BAB II
TINJAUAN PUSTAKA DAN DASAR TEORI

2.1. Tinjauan Pustaka

Pengaduan masyarakat muncul ketika merasa tidak puas terhadap layanan suatu instansi atau lembaga penyelenggara layanan. Pengaduan menjadi indikasi kualitas layanan yang diberikan kurang baik. Akan tetapi ketika penyelenggara mampu menangani pengaduan dengan baik, penyelenggara akan memperoleh kepuasan dan loyalitas masyarakat (Jeanpert dkk., 2021). Layanan pengaduan menuntut aksi yang cepat, mudah, transparan, dan tepat sasaran dari penyelenggara untuk menjawab kebutuhan masyarakat. Oleh karena itu, seiring dengan perkembangan teknologi digital saat ini, hampir setiap instansi penyelenggara layanan memiliki layanan pengaduan secara daring (Yan dkk., 2021), baik melalui media sosial maupun *platform online* (*website*) milik instansi yang dapat diakses oleh masyarakat untuk menyampaikan keluhan atas ketidakpuasan terhadap layanan yang diberikan (Stevens dkk., 2018).

Layanan penanganan pengaduan secara daring telah diterapkan dalam berbagai sektor layanan.publik, misalnya pada sektor pertanian di Negara Mesir. Di sana telah dikembangkan *AgroSupportAnalytics* yang merupakan sistem manajemen pengaduan dan pendukung keputusan berbasis *cloud* untuk pertanian berkelanjutan yang dapat diakses oleh para petani dan terhubung langsung dengan para petugas dan ahli pertanian untuk menjawab pengaduan-pengaduan yang disampaikan. Pada sistem tersebut terdapat tiga status pengaduan, yaitu belum diselesaikan, dalam proses penyelesaian, dan terselesaikan (Munir dkk., 2021).

Pengaduan-pengaduan yang masuk sangat penting untuk diidentifikasi dan diklasifikasikan guna mengetahui inti permasalahan. Proses tersebut jika dilakukan secara manual maka akan menjadi tidak efisien dan menyita banyak waktu. jika terjadi peningkatan jumlah pengaduan yang signifikan dengan berbagai topik yang berbeda. Sehubungan dengan hal tersebut, dikembangkan klasifikasi pengaduan secara otomatis melalui klasifikasi teks pengaduan ke dalam beberapa kategori dengan metode pembelajaran mesin (*machine learning)* (HaCohen-Kerner dkk.,

2019). Misalnya, dengan menggunakan algoritma *semi-supervised mincuts* (Singh dkk., 2021), *Naïve Bayes Classifier, K-NN, ID3, Random Trees* (Ghazzawi dan Alharbi, 2019; Khedkar dan Shinde, 2020).

Dalam penelitian ini digunakan tiga algoritma *Naïve Bayes Classifier,* yaitu *Bernoulli NB, Multinomial NB,* dan *Gaussian NB.* Berdasarkan beberapa literatur yang diperoleh, terdapat penelitian-penelitian yang menerapkan algoritma *Naïve Bayes Classifier* untuk menentukan klasifikasi terhadap berbagai data untuk maksud dan tujuan tertentu. Dalam bidang kesehatan algoritma *Naïve Bayes Classifier* digunakan untuk beberapa tujuan, diantaranya mengklasifikasikan sel darah merah normal dan tidak normal untuk mendeteksi adanya kelainan. Hasil akurasi yang diperoleh sebesar 98,212% (Patgiri dan Ganguly, 2021). *Naïve Bayes Classifier* telah digunakan untuk mendiagnosis penyakit Parkinson dengan mengklasifikasi data menjadi dua kelas, yaitu sehat dan menderita penyakit Parkinson. Hasil akurasi yang diperoleh sebesar 70,26% (Avuçlu dan Elen, 2020). *Naïve Bayes Classifier* digunakan untuk klasifikasi sinyal elektromiografi permukaan (sEMG) yang merupakan sinyal berisi seluruh informasi yang berkaitan dengan fungsi otot dengan hasil akurasi sebesar 94,2% (Narayan, 2020), klasifikasi model prediksi potensi Neurotoksisitas pada bahan kimia sebagai tahap awal desain obat agar tidak menimbulkan efek yang merugikan bagi sistem saraf dengan akurasi sebesar 83,3% (Zhang dkk., 2020).

Naïve Bayes Classifier telah digunakan untuk mendeteksi penyakit kulit dengan akurasi sebesar 72,7% (Balaji dkk., 2020), klasifikasi data klinis *METABRIC (Molecular Taxonomy of Breast Cancer International Consortium)* berjumlah 2504 data pasien kanker payudara dengan 35 fitur gejala menjadi dua kelas, yaitu pasien positif dan negatif kanker payudara dengan akurasi sebesar 70% (Sweetlin dan Ponraj, 2019). Selain itu, digunakan untuk mendeteksi penyakit ginjal dengan menggunakan 400 data pasien dengan 25 fitur gejala dan diklasifikasikan dalam dua kelas, yaitu pasien penderita gagal ginjal kronis dan bukan penderita gagal ginjal kronis dengan hasil akurasi sebesar 95% (Kapoor dkk., 2019). *Naïve Bayes Classifier* juga digunakan untuk mendeteksi *osteonecrosis* atau kematian jaringan tulang karena kekurangan suplai darah. Data yang digunakan

adalah data 150 pasien dari Januari 2011 hingga Juni 2013. Hasil akurasi yang diperoleh sebesar 76,9% (Cui dkk., 2018).

Beberapa kegunaan algoritma *Naïve Bayes Classifier* dalam bidang pertanian, yaitu mendeteksi penyakit bintik hitam pada biji gandum menggunakan sampel gambar digital dengan hasil akurasi sebesar 83% (Zhou dkk., 2021). Kemudian untuk mendeteksi genotip progeni jagung (*Zea mays L.*) dengan akurasi sebesar 85% (Seka dkk., 2019), dan mendeteksi gejala penyakit pada daun okra dan labu pahit (pare) dengan akurasi sebesar 95% untuk daun okra dan 82,67% untuk labu pahit (pare) (Mondal dkk., 2017).

Dalam bidang energi, *Naïve Bayes Classifier* digunakan untuk mendeteksi titik panas/*hot spot* untuk mengklasifikasikan tingkat kecacatan pada sel surya fotovoltaik yang dapat berefek buruk pada efisiensinya. Data yang digunakan berupa gambar digital sel surya yang diklasifikasikan ke dalam 3 kelas, yaitu cacat total, tidak cacat dengan ada titik panas, tidak cacat dengan tidak ada titik panas. Hasil akurasi yang diperoleh sebesar 94,10% (Niazi dkk., 2019). Selanjutnya, *Naïve Bayes Classifier* digunakan untuk mendeteksi kesalahan dalam jalur transmisi dengan data sinyal harmonik *Leakage Current* (LC) sebagai residu. Data diklasifikasikan ke dalam dua kelas, yakni kesalahan tunggal dan multi kesalahan. Hasil akurasi yang diperoleh sebesar 95% (da Silva dkk., 2018). Selain itu, *Naïve Bayes Classifier* digunakan untuk mengklasifikasi peralatan-peralatan guna mengefisienkan konsumsi energi dengan hasil akurasi sebesar 80% (Yang dkk., 2017) dan untuk mendeteksi kesalahan guna perlindungan jalur transmisi paralel yang melibatkan kesalahan antar sirkuit sehingga aliran daya menjadi lebih efisien. Hasil akurasi yang diperoleh sebesar 99,99% (Swetapadma dan Yadav, 2016).

Algoritma *Naïve Bayes Classifier* telah digunakan dalam banyak penelitian bukan saja untuk mengklasifikasi gambar digital dalam berbagai bidang untuk tujuan tertentu, seperti penelitian untuk mengklasifikasi secara otomatis gambar digital yang merupakan gambar iklan promosi atau bukan dengan tingkat akurasi 94,31% (Hubert dkk., 2021) dan beberapa penelitian lainnya. Namun, algoritma tersebut juga dapat mengklasifikasi teks dengan baik misalnya untuk menentukan apakah sebuah kalimat merupakan pernyataan atau pertanyaan (Kolluri dan Razia,

2020). Selain itu, *Naïve Bayes Classifier* pernah digunakan untuk mengklasifikasi kepuasan pelanggan terhadap layanan 33 hotel di Las Vegas berdasarkan 47.172 ulasan online pada periode bulan Maret 2005 hingga Januari 2017. Penelitian tersebut mendeteksi secara otomatis layanan hotel berada pada kelas bintang 5 untuk ulasan positif atau bintang 1 untuk ulasan negatif. Hasil akurasi yang diperoleh sebesar 86% (Sánchez-Franco dkk., 2019).

Terkait penelitian yang membandingkan akurasi tiga tipe algoritma *Naïve Bayes Classfier*, terdapat penelitian yang melakukan sentimen analisis terhadap data *twitter* yang menggunakan bahasa Indonesia. Hasil perbandingan akurasi yang diperoleh adalah skor akurasi dengan *Bernoulli NB* sebesar 63,37%, skor akurasi dengan *Multinomial NB* sebesar 63,74%, dan skor akurasi dengan *Gaussian NB* sebesar 50,51%. Untuk kasus penelitian ini, algoritma dengan skor akurasi terbaik adalah *Multinomial NB* (Umar N., dan Nur, M. A, 2022).

2.2. Dasar Teori

2.2.1. Pengaduan/ Laporan Masyarakat

Pengaduan masyarakat merupakan salah satu fungsi masyarakat dalam melakukan pengawasan terhadap penyelenggaraan pelayanan publik berupa gagasan, keluhan, atau saran yang bersifat membangun untuk perbaikan layanan, yang disampaikan baik secara lisan maupun tertulis. Masyarakat cenderung peka terhadap kegagalan layanan dan memiliki ekspektasi terhadap perbaikan layanan di masa depan. Menyampaikan pengaduan adalah suatu tindakan pengungkapan emosi atas rasa ketidakpuasan terhadap suatu hal yang terjadi baik mengenai individu, barang, atau situasi. Pengaduan merupakan cara penting atas ketidaksetujuan dalam interaksi sosial yang dikomunikasikan secara berbeda tergantung temperamen dan keadaan pengadu/ pelapor (Singh A., dan Saha S., 2023).

Pengaduan masyarakat dapat dijadikan sumber informasi untuk mengidentifikasi kemungkinan munculnya masalah yang lebih besar atau dapat dijadikan faktor pertimbangan dalam analisis risiko layanan oleh penyelenggara. Pengaduan masyarakat memiliki tingkat permasalahan yang berbeda-beda sehingga

perlu adanya pengelompokkan tertentu untuk tidak menyamaratakan pengelolaan terhadap setiap pengaduan masyarakat. Pengelompokkan yang dilakukan oleh penyelenggara layanan bergantung pada kebijakan yang dibuat dalam organisasinya dengan parameter-parameter penentu yang telah dipertimbangkan secara matang. Dengan demikian, pengelolaan pengaduan dapat lebih terarah, efektif, dan efisien (Shin J., Son S., dan Cha Y, 2022).

Pengaduan masyarakat dapat dipandang sebagai pastisipasi aktif masyarakat dalam memberikan informasi yang berharga untuk layanan yang lebih baik. Apabila masalah pengaduan yang disampaikan dikelola dengan baik oleh para penyelenggara layanan, maka akan dapat menghasilkan solusi yang tepat sasaran dan menimbulkan rasa kepercayaan masyarakat terhadap penyelenggara layanan (DiCarlo M. dkk., 2023)

2.2.2. *Preprocessing Data*

Preprocessing data dilakukan untuk memastikan bahwa data yang akan diproses merupakan data yang berkualitas. Ini dikarenakan data yang berkualitas akan menghasilkan proses pembelajaran (*training*), klasifikasi, dan informasi, serta keputusan yang berkualitas pula. Data yang berkualitas dapat diketahui dari nilai akurasi, kelengkapan, konsistensi, ketepatan waktu, tingkat kepercayaan, dan interpretasi/ penafsirannya. *Preprocessing data* diperlukan oleh sebab data mentah yang diperoleh biasanya "kotor", yakni adanya ketidaklengkapan data seperti nilai data pada suatu atribut tidak tersedia, adanya *noisy* (memuat *error*/ *outliers*), dan adanya inkonsistensi data, yakni perbedaan dalam pengkodean atau penamaan data. *Preprocessing* data diawali dengan pemilihan/seleksi data (*selection data*) yang relevan terhadap analisis klasifikasi yang akan dilakukan. Secara umum, *preprocessing data* terdiri dari pembersihan data (*cleaning data*), integrasi data (*integration data*), reduksi data (*reduction data*), dan transformasi data (*transformation data*) (Han dkk, 2012). Namun, pada penelitian ini hanya dilakukan proses pembersihan data (*cleaning data*) berupa pengecekkan nilai-nilai yang hilang (*missing value*) dan proses transformasi data (*transformation data*), sebagai berikut:

1. Pembersihan Data (*Cleaning Data*)
   - Imputasi/ pengisian nilai-nilai yang hilang (*missing value*) dengan cara mengecek terlebih dahulu kecondongan dari data (*skewness*) untuk menentukan imputasi yang tepat.
   - Pengisian *missing value* dapat dilakukan secara otomatis dengan menggunakan nilai *mean* dari atribut yang mengandung *missing value* jika distribusi data normal dan menggunakan nilai median jika distribusi data tidak normal (condong/ *skew*).
2. Transformasi Data (*Transformation Data*)

   Langkah terakhir dari *preprocessing data* adalah transformasi data. Transformasi data dilakukan dengan mengubah data menjadi bentuk yang sesuai untuk proses lebih lanjut.

2.2.3. Klasifikasi

Klasifikasi dapat dikatakan sebagai salah satu hal yang paling mendasar baik dalam penelitian penambangan data maupun pengambilan keputusan. Klasifikasi merujuk pada proses identifikasi suatu basis data kemudian mengatur dan mengumpulkannya ke dalam salah satu dari beberapa kelas yang telah ditentukan. Dalam klasifikasi, algoritma pengklasifikasi melatih suatu kumpulan data dengan kelas-kelas yang telah ditentukan terlebih dahulu untuk membentuk pola klasifikasi yang akan diterapkan pada data-data baru sebagai masukan/*input*. Hal ini disebut juga *supervised learning*. Klasifikasi diterapkan dengan melibatkan atribut-atribut sebagai parameter atau kriteria dalam prosesnya untuk menentukan kelas data. Proses ini dilakukan secara sistematis untuk mengetahui informasi-informasi tertentu berdasarkan *record* data yang diperoleh (Jacob, 2004).

Secara umum, klasifikasi data meliputi dua tahap, yaitu tahap belajar (*learning*) yang mana model klasifikasi dibangun pada tahap ini, dan tahap klasifikasi yang mana model yang diperoleh digunakan untuk memprediksi label kelas untuk data *input* yang diberikan. Pada tahap pertama, model klasifikasi yang dibangun menggambarkan serangkaian kelas data yang telah ditentukan sebelumnya. Ini adalah tahap belajar atau fase pelatihan (*training*), dimana

algoritma klasifikasi membangun model klasifikasi dengan menganalisis atau "belajar dari" serangkaian pelatihan yang terdiri dari basis data *tuple* dan label kelasnya, biasanya disebut sebagai data latih. *Tuple* X, diwakili oleh vektor atribut n-dimensi, $X = (x_1, x_2, \dots, x_n)$, menggambarkan data n yang dibuat pada *tuple* dari atribut basis data n, masing masing $A_1, A_2, \dots, A_n$. Setiap atribut menunjukkan suatu fitur dari X. Setiap *tuple* X diasumsikan sebagai milik kelas yang telah ditentukan sebagaimana ditentukan oleh atribut basis data lain yang disebut atribut label kelas. Atribut label kelas bernilai diskrit dan tidak teratur. Setiap *tuple* membentuk serangkaian pelatihan yang disebut sebagai pelatihan *tuple* (*training tuples*) dan secara acak diambil sampelnya dari basis data yang sedang dianalisis. Dalam literatur pembelajaran mesin, pelatihan *tuple* seringkali disebut sebagai pelatihan sampel (*training samples*). Dalam konteks klasifikasi kumpulan data dapat disebut sebagai sampel, data contoh, titik data atau objek. Pada tahap belajar data latih dianalisis oleh algoritma klasifikasi dan membangun model klasifikasi. Pada tahap klasifikasi, data uji digunakan untuk mengestimasi akurasi dari model klasifikasi yang telah dibangun. Jika akurasi dapat diterima, maka model klasifikasi tersebut dapat diterapkan untuk klasifikasi kumpulan data baru. Banyak metode klasifikasi telah diusulkan dalam pembelajaran mesin, pengenalan pola, dan statistik (Han dkk., 2012). Dalam beberapa penerapannya, klasifikasi sebagai metode analisis data banyak dipakai untuk membentuk model prediksi atau *trend* dari suatu kumpulan data.

2.2.4. *Naïve Bayes Classifier*

Naïve Bayes Classifier merupakan pengklasifikasi statistik berdasarkan teorema Bayes. Dalam teorema Bayes, dimisalkan *X* adalah data *tuple*. Dalam istilah Bayesian, *X* disebut sebagai bukti. Hal itu dijelaskan oleh pengukuran yang dilakukan pada satu set atribut n. Kemudian terdapat *H* sebagai beberapa hipotesis bahwa data *tuple X* termasuk dalam kelas *C* tertentu. Untuk masalah klasifikasi, akan ditentukan *P(H|X)*, probabilitas bahwa hipotesis *H* memiliki bukti yang diberikan atau data *tuple X*. Dengan kata lain, dicari probabilitas bahwa *tuple X*

termasuk dalam kelas C, mengingat bahwa diketahui gambaran atribut *X* (Han dkk., 2012).

P(H|X) adalah probabilitas posterior (posteriori) atau probabilitas akhir dari kondisi *H* pada *X*. Sebaliknya, *P(H)* adalah probabilitas prior (priori) atau probabilitas awal dari *H*. Demikian pula, *P(X|H)* adalah probabilitas posterior (posteriori) atau probabilitas akhir dari kondisi *X* pada *H*. Sedangkan *P(X)* probabilitas prior (priori) atau probabilitas awal dari *X*. Teorema Bayes sangat berguna karena menyediakan perhitungan probabilitas akhir *P(H|X)* dari *P(H)*, *P(X|H)*, dan *P(X)*. Formula teorema Bayes sebagaimana persamaan (2.1) (Han dkk., 2012).

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)} \tag{2.1}$$

Studi yang membandingkan algoritma pengklasifikasi telah menemukan pengklasifikasi Bayesian sederhana dari teorema Bayes yang dikenal sebagai *Naïve Bayes Classifier* yang menunjukkan akurasi dan kecepatan yang tinggi ketika diterapkan pada basis data yang besar. *Naïve Bayes Classifier* adalah salah satu algoritma *supervised learning* untuk melakukan klasifikasi dengan pendekatan probabilitas yang menghitung *likelihood* setiap atribut untuk memperoleh hasil yang efektif dengan cara yang cepat (efisien) (Kapoor dkk., 2019). *Naïve Bayes Classifier* mengasumsikan bahwa nilai suatu atribut pada kelas tertentu tidak bergantung pada nilai atribut lainnya. Asumsi ini disebut independensi kelas bersyarat. Hal ini dilakukan dengan tujuan menyederhanakan perhitungan yang rumit dan dalam pengertian ini dianggap "*naïve*" (Han dkk., 2012). *Naïve Bayes Classifier* bekerja sebagai berikut:

a. Misalkan D merupakan satu set pelatihan *tuple* dengan label kelasnya. Kemudian, setiap *tuple* diwakili oleh satu vector atribut n dimensi, $X = (x_1, x_2, \dots, x_n)$, menggambarkan data n yang dibuat pada *tuple* dari atribut n, masing masing $A_1, A_2, \dots, A_n$.
b. Terdapat *m* kelas, $C_1, C_2, \dots, C_m$. Diberikan sebuah *tuple* X, pengklasifikasi akan memprediksi bahwa X akan termasuk dalam kelas yang memiliki probabilitas posterior tertinggi, dikondisikan pada *X*.

Artinya, *Naïve Bayes Classifier* memprediksi bahwa *tuple X* termasuk dalam kelas C_i jika dan hanya jika $P(C_i|\boldsymbol{X}) > P(C_j|\boldsymbol{X})$ untuk $1 \leq j \leq m, j \neq i$. Dengan demikian, $P(C_i|\boldsymbol{X})$ dimaksimalkan. Kelas C_i untuk $P(C_i|\boldsymbol{X})$ dimaksimalkan yang disebut hipotesis posteriori maksimum. Berdasarkan teorema Bayes pada persamaan (2.1), diperoleh persamaan (2.2):

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)} \tag{2.2}$$

c. Karena *P(X)* adalah konstanta untuk semua kelas, maka hanya $P(X|C_i)P(C_i)$ yang perlu dimaksimalkan. Jika probabilitas kelas tidak diketahui, maka secara umum diasumsikan bahwa setiap kelas kemungkinan sama, yaitu $P(C_1) = P(C_2) = \cdots = P(C_m)$, oleh karena itu dimaksimalkan $P(X|C_i)$. Sebaliknya, dimaksimalkan $P(X|C_i)P(C_i)$. Probabilitas kelas dapat diestimasi oleh $P(C_i) = |C_{i,D}|/|D|$, dimana $|C_{i,D}|$ adalah jumlah data latih dari kelas C_i dalam *D*.

d. Apabila terdapat kumpulan data dengan banyak atribut, akan sangat banyak secara komputasi untuk menghitung $P(X|C_i)$. Untuk mengurangi perhitungan dalam mengevaluasi $P(X|C_i)$, dibuat asumsi *naïve* "independensi kelas-bersyarat". Ini mengasumsikan bahwa nilai-nilai atribut secara kondisional saling bebas (*independent*) satu sama lain. Dengan demikian diperoleh persamaan (2.3),

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i) \tag{2.3}$$
$$= P(x_1|C_i) \times P(x_2|C_i) \times \ldots \times P(x_n|C_i)$$

Secara mudah dapat diperkirakan probabilitas $P(x_1|C_i), P(x_2|C_i), \ldots, P(x_n|C_i)$ dari data latih. Perlu diingat bahwa di

x_k mengacu pada nilai atribut A_k untuk tuple X. Untuk setiap atribut, dilihat apakah atribut tersebut bernilai diskrit atau *continue*.

e. Untuk memprediksi label kelas *X*, $P(X|C_i)P(C_i)$ dievaluasi untuk setiap kelas C_i. Pengklasifikasi memprediksi label kelas *tuple X* adalah kelas C_i jika dan hanya jika $P(\boldsymbol{X}|C_i)P(C_i) > P(\boldsymbol{X}|C_j)P(C_j)$ untuk $1 \leq j \leq m$, $j \neq i$. Dengan kata lain, label kelas yang diprediksi adalah kelas C_i yang mana $P(X|C_i)P(C_i)$ bernilai maksimum.

Secara umum, Naïve Bayes Classifier memiliki tiga tipe yaitu, *Bernoulli NB*, *Multinomial NB*, dan *Gaussian NB. Bernoulli NB* adalah tipe algoritma *Naïve Bayes Classifier* yang mengasumsikan semua fitur adalah biner/boolean sehingga akan menghasilkan nilai ya/tidak atau 0 atau 1. Misalnya 0 mewakili suatu kata tidak muncul dalam dokumen dan 1 mewakili suatu kata muncul dalam dokumen. Untuk algoritma *Bernoulli NB*, data diasumsikan terdistribusi *Bernoulli*, sehingga persamaan untuk *Bernoulli NB* diperoleh persamaan (2.4).

$$P(x_i|y) = P(i|y)x_i + (1 - P(i|y))(1 - x_i) \tag{2.4}$$

Selanjutnya *Multinomial NB* adalah tipe algoritma *Naïve Bayes Classifier* yang dapat digunakan pada data-data diskrit. Misalkan untuk menghitung frekuensi kata-kata dalam teks untuk memprediksi kelas atau label. Untuk algoritma *Multinomial NB*, data diasumsikan terdistribusi secara *Multinomial*, sehingga persamaan untuk *Multinomial NB* diperoleh persamaan (2.5).

$$\hat{\theta}_{yi} = \frac{N_{yi}+\alpha}{N_y+\alpha n} \tag{2.5}$$

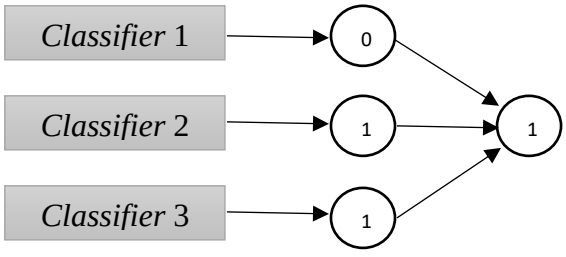
Berikutnya *Gaussian NB* adalah tipe algoritma *Naïve Bayes Classifier* yang dapat digunakan apabila semua fitur data adalah kontinu karena asumsi distribusi normal. Untuk algoritma *Gaussian NB* diperoleh persamaan (2.6).

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right) \quad (2.6)$$

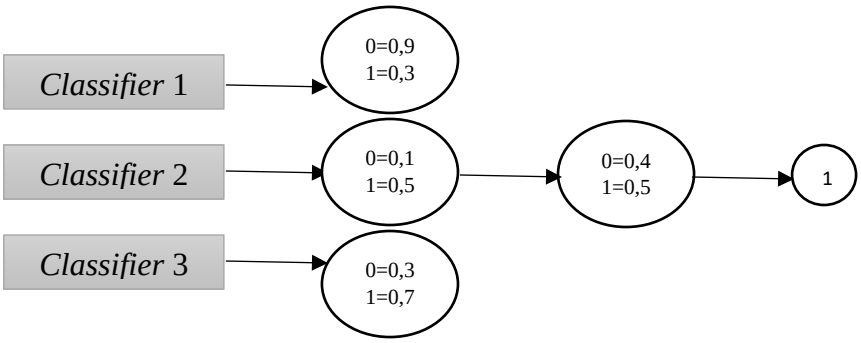
Secara teori, pengklasifikasi Bayesian memiliki tingkat kesalahan minimum dibandingkan dengan semua pengklasifikasi lainnya. Namun, dalam praktiknya ini tidak selalu terjadi, karena ketidakakuratan dalam asumsi yang dibuat untuk penggunaannya, seperti independensi kelas-bersyarat, dan kurangnya data probabilitas yang tersedia (Han dkk., 2012).

2.2.5. *Voting*

Voting merupakan salah satu cara menggabungkan hasil prediksi/ klasifikasi dari beberapa algoritma *machine learning*. Setiap algoritma *machine learning* yang digunakan melakukan kinerja terbaiknya, kemudian dilakukan *voting* terhadap hasil klasifikasi yang diperoleh dari masing-masing algoritma. Dua jenis teknik *voting* yang sering digunakan, yaitu *hard voting (majority voting)* dan *soft voting*. *Hard voting* bekerja dengan menggunakan suara terbanyak dari hasil klasifikasi masing-masing algoritma untuk menentukan kelas/ label. Dengan kata lain, *hard voting* hanya beroperasi pada kelas/ label yang dihasilkan masing-masing algoritma pengklasifikasi. Sedangkan, *soft voting* bekerja dengan menghitung rata-rata probabilitas hasil klasifikasi masing-masing algoritma untuk menentukan kelas/ label sehingga *soft voting* memungkinkan untuk meningkatkan kinerja penggabungan algoritma-algoritma pengklasifikasi yang berbasis probabilitas (Zhou Z. H., 2012). Cara kerja *hard voting* dan *soft voting* ditunjukkan pada Gambar 2.1 dan Gambar 2.2.



Gambar 2.1. *Hard Voting*



Gambar 2.2. *Soft Voting*

2.2.6. *Python* dan *Jupyter Notebook*

Python adalah bahasa pemrograman berorientasi objek dan dapat digunakan secara gratis karena bersifat *open source*. *Python* menyediakan segala kesederhanaan dan kemudahan dalam penulisan kode-kode program dan untuk menjalankan program pengguna cukup melakukan *run* pada baris kode program (Lutz M., 2009).

Python menyediakan *library-library* yang memuat fungsi-fungsi untuk digunakan sesuai kebutuhan. Terkait analisis data dan proses *machine learning*, terdapat beberapa *library python* yang sering digunakan, yaitu *library* pandas, numpy, dan *scikit-learn*. *Library* pandas menyediakan struktur-struktur data yang kaya dan fungsi-fungsi yang didesain untuk membuat pekerjaan menjadi cepat dan mudah. *Library* numpy adalah paket dasar untuk komputasi numerik dalam *python*. Numpy merupakan singkatan dari *numerical python*. Numpy menyediakan objek *array* multidimensi yang cepat dan efisien. Selain itu, numpy juga menyediakan fungsi-fungsi untuk membentuk komputasi elemen dengan *array* atau operasi matematika antar *array* (McKinney W., 2012). *Library scikit-learn* adalah *library* untuk proses *machine learning* pada *python* yang dirilis pada tahun 2007. *Scikit-learn* atau yang biasa disingkat *sklearn* menyediakan algoritma-algoritma *machine learning*, untuk melakukan klasifikasi, regresi, reduksi dimensionalitas, dan *clustering*. *Scikit-learn* juga menyediakan modul-modul fitur ekstraksi, pemrosesan data, dan evaluasi model (Garreta R., dan Moncecchi G, 2013).

Jupiter notebook adalah antarmuka web *open source* yang memungkinkan masukan data berupa teks, video, audio, dan gambar, bersama dengan kemungkinan untuk mengeksekusi kode dari berbagai bahasa pemrograman. Eksekusi ini dilakukan dengan cara komunikasi pada inti komputasi (*kernel*). Pada standarnya, *jupyter notebook* hanya menyertakan *kernel python*. Namun, sebagai proyek *open source*, dimungkinkan untuk meningkatkan jumlah *kernel* yang tersedia misalnya *kernel* untuk Julia, R, Ruby, C/C++, Matlab, *Mathematica*, dll. (Granado E, C., dan Garcia E. D., 2018).

2.2.7. *Confusion Matrix*

Confusion matrix adalah alat yang biasa digunakan untuk mengukur kinerja dari algoritma dalam melakukan proses klasifikasi. Dengan *confusion matrix*, dapat diketahui seberapa banyak data yang diklasifikasi dengan benar pada label/ kelasnya sesuai dengan label/ kelas yang sebenarnya (Markoulidakis I dkk., 2021). Distribusi pada *confusion matrix* ditunjukkan pada Gambar 2.3.

		Prediksi	
		Positif	Negatif
Aktual	Positif	TP	FN
	Negatif	FP	TN

Gambar 2.3. Confusion Matrix (Markoulidakis I dkk., 2021)

Pada Gambar 2.3, TP merupakan singkatan dari *true positive* yang berarti data yang berada pada kondisi aktual memiliki kelas positif, diprediksi algoritma dengan tepat pada kelas positif. TN merupakan singkatan dari *true negative* yang berarti data yang berada pada kondisi aktual memiliki kelas negatif, diprediksi algoritma dengan tepat pada kelas negatif. FP merupakan singkatan dari *false*

positive yang berarti data yang berada pada kondisi aktual memiliki kelas negatif, diprediksi oleh algoritma memiliki kelas positif. Adapun FN merupakan singkatan dari *false negative* yang berarti data yang berada pada kondisi aktual memiliki kelas positif, diprediksi oleh algoritma memiliki kelas negatif.

Dari informasi *confusion matrix*, dapat dilakukan perhitungan untuk memperoleh nilai akurasi, presisi, *recall*, dan *f1-score* menggunakan persamaan-persamaan sebagaimana persamaan (2.7), (2.8), (2.9), dan (2.10).

$$Akurasi = \frac{TP+TN}{TP+TN+FP+FN} \times 100\% \quad (2.7)$$

$$Presisi = \frac{TP}{TP+FP} \times 100\% \quad (2.8)$$

$$Recall = \frac{TP}{TP+FN} \times 100\% \quad (2.9)$$

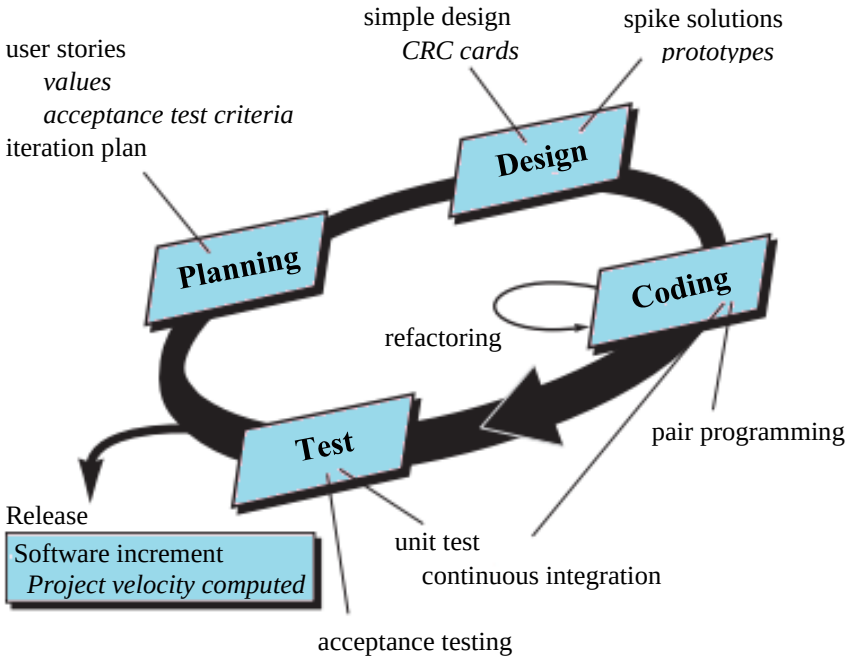
$$f1 - score = \frac{2 \times Presisi \times Recall}{Presisi+Recall} \times 100\% \quad (2.10)$$

Berdasarkan persamaan (2.4), nilai akurasi dihitung dengan membandingkan jumlah data yang diprediksi benar dengan jumlah data keseluruhan. Persamaan (2.5) menunjukkan nilai presisi dihitung dengan membandingkan jumlah data diprediksi benar pada kelas positif dengan semua jumlah data diprediksi pada kelas positif. Persamaan (2.6) menunjukkan nilai *recall* dihitung dengan membandingkan jumlah diprediksi benar pada kelas positif dengan jumlah data yang diprediksi benar pada kelas positif ditambah jumlah data yang diprediksi pada kelas negatif namun pada kondisi aktual memiliki kelas positif.

2.2.8. *Extreme Programming (XP)*

Extreme programming (XP) merupakan salah satu model *Agile* yang banyak digunakan sebagai metode pengembangan perangkat lunak. *Extreme programming*

menggunakan pendekatan berorientasi objek sebagai pilihan paradigma pengembangan dan mencakup seperangkat aturan dan praktik yang dilakukan dalam konteks empat aktivitas/ tahapan kerangka kerja, yaitu perencanaan, desain, pengkodean, dan pengujian. Ilustrasi proses *extreme programming* ditunjukkan pada Gambar 2.4.



Gambar 2.4. Tahapan Proses *Extreme Programming* (XP) (Pressman R. S., 2010)

Tahapan proses *extreme programming* dijelaskan sebagai berikut:

a. *Planning* (Perencanaan)

   Tahap perencanaan dimulai dengan mengumpulkan informasi atau data-data yang memungkinkan pengembang untuk memperoleh wawasan guna memahami konteks bisnis perangkat lunak yang akan dibuat untuk menghasilkan output yang dibutuhkan, dan mempertimbangkan fungsionalitasnya.

b. *Design* (Desain)

   Desain pada *extreme programming* mengacu pada prinsip "*Keep It Simple (KIS)*". Desain yang sederhana selalu lebih disukai daripada representasi yang lebih kompleks. Selain itu, desain memberikan

panduan implementasi untuk sebuah kasus sebagaimana adanya, tidak lebih dan tidak kurang. Untuk desain yang sulit, *extreme programming* dapat menggunakan *spike solution*, dimana prototipe desain dibuat langsung untuk diimplementasikan dan dievaluasi pada bagian tertentu dari desain secara keseluruhan.

c. *Coding* (Pengkodean)

   Setelah tahap perencanaan dan desain awal selesai, pengembang melakukan pengujian unit desain apakah sudah sesuai dengan *output* yang diharapkan pada tahap perencanaan. Setelah itu, pengembang melakukan implementasi kode hingga selesai. Untuk pengoptimalan sistem perangkat lunak, *estreme programming* mendukung adanya *refactoring* yang merupakan proses mengubah sistem perangkat lunak sedemikian rupa melalui modifikasi atau penyederhanaan struktur kode, namun tidak mengubah hasil keluaran yang diberikan. Pada proses ini, biasanya dilakukan *pair programming* dimana ada dua orang *programmer* yang bekerja sama di satu komputer untuk menulis detail kode dan memastikan kode-kode yang dibuat telah memenuhi standar desain.

d. *Test* (Pengujian)

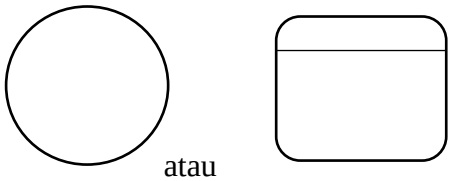
   Kode-kode yang ditulis kemudian diuji sehingga memberikan umpan balik kepada pengembang. Pengujian dapat dilakukan terus-menerus agar dapat memberikan peringatan dini bila terjadi kesalahan dalam implementasi kode pada sistem perangkat lunak. Setelah tahap ini, sistem perangkat lunak dapat dirilis dan dapat dikembangkan lebih lanjut apabila terdapat hal-hal yang perlu dimodifikasi.

2.2.9. Diagram Alir Data (*Data Flow Diagram*/ DFD)

Diagram alir data atau disebut juga *Data Flow Diagram*/ DFD merupakan salah satu alat yang dapat digunakan untuk memodelkan sistem. Diagram alir data yang paling utama disebut diagram konteks yang berada pada level tertinggi (*top level*) dan dapat dikembangkan sehingga memiliki tingkatan diagram alir data level

berikutnya, seperti DFD level 1, 2, dan seterusnya dengan maksud menampilkan lebih rinci proses-proses yang terjadi dalam sistem. Diagram alir data digambarkan dengan simbol-simbol, yaitu simbol entitas/ *terminator*, aliran data (*data flow*), proses, dan penyimpanan data (*data store*) seperti ditunjukkan pada Tabel 2.1.

Tabel 2.1. Daftar Simbol pada Diagram Alir Data (Soufitri, 2019)

Simbol	Keterangan
	Entitas (*terminator*)
	Aliran data (*data flow*)
 atau	Proses
	Penyimpanan data (*data store*)

Entitas/ *terminator* dapat berupa orang, organisasi, atau bagian yang ditandai dengan pemberian label pada simbolnya. Entitas dapat memberi dan menerima data dari sistem. Aliran data merupakan penghubung dari satu titik ke titik lainnya. Proses menggambar hal yang terjadi di dalam sistem dengan cara menerima *input*, melakukan pemrosesan *input*, lalu menghasilkan *output*, Penamaan proses dapat berupa nama sistem, subsistem, dan menggunakan kata

kerja. Penyimpanan data (*data store*) dapat berupa penyimpanan manual atau basis data yang terkomputerisasi (Soufitri, 2019).

2.2.10. *Django Framework*

Django adalah *framework* (kerangka kerja) web dengan bahasa pemrograman tingkat tinggi Python Django diterbitkan di bawah lisensi BSD yang menjamin bahwa aplikasi web dapat digunakan dan dimodifikasi secara bebas tanpa masalah, serta dapat digunakan secara gratis (Holovaty dan Kaplan-Moss, 2009). Dengan menggunakan Python, memungkinkan pengembang memanfaatkan semua *library python*. Hingga kini Django terus dikembangkan hingga versi 3.1 tahun 2020 dengan telah menambahkan fungsionalitas baru dan perbaikan *bug*, mulai dari dukungan untuk jenis database baru, mesin *template*, dan *caching*, hingga penambahan fungsi dan kelas tampilan "generik" (yang mengurangi jumlah kode yang harus ditulis pengembang untuk sejumlah tugas pemrograman). Django mendukung situs web multibahasa melalui sistem internasionalisasi bawaannya. Sistem ini membuat penerjemahan antarmuka menjadi tugas yang sangat sederhana (Dauzon dkk., 2016).

Kode Django ditulis menggunakan prinsip dan pola desain yang mendorong pembuatan kode yang dapat dipertahankan dan dapat digunakan kembali. Secara khusus, ini menggunakan prinsip *Don't Repeat Yourself* (DRY) sehingga tidak ada duplikasi yang tidak perlu dan mengurangi jumlah kode. Django juga melakukan pengelompokan fungsi terkait menjadi "aplikasi" yang dapat digunakan kembali dan pada tingkat yang lebih rendah, mengelompokkan kode terkait ke dalam modul (di sepanjang garis pola *Model View Controller* (MVC) (Holovaty dan Kaplan-Moss, 2009). Django membantu menghindari banyak kesalahan keamanan umum dengan menyediakan kerangka kerja yang telah direkayasa untuk "melakukan hal-hal yang benar" guna melindungi situs web secara otomatis. Misalnya, Django menyediakan cara yang aman untuk mengelola akun pengguna dan kata sandi, menghindari kesalahan umum seperti menempatkan informasi sesi di *cookie* yang mana itu rentan (sebaliknya *cookie* hanya berisi kunci, dan data aktual disimpan dalam basis data) atau langsung menyimpan kata sandi daripada kumpulan kata

sandi. Django memungkinkan perlindungan terhadap banyak kerentanan secara *default*, termasuk injeksi SQL, *scripting* lintas situs, pemalsuan permintaan lintas situs dan *clickjacking* (Jaiswal dan Kumar, 2015).