan data-data hasil observasi langsung atau wawancara dari Kepala Desa, Ketua RT maupun Perangkat Desa lainnya.

Dalam penelitian ini pengumpulan data terbagi dua yaitu data primer dan sekunder. Data primer diperoleh dengan cara melakukan observasi langsung dan diskusi dengan narasumber yang berkaitan dengan penelitian ini, yaitu Kepala Desa dan Perangkat Desa lainnya, untuk mendefinisikan factor-faktor yang mempengaruhi hasil klasifikasi penduduk yang nantinya dalam penelitian ini akan diklasifikasi sebagai penduduk miskin. Adapun data factor-faktor yang mempengaruhi hasil kalsifikasi penduduk miskin antara lain jenis kelamin, pendidikan terakhir, pekerjaan, status kawin, tanggungan anak masih sekolah, lantai rumah, dinding rumah, daya listrik dan sumber air.

Data sekunder adalah data hasil pengamatan dan obervasi langsung dari perangkat terkait yang dapat menunjang variabel-variabel penelitian yang dilakukan. Beberapa data penduduk sudah ada dalam buku Induk Penduduk Desa tahun 2015. Data sekunder yang digunakan dalam penelitian ini meliputi data faktor-faktor klasifikasi pada setiap penduduk yang sudah dijelaskan sebelumnya.

3.1.2.1. Definisi Variabel

Untuk mempermudah dalam pengumpulan data, baik data primer maupun data sekunder pada objek penelitian ini digunakan definisi variabel, yaitu merupakan proses pendefinisian variabel sehingga menjadi faktor-faktor yang dapat diukur dan ditentukan indikatornya. Variabel dan definisi data pada penelitian ini terdapat pada Tabel. 3.1.

TABEL: 3.1.Variabel, Definisi dan Indikator Data

No,	Variabel	Definisi	Indikator
1.	Jenis Kelamin	Perbedaan pada setiap penduduk, terbagi dalam jenis kelamin Laki-laki dan Perempuan	Laki-laki atau Perempuan
2.	Pendidikan	Pendidikan terakhir Kepala Keluarga	Perguruan Tinggi, SLTA,SLTP atau SD
3.	Pekerjaan	Jenis Pekerjaan Kepala Keluarga	Pegawai/Karyawan, Dagang, Tani, Buruh, atau Tidak Bekerja
4.	Status Perkawinan	Status Perkawinan	Kawin, Duda/ Janda
5.	Tanggungan Anak	Anak masih sekolah yang menjadi tanggungan keluarga	Jumlah anak masih sekolah
6.	Lantai Rumah	Bahan yang digunakan untuk lantai rumah	Keramik, Tembok, Kayu/bambu
7.	Dinding Rumah	Bahan yang digunakan untuk dinding rumah	Tembok, Semi-tembok, atau kayu/bambu
8.	Daya Listrik	Ukuran standar daya PLN yang terpasang	≥ 900 VA atau 450 VA
9.	Sumber Air	Sumber air untuk keperluan air minum, memasak dan MCK	PAM/berbayar, sumur atau mata air/sungai

3.1.2.2. Identifikasi Variabel

Identifikasi variabel penelitian ini dilaksanakan melalui proses studi literatur pada penelitian sebelumnya sebagai dasar untuk memperoleh variabel yang relevan untuk dianalisa dan dimodelkan. Dalam penelitian ini dilakukan identifikasi variabel yang akan diperlukan, terdiri dari dua jenis variabel yaitu varabel untuk

masukan (*input*) dan variabel untuk keluaran (*output*).

Variabel masukan (*input*) merupakan variabel yang akan dimasukkan pada sistem untuk diproses dan mendapatkan hasil yang diperlukan. Dalam penelitian ini, variabel yang dimaksud adalah data-data penduduk yang diharapkan mempengaruhi hasil klasifikasi penduduk miskin. Variabel yang sudah didefinisikan sebelumnya pada Tabel 3.1 kemudian dikonversi menjadi data numerik dengan tujuan supaya dapat diolah untuk kebutuhan penelitian ini. Dari data yang sudah ada beberapa merupakan variabel yang sudah bersifat numerik sehingga tidak perlu dilakukan konversi data.

Adapun variabel yang tidak memerlukan konversi yaitu Variabel Tanggungan Anak, karena merupakan jenis variabel numerik dan memiliki besaran angka yang jelas. Sedangkan ada delapan variabel yang perlu dilakukan proses konversi dulu sebagai variabel masukan yaitu Jenis Kelamin, Pendidikan, Pekerjaan, Status Perkawinan, Lantai Rumah, Dinding Rumah, Daya Listrik dan Sumber Air yang terdapat pada Tabel 3.2 di bawah ini.

TABEL: 3.2. Variabel Masukan dan Proses Konversi

No.	**Nama Variabel**	**Konversi Data Numerik**				
		1	**2**	**3**	**4**	**5**
1.	Jenis Kelamin	Laki-laki	Perempuan	-	-	-
2.	Pendidikan	Perguruan Tinggi	SLTA	SLTP	SD	-
3.	Pekerjaan	Pegawai/ Karyawan	Dagang	Tani	Buruh	Tidak Bekerja
4.	Status Perkawinan	Kawin	Duda	Janda	-	-
5.	Lantai Rumah	Keramik	Tembok	Kayu/ Bambu	-	-
6.	Dinding Rumah	Tembok	Semi-Tembok	Kayu/ Bambu	-	-
7.	Daya Listrik	≥ 900 VA	450 VA	-	-	-
8.	Sumber Air	PAM/ berbayar	Sumur	Mata air/sungai	-	-

Pada Tabel 3.2 terdapat delapan variabel masukan yang perlu dilakukan proses konversi menjadi data numerik. Variabel Jenis Kelamin yang memiliki data berupa Laki-laki dan Perempuan dikonversi menjadi data numerik nilai 1 untuk Laki-laki dan nilai 2 untuk Perempuan. Variabel Pendidikan dikategorikan menjadi empat (4) kategori yaitu nilai 1 untuk Perguruan Tinggi, nilai 2 untuk SLTA, nilai 3 untuk SLTP dan nilai 4 untuk SD. Begitupun dengan Variabel-variabel yang lainnya, antara lain Variabel Pekerjaan dikategorikan menjadi lima kategori, Variabel Status Perkawinan dikategorikan menjadi tiga kategori, Variabel Lantai Rumah dikategorikan menjadi tiga kategori, Variabel Dinding Rumah dikategorikan menjadi tiga kategori dan Variabel Daya Listrik dikategorikan menjadi dua kategori serta Variabel Sumber Air dikategorikan menjadi tiga kategori.

Variabel keluaran (*output*) merupakan variabel yang dihasilkan oleh sistem dari proses pengolahan variabel masukan. Besarnya perubahan pada variabel ini tergantung dari besaran variabel masukan. Dalam penelitian ini variabel keluaran yang dimaksud adalah klasifikasi penduduk yang termasuk pada kriteria Penduduk miskin dan Penduduk tidak miskin. Karena variabel keluaran tersebut bukan data numerik jadi perlu dilakukan konveri dulu. Berikut Tabel 3.3 menyajikan variabel keluaran dan proses konversi.

TABEL: 3.3.Variabel Keluaran dan Proses Konversi

No.	Nama Variabel	Konversi Data Numerik	
		1	-1
1.	Klasifikasi Penduduk	Penduduk Miskin	Penduduk tidak Miskin

Sebagai contoh data yang sudah dikumpulkan disajikan pada Tabel 3.4 merupakan data penduduk sebelum dilakukan konversi dan Tabel 3.5 data penduduk yang sudah dilakukan konversi.

TABEL: 3.4. Contoh data penduduk sebelum dilakukan konversi

No	NIK	Jenis Kelamin	Pendidikan Terakhir	Pekerjaan	Status Kawin	Tanggungan Anak Sekolah	Lantai Rumah	Dinding Rumah	Daya Listrik (VA)	Sumber Air	Kelas
1	320613071 XXXXXXX	Laki-laki	SLTP	Tani	Kawin	0	Keramik	Tembok	900	PAM	Tidak Miskin
2	320613810 XXXXXXX	Laki-laki	SLTA	Buruh	Kawin	2	Kayu	Bambu	450	Mata air/ sungai	Miskin
3	320613550 XXXXXXX	Perempuan	SD	Tidak Bekerja	Janda	0	Kayu	Bambu	450	Mata air/ sungai	Miskin
4	320613240 XXXXXXX	Laki-laki	SLTP	Buruh	Kawin	2	Kayu	Bambu	450	Mata air/ sungai	Miskin
5	320613121 XXXXXXX	Laki-laki	SLTA	Dagang	Kawin	2	Keramik	Tembok	450	PAM	Tidak Miskin
6	320613010 XXXXXXX	Laki-laki	SLTA	Dagang	Kawin	2	Keramik	Tembok	900	PAM	Tidak Miskin
7	320613471 XXXXXXX	Perempuan	SD	Tidak Bekerja	Janda	0	Kayu	Bambu	450	Mata air/ sungai	Miskin
8	320613611 XXXXXXX	Perempuan	SD	Tidak Bekerja	Janda	0	Kayu	Bambu	450	Mata air/ sungai	Miskin
9	320613070 XXXXXXX	Laki-laki	SLTP	Dagang	Kawin	2	Keramik	Tembok	450	PAM	Tidak Miskin
10	320613081 XXXXXXX	Laki-laki	SLTA	Dagang	Kawin	2	Kayu	Bambu	450	Mata air/ sungai	Miskin

TABEL: 3.5. Contoh data penduduk setelah dilakukan konversi

No	NIK	Jenis Kelamin	Pendidikan Terakhir	Pekerjaan	Status Kawin	Tanggungan Anak Sekolah	Lantai Rumah	Dinding Rumah	Daya Listrik (VA)	Sumber Air	Kelas
1	320613071 XXXXXXX	1	3	3	1	0	1	1	1	1	-1
2	320613810 XXXXXXX	1	2	4	1	2	3	3	2	3	1
3	320613550 XXXXXXX	2	4	5	3	0	3	3	2	3	1
4	320613240 XXXXXXX	1	3	4	1	2	3	3	2	3	1
5	320613121 XXXXXXX	1	2	2	1	2	1	1	2	1	-1
6	320613010 XXXXXXX	1	2	2	1	2	1	1	1	1	-1
7	320613471 XXXXXXX	2	4	5	3	0	3	3	2	3	1
8	320613611 XXXXXXX	2	4	5	3	0	3	3	2	3	1
9	320613070 XXXXXXX	1	3	2	1	2	1	1	2	1	-1
10	320613081 XXXXXXX	1	2	2	1	2	3	3	2	3	1

3.2. Studi Literatur

Pada Studi Literatur ini penulis mengumpulkan bahan-bahan materi dari berbagai sumber misalnya Buku, Jurnal, artikel, dokumen penelitian-penelitian terdahulu dan sumber lainnya yang berkaitan dengan penelitian ini. Kemudian menghimpun dan menganalisanya. Dari bahan-bahan yang telah dipelajari penulis mendapat gambaran dan pengetahuan tentang landasan teori serta metode yang digunakan dalam penelitian ini. Berikut adalah daftar jurnal yang penulis gunakan sebagai referensi penelitian ini.

TABEL: 3.6. Daftar Jurnal Referensi

No.	Literatur	Pembahasan
1	Raudlatul Munawarah, Oni Soesanto2 dan M. Reza Faisal "Penerapan Metode Support Vector Machine Pada Diagnosa Hepatitis" Vol. 4, Februari 2016.	Penelitian untuk mendiagnosa penyakit Hepatisis dengan menganalisa dataset medis. Untuk menemukan fungsi pemisah optimal pada dua set data dari dua kelas berbeda digunakan metoda SVM (Support Vektor Machine) . Hasil dari penelitian/ uji coba didapat persentasi benar antara 68 – 83% dan fungsi kernel RBF 70 - 96% . Metode SVM dapat digunakan untuk mendiagnosa penyakit hepatitis dengan tingkat akurasi cukup tinggi dan fungsi kernel RBF memiliki tingkat akurasi cenderung lebih tinggi dibandingkan fungsi kernel linier.
2	Ana Mariyam Puspitasari, Dian Eka Ratnawati dan Agus Wahyu Widodo "Klasifikasi Penyakit Gigi Dan Mulut Menggunakan Metode Support Vector Machine" Vol. 2, Februari 2018	Penelitian untuk mendiagnosa awal terhadap penyakit gigi dan mulut. Dalam penelitian ini sistem klasifikasi yang digunakan yakni menggunakan metode SVM, karena metode SVM dapat mengatasi masalah klasifikasi dan regresi dengan linear maupun non-linear sehingga dapat menjadi suatu kemampuan algoritma pembelajaran pada klasifikasi ataupun regresi. Hasil klasifikasi yang diperoleh dengan menggunakan metode SVM mempunyai rata – rata nilai akurasi sebesar 94.442% . Penelitian ini dapat diterapkan untuk membantu melakukan klasifikasi penyakit gigi dan mulut dengan metode support vector machine.

TABEL: 3.6. Daftar Jurnal Referensi (Lanjutan)

No.	Literatur	Pembahasan
3	Indri Monika Parapat, Muhammad Tanzil Furqon dan Sutrisno " Penerapan Metode Support Vector Machine (SVM) Pada Klasifikasi Penyimpangan Tumbuh Kembang Anak " Vol. 2, Oktober 2018	Penelitian untuk klasifikasi penyimpangan tumbuh kembang anak. Bertujuan mengetahui penyimpangan tumbuh kembang anak sedini mungkin. Terdiri dari 3 kelas/ jenis penyimpangan tumbuh kembang anak yaitu Down Syndrome, Autisme, dan Attention Deficit Hyperactivity Disorder (ADHD). Algoritma SVM merupakan metode klasifikasi linier, sehingga menggunakan kernel untuk mengatasi data yang bersifat nonlinier. Hasil akhir dari penilitan ini menghasilkan rata-rata akurasi tertinggi sebesar 63,11% $\lambda = 10$, C = 1, itermax = 200 dan juga menggunakan kernel polynomial. Perbandingan dari hasil klasifikasi kembang anak dengan bantuan psikolog menunjukkan bahwa sistem menghasilkan akurasi yang kurang baik. Hal ini dapat disebabkan oleh sedikit dan tidak seimbangnya data yang digunakan untuk penelitian.
4	Alven Safik Ritonga, Endah Supeni Purwaningsih " Penerapan Metode Support Vector Machine (SVM) dalam Klasifikasi Kualitas Pengelasan SMAW (Shield Metal Arc Welding) " Vol. 5, November 2018	Penelitian untuk klasifikasi kualitas pengelasan, Metode Support Vector Machine (SVM) karena merupkan metoda klasifikasi yang sangat baik dibandingkan metoda konvensional. Hasil pengujian model dengan menggunakan kernel fungsi quadratik menunjukkan hasil akurasi sebesar 96,2%, dan pengujian menggunakan data uji menunjukkan hasil akurasi sebesar 98% dengan menggunakan kernel fungsi quadratik.
5	Arif Pratama, Randy Cahya Wihandika dan Dian Eka Ratnawati "Implementasi Algoritme Support Vector Machine (SVM) untuk Prediksi Ketepatan Waktu Kelulusan Mahasiswa " Vol. 2, No. 4, April 2018	Penelitian untuk mengklasifikasikan dan prediksi ketepatan waktu kelulusan mahasiswa. Algoritme Support Vector Machine (SVM) mengklasifikasikan data menjadi 2 kelas menggunakan kernel Gaussian RBF dengan menggunakan data latih sebanyak 170 dataset. Penelitian ini menghasilkan rata-rata akurasi sebesar 80,55 %.

Dari studi literatur penulis dapat menyimpulkan bahwa penggunaan metode *Support Vektor Machine* (SVM) untuk klasifikasi 2 (dua) kelas hasilnya rata-rata menunjukkan akurasi yang sangat baik kecuali untuk klasifikasi 3 (tiga) kelas,

karena sedikit dan kurang seimbangnya data yang digunakan. Karena pada penelitian ini klasifikasi 2 (dua) kelas, maka metode *Support Vektor Machine* (SVM) ini dapat digunakan dalam penelitian ini.

3.3. Analisis

Sesuai dengan tahapan metode pengembangan sistem yang digunakan yaitu SDLC (*System Development Life Cycle*) model *Waterfall* (Rosa A dan Shalahuddin, 2019). Maka akan dilakukan tahapan analisis, seperti analisa permasalahan, gambaran umum sistem yang akan dibuat, analisis perhitungan *metode Support Vektor Machine* (SVM) dan analisis kebutuhan perangkat keras juga perangkat lunak.

3.3.1. Analisa Permasalahan

Dalam kecerdasan buatan (*artificial intelligence*) terdapat cabang disiplin ilmu, salah satunya yaitu *machine learning* yang mencakup perancangan dan pengembanga algoritma pembelajaran. *Machine learning* sudah banyak manfaatnya dalam berbagai aplikasi, salah satunya aplikasi berbasis data. Dalam penerapannya untuk mengolah data penduduk, *machine learning* dapat digunakan untuk mempelajari pola data dari beberapa variabel penduduk guna mengklasifikasikan penduduk yang masuk kriteria penduduk miskin.

Berdasarkan latar belakang dan tujuan penelitian yang sudah diuraikan maka dalam penelitian ini akan dibuat sebuah aplikasi untuk melalukan klasifikasi penduduk miskin di wilayah Desa Taraju Kapupaten Tasikmalaya menggunakan *machine learning* dengan metode *support vector machine* (SVM). Dengan harapan

penelitian ini dapat membantu pemerintahan desa dalam menentukan atau mengetahui data penduduk miskin secara cepat dan tepat di wilayahnya, guna pemerataan kesejahteraan masyarakatnya dan program pemerintah dalam mengentaskan kemiskinan.

Aplikasi yang akan dibuat dalam penelitian ini dapat melakukan klasifikasi penduduk dengan kriteria miskin menggunakan metode *support vector machine* (SVM) sengan bahasa pemograman *PHP* dan berbasis *Web*.

3.3.2. Gambaran Umum Sistem

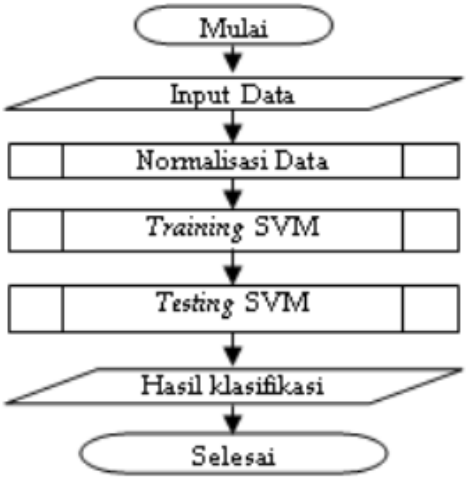
Aplikasi dari ilmu kecerdasan buatan (*Artificial Intelligence*) salah satunya adalah *Machine Learning* (Pembelajaran mesin), yang merupakan program komputer yang memiliki kemampuan belajar dengan mempelajari sejumlah data sebagai *train dataset* menggunakan algoritma khusus (*machine learning algoritms*), sehingga otomatis menjadi pintar dari pengalaman yang didapatkannya tanpa diprogram secara ekspisit (Nurhayati dkk, 2019). .

Dalam penelitian ini dibangun sebuah aplikasi dengan menerapkan *machine learning* untuk mengklasifikasi kriteria penduduk miskin di wilayah Desa Taraju Kabupaten Tasikmalaya dengan metode *support vektor machine* (SVM) . Adapun hasil yang diharapkan dari aplikasi ini adalah klasifikasi penduduk yang dikelompokan menjadi dua kelas, yaitu kelas **Penduduk Miskin** dan kelas **Penduduk Tidak Miskin**.

Proses klasifikasi pada sistem menggunakan sebagian data-data penduduk sesuai dengan data yang diperlukan untuk dijadikan data pembelajaran (*train dataset*), kemudian proses klasifikasi dilakukan dengan cara memasukan data-data

atau variabel-variabel yang sudah ditentukan pada form masukan sistem (*input*).

Berikut diagram alir sistem secara keseluruhan disajikan pada Gambar 3.1, yang mana proses dimulai dengan input data penduduk. Kemudian data tersebut dilakukan normalisasi data dengan konversi data dan menjadi data *train* untuk proses training SVM. Selanjutnya dilakukan Testing dengan SVM menggunakan data uji, sehingga didapatkan hasil klasifikasi (Pratama dkk, 2018).



GAMBAR: 3.1. Diagram Alir Sistem (Pratama dkk, 2018)

3.3.3. Analisis Perhitungan *Support Vektor Machine* (SVM)

Pada penelitian ini menggunakan enam data sampel untuk tahap analisis perhitungan *support vektor machine* (SVM) yang terbagi dalam dua tahap, yaitu tahap *training* untuk menemukan model dan tahap *testing* untuk melakukan pengujian.

Data *training* dihitung dengan menggunakan metode sekuantial yang merupakan salah satu metode penyelesaian data SVM. Hasil *training* adalah sebuah model atau pembelajaran yang tersimpan didalam sistem yang akan menjadi acuan

bagi sistem untuk menentukan miskin tidaknya suatu penduduk dari inputan data penduduk yang menjadi data *testing*. Adapun tahapan penyelesaian dilakukan seperti dibawah ini:

1. Menginisiasi awal untuk nilai α, C, ε, ϒ dan λ

   $\alpha = 0$

   $C = 1$

   $\varepsilon = 0.001$

   $\Upsilon = 0.5$

   $\lambda = 0.5$

2. Memasukkan data latih (*training*)

   Data latih menggunakan 5 (lima) data pada Tabel. 3.5 hasil dari observasi.

   TABEL: 3.7. Contoh Data Latih (*Training*)

No.	X1	X2	X3	X4	X5	X6	X7	X8	X9	Y
A1	1	3	3	1	0	1	1	1	1	-1
A2	1	2	4	1	2	3	3	2	3	1
A3	2	4	5	3	0	3	3	2	3	1
A4	1	3	4	1	2	3	3	2	3	1
A5	1	2	2	1	2	1	1	2	1	-1

   Dimana label 1 adalah label positif (penduduk miskin) dan -1 adalah label negatif (penduduk tidak miskin).

3. Menentukan dot product setiap data dengan memasukkan fungsi kernel (K). Fungsi kernel yang digunakan adalah fungsi kernel linier dengan rumus:

   K(x,y) = x.y , yang mana data harus di transpose dulu karena menggunakan perkalian matrik A x $A^{T.}$

TABEL: 3.8. Transpose Data

A1	A2	A3	A4	A5
1	1	2	1	1
3	2	4	3	2
3	4	5	4	2
1	1	3	1	1
0	2	0	2	2
1	3	3	3	1
1	3	3	3	1
1	2	2	2	2
1	3	3	3	1

Setiap data akan dibandingkan dengan dirinya dan data yang lainnya, karena pada metode kernel data tidak dipresentasikan secara individual melainkan lewat perbandingan antara sepasang data lainnya. Pada Tabel 3.9 merupakan perbandingan data dari data yang berjumlah 5 (lima) data.

TABEL: 3.9. Perbandingan Data

	A1	A2	A3	A4	A5
A1	K(A1,A1)	K(A1,A2)	K(A1,A3)	K(A1,A4)	K(A1,A5)
A2	K(A2,A1)	K(A2,A2)	K(A2,A3)	K(A2,A4)	K(A2,A5)
A3	K(A3,A1)	K(A3,A2)	K(A3,A3)	K(A3,A4)	K(A3,A5)
A4	K(A4,A1)	K(A4,A2)	K(A4,A3)	K(A4,A4)	K(A4,A5)
A5	K(A5,A1)	K(A5,A2)	K(A5,A3)	K(A5,A4)	K(A5,A5)

Berikut perhitungan perbandingan data A1 dan A1:

K(A1,A1) = ((1 x 1) + (3 x 3) + (3 x 3) + (1 x 1) + (0 x 0) + (1 x 1) + (1 x 1) + (1 x 1) + (1 x 1)) = 24.00

Semua data dihitung dengan cara yang sama, baris x kolom sehingga mendapatkan dot product seperti ditunjukkan pada Tabel 3.10. Karena data yang digunakan terdapat 5 data maka akan didapatkan matriks 5 x 5.

TABEL: 3.10. Hasil Perhitungan Kernel

	A1	**A2**	**A3**	**A4**	**A5**
A1	24.00	31.00	43.00	34.00	19.00
A2	31.00	57.00	64.00	59.00	31.00
A3	43.00	64.00	85.00	68.00	36.00
A4	34.00	59.00	68.00	62.00	33.00
A5	19.00	31.00	36.00	33.00	21.00

4. Menghitung matriks Hessian dengan persamaan berikut:

$$D_{ij} = y_i y_j \left(K\left(x_i, x_j\right) + \lambda^2\right)$$

Keterangan:
D_{ij} = elemen matriks Hessian ke-*ij*
x_i = data ke - *i*
x_j = data ke - *j*
y_i = kelas data ke - *i*
y_j = kelas data ke - *j*
$K\left(x_i, x_j\right)$ = fungsi kernel yang digunakan

Perhitungan untuk pasangan data A1 dengan A1:

D_{ij} = (-1) (-1) (24.00) + 0.5 ^ 2 = 24.25

Semua data dihitung dengan cara yang sama, baris x kolom sehingga mendapatkan hasil seperti ditunjukkan pada Tabel 3.11.

TABEL: 3.11. Hasil Perhitungan Matriks

	a1	**a2**	**a3**	**a4**	**a5**
a1	24.25	-31.25	-43.25	-34.25	19.25
a2	-31.25	57.25	64.25	59.25	-31.25
a3	-43.25	64.25	85.25	68.25	-36.25
a4	-34.25	59.25	68.25	62.25	-33.25
a5	19.25	-31.25	-36.25	-33.25	21.25

5. Mencari nilai E (error)

$$E_i = \sum_{j=1}^{i} \alpha_j D_{ij}$$

$$E_i = \sum_{j=1}^{i} (0{,}5\ .\ 24.25) + (0{,}5\ .\ (-31.25)) + (0{,}5\ .\ (-43.25)) + (0{,}5\ .\ (-34.25)) + (0{,}5\ .\ 19.25) = -32.625$$

Semua data dihitung dengan cara yang sama, sehingga mendapatkan nilai error seperti pada Tabel 3.12.

TABEL: 3.12. Hasil Perhitungan Nilai Error

E_1	-32,625
E_2	59,125
E_3	69,125
E_4	61,125
E_5	-30,125

6. Menghitung $\delta\alpha$ (delta alpha) :

$\delta\alpha_i = \min\{max[\gamma\ (1 - E_i), -\alpha_i], C - \alpha_I\}$

$\delta\alpha_1$ = min{max [0,5 (1 – (-32.625)), -0,5], 1-0,5}

$\delta\alpha_1$ = min{max (16.8125, -0.5), 0.5}

$\delta\alpha_1$ = min {16.8125, 0.5}

$\delta\alpha_1$ = 0.5

Maka didapatkan nilai $\delta\alpha_i$ seperti pada Tabel 3.13 sebagai berikut:

TABEL: 3.13. Hasil Perhitungan $\delta\alpha$ (delta alpha)

$\delta\alpha_1$	0.5
$\delta\alpha_2$	-0.5
$\delta\alpha_3$	-0.5
$\delta\alpha_4$	-0.5
$\delta\alpha_5$	0.5

Karena nilai maksimum $\delta\alpha$ adalah 0.5 dan lebih dari epsilon (0.001) maka iterasi berlanjut.

7. Menghitung nilai α baru :

$\alpha_i = \alpha_i + \delta\alpha_i$

α_i = 0.5 + 0.5 = 1, maka didapatkan nilai α sebagai berikut:

α_1 = 1

α_2 = 0

α_3 = 0

α_4 = 0

α_5 = 1

8. Mencari nilai *b* (bias) :

$$b = -\frac{1}{2}\,(w.x^{+} + w.x^{-})$$

Sebelumnya dihitung dulu nilai w :

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i$$

w_i^{+} adalah bobot dot product data dengan alpha terbesar di kelas positif

w_i^{-} adalah bobot dot product data dengan alpha terbesar di kelas negative

$w.x^+$ = (1 x -1 x 43.00) + (0 x 1 x 64.00) + (0 x 1 x 85.00) + (0 x 1 x 68.00) + (1 x -1 x 36.00) = - 79.00

$w.x^-$ = (1 x -1 x 24.00) + (0 x 1 x 31.00) + (0 x 1 x 43.00) + (0 x 1 x 34.00) + (1 x -1 x 19.00) = - 43.00

Maka nilai b :

$$b = -\frac{1}{2}(-79.00 + (-43.00))$$

= 61.00

Perhitungan diatas belum bisa digunakan untuk fungsi keputusan karena iterasi masih harus diteruskan.

9. Setelah mendapatkan nilai α, w dan *b* maka sebagai contoh dapat dilakukan pengujian dengan 1 (satu) data uji hasil dari observasi berikut :

TABEL: 3.14. Contoh Data Uji (*Testing*)

No.	**X1**	**X2**	**X3**	**X4**	**X5**	**X6**	**X7**	**X8**	**X9**	**Y**
A	1	2	2	1	2	3	3	2	3	?

Langkah pertama untuk menguji adalah menghitung *dot product* antara data uji dengan semua data latih dengan fungsi kernel.

K(x,y) = x.y , dimana **x** adalah data uji dan **y** adalah semua data latih.

$K(x_i, x)$ = (1 x 1) + (2 x 3) + (2 x 3) + (1 x 1) + (2 x 0) + (3 x 1) + (3 x 1) + (2 x 1) + (3 x 1) = 25.00

Begitu seterusnya untuk semua data latih maka didapatkan *dot product* data uji seperti pada Tabel 3.15 dibawah ini.

TABEL: 3.15. Hasil perhitungan *dot product* data uji dengan data latih

A1	A2	A3	A4	A5
25.00	49.00	54.00	51.00	27.00

Selanjutnya dilakukan perhitungan fungsi keputusan:

$$f(x) = \sum_{i=1}^{m} \alpha_i y_i K(x, x_i) + b$$

$f(x)$ = ((1 x (-1) x 25.00) + (0 x 1 x 49.00) + (0 x 1 x 54.00) + (0 x 1 x 51.00) + (1 x (-1) x 27.00)) + 61.00

= 9.00

Karena hasilnya bernilai positif maka data contoh tersebut merupakan kelas +1, maka termasuk Penduduk Miskin.

3.3.4. Analisis Kebutuhan Perangkat Keras dan Perangkat Lunak

Perangkat keras (*Hardware*) dibutuhkan untuk menjalankan beberapa perangkat lunak (*Software*) dalam membangun dan menjalankan aplikasi yang dibuat dalam penelitian ini. Minimum perangkat keras yang dibutuhkan adalah sebagai berikut:

1. Processor Intel Pentium Core i3
2. RAM 512 MB
3. Kapasitas Harddisk 40 GB
4. Monitor
5. Keyboard
6. Mouse

Selain perangkat keras, dibutuhkan juga perangkat lunak (software) yang merupakan pendukung sistem yang terdiri dari sistem operasi dan aplikasi database. Perangkat lunak minimum yang dibutuhkan dalam penelitian ini adalah sebagai berikut:

1. Sistem Operasi Windows 7
2. Bahasa Pemrograman, PHP 7.0 dan Framework
3. PHP AI Library
4. StarUML untuk perancangan UML dan ERD
5. XAMPP V7.3.4
6. Mozilla Firefox

3.4. Perancangan Program / Desain

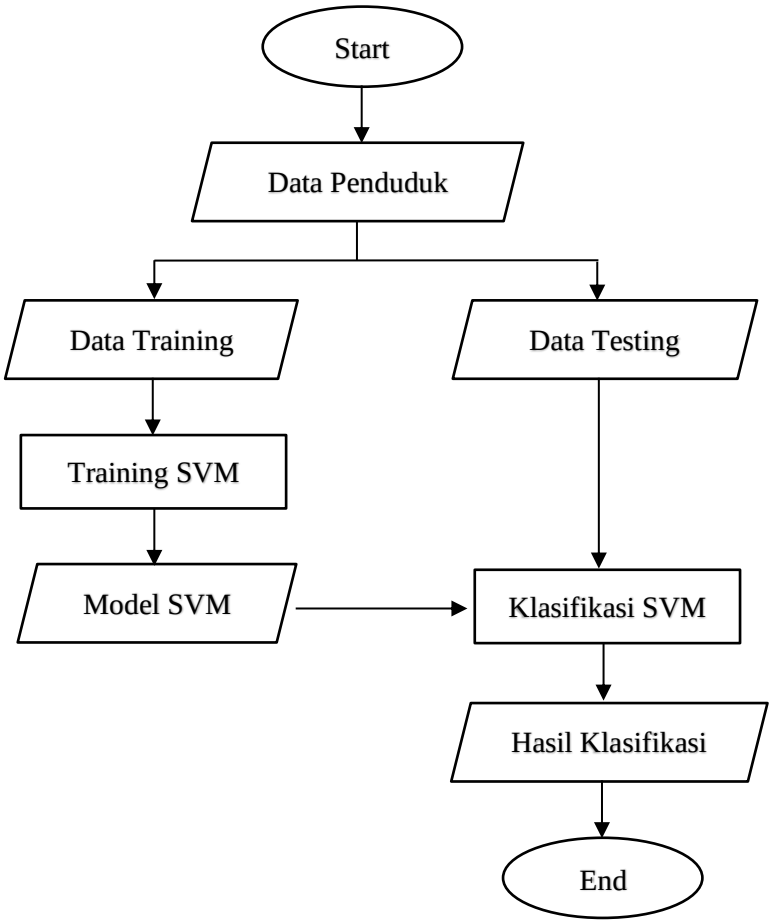
Tahapan selanjutnya dari metode *waterfall* adalah perancangan atau desain, pada tahap ini mentranslasi kebutuhan aplikasi klasifikasi penduduk dari tahap analisis, yaitu melakukan perancangan metode klasifikasi dengan SVM dan perancangan perangkat lunak yang meliputi perancangan diagram arsitektur perangkat lunak, struktur data dan representasi antarmuka agar nanti dapat diimplementasikan menjadi aplikasi. Desain pemodelan menggunakan *Unified Modeling Language* (UML) dan akan digunakan 4 diagram yaitu *use case* diagram, *activity* Diagram, *sequence* diagram dan *class* diagram. Untuk pemodelan basis data menggunakan *Entity Relationship Diagram* (ERD) dengan notasi Chen.

3.4.1. Perancangan Klasifikasi dengan *Support Vector Machine* (SVM)

Dalam klasifikasi dengan menggunakan SVM harus melakukan beberapa

tahapan, yaitu pertama pelatihan (*training*) dengan mempelajari nilai-nilai variabel data penduduk yang digunakan menjadi model, sesuai acuan yang diberikan. Acuan berupa kelas/kriteria penduduk, penduduk miskin dan penduduk tidak miskin. Kemudian tahap kedua yaitu pengujian *(testing*) dengan menggunakan model untuk mengklasifikasi penduduk.

Pada rancangannya seperti pada Gambar 3.2 variabel-variabel penduduk dan kelasnya akan dibagi menjadi data training dan data testing. Kemudian setelah dilakukan pelatihan dilanjutkan dengan proses klasifikasi sistem sesuai dengan metode yang digunakan yaitu SVM.

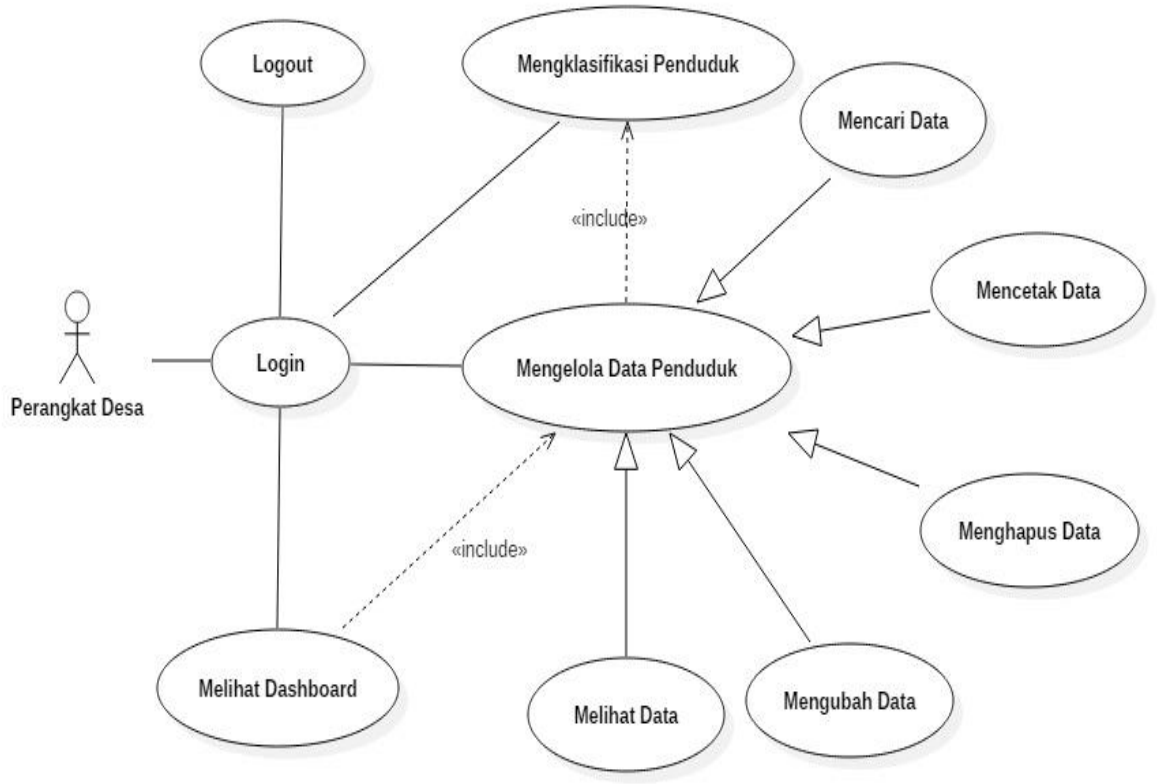


GAMBAR: 3.2. Proses Klasifikasi Sistem dengan Metode SVM

3.4.2. *Use Case* Diagram

Use case diagram merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat, yaitu mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. *Use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu (Rosa dan Shalahuddin, 2019).

3.3.1.1 Diagram *Use Case*



GAMBAR: 3.3. Use Case Diagram Klasifikasi Penduduk

3.3.1.2 Definisi Aktor

Berikut adalah deskripsi pendefinisian aktor pada aplikasi klasifikasi penduduk miskin :

TABEL: 3.16. Pendefinisian Aktor

No.	Aktor	Deskripsi
1.	Perangkat Desa	*User* atau Orang dari pemerintahan desa yang dapat melakukan proses klasifikasi dengan cara memasukan nilai indikator dari data penduduk desa wilayahnya. *User* juga dapat mengolah data berupa *record* hasil proses klasifikasi penduduk miskin, serta mencetaknya.

3.3.1.3 Definisi *Use Case*

Berikut adalah deskripsi pendefinisian use case pada aplikasi klasifikasi penduduk miskin :

TABEL: 3.17. Pendefinisian *Use Case*

No.	*Use Case*	Deskripsi
1.	Login	merupakan proses untuk melakukan *login* perangkat desa.
2.	Logout	merupakan proses untuk melakukan *logout* perangkat desa.
3.	Mengklasifikasi penduduk	merupakan proses klasifikasi penduduk miskin dengan cara memasukkan data penduduk beserta nilai indikator pada form yang telah disediakan sistem dan memprosesnya.

TABEL: 3.17. Pendefinisian *Use Case* (Lanjutan)

No.	*Use Case*	Deskripsi
4.	Mengelola data penduduk	merupakan proses generalisasi yang meliputi lima buah proses pengeloaan data penduduk yaitu melihat data, mengubah data, mencari data, menghapus data, dan mencetak data penduduk hasil klasifikasi.
5.	Mencari data	merupakan proses mencari data penduduk miskin hasil kalsifikasi yang terdapat pada database.
6.	Melihat data	merupakan proses melihat data penduduk miskin hasil kalsifikasi yang terdapat pada database.
7.	Mengubah data	merupakan proses mengubah data penduduk miskin hasil kalsifikasi yang terdapat pada database.
8.	Menghapus data	merupakan proses menghapus data penduduk miskin hasil kalsifikasi yang terdapat pada database.
9.	Mencetak data	merupakan proses mencetak data penduduk miskin hasil kalsifikasi yang terdapat pada database.
10.	Melihat dashboar	merupakan proses untuk melihat laporan atau rekapitulasi data penduduk.

3.3.1.4 Skenario *Use Case*

1. Nama *Use case* : Login

TABEL: 3.18. Skenario *Use Case* Login

Aksi Aktor	Reaksi Sistem
Skenario Normal	
1. Mengklik button Login	
	2. Menampilkan Form Login
3. Memasukkan *username* dan *password*	
	4. Memeriksa valid tidaknya data masukan dengan memeriksa ke table user
	5. Masuk ke aplikasi klasifikasi penduduk
Skenario Alternatif	
1. Mengklik button Login	
	2. Menampilkan Form Login
3. Memasukkan *username* dan *password*	
	4. Memeriksa valid tidaknya data masukan dengan memeriksa ke table user
	5. Menampilkan pesan *login* tidak *valid*
6. Memasukkan *username* dan *password* yang valid	
	7. Memeriksa valid tidaknya data masukan dengan memeriksa ke table user
	8. Masuk aplikasi klasifikasi penduduk

2. Nama *Use case* : Logout

TABEL: 3.19. Skenario *Use Case* Logout

Aksi Aktor	Reaksi Sistem
Skenario Normal	
1. Mengklik button Logout	
	2. Melakukkan logout atau keluar aplikasi klasifikasi penduduk

3. Nama *Use case* : Mengklasifikasi penduduk

TABEL: 3.20. Skenario *Use Case* Mengklasifikasi Penduduk

Aksi Aktor	Reaksi Sistem
Skenario Normal	
1. Memilih menu klasisifikasi	
	2. Menampilkan halaman form untuk input data penduduk berikut variabel-variabel yang diperlukan.
3. Mengisi form data penduduk	
	4. Memeriksa valid tidaknya data masukan
	5. Menyimpan data masukan ke dalam basis data
	6. Menampilkan pesan bahwa data sukses tersimpan
7. Mengklik button Klasifikasi	
	8. Melakukan proses klasifikasi dengan metode SVM
	9. Menampilkan hasil klasifikasi
Skenario Alternatif	
1. Memilih menu klasisifikasi	
	2. Menampilkan halaman form untuk input data penduduk berikut variabel-variabel yang diperlukan.
3. Mengisi form data penduduk	
	4. Memeriksa valid tidaknya data masukan
	5. Menampilkan pesan bahwa data masukan tidak valid
6. Memperbaiki data masukan	
	7. Memeriksa valid tidaknya data masukan
	8. Menyimpan data masukan ke dalam basis data
9. Mengklik button Klasifikasi	
	10. Melakukan proses klasifikasi dengan metode SVM
	11. Menampilkan hasil klasifikasi

4. Nama *Use case* : Mengelola data penduduk

TABEL: 3.21. Skenario *Use Case* Mengelola Data Penduduk

Aksi Aktor	Reaksi Sistem
Skenario Normal	
1. Memilih menu kelola data penduduk	
	2. Menampilkan halaman kelola data penduduk, yang didalamnya terdapat daftar penduduk dan menu cari (*search*), lihat, ubah, hapus dan cetak data.
3. Memilih salah satu menu (cari (*search*), lihat, ubah, hapus dan cetak data)	

5. Nama *Use case* : Mencari data

TABEL: 3.22. Skenario *Use Case* Mencari Data

Aksi Aktor	Reaksi Sistem
Skenario Normal	
1. Memasukkan kata kunci pencarian	
	2. Mencari data penduduk yang akan dicari
	3. Menampilkan data penduduk yang dicari
4. Memilih data penduduk yang dicari	
	5. Menampilkan data penduduk (Semua kolom) dari data penduduk yang dipilih
Skenario Alternatif	
1. Memasukkan kata kunci pencarian	
	2. Mencari data penduduk yang akan dicari
	3. Menampilkan pesan data penduduk yang dicari tidak ada
4. Memasukkan kata kunci pencarian	
	5. Mencari data penduduk yang akan dicari
	6. Menampilkan data penduduk yang dicari
7. Memilih data penduduk yang dicari	

TABEL: 3.22. Skenario *Use Case* Mencari Data (Lanjutan)

Skenario Alternatif	
	8. Menampilkan data penduduk (Semua kolom) dari data penduduk yang dipilih

6. Nama *Use case* : Melihat data

TABEL: 3.23. Skenario *Use Case* Melihat Data

Aksi Aktor	**Reaksi Sistem**
Skenario Normal	
1. Memilih data penduduk yang akan dilihat	
	2. Menampilkan data penduduk

7. Nama *Use case* : Mengubah data

TABEL: 3.24. Skenario *Use Case* Mengubah Data

Aksi Aktor	**Reaksi Sistem**
Skenario Normal	
1. Memilih data penduduk yang akan diubah	
	2. Menampilkan semua kolom data penduduk yang akan diubah
3. Mengubah data penduduk	
	4. Memeriksa valid tidaknya data masukan
	5. Menyimpan data yang telah diubah ke basis data
	6. Menampilkan pesan bahwa data sukses tersimpan
Skenario Alternatif	
1. Memilih data penduduk yang akan dubah	
	2. Menampilkan semua kolom data penduduk yang akan diubah
3. Mengubah data penduduk	
	4. Memeriksa valid tidaknya data masukan
	5. Menampilkan pesan bahwa data masukan tidak valid

TABEL: 3.24. Skenario *Use Case* Mengubah Data (Lanjutan)

Skenario Alternatif	
6. Memperbaiki data masukan yang diubah dan tidak valid	
	7. Memeriksa valid tidaknya data masukan
	8. Menyimpan data yang telah diubah ke basis data
	9. Menampilkan pesan bahwa data sukses tersimpan

8. Nama *Use case* : Menghapus data

TABEL: 3.25. Skenario *Use Case* Menghapus Data

Aksi Aktor	**Reaksi Sistem**
Skenario Normal	
1. Memilih data penduduk dihapus	
	2. Menampilkan pesan konfirmasi apakah data benar akan dihapus
3. Mengklik pilihan setuju data dihapus	
	4. Menghapus data penduduk dari basis data
	5. Menampilkan pesan bahwa data sukses dihapus
Skenario Alternatif	
1. Memilih data penduduk yang akan dihapus	
	2. Menampilkan pesan konfirmasi apakah data benar akan dihapus
3. Mengklik pilihan tidak setuju data dihapus	
	4. Kembali ke halaman kelola data penduduk

9. Nama *Use case* : Mencetak data

TABEL: 3. 26. Skenario *Use Case* Mencetak Data

Aksi Aktor	**Reaksi Sistem**
Skenario Normal	
1. Memilih data penduduk yang akan dicetak	
	2. Mencetak data penduduk

10. Nama *Use case* : Melihat dashboard

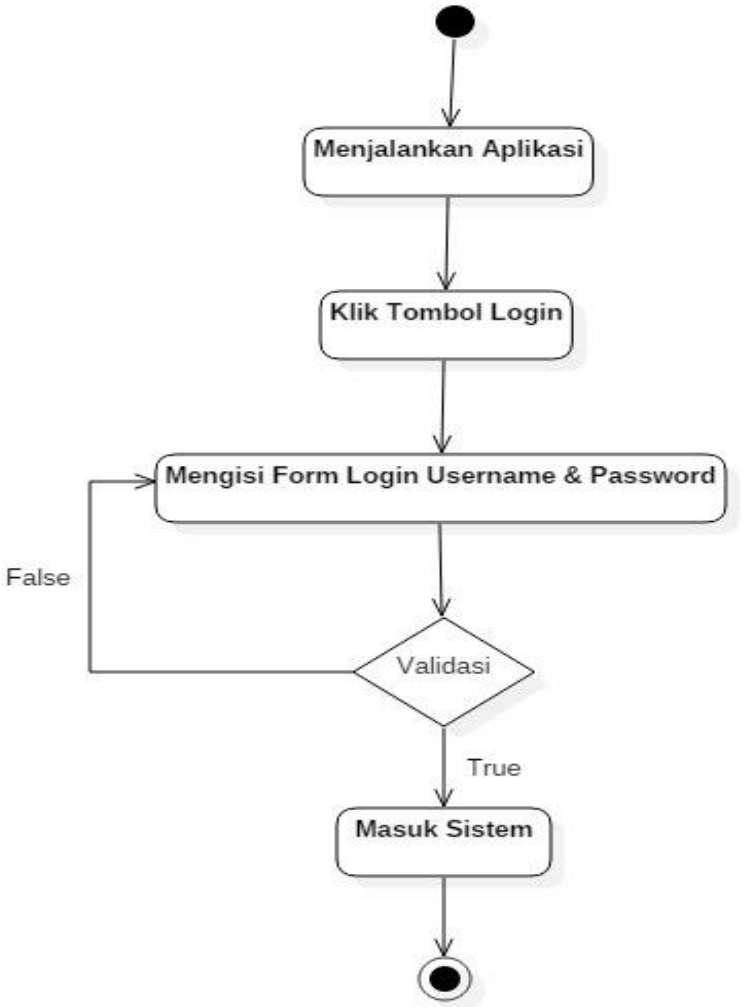
TABEL: 3. 27. Skenario *Use Case* Melihat Dashboard

Aksi Aktor	Reaksi Sistem
Skenario Normal	
1. Memilih menu dashboard	
	2. Menampilkan halaman dashboard

3.4.3. *Activity* Diagram

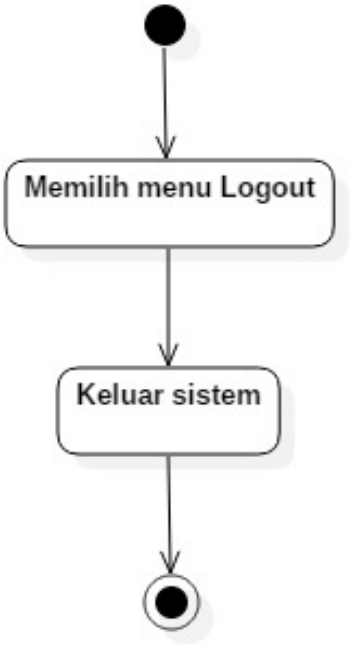
Berikut adalah diagram aktivitas atau *activity* diagram untuk menggambarkan *workflow* (aliran kerja) atau aktivitas dari aplikasi klasifikasi penduduk :

1. Diagram *Activity* Login



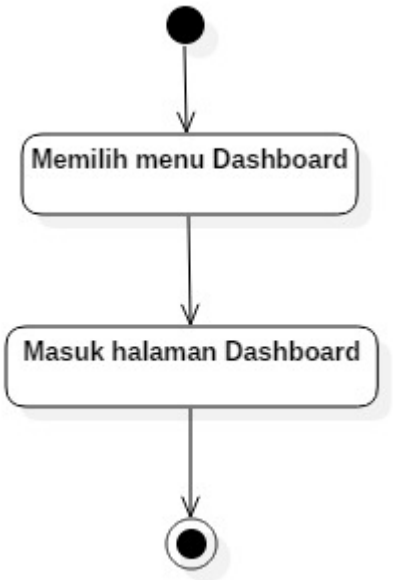
GAMBAR: 3.4. Diagram *Activity* Login

2. Diagram *Activity* Logout



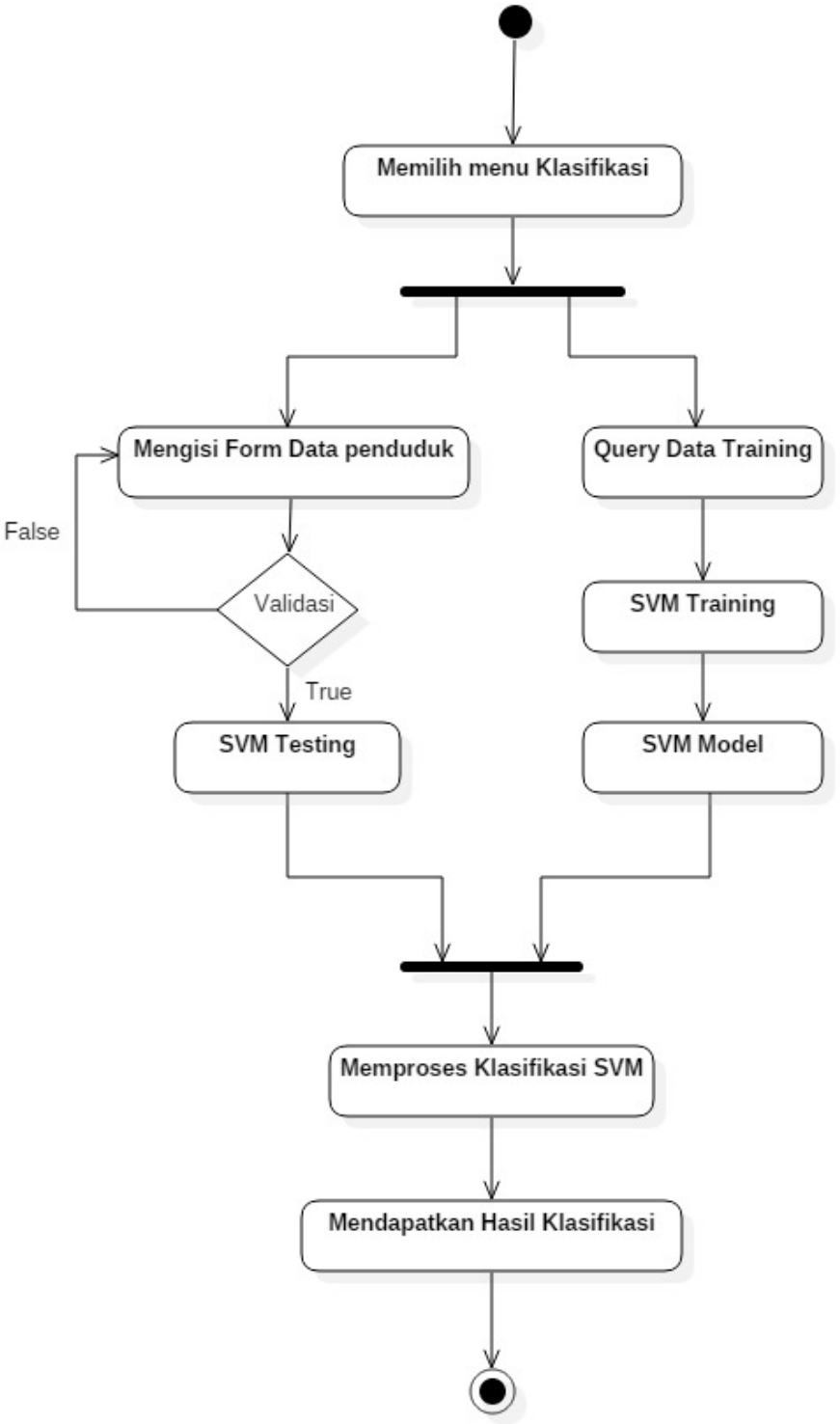
GAMBAR: 3.5. Diagram *Activity* Logout

3. Diagram *Activity* Melihat Dashboard



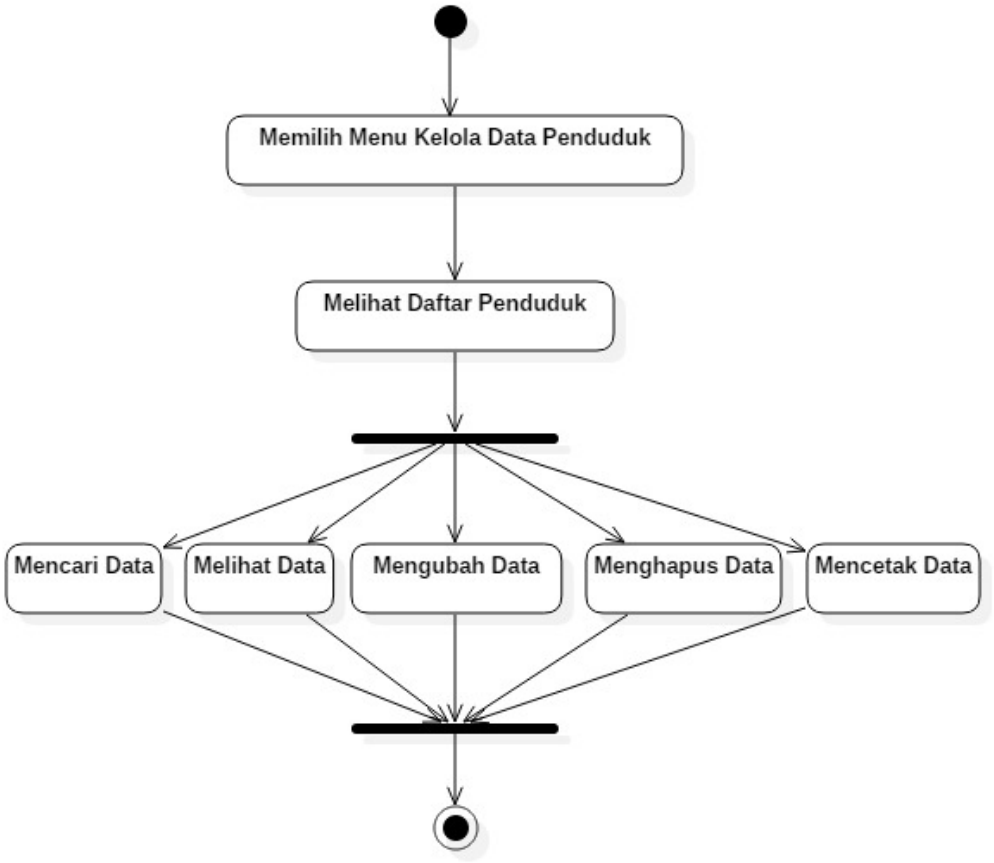
GAMBAR: 3.6. Diagram *Activity* Melihat Dashboard

4. Diagram *Activity* Mengklasifikasi Penduduk



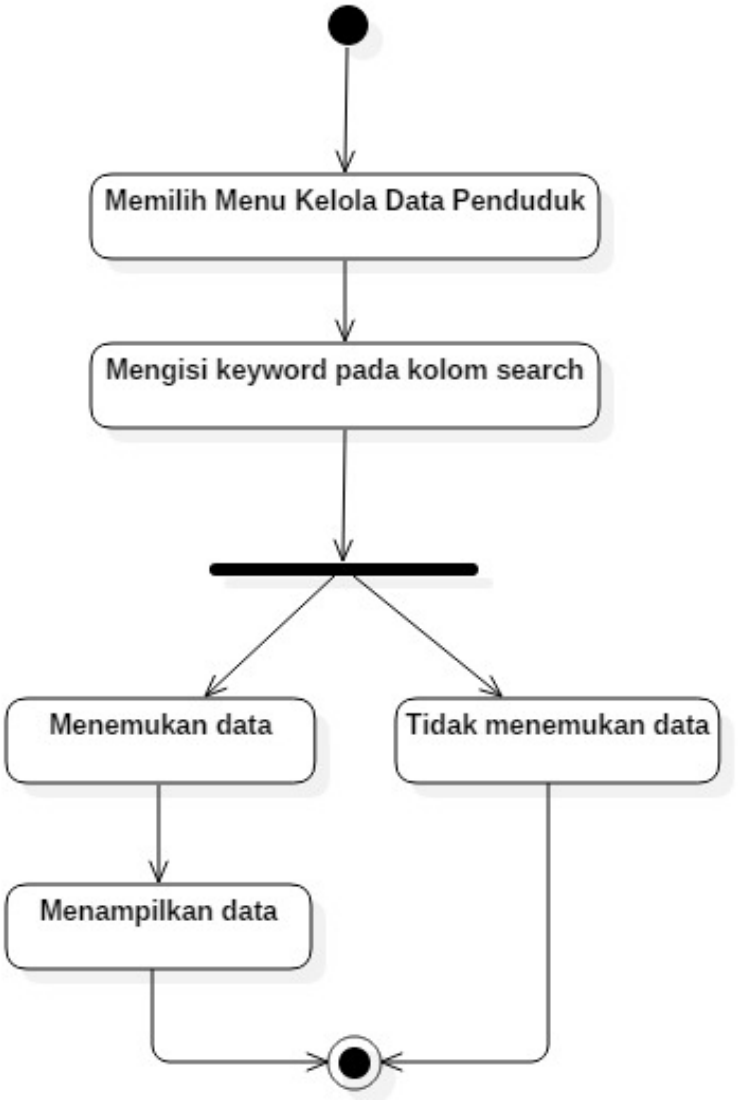
GAMBAR: 3.7. Diagram *Activity* Mengklasifikasi Penduduk

5. Diagram *Activity* Kelola Data Penduduk



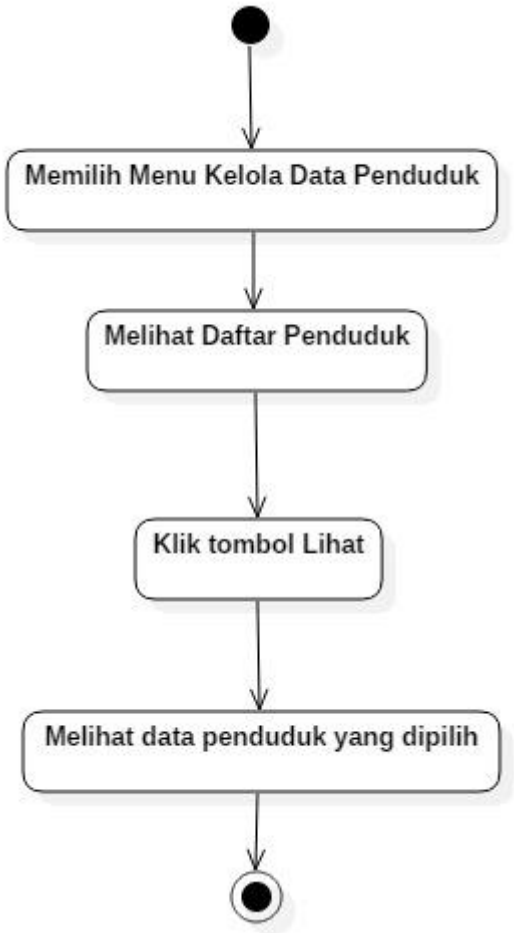
GAMBAR: 3.8. Diagram *Activity* Kelola Data Penduduk

6. Diagram *Activity* Mencari Data



GAMBAR: 3.9. Diagram *Activity* Mencari Data Penduduk

7. Diagram *Activity* Melihat Data

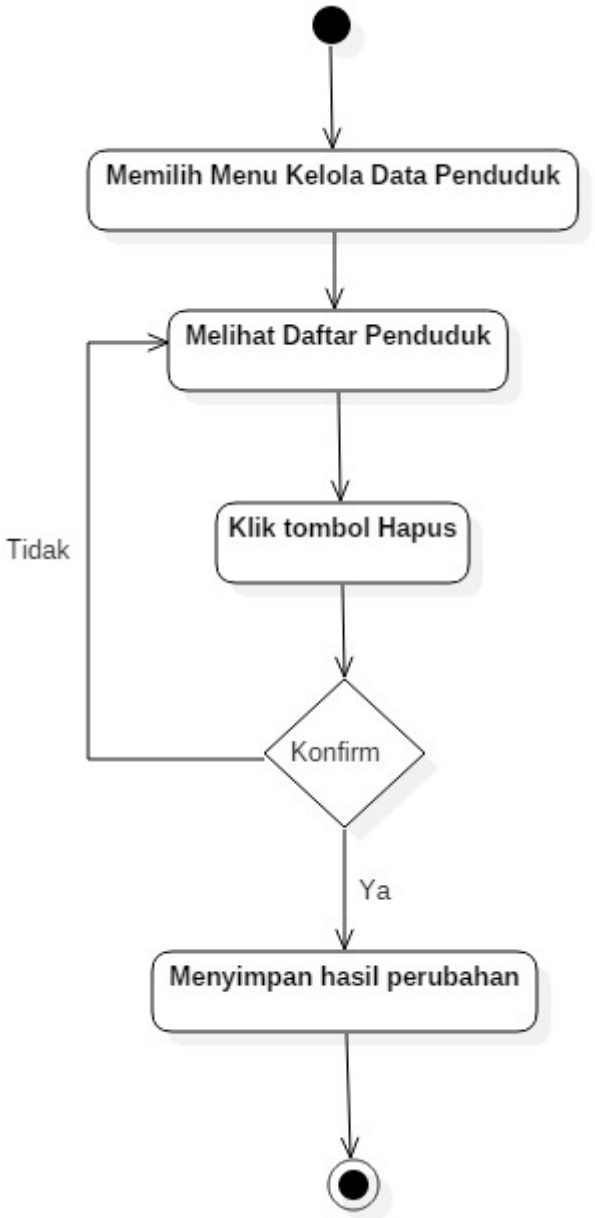


GAMBAR: 3.10. Diagram *Activity* Melihat Data Penduduk

8. Diagram *Activity* Mengubah Data

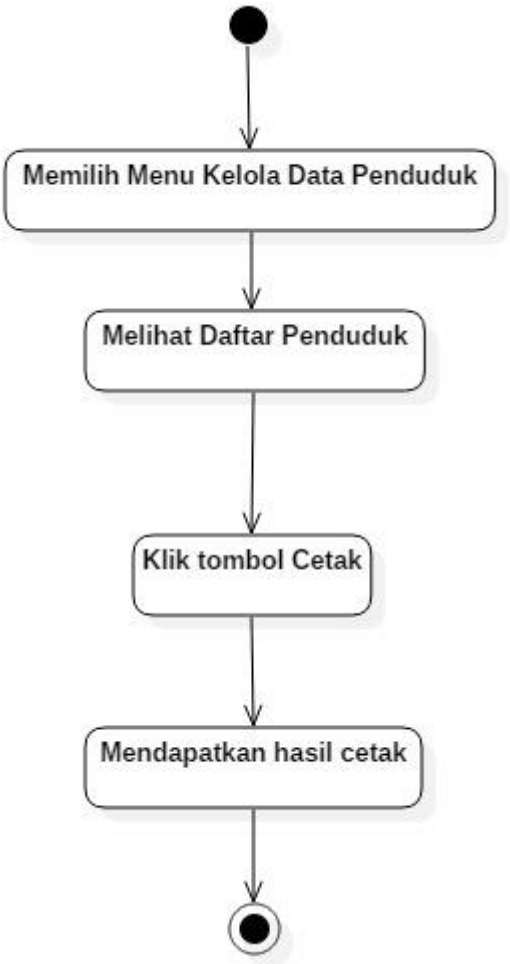
GAMBAR: 3.11. Diagram *Activity* Mengubah Data Penduduk

9. Diagram *Activity* Menghapus Data



GAMBAR: 3.12. Diagram *Activity* Menghapus Data Penduduk

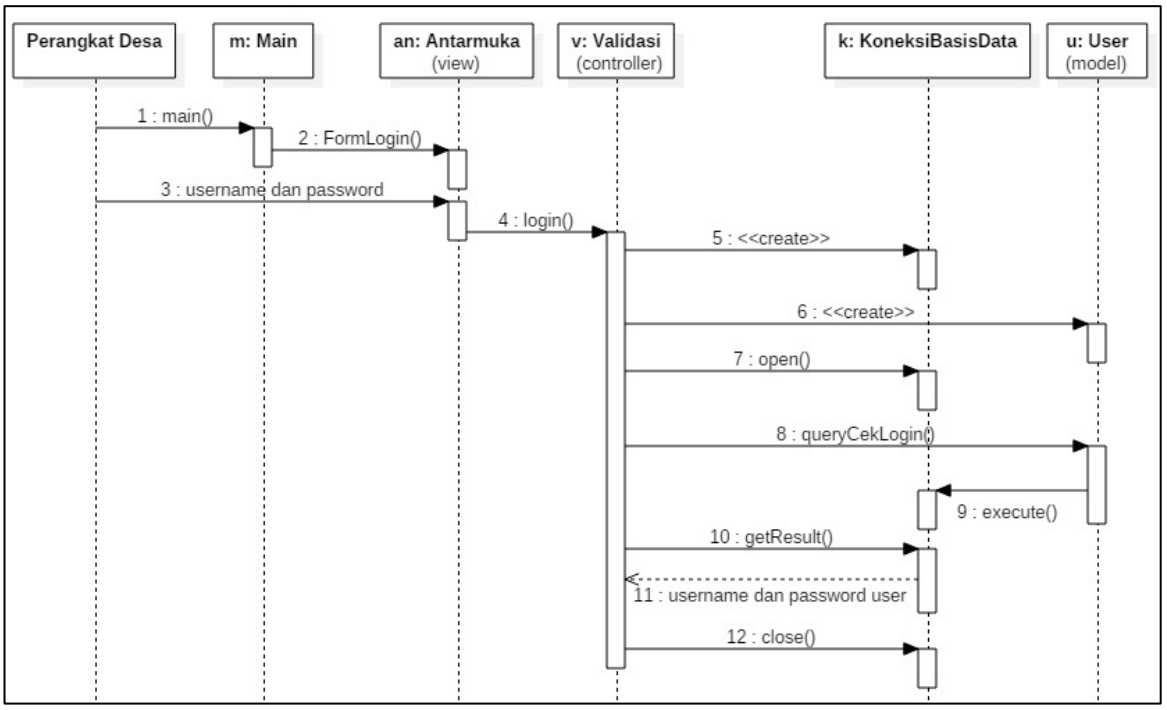
10. Diagram *Activity* Mencetak Data



GAMBAR: 3.13. Diagram *Activity* Mencetak Data Penduduk

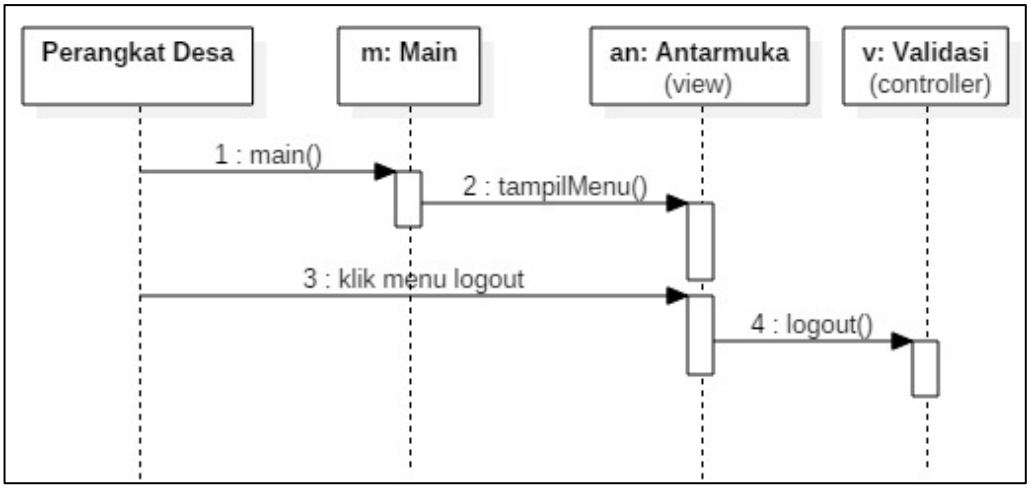
3.4.4. *Sequence* Diagram

1. Diagram *Sequence* Login



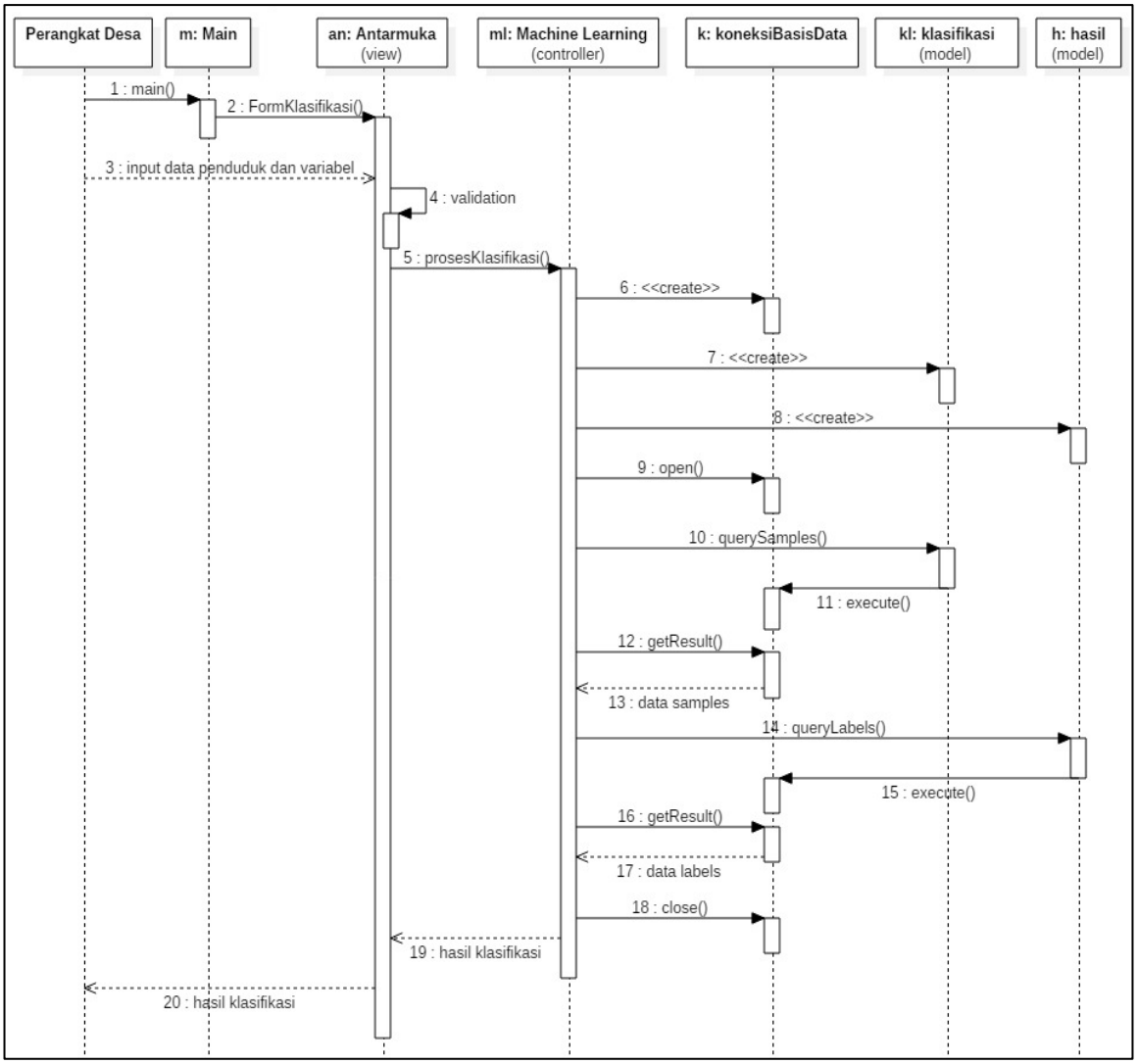
GAMBAR: 3.14. Diagram *Sequence* Login

2. Diagram *Sequence* Logout



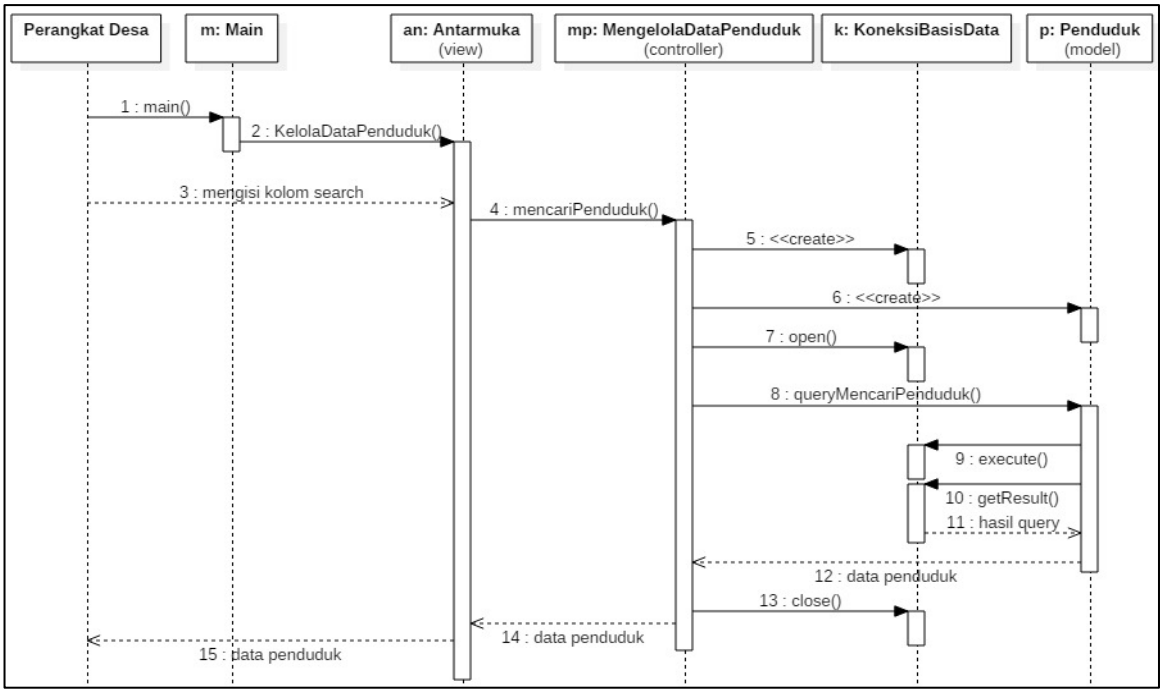
GAMBAR: 3.15. Diagram *Sequence* Logout

3. Diagram *Sequence* Mengklasifikasi Penduduk



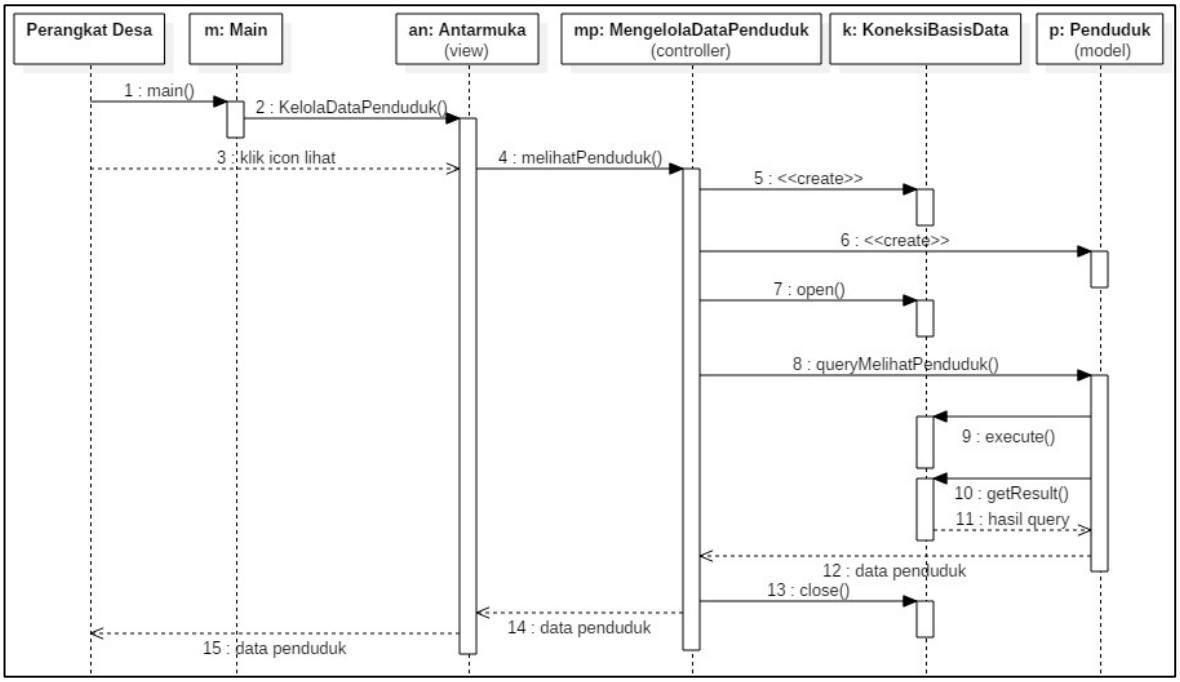
GAMBAR: 3.16. Diagram *Sequence* Mengklasifikasi Penduduk

4. Diagram *Sequence* Mencari Data Penduduk



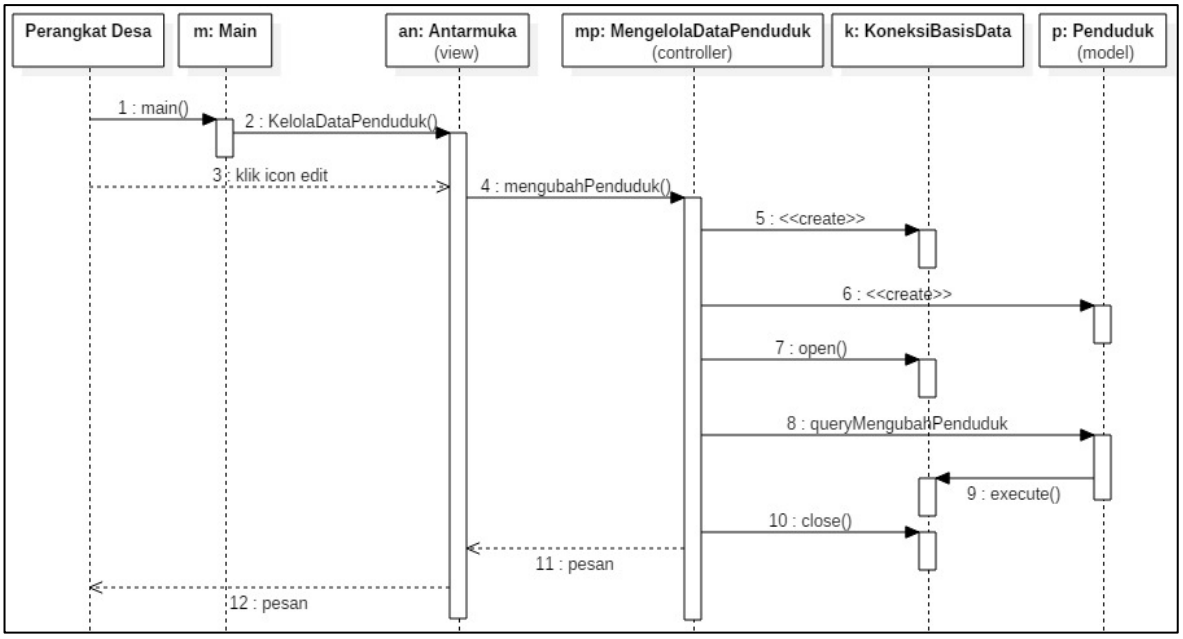
GAMBAR: 3.17. Diagram *Sequence* Mencari Data Penduduk

5. Diagram *Sequence* Melihat Data Penduduk



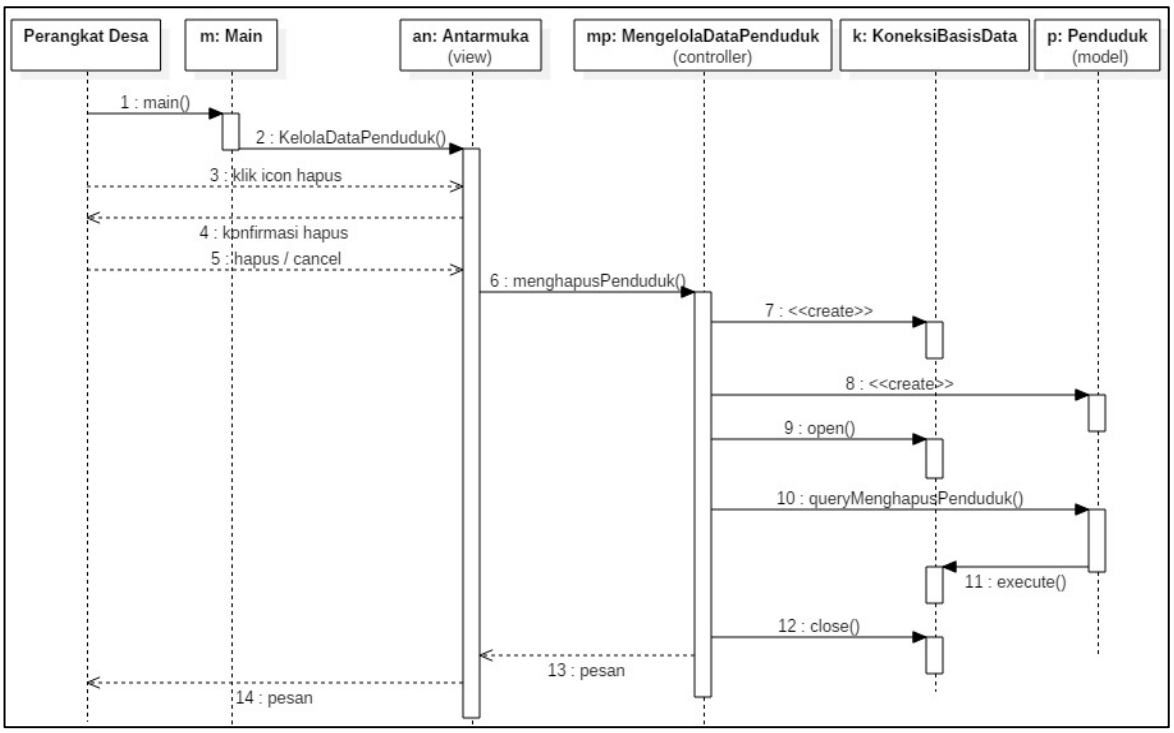
GAMBAR: 3.18. Diagram *Sequence* Melihat Data Penduduk

6. Diagram *Sequence* Mengubah Data Penduduk



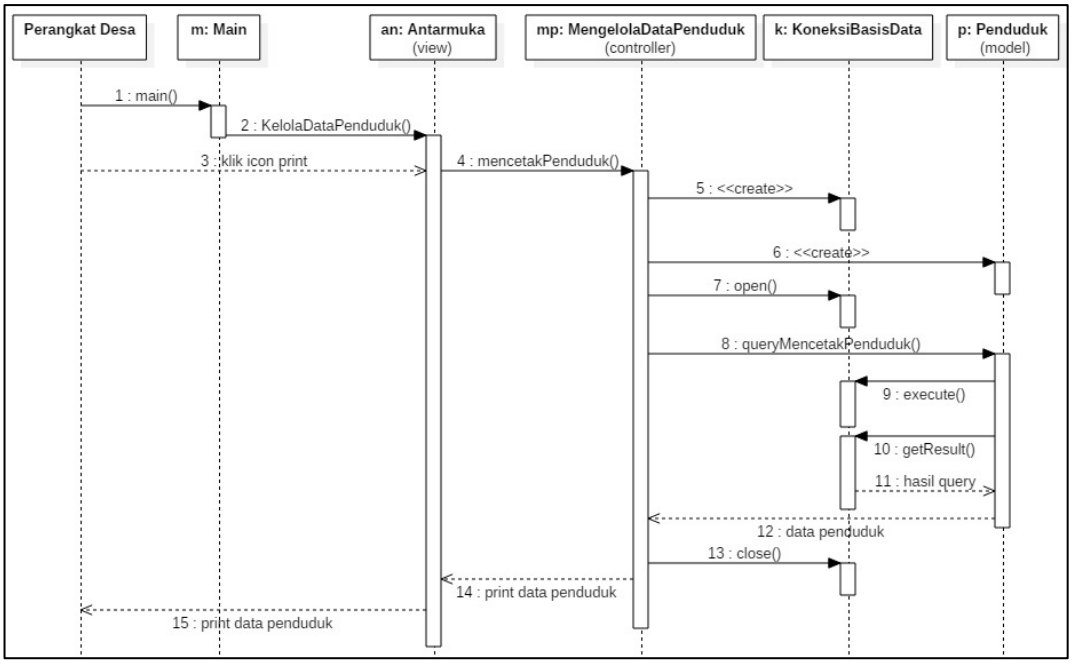
GAMBAR: 3.19. Diagram *Sequence* Mengubah Data Penduduk

7. Diagram *Sequence* Menghapus Data Penduduk



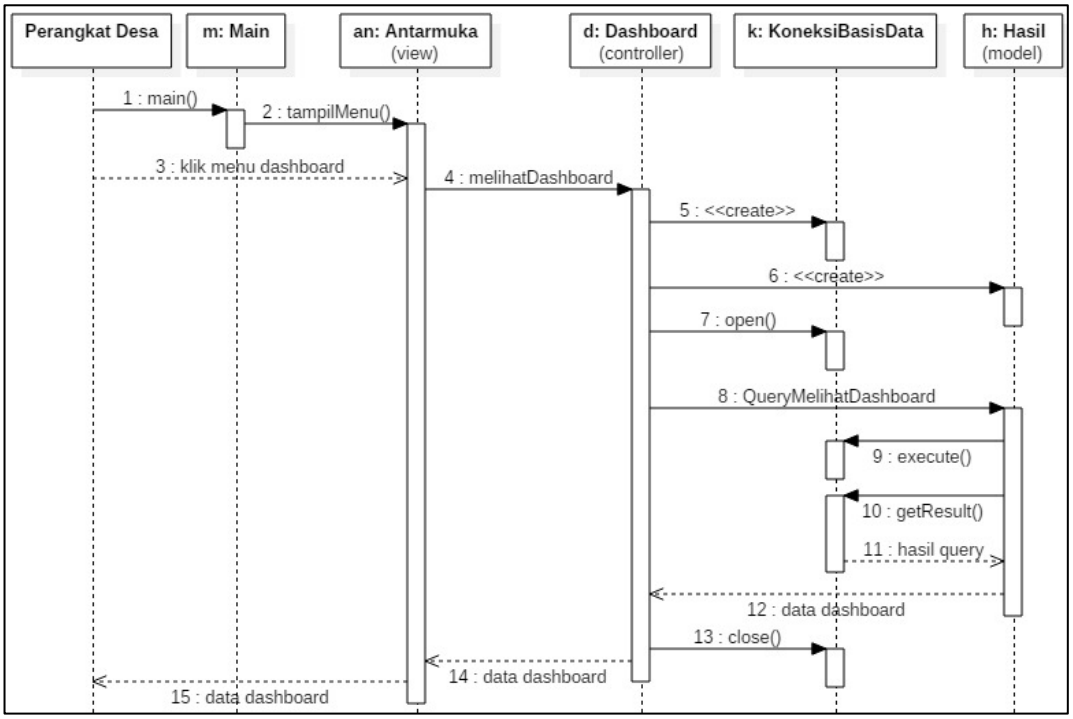
GAMBAR: 3.20. Diagram *Sequence* Menghapus Data Penduduk

8. Diagram *Sequence* Mencetak Data Penduduk



GAMBAR: 3.21. Diagram *Sequence* Mencetak Data Penduduk

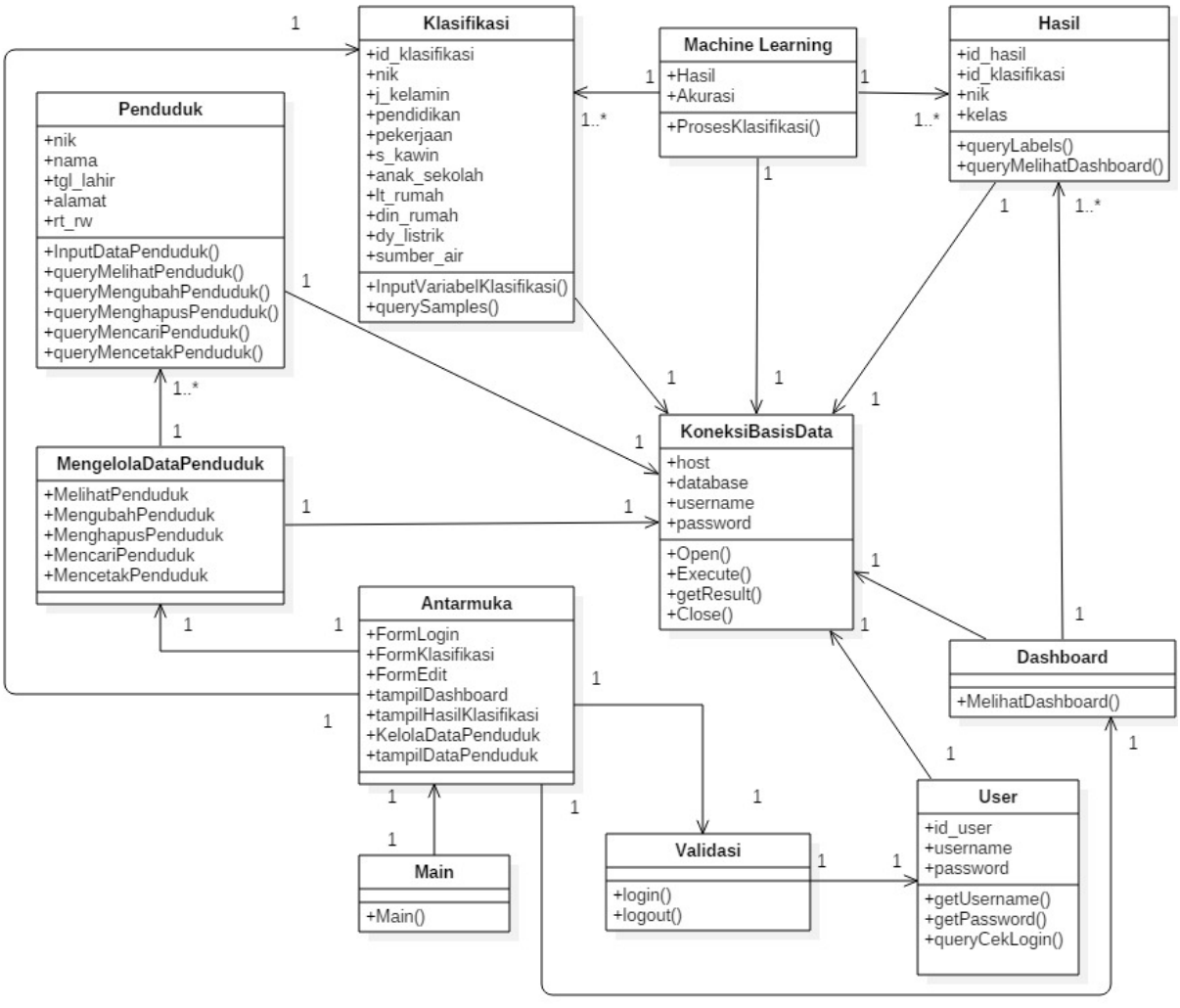
9. Diagram *Sequence* Melihat Dashboard



GAMBAR: 3.22. Diagram *Sequence* Melihat Dashboard

3.4.5. *Class* Diagram

Dalam membangun aplikasi klasifikasi penduduk untuk menggambarkan struktur sistem, penulis mendefinisikan menjadi 11 kelas yang terdiri dari 4 jenis yaitu kelas main, antarmuka, proses, data dan utilitas. Diagram *Class* aplikasi klasifikasi penduduk terdapat pada Gambar 3.23.



GAMBAR: 3.23. Diagram *Class* Aplikasi Klasifikasi Penduduk

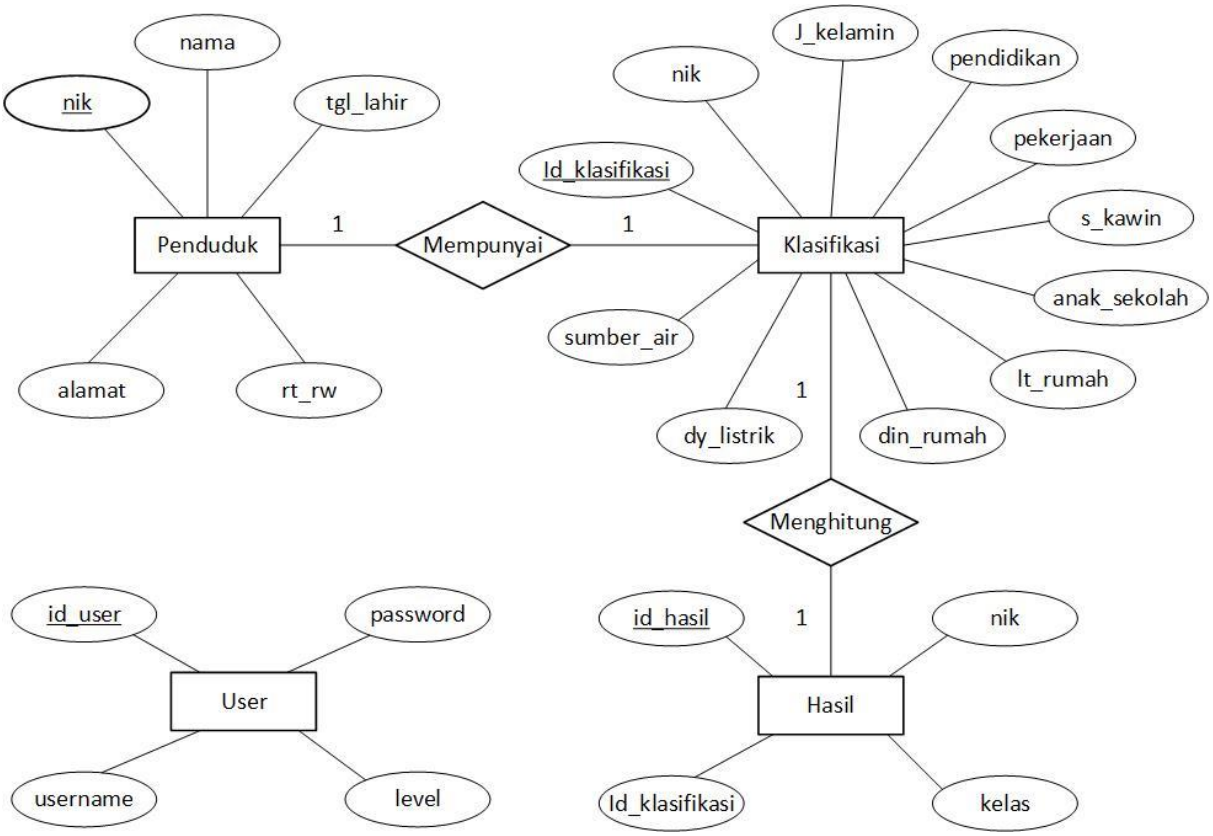
Untuk keterangan Gambar 3.22 dapat dilihat pada Tabel 3.16.

TABEL: 3.28. Keterangan Kelas pada Aplikasi Klasifikasi Penduduk

No	Nama Kelas	Keterangan
1.	Main	Merupakan kelas utama, yaitu fungsi awal dieksekusi ketika sistem dijalankan
2.	Antarmuka	Merupakan kelas yang menangani tampilan
3.	Validasi	Merupakan kelas proses untuk validasi login dan logout
4.	MengelolaDataPenduduk	Merupakan kelas proses yang diambil dari pendefinisian *use case* mengelola data penduduk yang didalamnya harus juga menangani proses mencari, melihat, mengubah, menghapus dan mencetak data penduduk
5.	Penduduk	Merupakan kelas data yang digunakan untuk memproses segala pengaksesan terhadap tabel penduduk
6.	Klasifikasi	Merupakan kelas data yang digunakan untuk memproses segala pengaksesan terhadap tabel klasifikasi
7.	Hasil	Merupakan kelas data yang digunakan untuk memproses segala pengaksesan terhadap tabel hasil
8.	User	Merupakan kelas data yang digunakan untuk memproses segala pengaksesan terhadap tabel user
9.	MachineLearning	Merupakan kelas proses untuk memproses klasifikasi data penduduk dengan metode SVM
10.	Dashboard	Merupakan kelas proses untuk menampilkan laporan atau rekapitulasi data penduduk yang sudah masuk di basis data
11.	KoneksiBasisData	Merupakan kelas utilitas untuk koneksi ke basis data dan melakukan *query*

3.4.6. *Entity Relationship Diagram* (ERD)

Untuk menyimpan data agar dapat diakses dengan mudah dan cepat maka diperlukan media penyimpan berupa basis data. Untuk pembangunan aplikasi ini berikut pada Gambar 3.24 adalah pemodelan basis data dengan *Entity Relationship Diagram* (ERD).



GAMBAR: 3.24. ER Diagram Aplikasi Klasifikasi Penduduk

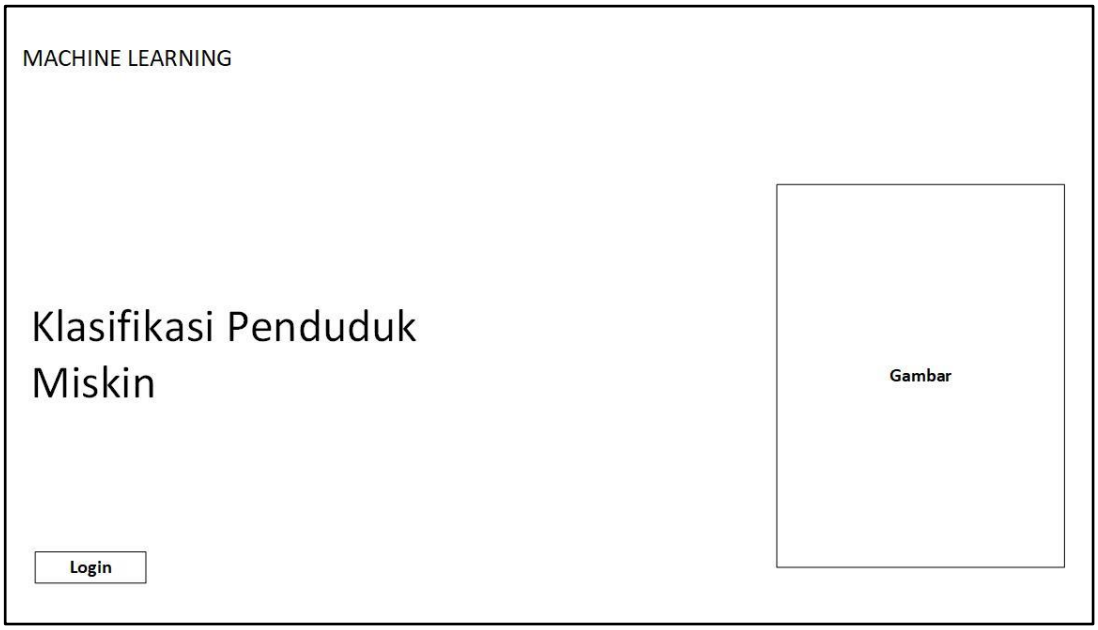
Dari pemodelan ERD yang telah dibuat dapat ditentukan table-tabel yang akan digunakan dalam pembangunan aplikasi klasifikasi penduduk menggunakan *machine learning* dengan metode *Support Vektor Machine* (SVM) ini. Terdiri dari Tabel Penduduk untuk menyimpan data informasi penduduk, Tabel Klasifikasi untuk menyimpan data variable-variabel dan Tabel Hasil untuk menyimpan data hasil proses klasifikasi.

3.4.7. Perancangan Antarmuka (*Interface*)

Untuk memudahkan dalam membangun aplikasi klasifikasi penduduk miskin ini dibuat perancangan antarmuka yang merupakan bentuk tampilan grafis yang berhubungan langsung dengan pengguna. Dengan tujuan supaya pengguna (*user*) mudah, efisien dan menyenangkan dalam penggunaanya. Adapun perancangan

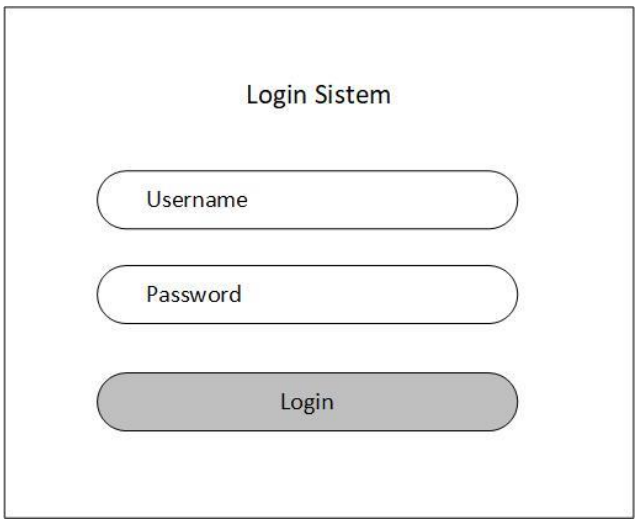
antarmuka aplikasi klasifikasi penduduk adalah sebagai berikut:

1. Antarmuka Halaman Utama



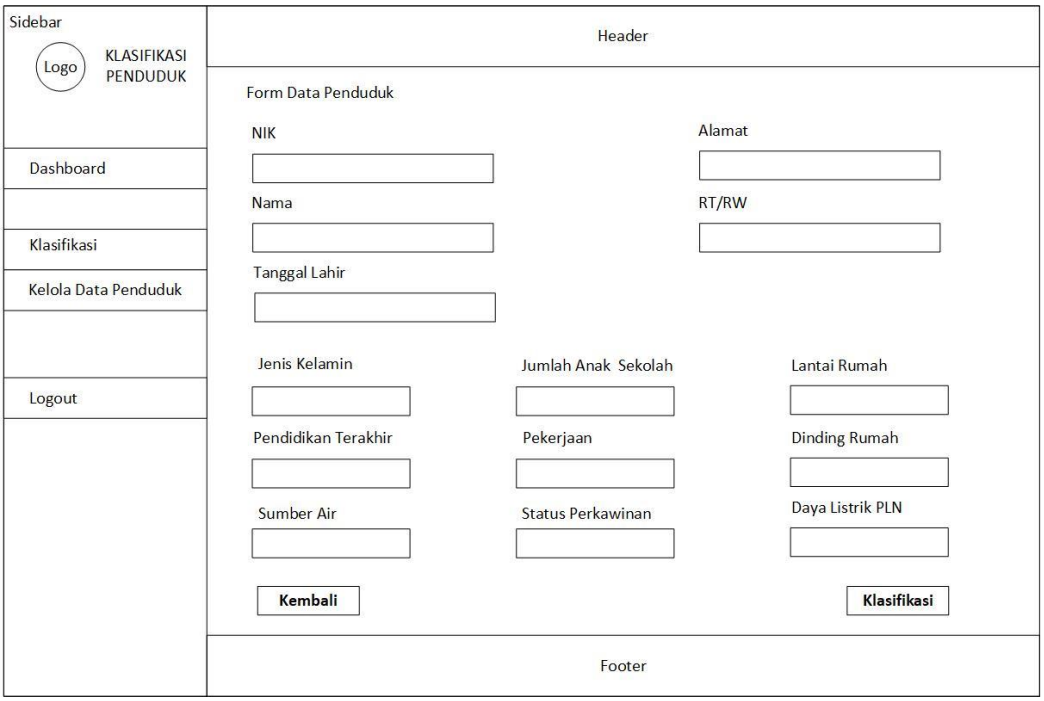
GAMBAR: 3. 25. Antarmuka Halaman Utama

2. Antarmuka Halaman Login



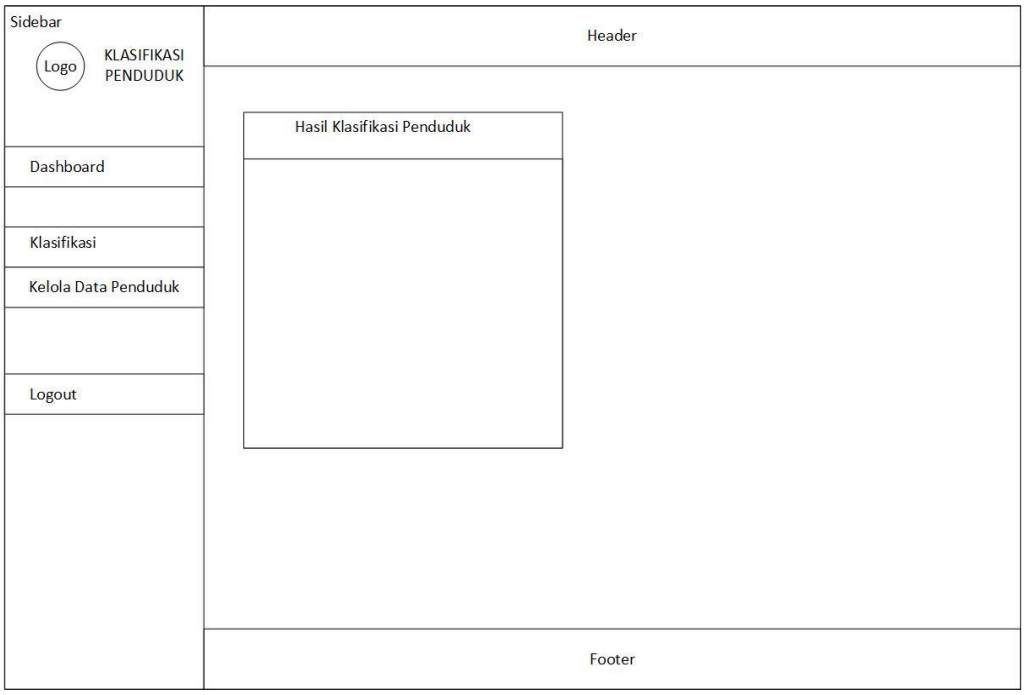
GAMBAR: 3.26. Antarmuka Halaman Login

3. Antarmuka Halaman Form Klasifikasi



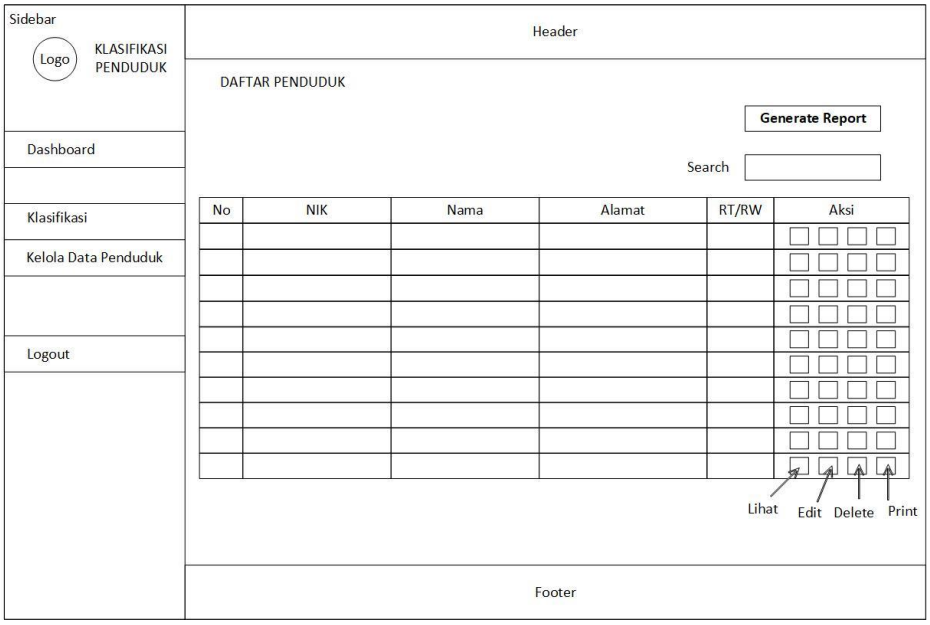
GAMBAR: 3.27. Antarmuka Halaman Form Klasifikasi

4. Antarmuka Halaman Hasil Proses Klasifikasi



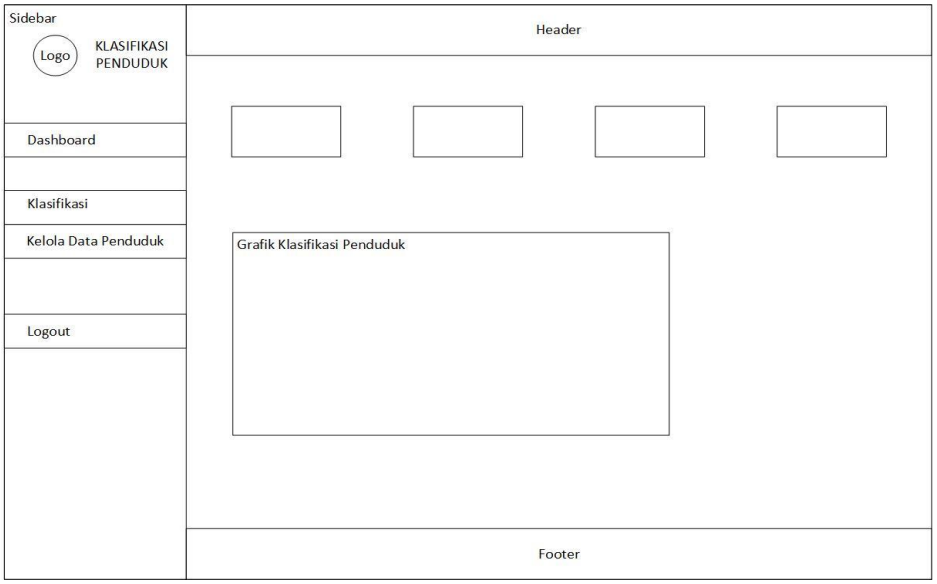
GAMBAR: 3.28. Antarmuka Halaman Hasil Proses Klasifikasi

5. Antarmuka Halaman Kelola Data Penduduk



GAMBAR: 3.29. Antarmuka Halaman Kelola Data Penduduk

6. Antarmuka Halaman Dashboard



GAMBAR: 3.30. Antarmuka Halaman Dashboard

BAB IV

IMPLEMENTASI DAN UJI COBA

4.1. Implementasi

Pada tahapan ini sesuai dengan tahapan metode pengembangan sistem yang digunakan yaitu SDLC (*System Development Life Cycle*) model *Waterfall* adalah pengodean atau pembuatan kode program yang merupakan implementasi dari rancangan pembangunan aplikasi klasifikasi penduduk yang telah dibuat sebelumnya.

4.1.1. Perangkat Keras (*Hardware*)

Perangkat keras (*Hardware*) yang digunakan untuk menjalankan beberapa perangkat lunak (*Software*) dalam membangun dan menjalankan aplikasi klasifikasi penduduk adalah sebagai berikut:

1. Processor Intel Pentium Core i5
2. RAM 2 GB
3. Kapasitas Harddisk 40 GB
4. Monitor
5. Keyboard
6. Mouse

4.1.2. Perangkat Lunak (*Software*)

Selain perangkat keras, digunakan juga perangkat lunak (software) yang merupakan pendukung sistem yang terdiri dari sistem operasi dan aplikasi database. Perangkat lunak yang digunakan dalam pembangunan aplikasi klasifikasi penduduk

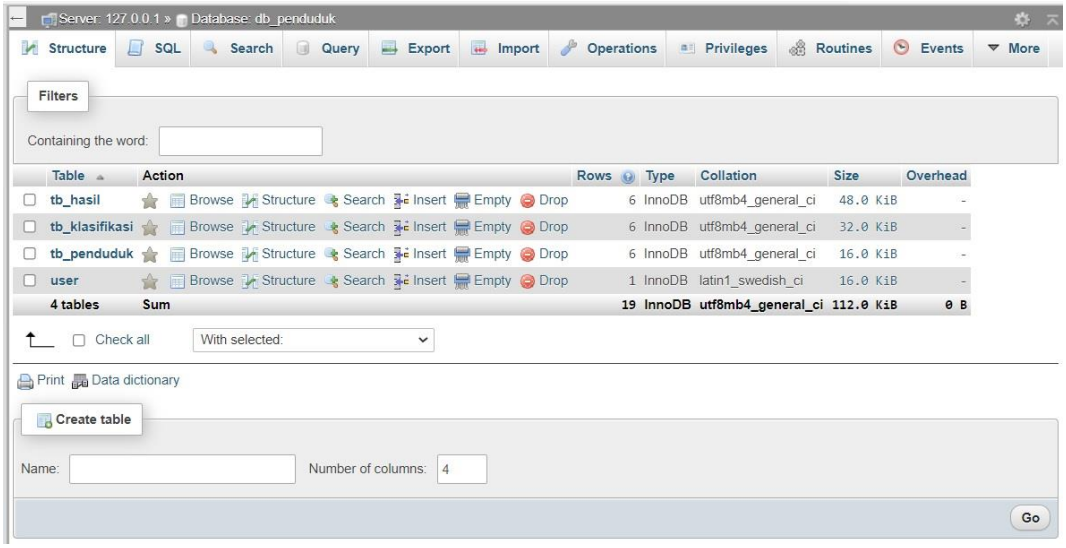
adalah sebagai berikut:

1. Sistem Operasi Windows 10
2. StarUML untuk perancangan UML dan ERD
3. Visual Studio Code
4. Bahasa Pemrograman, PHP 7.0 dan Framework Codeigniter 3
5. PHP AI Library
6. XAMPP V7.3.4
7. Mozilla Firefox

4.1.3. Implementasi Basis Data (*Database)*

Berdasarkan pemodelan ERD dan rancangan tabel-tabel yang telah dibuat sebelumnya, maka implementasi basis data (*database*) untuk aplikasi klasifikasi penduduk menggunakan web server dari XAMPP dengan MySQL. Berikut implementasi basis data aplikasi klasifikasi penduduk :

1. Implementasi struktur basis data (*database*)

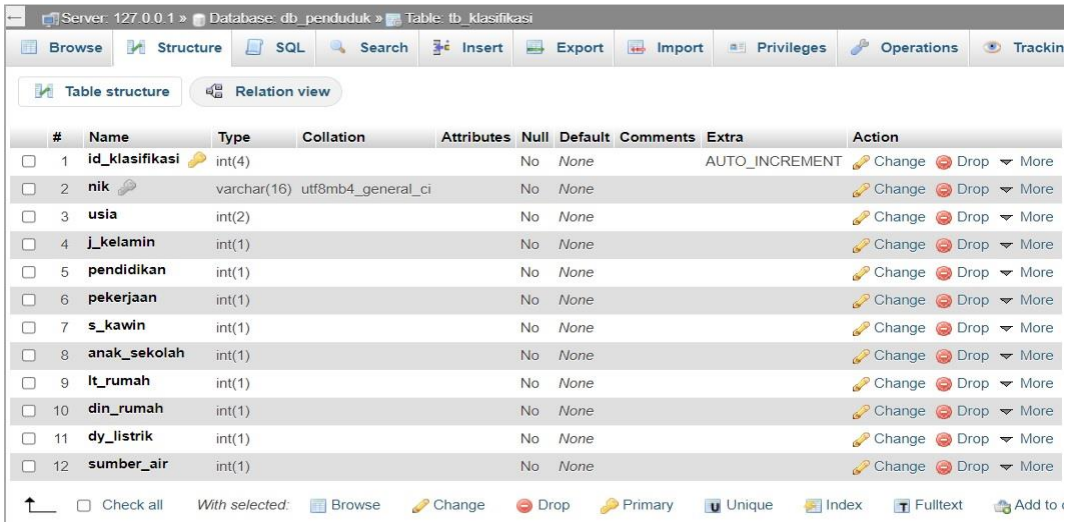


GAMBAR: 4.1. Struktur Basis Data (*Database*)

2. Implementasi tabel penduduk

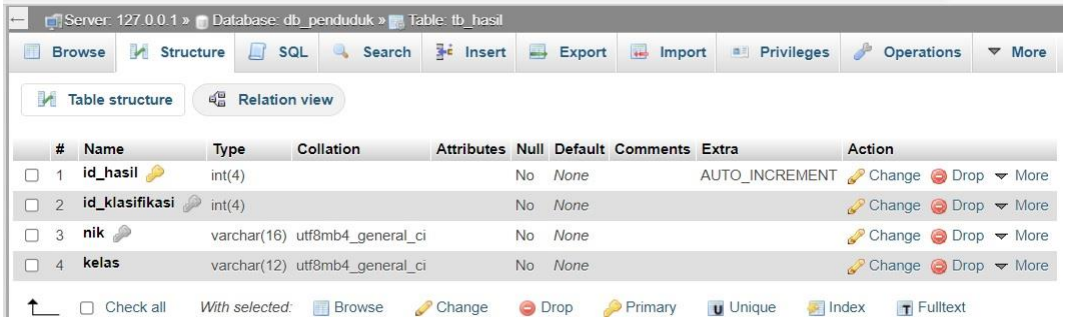
GAMBAR: 4.2. Struktur Tabel Penduduk

3. Implementasi tabel klasifikasi



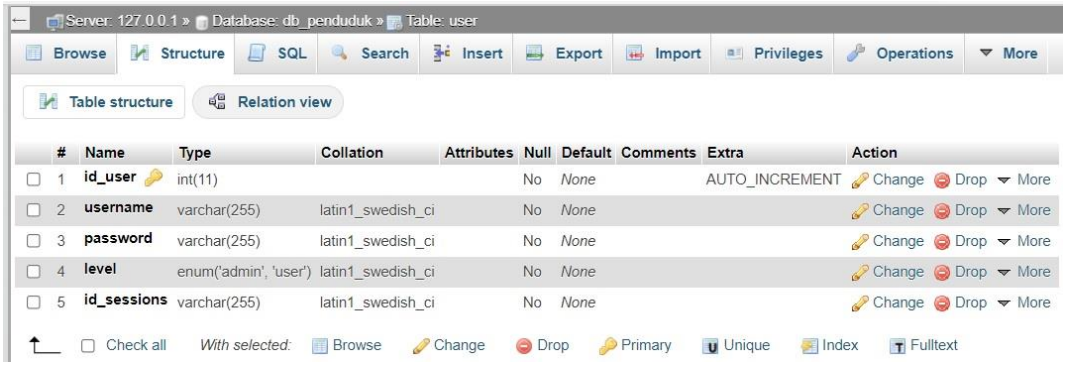
GAMBAR: 4. 3. Strukur Tabel Klasifikasi

4. Implementasi tabel hasil



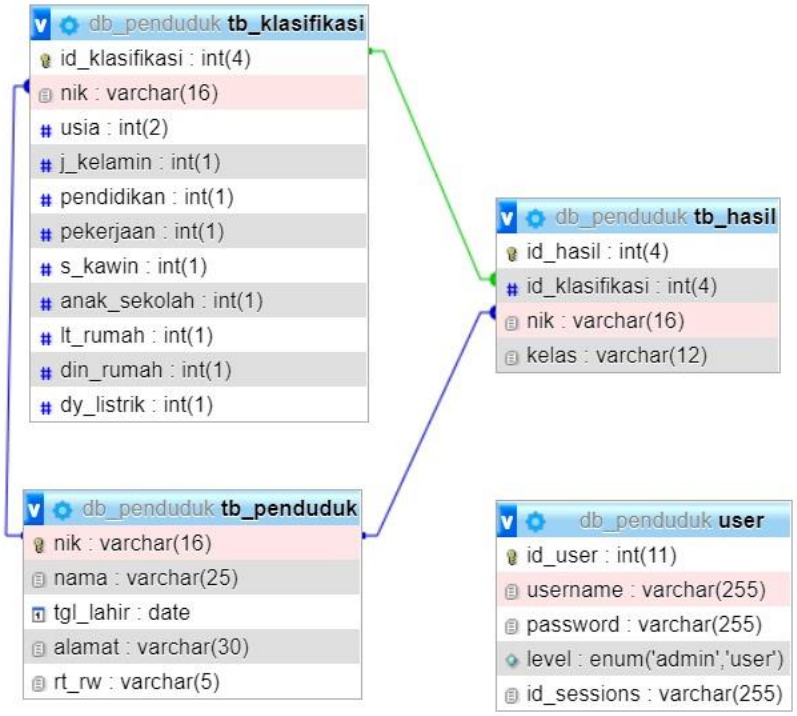
GAMBAR: 4.4. Struktur Tabel Hasil

5. Implementasi tabel user



GAMBAR: 4. 5. Struktur Tabel User

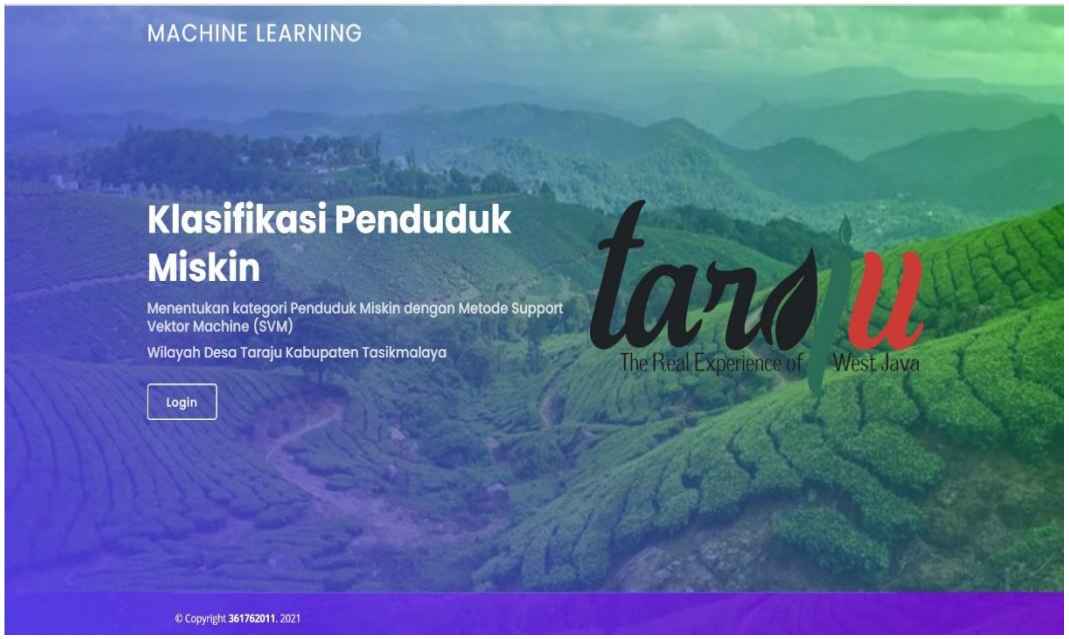
6. Implementasi relasi antar tabel



GAMBAR: 4.6. Relasi Tabel

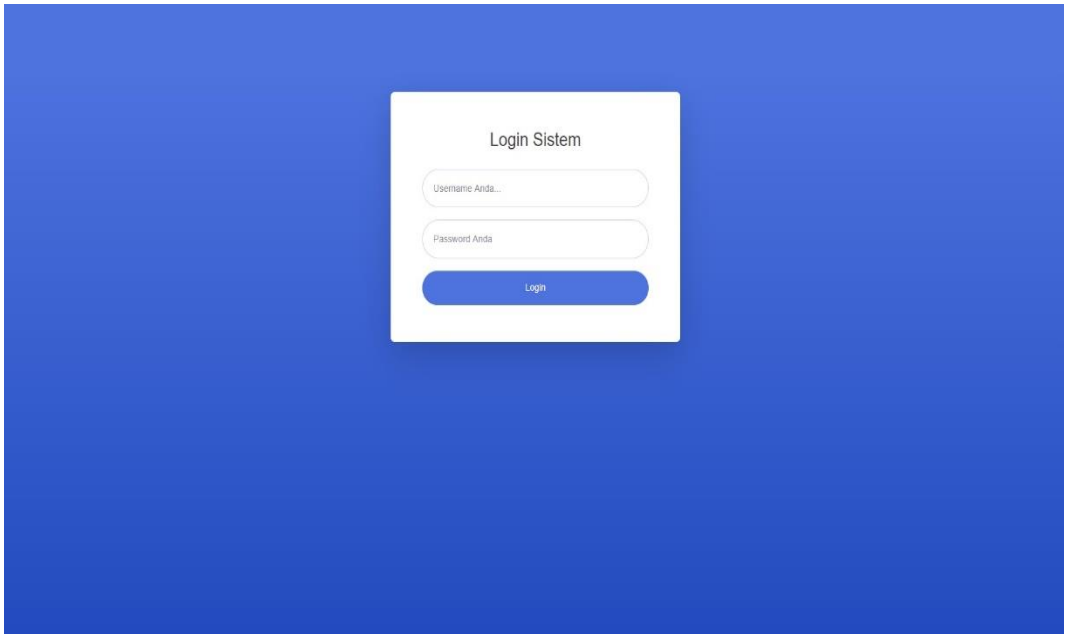
4.1.4. Implementasi Antarmuka *(Interface)*

1. Implementasi antarmuka halaman utama



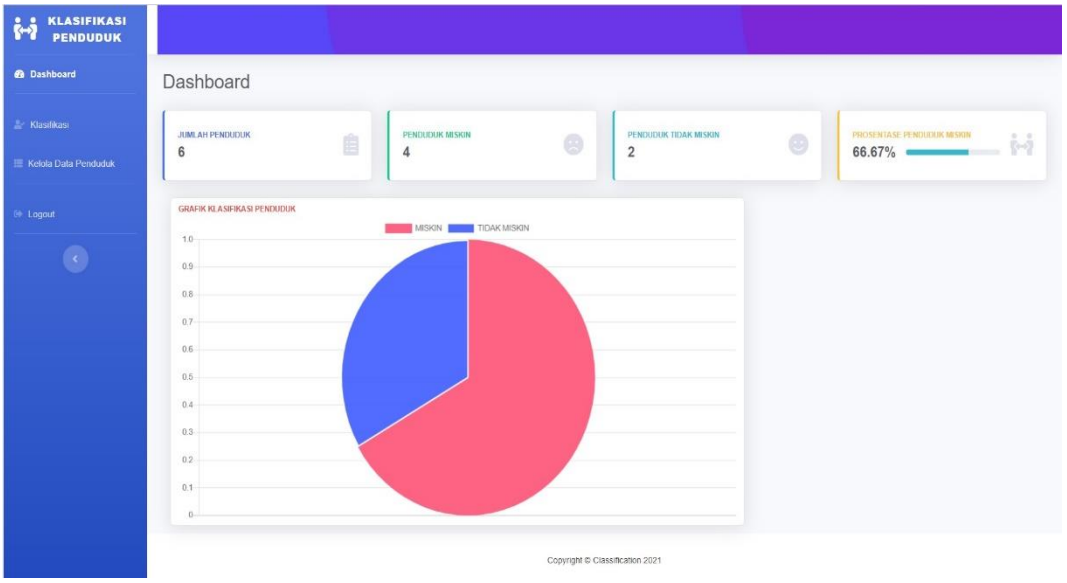
GAMBAR: 4.7. Antarmuka Halaman Utama

2. Implementasi antarmuka halaman login



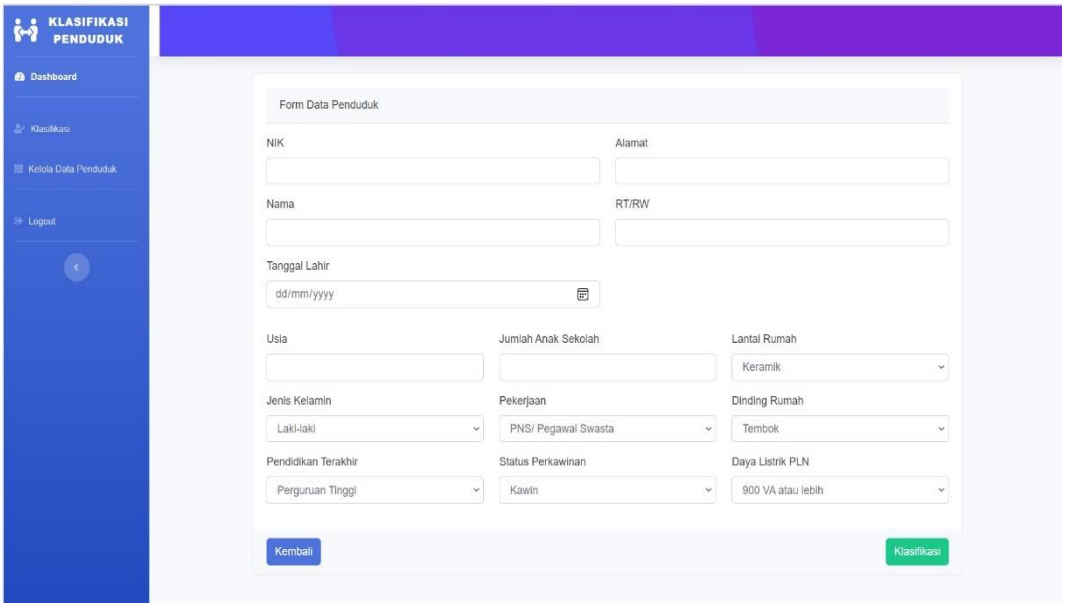
GAMBAR: 4.8. Antarmuka Halaman Login

3. Implementasi antarmuka halaman dashboard



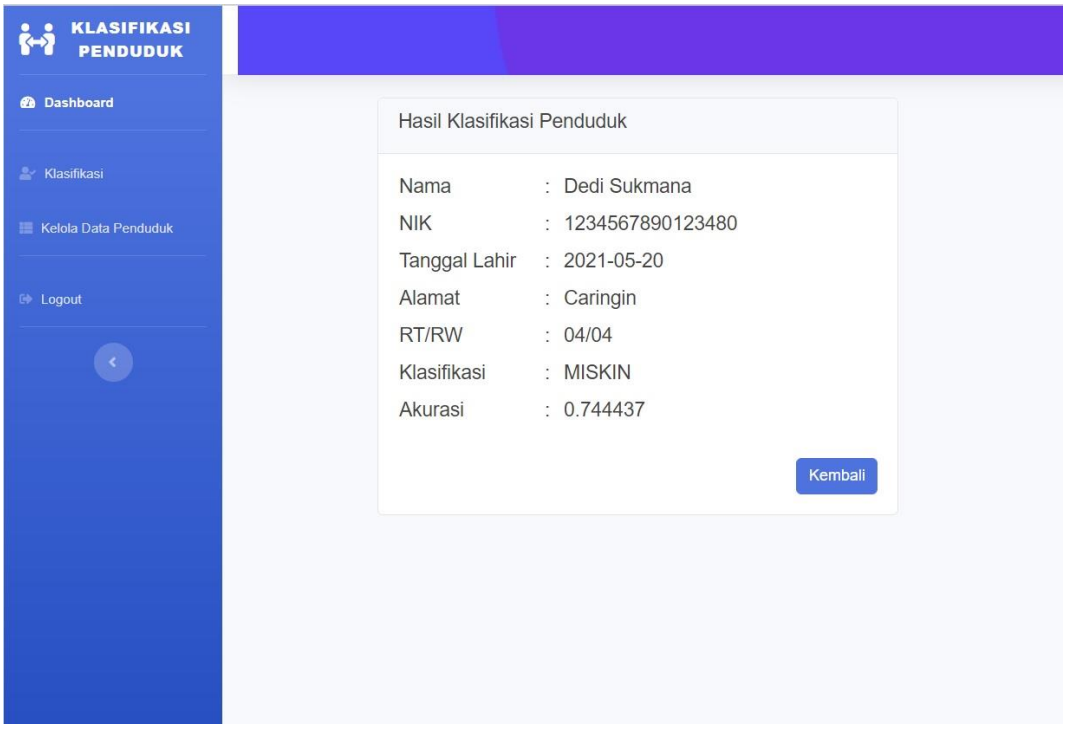
GAMBAR: 4.9. Antarmuka Halaman Dashboard

4. Implementasi antarmuka halaman form klasifikasi



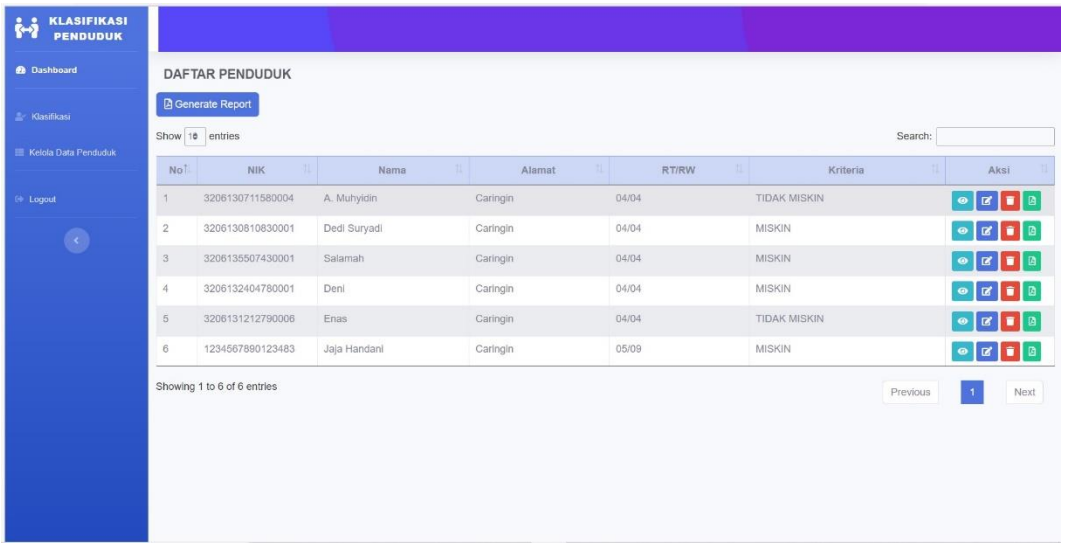
GAMBAR: 4. 10. Antarmuka Halaman Form Klasifikasi

5. Implementasi antarmuka halaman hasil klasifikasi



GAMBAR: 4.11. Antarmuka Halaman Hasil Klasifikasi

6. Implementasi antarmuka halaman Kelola data penduduk



GAMBAR: 4. 12. Antarmuka Halaman Kelola Data Penduduk

4.2. Uji Coba (*Testing*)

Pada tahap ini dilakukan uji coba atau pengujian aplikasi yang sudah dibangun. Pengujian dimaksudkan untuk mengetahui apakah fungsi-fungsi, masukan (*input*), dan keluaran (*output*) dari perangkat lunak sesuai dengan spesifikasi yang diharapkan.

Metode atau pendekatan pengujian yang digunakan dalam tahapan ini adalah dengan *Black-Box Testing* (pengujian kotak hitam) yang merupakan pengujian perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program.

Berikut adalah hasil dari pengujian menggunakan metode *Blackbox Testing* :

4.2.1. Pengujian Login dan Logout User

TABEL: 4.1. Tabel Pengujian Login dan Logout User

No	Kasus Uji	Masukan/ Aksi	Hasil yang diharapkan	Hasil didapat
1.	Menguji login jika username dan password tidak diisi	Username : tidak diisi Password : tidak diisi	Menampilkan pesan username dan password harus diisi	Sesuai
2.	Menguji login jika username diisi dan password tidak diisi	Username : diisi Password : tidak diisi	Menampilkan pesan password harus diisi	Sesuai
3.	Menguji login jika username tidak diisi dan password diisi	Username : tidak diisi Password : diisi	Menampilkan pesan username harus diisi	Sesuai
4.	Menguji login jika username dan password salah	Username : salah Password : salah	Menampilkan pesan username dan password salah	Sesuai
5.	Menguji login jika username benar dan password salah	Username : benar Password : salah	Menampilkan pesan username dan password salah	Sesuai

TABEL: 4.1. Tabel Pengujian Login dan Logout User (Lanjutan)

No	Kasus Uji	Masukan/ Aksi	Hasil yang diharapkan	Hasil didapat
6.	Menguji login jika username salah dan password benar	Username : salah Password : benar	Menampilkan pesan username dan password salah	Sesuai
7.	Menguji login jika username dan password benar	Username : benar Password : benar	Menuju halaman dashboard	Sesuai
8.	Menguji logout	Menu Logout di klik	Keluar system Menuju halaman Depan	Sesuai

4.2.2. Pengujian Proses Klasifikasi

TABEL: 4.2. Tabel Pengujian Proses Klasifikasi

No	Kasus Uji	Masukan/ Aksi	Hasil yang diharapkan	Hasil didapat
1.	Menguji proses klasifikasi untuk menu klasifikasi	Menu klasifikasi di klik	Menuju halaman form klasifikasi	Sesuai
2.	Menguji form klasifikasi kolom NIK	NIK : tidak diisi	Menampilkan pesan harus diisi	Sesuai
3.	Menguji form klasifikasi kolom NIK	NIK : diisi huruf	Menampilkan pesan hanya boleh diisi angka	Sesuai
4.	Menguji form klasifikasi kolom NIK	NIK : diisi kombinasi huruf dan angka	Menampilkan pesan hanya boleh diisi angka	Sesuai
5.	Menguji form klasifikasi kolom NIK	NIK : diisi angka kurang dari 16 karakter	Menampilkan pesan harus diisi 16 karakter	Sesuai
6.	Menguji form klasifikasi kolom NIK	NIK : diisi angka lebih dari 16 karakter	Menampilkan pesan harus diisi 16 karakter	Sesuai
7.	Menguji form klasifikasi kolom Nama	Nama : tidak diisi	Menampilkan pesan Nama harus diisi	Sesuai
8.	Menguji form klasifikasi kolom Alamat	Alamat : tidak diisi	Menampilkan pesan Alamat harus diisi	Sesuai

TABEL: 4.2. Tabel Pengujian Proses Klasifikasi (Lanjutan)

No	Kasus Uji	Masukan/ Aksi	Hasil yang diharapkan	Hasil didapat
9.	Menguji form klasifikasi kolom RT/RW	RT/RW : tidak diisi	Menampilkan pesan RT/RW harus diisi	Sesuai
10.	Menguji form klasifikasi kolom Tanggal Lahir	Tanggal Lahir : tidak diisi	Menampilkan pesan Tanggal Lahir harus diisi	Sesuai
11.	Menguji form klasifikasi kolom Jumlah anak sekolah	Jumlah anak sekolah : tidak diisi	Menampilkan pesan Jumlah anak sekolah harus diisi	Sesuai
12.	Menguji proses klasifikasi untuk button klasifikasi	Button klasifikasi di klik	Menuju halaman Hasil klasifikasi	Sesuai
13.	Menguji proses klasifikasi untuk button kembali	Button kembali di klik	Menuju halaman Kelola Data Penduduk	Sesuai

4.2.3. Pengujian Kelola Data Penduduk

TABEL: 4.3. Tabel Pengujian Kelola Data Penduduk

No	Kasus Uji	Masukan/ Aksi	Hasil yang diharapkan	Hasil didapat
1.	Menguji Kelola data penduduk untuk menu Kelola data penduduk	Menu Kelola data penduduk di klik	Menuju halaman Kelola data penduduk	Sesuai
2.	Menguji Kelola data penduduk untuk fungsi melihat data	Icon lihat di klik	Menuju halaman detail data penduduk	Sesuai
3.	Menguji Kelola data penduduk untuk fungsi mengubah data	Icon Ubah di klik	Menuju halaman form ubah data penduduk	Sesuai
4.	Menguji Kelola data penduduk untuk fungsi menghapus data	Icon hapus di klik	Menampilkan pesan konfirmasi yakin akan dihapus	Sesuai
5.	Menguji Kelola data penduduk untuk fungsi konfirmasi menghapus data	button Hapus data di klik	Record data terhapus	Sesuai

TABEL: 4.3. Tabel Pengujian Kelola Data Penduduk (Lanjutan)

No	Kasus Uji	Masukan/ Aksi	Hasil yang diharapkan	Hasil didapat
6.	Menguji Kelola data penduduk untuk fungsi konfirmasi menghapus data	button Cancel di klik	Tidak terjadi apa-apa dan Kembali ke halaman Kelola data penduduk	Sesuai
7.	Menguji Kelola data penduduk untuk fungsi mencetak data	Icon print di klik	Menampilkan file PDF data penduduk yang dipilih	Sesuai
8.	Menguji Kelola data penduduk untuk fungsi mencari data	Mengisi keyword pada kolom search	Menampilkan data penduduk yang sesuai dengan keyword	Sesuai

BAB V

PENUTUP

5.1. Kesimpulan

Dari penelitian yang telah dilakukan yaitu klasifikasi penduduk miskin di wilayah Desa Taraju Kabupaten Tasikmalaya dengan machine learning menggunakan metode *Support Vektor Machine* (SVM) didapat kesimpulan sebagai berikut:

1. Dengan dibangunnya aplikasi machine learning untuk klasifikasi penduduk miskin, Pemerintah Desa Taraju Kabupaten Tasikmalaya mudah, cepat dan tepat dalam mendapatkan data penduduk miskin di wilayahnya. Karena dengan adanya aplikasi, untuk penentuan klasifikasi penduduk miskin langsung didapat dengan waktu yang singkat dan tepat sesuai dengan data penduduk sebenarnya.
2. Penggunaan metode *Support Vektor Machine* (SVM) untuk klasifikasi penduduk miskin hasil yang didapatkan akurat dan obyektif, karena semakin banyak data penduduk yang dimasukan maka akan menambah tingkat keakuratan hasilnya.

5.2. Saran

Penelitian ini masih banyak kekurangan untuk peningkatan yang dapat digunakan di masa mendatang penulis mempunyai beberapa saran, antara lain :

1. Aplikasi machine learning ini dapat ditambahkan fitur-fitur untuk menunjang kebutuhan pengguna dalam mengelola hasil klasifikasi.
2. Aplikasi machine learning ini dapat diintegrasikan dengan aplikasi yang digunakan administrasi desa, misalnya untuk keperluan surat-surat keterangan penduduk.
3. Untuk meningkatkan lagi keakuratan dan obyektifitas hasil klasifikasi penduduk perlu ditambah lagi variable-variabel yang lebih spesifik, misalnya kepemilikan rumah, biaya yang dikeluarkan perbulan dan sebagainya.

DAFTAR PUSTAKA

Hidayatullah, P. and Kawistara, K. J. (2017) *Pemrograman WEB*. Revisi. Bandung: Informatika Bandung.

Jacobson, I. (1992) *Object-oriented software engineering: a use case driven approach*. Addison-Wesley.

Mahmudah, K. *et al.* (2016) 'Pengembangan Sistem Informasi Dissolved Gas Analysis (Dga)', 12, pp. 54–59.

Mitchell, T. . (2017) *Machine learning*, *Machine Learning*. New York: McGraw-Hill.

Munawarah, R., Soesanto, O. and Faisal, M. R. (2016) 'Penerapan Metode Support Vector Machine Pada Diagnosa Hepatitis', *Kumpulan jurnaL Ilmu Komputer (KLIK)*, Volume 04,(February), pp. 103–113. doi: 10.20527/klik.v3i1.39.

Nasution, D. Q. (2018) 'Studi tentang kemiskinan di Kabupaten Batang Hari dan Kabupaten Muaro Jambi', 7(2), pp. 79–90.

Nurhayati, Busman and Iswara, R. (2019) 'Pengembangan Algoritma Unsupervised Learning Technique Pada Big Data Analysis di Media Sosial sebagai media promosi Online Bagi Masyarakat', *Jurnal Teknik Informatika*, 12(1), pp. 79–96. doi: 10.15408/jti.v12i1.11342.

Parapat, I. M., Furqon, M. T. and Sutrisno (2018) 'Penerapan Metode Support Vector Machine (SVM) Pada Klasifikasi Penyimpangan Tumbuh Kembang Anak', *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, 2(10), pp. 3163–3169.

Pratama, A., Wihandika, R. C. and Ratnawati, D. E. (2018) 'Implementasi Algoritme Support Vector Machine (SVM) untuk Prediksi Ketepatan Waktu Kelulusan Mahasiswa', *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, 2(March), pp. 1704–1708.

Pratiwi, E. S. (2019) *Perkembangan Tingkat Kemiskinan Provinsi Jawa Barat Maret 2019, Badan Pusat Statistik Provinsi Jawa Barat*. Edited by Y. Hidayat, J. Trisnadi, and D. Mulyahati. Badan Pusat Statistik Provinsi Jawa Barat. Available at: https://jabar.bps.go.id/publication/2019/08/29/a494f46c41b1efada30b448b/perkembangan-tingkat-kemiskinan-provinsi-jawa-barat-maret-2019.

Puspitasari, A. M., Ratnawati, D. E. and Widodo, A. W. (2018) 'Klasifikasi Penyakit Gigi Dan Mulut Menggunakan Metode Support Vector Machine', *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*, 2(2), pp. 802–810.

Ritonga, A. S. and Purwaningsih, E. S. (2018) 'Penerapan Metode Support Vector Machine (SVM) Dalam Klasifikasi Kualitas Pengelasan Smaw (Shield Metal Arc Welding)', *Ilmiah Edutic*, 5(1), pp. 17–25.

Rosa A, S. and Shalahuddin, M. (2019) *Rekayasa Perangkat Lunak Terstruktur dan Berorientasi Objek*. 2nd-Revisi edn. Edited by S. Rosa A and M. Shalahuddin. Bandung: Informatika Bandung.

Suyanto (2018) *Machine Learning Tingkat Dasar dan Lanjut*. 1st edn. Bandung: INFORMATIKA Bandung.

LAMPIRAN

```
<?php
defined('BASEPATH') OR exit('No direct script access allowed');

use Phpml\Classification\SVC;
use Phpml\SupportVectorMachine\Kernel;

class Penduduk extends CI_Controller {

    public function __construct()
    {
        parent::__construct();
        $this->load->model('Penduduk_model');
        $this->load->library('form_validation');
    }

    public function index()
    {
        $data['judul'] = 'Daftar Penduduk';
        $data['penduduk'] = $this->Penduduk_model->getAllPenduduk();
        if( $this->input->post('keyword') ) {
            $data['penduduk'] = $this->Penduduk_model-
>cariDataPenduduk();
        }
        $this->load->view('templates_administrator/header', $data);
        $this->load->view('templates_administrator/sidebar');
        $this->load->view('penduduk/index', $data);
        $this->load->view('templates_administrator/footer');
    }

    public function tambah()
    {
        $data['judul'] = 'Form Tambah Data Penduduk';
        if ($this->form_validation->run() == FALSE) {
            $this->load-
>view('templates_administrator/header', $data);
            $this->load->view('templates_administrator/sidebar');
            $this->load->view('penduduk/tambah');
            $this->load->view('templates_administrator/footer');
        } else {
            $this->Penduduk_model->hitung();
            $this->session->set_flashdata('flash', 'Ditambahkan');
            redirect('penduduk');
        }
        $this->Penduduk_model->hitung();
    }
```

```
    public function hapus($id)
    {
        $this->Penduduk_model->hapusDataPenduduk($id);
        $this->session->set_flashdata('flash', 'Dihapus');
        redirect('penduduk');
    }

    public function detail($nik)
    {
        $data['judul'] = 'Detail Data Penduduk';
        $data['penduduk'] = $this->Penduduk_model-
>getPendudukById($nik);
        $this->load->view('templates_administrator/header', $data);
        $this->load->view('templates_administrator/sidebar');
        $this->load->view('penduduk/detail', $data);
        $this->load->view('templates_administrator/footer');
    }

    public function hasil()
    {
        $this->form_validation-
         >set_rules('nik', 'NIK', 'trim|required|numeric|min_length[1
         6]|max_length[16]|is_unique[tb_penduduk.nik]');
        $this->form_validation-
>set_rules('nama', 'Nama', 'required');
        $this->form_validation->set_rules ('alamat' , 'Alamat'
         , 'required') ;
        $this->form_validation-
         >set_rules('tgl_lahir', 'Tanggal Lahir', 'required');
        $this->form_validation-
>set_rules('rt_rw', 'RT/RW', 'required');
        $this->form_validation-
       >set_rules('anak_sekolah', 'Jumlah anak sekolah', 'required|n
       umeric');

        $data1['judul'] = 'Hasil Klasifikasi Penduduk';
        if ($this->form_validation->run()==FALSE)
        {
            $this->load-
>view('templates_administrator/header', $data1);
            $this->load->view('templates_administrator/sidebar');
            $this->load->view('penduduk/tambah');
            $this->load->view('templates_administrator/footer');
        }
        else
        {
            $data['nik'] = $this->input->post('nik', true);
            $data['nama'] = $this->input->post('nama', true);
            $data['tgl_lahir'] = $this->input-
>post('tgl_lahir', true);
            $data['alamat'] = $this->input->post('alamat', true);
            $data['rt_rw'] = $this->input->post('rt_rw', true);
```

```
            $data['kelas'] = $this->Penduduk_model->hitung();
            $data['akurasi']= $this->Penduduk_model-
>hitungAccuracy();
            $this->load-
>view('templates_administrator/header', $data1);
            $this->load->view('templates_administrator/sidebar');
            $this->load->view('penduduk/hasil', $data);
            $this->load->view('templates_administrator/footer');
            $this->Penduduk_model->simpan();
        }
    }

    public function ubah($nik)
    {
        $data['judul'] = 'Form Ubah Data Penduduk';
        $data['penduduk'] = $this->Penduduk_model-
>getPendudukById($nik);

        $this->form_validation-
>set_rules('nama', 'Nama', 'required');
        $this->form_validation->set_rules('nik', 'NIK', 'required');
        $this->form_validation->set_rules ('tgl_lahir'
        , 'Tanggal Lahir' , 'required');
        $this->form_validation->set_rules ('alamat', 'Alamat'
        , 'required');

        if ($this->form_validation->run() == FALSE) {
            $this->load-
>view('templates_administrator/header', $data);
            $this->load->view('templates_administrator/sidebar');
            $this->load->view('penduduk/ubah', $data);
            $this->load->view('templates_administrator/footer');
        } else {
            $this->Penduduk_model->ubahDataPenduduk();
            $this->session->set_flashdata('flash', 'Diubah');
            redirect('penduduk');
        }
    }

    public function laporan()
    {
        $data['penduduk'] = $this->Penduduk_model->getAllPenduduk();
        $this->load->library('pdf');
        $this->pdf->setPaper('A4', 'potrait');
        $this->pdf->filename = "Daftar_Penduduk.pdf";
        $this->pdf->load_view('penduduk/daftar_penduduk', $data);
    }

    public function laporanhasil($nik)
    {
        $data['penduduk'] = $this->Penduduk_model->getPendudukById
       ($nik);
```

```
        $this->load->library('pdf');
        $this->pdf->setPaper('A4', 'potrait');
        $this->pdf->filename = "Hasil_Klasifikasi.pdf";
        $this->pdf->load_view('penduduk/lap_hasil', $data);
    }

   public function kelas()
    {
        $data['kelas'] = $this->Penduduk_model->jml_kelas();
        $this->load->view('penduduk/test1', $data);
        var_dump($data);
    }
}
```

```
<?php
defined('BASEPATH') OR exit('No direct script access allowed');
use Phpml\Classification\SVC;
use Phpml\SupportVectorMachine\Kernel;
class Penduduk_model extends CI_model {
    public function getAllPenduduk()
    {
        $this->db->select('*');
        return $this->db->from('tb_penduduk')
        ->join('tb_hasil', 'tb_hasil.nik = tb_penduduk.nik')
        ->get()
        ->result_array();
    }
    public function getPendudukById($nik) {
        $this->db->select('*');
        $this->db->from('tb_penduduk');
        $this->db-
>join('tb_hasil', 'tb_hasil.nik = tb_penduduk.nik', 'left');
        $this->db->where('tb_hasil.nik', $nik);
        return $this->db->get()->row_array();
      }
    public function tambahDataPenduduk()
    {
        $data = [
            "nik" => $this->input->post('nik', true),
            "nama" => $this->input->post('nama', true),
            "tgl_lahir" => $this->input->post('tgl_lahir', true),
            "alamat" => $this->input->post('alamat', true),
            "rt_rw" => $this->input->post('rt_rw', true)
        ];
        $this->db->insert('tb_penduduk', $data);
    }


    public function hapusDataPenduduk($nik)
    {
        $this->db->where('nik', $nik);
        $this->db->delete('tb_penduduk');
    }
    public function ubahDataPenduduk()
    {
        $data = [
            "nik" => $this->input->post('nik', true),
            "nama" => $this->input->post('nama', true),
            "tgl_lahir" => $this->input->post('tgl_lahir', true),
            "alamat" => $this->input->post('alamat', true),
            "rt_rw" => $this->input->post('rt_rw', true)
        ];
        $this->db->where('nik', $this->input->post('nik'));
        $this->db->update('tb_penduduk', $data);
    }

    public function cariDataPenduduk()
```

```
    {
        $keyword = $this->input->post('keyword', true);
        $this->db->like('nik', $keyword);
        $this->db->or_like('nama', $keyword);
        $this->db->or_like('tgl_lahir', $keyword);
        $this->db->or_like('alamat', $keyword);
        return $this->db->get('tb_penduduk')->result_array();
    }
    public function getAllSamples()
    {
        $this->db->select(
                            'j_kelamin, pendidikan, pekerjaan, s_kawin
, anak_sekolah, lt_rumah, din_rumah, dy_listrik, sumber_air'
                            );
        return $this->db->get('tb_klasifikasi')->result_array();
    }
    public function getAllLabels()
    {
        $this->db->select('kelas');
        return $this->db->get('tb_hasil')->result_array();
    }
    public function tambahDataVariabel()
    {
       $data = [
            "j_kelamin" => $this->input->post('j_kelamin', true),
            "pendidikan" => $this->input->post('pendidikan', true),
            "pekerjaan" => $this->input->post('pekerjaan', true),
            "s_kawin" => $this->input->post('s_kawin', true),
            "anak_sekolah" => $this->input-
>post('anak_sekolah', true),
            "lt_rumah" => $this->input->post('lt_rumah', true),
            "din_rumah" => $this->input->post('din_rumah', true),
            "dy_listrik" => $this->input->post('dy_listrik', true),
            "sumber_air" => $this->input->post('sumber_air', true)
        ];
           return array($data);
    }

     public function hitung()
    {
        $arr = ["j_kelamin","pendidikan","pekerjaan","s_kawin","anak
_sekolah",
                "lt_rumah","din_rumah","dy_listrik", "sumber_air"
               ];
        $klasifikasi = $this->Penduduk_model->getAllSamples();
        $samples = [];
        $sample = [];


        foreach ($klasifikasi as $row) {
                for ($i=0;$i<count($arr);$i++) {
                    array_push($sample, $row[$arr[$i]]);
```

```
                }
            array_push($samples, $sample);
            $sample = [];
        }

        $arr1 = ["kelas"];
        $kelas = $this->Penduduk_model->getAllLabels();
        $labels = [];
        foreach ($kelas as $row) {
                for ($i=0;$i<count($arr1);$i++) {
                    array_push($labels, $row[$arr1[$i]]);
                }};

        $classifier = new SVC(Kernel::LINEAR, $cost = 1000);

        $classifier->train($samples, $labels);
        $arr2 = ["j_kelamin","pendidikan","pekerjaan","s_kawin","ana
k_sekolah",
                 "lt_rumah","din_rumah","dy_listrik", "sumber_air"
                ];
        $input  = $this->Penduduk_model->tambahDataVariabel();

        $data = [];
        foreach ($input as $row) {
                for ($i=0;$i<count($arr2);$i++) {
                    array_push($data, $row[$arr2[$i]]);
                }};
            $result = $classifier->predict($data);
            return $result;
    }

    public function hitungAccuracy()
    {
        $arr = ["j_kelamin","pendidikan","pekerjaan","s_kawin","anak
_sekolah",
                 "lt_rumah","din_rumah","dy_listrik", "sumber_air"
                ];
        $klasifikasi = $this->Penduduk_model->getAllSamples();
        $samples = [];
        $sample = [];
        foreach ($klasifikasi as $row) {
                for ($i=0;$i<count($arr);$i++) {
                    array_push($sample, $row[$arr[$i]]);
                }
            array_push($samples, $sample);
            $sample = [];
        }
         $arr1 = ["kelas"];
        $kelas = $this->Penduduk_model->getAllLabels();
        $labels = [];
        foreach ($kelas as $row) {
                for ($i=0;$i<count($arr1);$i++) {
```

```
                    array_push($labels, $row[$arr1[$i]]);
                }};

        $classifier = new SVC(
                    Kernel::LINEAR, // $kernel
                    1.0,            // $cost
                    3,              // $degree
                    null,           // $gamma
                   0.0,            // $coef0
                    0.001,          // $tolerance
                    100,            // $cacheSize
                    true,           // $shrinking
                    true            // $probabilityEstimates, set to
 true
                    );
        $classifier->train($samples, $labels);
        $arr2 = ["j_kelamin","pendidikan","pekerjaan","s_kawin","ana
k_sekolah",
                 "lt_rumah","din_rumah","dy_listrik", "sumber_air"
                ];
        $input  = $this->Penduduk_model->tambahDataVariabel();

        $data = [];
        foreach ($input as $row) {
                for ($i=0;$i<count($arr2);$i++) {
                    array_push($data, $row[$arr2[$i]]);
                }};
        $accuracy = $classifier->predictProbability($data);
        $akurasi = max($accuracy);
        return $akurasi;
    }


    public function simpan() {
     $penduduk = [
            "nik" => $this->input->post('nik', true),
            "nama" => $this->input->post('nama', true),
            "tgl_lahir" => $this->input->post('tgl_lahir', true),
            "alamat" => $this->input->post('alamat', true),
            "rt_rw" => $this->input->post('rt_rw', true)
        ];

        $var_klasifikasi = [
            "nik" => $this->input->post('nik', true),
            "j_kelamin" => $this->input->post('j_kelamin', true),
            "pendidikan" => $this->input->post('pendidikan', true),
            "pekerjaan" => $this->input->post('pekerjaan', true),
            "s_kawin" => $this->input->post('s_kawin', true),
            "anak_sekolah" => $this->input-
>post('anak_sekolah', true),
            "lt_rumah" => $this->input->post('lt_rumah', true),
            "din_rumah" => $this->input->post('din_rumah', true),
```

```
        "dy_listrik" => $this->input->post('dy_listrik', true),
        "sumber_air" => $this->input->post('sumber_air', true)
    ];
    $nik1 =  $this->input->post('nik', true);
    $hasil = $this->Penduduk_model->hitung();


    $this->db->insert('tb_penduduk', $penduduk);
    $this->db->insert('tb_klasifikasi', $var_klasifikasi);
    $last_insert_id = $this->db->insert_id();
    $New_hasil = ["kelas" => $hasil,
                  "id_klasifikasi" => $last_insert_id,
                  "nik" => $nik1
                 ];
    $this->db->insert('tb_hasil', $New_hasil);
}

public function hasil() {
   $hasil = $this->Penduduk_model->hitung();
    $data1 = [
        "nik" => $this->input->post('nik', true),
        "nama" => $this->input->post('nama', true),
        "tgl_lahir" => $this->input->post('tgl_lahir', true),
       "alamat" => $this->input->post('alamat', true),
        "rt_rw" => $this->input->post('rt_rw', true),
        "kelas" => $hasil
    ];
    return array($data1);
}

public function count_pnddk() {
    $count = $this->db->count_all_results('tb_penduduk');
    return $count;
}
public function count_Miskin() {
    $this->db->where('kelas', 'MISKIN');
    $this->db->from('tb_hasil');
    $miskin = $this->db->count_all_results();
    return $miskin;
}
public function count_tidakMiskin() {
    $this->db->where('kelas', 'TIDAK MISKIN');
    $this->db->from('tb_hasil');
    $tdkmiskin = $this->db->count_all_results();
    return $tdkmiskin;
}
public function prosentase() {
    $miskin = $this->Penduduk_model->count_Miskin();
    $tdk_miskin = $this->Penduduk_model->count_tidakMiskin();
    $penduduk = $this->Penduduk_model->count_pnddk();
    $hasil = $miskin / $penduduk * 100 ;
    $prosentase = number_format($hasil,2);
```

```
        return $prosentase;
    }
    function jml_kelas()
    {
        $this->db->group_by('kelas');
        $this->db->select('kelas');
        $this->db->select("count(*) as total");
        return $this->db->from('tb_hasil')
          ->get()
          ->result();
  }
}
```