BAB II
METODE PENELITIAN

2.1 Objek Penelitian

Objek penelitian adalah data Twitter yang berisi cuitan *tweet* terkait *Quick Count* Pemilihan Presiden Indonesia tahun 2024. Data tersebut mencakup cuitan dari berbagai pengguna Twitter yang membahas, mengungkapkan opini, atau memberikan informasi seputar hasil sementara pemilihan presiden yang diumumkan melalui *Quick Count* oleh lembaga-lembaga survei. Data Twitter ini akan digunakan sebagai bahan utama untuk diklasifikasi lebih lanjut.

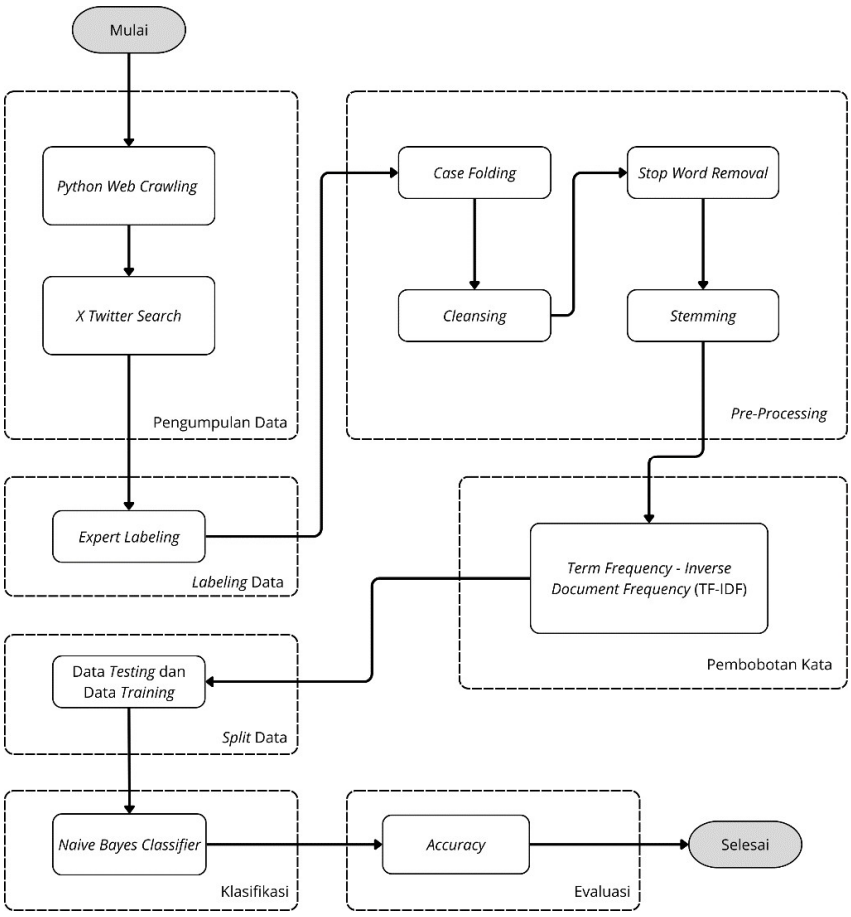
2.2 Alat dan Bahan

Dalam penelitian ini memanfaatkan berbagai alat perangkat lunak yang saling terintegrasi untuk melakukan klasifikasi teks. *Google Colab* versi 1.0.0 dipilih sebagai platform utama untuk menjalankan kode dan analisis, dengan *Python* versi 3.10.12 sebagai bahasa pemrograman dasarnya. Beberapa *library Python* yang digunakan dalam penelitian ini adalah (i) *Pandas* versi 2.0.3 untuk manipulasi dan analisis data setelah data diambil dan disimpan ke dalam file *CSV*, (ii) *NumPy* versi 1.25.2 untuk operasi numerik yang efisien, (iii) *Re* versi 2.2.1 untuk operasi pencocokan pola regex, (iv) *NLTK* versi 3.8.1 untuk pemrosesan bahasa alami, (v) *Scikit-learn* versi 1.2.2 untuk pembelajaran mesin dan analisis data, (vi) *Matplotlib* versi 3.7.1 untuk visualisasi data, (vii) *Sastrawi* versi 1.0.1 untuk stemming bahasa Indonesia. (viii) *Framework* tambahan yang digunakan adalah *Node.js* versi 14.16.0 untuk menjalankan alat *tweet-harvest*, serta *tweet-harvest* versi 2.6.0 untuk mengambil data *tweet* dari Twitter.

Bahan yang digunakan penelitian ini adalah dataset tweet terkait *Quick Count* pemilu di Indonesia. *Tweet* berbahasa Indonesia yang dikumpulkan pada periode 14-15 Februari 2024. Data tersebut menjadi sumber utama untuk analisis percakapan dan opini publik mengenai perhitungan cepat hasil pemilihan presiden di Indonesia.

2.3 Prosedur Penelitian

Penelitian ini mencakup tahapan dalam mengklasifikasi teks data X Twitter, dimulai dengan pengumpulan data melalui pencarian X Twitter dan *web crawling* menggunakan Python. Setelah data terkumpul, dilakukan *labeling* oleh ahli pelabelan data. Data yang telah di *labeling* kemudian melalui tahap *preprocessing* seperti *case folding*, *cleansing*, *stopwords removal*, dan *stemming*. Selanjutnya, fitur ekstraksi dilakukan dengan metode TF-IDF untuk pembobotan kata. Data yang telah diolah kemudian dibagi menjadi data pelatihan dan data pengujian. Klasifikasi teks dilakukan dengan *Naive Bayes classifier*. Kinerja model dievaluasi dengan mengukur akurasi yang dihasilkan. Gambar 2.1 menunjukkan alur penelitian secara keseluruhan.



Gambar 2.1 Alur Penelitian

2.3.1 Pengumpulan Data

Twitter menyediakan kunci antarmuka pemrograman aplikasi yang memungkinkan pengguna untuk mengakses data Twitter secara terprogram, namun dengan keterbatasan jumlah data yang dapat diambil. Pengumpulan data dilakukan menggunakan *Tweet Harvest*, sebuah alat yang memungkinkan pengguna untuk mengumpulkan lebih banyak data Twitter dapat diakses dan dijalankan melalui *Command Line Interface* (CLI) hanya dengan menggunakan *auth_token*. Proses *crawling* dilakukan dengan menentukan kata kunci pencarian, rentang tanggal pencarian, dan batas jumlah data yang akan diambil. Setelah proses *crawling* selesai, data yang berhasil diambil akan disimpan dalam format CSV untuk digunakan dalam klasifikasi selanjutnya (Darman, 2023). Tahapan pengumpulan data yang dilakukan dalam penelitian ini adalah sebagai berikut (Lampiran 2 Code Pengumpulan Data):

a. Pada tahap pertama, token autentikasi Twitter disimpan dalam variabel *twitter_auth_token*. Token ini diperlukan untuk mengakses API Twitter dan mengambil data *tweet* yang relevan.
b. Tahap kedua melibatkan instalasi paket *pandas* menggunakan *pip* dan *Node.js* melalui *apt-get*. Proses ini mencakup pembaruan paket, pengaturan repositori, dan verifikasi instalasi *Node.js*.
c. Pada tahap terakhir, parameter untuk pengambilan data tweet ditentukan. Kata kunci pencarian yang digunakan adalah *"quick count"*, dengan fokus pada *tweet* berbahasa Indonesia *(lang:id)* yang diunggah antara 14 Februari 2024 hingga 15 Februari 2024. *Tweet-harvest* kemudian dijalankan untuk mengambil data *tweet* sesuai parameter tersebut. Data yang dikumpulkan disimpan dalam format CSV, dengan jumlah maksimum *tweet* sesuai batas yang telah ditentukan.

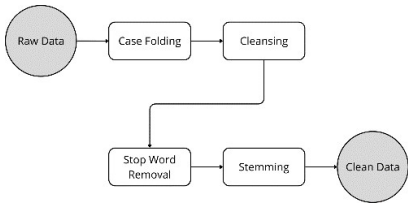
2.3.2 *Labeling* Data

Dalam proses melakukan Klasifikasi Teks pada data *tweet* sebagai bagian dari skripsi tugas akhir, peneliti mencari ahli dalam pelabelan data yang memiliki pengalaman dalam Labeling data dan memiliki pengetahuan mendalam tentang bahasa Indonesia. Setelah mengajukan permintaan pada website projects.co.id, peneliti berhasil mendapatkan tenaga ahli yang sesuai dengan kriteria, yaitu yang memiliki kualifikasi dan pengalaman yang relevan. Calon tenaga ahli telah memasukkan penawaran dengan mencantumkan pekerjaan saat ini, pengalaman yang relevan dalam pelabelan data, serta gelar akademik yang dimiliki. Pada lampiran CV dari ahli pelabelan data telah dicantumkan untuk memberikan informasi lebih lanjut mengenai kualifikasi dan pengalaman yang dimilikinya dalam melakukan *labeling* data. Dasar atau acuan yang digunakan oleh ahli pelabelan data dalam melakukan pelabelan pada data adalah sebagai berikut (Lampiran 1 CV *Expert Labeling*):

a. Positif: *Quick count* telah terbukti memberikan hasil yang akurat dan dapat dipercaya dalam berbagai pemilihan. Metode ini membantu memberikan gambaran awal yang valid tentang hasil pemungutan suara, mendukung harapan masyarakat akan transparansi proses demokrasi. Banyak pihak menyambut baik pelaksanaan *Quick Count*, mengikuti perkembangannya dengan antusias, dan menggunakan hashtag *#QuickCount* untuk berbagi informasi terkini. Media pun turut menyajikan berita-berita objektif mengenai hasil *Quick Count*, membantu publik memahami situasi dengan lebih baik.
b. Negatif: Di sisi lain, sejumlah pihak mengungkapkan keraguan terhadap metodologi dan akurasi *Quick Count*. Beberapa kritik muncul terkait potensi penggiringan opini publik atau manipulasi data yang menimbulkan perdebatan di media sosial. Sebagian masyarakat mengekspresikan kekecewaan atau kemarahan terhadap hasil *Quick Count* yang tidak sesuai ekspektasi mereka, bahkan ada yang menggunakan bahasa kasar atau sindiran untuk mengungkapkan ketidakpuasan. Keraguan juga muncul terkait objektivitas

lembaga pelaksana *Quick Count*, dengan tuduhan adanya kepentingan tertentu di balik penyelenggaraannya.

2.3.3 Pre-Processing



Gambar 2.2 *Pre-Processing*

Gambar 2.2 *Preprocessing* menunjukkan tahapan-tahapan yang diperlukan dalam proses *preprocessing* teks. *Preprocessing* adalah tahap yang krusial sebelum memulai sebuah penelitian. Text *preprocessing* dilakukan untuk memastikan bahwa data awal melewati serangkaian tahapan sehingga menjadi siap digunakan sepenuhnya dalam proses klasifikasi (Albab et al., 2023). Tahapan *preprocessing* yang dilakukan dalam penelitian ini adalah sebagai berikut (Lampiran 3 Code *Preprocessing*):

a. Pada tahap pertama, *Case folding* merupakan tahap di mana semua huruf dalam teks diubah menjadi huruf kecil Python *string lower method*. Tujuannya adalah untuk memastikan konsistensi dalam format huruf sehingga semua teks diproses dalam bentuk huruf kecil.
b. Pada tahap kedua, *Cleansing* pembersihan teks bertujuan untuk menghapus kata-kata atau karakter yang tidak diinginkan yang tidak relevan untuk klasifikasi teks. Ini termasuk penghapusan karakter khusus seperti URL, *emoji, username*, angka, dan simbol-simbol kecuali *underscore*, karena tidak memberikan kontribusi pada penilaian sentimen.
c. Tahap ketiga adalah *Stopword Removal* tahap ini melibatkan penghapusan kata-kata umum yang tidak memberikan makna tambahan dalam klasifikasi teks. Kata-kata tersebut

dikenal sebagai *stopwords*. Penghapusan *stopwords* membantu dalam fokus pada kata-kata yang lebih informatif dan signifikan dalam menentukan sentimen.

d. Pada tahap terakhir proses *Stemming* merupakan langkah selanjutnya yang bertujuan untuk mengubah kata-kata ke dalam bentuk dasarnya dengan menghapus imbuhan-imbuhan yang ada. Contohnya, kata-kata seperti "berlari," "berlari-lari," dan "lari" dapat diubah menjadi bentuk dasar "lari." Ini membantu dalam penyederhanaan kata-kata dengan akar yang sama untuk klasifikasi teks dengan lebih akurat.

2.3.4 Pembobotan Kata

Term Frequency-Inverse Document Frequency (TF-IDF) adalah proses penting dalam analisis data teks, di mana pembobotan dilakukan menggunakan algoritma TF-IDF untuk memberikan skor frekuensi setiap *term* dalam sebuah dokumen. *Term Frequency* (TF) menentukan seberapa sering *term* muncul dalam dokumen, sedangkan *Document Frequency* (DF) menunjukkan jumlah dokumen di mana *term* tersebut muncul (Hamka et al., 2022). Salah satu *transformer* atau metode untuk melakukan transformasi data teks menjadi representasi numerik adalah *TfidfVectorizer* yang mencakup ekstraksi fitur kata dan penghitungan frekuensi kemunculannya. Dengan demikian, *TfidfVectorizer* tidak hanya mengonversi teks tetapi juga menghitung bobot TF-IDF untuk setiap kata dalam dataset (Surya et al., 2024). Tahapan pembobotan kata yang dilakukan dalam penelitian ini adalah sebagai berikut (Lampiran 5 Code Pembobotan Kata):

a. Pada tahap pertama, *library pandas, numpy, dan TfidfVectorizer* dari *scikit-learn* diimpor. Kemudian, file CSV hasil *preprocessing* dibaca menggunakan *pandas* dan disimpan dalam *DataFrame* df. Kolom *'stemming'* dari *DataFrame* df diambil dan disimpan dalam *list documents*, sedangkan kolom 'Sentimen' disimpan dalam *list sentiments*. Jumlah dokumen dihitung dan disimpan dalam variabel N.

b. Tahap kedua, objek *TfidfVectorizer* diinisialisasi untuk melakukan pembobotan TF-IDF pada data teks. Kemudian, metode *fit_transform()* dipanggil pada objek *vectorizer* dengan menggunakan data teks *documents* sebagai input. Hasilnya disimpan dalam *tfidf_matrix*, yang merupakan matriks TF-IDF untuk setiap dokumen.

c. Tahap ketiga melibatkan konversi *tfidf_matrix* menjadi *DataFrame pandas tfidf_df* dengan kolom berisi daftar *term* yang dihasilkan oleh *vectorizer. get_feature_names_out().*

d. Pada tahap keempat, hasil pembobotan TF-IDF ditampilkan. Kemudian, 40 elemen pertama dari matriks TF-IDF yang tidak nol ditampilkan, diikuti oleh tiga titik sebagai pemisah, dan terakhir 3 elemen terakhir dari matriks TF-IDF yang tidak nol ditampilkan.

e. Tahap terakhir kode untuk menyimpan hasil TF-IDF ke dalam file Excel disediakan. DataFrame tfidf_df yang berisi hasil pembobotan TF-IDF disimpan ke dalam file Excel tanpa menyertakan indeks.

Nilai ekstraksi fitur dihitung menggunakan Persamaan (2.1) sebagai berikut:

$$TF - IDF(t,d) = TF(t,d) * IDF(t) \tag{2.1}$$

Keterangan:

a. $TF(t,d)$ adalah *term frequency* dari kata t dalam dokumen d, yang dihitung sebagai berikut:

$$TF(t,d) = \frac{jumlah\ kemunculan\ kata\ t\ dalam\ dokumen\ d}{total\ jumlah\ kata\ dalam\ dokumen\ d} \tag{2.2}$$

b. $IDF(t)$ adalah *inverse document frequency* dari kata t, yang dihitung sebagai berikut:

$$IDF(t) = log\frac{total\ jumlah\ dokumen\ dalam\ koleksi}{jumlah\ dokumen\ yang\ mengandung\ kata\ t + 1} \tag{2.3}$$

2.3.5 *Split* Data

Split Data adalah proses membagi dataset yang diperoleh menjadi Ada dua komponen utama, yakni *training* data dan *testing* data. Data pelatihan digunakan untuk melatih model klasifikasi *Naive Bayes* agar dapat mempelajari pola dan karakteristik data. Sedangkan data uji digunakan untuk menilai performa model yang sudah dilatih pada data baru yang belum pernah diproses sebelumnya (Putri et al., 2023). Dalam penelitian tentang klasifikasi komentar toxic menggunakan TF-IDF dan Naive Bayes, rasio 80:20 menghasilkan model dengan akurasi tertinggi. Rasio 80:20 dipilih karena memberikan keseimbangan yang optimal antara jumlah data yang cukup untuk melatih model dan data yang cukup untuk menguji model secara akurat. Hal ini menunjukkan bahwa rasio 80:20 memberikan hasil yang optimal dalam menjaga keseimbangan antara data pelatihan dan pengujian sehingga model dapat dievaluasi dengan baik pada data yang belum pernah dilihat sebelumnya (Sidiq et al., 2020). Tahapan *split* data yang dilakukan dalam penelitian ini adalah sebagai berikut (Lampiran 6 Code *Split* Data):

a. Pada tahap pertama, fungsi-fungsi yang diperlukan diimpor dari *library scikit-learn* dan matplotlib. Fungsi *train_test_split dari scikit-learn* digunakan untuk membagi dataset menjadi data latih dan data uji, sementara *matplotlib* digunakan untuk membuat visualisasi grafik.
b. Tahap kedua melibatkan pemrosesan data dan pembagian dataset, di mana data teks diubah menjadi representasi numerik dan kemudian dataset dibagi menjadi data latih dan data uji dengan rasio 80:20. *Random_state=42* digunakan untuk menetapkan *seed* generator angka acak, menjamin pembagian data yang konsisten dan hasil yang dapat direproduksi setiap kali kode dijalankan.
c. Pada tahap ketiga, jumlah sampel untuk total dataset, data latih, dan data uji dihitung. Selanjutnya, sebuah visualisasi *bar chart* dibuat menggunakan *matplotlib* untuk menampilkan distribusi sampel dalam dataset.

d. Tahap keempat melibatkan penyempurnaan visualisasi. Jumlah sampel ditambahkan di atas setiap *bar chart*, label sumbu y dan judul grafik ditambahkan, serta persentase untuk data latih dan data uji ditampilkan pada bar masing-masing.

e. Pada tahap terakhir, *layout* grafik diatur menggunakan *plt.tight_layout()* untuk memastikan semua elemen grafik terlihat dengan jelas dan tidak tumpang tindih. Grafik ditampilkan menggunakan fungsi *plt.show()*.

2.3.6 Klasifikasi

Klasifikasi *Naive Bayes* adalah metode klasifikasi yang mengandalkan *Teorema Bayes*, yang awalnya diusulkan oleh ilmuwan Inggris bernama *Thomas Bayes*. Metode ini memanfaatkan pendekatan probabilitas dan statistik untuk memproyeksikan kemungkinan kejadian di masa depan berdasarkan pengalaman masa lalu, sehingga sering disebut sebagai *Teorema Bayes*. *Naive Bayes classifier* menyediakan kemudahan bagi penggunanya seperti prosesnya yang cepat, mudah diterapkan, cukup sederhana dalam strukturnya dan sangat efektif (Sriani, Suhardi, 2024). Salah satu pengembangan dari algoritma klasifikasi *Naïve Bayes* adalah *Multinomial Naïve Bayes* yaitu model dimana kelas tidak hanya ditentukan dengan kata yang muncul, tapi juga dengan jumlah kemunculannya sehingga cocok untuk klasifikasi teks atau dokumen (Husada & Toba, 2020). *Teorema Bayes* dirumuskan dalam Persamaan (2.4) sebagai berikut:

$$P(C|X) = \frac{P(X|C) \cdot P(C)}{P(X)} \tag{2.4}$$

Keterangan:

a. X: Data yang kelasnya belum diketahui
b. C: Hipotesis bahwa data X merupakan kelas tertentu
c. P(C|X): Probabilitas hipotesis C benar jika diberikan data X (*probabilitas posterior*)
d. P(C): Probabilitas awal hipotesis C sebelum melihat data X (*probabilitas prior*)

e. P(X|C): Probabilitas munculnya data X jika hipotesis C benar (*likelihood*)

f. P(X): Probabilitas kemunculan data X secara umum (*probabilitas evidence*)

P(C|X) adalah probabilitas bahwa suatu data X termasuk ke dalam kelas C, yang dihitung pada probabilitas prior P(C), likelihood P(X|C), dan probabilitas evidence P(X). Semakin tinggi nilai P(C|X), semakin besar kemungkinan data X termasuk ke dalam kelas C.

Confusion matrix adalah alat yang digunakan untuk mengukur performa model klasifikasi. *Confusion matrix* memberikan gambaran jumlah prediksi yang akurat dan tidak akurat yang dibuat oleh model (Fikri et al., 2020). Tahapan evaluasi yang dilakukan dalam penelitian ini adalah sebagai berikut (Lampiran 7 Code Klasfikasi):

a. Pada tahap pertama, fungsi-fungsi yang diperlukan diimpor dari *library scikit-learn*, *seaborn*, dan *matplotlib*. Ini mencakup metrik evaluasi seperti *classification_report*, *confusion_matrix*, dan *accuracy_score*, serta *MultinomialNB* untuk *Naive Bayes.*

b. Tahap kedua melibatkan inisialisasi objek klasifikasi *Naive Bayes Multinomial* menggunakan *MultinomialNB()*.

c. Pada tahap ketiga, metode fit digunakan untuk melatih model dengan data latih *(X_train, y_train)*.

d. Tahap keempat melibatkan penggunaan metode predict untuk melakukan prediksi pada data uji *(X_test)* menggunakan model yang telah dilatih sebelumnya.

e. Pada tahap kelima, confusion matrix dihitung menggunakan fungsi confusion_matrix, dengan label sebenarnya *(y_test)* dan label prediksi sebagai input.

f. Tahap terakhir melibatkan visualisasi confusion matrix menggunakan *seaborn*. *Heatmap* dibuat dengan label yang sesuai untuk sumbu x dan y, serta judul plot.

Struktur *Confusion matrix* untuk klasifikasi biner ditunjukkan pada Tabel 2.1 sebagai berikut:

Tabel 2.1 *Confusion matrix*

		Predicted Class	
		Positive	***Negative***
Actual Class	***Positive***	*True Positive (TP)*	*False Negative (FN)*
	Negative	*False Positive (FP)*	*True Negative (TN)*

Keterangan:

a. *True Positive* (TP): Jumlah data positif yang diprediksi dengan akurat sebagai positif.

b. *False Positive* (FP): Jumlah data negatif yang keliru diprediksi sebagai positif.

c. *False Negative* (FN): Jumlah data positif yang keliru diprediksi sebagai negatif.

d. *True Negative* (TN): Jumlah data negatif yang diprediksi dengan tepat sebagai negatif.

2.3.7 Evaluasi

Akurasi adalah cara untuk mengukur seberapa tepat suatu model dalam membuat prediksi. Metode ini membandingkan seberapa dekat hasil prediksi dengan nilai yang sebenarnya untuk keseluruhan data (Farokhah, 2020). Akurasi model dihitung dengan Persamaan (2.5) sebagai berikut:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{2.5}$$