BAB II

LANDASAN TEORI

2.1 Tinjauan Terhadap Penelitian Terkait

Dalam penelitian ini penulis memaparkan beberapa penelitian terdahulu yang sesuai dengan permasalahan yang diteliti :

1. Klasifikasi rumah tangga miskin menggunakan *Ordinal Class Classifier* (Effendy & Purbandini, 2018). Penelitian ini menggunakan 14 variabel yaitu luas lantai, jenis atap, jenis dinding, jenis lantai, tempat buang air besar, sumber penerangan, sumber air minum, bahan bakar, konsumsi daging, konsumsi pakaian, konsumsi makanan per hari, pengobatan, ijazah terakhir, kepemilikan aset dalam mengklasifikasikan rumah tangga miskin. hasil dari penelitian ini menunjukan bahwa *Ordinal Class Classifier* memiliki *accuracy* sebesar 90,5437 %, *precision* sebesar 0,919, dan *recall* 0,905.
2. Klasifikasi rumah tangga miskin di Kecamatan Kaubun Tahun 2020 dengan menggunakan Metode *Improved Chi-Square Automatic Interaction Detection* (Yuliasari et al., 2021). Pada penelitian ini menggunakan variabel dependen status rumah tangga sedangkan variabel independen terdiri dari luas lantai tempat tinggal, jenis lantai tempat tinggal, jenis dinding tempat tinggal, sumber penerangan tempat tinggal, sumber air bersih, kepemilikan jamban/WC, bahan bakar memasak, mengkonsumsi daging/ayam/susu dalam seminggu, kemampuan membeli pakaian baru dalam satu tahun, frekuensi makan dalam satu hari, tempat pengobatan, penghasilan kepala rumah tangga, pendidikan tertinggi kepala rumah tangga, kepemilikan

aset/tabungan. Hasil dari penelitian ini adalah terdapat 10 rumah tangga yang berstatus miskin, dan rumah tangga tergolong miskin apabila frekuensi makanya kurang dari 3 kali sehari, serta hasil akurasi terbaik menggunakan proporsi data latih 60% dan data pengujian sebesar 40 %.

3. Algoritma *CHAID* pada Klasifikasi rumah tangga miskin kota Palopo (Hastuti & Muzaini, 2021). Pada penelitian ini menggunakan variabel dependen skala data dan variabel independen terdiri dari status kepala rumah tangga, pendapatan kepala rumah tangga, status pekerjaan, pendidikan terakhir kepala rumah tangga, luas lantai rumah, jenis lantai rumah, jenis atap rumah, jenis dinding rumah, status kepemilikan rumah, sumber penerangan rumah, sumber air untuk minum, tempat buang air besar, dan tempat buang sampah. Hasil analisis *CHAID* menunjukkan bahwa dari 13 variabel yang telah diolah hanya terdapat 3 variabel yang berasosiasi atau memiliki keterkaitan dengan status kemiskinan yaitu pendapatan per bulan, jenis lantai rumah, dan status pekerjaan. Tingkat ketepatan model *CHAID* dalam mengklasifikasikan rumah tangga adalah sebesar 98.5 %.
4. Implementasi *Support Vector Machine* pada klasifikasi penduduk miskin wilayah Desa Taraju Kabupaten Tasikmalaya (Chazar et al., 2022). Penelitian ini menggunakan variabel jenis kelamin, pendidikan, pekerjaan, status perkawinan, tanggungan anak, lantai rumah, dinding rumah, daya listrik, sumber air dalam mengklasifikasikan penduduk miskin. Hasil dari penelitian ini adalah penerapan aplikasi *Machine Learning* menggunakan

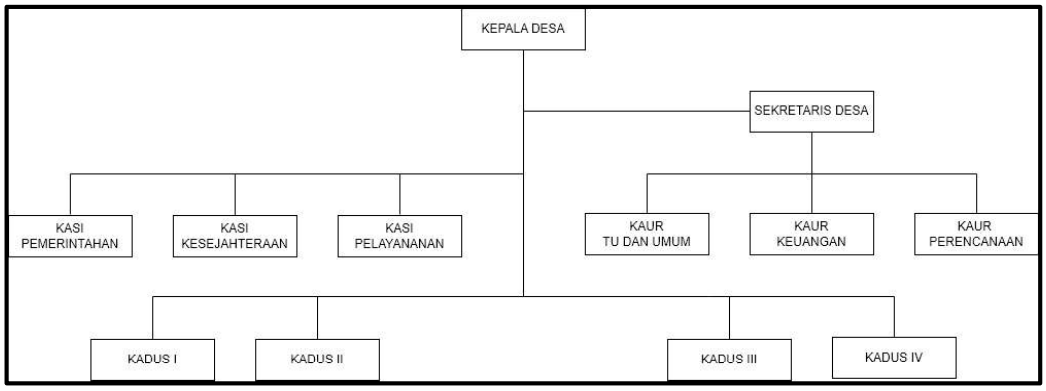
metode *Support Vector Machine* untuk klasifikasi penduduk miskin berdasarkan variabel – variabel yang telah ditentukan.

5. *Support Vector Machine* (SVM) *Optimization Using Grid Search and Unigram to Improve E-Commerce Review Accuracy* (Sulistiana & Muslim, 2020). Pada penelitian ini metode *Unigram* digunakan untuk ekstraksi ciri dan *Grid Search* digunakan sebagai optimasi parameter untuk meningkatkan akurasi SVM. Hasil penelitian ini menunjukan penggunaan *Unigram* dan *Grid Search* meningkatkan akurasi *review* Amazon sebesar 26,4 % dan *review* Lazada sebesar 4,26 %.

2.2 Desa Rigi

Desa Rigi merupakan salah satu desa yang berada di kecamatan Boawae, Kabupaten Nagekeo, Provinsi Nusa Tenggara Timur. Desa Rigi terletak bagian barat dari pusat pemerintahan Kecamatan Boawae. Batas-batas wilayah Desa Rigi adalah disebelah utara berbatasan dengan Kelurahan Olakile, di sebelah barat berbatasan dengan Desa Solo, di sebelah selatan berbatasan dengan Desa Leguderu, dan di sebelah timur berbatasan dengan Kelurahan Nagesapadhi.

Jarak Desa Rigi ke Ibu Kota Kabupaten Nagekeo ± 22 km dapat ditempuh dengan waktu ± 1 jam apabila menggunakan kendaraan bermotor dan jarak Desa Rigi ke Ibu Kota Kecamatan Boawae ± 4 km dapat ditempuh dengan waktu ± 15 menit apabila menggunakan kendaraan bermotor. Luas Desa Rigi ± 10.05 km2.



Gambar 2.1 Struktur Organisasi Desa Rigi

Struktur organisasi pemerintah Desa Rigi seperti yang terlihat pada gambar 2.1 adalah sebagai berikut:

1. Kepala Desa : kepala desa dipilih langsung oleh penduduk desa dari beberapa calon yang memenuhi syarat. Untuk periode tahun 2020-2025 Desa Rigi dibawah pimpinan kepala desa atas nama Ferdinandus Pelo Mae, SH yang diangkat berdasarkan surat keputusan Bupati Nagekeo No. 216/KEP/HK/02/2019.
2. Sekretaris Desa : sekretaris Desa Rigi dipimpin oleh Bernardus Siga Dhuge yang diangkat berdasarkan surat keputusan Kepala Desa Rigi No. 05 Tahun 2019.
3. Kaur TU dan Umum : membantu sekretaris desa dalam urusan ketatausahaan seperti tata naskah, administrasi surat menyurat, arsip dan ekspedisi, penataan administrasi perangkat desa, penyediaan prasarana perangkat desa dan kantor, penyiapan rapat, pengadministrasi aset, inventaris, perjalanan dinas dan pelayanan umum sesuai dengan peraturan perundang-undangan. Kepala Urusan Tata Usaha dan Umum Desa Rigi dipimpin oleh Constantinus Lengi To yang diangkat berdasarkan surat keputusan Kepala Desa Rigi No. 09 Tahun 2019.

4. Kaur Keuangan : membantu sekretaris desa dalam urusan keuangan dan pengurusan administrasi keuangan, administrasi sumber-sumber pendapatan dan pengeluaran, verifikasi administrasi keuangan, dan administrasi penghasilan kepala desa, perangkat desa, BPD dan lembaga pemerintahan desa lainnya, sesuai dengan peraturan perundangan-undangan. Kepala Urusan Keuangan Desa Rigi dipimpin oleh Karolina Juniati Aso yang diangkat berdasarkan surat keputusan Kepala Desa Rigi No. 10 Tahun 2019.
5. Kaur Perencanaan : membantu sekretaris desa dalam urusan perencanaan program kegiatan desa dan menyusun rencana anggaran pendapatan dan belanja desa, inventaris data-data dalam rangka pembangunan, melakukan monitoring dan evaluasi program, serta penyusunan laporan sesuai dengan peraturan perundang-undangan. Kepala Urusan Perencanaan Desa Rigi dipimpin oleh Donatus Aga Azi yang diangkat berdasarkan surat keputusan Kepala Desa Rigi No. 11 Tahun 2019.
6. Kasi Pemerintahan : membantu kepala desa sebagai pelaksanaan teknis, pelaksanaan tugas operasional dan tugas lainnya sesuai dengan peraturan perundang-undangan. Melaksanakan manajemen tata praja pemerintahan, membantu sekretaris desa dalam menyusun rancangan produk-produk hukum di desa, pembinaan masalah pertanahan, pembinaan ketentraman dan ketertiban, pelaksanaan upaya perlindungan masyarakat, kependudukan, pendataan dan pengelolaan profil desa. Kepala Seksi Pemerintah Desa Rigi dipimpin oleh Veronika

Noi Bupu yang diangkat berdasarkan surat keputusan Kepala Desa Rigi No. 06 Tahun 2019.

7. Kasi Kesejahteraan : membantu kepala desa sebagai pelaksanaan teknis. Melaksanakan pembangunan sarana prasarana perdesaan, pembangunan bidang pendidikan, kesehatan dan sosialisasi serta motivasi masyarakat dalam bidang budaya, ekonomi, politik, lingkungan hidup, pemberdayaan keluarga, pemuda, olahraga dan karang taruna. Kepala Seksi Kesejahteraan Desa Rigi dipimpin oleh Lusia Goo yang diangkat berdasarkan surat keputusan Kepala Desa Rigi No. 07 Tahun 2019.
8. Kasi Pelayanan : membantu kepala desa sebagai pelaksanaan teknis sesuai dengan peraturan undang-undangan. Melaksanakan penyuluhan dan motivasi terhadap pelaksanaan hak dan kewajiban masyarakat, meningkatkan upaya partisipasi masyarakat, pelestarian nilai sosial budaya masyarakat, keagamaan dana ketenagakerjaan. Kepala Seksi Pelayanan Desa Rigi dipimpin oleh Maria Anjeliana Bupu yang diangkat berdasarkan surat keputusan Kepala Desa Rigi No. 08 Tahun 2019.
9. Kadus I : kepala wilayah Dusun I yaitu Oktavianus Meo Wea. Diangkat berdasarkan surat keputusan Kepala Desa Rigi No. 12.
10. Kadus II : kepala wilayah Dusun II yaitu Anselmus Deu Wawo. Diangkat berdasarkan surat keputusan Kepala Desa Rigi No. 13 Tahun 2019.

11. Kadus III : kepala wilayah Dusun III yaitu Robertus Soda Uma. Diangkat berdasarkan surat keputusan Kepala Desa Rigi No. 14 Tahun 2019.
12. Kadus IV : kepala wilayah Dusun IV yaitu Hendrikus Lako Ule. Diangkat berdasarkan surat keputusan Kepala Desa Rigi No.15 Tahun 2019.

2.3 Rumah Tangga

2.3.1 Konsep Rumah Tangga

Rumah tangga dapat dikategorikan kedalam dua kelompok (Samudro, 2022) yaitu rumah tangga biasa dan rumah tangga khusus.

a. Rumah tangga biasa adalah seorang atau sekelompok yang mendiami seluruh atau sebagian bagunan fisik atau sensus, dan biasanya tinggal bersama serta makan dari satu dapur.
b. Rumah tangga khusus adalah orang-orang yang tinggal di asrama, tangsi, panti asuhan, lembaga pemasyarakatan, atau rumah tahanan yang mengurus kebutuhan sehari-harinya dikelola oleh suatu yayasan atau lembaga, dan kelompok orang yang mondok dengan makan (indekos) dan berjumlah sepuluh orang atau lebih.

2.3.2 Profil Rumah Tangga Miskin

Dalam mengukur kemiskinan, Badan Pusat Statistik (BPS) menggunakan konsep kemampuan pemenuhan kebutuhan dasar (*Basic Needs Approach*) (Azis,

2021). Kemiskinan dipandang sebagai ketidakmampuan dari sisi ekonomi untuk memenuhi kebutuhan dasar makanan dan bukan makanan yang diukur menurut garis kemiskinan (makanan dan bukan makanan). Berdasarkan hal tersebut sebuah rumah tangga tergolong kedalam rumah tangga miskin apabila rata-rata pengeluaran perkapita rumah tangga berada dibawah garis kemiskinan yang telah ditetapkan.

Menurut (Santi et al., 2022) profil rumah tangga miskin di indonesia dapat dilihat dari :

1. Karakteristik sosial demografi : anggota rumah tangga lebih banyak, rata-rata umur kepala rumah tangga lebih tua, dan rata-rata lama sekolah kepala rumah tangga lebih rendah dibanding dengan rumah tangga tidak miskin.
2. Karakteristik pendidikan : tingkat buta huruf kepala rumah tangga miskin lebih tinggi dibandingkan dengan rumah tangga tidak miskin dan tingkat pendidikan kepala rumah tangga miskin lebih rendah dibanding rumah tangga tidak miskin.
3. Karakteristik ketenagakerjaan : Sebagian besar rumah tangga miskin menggantungkan hidupnya pada sektor pertanian dan pada umumnya kepala rumah tangga miskin berstatus sebagai pekerja informal.
4. Karakteristik tempat tinggal (perumahan) : rumah tangga miskin menempati luas lantai perkapita kurang dari $8m^2$, rumah tangga miskin memiliki rumah dengan jenis lantai tanah, jenis atap rumah terbuat dari ijuk/rumbia, jenis dinding terbuat dari kayu/bambu, jenis penerangan rumah listrik bukan dari PLN, tidak mampu mengakses sumber air

bersih, tidak menggunakan jamban sendiri, dan status kepemilikan rumah tempat tinggal bukan milik sendiri.

2.3.3 Variabel Kemiskinan

Dalam menentukan suatu rumah tangga tergolong kedalam rumah tangga miskin dan rumah tangga tidak miskin, Badan Pusat Statistik menggunakan 14 variabel kemiskinan (Effendy & Purbandini, 2018) yaitu:

1. Luas bangunan
2. Jenis lantai
3. Jenis dinding
4. Fasilitas buang air besar
5. Sumber air minum
6. Sumber penerangan
7. Jenis bahan bakar untuk memasak
8. Frekuensi membeli daging, ayam, dan susu dalam seminggu
9. Frekuensi makan dalam sehari
10. Jumlah stel pakain baru yang dibeli dalam setahun
11. Akses ke puskesmas atau poliklinik
12. Akses ke lapangan pekerjaan
13. Pendidikan terakhir kepala rumah tangga, dan
14. Kepemilikan aset

Dalam PSE05, suatu rumah tangga dikatakan miskin apabila :

1. Luas lantai bangunan tempat tinggalnya < $8m^2$ per orang
2. Lantai bangunan tempat tinggal terbuat dari tanah/bambu/kayu murahan

3. Dinding bangunan tempat tinggal terbuat dari bambu/rumbia/kayu berkualitas rendah atau tembok tanpa di plester
4. Tidak memiliki fasilitas buang air besar/bersama-sama rumah tangga lain menggunakan satu jamban
5. Sumber penerangan rumah tangga tidak menggunakan listrik
6. Air minum berasal dari sumur/mata air yang tidak terlindungi/sungai/air hujan
7. Bahan bakar untuk memasak sehari-hari adalah kayu bakar/arang/minyak tanah
8. Hanya mengonsumsi daging/susu/ayam satu kali dalam seminggu
9. Hanya membeli satu stel pakaian baru dalam setahun
10. Hanya mampu makan satu/dua kali dalam sehari
11. Tidak sanggup membayar biaya pengobatan di puskesmas/poliklinik
12. Sumber penghasilan kepala rumah tangga adalah petani dengan luas lahan 0,5 ha, buruh tani, nelayan, buruh bangunan, buruh perkebunan, atau pekerjaan lainya dengan pendapatan dibawah Rp.600.000 per bulan
13. Pendidikan terakhir kepala rumah tangga tidak sekolah/tidak tamat sekolah dasar/hanya sekolah dasar
14. Tidak memiliki tabungan/barang yang mudah dijual dengan nilai minimal Rp.500.000 seperti sepeda motor, emas, hewan ternak, kapal motor ataupun barang modal lainya.

2.4 Data Mining

2.4.1 Definisi Data Mining

Data Mining adalah pendekatan interdisipliner yang telah diadopsi dalam berbagai bidang seperti mengoptimalkan sistem industri, analisis pasar untuk menemukan model-model yang menarik secara finansial, analisis data medis, dan *web* (Issad et al., 2019). *Data Mining* adalah salah satu bidang dari proses penemuan pengetahuan yang mampu memberikan jalur-jalur inovatif dalam menginterpretasi data, ekstraksi informasi yang sebelumnya tidak teridentifikasi, dan mendeteksi pola yang tersembunyi di antara sekelompok data, dalam set data yang besar untuk memprediksi hasil (Cruz et al., 2021).

2.4.2 Manfaat Data Mining

Data Mining dapat dimanfaatkan untuk mendeteksi kejadian – kejadian yang ganjil seperti penyakit tertentu, transaksi mencurigakan, hingga mendeteksi telepon yang dilakukan oleh penipu (Handayanto & Herlawati, 2020). Berikut adalah beberapa manfaat lain dari penggunaan data mining:

1. Meningkatkan efisiensi dan produktivitas bisnis : *Data Mining* dapat membantu perusahaan dalam mengoptimalkan proses bisnis dan meningkatkan efisiensi serta produktivitas karyawan.
2. Meningkatkan kualitas produk dan layanan : *Data Mining* dapat membantu perusahaan dalam mengidentifikasi pola dan tren konsumen, sehingga dapat meningkatkan kualitas produk dan layanan yang ditawarkan.

3. Meningkatkan efektivitas pemasaran : *Data Mining* dapat membantu perusahaan dalam mengidentifikasi preferensi pelanggan dan membuat kampanye pemasaran yang lebih efektif dan efisien.
4. Meningkatkan keamanan : *Data Mining* dapat membantu perusahaan dalam mengidentifikasi ancaman keamanan dan melakukan tindakan pencegahan yang tepat.
5. Meningkatkan pengambilan keputusan : *Data Mining* dapat membantu perusahaan dalam membuat keputusan yang lebih baik dan berdasarkan data, sehingga mengurangi risiko kesalahan dan meningkatkan efektivitas keputusan.
6. Meningkatkan kualitas perawatan kesehatan : *Data Mining* dapat membantu dalam diagnosis dan pengobatan penyakit dengan mengidentifikasi pola dalam data medis.
7. Meningkatkan akurasi prediksi : *Data Mining* dapat membantu dalam memprediksi hasil masa depan berdasarkan data historis dan mengurangi risiko kesalahan dalam prediksi.
8. Meningkatkan kepuasan pelanggan : *Data Mining* dapat membantu dalam mengidentifikasi preferensi dan kebutuhan pelanggan serta memprediksi perilaku pembelian masa depan, sehingga dapat meningkatkan kepuasan pelanggan.

Secara keseluruhan, *Data Mining* memiliki manfaat yang sangat luas dan dapat membantu dalam meningkatkan efisiensi, produktivitas, kualitas produk dan layanan, keamanan, pengambilan keputusan, kualitas perawatan kesehatan, akurasi prediksi, dan kepuasan pelanggan.

2.4.3 Proses Data Mining

Data Mining adalah proses mengekstrak informasi yang berguna dari dataset besar dan kompleks. Proses ini melibatkan penggunaan teknik-teknik statistik, matematika, dan *Machine Learning* untuk mengidentifikasi pola dan tren dalam data yang ada. Proses pada *Data Mining* terdiri dari beberapa tahapan (Cembranel et al., 2019) :

1. Seleksi Data : Tahapan ini melibatkan identifikasi dan seleksi data yang relevan untuk dianalisis serta memilih teknik yang sesuai untuk menyelesaikan masalah.
2. *Pre-Processing* : Berbagai teknik digunakan untuk mempersiapkan data, termasuk pembersihan untuk menghapus data yang tidak konsisten dan *outlier*, integrasi sumber data, dan transformasi data menjadi bentuk yang tepat untuk *mining*.
3. *Data Mining* : Metode yang berbeda seperti aturan asosiasi, model klasifikasi, dan analisis *Clustering* diterapkan pada dataset untuk mengekstraksi pola data.
4. Evaluasi : Evaluasi data melibatkan analisis pola yang ditemukan pada tahap *Data Mining* dan mengidentifikasi pola mana yang relevan untuk studi.
5. Pengetahuan : Tahap ini melibatkan menentukan pola yang relevan dan cara terbaik untuk menerapkan pengetahuan tersebut pada masalah.

Secara keseluruhan, proses dan teknik *Data Mining* membantu untuk mengekstrak wawasan dan pengetahuan yang berharga dari dataset besar, yang

dapat digunakan untuk berbagai tujuan seperti pengambilan keputusan, prediksi, dan optimasi.

2.4.4 Teknik Data Mining

Terdapat berbagai teknik yang digunakan dalam data mining untuk mengidentifikasi pola dan informasi yang tersembunyi dalam data. Menurut (Issad et al., 2019) teknik yang biasa digunakan dalam data mining adalah :

1. *Association Rule Mining* : Teknik ini digunakan untuk menemukan hubungan atau asosiasi antara item atau variabel dalam dataset. Contohnya, dapat digunakan untuk menemukan hubungan antara barang yang dibeli oleh pelanggan di toko *online*.
2. *Clustering* : Teknik ini digunakan untuk mengelompokkan data dalam kelompok-kelompok atau *Cluster* berdasarkan kesamaan atau perbedaan tertentu. Teknik ini berguna untuk mengidentifikasi pola atau struktur dalam *dataset*.
3. *Classification* : Teknik ini digunakan untuk membangun model yang dapat digunakan untuk mengklasifikasikan data dalam kategori tertentu. Teknik ini memerlukan data yang telah diberi label sebelumnya.
4. *Regression* : Teknik ini digunakan untuk membangun model yang dapat digunakan untuk memprediksi nilai berdasarkan variabel lain dalam dataset. *Regresi linear* adalah salah satu teknik yang paling umum digunakan dalam *Data Mining*.

5. *Time Series Analysis* : Teknik ini digunakan untuk menganalisis data yang diambil dari waktu ke waktu. Analisis ini dapat membantu dalam memprediksi *trend* atau pola berulang dalam data.
6. *Text Mining* : Teknik ini digunakan untuk menganalisis data teks dan mengidentifikasi pola atau informasi yang bermanfaat dari data tersebut. Teknik ini dapat digunakan untuk menganalisis pesan, *email*, dokumen, atau postingan di media sosial.
7. *Web Mining* : Teknik ini digunakan untuk menganalisis data yang diambil dari *web*. Teknik ini dapat digunakan untuk menganalisis pola lalu lintas web, perilaku pengguna, atau preferensi pelanggan.

2.4.5 Klasifikasi

Klasifikasi banyak digunakan dalam berbagai aplikasi seperti pengenalan gambar, kategorisasi teks, skoring kredit, dan deteksi kecurangan. Klasifikasi adalah proses untuk mengklasifikasikan observasi baru berdasarkan kelas yang telah ditentukan sebelumnya, yaitu pembelajaran terawasi (Gupta & Chandra, 2020). Proses pembelajaran yang terawasi memungkinkan untuk melakukan prediksi label kelas dari satu set data latihan (Issad et al., 2019). Hal ini terdiri dari pemetaan setiap elemen data yang dipilih menjadi salah satu dari satu set kelas yang telah ditentukan sebelumnya.

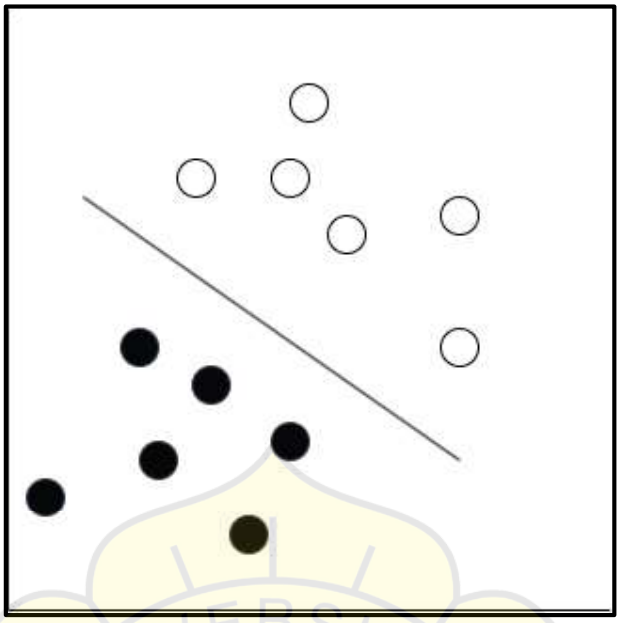
Klasifikasi adalah teknik *Data Mining* yang melibatkan pengkategorian data ke dalam kelas atau grup tertentu berdasarkan serangkaian fitur atau atribut. Tujuan dari klasifikasi adalah membangun model yang dapat memprediksi dengan akurat kelas dari titik data baru atau tidak diketahui berdasarkan karakteristik data.

Model klasifikasi dapat dibangun menggunakan berbagai algoritma, seperti *Decision Tree, K-Nearest Neighbor*, *Support Vector Machines*, dan *Neural Network*. Model ini dapat dilatih menggunakan data yang telah diberi label, yang berarti label kelas untuk data sudah diketahui, atau data tanpa label, yang berarti label kelas tidak diketahui dan harus diprediksi oleh model.

2.4.6 Support Vector Machine

Support Vector Machine (SVM) diperkenalkan oleh Vapnik sebagai model pembelajaran mesin berbasis kernel untuk mengerjakan tugas klasifikasi dan regresi. SVM merupakan teknik yang kuat yang digunakan dalam klasifikasi data dan analisis regresi (Cervantes et al., 2020). Keunggulan SVM terletak pada kenyataan bahwa mereka memperoleh *subset* vektor dukungan selama fase pembelajaran, yang seringkali hanya merupakan bagian kecil dari himpunan data asli. *Set vector* dukungan ini mewakili tugas klasifikasi tertentu dan terdiri dari himpunan data yang kecil.

2.4.6.1 Data yang terpisah Secara Linear



Gambar 2.2 Data Terpisah Secara Linear

Gambar 2.2 menunjukan data yang terpisah secara linear. Berikut adalah tahapan dalam SVM untuk menyelesaikan masalah klasifikasi dengan data yang terpisah secara linear:

1. *Preprocessing* data : pada tahap ini perlu dipersiapkan data pelatihan dengan memilih fitur yang tepat dan melakukan normalisasi data jika diperlukan.
2. Memilih kernel : Pilih kernel yang tepat untuk masalah klasifikasi. Kernel yang paling umum digunakan dalam SVM untuk masalah klasifikasi linear adalah kernel linear. Kernel ini digunakan ketika data dapat dipisahkan secara linear di ruang fitur asli atau setelah dilakukan transformasi linear. Kernel linear dapat didefinisikan sebagai produk dot antara dua data, seperti berikut:

   $K(x, x') = \langle x, x' \rangle$

dimana x dan x' adalah dua data yang akan dihitung kernel-nya, dan <.> adalah operator dot product.

3. Menentukan *hyperplane* : Menentukan *hyperplane* terbaik dengan menggunakan teknik optimasi seperti teknik pemrograman kuadrat bertingkat. *Hyperplane* yang terbaik adalah *hyperplane* yang memisahkan dua kelas data dengan margin yang maksimal. Dalam kasus klasifikasi dua kelas, *hyperplane* dapat didefinisikan sebagai:

   w^T x + b = 0

   dimana x adalah vektor fitur data, w adalah vektor bobot, b adalah bias atau *offset*, dan ^T adalah operator transpose. Jarak antara *hyperplane* dan titik data x dapat dihitung dengan rumus:

   y(x) = w^T x + b

   dimana y(x) adalah jarak antara *hyperplane* dan titik data x. Jika y(x) > 0, maka titik data x diklasifikasikan ke kelas pertama, sedangkan jika y(x) < 0, maka titik data x diklasifikasikan ke kelas kedua. Jarak antara *hyperplane* dan titik data terdekat dari masing-masing kelas dapat dihitung sebagai:

   margin = 2 / ||w||

   dimana ||w|| adalah norma *Euclidean* dari vektor bobot w. Selanjutnya, *hyperplane* yang optimal adalah *hyperplane* yang memaksimalkan margin.

4. Menentukan *support vector* : Setelah menentukan *hyperplane*, SVM menemukan *support vector* yang merupakan data pelatihan yang berada pada margin terdekat. Setelah nilai margin dihitung, carilah titik data

dengan margin terkecil pada masing-masing kelas. Titik data tersebut adalah *support vector* pada kelas tersebut. Secara matematis, *support vector* pada kelas positif (y=1) dapat didefinisikan sebagai:

y_i(w^T x_i + b) = 1

dimana x_i adalah vektor fitur titik data ke-i, y_i adalah label kelas titik data ke-i, w adalah vektor bobot, dan b adalah bias. Dalam hal ini, *support vector* adalah titik data dengan margin terkecil, atau dengan kata lain, titik data yang memenuhi persamaan di atas dengan nilai margin terkecil. Proses yang sama juga dilakukan untuk kelas negatif (y=-1) untuk menentukan *support vector* pada kelas tersebut. *Support vector inilah* yang digunakan untuk menentukan margin dan memperbaiki *hyperplane* jika diperlukan.

5. Menentukan margin : Menentukan margin terbesar antara *support vector* dan *hyperplane*. Margin ini didefinisikan sebagai jarak antara *hyperplane* dan *support vector* terdekat. Margin merupakan ukuran seberapa jauh titik-titik data pada masing-masing kelas dari *hyperplane*. Untuk menentukan margin, perlu menggunakan rumus:

   margin = 2 / ||w||

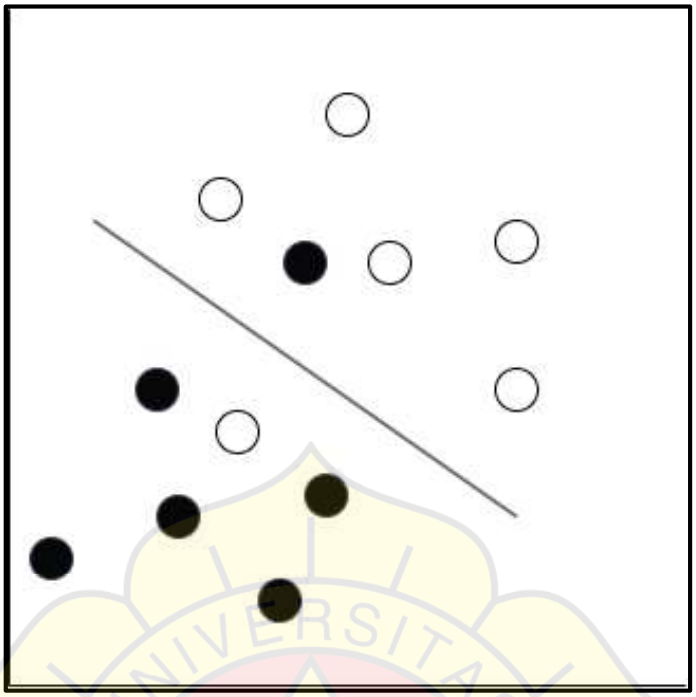
   dimana ||w|| adalah norma *Euclidean* dari vektor bobot. Norma *Euclidean* dari sebuah vektor adalah akar kuadrat dari jumlah kuadrat setiap komponen vektor tersebut. Selain itu, margin juga dapat dihitung sebagai jarak antara *hyperplane* dan *support vector*. Dalam hal ini, margin dapat dihitung dengan menggunakan rumus:

   margin = y_i(w^T x_i + b) / ||w||

dimana y_i adalah label kelas pada titik data ke-i, w adalah vektor bobot, b adalah bias, x_i adalah vektor fitur pada titik data ke-i, dan ||w|| adalah norma *Euclidean* dari vektor bobot. Margin yang lebih besar menunjukkan model yang lebih general dan dapat mencegah *overfitting*, sedangkan margin yang lebih kecil menunjukkan model yang lebih spesifik dan dapat meningkatkan akurasi pada data latih. Namun, margin yang terlalu kecil juga dapat menyebabkan *overfitting* pada data latih. Oleh karena itu, pemilihan nilai margin yang tepat sangat penting dalam SVM.

6. Prediksi : Gunakan model SVM untuk melakukan klasifikasi pada data baru. Untuk setiap data baru, nilai prediksi dapat dihitung dengan mengalikan kernel antara data baru dan data pelatihan dengan multipliers *Lagrange* dan *hyperplane* terbaik.

2.4.6.2 Data yang Tidak Terpisah Secara Linear



Gambar 2.3 Data yang Tidak Terpisah Secara Linear

Gambar 2.3 menunjukan data yang tidak terpisahkan secara linear. Berikut adalah tahapan dalam SVM untuk menyelesaikan masalah klasifikasi dengan data yang terpisah secara non-linear:

1. *Preprocessing* data : Persiapkan data pelatihan dengan memilih fitur yang tepat dan melakukan normalisasi data jika diperlukan.
2. Memilih kernel : Pilih kernel non-linear yang tepat untuk masalah klasifikasi. Beberapa kernel non-linear yang umum digunakan dalam SVM adalah :
    a. *Radial Basis Function* (RBF) : Kernel Kernel RBF digunakan untuk mengubah data ke dalam ruang fitur non-linear yang memiliki dimensi yang lebih tinggi. Kernel RBF didefinisikan sebagai berikut:

    K(x, x') = exp(-gamma * ||x - x'||^2)

dimana x dan x' adalah dua data yang akan dihitung kernel-nya, ||.|| adalah norma *Euclidean*, dan *gamma* adalah parameter kernel yang dapat diatur untuk mengontrol kompleksitas model.

b. *Polynomial Kernel* : Kernel *polynomial* mengubah data ke dalam ruang fitur yang memiliki dimensi yang lebih tinggi menggunakan fungsi polinomial. Kernel *polynomial* didefinisikan sebagai berikut:

   K(x, x') = (gamma * <x, x'> + coef0)^degree

   dimana x dan x' adalah dua data yang akan dihitung kernel-nya, gamma, coef0, dan degree adalah parameter kernel yang dapat diatur.

c. *Sigmoid Kernel* : Kernel sigmoid digunakan untuk mengubah data ke dalam ruang fitur non-linear yang memiliki dimensi yang lebih tinggi. Kernel sigmoid didefinisikan sebagai berikut:

   K(x, x') = tanh(gamma * <x, x'> + coef0)

   dimana x dan x' adalah dua data yang akan dihitung kernel-nya, gamma dan coef0 adalah parameter kernel yang dapat diatur.

3. Transformasi data : Transformasi data pelatihan ke dalam ruang fitur yang diubah oleh kernel. Dalam ruang fitur baru, SVM mencari *hyperplane* yang memisahkan dua kelas data dengan margin maksimal.
4. Menentukan *support vector* : Setelah menentukan *hyperplane*, SVM menemukan *support vector* yang merupakan data pelatihan yang berada pada margin terdekat. *Support vector* inilah yang digunakan untuk menentukan margin dan memperbaiki *hyperplane* jika diperlukan.

5. Menentukan margin : Menentukan margin terbesar antara *support vector* dan *hyperplane*. Margin ini didefinisikan sebagai jarak antara *hyperplane* dan *support vector* terdekat.
6. Prediksi : Gunakan model SVM untuk melakukan klasifikasi pada data baru. Untuk setiap data baru, nilai prediksi dapat dihitung dengan mengalikan kernel antara data baru dan data pelatihan dengan multipliers *Lagrange* dan *hyperplane* terbaik.

2.4.7 GRID Search

Grid Search (GS) merupakan salah satu algoritma yang digunakan untuk melakukan optimasi pada proses klasifikasi *Data Mining*. *Grid search* adalah pencarian yang menyeluruh berdasarkan pada subset yang telah didefinisikan pada ruang *hyperparameter* yang ditentukan menggunakan nilai minimal (batas bawah), nilai maksimal (batas atas), dan jumlah langkah (Sulistiana & Muslim, 2020).

Prinsip dasar dari metode *Grid Search* adalah dengan membagi *grid* pada rentang tertentu dan menelusuri semua titik pada jaringan dengan nilai-nilai parameter C dan σ (Yao et al., 2021). Algoritma ini digunakan untuk mencari kombinasi parameter terbaik pada model yang akan digunakan. Berikut adalah langkah-langkah algoritma *Grid Search*:

1. Tentukan parameter yang ingin dioptimalkan pada model. Misalnya, pada model *Support Vector Machine* parameter yang ingin dioptimalkan adalah C (penalty parameter) dan *gamma* (kernel coefficient).
2. Tentukan range nilai untuk setiap parameter yang ingin dioptimalkan. Misalnya, range nilai untuk parameter C dapat ditentukan sebagai

[0.001, 0.01, 0.1, 1, 10, 100] dan range nilai untuk parameter gamma dapat ditentukan sebagai [0.1, 0.01, 0.001].

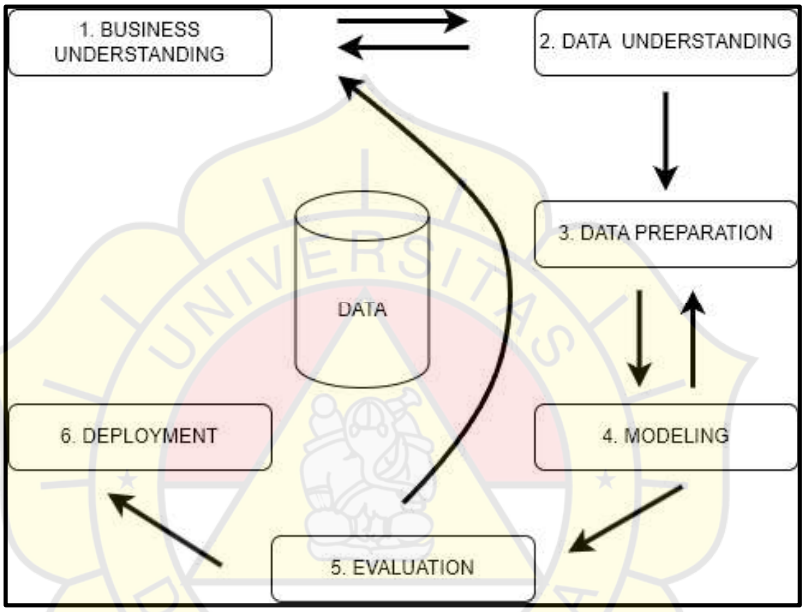
3. Buat semua kombinasi parameter dari nilai yang telah ditentukan pada langkah kedua.
4. Untuk setiap kombinasi parameter, lakukan *Cross Validation* pada data *training* untuk mengevaluasi performa model pada setiap kombinasi parameter tersebut.
5. Pilih kombinasi parameter yang memberikan performa terbaik pada data testing.
6. Gunakan kombinasi parameter terbaik untuk membuat model akhir pada seluruh data training.

2.4.8 CRISP DM

CRISP-DM (*Cross-Industry Standard Process for Data Mining*) adalah metodologi umum dalam proyek penambangan data. Metodologi CRISP-DM sangat fleksibel dan dapat diadaptasi untuk berbagai jenis proyek analisis data. Metodologi ini memberikan kerangka kerja yang jelas dan terstruktur untuk mengelola proyek analisis data, dan memastikan bahwa proses tersebut dapat dilakukan dengan efisien dan efektif. Hal ini juga membantu dalam mengurangi risiko kesalahan dan meningkatkan keberhasilan proyek analisis data.

CRISP-DM juga menerapkan prinsip iteratif, yang berarti bahwa setiap tahap dapat dilakukan ulang sebanyak yang diperlukan untuk memastikan bahwa hasilnya memenuhi tujuan bisnis. Ini memungkinkan untuk menyesuaikan proyek analisis data seiring dengan perkembangan bisnis atau kebutuhan data.

CRISP-DM digunakan sebagai strategi pemecahan masalah secara umum dari bisnis atau unit penelitian (Wibowo & Jananto, 2020), dan telah terbukti berhasil dalam mengelola proyek analisis data yang kompleks. CRISP-DM memberikan kerangka kerja yang dapat digunakan oleh analis data dan manajer bisnis untuk mengelola proyek analisis data dengan baik, sehingga dapat meningkatkan pengambilan keputusan bisnis yang tepat dan efektif.



Gambar 2.4 CRISP-DM

Metodologi CRISP-DM terdiri dari enam tahap utama (Hasanah et al., 2021) dan berfokus pada tiga aspek utama yaitu bisnis, data, dan teknologi. Tahapan-tahapan dalam metodologi CRISP-DM seperti pada gambar 2.4 adalah sebagai berikut:

1. *Business Understanding* : Tahap pertama ini adalah identifikasi tujuan bisnis, kebutuhan pengguna, dan sumber daya yang tersedia untuk menyelesaikan masalah bisnis. Dalam tahap ini, tim proyek mengumpulkan informasi tentang tujuan proyek, tantangan, dan batasan yang mungkin terjadi.

2. *Data Understanding* : Tahap ini adalah identifikasi dan eksplorasi data yang relevan yang tersedia untuk proyek. Tim proyek mengumpulkan dan menganalisis data yang berkaitan dengan masalah bisnis yang ditentukan untuk memastikan bahwa data yang akan digunakan untuk analisis adalah data yang tepat.
3. *Data Preparation* : Tahap ini adalah persiapan data untuk analisis. Tim proyek membersihkan data, menggabungkan data dari beberapa sumber, dan mentransformasi data untuk memastikan data siap untuk analisis.
4. *Modeling* : Tahap ini adalah proses pemilihan dan pengembangan model untuk analisis. Tim proyek mengevaluasi model yang berbeda dan memilih model terbaik untuk digunakan dalam analisis.
5. *Evaluation* : Tahap ini adalah evaluasi model yang dikembangkan pada tahap sebelumnya. Tim proyek mengevaluasi kinerja model dan memastikan bahwa model dapat digunakan untuk memecahkan masalah bisnis yang ditentukan.
6. *Deployment* : Tahap terakhir ini adalah penggunaan model untuk memecahkan masalah bisnis. Tim proyek mengimplementasikan model dan memantau kinerjanya untuk memastikan bahwa model memberikan nilai bagi bisnis.

2.4.9 Standard Scaler

Standard Scaler adalah salah satu teknik yang penting dalam langkah-langkah *preprocessing* sebelum proses pada *Machine Learning*. Tujuan dari *Standard Scaler* adalah untuk menstandarisasi rentang fungsionalitas input pada

dataset (Jasman et al., 2022). Teknik ini digunakan untuk mengubah atribut menjadi memiliki mean 0 dan standar deviasi 1.

Dalam *Standard Scaler*, setiap atribut diubah dengan mengurangi rata-ratanya dan kemudian membaginya dengan standar deviasi. Dengan demikian, distribusi atribut diubah sehingga memiliki mean yang mendekati 0 dan varians yang sama dengan 1. Varians yang sama dengan 1 berarti seluruh nilai pada atribut telah dibagi dengan standar deviasi.

Perlu diperhatikan bahwa *Standard Scaler* hanya dapat digunakan untuk data numerik. Jika terdapat atribut dengan tipe data *boolean*, atribut tersebut perlu dikonversi terlebih dahulu ke bentuk data numerik sebelum diterapkan *Standard Scaler*. Dengan menggunakan *Standard Scaler*, kita dapat mengubah *dataset* menjadi bentuk yang lebih standar dan memudahkan proses analisis data menggunakan algoritma *Machine Learning*.

2.4.10 One-Hot Encoding

Teknik *One-Hot Encoding* digunakan pada data kategori ketika fitur-fiturnya bersifat nominal (tidak memiliki urutan). Dalam *One-Hot Encoding*, untuk setiap level dari fitur kategorial, kita membuat variabel baru, dan setiap kategori dihubungkan dengan variabel biner yang berisi angka 0 atau 1 (Dahouda & Joe, 2021).

2.4.11 Confusion Matrix

Confusion Matrix adalah sebuah tabel yang digunakan untuk mengevaluasi performa dari sebuah model klasifikasi pada *Machine Learning*. *Confusion Matrix*

menampilkan jumlah prediksi yang benar dan salah yang dilakukan oleh model terhadap data yang telah diklasifikasikan dengan label yang sebenarnya.

Confusion Matrix biasanya berbentuk tabel dengan empat sel atau kotak yang mewakili empat kemungkinan hasil dari prediksi yang dilakukan oleh model. Empat kotak tersebut umumnya dinamakan sebagai berikut:

1. *True Positive* (TP) : Jumlah observasi yang benar diklasifikasikan sebagai positif oleh model.
2. *True Negative* (TN) : Jumlah observasi yang benar diklasifikasikan sebagai negatif oleh model.
3. *False Positive* (FP) : Jumlah observasi yang salah diklasifikasikan sebagai positif oleh model (juga dikenal sebagai Type I error).
4. *False Negative* (FN) : Jumlah observasi yang salah diklasifikasikan sebagai negatif oleh model (juga dikenal sebagai Type II error).

Dengan menggunakan nilai-nilai di atas, confusion matrix dapat memberikan informasi yang lebih rinci tentang performa model klasifikasi (Luque et al., 2019), seperti :

a. Akurasi : $\frac{TP+TN}{TP+FN+TN+FP}$

b. Presisi : $\frac{TP}{TP+FP}$

c. *Recall* : $\frac{TP}{TP+FN}$

d. dan *F1-score* : $2\ x\ \frac{PRESISI\ X\ RECALL}{PRESISI+RECALL}$

Dengan memeriksa nilai-nilai dalam *confusion matrix*, kita dapat mengevaluasi sejauh mana model mampu mengklasifikasikan data dengan benar

dan mengidentifikasi jenis kesalahan yang paling umum dilakukan oleh model tersebut.

2.5 Web

2.5.1 Aplikasi Web

Aplikasi *web* adalah program komputer yang dirancang untuk berjalan di dalam *browser web* dan digunakan untuk menjalankan tugas tertentu melalui jaringan internet (Rizky & Op, 2021). Contoh umum dari aplikasi *web* adalah *email*, media sosial, situs *e-commerce*, dan aplikasi perbankan online. Dibandingkan dengan aplikasi *desktop* yang perlu dipasang pada perangkat pengguna, aplikasi *web* hanya memerlukan *browser web* untuk diakses. Hal ini membuat aplikasi *web* lebih mudah diakses dan digunakan secara fleksibel, karena pengguna dapat mengaksesnya dari mana saja dan kapan saja dengan koneksi internet yang tersedia.

2.5.2 Bahasa Pemrograman Web

Aplikasi web biasanya dibuat dengan menggunakan bahasa pemrograman seperti HTML, CSS, dan JavaScript, serta dapat diakses melalui berbagai perangkat yang terhubung ke internet, seperti komputer, tablet, atau ponsel cerdas (Purwanto et al., 2021).

2.5.2.1 HTML

Hypertext Markup Language (HTML) merupakan bahasa *markup* untuk membuat halaman *web* dan aplikasi *web*. HTML digunakan untuk mengatur tampilan dan struktur suatu halaman *web* dengan menggunakan *tag-tag* atau elemen-elemen yang memiliki arti atau fungsi tertentu (Sari, Azzahrah, et al.,

2022). Setiap *tag* dalam HTML memiliki struktur yang terdiri dari awalan, konten atau isi, dan penutup. Konten atau isi dari tag dapat berupa teks, gambar, video, audio, tabel, formulir, dan banyak lagi.

Dalam pengembangan *web*, HTML merupakan bagian penting dari teknologi *web* yang harus dikuasai oleh setiap *developer web*. HTML digunakan bersama dengan CSS (*Cascading Style Sheets*) dan *JavaScript* untuk membuat tampilan dan interaktivitas pada halaman *web* yang lebih kompleks. Dengan HTML, *developer web* dapat membangun halaman *web* dengan tampilan yang lebih menarik dan interaktif, sehingga meningkatkan pengalaman pengguna.

2.5.2.2 CSS

Cascading Style Sheets (CSS) adalah bahasa pemrograman yang digunakan untuk mengatur tampilan atau gaya pada halaman *web* yang dibuat dengan HTML (Sari, Jannah, et al., 2022). CSS memungkinkan pengembang *web* untuk memisahkan tampilan dari struktur dan konten dari halaman *web*. Dengan menggunakan CSS, pengembang *web* dapat menentukan warna, font, ukuran, tata letak, dan banyak lagi untuk halaman *web*.

CSS bekerja dengan cara memberikan aturan atau instruksi gaya untuk elemen-elemen HTML pada halaman *web*. Setiap aturan CSS terdiri dari *selector* dan deklarasi. *Selector* menentukan elemen HTML mana yang akan diberi gaya, sedangkan deklarasi berisi instruksi gaya yang diterapkan pada elemen tersebut. CSS juga memiliki fitur seperti *inheritance* dan *cascading* yang memungkinkan pengembang *web* untuk menerapkan aturan CSS secara hierarkis dan menimpa aturan yang didefinisikan sebelumnya.

Dengan menggunakan CSS, pengembang *web* dapat membuat halaman *web* yang lebih menarik, mudah dibaca, dan mudah dikelola. Selain itu, CSS juga memungkinkan pengembang *web* untuk membuat tampilan yang responsif dan dapat menyesuaikan diri dengan perangkat yang berbeda, seperti layar *desktop*, tablet, dan ponsel.

2.5.2.3 Python

Python adalah bahasa pemrograman tingkat tinggi yang bersifat interpretatif, dinamis, dan serba guna (Wahyudi et al., 2022). *Python* didesain oleh Guido van Rossum pada tahun 1989 dan dirilis pertama kali pada tahun 1991. *Python* banyak digunakan dalam berbagai bidang, seperti pengembangan *web*, pemrosesan data, kecerdasan buatan, pengembangan permainan, dan banyak lagi.

Python memiliki sintaks yang mudah dibaca dan ditulis, sehingga sangat mudah dipelajari oleh pemula. Selain itu, *Python* juga memiliki dukungan yang kuat dari komunitas pengembang yang besar, sehingga banyak tersedia *library* dan modul yang siap digunakan untuk mempercepat pengembangan aplikasi.Beberapa fitur penting dari *Python* adalah:

1. Memiliki sintaks yang sederhana dan mudah dibaca
2. Mendukung pemrograman berorientasi objek
3. Memiliki pengelolaan memori yang otomatis
4. Memiliki dukungan pustaka yang lengkap
5. Mendukung banyak sistem operasi dan *platform*

Python juga sering digunakan dalam bidang kecerdasan buatan dan pemrosesan data, karena *Python* memiliki *library* yang kuat seperti *NumPy*, *Pandas*, *Matplotlib*,

dan *Scikit-learn* yang memudahkan pengolahan data dan analisis statistik. Dalam pengembangan *web*, *Python* sering digunakan dalam kerangka kerja seperti *Django* dan *Flask* untuk membangun aplikasi *web* yang efisien dan *scalable*.

2.5.3 Manajemen Basis Data MySql

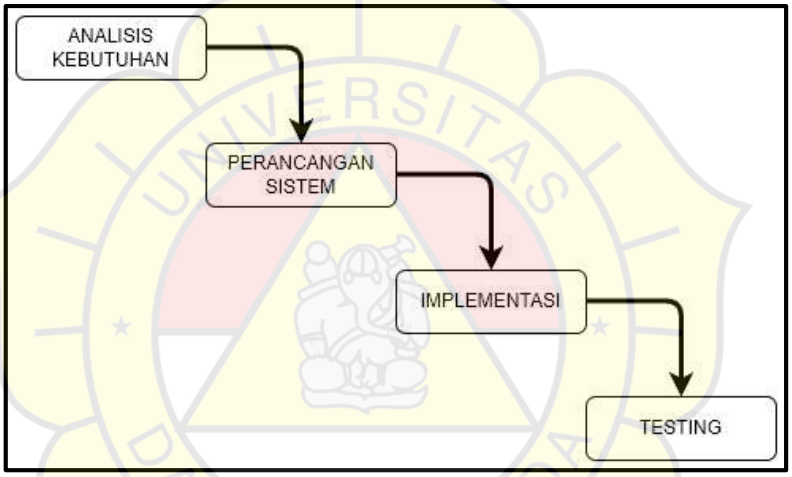
MySQL adalah sebuah sistem manajemen basis data relasional atau RDBMS (*Relational Database Management System*) yang banyak digunakan dalam pengembangan aplikasi *web*. *MySQL* adalah salah satu *software* basis data *open-source* yang paling populer dan banyak digunakan di dunia (Yandri, 2022). MySQL digunakan oleh banyak aplikasi web besar seperti *Facebook*, *Twitter*, *Youtube*, dan lainnya.

MySQL menyediakan fitur-fitur yang lengkap dan mudah digunakan, seperti kemampuan untuk menyimpan dan mengelola data dalam tabel dan relasi antar tabel, fitur pencarian dan pengurutan data, serta dukungan untuk transaksi dan integritas data. MySQL juga memiliki kemampuan untuk mengakses *database* secara *remote* dan mendukung bahasa pemrograman seperti PHP, Java, dan Python. MySQL tersedia dalam dua versi, yaitu versi komunitas (*MySQL Community Edition*) yang gratis untuk digunakan, dan versi *enterprise* (*MySQL Enterprise Edition*) yang berbayar dan menawarkan fitur-fitur tambahan dan dukungan teknis yang lebih baik.

2.6 Pengembangan Sistem

2.6.1 Metode Waterfall

SDLC (*Software Development Life Cycle*) adalah suatu model atau kerangka kerja yang digunakan dalam pengembangan perangkat lunak untuk memandu dan mengatur seluruh tahapan dalam proses pengembangan perangkat lunak, dari awal sampai akhir . SDLC adalah suatu pendekatan terstruktur dalam mengembangkan perangkat lunak yang melibatkan serangkaian tahap yang terurut dan terdokumentasi dengan baik.



Gambar 2.5 Waterfall

Salah satu model SDLC yang biasa digunakan adalah Model *Waterfall.* Model *Waterfall* adalah model pengembangan perangkat lunak yang diatur secara linear dan terstruktur dengan langkah-langkah yang harus diikuti secara berurutan (Nurseptaji et al., 2021). Berikut ini adalah penjelasan detail mengenai tahapan dalam metodologi pengembangan sistem *Waterfall* seperti yang ditunjukan pada gambar 2.5:

1. Analisis Kebutuhan : Pada tahap ini, dilakukan analisis kebutuhan pengguna dan sistem yang akan dikembangkan. Proses analisis ini bertujuan untuk memahami kebutuhan bisnis, tujuan sistem, dan

kebutuhan fungsional sistem yang akan dibangun. Tahapan ini dilakukan dengan beberapa langkah, yaitu:

a. Identifikasi kebutuhan : tahap ini dilakukan dengan melakukan wawancara dengan pengguna atau pemangku kepentingan untuk memahami kebutuhan bisnis dan tujuan sistem.
b. Analisis kebutuhan : tahap ini dilakukan untuk memahami kebutuhan fungsional sistem yang dibangun. Dokumen yang dihasilkan dari tahap ini adalah dokumen kebutuhan.

2. Perancangan Sistem : Setelah dokumen kebutuhan selesai, tahap selanjutnya adalah membuat desain sistem. Pada tahap ini, dibuat desain arsitektur, desain detail, dan spesifikasi teknis. Tujuan dari tahapan ini adalah untuk membuat rencana implementasi sistem yang jelas. Tahap ini dilakukan dengan beberapa langkah, yaitu:
   a. Desain arsitektur : tahap ini dilakukan dengan membuat sketsa dasar sistem, termasuk proses bisnis dan fungsi yang diperlukan. Desain ini membantu memahami hubungan antara komponen sistem.
   b. Desain detail : tahap ini dilakukan dengan membuat rancangan detail dari setiap komponen sistem. Hal ini termasuk pembuatan diagram dan spesifikasi detail dari setiap modul sistem.
   c. Spesifikasi teknis : tahap ini dilakukan dengan membuat spesifikasi teknis yang menguraikan persyaratan teknis dari setiap modul sistem. Hal ini termasuk memilih teknologi yang tepat untuk mengimplementasikan sistem.

3. Implementasi : Setelah desain sistem selesai, langkah selanjutnya adalah mengimplementasikan desain menjadi kode program. Pada tahap ini, dilakukan pengkodean dan integrasi modul-modul yang sudah dibuat. Tujuan dari tahap ini adalah untuk menghasilkan program yang sesuai dengan desain sistem. Tahap ini dilakukan dengan beberapa langkah, yaitu:
    a. Pengkodean : tahap ini dilakukan dengan mengimplementasikan desain sistem menjadi kode program.
    b. Integrasi modul : tahap ini dilakukan dengan mengintegrasikan modul-modul yang sudah dibuat menjadi satu sistem.
    c. Pengujian unit : tahap ini dilakukan dengan menguji setiap modul secara individual untuk memastikan bahwa mereka bekerja dengan benar.
4. *Testing* : Setelah program selesai diimplementasikan, tahap selanjutnya adalah melakukan pengujian untuk menjamin kualitas program. Pada tahap ini, dilakukan pengujian fungsional, pengujian integrasi, dan pengujian sistem. Tujuan dari tahap ini adalah untuk memastikan bahwa program yang dihasilkan sesuai dengan kebutuhan pengguna. Tahap ini dilakukan dengan beberapa langkah, yaitu:
    a. Pengujian fungsional : tahap ini dilakukan dengan menguji fungsi setiap modul untuk memastikan bahwa mereka berfungsi dengan benar.
    b. Pengujian integrasi : tahap ini dilakukan dengan mengintegrasikan modul-modul yang sudah diuji fungsional menjadi satu sistem dan

menguji kembali sistem secara keseluruhan untuk memastikan bahwa sistem berjalan dengan baik.

c. Pengujian sistem : tahap ini dilakukan dengan menguji sistem secara keseluruhan, termasuk interaksi antarmuka pengguna dan sistem, untuk memastikan bahwa sistem memenuhi kebutuhan dan spesifikasi teknis.

2.6.2 Pemodelan Sistem dengan UML

Unified Modeling Language (UML) adalah bahasa pemodelan visual yang digunakan untuk merancang, memodelkan, dan mendokumentasikan sistem berorientasi objek. UML terdiri dari beberapa diagram yang digunakan untuk merepresentasikan berbagai aspek sistem yang berbeda (Munawar, 2021). Diagram diagram tersebut adalah sebagai berikut :

a. *Use Case* : diagram merupakan salah satu jenis diagram UML yang paling populer dan digunakan secara luas dalam pengembangan perangkat lunak. *Use case* diagram digunakan untuk menggambarkan interaksi antara sistem dan pengguna dengan menunjukkan skenario penggunaan sistem. *Use Case* diagram terdiri dari beberapa komponen penting, di antaranya:

   1. Aktor : Merupakan entitas atau pengguna yang berinteraksi dengan sistem. Aktor direpresentasikan oleh simbol manusia atau bagian dari sistem lain, seperti perangkat lunak lain atau sistem yang terhubung.
   2. *Use Case* : Merupakan aksi atau aktivitas yang dilakukan oleh pengguna atau sistem dalam rangka mencapai tujuan tertentu. *Use Case*

direpresentasikan oleh oval dan ditulis dengan kata kerja dalam bentuk infinitif.

3. Hubungan antara aktor dan *Use Case* : Merupakan relasi antara aktor dan *Use Case*, menunjukkan hubungan antara pengguna atau sistem dan tindakan yang dilakukan untuk mencapai tujuan tertentu. Ada tiga jenis hubungan antara aktor dan *Use Case*, yaitu *Association*, *Generalization*, dan *Include/Extend*.
4. *System Boundary* : Merupakan batas antara sistem dan lingkungannya. *System Boundary* direpresentasikan oleh kotak persegi panjang dan dapat membantu membatasi dan memfokuskan *Use Case* diagram.

b. *Activity* diagram adalah salah satu jenis diagram UML yang digunakan untuk menggambarkan alur kerja atau proses bisnis dari suatu sistem atau aplikasi. *Activity* diagram menggunakan simbol-simbol yang menggambarkan aktivitas atau tindakan dan hubungan antara aktivitas tersebut. Beberapa simbol yang digunakan dalam *Activity* diagram adalah:

1. *Initial node* : Simbol ini menunjukkan titik awal dari aktivitas atau proses.
2. *Activity node* : Simbol ini menunjukkan suatu aktivitas atau tindakan yang dilakukan dalam proses.
3. *Decision node* : Simbol ini menunjukkan titik keputusan dalam proses, di mana sistem harus memilih salah satu dari beberapa jalur yang berbeda untuk melanjutkan proses.
4. *Fork node* : Simbol ini menunjukkan pemisahan beberapa aktivitas atau proses sekaligus.

5. *Join node* : Simbol ini menunjukkan penggabungan beberapa aktivitas atau proses yang terpisah.
6. *Final node* : Simbol ini menunjukkan titik akhir dari proses.

c. *Sequence* diagram : adalah salah satu jenis diagram UML yang digunakan untuk menggambarkan urutan interaksi antara objek-objek dalam sistem atau aplikasi. *Sequence* diagram menggunakan simbol-simbol yang menggambarkan pesan atau panggilan antara objek-objek dan urutan waktu di mana pesan atau panggilan tersebut terjadi. Beberapa simbol yang digunakan dalam *Sequence* diagram adalah:
   1. *Object* : Simbol ini menunjukkan objek atau entitas dalam sistem yang terlibat dalam interaksi.
   2. *Lifeline* : Simbol ini menunjukkan waktu hidup objek dalam sistem atau aplikasi.
   3. *Message* : Simbol ini menunjukkan pesan atau panggilan antara objek-objek dalam sistem atau aplikasi.
   4. *Activation* : Simbol ini menunjukkan waktu di mana objek mengambil tindakan atau melakukan aktivitas setelah menerima pesan atau panggilan.

d. *Deployment* diagram adalah salah satu jenis diagram dalam *Unified Modeling Languag*e (UML) yang digunakan untuk menggambarkan struktur fisik atau konfigurasi sistem perangkat lunak yang akan diimplementasikan atau di *deploy*. Diagram ini memberikan pandangan tentang bagaimana komponen perangkat lunak, perangkat keras, jaringan,

dan lingkungan lainnya berinteraksi dan berkomunikasi satu sama lain dalam konteks sistem yang diimplementasikan.

1. *Node* (simpul) : *Node* mewakili entitas fisik yang berperan dalam menjalankan perangkat lunak, seperti *server*, komputer, perangkat keras, atau perangkat lunak independen yang dapat dijalankan sendiri. Setiap node mungkin memiliki atribut seperti nama, jenis perangkat keras, atau sistem operasi yang terpasang.
2. *Artifact* (artefak) : *Artifact* adalah entitas perangkat lunak yang di-*deploy* pada *node* tertentu. Artifak mewakili file, paket, atau unit lainnya yang berisi kode sumber, eksekutabel, atau data yang akan dijalankan pada *node*. Contoh artifak termasuk file JAR, DLL, atau file konfigurasi.
3. *Connection* (koneksi) : Koneksi menggambarkan hubungan fisik atau logis antara *node* atau artifak. Koneksi ini mewakili jalur komunikasi, seperti koneksi jaringan atau protokol komunikasi yang digunakan untuk menghubungkan antara *node* atau artifak.
4. *Stereotype* (stereotip) : Stereotip adalah anotasi tambahan yang dapat digunakan untuk memberikan informasi tambahan atau penjelasan pada elemen-elemen dalam deployment diagram. Stereotip dapat digunakan untuk menandai elemen sebagai *server*, *client*, *database*, atau elemen khusus lainnya.