

SKRIPSI

**ANALISIS SENTIMEN DAN DETEKSI BUZZER DI TWITTER DALAM
PREDIKSI PILKADA DKI JAKARTA 2017**

**SENTIMENT ANALYSIS AND BUZZER DETECTION IN TWITTER
FOR 2017 JAKARTA GUBERNATORIAL ELECTION PREDICTION**



FELIX GIOVANNI VIRGO

14/365960/PA/16167

PROGRAM STUDI ILMU KOMPUTER

DEPARTEMEN ILMU KOMPUTER DAN ELEKTRONIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS GADJAH MADA

2017

SKRIPSI

**ANALISIS SENTIMEN DAN DETEKSI BUZZER DI TWITTER DALAM
PREDIKSI PILKADA DKI JAKARTA 2017**

**SENTIMENT ANALYSIS AND BUZZER DETECTION IN TWITTER
FOR 2017 JAKARTA GUBERNATORIAL ELECTION PREDICTION**

Diajukan untuk memenuhi salah satu syarat memperoleh derajat Sarjana Ilmu
Komputer



FELIX GIOVANNI VIRGO

14/365960/PA/16167

**PROGRAM STUDI ILMU KOMPUTER
DEPARTEMEN ILMU KOMPUTER DAN ELEKTRONIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS GADJAH MADA**

2017

HALAMAN PERSETUJUAN

SKRIPSI

**ANALISIS SENTIMEN DAN DETEKSI BUZZER DI TWITTER DALAM
PREDIKSI PILKADA DKI JAKARTA 2017**

Telah dipersiapkan dan disusun oleh

FELIX GIOVANNI VIRGO

14/365960/PA/16167

Telah disetujui oleh Pembimbing
pada tanggal 8 Februari 2018

Isna Alfi Bustoni, S.T., M.Eng.

Pembimbing

PERNYATAAN

Dengan ini saya menyatakan bahwa Skripsi ini tidak terdapat karya yang pernah diajukan untuk memperoleh gelar kesarjanaan di suatu Perguruan Tinggi, dan sepanjang pengetahuan saya juga tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis diacu dalam naskah ini dan disebutkan dalam daftar pustaka.

Yogyakarta, 8 Februari 2018

Felix Giovanni Virgo

PRAKATA

Puji syukur ke hadirat Tuhan Yang Maha Esa atas limpahan rahmat, kasih, karunia, serta petunjuk-Nya sehingga tugas akhir berupa penyusunan skripsi ini telah terselesaikan dengan baik.

Dalam penyusunan tugas akhir untuk mendapatkan gelar sarjana ini, penulis telah banyak mendapatkan arahan, bantuan, serta dukungan dari berbagai pihak. Oleh karena itu pada kesempatan ini penulis mengucapkan terima kasih kepada:

1. Ibu, Bapak, Adik, beserta keluarga yang senantiasa memberikan motivasi dan dukungan dalam bentuk fisik maupun moril kepada penulis,
2. Ibu Isna Alfi Bustoni, S.T., M.Eng. selaku dosen pembimbing skripsi yang telah berkenan meluangkan waktu, pikiran, dan senantiasa memberikan bimbingan, saran, serta arahan selama proses penelitan hingga penulisan tugas akhir,
3. Ibu Dra. Leliwati Djohan yang dengan sukarela telah meluangkan waktu, pikiran, dan tenaga untuk mengevaluasi data dari penulis,
4. seluruh dosen Fakultas Matematika dan Ilmu Pengetahuan Alam, khususnya program studi Ilmu Komputer yang telah memberikan banyak sekali ilmu yang bermanfaat,
5. serta pihak-pihak lain yang telah membantu dalam penyelesaian skripsi ini yang tidak dapat disebutkan satu per satu.

Akhir kata penulis berharap semoga skripsi ini dapat memberikan manfaat bagi kita semua, terutama bagi perkembangan ilmu pengetahuan serta perkembangan Ilmu Komputer dan Teknologi Informasi.

Yogyakarta, 8 Februari 2018

Penulis

DAFTAR ISI

HALAMAN PERSETUJUAN	iii
PERNYATAAN	iv
PRAKATA	v
DAFTAR ISI	vi
DAFTAR TABEL	ix
DAFTAR GAMBAR	xi
INTISARI	ii
ABSTRACT	ii
BAB I : PENDAHULUAN	1
1.1 Latar Belakang Masalah	1
1.2 Rumusan Masalah	3
1.3 Batasan Masalah	3
1.4 Tujuan Penelitian	4
1.5 Manfaat Penelitian	4
BAB II : TINJAUAN PUSTAKA	6
BAB III : LANDASAN TEORI	10
3.1 Data Mining	10
3.2 Information Retrieval	10
3.3 Analisis Sentimen	11
3.4 Naive Bayes	11
3.4.1 Multinomial Naive Bayes	12
3.4.2 Gaussian Naive Bayes	15
3.5 Buzzer Detection	16
3.6 Pengujian	16
3.6.1 K-fold Cross Validation	17
3.6.2 Perhitungan Performa	17
3.6.3 Mean Absolute Error	20
BAB IV : ANALISIS DAN RANCANGAN	21
4.1 Analisis Sistem	21
4.1.1 Deskripsi Sistem	21
4.1.2 Deskripsi Data	22
4.2 Perancangan Sistem	26
4.2.1 Pengambilan Data Tweet	31

4.2.2 Pengambilan Data Informasi Akun	33
4.2.3 Pre-processing Data	35
4.2.4 Pelatihan Model Klasifikasi Buzzer	49
4.2.5 Pelatihan Model Klasifikasi Sentimen	52
4.2.6 Penghapusan Tweet Buzzer	55
4.2.7 Pengujian Model Klasifikasi	55
4.2.8 Perbandingan Metode Prediksi Hasil Pilkada	57
4.2.9 Skema Evaluasi Metode Prediksi Hasil Pilkada	59
BAB V : IMPLEMENTASI	25
5.1 Spesifikasi Hardware dan Software	61
5.2 Implementasi Sistem	61
5.2.1 Pengambilan Data	61
5.2.2 Pre-processing Data	65
5.2.3 Pelatihan Model Klasifikasi Sentimen	68
5.2.4 Pelatihan Model Klasifikasi Buzzer	76
5.2.5 Penghapusan Tweet Buzzer	82
5.2.6 Pengujian Model Klasifikasi Buzzer	83
5.2.7 Pengujian Model Klasifikasi Sentimen	88
5.2.8 Evaluasi Metode Prediksi Hasil Pilkada	91
BAB VI : HASIL DAN PEMBAHASAN	93
6.1 Hasil Pengujian Klasifikasi Buzzer	93
6.2 Hasil Pengujian Klasifikasi Sentimen	94
6.2.1 Hasil Pengujian Klasifikasi Sentimen Untuk Pasangan Ahok-Djarot	94
6.2.2 Hasil Pengujian Klasifikasi Sentimen Untuk Pasangan Anies-Sandi	95
6.3 Hasil Klasifikasi Buzzer Pada Data Tes	96
6.4 Hasil Klasifikasi Sentimen Pada Data Tes	97
6.4.1 Hasil Klasifikasi Sentimen Pasangan Ahok-Djarot Pada Data Tes	97
6.4.2 Hasil Klasifikasi Sentimen Pasangan Anies-Sandi Pada Data Tes	98
6.5 Hasil Penghapusan Tweet Buzzer	99
6.6 Hasil dan Analisis Prediksi Pilkada DKI Jakarta 2017	100
BAB VII : PENUTUP	106
7.1 Kesimpulan	106
7.2 Saran	106
REFERENSI	108

LAMPIRAN.....	111
---------------	-----

DAFTAR TABEL

Tabel 2.1 : Tinjauan Pustaka	8
Tabel 4.1 : Spesifikasi data latih klasifikasi sentimen	23
Tabel 4.2 : Sampel data latih klasifikasi sentimen	24
Tabel 4.3 : Spesifikasi data latih klasifikasi buzzer	24
Tabel 4.4 : Sampel data latih klasifikasi buzzer	25
Tabel 4.5 : Sampel data tes klasifikasi sentimen	25
Tabel 4.6 : Sampel data tes klasifikasi buzzer	26
Tabel 4.7 : Fitur dari buzzer	27
Tabel 4.8 : Daftar kata kunci dalam proses pencarian tweet	31
Tabel 4.9 : Contoh tweet beserta labelnya	32
Tabel 4.10 : Contoh informasi akun beserta labelnya	34
Tabel 4.11 : Ilustrasi proses case folding	35
Tabel 4.12 : Ilustrasi proses URLs removal	37
Tabel 4.13 : Ilustrasi proses brackets removal	39
Tabel 4.14 : Ilustrasi proses mentions removal	41
Tabel 4.15 : Ilustrasi proses non-alphanumeric characters removal	43
Tabel 4.16 : Ilustrasi proses RT removal	45
Tabel 4.17 : Ilustrasi proses stopwords removal	46
Tabel 4.18 : Contoh informasi akun yang akan melalui proses Gaussian Naive Bayes	49
Tabel 4.19 : Contoh informasi akun yang akan melalui proses Gaussian Naive Bayes setelah dilakukan Min-Max Normalization	50
Tabel 4.20 : Contoh hasil perhitungan mean dan variansi untuk kelas normal	50
Tabel 4.21 : Contoh hasil perhitungan mean dan variansi untuk kelas buzzer	51
Tabel 4.22 : Contoh hasil perhitungan probabilitas menggunakan fungsi Gaussian	52
Tabel 4.23 : Contoh tweet yang akan melalui proses Multinomial Naive Bayes	52
Tabel 4.24 : Contoh hasil perhitungan probabilitas prior	53
Tabel 4.25 : Contoh hasil perhitungan probabilitas kondisional untuk masing-masing token pada tiap kelas	53
Tabel 6.1 : Jumlah masing-masing kelas yang dihasilkan dalam proses klasifikasi buzzer pada data tes	97
Tabel 6.2 : Jumlah masing-masing kelas yang dihasilkan dalam proses klasifikasi sentimen pasangan Ahok-Djarot pada data tes	98

Tabel 6.3 : Jumlah masing-masing kelas yang dihasilkan dalam proses klasifikasi sentimen pasangan Ahok-Djarot pada data tes	99
Tabel 6.4 : Jumlah tweet yang dihasilkan oleh buzzer untuk masing-masing pasangan calon dan sentimen pada seluruh data	100
Tabel 6.5 : Jumlah tweet untuk masing-masing pasangan caklon dan sentimen pada seluruh data	100
Tabel 6.6 : Jumlah tweet untuk masing-masing pasangan calon dan sentimen pada seluruh data sesudah dilakukan proses buzzer detection	100
Tabel 6.7 : Persentase tweet yang dihasilkan oleh buzzer untuk masingmasing kelas sentimen	101
Tabel 6.8 : Persentase tweet yang dihasilkan oleh buzzer untuk masingmasing pasangan calon	101
Tabel 6.9 : Jumlah suara sah Pilkada DKI Jakarta 2017 putaran kedua	103
Tabel 6.10 : Tabel perbandingan Mean Absolute Error antara hasil survei independen dengan metode berbasis data Twitter yang dilakukan pada penelitian ini	103

DAFTAR GAMBAR

Gambar 3.1: Ilustrasi Confusion Matrix	18
Gambar 3.2: Ilustrasi Confusion Matrix 3 kelas beserta rumus perhitungan performa	20
Gambar 4.1: Skema Perancangan Sistem.....	30
Gambar 4.2: Alur proses pengambilan data tweet	32
Gambar 4.3: Alur proses pengambilan data informasi akun.....	34
Gambar 4.4: Alur proses case folding.....	36
Gambar 4.5: Alur proses URLs removal	38
Gambar 4.6: Ilustrasi proses regular expression URLs removal.....	38
Gambar 4.7: Alur proses brackets removal.....	40
Gambar 4.8: Ilustrasi proses regular expression brackets removal.....	40
Gambar 4.9: Alur proses mentions removal	42
Gambar 4.10: Ilustrasi proses regular expression mentions removal	42
Gambar 4.11: Alur proses non-alphanumeric characters removal.....	44
Gambar 4.12: Ilustrasi proses regular expression non-alphanumeric characters removal.....	44
Gambar 4.13: Alur proses RT removal.....	46
Gambar 4.14: Alur proses stopwords removal	48
Gambar 4.15: Alur proses penghapusan tweet buzzer	55
Gambar 4.16: Ilustrasi proses 10-fold cross validation.....	56
Gambar 4.17: Ilustrasi tiga metode prediksi hasil pilkada DKI Jakarta 2017 tanpa melalui proses buzzer detection	58
Gambar 4.18: Ilustrasi tiga metode prediksi hasil pilkada DKI Jakarta 2017 sesudah melalui proses buzzer detection	59
Gambar 4.19: Skema evaluasi metode prediksi hasil pilkada.....	60
Gambar 5.1: Bagian program untuk melakukan proses scraping pada Twitter	62
Gambar 5.2: Bagian program untuk membuat objek API Twitter.....	63
Gambar 5.3: Bagian program untuk melakukan proses look up data profil pengguna Twitter	64
Gambar 5.4: Bagian program untuk menyimpan data profil pengguna Twitter ...	65
Gambar 5.5: Bagian program untuk melakukan case folding.....	65
Gambar 5.6: Bagian program untuk URLs removal	65
Gambar 5.7: Bagian program untuk brackets removal	66
Gambar 5.8: Bagian program untuk mentions removal	66

Gambar 5.9: Bagian program untuk non-alphanumeric characters removal	66
Gambar 5.10: Bagian program untuk rt removal	67
Gambar 5.11: Bagian program untuk stopwords removal	68
Gambar 5.12: Bagian program untuk mengimpor data latih dan data tes.....	69
Gambar 5.13: Bagian program untuk ekstraksi fitur.....	70
Gambar 5.14: Contoh kosakata data latih dan data tes	70
Gambar 5.15: Contoh hasil transformasi vektor dalam bentuk array	71
Gambar 5.16: Bagian program untuk menghitung probabilitas prior	72
Gambar 5.17: Bagian program untuk menghitung probabilitas kondisional	73
Gambar 5.18: Bagian program untuk mencari setiap elemen yang sama dengan suatu nilai tertentu pada variabel 'classlabel.....	73
Gambar 5.19: Bagian program untuk mencari indeks kata-kata pada data tes yang hanya terdapat di vocabulary	74
Gambar 5.20: Bagian utama program klasifikasi sentimen	76
Gambar 5.21: Bagian program untuk melakukan normalisasi data buzzer	77
Gambar 5.22: Bagian program untuk menghitung mean dan variansi untuk masing-masing fitur	79
Gambar 5.23: Bagian program untuk menghitung fungsi Gaussian	79
Gambar 5.24: Bagian program untuk melakukan klasifikasi buzzer	81
Gambar 5.25: Bagian utama program klasifikasi buzzer	81
Gambar 5.26: Bagian program untuk menyimpan daftar buzzer yang terklasifikasi	82
Gambar 5.27: Bagian program untuk mengimpor data dalam proses penghapusan tweet buzzer	82
Gambar 5.28: Bagian program untuk menghapus tweet-tweet yang dihasilkan oleh akun buzzer	83
Gambar 5.29: Bagian program untuk menyimpan tweet-tweet yang bersih dari buzzer	83
Gambar 5.30: Bagian program untuk melakukan proses normalisasi data buzzer pada saat proses 10-fold cross validation.....	84
Gambar 5.31: Bagian program untuk melakukan proses k-fold cross validation pada klasifikasi buzzer	86
Gambar 5.32: Bagian utama program pengujian model klasifikasi buzzer	87
Gambar 5.33: Bagian program untuk melakukan pembagian data menjadi bagian tweet dan bagian label	88
Gambar 5.34: Bagian program untuk melakukan proses k-fold cross validation pada klasifikasi sentimen	90

Gambar 5.35: Bagian utama program pengujian model klasifikasi sentimen	91
Gambar 5.36: Bagian program untuk menghitung persentase suara dan Mean Absolute Error	92
Gambar 6.1: Hasil pengujian klasifikasi buzzer menggunakan 10-fold cross validation	94
Gambar 6.2: Hasil pengujian klasifikasi sentimen pasangan Ahok-Djarot menggunakan 10-fold cross validation	95
Gambar 6.3: Hasil pengujian klasifikasi sentimen pasangan Anies-Sandi menggunakan 10-fold cross validation	96
Gambar 6.4: Contoh hasil klasifikasi buzzer pada data tes.....	97
Gambar 6.5: Contoh hasil klasifikasi sentimen pasangan Ahok-Djarot pada data tes	98
Gambar 6.6: Contoh hasil klasifikasi sentimen pasangan Anies-Sandi pada data tes	99

INTISARI

ANALISIS SENTIMEN DAN DETEKSI BUZZER DI TWITTER DALAM PREDIKSI PILKADA DKI JAKARTA 2017

Felix Giovanni Virgo
14/365960/PA/16167

Dengan menggunakan Twitter sebagai sumber data, penelitian ini menyajikan sebuah pendekatan untuk memprediksi hasil Pilkada DKI Jakarta 2017 dan menganalisis pengaruh tweet yang dihasilkan oleh *buzzer* dalam pilkada tersebut. Data Twitter yang mudah dikumpulkan dieksplorasi agar dapat dimanfaatkan sebagai alat pendukung untuk memahami opini publik. Pertama-tama, data di Twitter dikumpulkan selama masa kampanye. Kedua, deteksi *buzzer* otomatis dilakukan pada data Twitter untuk menghapus tweet yang dihasilkan oleh bot komputer, pengguna yang dibayar, serta pengguna fanatik yang biasanya akan menjadi *noise* dalam data. Ketiga, analisis sentimen dilakukan untuk menentukan polaritas sentimen untuk masing-masing tweet yang terkait dengan masing-masing kandidat. Dan akhirnya, untuk memprediksi hasil pilkada, jumlah tweet yang mendukung masing-masing kandidat sebelum dan sesudah dilakukan proses deteksi *buzzer* dibandingkan untuk mendapatkan persentase suara dukungan antar kandidat.

Percobaan ini menunjukkan bahwa penggunaan hanya sentimen positif dikombinasikan dengan teknik deteksi *buzzer* mampu menghasilkan prediksi terbaik dengan tingkat kesalahan 0,97%, lebih baik dibandingkan hasil prediksi yang dilakukan oleh beberapa lembaga survei independen. Penggunaan hanya sentimen positif dalam prediksi pilkada menunjukkan hasil terbaik dengan tingkat kesalahan rata-rata 2,04%. Dari penelitian ini ditemukan bahwa deteksi *buzzer* dapat mengurangi tingkat kesalahan hasil prediksi pilkada dengan rata-rata penurunan sebesar 0,85%. Selain itu, ditemukan juga bahwa *buzzer* cenderung menghasilkan tweet yang mendukung kandidat pilihan mereka lebih banyak dari tweet yang menjelekkan lawan mereka.

Kata kunci: Twitter, prediksi pilkada, analisis sentimen, deteksi buzzer.

ABSTRACT

SENTIMENT ANALYSIS AND BUZZER DETECTION IN TWITTER FOR 2017 JAKARTA GUBERNATORIAL ELECTION PREDICTION

Felix Giovanni Virgo
14/365960/PA/16167

By using Twitter as the main resource, this research presents an approach for predicting the results of 2017 Jakarta Gubernatorial Election and analyzing the influence of tweets generated by buzzers in the election. The possibility of easy-to-gather Twitter data was explored to be utilized as a supporting tool to understand public opinion. First, Twitter data were collected during the campaign period. Second, automatic buzzer detection was performed on Twitter data to remove those tweets generated by computer bots, paid users, and fanatic users that usually become noise in the data. Third, sentiment analysis was performed to assign sentiment polarity for each tweet related to each candidates. And finally, to predict the election results, the number of tweets that support each candidate before and after buzzer detection were compared to get the percentage of supporting votes among candidates.

The experiment shows that the use of positive-sentiment-only combined with buzzer detection technique was capable of producing the best prediction with an error rate of 0.97%, which is better than the prediction results published by several independent survey institutions. The use of positive-sentiment-only in the election prediction showed the best result with an average error rate of 2.04%. From this research it was found that buzzer detection can reduce the error rate of election prediction result with the average decrease of 0.85%. Moreover, it was also found that the buzzers tend to produce tweets that support the the candidate of their choice more than tweets that vilify their opponent.

Keywords: Twitter, electoral prediction, sentiment analysis, buzzer detection.

BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Hadirnya media sosial seperti Twitter dan Facebook memungkinkan ratusan juta penggunanya untuk saling berbagi informasi ke seluruh dunia dalam waktu yang singkat (Ibrahim, 2015). Saat pengguna Twitter mengikuti akun-akun media, otomatis pengguna akan melihat informasi yang dibagikan. Bahkan akun-akun pribadi pun, yang bukan bagian dari media, ternyata juga sering menyebarkan informasi, baik secara langsung maupun me-retweet dari akun-akun berita. Informasi tersebut ternyata dapat mempengaruhi pandangan orang lain terhadap suatu objek, seperti individu, produk, hingga fenomena yang terjadi. Hal ini merupakan salah satu alasan pentingnya analisis dari sentimen yang diekspresikan oleh netizen di media sosial (Ramteke, 2016).

Pengaruh media sosial sangat besar dalam dunia politik. Sejak tahun 2008, para pasangan calon presiden Amerika Serikat telah menggunakan media sosial seperti Facebook dan Twitter untuk menggalang dukungan. Menurut Cornfield, kesuksesan Barrack Obama merupakan hasil dari strategi politik online yang dilakukan oleh tim kampanyenya. Strategi politik online tidak hanya dilakukan di luar negeri saja. Di Indonesia, pada pemilu presiden RI tahun 2014 lalu, tim kampanye pasangan calon dan juga para pendukungnya dengan gencar menggunakan media sosial dengan mengunggah beragam video, foto atau pun status seputar pilpres melalui Facebook dan Twitter. Dikutip dari Kompas, salah satu calon presiden, Joko Widodo, bahkan sudah menggunakan media sosial sebagai media kampanye ketika mencalonkan diri sebagai gubernur DKI Jakarta pada 2012 lalu.

Sebuah penelitian menarik dilakukan oleh Tumasjan, yang melakukan penelitian untuk mencari tahu apakah pesan-pesan di Twitter (disebut sebagai juga sebagai *tweet*) memiliki korelasi dengan hasil pemilu parlemen di Jerman. Ditemukan bahwa setelah dianalisis, tweet di Twitter ternyata mencerminkan hasil

dari pemilu tersebut, dengan perbedaan hasil yang sangat kecil bila dibandingkan dengan hasil resmi. Hal ini mengindikasikan bahwa kumpulan tweet di Twitter dapat digunakan sebagai sumber data dalam memprediksi pemilu.

Pada tahun 2017, DKI Jakarta mengadakan pemilihan gubernur dengan tiga pasangan calon untuk putaran pertama dan dua pasangan calon untuk putaran kedua. Pada putaran kedua yang diadakan pada tanggal 19 April 2017, terdapat dua pasang calon gubernur dan wakil gubernur, yakni Basuki Tjahaja Purnama (Ahok) - Djarot Saiful Hidayat (Djarot) dan Anies Baswedan (Anies) - Sandiaga Uno (Sandi). Dikutip dari Kompas, kedua pasangan kandidat secara aktif menggunakan Twitter untuk keterlibatan politik dan kampanye.

Kumpulan tweet di Twitter bisa dijadikan alternatif metode survei konvensional untuk memprediksi hasil suatu pemilihan umum. Namun menurut Ibrahim, hal ini adalah tugas yang menantang karena tweet dibatasi hanya sebanyak 140 karakter, jauh lebih pendek dari pada pesan yang berasal dari *platform* media sosial lainnya. Selain itu, berdasarkan penelitian yang dilakukan oleh Pear Analytics diketahui bahwa sekitar 40% dari semua tweet yang ada hanyalah "ocehan tanpa pikiran". Tantangan lainnya adalah bagaimana mendeteksi tweet yang dihasilkan oleh akun *buzzer* dari kumpulan tweet di Twitter yang sangat besar. Ibrahim (2017) mendefinisikan *buzzer* sebagai akun yang hanya memuji salah satu kandidat, sementara menjelek-jelekkan kandidat lainnya. *Buzzer* yang dimaksud dapat berupa bot komputer, pengguna yang dibayar, serta pengguna fanatik dan biasanya ditandai dengan umur akun yang relatif muda. Menurut Ibrahim, sebagian *buzzer* dibayar untuk secara sengaja memobilisasi opini pengguna dengan menggunakan informasi palsu. Dikutip dari Kompas, *buzzer* digunakan sebagai bagian dari *black campaign* dengan mengirimkan tweet-tweet bersentimen negatif berupa fitnah dan *hoax* terhadap suatu kandidat. Jika tweet yang dihasilkan *buzzer* tidak dihapus maka akan menjadi *noise* dan berpotensi mempengaruhi hasil prediksi. Oleh karena itu, diperlukan suatu metode yang mampu mengklasifikasi akun *buzzer* dengan akun pengguna normal.

Untuk mengklasifikasi akun dan sentimen tweet pada Twitter dibutuhkan suatu algoritma klasifikasi. Salah satu algoritma klasifikasi yang cukup sering digunakan adalah Naive Bayes. Jurafsky dan Martin (2016) menyatakan bahwa Naive Bayes dapat melakukan klasifikasi dengan sangat cepat dan membutuhkan kapasitas penyimpanan yang kecil. Naive Bayes juga dianggap tahan terhadap fitur-fitur yang tidak relevan. Salah satu algoritma Bayesian yang sering digunakan dalam klasifikasi sentimen adalah Multinomial Naive Bayes. Menurut Jurafsky, Multinomial Naive Bayes merupakan *baseline* yang baik dan dapat diandalkan dalam melakukan klasifikasi teks. Hand (2001) menyatakan bahwa Gaussian Naive Bayes dapat digunakan dalam melakukan klasifikasi data numerik seperti data informasi akun dengan performa yang cukup baik.

Atas dasar-dasar tersebut diatas, penelitian ini mengusulkan topik analisis sentimen pada data twitter untuk mengetahui pengaruh masing-masing sentimen (positif, negatif, dan netral) dan pengaruh tweet yang dihasilkan oleh *buzzer* terhadap prediksi pilkada DKI Jakarta 2017. Algoritma klasifikasi sentimen yang digunakan adalah Multinomial Naive Bayes, sedangkan algoritma klasifikasi *buzzer* yang digunakan adalah Gaussian Naive Bayes.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijabarkan di atas, diketahui bahwa tweet di Twitter memiliki kecenderungan mencerminkan hasil dari suatu pemilihan umum. Sehingga dalam penelitian ini, ingin dilihat apakah populasi sentimen tweet (positif, negatif, dan netral) di Twitter benar-benar mencerminkan hasil dari pemilihan kepala daerah DKI Jakarta 2017. Serta ingin dilihat juga pengaruh tweet yang dihasilkan oleh *buzzer* terhadap hasil prediksi pilkada DKI Jakarta 2017. Selain itu, ingin dilihat pula seberapa efektif prediksi pilkada berbasis tweet dengan dan tanpa proses *buzzer detection* bila dibandingkan dengan metode survei konvensional.

1.3 Batasan Masalah

Batasan masalah dalam penelitian ini adalah sebagai berikut:

1. Penelitian ini hanya membahas tentang Pemilihan kepala daerah DKI Jakarta 2017.
2. Tweet berasal dari hasil fitur *advanced search* Twitter dengan menggunakan kata kunci nama masing-masing pasangan calon. Kata kunci yang digunakan untuk kandidat nomor urut 2 adalah “ahok”, ”djarot”, ”ahok-djarot”, “basuki tjahaja purnama”, “djarot saiful hidayat”. Sedangkan untuk kandidat nomor urut 3, kata kunci yang digunakan adalah “anies”, ”sandi”, ”anies-sandi”, “anies baswedan”, “sandiaga uno”.
3. Tweet yang digunakan hanya yang berasal dari Provinsi DKI Jakarta.
4. Tweet yang digunakan hanya yang berasal dalam rentang waktu 16 Februari 2017 sampai 16 April 2017.
5. Tweet yang akan diklasifikasikan sebanyak 5281 tweet berbahasa Indonesia, dengan 1584 data latih dan 3697 data tes.
6. Data informasi akun yang akan diklasifikasikan sebanyak 4.887 buah, dengan 1466 data latih dan 3421 data tes.

1.4 Tujuan Penelitian

Berdasarkan rumusan masalah di atas, penelitian ini bertujuan untuk:

1. Mengetahui apakah populasi sentimen (positif, negatif, dan netral) pada data Twitter mencerminkan hasil dari pilkada DKI Jakarta 2017.
2. Mengetahui sentimen (positif, negatif, atau netral) manakah yang merupakan faktor yang paling mempengaruhi hasil prediksi pilkada berbasis data Twitter.
3. Mengetahui seberapa besar pengaruh *buzzer* di Twitter terkait prediksi hasil pilkada dan kecenderungan sentimen yang dihasilkan oleh *buzzer* untuk masing-masing pasangan calon.
4. Mengetahui seberapa efektif prediksi pilkada berbasis tweet dengan dan tanpa proses *buzzer detection* dibandingkan dengan metode survei konvensional seperti jajak pendapat secara *offline* di DKI Jakarta.

1.5 Manfaat Penelitian

Berdasarkan tujuan penelitian di atas, penelitian ini memiliki manfaat:

1. Sebagai alternatif metode survei konvensional untuk memprediksi hasil suatu pemilihan umum dengan biaya jauh yang lebih rendah namun dengan tingkat akurasi yang lebih tinggi.

BAB II

TINJAUAN PUSTAKA

Tumasjan (2010) meneliti apakah Twitter dapat digunakan sebagai forum musyawarah politik dan apakah pesan-pesan di Twitter mencerminkan sentimen politis dalam pemilu federal Jerman tahun 2009. Dengan menggunakan perangkat lunak analisis teks LIWC, penelitian ini melakukan analisis sentimen lebih dari 100.000 tweet yang berisi rujukan ke partai politik atau politisi. Hasilnya menunjukkan bahwa Twitter memang banyak digunakan dalam musyawarah politik. Penelitian ini menemukan bahwa jumlah tweet yang menyebut sebuah partai ternyata mencerminkan hasil pemilu di dunia nyata. Apalagi, dua partai yang sering disebut bersama dalam satu tweet ternyata sejalan dengan ikatan politik dan koalisi di dunia nyata. Analisis sentimen politik tweet tersebut menunjukkan korespondensi yang erat dengan posisi politik partai dan politisi. Hal ini menunjukkan bahwa isi tweet di Twitter mencerminkan lanskap politik di dunia nyata.

Berdasarkan penelitian Tusmajan, Prasetyo (2014) meneliti sebuah metode untuk memprediksi hasil pemilu Presiden RI 2014 dengan melakukan analisis pada data Twitter. Analisis dilakukan dengan membandingkan nilai MAE dari sembilan faktor yang berbeda terhadap hasil KPU. Sembilan faktor tersebut adalah *tweet count*, durasi data, seleksi kata kunci, *user count*, penyaringan data, berat populasi, informasi gender, analisis sentimen, dan jumlah pengguna. Metode ini mampu mencapai MAE terendah, yakni 0,62% dengan menerapkan analisis sentimen pada masing-masing tweet. Sentimen masing-masing tweet diklasifikasikan dengan menghitung nilai polaritas sentimen dengan menggunakan metode Sentiment Lexicon. Perhitungan prediksi hasil pilkada dilakukan dengan membandingkan jumlah tweet positif untuk masing-masing pasangan calon, kemudian dihitung persentasenya. Hasil dari penelitian ini jauh lebih baik bila dibandingkan dengan metode prediksi konvensional yang dilakukan oleh beberapa lembaga survei independen.

Ibrahim et al. (2015) juga meneliti sebuah metode untuk memprediksi hasil pemilu Presiden RI 2014 dengan melakukan analisis sentimen pada data Twitter. Penelitian ini didasarkan pada penelitian Prasetyo (2014) namun menambah proses *automatic buzzer detection*. Klasifikasi sentimen dilakukan dengan menghitung nilai polaritas sentimen dengan menggunakan Indonesian Sentiment Lexicon. Proses *automatic buzzer detection* berfungsi untuk mendeteksi akun-akun *buzzer* dan menghapus tweet yang dihasilkan oleh *buzzer* tersebut. *Buzzer* yang dimaksud adalah bot komputer, pengguna yang dibayar, serta pengguna fanatik dan biasanya ditandai dengan umur akun yang relatif muda. Dengan menerapkan analisis sentimen politik dan memberikan pertimbangan terhadap keberadaan akun *buzzer*, metode ini mampu menghasilkan nilai MAE sebesar 0.61%, lebih baik dibandingkan penelitian Prasetyo dan metode prediksi konvensional yang dilakukan oleh beberapa lembaga survei independen.

Pada tahun 2016, Wicaksono et al. meneliti metode untuk memprediksi hasil pemilu Presiden Amerika Serikat 2016 dengan melakukan analisis sentimen pada media sosial. Media sosial yang digunakan adalah Twitter dengan tweet berbahasa Inggris. Ada tiga metode klasifikasi yang digunakan dalam penelitian ini, Binary Multinomial Naive Bayes, SentiWordNet, dan AFINN-11. Setelah itu, ketiga metode tersebut dibandingkan akurasi. AFINN-111 dan Binary Multinomial Naive Bayes keduanya memberikan skor yang bagus. Namun, Binary Multinomial Naive Bayes dipilih untuk digunakan karena menghasilkan nilai F1-score yang lebih tinggi daripada AFINN-111.

Selain Wicaksono, Ramteke et al. (2016) juga meneliti tentang metode memprediksi hasil pemilu presiden Amerika Serikat 2016 dengan menganalisis sentimen dari data Twitter. Data yang diolah adalah data tweet berbahasa Inggris. Ada dua metode yang dibandingkan, Multinomial Naive Bayes dan Support Vector Machine (SVM). Dari hasil yang didapat, SVM ternyata menghasilkan akurasi sedikit lebih baik dibandingkan Multinomial Naive Bayes dengan perbedaan akurasi hanya dua persen.

Dalam penelitian ini diusulkan sebuah metode untuk mengetahui pengaruh antara masing-masing populasi sentimen (positif, negatif, dan netral) dan pengaruh tweet yang dihasilkan oleh *buzzer* terhadap hasil prediksi pilkada DKI Jakarta 2017. Algoritma klasifikasi sentimen yang digunakan adalah Multinomial Naive Bayes, sedangkan algoritma klasifikasi *buzzer* yang digunakan adalah Gaussian Naive Bayes,

Tabel 2.1 : Rangkuman tinjauan pustaka

No.	Peneliti	Topik	Metode	Perbedaan
1	Tumasjan et al.	Prediksi pemilu presiden di Jerman dengan menganalisis sentimen teks di Twitter	-Analisis sentimen menggunakan <i>software</i> LIWC	-Dataset -Metode analisis sentimen -Proses <i>Buzzer Detection</i>
2	Prasetyo	Prediksi pemilu presiden RI dengan membandingkan sembilan faktor berbeda menggunakan data Twitter	-Membandingkan sembilan faktor berbeda, salah satunya analisis sentimen -Klasifikasi sentimen menggunakan Sentiment Lexicon	-Dataset -Metode klasifikasi sentimen -Proses <i>Buzzer Detection</i>
3	Ibrahim et al.	Prediksi pemilu presiden RI disertai dengan deteksi <i>buzzer</i> menggunakan	-Klasifikasi sentimen menggunakan Indonesian Sentiment Lexicon -Buzzer Detection	-Dataset -Metode klasifikasi <i>buzzer</i>

No.	Peneliti	Topik	Metode	Perbedaan
		analisis sentimen dari data Twitter		-Metode klasifikasi sentimen -Metode <i>pre-processing</i>
4	Wicaksono et al.	Prediksi pemilu presiden AS dengan membandingkan performa tiga algoritma klasifikasi sentimen dari data Twitter	-Membandingkan performa algoritma Binarized Multinomial Naive Bayes, SentiWordNet, dan AFINN-11 untuk klasifikasi sentimen	-Bahasa dari dataset -Proses <i>Buzzer Detection</i>
5	Ramteke et al.	Prediksi pemilu presiden AS dengan membandingkan performa dua algoritma klasifikasi sentimen dari data Twitter	-Membandingkan performa algoritma SVM dan Multinomial Naive Bayes untuk klasifikasi sentimen	-Bahasa dari dataset -Proses <i>Buzzer Detection</i>

BAB III

LANDASAN TEORI

3.1 Data Mining

Data mining adalah bidang ilmu yang berfokus pada proses penemuan pola dari data set yang besar dengan menggabungkan metode-metode kecerdasan buatan, machine learning, statistika, dan sistem basis data. Tujuan secara keseluruhan dari data mining adalah proses ekstraksi informasi dari suatu data kemudian direpresentasikan ke dalam suatu struktur yang dapat dianalisa, dimengerti, dan dapat digunakan dalam membuat keputusan (Chakrabarti et al., 2006).

Data mining dapat digunakan untuk melakukan berbagai tugas seperti, summarization, clustering (unsupervised learning), classification (supervised learning), regression, dan association (Gheware et al., 2014).

Menurut Gheware (2014), alasan utama mengapa data mining sangat menarik perhatian industri informasi dalam beberapa tahun belakangan ini adalah karena tersedianya data dalam jumlah yang besar dan semakin besarnya kebutuhan untuk mengubah data tersebut menjadi informasi dan pengetahuan yang berguna karena sesuai fokus bidang ilmu ini yaitu melakukan kegiatan mengekstraksi pengetahuan dari data yang berukuran besar. Informasi inilah yang nantinya akan digunakan dalam bidang ilmu terkait dalam membuat berbagai keputusan.

3.2 Information Retrieval

Information Retrieval merupakan ilmu yang mempelajari metode dan prosedur untuk menemukan kembali informasi yang tersimpan dari berbagai sumber yang relevan atau koleksi sumber informasi yang dicari atau dibutuhkan. Proses-proses dalam IR dapat berupa pembuatan index (indexing), panggilan (searching), dan pemanggilan data kembali (recalling) (Manning et al., 2008).

Dalam pencarian data, beberapa jenis data yang dapat ditemukan dapat berupa teks, tabel, gambar, video, dan audio. Tujuan dari Information Retrieval adalah untuk memenuhi informasi pengguna dengan cara mendapatkan dokumen

yang relevan atau mengurangi dokumen pencarian yang tidak relevan (Gheware et al., 2014).

Menurut Manning (2008), secara konsep sederhana IR merupakan proses mencari, dan kemudian mendapatkan apa yang dicari itu. Prosesnya berupa proses perjalanan informasi yang diminta, menjadi informasi yang diberikan.

3.3 Analisis Sentimen

Analisis sentimen (dikenal juga sebagai opinion mining) mengacu pada penggunaan natural language processing, text analysis, computational linguistics, dan biometrics dalam mengidentifikasi, mengekstrak, mengukur, dan mempelajari informasi yang bersifat subyektif (Pang, 2008). Analisis sentimen banyak diterapkan bidang-bidang yang berhubungan dengan kepuasan pelanggan seperti review dan tanggapan survei, serta media sosial. Ekstraksi sentimen konsumen secara otomatis sangat penting dalam proses pemasaran produk serta mengukur sentimen publik penting bagi politik dan prediksi pasar (Deshwal, 2016).

Menurut Pang (2008), secara umum analisis sentimen bertujuan untuk mengetahui sikap pembicara, penulis, atau subjek lain sehubungan dengan polaritas atau reaksi emosional terhadap berbagai peristiwa yang terjadi apakah bersentimen positif, negatif, ataupun netral. Sikap yang dimaksud dapat berupa penilaian ataupun evaluasi.

3.4 Naive Bayes

Naive Bayes classifier merupakan suatu metode pengklasifikasi probabilitas sederhana yang mengaplikasikan Teorema Bayes dengan asumsi ketidaktergantungan (independen) yang tinggi (Jurafsky, 2016).

Naive Bayes classifier lebih unggul dalam hal konsumsi CPU dan memori seperti yang ditunjukkan oleh Huang, J. (2003), dan dalam beberapa kasus kinerjanya sangat dekat dengan teknik yang lebih rumit dan lebih lambat.

Seperti yang dinyatakan sebelumnya, Naive Bayes classifier mengasumsikan fitur yang digunakan dalam klasifikasi bersifat independen. Terlepas dari kenyataan

bahwa asumsi ini biasanya salah, analisis dari masalah klasifikasi Bayesian menunjukkan bahwa ada beberapa alasan teoritis untuk keberhasilan Naive Bayes classifier yang tampaknya tidak masuk akal seperti yang ditunjukkan oleh Zhang (2004). Hal ini dapat dibuktikan bahwa meskipun perkiraan probabilitas Naive Bayes berkualitas rendah, keputusan klasifikasinya cukup baik (Manning et al., 2008). Jadi, terlepas dari kenyataan bahwa Naive Bayes biasanya menaksir terlalu tinggi probabilitas kelas yang dipilih, mengingat bahwa Naive Bayes digunakan hanya untuk membuat keputusan dan tidak memprediksi probabilitas aktual secara akurat, pengambilan keputusan adalah benar sehingga modelnya akurat.

3.4.1 Multinomial Naive Bayes

Menurut Jutafsky (2016), Naive Bayes adalah suatu metode pengklasifikasian berdasarkan probabilitas, yang berarti untuk dokumen d , untuk semua kelas $c \in C$ classifier akan mengembalikan kelas \hat{c} yang mana memiliki posterior maksimum. Pada persamaan (3.1) digunakan notasi $\hat{\cdot}$ untuk mengartikan estimasi dari kelas yang benar.

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c|d) \quad (3.1)$$

Ide dari Inferensi Bayesian telah dikenal sejak karya Bayes (1763), dan pertama kali diaplikasikan untuk klasifikasi teks oleh Monsteller dan Wallace (1964). Intuisi dari klasifikasi bayesian adalah dengan menggunakan aturan Bayes untuk mengubah persamaan (3.1) ke probabilitas lainnya. Aturan Bayes disajikan dalam persamaan (3.2); memberikan suatu cara untuk merinci probabilitas kondisional $P(x|y)$ ke tiga probabilitas lainnya:

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)} \quad (3.2)$$

Lalu persamaan (3.2) disubstitusikan ke persamaan (3.1) untuk mendapatkan persamaan (3.3):

$$\hat{c} = \operatorname{argmax}_{c \in C} P(c|d) = \operatorname{argmax}_{c \in C} \frac{P(d|c)P(c)}{P(d)} \quad (3.3)$$

Persamaan (3.3) lalu disederhanakan dengan membuang penyebut $P(d)$. Hal ini bisa dilakukan karena $\frac{P(d|c)P(c)}{P(d)}$ akan dihitung untuk setiap kelas yang mungkin. Tetapi $P(d)$ tidak berubah untuk setiap kelas. Ingin dicari kelas yang paling mungkin untuk dokumen d yang sama, yang mana harus memiliki probabilitas $P(d)$ yang sama. Sehingga, dapat dipilih kelas yang memaksimalkan rumus yang lebih sederhana di bawah ini:

$$\hat{c} = \operatorname{argmax}_{c \in C} P(c|d) = \operatorname{argmax}_{c \in C} P(d|c)P(c) \quad (3.4)$$

Telah dihitung kelas \hat{c} yang paling mungkin jika diberikan dokumen d dengan memilih kelas yang mempunyai hasil kali tertinggi dari dua probabilitas: probabilitas *prior* dari kelas $P(c)$ dan *likelihood* dari dokumen $P(d|c)$:

$$\hat{c} = \operatorname{argmax}_{c \in C} \overbrace{P(d|c)}^{\text{likelihood}} \overbrace{P(c)}^{\text{prior}} \quad (3.5)$$

Tanpa menghilangkan generalisasi, dokumen d dapat direpresentasikan sebagai suatu himpunan fitur f_1, f_2, \dots, f_n :

$$\hat{c} = \operatorname{argmax}_{c \in C} \overbrace{P(f_1, f_2, \dots, f_n|c)}^{\text{likelihood}} \overbrace{P(c)}^{\text{prior}} \quad (3.6)$$

Sayangnya, persamaan (3.6) masih terlalu rumit untuk dihitung secara langsung. Tanpa menyederhanakan asumsi, memperkirakan probabilitas dari setiap kombinasi fitur yang mungkin (contohnya, setiap himpunan kata dan posisi yang mungkin) akan membutuhkan jumlah parameter dan jumlah data latih yang besar. Oleh karena itu, Naive Bayes classifier membuat dua asumsi yang menyederhanakan.

Yang pertama adalah asumsi *bag of words*. Diasumsikan posisi tidak berpengaruh dan suatu kata memiliki efek yang sama pada klasifikasi apakah kata tersebut muncul pada posisi pertama, ke-dua puluh, ataupun terakhir dari dokumen. Sehingga, bisa diasumsikan fitur f_1, f_2, \dots, f_n hanya menggunakan identitas kata dan bukan posisinya.

Yang kedua pada umumnya disebut sebagai asumsi Naive Bayes, hal ini merupakan asumsi independen kondisional probabilitas $P(f_n|c)$ yang mana probabilitas $P(f_i|c)$ adalah independen jika diberikan kelas c dan oleh karena itu, dapat dikalikan secara *naive* seperti berikut:

$$P(f_1, f_2, \dots, f_n|c) = P(f_1|c) \cdot P(f_2|c) \cdot \dots \cdot P(f_n|c) \quad (3.7)$$

Persamaan akhir untuk kelas yang dipilih oleh naive bayes classifier adalah sebagai berikut:

$$c_{NB} = \operatorname{argmax}_{c \in C} P(c) \prod_{f \in F} P(f|c) \quad (3.8)$$

Untuk menerapkan Naive Bayes classifier ke teks, posisi dari kata harus dipertimbangkan dengan menjalankan index melewati setiap posisi kata dalam dokumen:

posisi \leftarrow semua posisi kata pada dokumen tes

$$c_{NB} = \operatorname{argmax}_{c \in C} P(c) \prod_{i \in \text{positions}} P(w_i|c) \quad (3.9)$$

Perhitungan Naive Bayes dilakukan dalam *log space* untuk menghindari *underflow* dan meningkatkan kecepatan. Sehingga persamaan (3.9) dapat dinyatakan secara umum sebagai:

$$c_{NB} = \operatorname{argmax}_{c \in C} \log P(c) + \sum_{i \in \text{positions}} \log P(w_i|c) \quad (3.10)$$

Dengan memepertimbangkan fitur di *log space*, persamaan (3.10) menghitung kelas yang diprediksi sebagai sebuah fungsi linear dari fitur input. *Classifier* yang menggunakan kombinasi linear dari input untuk melakukan klasifikasi disebut sebagai *linear classifier*.

Untuk menghitung probabilitas $P(c)$ dan $P(f_i|c)$, pertama-tama estimasi *maximum likelihood* harus dipertimbangkan. Oleh karena itu, akan digunakan frekuensi dari data. Untuk probabilitas *prior* dokumen $P(c)$, perlu dihitung berapa persentase dokumen dalam data latih di setiap kelas c . Misalkan N_c adalah jumlah dokumen dalam data latih dengan kelas c dan N_{doc} adalah jumlah total dokumen. Sehingga:

$$\hat{P}(c) = \frac{N_c}{N} \quad (3.11)$$

Untuk menghitung probabilitas $P(f_i|c)$, diasumsikan sebuah fitur sebagai suatu kemunculan kata pada *bag of words* dokumen. Untuk mendapatkan $P(w_i|c)$, fraksi kata w_i yang muncul dihitung di antara semua kata dalam semua dokumen topik c . Pertama-tama, semua dokumen dengan kategori c digabungkan menjadi satu teks "kategori c " besar. Kemudian frekuensi w_i digunakan dalam dokumen yang tergabung ini untuk memberikan estimasi *maximum likelihood* dari probabilitas:

$$\hat{P}(w|c) = \frac{\text{count}(w, c) + 1}{\text{count}(c) + |V|} \quad (3.12)$$

Disini kosakata V terdiri dari penyatuan semua jenis kata di semua kelas, tidak hanya kata-kata dalam satu kelas c .

3.4.2 Gaussian Naive Bayes

Salah satu cara khas untuk menangani atribut kontinu dalam klasifikasi Naive Bayes adalah dengan menggunakan distribusi Gaussian untuk mewakili kemungkinan fitur yang disyaratkan pada setiap kelasnya (Mitchell, 1997). Sehingga setiap atribut didefinisikan oleh fungsi kepadatan probabilitas Gaussian sebagai:

$$X_i \sim N(\mu, \sigma^2) \quad (3.13)$$

Fungsi kepadatan probabilitas Gaussian memiliki bentuk bel dan didefinisikan oleh persamaan berikut:

$$N(\mu, \sigma^2)(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (3.14)$$

di mana μ adalah rata-rata (mean) dan σ^2 adalah variansi. Rumus mean ditunjukkan oleh persamaan (3.15) sedangkan rumus variansi ditunjukkan oleh persamaan (3.16).

$$\mu = \frac{1}{n} \sum_{i=1}^n x_i \quad (3.15)$$

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2 \quad (3.16)$$

di mana n adalah jumlah data dan x_i adalah data ke- i .

3.5 Buzzer Detection

Menurut Ibrahim (2015), buzzer detection berfungsi untuk mendeteksi *buzzer* dan menghapus tweet yang dihasilkan oleh *buzzer* tersebut. *Buzzer* yang dimaksud adalah bot komputer, pengguna yang dibayar, serta pengguna fanatik dan biasanya ditandai dengan umur akun yang relatif muda. Jika tweet yang dihasilkan *buzzer* tidak dihapus maka akan menjadi *noise* dan berpotensi mempengaruhi hasil klasifikasi sentimen. Teknik deteksi *buzzer* merupakan masalah klasifikasi dan dapat diselesaikan dengan *supervised learning*. Dengan memperhatikan fitur-fitur yang telah ditetapkan, sebagian akun kemudian akan dilabeli sebagai buzzer dan bukan buzzer. Data yang telah dilabeli tersebut akan dijadikan data latih untuk kemudian digunakan dalam membangun model klasifikasi *buzzer*.

3.6 Pengujian

Pengujian dilakukan untuk menghitung performa dari sistem atau dalam penelitian ini performa klasifikasi sentimen. Untuk menghitung performa

klasifikasi, data latih digunakan untuk melakukan validasi. Validasi dapat dilakukan dengan menggunakan metode yang dinamakan *k-fold cross validation*.

3.6.1 K-fold Cross Validation

Cross Validation merupakan salah satu metode pengujian dalam *data mining* dan *machine learning*. Dalam melakukan *cross validation*, dataset dibagi menjadi beberapa bagian atau dalam hal ini disebut *fold* secara acak. Satu bagian dari hasil pembagian tersebut digunakan sebagai data pengujian dan sisanya digunakan sebagai data pelatihan. Pengujian dilakukan sejumlah bagian yang ada dengan bergantian menggunakan bagian yang berbeda sebagai data pengujian. Hasil pengujian yang dihitung adalah hasil keseluruhan yaitu merata-rata hasil dari keseluruhan pengujian yang dilakukan (Witten et al., 2011).

Cross validation yang dilakukan dengan pembagian sebanyak k disebut sebagai *k-fold cross validation*. Jumlah bagian yang umum digunakan untuk pengujian adalah sebanyak sepuluh. Pengujian banyak dilakukan dengan menggunakan *10-fold cross validation* dikarenakan dari hasil percobaan menunjukkan *10-fold cross validation* memberikan perkiraan kesalahan dari klasifikasi yang terbaik.

Pengujian *k-fold cross validation* yang dilakukan sekali sering tidak memberikan perkiraan kesalahan yang bisa diandalkan. Hal ini disebabkan pengujian yang dilakukan beberapa kali dengan parameter yang sama dapat memberikan hasil yang berbeda yang disebabkan oleh unsur *random* yang ada pada saat pembagian data. Oleh karena itu untuk mendapatkan hasil yang dapat diandalkan pengujian harus dilakukan berkali-kali dengan hasilnya merupakan rata-rata dari semua hasil pengujian.

3.6.2 Perhitungan Performa

Model klasifikasi adalah pemetaan dari suatu input data menjadi suatu output yang merupakan prediksi kelas. Klasifikasi yang hanya menghasilkan dua kelas sebagai outputnya disebut klasifikasi biner. Kedua kelas tersebut sering kali merupakan kelas positif dan kelas negatif.

Terdapat empat kemungkinan yang terjadi dari proses pengklasifikasian biner seperti yang dijabarkan oleh Fawcett (2006):

- True Positive (TP) : Prediksi menghasilkan *true*, dan terjadiannya *true*. Sebagai contoh jika model memprediksi seseorang akan **membeli** produk tertentu dan pada kenyataannya mereka **membeli**.
- False Positive (FP) : Prediksi menghasilkan *true*, dan terjadiannya *false*. Sebagai contoh jika model memprediksi seseorang akan **membeli** produk tertentu dan pada kenyataannya mereka **tidak** membeli.
- True Negative (TN) : Prediksi menghasilkan *false*, dan terjadiannya *false*. Sebagai contoh jika model memprediksi seseorang **tidak** akan membeli produk tertentu dan pada kenyataannya mereka **tidak** membeli.
- False Negative (FN) : Prediksi menghasilkan *false*, dan terjadiannya *true*. Sebagai contoh jika model memprediksi seseorang **tidak** akan membeli produk tertentu dan pada kenyataannya mereka **membeli**.

Hasil klasifikasi biner pada suatu data set dapat direpresentasikan pada suatu matriks 2x2 yang disebut *confusion matrix* seperti yang ditunjukkan pada Gambar 3.1.

		Kelas Data	
		Positive	Negative
Prediksi kelas	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

Gambar 3.1: Ilustrasi *Confusion Matrix*

Terdapat beberapa rumus umum yang dapat digunakan untuk menghitung performa klasifikasi yang dapat dijabarkan sebagai berikut menurut Fawcett (2006):

- *Accuracy* (Akurasi) : perbandingan kasus yang diidentifikasi benar dengan jumlah semua kasus. Persamaan 3.17 menunjukkan rumus untuk *accuracy*.
- *Precision/PPV* : proporsi kasus yang diidentifikasi dengan benar sebagai milik kelas 'a' di antara semua kasus yang diklasifikasikan oleh pengklasifikasi bahwa mereka termasuk dalam kelas 'a'. Persamaan 3.18 menunjukkan rumus untuk menghitung *precision*.
- *Recall/Sensitivity* : proporsi kasus yang diidentifikasi dengan benar sebagai milik kelas 'a' di antara semua kasus yang benar-benar termasuk dalam kelas 'a'. Persamaan 3.19 menunjukkan rumus untuk menghitung *recall*.
- *F1 score/F-measure* : bobot rata-rata antara *precision* dan *recall*. Persamaan 3.20 menunjukkan rumus *F1 score*.

$$Akurasi = \frac{TP + TN}{TP + TN + FP + FN} \quad (3.17)$$

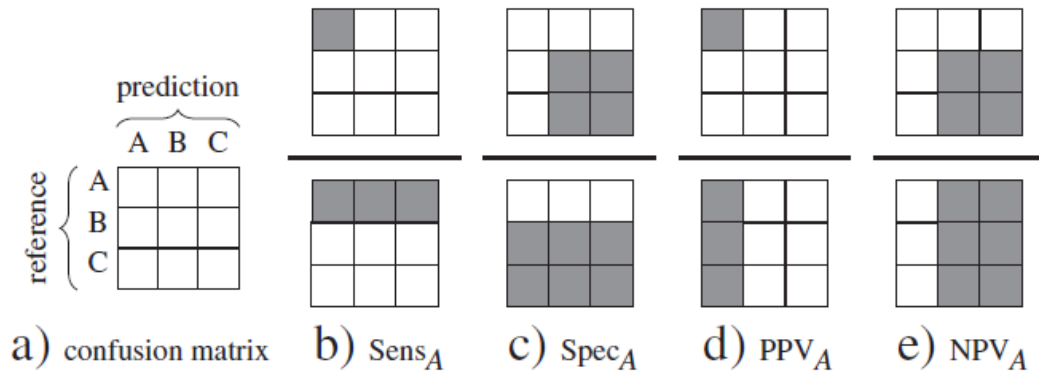
$$Precision = \frac{TP}{TP + FP} \quad (3.18)$$

$$Recall = \frac{TP}{TP + FN} \quad (3.19)$$

$$F1\ Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (3.20)$$

Nilai TP, TN, FP, dan FN yang ada pada ketiga rumus diatas masing-masing merupakan nilai *true positive*, *true negative*, *false positive*, dan *false negative*. Keempat nilai akurasi, *precision*, *recall*, dan *F1 score* ditampilkan dalam bentuk persentase.

Untuk kasus di mana proses klasifikasi mempunyai tiga kelas, bentuk *confusion matrix*-nya ditunjukkan pada Gambar 3.2.



Gambar 3.2: Ilustrasi *Confusion Matrix* 3 kelas beserta rumus perhitungan performa (Beleites et al., 2013)

3.6.3 Mean Absolute Error (MAE)

Mean absolute error (MAE) berfungsi untuk merepresentasikan rata – rata kesalahan (error) absolut antara hasil prediksi dengan nilai sebenarnya. Untuk menghitung performa dari prediksi pilkada, digunakan Mean Absolute Error (MAE), yang didefinisikan sebagai

$$MAE = \frac{\sum_{i=1}^N |e_i|}{N} \quad (3.21)$$

Pada persamaan (3.21), N adalah jumlah kandidat dan e_i adalah perbedaan antara hasil prediksi dan hasil sebenarnya berkaitan dengan kandidat ke- i . Faktanya, MAE juga sudah banyak digunakan dalam mengevaluasi prakiraan politik sebelumnya (Huber, 2005).

BAB IV

ANALISIS DAN RANCANGAN

4.1 Analisis Sistem

Pada bagian ini akan dijelaskan deskripsi sistem secara umum dan analisis mengenai data yang akan digunakan. Deskripsi sistem merupakan gambaran rangkaian proses yang dilakukan pada penelitian ini. Analisis data akan memberikan informasi mengenai data yang akan diproses.

4.1.1 Deskripsi Sistem

Tujuan penelitian ini adalah untuk mengetahui apakah populasi sentimen tweet (positif, negatif, dan netral) di Twitter mampu mencerminkan hasil dari pilkada DKI Jakarta 2017 dan seberapa besar pengaruh tweet yang dihasilkan oleh *buzzer* terhadap hasil prediksi pilkada tersebut. Untuk itu, dibutuhkan sistem yang mampu melakukan klasifikasi sentimen dan klasifikasi *buzzer*. Selain itu, sistem juga mampu melakukan prediksi pilkada dengan membandingkan jumlah tweet yang mendukung masing-masing pasangan calon serta menghitung tingkat kesalahan dari hasil prediksi pilkada tersebut.

Proses klasifikasi sentimen dilakukan dengan menggunakan data tweet di Twitter terkait Pilkada DKI Jakarta 2017. Tweet diambil dari situs Twitter melalui proses *scraping tag-tag* html pada halaman pencarian. Tweet yang sudah diambil kemudian dilabeli secara manual dengan berlandaskan teori dari penelitian Ibrahim (2015). Data yang sudah dilabeli kemudian dibaca dan dimasukkan kedalam proses *pre-processing* data. *Pre-processing* data dimaksudkan untuk mengurangi *noise* pada data dengan mengganti dan menghilangkan fitur-fitur yang tidak diperlukan. Setelah itu, algoritma Multinomial Naive Bayes digunakan untuk melakukan proses klasifikasi sentimen.

Proses *buzzer detection* dilakukan dengan menggunakan data informasi akun pengguna Twitter yang terdapat dalam *dataset* tweet klasifikasi sentimen. Informasi akun diambil menggunakan Twitter API berdasarkan *username* dari

seluruh data tweet yang digunakan dalam proses klasifikasi sentimen. Informasi akun yang sudah diambil kemudian dilabeli secara manual dengan berlandaskan teori dari penelitian Ibrahim (2015). *Buzzer detection* yang terdiri atas dua proses, yakni klasifikasi *buzzer* dan penghapusan tweet *buzzer*. Proses klasifikasi *buzzer* menggunakan algoritma Gaussian Naive Bayes.

Proses pelatihan dan pengujian model kemudian dilakukan untuk melihat seberapa bagus model yang dihasilkan pada dua kategori berikut:

1. Kategori klasifikasi sentimen yang membagi data menjadi kelas positif, netral, dan negatif. Total tiga kelas.
2. Kategori klasifikasi *buzzer* yang membagi data menjadi kelas normal dan *buzzer*. Total dua kelas.

Pengujian model dilakukan dengan menggunakan metode *10-fold cross validation*. Parameter keberhasilan model klasifikasi terletak pada akurasi yang dihasilkan dari pengujian tersebut.

Setelah semua tweet terklasifikasi positif, negatif, atau netral, jumlah tweet bersentimen tertentu terhadap masing-masing pasangan calon digunakan dalam menghitung prediksi hasil pilkada. Perhitungan prediksi hasil pilkada dilakukan dengan membandingkan jumlah tweet yang mendukung masing-masing pasangan calon, misalnya jumlah tweet bersentimen positif pasangan Ahok-Djarot dibandingkan dengan jumlah tweet bersentimen positif pasangan Anies-Sandi, kemudian dihitung persentasenya. Persentase tersebut kemudian dihitung tingkat kesalahannya menggunakan Mean Absolute Error dengan membandingkan persentase tersebut dengan hasil resmi dari KPUD DKI Jakarta. Semakin kecil nilai Mean Absolute Error, maka persentase tersebut semakin mendekati hasil resmi. Parameter keberhasilan metode prediksi hasil pilkada berdasarkan data Twitter terletak pada nilai Mean Absolute Error yang dihasilkan.

4.1.2 Deskripsi Data

Data diambil dari website Twitter dengan melakukan proses *scraping* menggunakan *library* Selenium dan proses menggunakan Twitter API. Data yang

tersedia terbagi menjadi dua, yaitu data informasi akun (*username*, umur akun, jumlah *following*, jumlah *follower*) serta data tweet (*username* dan tweet). Data informasi akun didapat melalui Twitter API sedangkan data tweet didapat melalui proses *scraping*. Data yang digunakan terdiri dari 5281 data tweet dan 4887 data akun. Setiap data latih memiliki satu label seperti yang ditunjukkan pada Tabel 4.2 dan Tabel 4.4, kemudian label tersebut dijabarkan menjadi tiga label untuk kategori klasifikasi sentimen dan dua label untuk kategori klasifikasi *buzzer*. Kategori klasifikasi sentimen menandakan apakah dalam tweet terdapat sentimen positif, negatif, atau netral, sedangkan kategori klasifikasi *buzzer* menandakan apakah suatu akun termasuk *buzzer* atau hanya pengguna biasa.

1. Data Latih

Data latih yang digunakan dalam proses klasifikasi sentimen adalah data tweet yang diperoleh sesuai dengan kata kunci yang tertera pada Tabel 4.10 melalui proses *scraping*. Data latih berjumlah 30% dari seluruh data tweet hasil *scraping* yakni sebanyak 1585 tweet, di mana 713 tweet merupakan data latih untuk pasangan calon nomor urut dua sedangkan 872 tweet merupakan data latih untuk pasangan calon nomor urut tiga. Untuk pasangan calon nomor urut dua, Ahok-Djarot, dari total 713 tweet, 359 tweet dilabeli sebagai positif, 148 tweet dilabeli sebagai netral, dan 206 tweet dilabeli sebagai negatif. Untuk pasangan calon nomor urut tiga, Anies-Sandi, dari total 872 tweet, 436 tweet dilabeli sebagai positif, 298 tweet dilabeli sebagai netral, dan 138 tweet dilabeli sebagai negatif. Data tweet tersebut dilabeli secara manual sesuai dengan sentimen yang terkandung pada masing-masing tweet. Tabel 4.1 menunjukkan persebaran data pelatihan klasifikasi sentimen yang akan digunakan, sedangkan Tabel 4.2 menunjukkan sampel dari isi data pelatihan klasifikasi sentimen.

Tabel 4.1 : Spesifikasi data latih klasifikasi sentimen

Sentimen	Ahok-Djarot	Anies-Sandi	Jumlah
Positif (pos)	359	436	795
Netral (net)	148	298	446

Sentimen	Ahok-Djarot	Anies-Sandi	Jumlah
Negatif (neg)	206	138	344
Total jumlah data latih			1585

Tabel 4.2 : Sampel data latih klasifikasi sentimen

Username	Tweet	Label
siskaamelia_83	Ada pak anies @IndosiarID alhamdulillah ada gubernur kita harus menang pak anies pasti menang @DAcademyID #dacademy4	pos
Delidolfina	Survei Pasca Debat: Anies-Sandi Unggul Tipis	net
arisPurr	Debat Ahok - Anies Ahok sibuk nemaparkan Programnya sedangkan Anies selalu mancing2 Ahok biar marah lalu lepas kontrol.	neg

Data latih yang digunakan dalam proses klasifikasi *buzzer* adalah data informasi akun pengguna yang diperoleh dengan menggunakan Twitter API. Data latih berjumlah 30% dari seluruh data informasi akun pengguna yakni sebanyak 1466 akun, dengan 337 akun dilabeli sebagai *buzzer* dan 1129 akun dilabeli sebagai normal. Setiap akun pengguna memuat informasi *username*, umur akun, jumlah mengikuti (*following*), jumlah pengikut (*follower*), serta label yang dilabeli secara manual. Tabel 4.3 menunjukkan persebaran data pelatihan klasifikasi *buzzer* yang akan digunakan, sedangkan Tabel 4.4 menunjukkan sampel dari isi data pelatihan klasifikasi *buzzer*.

Tabel 4.3 : Spesifikasi data latih klasifikasi *buzzer*

Jenis user	Jumlah
buzzer	337
normal	1129
Total jumlah data latih	1466

Tabel 4.4 : Sampel data latih klasifikasi *buzzer*

Username	Umur akun	Jumlah mengikuti	Jumlah pengikut	Label
Rica_Susantii	90	245	6838	normal
waherbot	16	109	15	buzzer
ekowBoy	82	1348	23721	normal

2. Data Tes

Data tes yang digunakan dalam proses klasifikasi sentimen adalah data tweet yang diperoleh sesuai dengan kata kunci yang tertera pada Tabel 4.10 melalui proses *scraping*. Data tes berjumlah 70% dari seluruh data tweet hasil *scraping* yakni sebanyak 3196 tweet, di mana 1663 tweet merupakan data tes untuk pasangan calon nomor urut dua sedangkan 2033 tweet merupakan data latih untuk pasangan calon nomor urut tiga. Proses pelabelan tidak dilakukan pada data tweet ini. Tabel 4.5 menunjukkan sampel dari isi data tes klasifikasi sentimen.

Tabel 4.5 : Sampel data tes klasifikasi sentimen

Username	Tweet
wartapolitik	Keren! Anies Baswedan Tampil Gemilang di "Kandang Lawan" Mata Najwa MetroTV http://www.muslimbersatu.net/2017/03/keren-anies-baswedan-tampil-gemilang-di.html?m=1 ...
Rica_Susantii	Yuksss .. Malem ini Nonton Pak Ahok & Pak Anies LIVE di SCTV.. #PromoKaliKali... (at Sales Dept. SCTV) [pic] —
daralisebony	Anies Sandi sibuk janji janji Basuki Djarot dah nyata nyata hasil krjanya masyarakat jkt lebih pintar & dewasa lah dlm memilih pemimpinnya.

Data tes yang digunakan dalam proses klasifikasi *buzzer* adalah data informasi akun pengguna yang diperoleh dengan menggunakan Twitter API. Data tes berjumlah 70% dari seluruh data informasi akun pengguna yakni sebanyak 3421 akun. Setiap akun pengguna memuat informasi *username*, umur akun, jumlah mengikuti (*following*), jumlah pengikut (*follower*), serta tidak memiliki label. Tabel 4.6 menunjukkan sampel dari isi data tes klasifikasi *buzzer*.

Tabel 4.6 : Sampel data tes klasifikasi *buzzer*

Username	Umur akun	Jumlah mengikuti	Jumlah pengikut
imanlagi	93	458	31187
rajasundawiwaha	13	105	29712
yusejahtera	94	1657	17109

4.2 Perancangan Sistem

Bagian ini membahas rancangan sistem yang akan dikembangkan. Terdapat berbagai tahapan yang dilakukan dalam penelitian ini. Bagian pengambilan data merupakan bagian untuk mengumpulkan data yang digunakan sekaligus melakukan pelabelan. Data tweet yang sudah dikumpulkan kemudian dibagi menjadi dua bagian, yaitu data latih dan data tes dengan persentase 30% untuk data latih dan 70% untuk data tes. Data yang akan dilabeli adalah data latih. Data latih akan digunakan untuk membangun model klasifikasi sedangkan data validasi digunakan untuk menguji performa model. Model yang sudah dibangun kemudian digunakan untuk mengklasifikasikan label pada data tes yang tidak dilabeli sebelumnya. Proses pelabelan dilakukan dengan melihat satu-persatu tweet dengan berlandaskan pada penelitian Ibrahim, et al. (2015), contoh label yang diberikan dapat dilihat pada Tabel 4.9 yang kemudian direpresentasikan menjadi tiga label. Tiga label tersebut adalah positif, netral, dan negatif. Sedangkan untuk proses *buzzer detection*, data akun yang sudah didapat kemudian dibagi menjadi dua bagian, yaitu data latih dan data tes dengan persentase 30% untuk data latih dan 70% untuk data tes. Data

latih kemudian dilabeli apakah akun tersebut termasuk ke dalam kelas *buzzer* atau *normal*. Proses pelabelan dilakukan secara manual.

Proses *pre-processing* data ditujukan untuk mengubah data mentah menjadi data yang berkualitas dan nantinya akan digunakan dalam membangun model klasifikasi sentimen. Ada beberapa langkah yang dilakukan dalam proses *pre-processing*. Yang pertama dengan melakukan *case folding* atau proses penyamaan case dalam sebuah tweet ke dalam suatu bentuk standar. Dalam hal ini, bentuk standarnya adalah huruf kecil. Kemudian dilakukan *stopwords removal* yaitu proses menghapus *stopwords* atau kata-kata yang sering muncul dalam Bahasa Indonesia dan dianggap tidak penting (contohnya: dan, atau, dari). Selanjutnya, proses *brackets removal* dilakukan untuk menghapus tanda kurung beserta isinya dari teks tweet. Proses *mentions removal* dilakukan untuk menghapus *mention* yang ditandai dengan karakter '@' di awal *username* pada teks tweet.

Dataset yang sudah dikumpulkan kemudian akan dilakukan *buzzer detection*. *Buzzer detection* berfungsi untuk mendeteksi *buzzer* dan menghapus tweet yang dihasilkan oleh *buzzer* tersebut. Teknik deteksi *buzzer* merupakan masalah klasifikasi dan dapat diselesaikan dengan *supervised learning*. Data latih yang digunakan dalam membangun model klasifikasi *buzzer* memiliki fitur-fitur yang didasarkan dari penelitian Ibrahim (2015). Daftar fitur yang digunakan ditunjukkan pada Tabel 4.7. Akan tetapi, penelitian ini hanya menggunakan 3 dari 5 fitur yang digunakan dalam penelitian Ibrahim. Fitur jumlah URL dan jumlah *retweet* dari 100 tweet terakhir tidak dimasukkan karena keterbatasan fungsi API. Algoritma klasifikasi yang digunakan adalah Gaussian Naive Bayes. Akurasi dari model klasifikasi dapat dihitung dengan menggunakan *10-fold cross validation* dan *confusion matrix*. Data yang belum dilabeli akan dijadikan data tes untuk diklasifikasi apakah termasuk ke kelas *buzzer* atau *normal*. Setelah semua akun sudah terklasifikasi, semua tweet yang dihasilkan oleh akun yang terklasifikasi sebagai *buzzer* akan dihapus.

Tabel 4.7 : Fitur dari buzzer (Ibrahim, 2015)

No.	Fitur
1	Umur akun (dari tanggal pembuatan)
2	Jumlah URL dari 100 tweet terakhir
3	Jumlah retweet dari 100 tweet terakhir
4	Jumlah pengikut (<i>follower</i>)
5	Jumlah mengikuti (<i>following</i>)

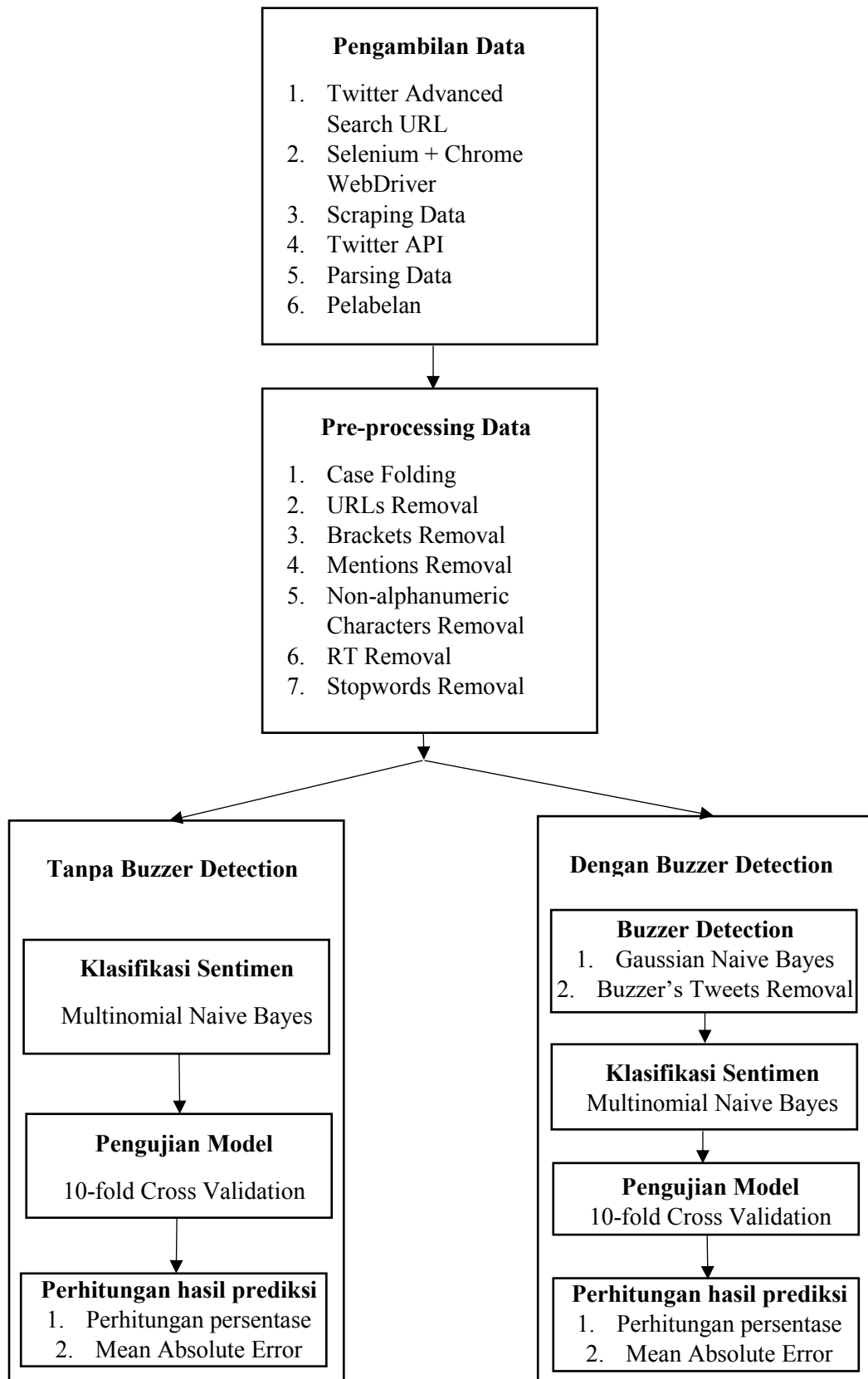
Data tweet yang sudah dilabeli kemudian akan dijadikan data latih untuk kemudian digunakan dalam membangun model klasifikasi sentimen tweet. Algoritma klasifikasi yang digunakan adalah Multinomial Naive Bayes. Akurasi dari model klasifikasi dapat dihitung dengan menggunakan *10-fold cross validation* dan *confusion matrix*. Sedangkan, data yang belum dilabeli akan dijadikan data tes untuk diklasifikasi sentimennya, apakah positif, negatif, atau netral.

Setelah semua tweet terklasifikasi positif, negatif, atau netral, jumlah tweet bersentimen tertentu terhadap masing-masing pasangan calon digunakan dalam menghitung prediksi hasil pilkada. Ibrahim (2015) dalam penelitiannya hanya menggunakan tweet bersentimen positif saja. Hal ini didasari oleh penelitian Prasetyo (2014) yang juga menggunakan tweet bersentimen positif setelah menyimpulkan bahwa penggunaan sentimen positif saja lebih baik bila dibandingkan dengan penggunaan sentimen negatif.

Perhitungan prediksi hasil pilkada dilakukan dengan membandingkan jumlah tweet yang mendukung masing-masing pasangan calon, misalnya jumlah tweet bersentimen positif pasangan Ahok-Djarot dibandingkan dengan jumlah tweet bersentimen positif pasangan Anies-Sandi, kemudian dihitung persentasenya. Persentase tersebut kemudian dihitung tingkat kesalahannya menggunakan Mean Absolute Error dengan membandingkan persentase tersebut dengan hasil resmi dari KPUD DKI Jakarta. Semakin kecil nilai Mean Absolute Error, maka persentase tersebut semakin mendekati hasil resmi. Sehingga, dengan semakin kecilnya nilai Mean Absolute Error, dapat dikatakan bahwa semakin besar pengaruh sentimen tersebut terhadap prediksi hasil pilkada di Twitter. Karena kandidat dalam pilkada

ini hanya ada dua, selain hanya menggunakan tweet positif, penelitian ini menggunakan asumsi, bahwa sentimen negatif terhadap suatu kandidat berarti tweet tersebut mendukung kandidat lainnya (positif terhadap kandidat lainnya). Asumsi ini juga digunakan pada penelitian Gayo-Avello (2011) dan Prasetyo (2014).

Penelitian ini membandingkan tiga metode prediksi hasil pilkada berdasarkan sentimen tweet yang digunakan, yakni hanya menggunakan tweet bersentimen positif saja, hanya menggunakan tweet bersentimen negatif saja, serta menggunakan tweet bersentimen positif dan negatif. Hal ini dimaksudkan untuk melihat sentimen manakah yang memiliki pengaruh paling besar terhadap hasil prediksi pilkada. Selain itu, dibandingkan juga hasil yang didapat jika data yang digunakan melalui proses *buzzer detection* dengan yang tidak. Hal ini dimaksudkan untuk melihat seberapa besar pengaruh *buzzer* di media sosial dalam mempengaruhi hasil prediksi pilkada serta menganalisa sentimen apa saja yang banyak dihasilkan oleh para *buzzer* untuk masing-masing pasangan calon. Skema perancangan sistem ditunjukkan pada Gambar 4.1.



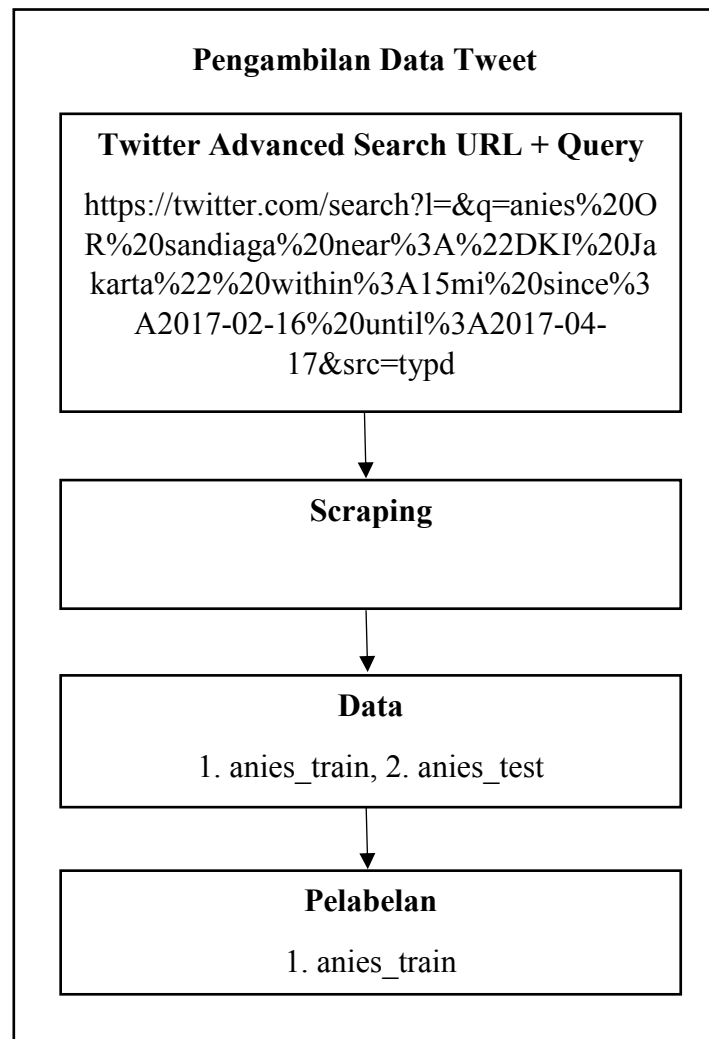
Gambar 4.1: Skema perancangan sistem

4.2.1 Pengambilan Data Tweet

Data tweet dari Twitter diambil dengan menggunakan *library* Selenium dan ditulis dalam bahasa Python. Kumpulan tweet yang relevan dengan kata kunci pada Tabel 4.8 dicari dan dikumpulkan ke dalam suatu file CSV. Pada proses pencarian digunakan fitur *advanced search* Twitter dengan tweet berlokasi di DKI Jakarta dan berbahasa Indonesia. Tweet yang dikumpulkan berada dalam rentang tanggal 16 Februari 2017 hingga 16 April 2017. Kemudian proses *scraping* dilakukan untuk mendapatkan seluruh teks tweet tersebut dari *tag-tag* html pada halaman Twitter. Alur pengambilan tweet ditunjukkan pada Gambar 4.2.

Tabel 4.8 : Daftar kata kunci dalam proses pencarian tweet

No. Urut	Kandidat	Kata Kunci
2	Basuki Tjahaja Purnama – Djarot Saiful Hidayat	“ahok”, ”djarot”, ”ahok-djarot”, “basuki tjahaja purnama”, “djarot saiful hidayat”
3	Anies Baswedan – Sandiaga Uno	“anies”, ”sandi”, ”anies-sandi”, “anies baswedan”, “sandiaga uno”



Gambar 4.2: Alur proses pengambilan data tweet

Data tweet yang sudah disimpan kemudian dilabeli secara manual. Dasar teori dalam pelabelan diambil dari penelitian Ibrahim, et al. (2015). Tabel 4.9 menunjukkan contoh komentar beserta labelnya.

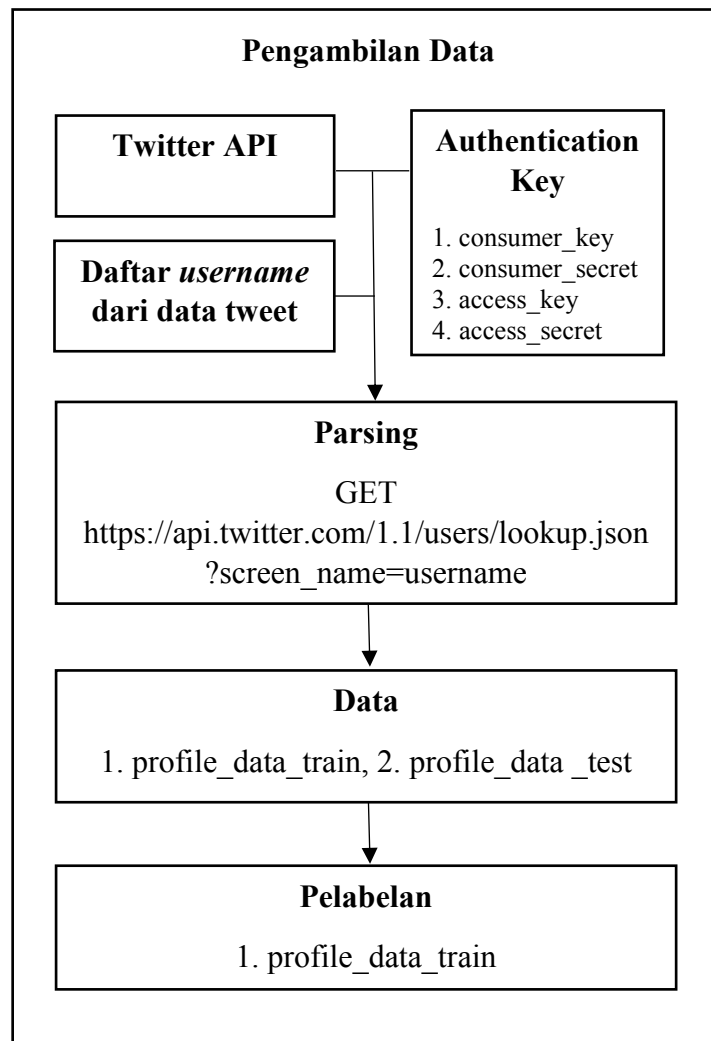
Tabel 4.9 : Contoh tweet beserta labelnya

Username	Tweet	Label
Joe_IRwan	Pilih yg udh nyata kerja nya... Ahok-djarot @Kayumanis Matraman - Jakarta Timur https://www.instagram.com/p/BS0EB2CBoGF/	pos

Username	Tweet	Label
VIVAcoid	Survei Charta Politika: Kekuatan Anies-Ahok Imbang Pekan Ini	net
atanghidayat89	Ahoker nyusup menghasud warga utk pilih ahok.. di tangkap oleh jawara betawi Sampe terkencing2 https://www.instagram.com/p/BRpLkEGl3tz/	neg

4.2.2 Pengambilan Data Informasi Akun

Data tweet dari Twitter diambil dengan menggunakan Twitter API dan *library* 'twitter' di Python. Kumpulan informasi akun-akun yang terdapat pada data tweet (baik data latih maupun data tes) dicari dan dikumpulkan ke dalam suatu file CSV. Informasi-informasi yang dikumpulkan adalah tanggal pembuatan akun, jumlah mengikuti (*following*), dan jumlah pengikut (*follower*). Umur akun didapat dengan menghitung jumlah bulan antara tanggal pembuatan akun dengan tanggal pilkada (19 April 2017). Alur pengambilan data informasi akun ditunjukkan pada Gambar 4.3.



Gambar 4.3: Alur proses pengambilan data informasi akun

Data informasi akun yang sudah disimpan kemudian dilabeli secara manual. Dasar teori dalam pelabelan diambil dari penelitian Ibrahim, et al. (2015). Tabel 4.10 menunjukkan contoh informasi akun beserta labelnya.

Tabel 4.10 : Contoh informasi akun beserta labelnya

Username	Umur akun	Jumlah mengikuti	Jumlah pengikut	Label
monstrar	96	120	506	normal
zahrihilir	11	1127	94	buzzer
pahalamansury	80	260	3465	normal

4.2.3 Pre-processing Data

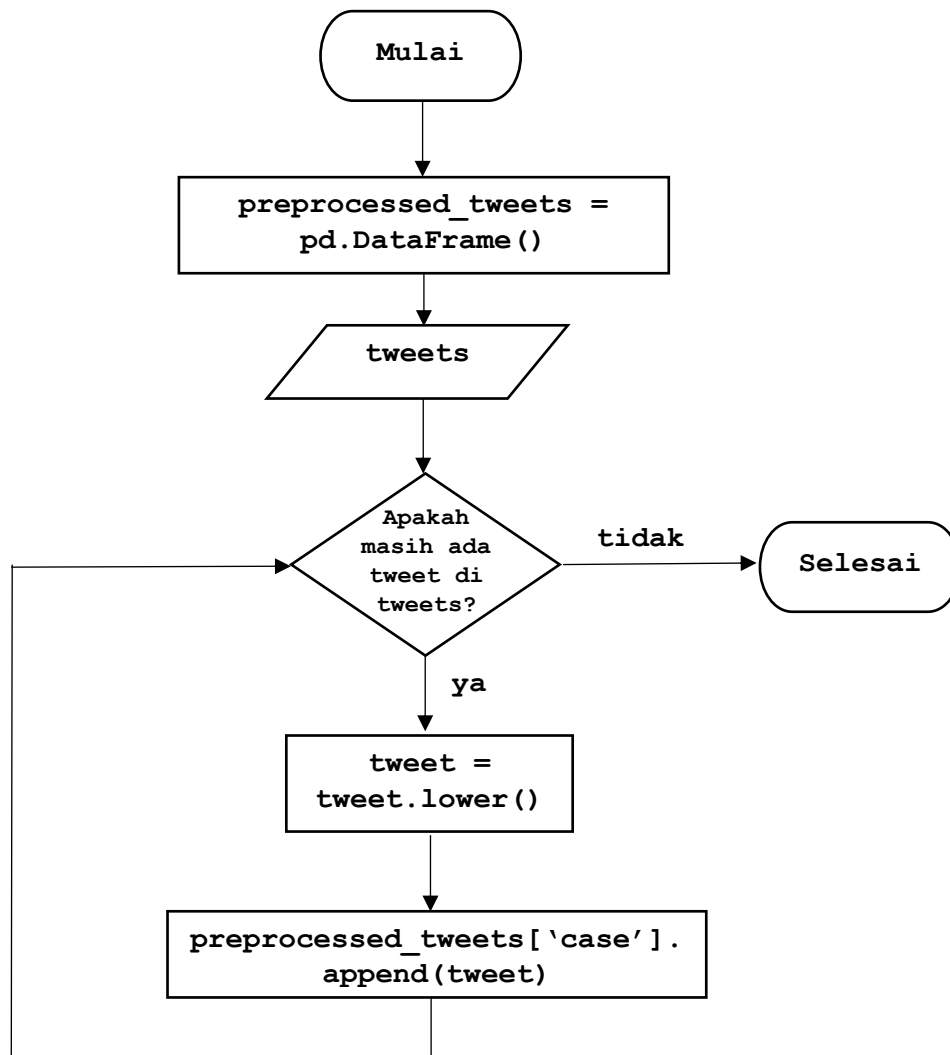
Dalam proses ini dilakukan proses *pre-processing* data yang akan digunakan dalam pemrosesan selanjutnya.

1. Case Folding

Case folding adalah proses penyamaan case dalam sebuah tweet. Tidak semua teks tweet konsisten dalam penggunaan huruf kapital. Oleh karena itu peran *case folding* dibutuhkan dalam mengkonversi keseluruhan teks dalam tweet menjadi suatu bentuk standar. Dalam hal ini, bentuk standarnya adalah huruf kecil. Ilustrasi proses *case folding* ditunjukkan pada Tabel 4.11. Alur proses *case folding* ditunjukkan pada Gambar 4.4.

Tabel 4.11 : Ilustrasi proses *case folding*

Sebelum Proses <i>Case Folding</i>	Sesudah Proses <i>Case Folding</i>
Pak Ahok yg sukses bikin semua tamu undangan lupa nyalamin (w/ & Ardhina at Gedung Dhanapala) [pic]	pak ahok yg sukses bikin semua tamu undangan lupa nyalamin (w/ & ardhina at gedung dhanapala) [pic]
Kalo yg tersangka? :."D RT@detikcom: Djarot: Jakarta Butuh Pemimpin yang Tak Pernah Dipecat http://detik.id/6OIPBS	kalo yg tersangka? :."d rt@detikcom: djarot: jakarta butuh pemimpin yang tak pernah dipecat http://detik.id/6oipbs
Dalam simulasi head to head Anis diprediksi mengungguli Ahok. Baca ulasannya di Majalah Gatra https://www.instagram.com/p/BQnDytKjPGy/	dalam simulasi head to head anis diprediksi mengungguli ahok. baca ulasannya di majalah gatra https://www.instagram.com/p/bqndytkjpgy/



Gambar 4.4: Alur proses *case folding*

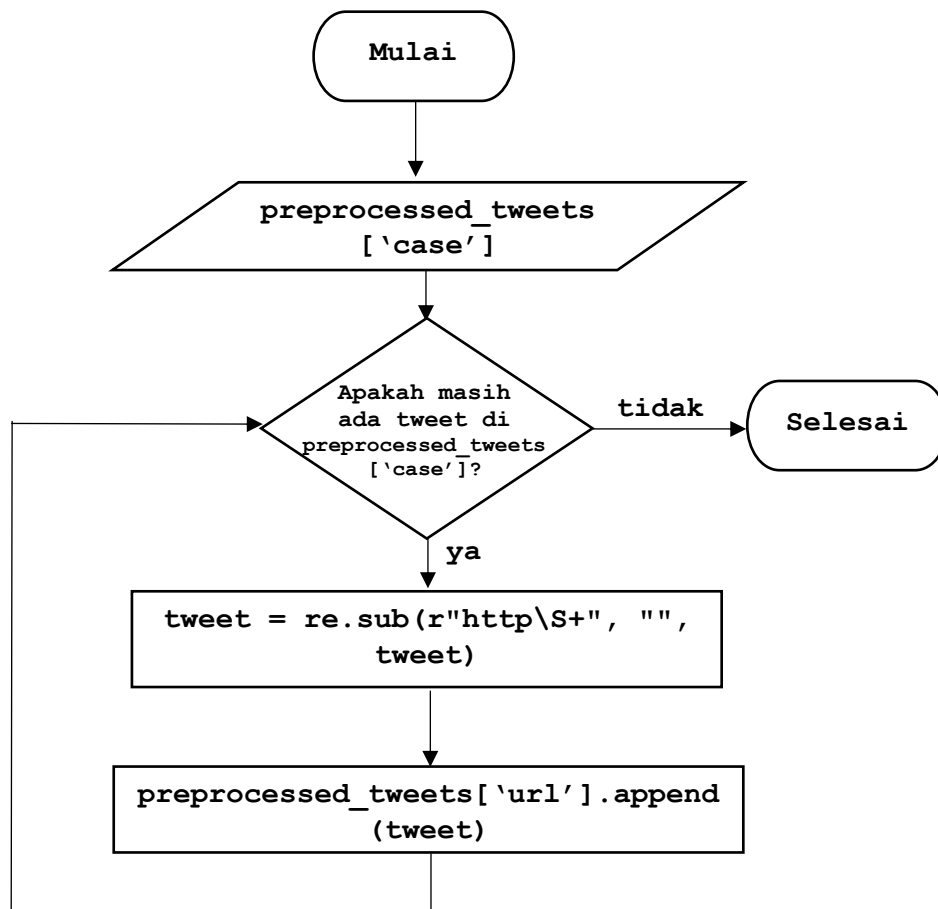
2. URLs Removal

URL adalah singkatan dari *Uniform Resource Locator*, yaitu merupakan rangkaian karakter menurut format standar tertentu, digunakan untuk menunjukan alamat dari suatu sumber, seperti dokumen, file dan gambar yang terdapat di internet. Keberadaan URL tidak dibutuhkan dalam melakukan klasifikasi sentimen. Oleh karena itu, URL-URL tersebut harus dihapus. *URLs Removal* adalah suatu proses untuk menghapus URL-URL yang berada pada sebuah tweet. Ilustrasi proses *URLs removal* ditunjukkan pada Tabel 4.12. Alur proses *URLs removal*

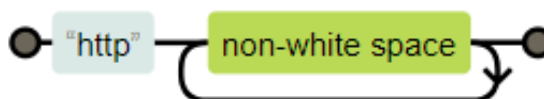
ditunjukkan pada Gambar 4.5. Ilustrasi proses *regular expression URLs removal* ditunjukkan pada Gambar 4.6.

Tabel 4.12 : Ilustrasi proses *URLs removal*

Sebelum Proses <i>URLs Removal</i>	Sesudah Proses <i>URLs Removal</i>
sebuah kebanggaan yang luar biasa dapat dikunjungi oleh wakil gubernur dki jakarta bpk. djarot https://www.instagram.com/p/bq9seggamy/	sebuah kebanggaan yang luar biasa dapat dikunjungi oleh wakil gubernur dki jakarta bpk. djarot
kalo yg tersangka? : "d rt@detikcom: djarot: jakarta butuh pemimpin yang tak pernah dipecat http://detik.id/6oipbs	kalo yg tersangka? : "d rt@detikcom: djarot: jakarta butuh pemimpin yang tak pernah dipecat
dalam simulasi head to head anis diprediksi mengungguli ahok. baca ulasannya di majalah gatra https://www.instagram.com/p/bqndytkjpgy/	dalam simulasi head to head anis diprediksi mengungguli ahok. baca ulasannya di majalah gatra



Gambar 4.5: Alur proses *URLs removal*



Gambar 4.6: Ilustrasi proses *regular expression URLs removal*

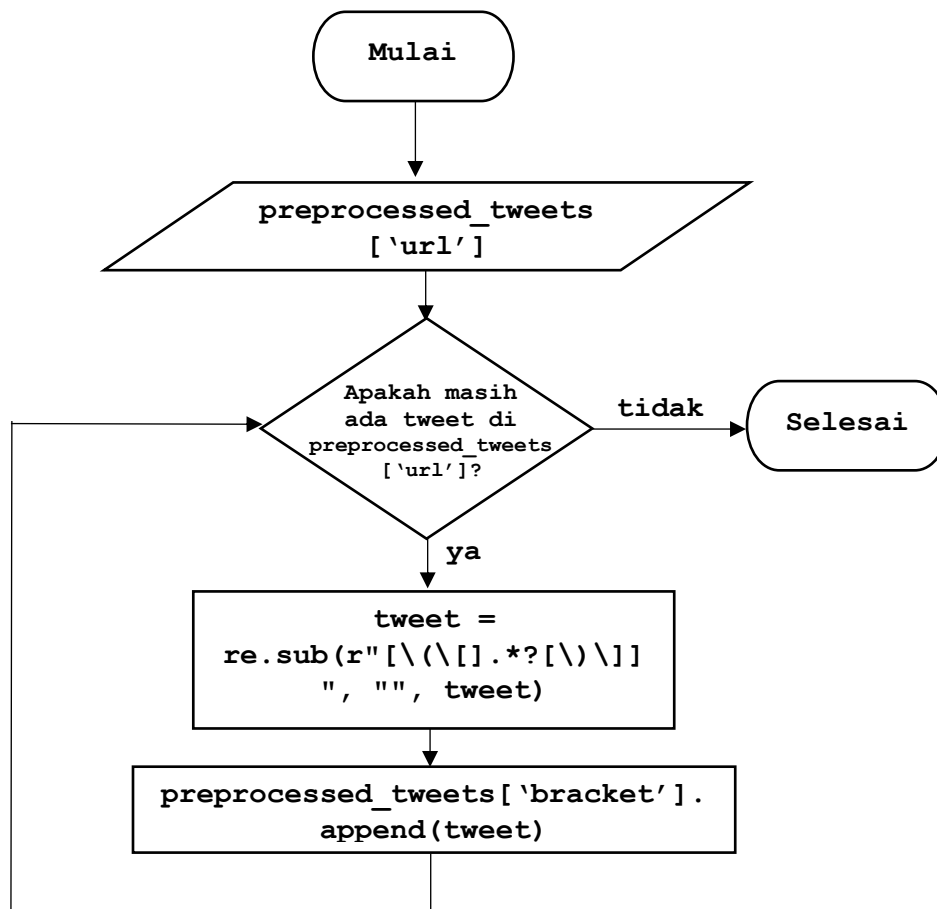
3. Brackets Removal

Dalam Twitter, tanda kurung (*bracket*) biasanya digunakan sebagai keterangan lokasi saat tweet dibuat, keterangan sedang bersama user lain saat tweet dibuat, serta sebagai penanda bahwa tweet tersebut memiliki gambar. Karena tidak relevan dalam proses klasifikasi sentimen, maka tanda kurung beserta isinya akan dihapus

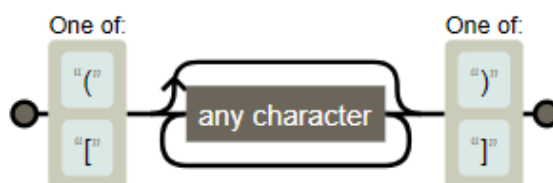
dalam proses *brackets removal*. Ilustrasi proses *brackets removal* ditunjukkan pada Tabel 4.13. Alur proses *brackets removal* ditunjukkan pada Gambar 4.7. Ilustrasi proses *regular expression brackets removal* ditunjukkan pada Gambar 4.8.

Tabel 4.13 : Ilustrasi proses *brackets removal*

Sebelum Proses <i>Brackets Removal</i>	Sesudah Proses <i>Brackets Removal</i>
pak ahok yg sukses bikin semua tamu undangan lupa nyalamin (w/ & ardhina at gedung dhanapala) [pic]	pak ahok yg sukses bikin semua tamu undangan lupa nyalamin
coblos kemeja putihnya aje! (with ramuhad at posko pemenangan anies sandi)	coblos kemeja putihnya aje!
yuksss .. malem ini nonton pak ahok & pak anies live di sctv.. #promokalikali (at sales dept. sctv) [pic]	yuksss .. malem ini nonton pak ahok & pak anies live di sctv.. #promokalikali



Gambar 4.7: Alur proses *brackets removal*



Gambar 4.8: Ilustrasi proses *regular expression brackets removal*

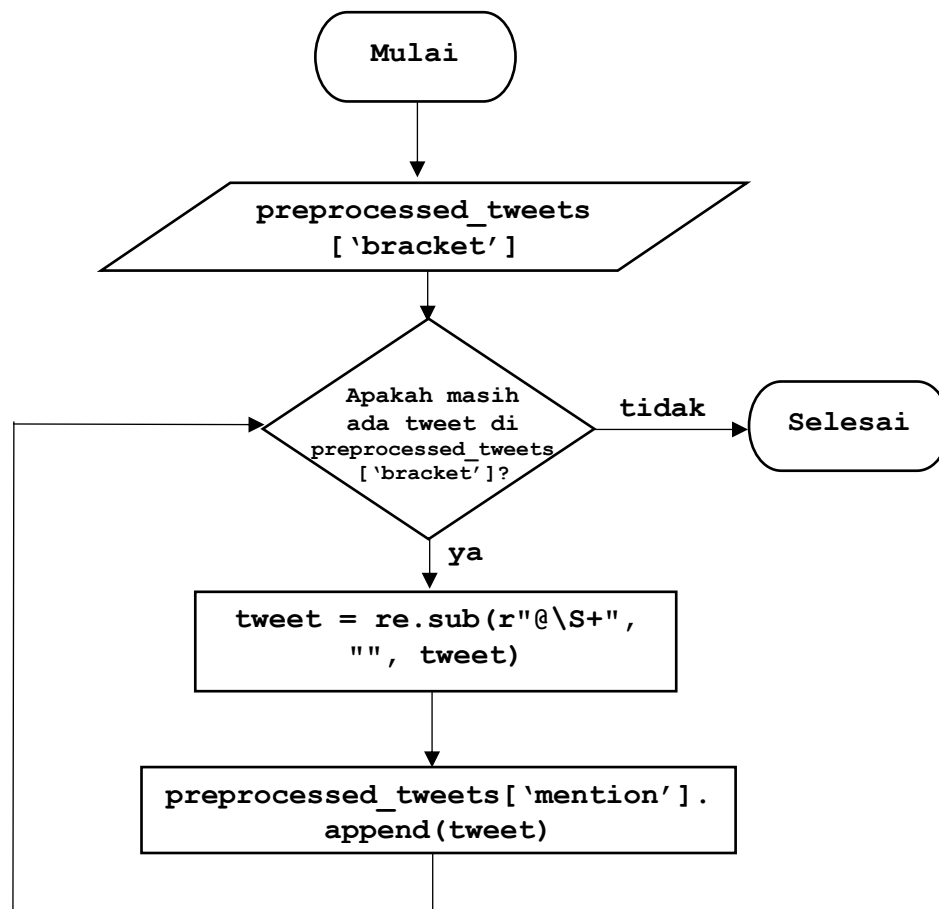
4. Mentions Removal

Mention adalah suatu cara untuk membuat link terhadap suatu akun Twitter. Cara ini biasanya digunakan saat kita akan membalas tweet atau ingin menandai suatu tweet kepada seseorang. Dalam proses klasifikasi sentimen, keberadaan

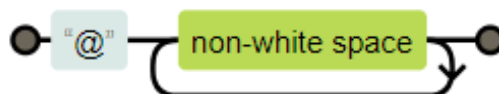
mention tidak dibutuhkan. Sehingga, *mention* akan dihapus dalam proses *mentions removal*. Ilustrasi proses *mentions removal* ditunjukkan pada Tabel 4.14. Alur proses *mentions removal* ditunjukkan pada Gambar 4.9. Ilustrasi proses *regular expression mentions removal* ditunjukkan pada Gambar 4.10..

Tabel 4.14 : Ilustrasi proses *mentions removal*

Sebelum Proses Mentions Removal	Sesudah Proses Mentions Removal
insyaallah besok di jl. boncel srengseng sawah mau ada senam anies sandi seruloh... semoga tim sukses @jktmajubersama dan @sandiuno hadir	insyaallah besok di jl. boncel srengseng sawah mau ada senam anies sandi seruloh... semoga tim sukses dan hadir
kalo yg tersangka? : "d rt@detikcom: djarot: jakarta butuh pemimpin yang tak pernah dipecat	kalo yg tersangka? : "d rt: djarot: jakarta butuh pemimpin yang tak pernah dipecat
@metro_tv: hadapi kasus hukum sandi irit bicara	: hadapi kasus hukum sandi irit bicara



Gambar 4.9: Alur proses *mentions removal*



Gambar 4.10: Ilustrasi proses *regular expression mentions removal*

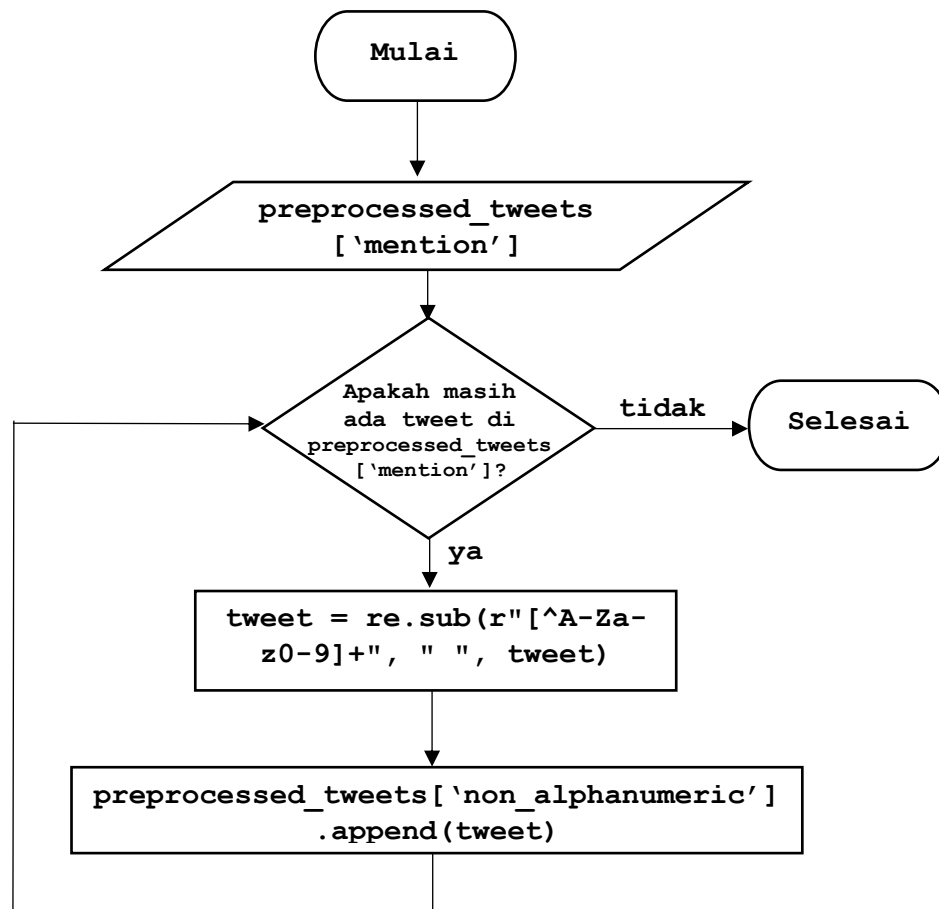
5. Non-alphanumeric Characters Removal

Karakter *non-alphanumeric* adalah karakter-karakter yang bukan huruf (*uppercase* dan *lowercase*), angka, dan *white space* (spasi, *tab*, atau *enter*). Karakter-karakter yang bukan *alphanumeric* dapat berupa tanda baca, seperti koma, titik, tanda tanya, tanda seru, dan yang lainnya. Dalam proses klasifikasi sentimen,

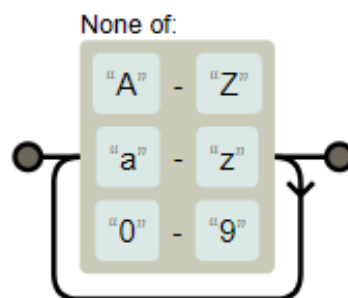
keberadaan karakter yang bukan *alphanumeric* tidak dibutuhkan sehingga, karakter-karakter yang bukan *alphanumeric* tersebut akan dihapus dalam proses *non-alphanumeric characters removal*. Ilustrasi proses *non-alphanumeric characters removal* ditunjukkan pada Tabel 4.15. Alur proses *non-alphanumeric characters removal* ditunjukkan pada Gambar 4.11. Ilustrasi proses *regular expression non-alphanumeric characters removal* ditunjukkan pada Gambar 4.12.

Tabel 4.15 : Ilustrasi proses *non-alphanumeric characters removal*

Sebelum Proses Non-alphanumeric Characters Removal	Sesudah Proses Non-alphanumeric Characters Removal
insyaallah besok di jl. boncel srengseng sawah mau ada senam anies sandi seruloh... semoga tim sukses dan hadir	insyaallah besok di jl boncel srengseng sawah mau ada senam anies sandi seruloh semoga tim sukses dan hadir
kalo yg tersangka? : "d rt: djarot: jakarta butuh pemimpin yang tak pernah dipecat	kalo yg tersangka d rt djarot jakarta butuh pemimpin yang tak pernah dipecat
: hadapi kasus hukum sandi irit bicara	hadapi kasus hukum sandi irit bicara



Gambar 4.11: Alur proses *non-alphanumeric characters removal*



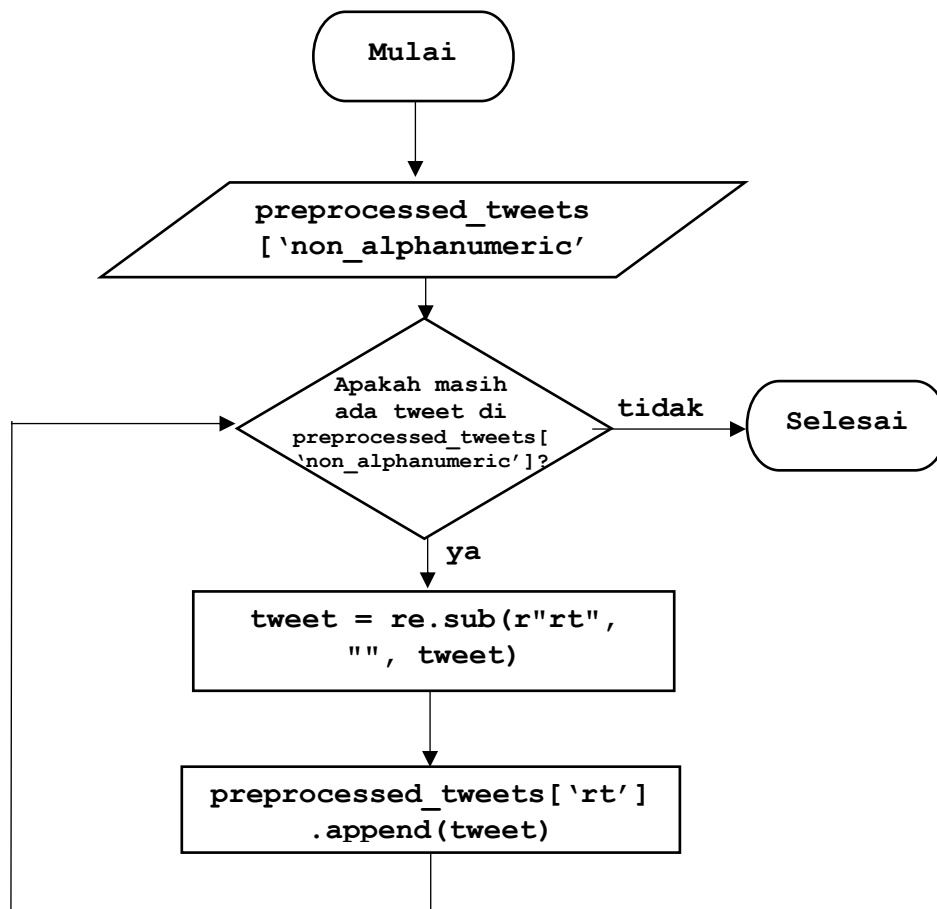
Gambar 4.12: Ilustrasi proses *regular expression non-alphanumeric characters removal*

6. RT Removal

Retweet atau RT, secara bahasa memiliki arti ‘mengicaukan kembali’. Dalam Twitter, *retweet* berfungsi untuk mengulang kembali tweet yang telah ada agar dapat dibagikan kembali kepada pengikut-pengikut yang ada di Twitter. *RT removal* adalah proses menghapus string ‘rt’ dari suatu tweet. Hal ini dikarenakan string ‘rt’ tidak relevan ketika melakukan klasifikasi sentimen. Ilustrasi proses *RT removal* ditunjukkan pada Tabel 4.16. Alur proses *RT removal* ditunjukkan pada Gambar 4.13.

Tabel 4.16 : Ilustrasi proses *RT removal*

Sebelum Proses <i>RT Removal</i>	Sesudah Proses <i>RT Removal</i>
tulisan menarik mengena rt pengakuan mengejutkan pragiwaksono dibayar anies baswedan	tulisan menarik mengena pengakuan mengejutkan pragiwaksono dibayar anies baswedan
kalo yg tersangka d rt djarot jakarta butuh pemimpin yang tak pernah dipecat	kalo yg tersangka d djarot jakarta butuh pemimpin yang tak pernah dipecat
hmmm rt novel diteror djarot kami makin kuat perang lawan koruptor	hmmm novel diteror djarot kami makin kuat perang lawan koruptor



Gambar 4.13: Alur proses *RT removal*

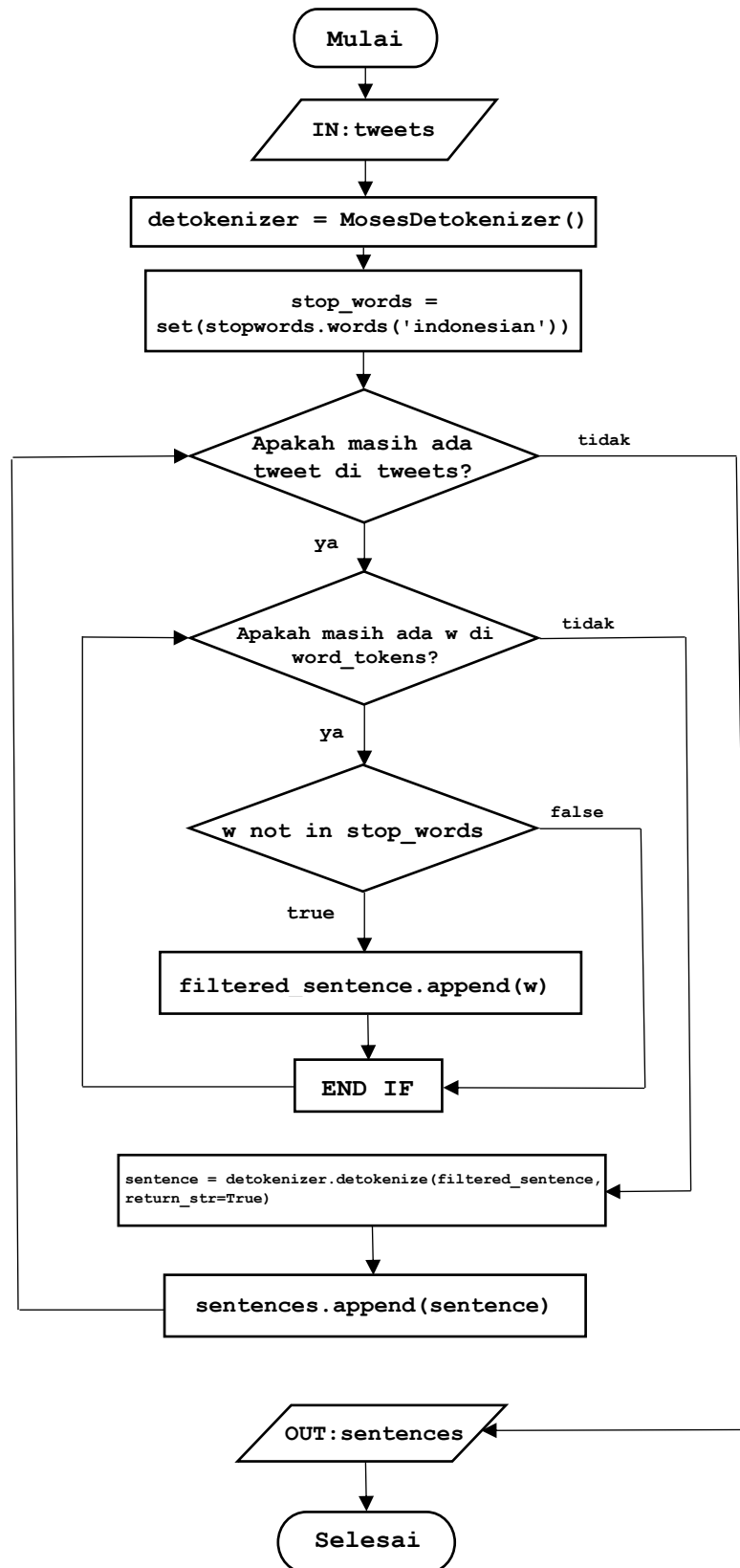
7. Stopwords Removal

Stopwords adalah kata-kata umum (*common words*) yang biasanya muncul dalam jumlah besar dan dianggap tidak memiliki makna. Karena tidak dibutuhkan pada proses klasifikasi sentimen maka *stopwords* dihapus dalam proses *stopwords removal*. Ilustrasi proses *stopwords removal* ditunjukkan pada Tabel 4.17. Alur proses *stopwords removal* ditunjukkan pada Gambar 4.14.

Tabel 4.17 : Ilustrasi proses *stopwords removal*

Sebelum Proses <i>Stopwords Removal</i>	Sesudah Proses <i>Stopwords Removal</i>
sidang 13 penodaan agama di pengadilan jakarta utara gedung	sidang 13 penodaan agama pengadilan jakarta utara gedung

Sebelum Proses <i>Stopwords Removal</i>	Sesudah Proses <i>Stopwords Removal</i>
memteri pertanian jakarta utara dengan terdakwa ahok	memteri pertanian jakarta utara terdakwa ahok
kalo yg tersangka d djarot jakarta butuh pemimpin yang tak pernah dipecat	kalo tersangka d djarot jakarta butuh pemimpin pernah dipecat
ruhut tanggapi soal jokowi ahok untuk pilpres 2019	ruhut tanggapi soal jokowi ahok pilpres 2019



Gambar 4.14: Alur proses *stopwords removal*

4.2.4 Pelatihan Model Klasifikasi Buzzer

Dalam proses pelatihan model klasifikasi *buzzer*, algoritma Gaussian Naive Bayes digunakan untuk melakukan proses klasifikasi antara akun *buzzer* dengan akun normal. Tweet-tweet yang dihasilkan oleh akun yang terdeteksi sebagai *buzzer* kemudian dihapus dalam proses *buzzer's tweets removal*. Berikut adalah contoh proses klasifikasi *buzzer* menggunakan algoritma Gaussian Naive Bayes. Contoh informasi akun yang akan melalui proses Gaussian Naive Bayes ditunjukkan pada Tabel 4.18.

Tabel 4.18 : Contoh informasi akun yang akan melalui proses Gaussian Naive Bayes

	No.	Umur akun	Jumlah mengikuti	Jumlah pengikut	Class
Data latih	1	95	2136	4330	normal
	2	9	6870	671	buzzer
	3	107	912	7318	normal
	4	21	1767	580	buzzer
Data tes	5	13	5813	231	?

Untuk setiap elemen pada kolom umur akun, jumlah mengikuti, dan jumlah pengikut pada data latih dilakukan normalisasi agar pada proses perhitungan probabilitas menggunakan fungsi Gaussian, data pada kolom yang memiliki nilai kecil mampu mengimbangi data pada kolom yang memiliki nilai lebih besar. Normalisasi dihitung dengan melakukan proses pengurangan pada data di suatu baris dan kolom dengan nilai minimal pada kolom tersebut. Hasilnya kemudian dibagi dengan hasil pengurangan antara nilai maksimum kolom tersebut dengan nilai minimalnya. Proses ini dinamakan *Min-Max Normalization* ditunjukkan pada persamaan (4.1).

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (4.1)$$

Hasil dari proses *Min-Max Normalization* ditunjukkan pada Tabel 4.19.

Tabel 4.19 : Contoh informasi akun yang akan melalui proses Gaussian Naive Bayes setelah dilakukan Min-Max Normalization

	No.	Umur akun	Jumlah mengikuti	Jumlah pengikut	Class
Data latih	1	0,8776	0,2054	0,5784	normal
	2	0	1	0,0621	buzzer
	3	1	0	1	normal
	4	0,1224	0,1435	0.0492	buzzer
Data tes	5	0,0408	0,8226	0	?

Fungsi Gaussian yang ditunjukkan pada persamaan (4.2) digunakan pada hasil dari proses *Min-Max Normalization* pada Tabel 4.19 untuk masing-masing kolom.

$$N(\mu, \sigma^2)(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (4.2)$$

di mana μ adalah rata-rata (mean) dari data latih, σ^2 adalah variansi dari data latih, $e \approx 2,71828$, $\pi \approx 3,14159$, dan x adalah nilai dari data tes pada kolom tersebut. Rumus mean ditunjukkan oleh persamaan (4.3) sedangkan rumus variansi ditunjukkan oleh persamaan (4.4).

$$\mu = \frac{1}{n} \sum_{i=1}^n x_i \quad (4.3)$$

$$\sigma^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2 \quad (4.4)$$

di mana n adalah jumlah data dan x_i adalah data ke- i .

Hasil perhitungan mean dan variansi data latih untuk kelas normal ditunjukkan pada Tabel 4.20 sedangkan hasil perhitungan mean dan variansi data latih untuk kelas *buzzer* ditunjukkan pada Tabel 4.21.

Tabel 4.20 : Contoh hasil perhitungan mean dan variansi untuk kelas normal

Normal	Variabel		
	Umur akun	Jumlah mengikuti	Jumlah pengikut
Mean	0,9388	0,1027	0,7892
Variansi	0,0037	0,0105	0,0444

Tabel 4.21 : Contoh hasil perhitungan mean dan variansi untuk kelas *buzzer*

<i>Buzzer</i>	Variabel		
	Umur akun	Jumlah mengikuti	Jumlah pengikut
Mean	0,0612	0,5718	0,2019
Variansi	0,0037	0,1834	0,0195

Hasil probabilitas masing-masing variabel yang didapat menggunakan fungsi Gaussian, yakni $P(\text{age}|\text{buzzer})$, $P(\text{following}|\text{buzzer})$, $P(\text{follower}|\text{buzzer})$, $P(\text{age}|\text{normal})$, $P(\text{following}|\text{normal})$, dan $P(\text{follower}|\text{normal})$ kemudian dikalikan untuk masing-masing kelas. Hasilnya berupa probabilitas $P(x|\text{buzzer})$ dan $P(x|\text{normal})$ di mana x adalah data tes x .

Untuk menentukan apakah suatu akun termasuk kelas *buzzer* atau bukan, maka perlu dihitung probabilitas suatu kelas untuk data x yang ditunjukkan pada kolom $P(\text{buzzer}|x)$ pada Tabel 4.22. Probabilitas tersebut dihitung dengan menggunakan persamaan (4.5).

$$P(\text{buzzer}|x) = \frac{P(x|\text{buzzer})P(\text{buzzer})}{P(x|\text{buzzer})P(\text{buzzer}) + P(x|\text{normal})P(\text{normal})} \quad (4.5)$$

di mana $P(\text{buzzer})$ adalah probabilitas kelas *buzzer* dan $P(\text{normal})$ adalah probabilitas kelas normal. Rumus probabilitas kelas *buzzer* ditunjukkan pada persamaan (4.6) sedangkan rumus probabilitas kelas normal ditunjukkan pada persamaan (4.7)

$$P(\text{buzzer}) = \frac{N_{\text{buzzer}}}{N} \quad (4.6)$$

$$P(buzzer) = \frac{N_{normal}}{N} \quad (4.7)$$

Tabel 4.22 : Contoh hasil perhitungan probabilitas menggunakan fungsi Gaussian

Class	$P(age class)$	$P(following class)$	$P(follower class)$	$P(x class)$	$P(buzzer x)$
normal	1,152E-46	8,308E-11	0,00171	1,639E-59	1
buzzer	6,16639	0,7847	1,00579	4,86686	

Dikarenakan nilai dari $P(buzzer|x)$ lebih besar daripada 0,5 maka akun tersebut diklasifikasikan sebagai *buzzer*.

4.2.5 Pelatihan Model Klasifikasi Sentimen

Dalam proses pelatihan model klasifikasi sentimen, algoritma Multinomial Naive Bayes digunakan untuk mengklasifikasikan suatu tweet apakah bersentimen positif, negatif, ataupun netral. Fitur yang digunakan dalam penelitian ini berupa *unigram*. Berikut adalah contoh proses klasifikasi sentimen menggunakan algoritma Multinomial Naive Bayes. Contoh tweet yang akan melalui proses Multinomial Naive Bayes ditunjukkan pada Tabel 4.23.

Tabel 4.23 : Contoh tweet yang akan melalui proses Multinomial Naive Bayes

	No.	Tweets	Class
Data latih	1	satu paket warga terhormat tdk gila kohormatan lanjutkan pak basuki pak djarot	pos
	2	sebetulnya ingin lihat debat gagasan pasangan anies sandi demgan ahok djarot	net
	3	diduga politik uang giring nidji pendukung ahok dilaporkan bawaslu serbet mania perlu khawatir	neg
	4	slmt menjalankan tugas sbgai pelayan warga dki pak basuki pak djarot	pos

Data tes	5	alhamdulillah bravo pak basuki pak djarot hasil kerjanya nyata warga dki bagusss	?
-----------------	---	--	---

Probabilitas *prior* dihitung dengan membagi jumlah suatu kelas pada data latih dengan jumlah seluruh data latih. Rumus perhitungan probabilitas *prior* ditunjukkan pada persamaan (4.5). Hasilnya ditunjukkan pada Tabel 4.24.

$$\hat{P}(c) = \frac{N_c}{N} \quad (4.5)$$

Tabel 4.24 : Contoh hasil perhitungan probabilitas *prior*

$P(pos)$	$\frac{1}{2}$
$P(net)$	$\frac{1}{4}$
$P(neg)$	$\frac{1}{4}$

Untuk setiap kelas yang ada, probabilitas kondisional untuk masing-masing token (kata yang unik) pada data tes dihitung dengan menggunakan persamaan (4.6).

$$\hat{P}(w|c) = \frac{\text{count}(w, c) + 1}{\text{count}(c) + |V|} \quad (4.6)$$

di mana $\text{count}(w, c)$ adalah jumlah suatu token tertentu pada suatu kelas, $\text{count}(c)$ adalah jumlah seluruh token pada suatu kelas, dan $|V|$ adalah jumlah seluruh token pada seluruh kelas.

Dengan menggunakan persamaan (4.6), probabilitas kondisional untuk masing-masing token pada data tes dihitung dan hasilnya ditunjukkan pada Tabel 4.25.

Tabel 4.25 : Contoh hasil perhitungan probabilitas kondisional untuk masing-masing token pada tiap kelas

Kata	$P(kata pos)$	$P(kata net)$	$P(kata neg)$
alhamdulillah	$\frac{0 + 1}{23 + 39} = \frac{1}{62}$	$\frac{0 + 1}{11 + 39} = \frac{1}{50}$	$\frac{0 + 1}{13 + 39} = \frac{1}{52}$

Kata	$P(kata pos)$	$P(kata net)$	$P(kata neg)$
bravo	$\frac{0+1}{23+39} = \frac{1}{62}$	$\frac{0+1}{11+39} = \frac{1}{50}$	$\frac{0+1}{13+39} = \frac{1}{52}$
pak	$\frac{4+1}{23+39} = \frac{5}{62}$	$\frac{0+1}{11+39} = \frac{1}{50}$	$\frac{0+1}{13+39} = \frac{1}{52}$
basuki	$\frac{2+1}{23+39} = \frac{3}{62}$	$\frac{0+1}{11+39} = \frac{1}{50}$	$\frac{0+1}{13+39} = \frac{1}{52}$
djarot	$\frac{2+1}{23+39} = \frac{3}{62}$	$\frac{1+1}{11+39} = \frac{2}{50}$	$\frac{0+1}{13+39} = \frac{1}{52}$
hasil	$\frac{0+1}{23+39} = \frac{1}{64}$	$\frac{0+1}{11+39} = \frac{1}{50}$	$\frac{0+1}{13+39} = \frac{1}{52}$
kerjanya	$\frac{0+1}{23+39} = \frac{1}{62}$	$\frac{0+1}{11+39} = \frac{1}{50}$	$\frac{0+1}{13+39} = \frac{1}{52}$
nyata	$\frac{0+1}{23+39} = \frac{1}{62}$	$\frac{0+1}{11+39} = \frac{1}{50}$	$\frac{0+1}{13+39} = \frac{1}{52}$
warga	$\frac{2+1}{23+39} = \frac{3}{62}$	$\frac{0+1}{11+39} = \frac{1}{50}$	$\frac{0+1}{13+39} = \frac{1}{52}$
dki	$\frac{1+1}{23+39} = \frac{2}{62}$	$\frac{0+1}{11+39} = \frac{1}{50}$	$\frac{0+1}{13+39} = \frac{1}{52}$
bagusss	$\frac{0+1}{23+39} = \frac{1}{62}$	$\frac{0+1}{11+39} = \frac{1}{50}$	$\frac{0+1}{13+39} = \frac{1}{52}$

Hasil perhitungan probabilitas kondisional setiap token kemudian dikalikan untuk masing-masing kelasnya dan dikalikan lagi dengan probabilitas *prior* kelas tersebut. Hasilnya berupa probabilitas suatu kelas pada suatu tweet seperti yang ditunjukkan di bawah ini:

$$P(pos|n5) \propto \frac{1}{62} * \frac{1}{62} * \frac{5}{62} * \frac{3}{62} * \frac{5}{62} * \frac{3}{62} * \frac{1}{62} * \frac{1}{62} * \frac{1}{62} * \frac{3}{62} * \frac{1}{31} * \frac{1}{62} * \frac{1}{2} = 2.0922015e-19$$

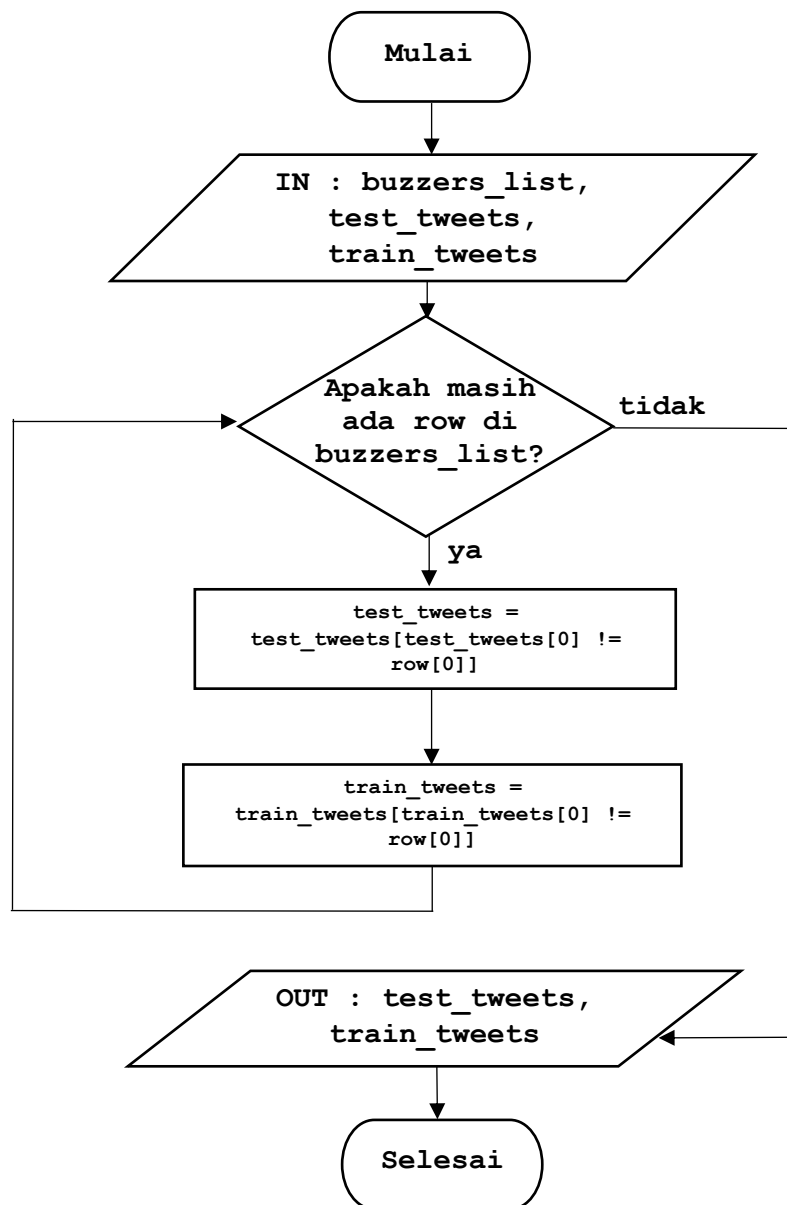
$$P(net|n5) \propto \frac{1}{50} * \frac{1}{50} * \frac{1}{50} * \frac{1}{50} * \frac{1}{50} * \frac{1}{25} * \frac{1}{50} * \frac{1}{50} * \frac{1}{50} * \frac{1}{50} * \frac{1}{50} * \frac{1}{50} * \frac{1}{4} = 2.048e-21$$

$$P(neg|n5) \propto \frac{1}{52} * \frac{1}{52} * \frac{1}{52} * \frac{1}{52} * \frac{1}{52} * \frac{1}{52} * \frac{1}{52} * \frac{1}{52} * \frac{1}{52} * \frac{1}{52} * \frac{1}{52} * \frac{1}{52} * \frac{1}{4} = 6.3958738e-22$$

Jika probabilitas suatu kelas pada suatu tweet lebih besar dibandingkan probabilitas kelas lainnya pada tweet tersebut, maka tweet tersebut diklasifikasikan sesuai dengan kelas dengan probabilitas terbesar tersebut. Dikarenakan nilai dari $P(pos|n5)$ lebih besar dibandingkan nilai $P(net|n5)$ dan $P(neg|n5)$ maka tweet tersebut diklasifikasikan sebagai positif.

4.2.6 Penghapusan Tweet Buzzer

Dalam proses penghapusan tweet *buzzer*, data tweet (baik data latih maupun data tes) yang dihasilkan oleh akun yang terklasifikasi sebagai *buzzer* dihapus untuk kemudian dibandingkan dengan data tweet yang tidak dilakukan penghapusan tweet *buzzernya*. Alur proses penghapusan tweet *buzzer* ditunjukkan pada Gambar 4.15.



Gambar 4.15: Alur proses penghapusan tweet *buzzer*

4.2.7 Pengujian Model Klasifikasi

Pengujian dilakukan dengan menggunakan metode *10-fold cross validation* dengan memproses seluruh data latih. Untuk setiap *fold*, data latih tersebut dibagi menjadi data latih dan data tes dengan rasio 9 banding 1 (untuk jumlah *fold* sama dengan sepuluh). Ilustrasi proses *10-fold cross validation* ditunjukkan pada Gambar 4.16 di mana kotak putih berarti data latih pada *fold* tersebut sedangkan kotak hitam berarti data tes pada *fold* tersebut.

<i>Fold ke-</i>	Dataset									
1	■	□	□	□	□	□	□	□	□	□
2	□	■	□	□	□	□	□	□	□	□
3	□	□	■	□	□	□	□	□	□	□
4	□	□	□	■	□	□	□	□	□	□
5	□	□	□	□	■	□	□	□	□	□
6	□	□	□	□	□	■	□	□	□	□
7	□	□	□	□	□	□	■	□	□	□
8	□	□	□	□	□	□	□	■	□	□
9	□	□	□	□	□	□	□	□	■	□
10	□	□	□	□	□	□	□	□	□	■

Gambar 4.16: Ilustrasi proses *10-fold cross validation*

Hasil klasifikasi dari *10 fold* dikumpulkan dan ditampilkan dalam bentuk *confusion matrix* untuk selanjutnya dilakukan perhitungan performa. Pengujian model klasifikasi dibagi menjadi dua bagian. Pengujian pertama ditujukan untuk menghitung performa dari model klasifikasi *buzzer* sedangkan pengujian kedua ditujukan untuk menghitung performa dari model klasifikasi sentimen. Pengujian model klasifikasi sentimen juga dibagi menjadi dua bagian. Pengujian pertama ditujukan untuk menghitung performa model klasifikasi sentimen untuk pasangan calon nomor urut dua, yakni Ahok-Djarot sedangkan pengujian kedua ditujukan untuk menghitung performa model klasifikasi sentimen untuk pasangan calon nomor urut tiga, yakni Anies-Sandi. Performa model klasifikasi *buzzer* yang dihasilkan meliputi *accuracy*, *precision*, *recall*, dan *f1-score*, sedangkan untuk performa model klasifikasi sentimen hanya menghasilkan nilai *accuracy* saja. Hal

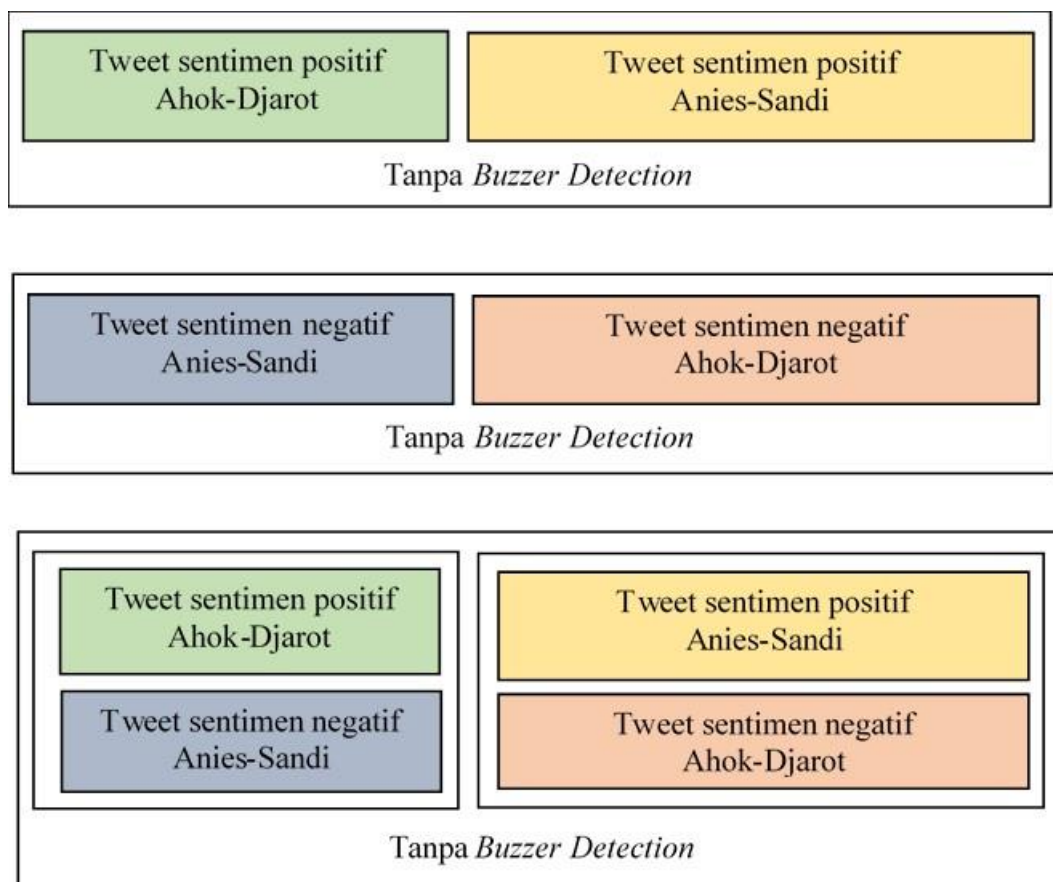
ini disebabkan karena pada saat melakukan klasifikasi dengan jumlah kelas lebih dari dua, nilai *precision* dan *recall* hanya dapat dihitung untuk masing-masing kelas individu. Walaupun nilai *precision* dan *recall* dapat dirata-ratakan pada masing-masing kelas untuk mendapatkan nilai umum dari keseluruhan sistem, namun hasilnya tidak terlalu berguna, lebih baik hanya menggunakan nilai *accuracy* untuk pengujian performa model (Beiletes, 2013).

4.2.8 Perbandingan Metode Prediksi Hasil Pilkada

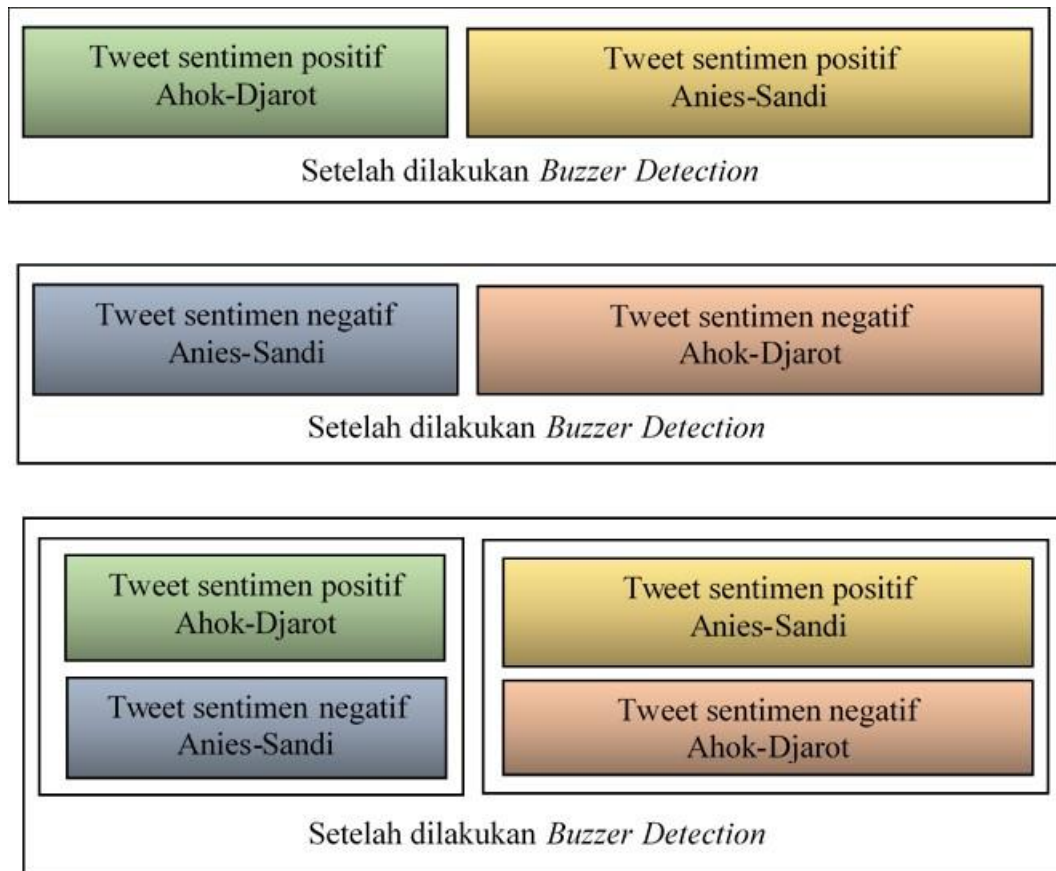
Setelah semua tweet terklasifikasi positif, negatif, atau netral, jumlah tweet bersentimen tertentu terhadap masing-masing pasangan calon digunakan dalam menghitung prediksi hasil pilkada. Ibrahim (2015) dalam penelitiannya hanya menggunakan tweet bersentimen positif saja. Hal ini didasari oleh penelitian Prasetyo (2014) yang juga menggunakan tweet bersentimen positif setelah menyimpulkan bahwa penggunaan sentimen positif saja lebih baik bila dibandingkan dengan penggunaan sentimen negatif.

Perhitungan prediksi hasil pilkada dilakukan dengan membandingkan jumlah tweet yang mendukung masing-masing pasangan calon, misalnya jumlah tweet bersentimen positif pasangan Ahok-Djarot dibandingkan dengan jumlah tweet bersentimen positif pasangan Anies-Sandi, kemudian dihitung persentasenya. Persentase tersebut kemudian dihitung tingkat kesalahannya menggunakan Mean Absolute Error dengan membandingkan persentase tersebut dengan hasil resmi dari KPUD DKI Jakarta. Semakin kecil nilai Mean Absolute Error, maka persentase tersebut semakin mendekati hasil resmi. Sehingga, dengan semakin kecilnya nilai Mean Absolute Error, dapat dikatakan bahwa semakin besar pengaruh sentimen tersebut terhadap prediksi hasil pilkada di Twitter. Karena kandidat dalam pilkada ini hanya ada dua, selain hanya menggunakan tweet positif, penelitian ini menggunakan asumsi, bahwa sentimen negatif terhadap suatu kandidat berarti tweet tersebut mendukung kandidat lainnya (positif terhadap kandidat lainnya). Asumsi ini juga digunakan pada penelitian Gayo-Avello (2011) dan Prasetyo (2014).

Penelitian ini membandingkan tiga metode prediksi hasil pilkada berdasarkan sentimen tweet yang digunakan, yakni hanya menggunakan tweet bersentimen positif saja, hanya menggunakan tweet bersentimen negatif saja, serta menggunakan tweet bersentimen positif dan negatif. Hal ini dimaksudkan untuk melihat sentimen manakah yang memiliki pengaruh paling besar terhadap hasil prediksi pilkada. Selain itu, dibandingkan juga hasil yang didapat jika data yang digunakan melalui proses *buzzer detection* dengan yang tidak. Hal ini dimaksudkan untuk melihat seberapa besar pengaruh *buzzer* di media sosial dalam mempengaruhi hasil prediksi pilkada serta menganalisa sentimen apa saja yang banyak dihasilkan oleh para *buzzer* untuk masing-masing pasangan calon. Total metode yang akan dibandingkan yakni sebanyak enam metode. Tiga metode tanpa melalui proses *buzzer detection* ditunjukkan pada Gambar 4.17, sedangkan tiga metode sesudah melalui proses *buzzer detection* ditunjukkan pada Gambar 4.18.



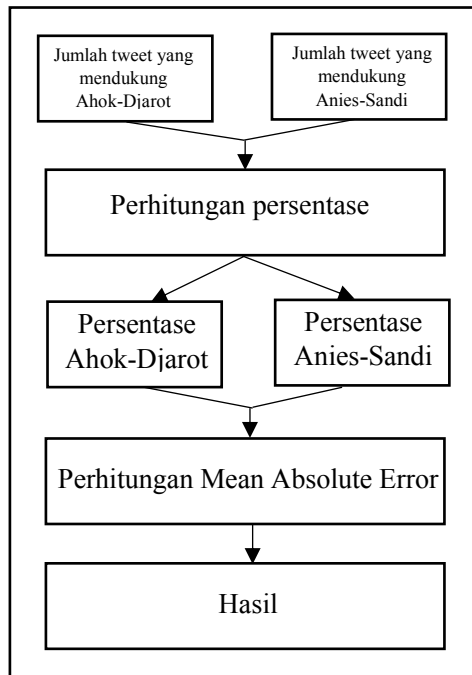
Gambar 4.17: Ilustrasi tiga metode prediksi hasil pilkada DKI Jakarta 2017 tanpa melalui proses *buzzer detection*



Gambar 4.18: Ilustrasi tiga metode prediksi hasil pilkada DKI Jakarta 2017 sesudah melalui proses *buzzer detection*

4.2.9 Skema Evaluasi Metode Prediksi Hasil Pilkada

Persentase tweet yang mendukung untuk masing-masing pasangan calon yang telah didapat pada proses sebelumnya, kemudian dievaluasi dengan menghitung tingkat kesalahannya terhadap hasil resmi dari KPUD DKI Jakarta. Metode perhitungan kesalahan yang digunakan adalah Mean Absolute Error. Skema evaluasi metode prediksi hasil pilkada ditunjukkan pada Gambar 4.19.



Gambar 4.19: Skema evaluasi metode prediksi hasil pilkada

BAB V

IMPLEMENTASI

5.1 Spesifikasi *Hardware* dan *Software*

Pada penelitian ini, perancangan program dan analisis diimplementasikan dengan menggunakan bahasa pemrograman Python dan diujikan pada komputer dengan spesifikasi sebagai berikut:

1. Processor Intel Core i7 2,4 GHz
2. RAM 8 GB
3. Sistem Operasi Windows 10

Dan dengan *software* serta *library* pendukung sebagai berikut:

1. Anaconda (Python 3.6)
2. Spyder IDE 3.2.3
3. Selenium 3.8.0
4. Scikit-learn 0.19.0
5. Numpy 1.13.3
6. Pandas 0.21.0

5.2 Implementasi Sistem

5.2.1 Pengambilan Data

Dalam proses pengambilan data tweet di Twitter, digunakan *library* Selenium dan Chrome WebDriver untuk melakukan proses *scraping* data tweet dari situs Twitter. Pertama-tama, URL dari halaman *advanced search* Twitter dimasukkan ke dalam variabel '*url*' seperti yang ditunjukkan pada baris 6 Gambar 5.1. Pada baris 12, dilakukan proses perulangan sebanyak 1100 kali, di mana untuk setiap iterasi dalam perulangan, elemen '*body*' pada halaman '*url*' dilakukan proses *page down* dengan jeda setiap iterasi selama 0,2 detik. Angka 1100 dipilih karena setelah melakukan beberapa kali percobaan, dibutuhkan sekitar 1100 kali *page down* untuk mencapai bagian terbawah halaman *search*. Setelah seluruh halaman

search termuat dengan sempurna, maka dengan menggunakan fungsi `find_elements_by_class_name()`, elemen-elemen dengan kelas `'tweet-text'` dan `'users'` dicari untuk dimasukkan ke dalam variabel `'tweets'` dan `'users'`. Kedua variabel tersebut memuat teks tweet dan *username*-nya. Proses ini ditunjukkan pada baris 16 dan 17. Setelah itu, kedua variabel tersebut disimpan ke dalam file CSV untuk digunakan dalam proses selanjutnya.

```

1 import time
2 from selenium import webdriver
3 from selenium.webdriver.common.keys import Keys
4
5 browser = webdriver.Chrome()
6 url =
u'https://twitter.com/search?l=&q=anies%20R%20sandiaga%20near%3A%22Jawa%20
Tengah%22%20within%3A15mi%20since%3A2017-02-16%20until%3A2017-04-
17&src=typd'
7
8 browser.get(url)
9 time.sleep(1)
10 body = browser.find_element_by_tag_name('body')
11
12 for _ in range(1100):
13     body.send_keys(Keys.PAGE_DOWN)
14     time.sleep(0.2)
15
16 tweets = browser.find_elements_by_class_name('tweet-text')
17 users = browser.find_elements_by_class_name('username')
18
19 with open('anies-sandi.csv', 'w', encoding="utf-8") as file:
20     for tweet in tweets:
21         file.write(tweet.text.replace('\n', ' ').replace('; ', '
').replace(', ', ' '))
22         file.write('\n')
23
24 with open('username.csv', 'w', encoding="utf-8") as file:
25     for user in users:
26         file.write(user.text)
27         file.write('\n')

```

Gambar 5.1: Bagian program untuk melakukan proses *scraping* pada Twitter

Data tweet yang telah terkumpul kemudian disimpan dalam suatu file CSV. Karakter *new space*, titik koma, dan koma yang terdapat pada tweet dihapus pada saat penyimpanan untuk menghindari terjadinya perbedaan data. Hal ini

ditunjukkan pada baris 21 Gambar 5.1. Data nama pengguna yang terkumpul juga disimpan pada suatu file CSV ditunjukkan pada baris 24.

Data profil pengguna Twitter diperlukan dalam melakukan proses klasifikasi *buzzer*. Proses pengambilan data tersebut diambil dengan menggunakan API Twitter dengan bantuan *library* twitter di Python. Pada baris 3 Gambar 5.2, objek API Twitter dibuat dengan memanggil *class* Twitter yang berparameter variabel ‘auth’. Variabel ‘auth’ berfungsi dalam proses autentikasi dan dibuat dengan memanggil *class* OAuth() yang berisikan parameter sebagai berikut : *access_key*, *access_secret*, *consumer_key*, dan *consumer_secret*. Keempat parameter ini didapat dengan mendaftarkan aplikasi yang dibuat ke situs <https://apps.twitter.com>. Hasilnya kemudian dimasukkan ke dalam variabel ‘twitter’ seperti yang ditunjukkan pada baris 3.

```
1 from twitter import Twitter, OAuth
2
3 twitter = Twitter(
4     auth = OAuth("211872504-
EZAtA4sS1m00lOLN4kxbHp06mmq6mc9EV5yKPdmL",
5                 "1F1lCzeiuC5NQYmG0iJEQ2lOhp1AF8j1FIj4Ueq0svMp5",
6                 "1gE3hvWxThRITSEZ5reYqKF5M",
7                 "2LaGGD8FqS5TWf1Jp5JksUvVCEMtp9omz1O3tjFGJy68t5NaDp"))
```

Gambar 5.2: Bagian program untuk membuat objek API Twitter

Selanjutnya, seluruh data tweet baik data latih maupun data tes, diimpor ke dalam variabel ‘dataset’ seperti yang ditunjukkan pada baris 6 Gambar 5.3. Untuk setiap elemen pada variabel ‘dataset’ dilakukan perulangan, di mana pada setiap iterasinya fungsi **users.lookup()** dipanggil pada variabel ‘twitter’ dengan parameter *username* akun yang ingin dicari. *Username* tersebut didapat dari variabel array ‘row’ kolom 0 dan hasilnya kemudian dimasukkan ke dalam variabel ‘results’ di mana variabel ini memuat *dictionary* berukuran 42 data yang memuat informasi-informasi pengguna. Beberapa dari informasi tersebut yang nantinya digunakan dalam proses klasifikasi *buzzer* adalah umur akun, jumlah mengikuti, dan jumlah pengikut. Masih dalam perulangan sebelumnya, pada baris 13, dilakukan perulangan lagi untuk setiap elemen pada variabel ‘results’, di mana pada

setiap iterasinya, informasi tanggal pembuatan akun di-*parsing* ke format `datetime` menggunakan fungsi `dateutil.parser.parse()` sehingga dapat digunakan untuk menghitung umur akun. Pada baris 15, dengan menggunakan fungsi `relativedelta()` yang didapat dari *library* `dateutil`, umur akun dihitung dengan memasukkan parameter tanggal pembuatan akun dan tanggal pilkada yang telah didefinisikan sebelumnya pada baris 8. Hasilnya kemudian dimasukkan ke dalam variabel `'diff'`. Informasi-informasi yang dibutuhkan seperti, `uesrname`, umur akun, jumlah mengikuti, dan jumlah pengikut dimasukkan ke dalam variabel `'user_profile'` seperti yang ditunjukkan pada baris 16. Variabel `'user_profile'` tersebut kemudian dikumpulkan ke dalam variabel array `'profile_data'` untuk disimpan dan digunakan digunakan dalam proses selanjutnya. Proses ini ditunjukkan pada baris 17.

```

1 import pandas as pd
2 from datetime import datetime
3 import dateutil.parser
4 from dateutil.relativedelta import relativedelta
5
6 dataset = pd.read_csv('ahok_dataset.csv', header=None)
7 profile_data = []
8 electionDate = datetime(2017, 4, 19)
9
10 for index, row in dataset.iterrows():
11     results = twitter.users.lookup(screen_name = row[0])
12
13     for user in results:
14         accountDate =
dateutil.parser.parse(user["created_at"]).replace(tzinfo=None)
15         diff = relativedelta(electionDate, accountDate)
16         user_profile =
str(user["screen_name"])+" "+str(diff.years*12+diff.months)+" "+str(user["f
riends_count"])+" "+str(user["followers_count"])
17         profile_data.append(user_profile)

```

Gambar 5.3: Bagian program untuk melakukan proses *look up* data profil pengguna Twitter

Data informasi pengguna yang telah terkumpul kemudian disimpan dalam suatu file CSV agar dapat digunakan dalam proses *buzzer detection*. Untuk setiap elemen pada variabel array `'profile_data'` ditulis ke dalam file CSV di mana antar

elemen dipisahkan dengan karakter *newline*. Proses ini ditunjukkan pada Gambar 5.4.

```
1 with open('profile_data.csv', 'w', encoding="utf-8") as file:
2     for user_profile in profile_data:
3         file.write(user_profile)
4         file.write('\n')
```

Gambar 5.4: Bagian program untuk menyimpan data profil pengguna Twitter

5.2.2 Pre-processing Data

Tahap pertama pada proses pre-processing adalah *case folding*, yaitu proses mengubah seluruh teks tweet menjadi *lowercase*. Proses ini ditunjukkan pada baris 3 Gambar 5.5 dengan menggunakan perintah `tweet.lower()`. Tweet yang telah menjadi *lowercase* kemudian disimpan pada variabel '*preprocessed_tweets*' kolom '*case*' yang berbentuk Pandas DataFrame.

```
1 import pandas as pd
2 preprocessed_tweets = pd.DataFrame()
3 for tweet in tweets:
4     tweet = tweet.lower()
5     preprocessed_tweets['case'].append(tweet)
```

Gambar 5.5: Bagian program untuk melakukan *case folding*

Setelah melakukan *case folding*, tahap selanjutnya adalah *URLs removal*. Setiap tweet pada variabel DataFrame '*preprocessed_tweets*' kolom '*case*' diiterasi pada baris 1 Gambar 5.6 dan dihilangkan *Uniform Resource Locator* (URL)-nya pada baris 2. Sebuah URL ditandai dengan adanya string '*http*' pada bagian awal URL tersebut. Tweet tersebut kemudian disimpan pada variabel DataFrame '*preprocessed_tweets*' kolom '*url*' ditunjukkan pada baris 3.

```
1 for tweet in preprocessed_tweets['case']:
2     tweet = re.sub(r"http\S+", "", tweet)
3     preprocessed_tweets['url'].append(tweet)
```

Gambar 5.6: Bagian program untuk *URLs removal*

Pada proses *brackets removal*, tanda kurung (baik kurung biasa maupun kurung siku) beserta isinya dihapus dari tweet. Proses penghapusan ditunjukkan pada baris 2 Gambar 5.7. Tweet yang telah dilakukan *brackets removal* kemudian disimpan pada variabel ‘*preprocessed_tweets*’ kolom ‘*bracket*’ yang berbentuk DataFrame.

```
1 for tweet in preprocessed_tweets['url']:
2     tweet = re.sub(r"[\(\[\].*?\]\)]", "", tweet)
3     preprocessed_tweets['bracket'].append(tweet)
```

Gambar 5.7: Bagian program untuk *brackets removal*

Proses berikutnya adalah *mentions removal*. Sebuah *mention* ditandai dengan adanya karakter ‘@’ di bagian awal *mention*. Proses ini ditunjukkan pada baris 2 Gambar 5.8. Tweet yang telah dihapus *mentions*-nya kemudian disimpan pada variabel ‘*preprocessed_tweets*’ kolom ‘*mention*’ yang berbentuk DataFrame.

```
1 for tweet in preprocessed_tweets['bracket']:
2     tweet = re.sub(r"@S+", "", tweet)
3     preprocessed_tweets['mention'].append(tweet)
```

Gambar 5.8: Bagian program untuk *mentions removal*

Setelah melakukan *mentions removal*, proses selanjutnya adalah *non-alphanumeric characters removal*. Setiap tweet pada variabel DataFrame ‘*processed_tweets*’ kolom ‘*mention*’ diiterasi pada baris 1 Gambar 5.9 dan dihilangkan karakter-karakter yang bukan *alphanumeric*-nya pada baris 2. Karakter-karakter yang bukan *alphanumeric* dapat berupa tanda baca, seperti koma, titik, tanda tanya, tanda seru, dan yang lainnya. Tweet tersebut kemudian disimpan pada variabel DataFrame ‘*preprocessed_tweets*’ kolom ‘*non_alphanumeric*’ ditunjukkan pada baris 3.

```
1 for tweet in preprocessed_tweets['mention']:
2     tweet = re.sub(r"[^A-Za-z0-9]+", "", tweet)
3     preprocessed_tweets['non_alphanumeric'].append(tweet)
```

Gambar 5.9: Bagian program untuk *non-alphanumeric characters removal*

Proses selanjutnya adalah *RT removal*. Dalam Twitter, *retweet* berfungsi untuk mengulang kembali tweet yang telah ada agar dapat dibagikan kembali kepada pengikut-pengikut yang ada di Twitter. Sebuah *retweet* ditandai dengan adanya string 'rt' di dalam suatu tweet. Proses penghapusan string 'rt' ditunjukkan pada baris 2 Gambar 5.10. Tweet yang telah dihapus *retweet*-nya kemudian disimpan pada variabel '*preprocessed_tweets*' kolom 'rt' yang berbentuk DataFrame.

```
1 for tweet in preprocessed_tweets['non_alphanumeric']:
2     tweet = re.sub(r"rt", " ", tweet)
3     preprocessed_tweets['rt'].append(tweet)
```

Gambar 5.10: Bagian program untuk *rt removal*

Pada proses *stopwords removal*, variabel array '*tweets*' diiterasi untuk setiap elemennya seperti yang ditunjukkan pada baris 9 Gambar 5.11. Pada baris 10, fungsi **word_tokenize()** yang ada pada *library* nltk digunakan untuk melakukan proses tokenisasi pada kalimat tweet menjadi bentuk token. Hasilnya kemudian dimasukkan ke dalam variabel '*word_tokens*'. Selanjutnya, variabel array '*word_tokens*' diiterasi untuk setiap elemennya seperti yang ditunjukkan pada baris 13. Jika suatu elemen pada variabel array '*word_tokens*' tidak ada pada variabel '*sstop_words*', maka elemen tersebut dimasukkan ke dalam variabel array '*filtered_sentence*'. Proses ini ditunjukkan pada baris 14 dan 15. Variabel '*stop_words*' Bahasa Indonesia yang digunakan didefinisikan pada baris 6. Daftar *stopword* Bahasa Indonesia dapat dilihat pada halaman lampiran. Hasilnya kemudian didetokenisasi menggunakan fungsi **detokenize()** yang berasal dari *class MosesDetokenizer()* *library* nltk hingga menjadi satu kalimat utuh. Hasilnya kemudian dimasukkan ke dalam variabel array '*sentences*'. Hal ini ditunjukkan pada baris 17 dan 18.

```

1 from nltk.corpus import stopwords
2 from nltk.tokenize import word_tokenize
3 from nltk.tokenize.moses import MosesDetokenizer
4
5 detokenizer = MosesDetokenizer()
6 stop_words = set(stopwords.words('indonesian'))
7 sentences = []
8
9 for tweet in tweets:
10     word_tokens = word_tokenize(tweet)
11     filtered_sentence = []
12
13     for w in word_tokens:
14         if w not in stop_words:
15             filtered_sentence.append(w)
16
17     sentence = detokenizer.detokenize(filtered_sentence,
return_str=True)
18     sentences.append(sentence)

```

Gambar 5.11: Bagian program untuk *stopwords removal*

Data yang telah dilakukan pre-processing kemudian disimpan dalam suatu file CSV agar dapat digunakan dalam proses selanjutnya.

5.2.3 Pelatihan Model Klasifikasi Sentimen

Data latih dan data uji yang telah tersimpan sebelumnya kemudian diimpor ke dalam variabel ‘*train_tweets*’, ‘*classlabels*’ dan ‘*test_tweets*’. Variabel ‘*train_tweets*’ berisikan tweet-tweet pada data latih, variabel ‘*classlabels*’ berisikan label-label kelas pada tiap tweet data latih, sedangkan variabel ‘*test_tweets*’ berisikan data tes yang akan diklasifikasi sentimennya. Proses ini ditunjukkan pada Gambar 5.12.

```

1 import csv
2
3 with open('anies-trainset.csv', encoding='utf-8') as csvfile:
4     readCSV = csv.reader(csvfile, delimiter=',')
5
6     train_tweets = []
7     classlabels = []
8
9     for row in readCSV:
10         train_tweet = row[0]
11         classlabel = row[1]
12         train_tweets.append(train_tweet)
13         classlabels.append(classlabel)
14
15 with open('anies-testset.csv', encoding='utf-8') as csvfile:
16     readCSV = csv.reader(csvfile, delimiter=',')
17
18     test_tweets = []
19
20     for row in readCSV:
21         test_tweet = row[0]
22         test_tweets.append(test_tweet)

```

Gambar 5.12: Bagian program untuk mengimpor data latihan dan data tes

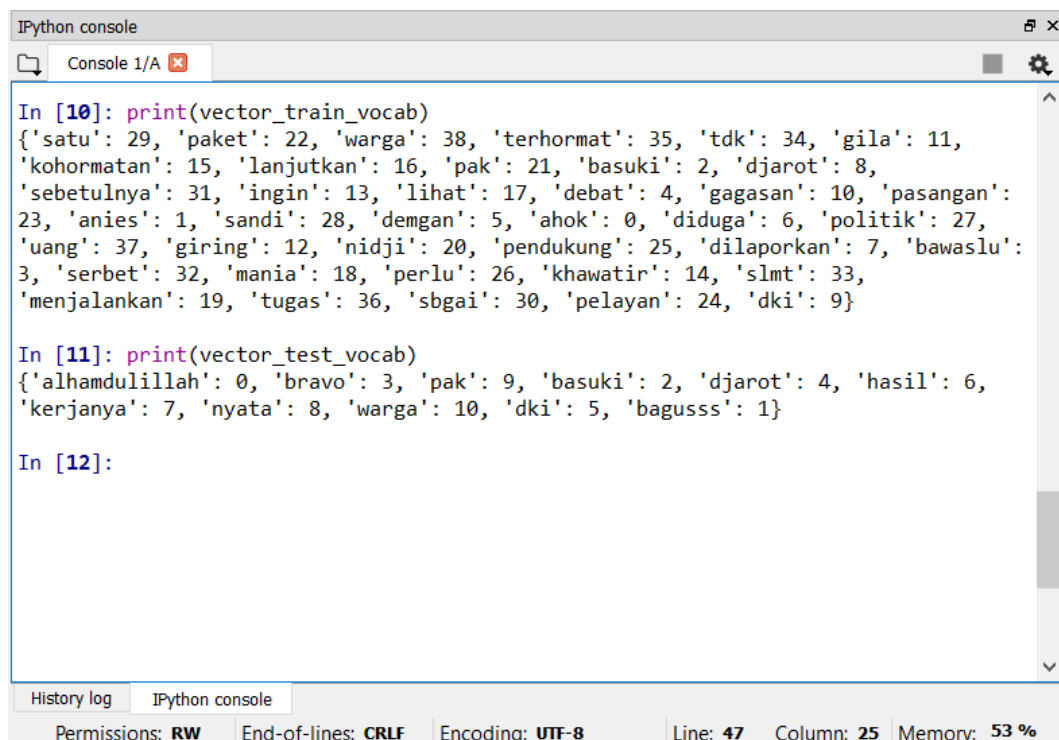
Selanjutnya, data latihan dan data tes yang telah diimpor ke dalam variabel ‘*train_tweets*’ dan ‘*test_tweets*’ dikonversikan ke bentuk matriks sesuai dengan jumlah tokennya. Proses ini berfungsi sebagai ekstraksi fitur data latihan agar data berubah ke bentuk vektornya sehingga dapat digunakan pada algoritma klasifikasi. Proses vektorisasi ini menggunakan *class CountVectorizer()* yang ada pada *library* scikit learn ditunjukkan pada baris 6 dan 7 Gambar 5.13. Hasilnya kemudian ditampilkan dalam bentuk kosakata atau *vocabulary* bertipe data *dictionary* pada baris 8 dan 9. Proses pembentukan kosakata didasarkan pada model Bag of Words. Contoh kosakata yang dihasilkan dapat dilihat pada Gambar 5.14. Dengan menggunakan fungsi **transform()**, pada baris 12 dan 13 Gambar 5.13, vektor data latihan dan data tes diubah ke bentuk array yang letak masing-masing token pada tiap elemen array sesuai dengan kosakata dan berisikan jumlah kemunculan token pada tweet tersebut. Contoh array hasil transformasi tersebut dapat dilihat pada Gambar 5.15.

```

1 from sklearn.feature_extraction.text import CountVectorizer
2
3 vectorizer_train = CountVectorizer()
4 vectorizer_test = CountVectorizer()
5
6 vectorizer_train.fit(train_tweets)
7 vectorizer_test.fit(test_tweets)
8
9 vector_train_vocab = vectorizer_train.vocabulary_
10 vector_test_vocab = vectorizer_test.vocabulary_
11
12 vector_train = vectorizer_train.transform(train_tweets)
13 vector_test = vectorizer_test.transform(test_tweets)
14
15 vector_train_array = vector_train.toarray()
16 vector_test_array = vector_test.toarray()
17
18 x1, y1 = vector_train.shape
19 x, y = vector_test.shape

```

Gambar 5.13: Bagian program untuk ekstraksi fitur



```

IPython console
Console 1/A

In [10]: print(vector_train_vocab)
{'satu': 29, 'paket': 22, 'warga': 38, 'terhormat': 35, 'tdk': 34, 'gila': 11,
'kohormatan': 15, 'lanjutkan': 16, 'pak': 21, 'basuki': 2, 'djarot': 8,
'sebetulnya': 31, 'ingin': 13, 'lihat': 17, 'debat': 4, 'gagasan': 10, 'pasangan':
23, 'anies': 1, 'sandi': 28, 'demgan': 5, 'ahok': 0, 'diduga': 6, 'politik': 27,
'uang': 37, 'giring': 12, 'nidji': 20, 'pendukung': 25, 'dilaporkan': 7, 'bawaslu':
3, 'serbet': 32, 'mania': 18, 'perlu': 26, 'khawatir': 14, 'slmt': 33,
'menjalankan': 19, 'tugas': 36, 'sbgai': 30, 'pelayan': 24, 'dki': 9}

In [11]: print(vector_test_vocab)
{'alhamdulillah': 0, 'bravo': 3, 'pak': 9, 'basuki': 2, 'djarot': 4, 'hasil': 6,
'kerjanya': 7, 'nyata': 8, 'warga': 10, 'dki': 5, 'bagusss': 1}

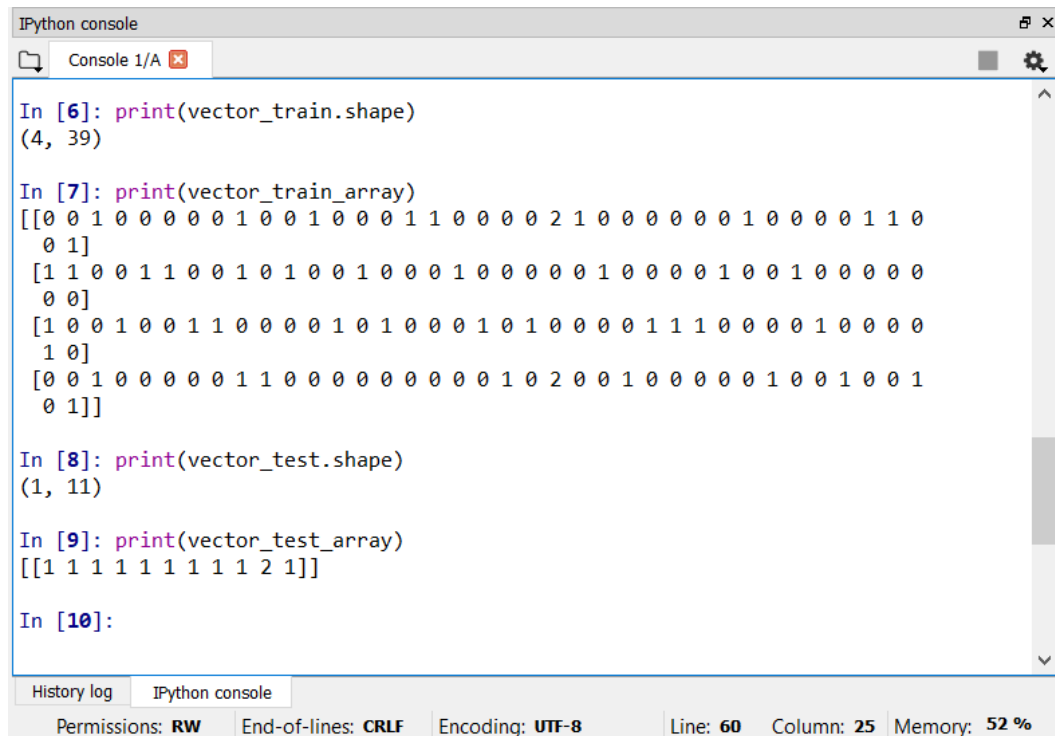
In [12]:

```

History log | IPython console

Permissions: RW | End-of-lines: CRLF | Encoding: UTF-8 | Line: 47 | Column: 25 | Memory: 53 %

Gambar 5.14: Contoh kosakata data latih dan data tes



```
IPython console
Console 1/A

In [6]: print(vector_train.shape)
(4, 39)

In [7]: print(vector_train_array)
[[0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 1 1 0 0 0 0 2 1 0 0 0 0 0 0 1 0 0 0 0 1 1 0
 0 1]
 [1 1 0 0 1 1 0 0 1 0 1 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 0
 0 0]
 [1 0 0 1 0 0 1 1 0 0 0 0 1 0 1 0 0 0 1 0 1 0 0 0 0 1 1 1 0 0 0 0 1 0 0 0 0
 1 0]
 [0 0 1 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 0 2 0 0 1 0 0 0 0 0 1 0 0 1 0 0 1
 0 1]]

In [8]: print(vector_test.shape)
(1, 11)

In [9]: print(vector_test_array)
[[1 1 1 1 1 1 1 1 2 1]]

In [10]:

History log  IPython console
Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 60 Column: 25 Memory: 52 %
```

Gambar 5.15: Contoh hasil transformasi vektor dalam bentuk array

Algoritma Multinomial Naive Bayes digunakan dalam melakukan klasifikasi sentimen. Oleh karena itu, diperlukan perhitungan probabilitas *prior* yang didasarkan pada persamaan (4.1). Proses perhitungan probabilitas *prior* ditunjukkan pada Gambar 5.16. Terdapat 3 jenis probabilitas *prior* yang akan dihitung sesuai dengan label kelas masing-masing, yakni positif, negatif, dan netral. Probabilitas *prior* positif didapatkan dengan membagi jumlah data latih berkelas positif dengan jumlah seluruh data latih seperti ditunjukkan pada baris 12. Probabilitas *prior* netral didapatkan dengan membagi jumlah data latih berkelas netral dengan jumlah seluruh data latih. Hal ini ditunjukkan pada baris 13. Probabilitas *prior* negatif dihitung dengan membagi jumlah data latih berkelas negatif dengan jumlah seluruh data latih seperti yang ditunjukkan pada baris 14.

```

1 npos = 0
2 nneg = 0
3 nnet = 0
4 for classlabel in classlabels:
5     if classlabel == "pos":
6         npos += 1
7     elif classlabel == "neg":
8         nneg += 1
9     elif classlabel == "net":
10        nnet += 1
11
12 ppos = npos/(npos+nneg+nnet)
13 pneg = nneg/(npos+nneg+nnet)
14 pnet = nnet/(npos+nneg+nnet)

```

Gambar 5.16: Bagian program untuk menghitung probabilitas *prior*

Selain perhitungan probabilitas *prior*, algoritma Multinomial Naive Bayes juga memerlukan perhitungan probabilitas kondisional. Proses perhitungan probabilitas kondisional terletak pada fungsi `conditional_probability()`. Fungsi `search_and_count()` yang terletak pada fungsi tersebut digunakan untuk menghitung . Hasilnya kemudian dimasukkan ke dalam variabel '*count*' seperti yang ditunjukkan pada baris 3 Gambar 5.17. Fungsi `sorted()` digunakan untuk mengurutkan variabel *dictionary* '*vector_test_vocab*' berdasarkan jumlah kemunculan terkecil pada masing-masing elemen variabel tersebut. Hasilnya kemudian dimasukkan ke dalam variabel '*sorted_vector_test_vocabs*'. Proses *sorting* ditunjukkan pada baris 4 Gambar 5.17. Variabel '*sorted_vector_test_vocabs*' kemudian diiterasi untuk setiap elemennya. Pada setiap iterasi tersebut, variabel *dictionary* '*vector_train_vocabs*' juga diiterasi untuk setiap *item*nya seperti yang ditunjukkan pada baris 5 dan 6. Jika suatu elemen pada '*sorted_vector_test_vocabs*' sama dengan *key* pada '*vector_train_vocabs*', maka fungsi `search_and_count()` digunakan untuk menghitung . Hasilnya kemudian dimasukkan ke dalam variabel '*count2*' seperti yang ditunjukkan pada baris 8. Selain itu, probabilitas kondisional juga dihitung berdasarkan pada persamaan (4.2). Hal ini ditunjukkan pada baris 9 dengan menggunakan rumus $(count2 + 1) / (count + y1)$. Hasilnya kemudian dimasukkan ke dalam variabel array '*results*' ditunjukkan pada baris 10. Fungsi

conditional_probability() mengembalikan variabel *'results'* melalui *statement return* pada baris 11.

```
1 def conditional_probability(labels):
2     results = []
3     count = search_and_count(labels,make_slice[:])
4     sorted_vector_test_vocab = sorted(vector_test_vocab,
key=vector_test_vocab.__getitem__)
5     for sorted_vector_test_vocab in sorted_vector_test_vocab:
6         for key2, value2 in vector_train_vocab.items():
7             if sorted_vector_test_vocab == key2:
8                 count2 = search_and_count(labels,value2)
9                 calc = (count2 + 1)/(count + y1)
10                results.append(calc)
11     return results
```

Gambar 5.17: Bagian program untuk menghitung probabilitas kondisional

Pada fungsi **search_and_count()**, label yang sama akan dicari untuk setiap elemen dengan nilai tertentu pada variabel array *'classlabels'*. Hal ini dilakukan dengan mengiterasi variabel *'classlabels'* pada baris 3 Gambar 5.18. Ketika label yang ingin dicari sama dengan elemen pada variabel array *'classlabels'*, maka fungsi **numpy.sum()** digunakan untuk menjumlahkan setiap elemen pada variabel array *'vector_train_array'* di mana barisnya sesuai dengan iterasi saat itu dan kolomnya berdasarkan *input* pengguna.

```
1 def search_and_count(labels, value):
2     count = 0
3     for position, classlabel in enumerate(classlabels):
4         if labels == classlabel:
5             count += np.sum(vector_train_array[position,value])
6     return count
```

Gambar 5.18: Bagian program untuk mencari setiap elemen yang sama dengan suatu nilai tertentu pada variabel *'classlabel'*

Fungsi **find_words_that_only_in_the_vocab()** berfungsi untuk mencari indeks kata-kata pada data tes yang hanya terdapat di *vocabulary*. Fungsi **sorted()** digunakan untuk mengurutkan variabel *dictionary* *'vector_test_vocab'* berdasarkan jumlah kemunculan terkecil pada masing-masing elemen variabel

tersebut. Hasilnya kemudian dimasukkan ke dalam variabel `'sorted_vector_test_vocabs'`. Proses *sorting* ditunjukkan pada baris 4 Gambar 5.19. Variabel `'sorted_vector_test_vocabs'` kemudian diiterasi untuk setiap elemennya. Pada setiap iterasi tersebut, variabel *dictionary* `'vector_train_vocabs'` juga diiterasi untuk setiap *item*nya seperti yang ditunjukkan pada baris 5 dan 6. Jika suatu elemen pada `'sorted_vector_test_vocabs'` sama dengan *key* pada `'vector_train_vocabs'`, maka elemen `'sorted_vector_test_vocabs'` yang sama tersebut dimasukkan ke dalam variabel array `'results'`. Hal ini ditunjukkan pada baris 7 dan 8. Ketika seluruh iterasi sebelumnya telah selesai, dilakukan perulangan untuk setiap elemen pada variabel `'results'`. Untuk setiap iterasi dalam perulangan, variabel *dictionary* `'vector_test_vocabs'` juga diiterasi untuk setiap *item*-nya seperti yang ditunjukkan pada baris 9 dan 10. Jika suatu elemen pada `'results'` sama dengan *key* pada `'vector_test_vocabs'`, maka *value* dari `'vector_test_vocabs'` yang sama tersebut dimasukkan ke dalam variabel array `'outputs'`. Proses ini ditunjukkan pada baris 11 dan 12. Fungsi `find_words_that_only_in_the_vocab()` mengembalikan variabel `'outputs'` melalui *statement return* pada baris 13.

```

1 def find_words_that_only_in_the_vocab():
2     results = []
3     outputs = []
4     sorted_vector_test_vocabs = sorted(vector_test_vocab,
key=vector_test_vocab.__getitem__)
5     for sorted_vector_test_vocab in sorted_vector_test_vocabs:
6         for key2, value2 in vector_train_vocab.items():
7             if sorted_vector_test_vocab == key2:
8                 results.append(sorted_vector_test_vocab)
9     for result in results:
10        for key2, value2 in vector_test_vocab.items():
11            if result == key2:
12                outputs.append(value2)
13    return outputs

```

Gambar 5.19: Bagian program untuk mencari indeks kata-kata pada data tes yang hanya terdapat di *vocabulary*

Pada proses pengklasifikasian sentimen, fungsi `find_words_that_only_in_the_vocab()` dipanggil untuk mencari indeks kata-kata pada data tes yang hanya terdapat di *vocabulary*. Hasilnya kemudian

dimasukkan ke dalam variabel `'idx_words_in_the_vocab'` seperti yang ditunjukkan pada baris 4 Gambar 5.20. Selanjutnya, fungsi `conditional_probability()` dipanggil dengan parameter `'pos'` untuk menghitung probabilitas kondisional pada kelas positif. Hasil keluarannya lalu diubah ke bentuk numpy array dengan menggunakan fungsi `np.array()` dan disimpan ke dalam variabel `'resultpos'` seperti pada baris 5. Selain pada kelas positif, fungsi `conditional_probability()` juga dipanggil dengan parameter `'neg'` untuk menghitung probabilitas kondisional pada kelas negatif. Hasil keluarannya lalu diubah ke bentuk numpy array dan dimasukkan ke dalam variabel `'resultneg'` seperti yang ditunjukkan pada baris 6. Fungsi `conditional_probability()` juga dipanggil dengan parameter `'net'` untuk menghitung probabilitas kondisional pada kelas netral. Hasil keluarannya diubah ke bentuk numpy array dan dimasukkan ke dalam variabel `'resultnet'`. Proses ini ditunjukkan pada baris 7. Untuk setiap elemen pada variabel array `'test_tweets'`, pada setiap iterasi perulangannya, variabel `'a'` didefinisikan sebagai numpy array yang berisikan elemen dari variabel array `'vector_test_array'` dengan *index* baris sesuai dengan iterasi perulangan dan *index* kolom sesuai dengan variabel `'idx_words_in_the_vocab'`. Hal ini ditunjukkan pada baris 8 dan 9. Variabel `'a'` adalah banyaknya kemunculan suatu kata pada data tes. Masih dalam perulangan, probabilitas suatu kelas pada suatu tweet dihitung dan didefinisikan sebagai variabel `'sum_pos'`, `'sum_neg'`, dan `'sum_net'`. Pada baris 10, untuk menghitung probabilitas kelas positif pada suatu tweet, variabel `'sum_pos'` didapat dengan menggunakan fungsi `numpy.prod()` pada variabel `'resultpos'` yang dipangkatkan dengan variabel `'a'`. Hasilnya kemudian dikalikan dengan probabilitas *prior* positif yang ditunjukkan oleh variabel `'ppos'`. Fungsi `numpy.prod()` berguna untuk menghitung hasil pengalihan antara elemen-elemen pada array. Selanjutnya pada baris 11, untuk menghitung probabilitas kelas negatif pada suatu tweet, variabel `'sum_neg'` didapat dengan menggunakan fungsi `numpy.prod()` pada variabel `'resultneg'` yang dipangkatkan dengan variabel `'a'`. Hasilnya kemudian dikalikan dengan probabilitas *prior* negatif yang ditunjukkan oleh variabel `'pneg'`. Untuk menghitung probabilitas kelas netral pada suatu tweet, variabel `'sum_net'` didapat dengan menggunakan fungsi `numpy.prod()` pada

variabel *'resultnet'* yang dipangkatkan dengan variabel *'a'*. Hasilnya kemudian dikalikan dengan probabilitas *prior* netral yang ditunjukkan oleh variabel *'pnet'*. Proses ini ditunjukkan pada baris 12. Jika probabilitas kelas positif pada suatu tweet lebih besar dibanding probabilitas kelas-kelas lainnya, maka tweet tersebut akan terklasifikasi sebagai positif seperti yang ditunjukkan pada baris 14. Jika probabilitas kelas negatif pada suatu tweet lebih besar dibanding probabilitas kelas-kelas lainnya, maka tweet tersebut akan terklasifikasi sebagai negatif. Hal ini ditunjukkan pada baris 15. Jika probabilitas kelas netral pada suatu tweet lebih besar dibanding probabilitas kelas-kelas lainnya, maka tweet tersebut akan terklasifikasi sebagai netral seperti yang ditunjukkan pada baris 16. Pada baris 24, jumlah masing-masing kelas positif, netral, dan negatif pada data tes yang telah terklasifikasi dicetak agar dapat dilihat oleh pengguna.

```

1 poscount = 0
2 negcount = 0
3 netcount = 0
4 idx_words_in_vocab = find_words_that_only_in_the_vocab()
5 resultpos = np.array(conditional_probability("pos"))
6 resultneg = np.array(conditional_probability("neg"))
7 resultnet = np.array(conditional_probability("net"))
8 for index, test_tweet in enumerate(test_tweets):
9     a = np.array(vector_test_array[index, idx_words_in_vocab])
10    sum_pos = np.prod(resultpos**a)*ppos
11    sum_neg = np.prod(resultneg**a)*pneg
12    sum_net = np.prod(resultnet**a)*pnet
13
14    if (sum_pos >= sum_neg) and (sum_pos >= sum_net):
15        print(str(index+1)+" || class: positive")
16        poscount += 1
17    elif (sum_neg > sum_pos) and (sum_neg > sum_net):
18        print(str(index+1)+" || class: negative")
19        negcount += 1
20    elif (sum_net > sum_pos) and (sum_net > sum_neg):
21        print(str(index+1)+" || class: neutral")
22        netcount += 1
23
24 print("POS = "+str(poscount)+" | NEG = "+str(negcount)+" | NET = "+str(netcount))

```

Gambar 5.20: Bagian utama program klasifikasi sentimen

5.2.4 Pelatihan Model Klasifikasi Buzzer

Data latih berisikan informasi akun yang telah tersimpan sebelumnya kemudian diimpor ke dalam variabel Pandas DataFrame ‘*df*’. Untuk setiap elemen pada kolom ‘*age*’, ‘*followings*’, dan ‘*followers*’ pada variabel ‘*df*’ dilakukan normalisasi agar pada proses perhitungan probabilitas menggunakan fungsi Gaussian, data pada kolom yang memiliki nilai kecil mampu mengimbangi data pada kolom yang memiliki nilai lebih besar. Normalisasi dihitung dengan melakukan proses pengurangan pada data di suatu baris dan kolom dengan nilai minimal pada kolom tersebut. Hasilnya kemudian dibagi dengan hasil pengurangan antara nilai maksimum kolom tersebut dengan nilai minimalnya. Proses ini dinamakan *Min-Max Normalization* dan ditunjukkan pada baris 4, 5, dan 6 Gambar 5.21. Variabel ‘*df*’ kemudian dibagi berdasarkan kelasnya menjadi dua variabel lain, yaitu ‘*df_buzzer*’ dan ‘*df_normal*’. Variabel ‘*df_buzzer*’ berisikan informasi pada akun *buzzer* sedangkan variabel ‘*df_normal*’ berisikan informasi pada akun pengguna normal. Proses ini ditunjukkan pada baris 8 dan 9.

```
1 import pandas as pd
2
3 df = pd.read_csv('buzz_sample_data_80.csv')
4 df['age'] = (df['age'] - df['age'].min()) / (df['age'].max() -
df['age'].min())
5 df['followings'] = (df['followings'] - df['followings'].min()) /
(df['followings'].max() - df['followings'].min())
6 df['followers'] = (df['followers'] - df['followers'].min()) /
(df['followers'].max() - df['followers'].min())
7
8 df_buzzer = df[df['class'] == "buzzer"]
9 df_normal = df[df['class'] == "normal"]
```

Gambar 5.21: Bagian program untuk melakukan normalisasi data *buzzer*

Untuk menghitung rata-rata umur akun *buzzer*, maka variabel ‘*df_buzzer*’ kolom ‘*age*’ dihitung *mean*/rata-ratanya dengan menggunakan fungsi **mean()** dari *library* pandas seperti yang ditunjukkan pada baris 1 Gambar 5.22. Hasilnya kemudian dimasukkan ke dalam variabel ‘*buzzer_age_mean*’. Sedangkan untuk untuk menghitung rata-rata umur akun pengguna normal, fungsi **mean()** digunakan pada variabel ‘*df_normal*’ kolom ‘*age*’. Hasilnya kemudian dimasukkan ke dalam variabel ‘*normal_age_mean*’. Proses ini ditunjukkan pada baris 2. Untuk

menghitung variansi umur akun *buzzer*, maka variabel '*df_buzzer*' kolom '*age*' dihitung variansinya dengan menggunakan fungsi **var()** dari *library* *pandas* seperti yang ditunjukkan pada baris 4. Hasilnya kemudian dimasukkan ke dalam variabel '*buzzer_age_var*'. Sedangkan untuk menghitung variansi umur akun pengguna normal, fungsi **var()** digunakan pada variabel '*df_normal*' kolom '*age*'. Hasilnya kemudian dimasukkan ke dalam variabel '*normal_age_var*'. Proses ini ditunjukkan pada baris 5.

Untuk menghitung rata-rata jumlah mengikuti (*following*) dari akun *buzzer*, maka variabel '*df_buzzer*' kolom '*followings*' dihitung *mean*/rata-ratanya dengan menggunakan fungsi **mean()** seperti yang ditunjukkan pada baris 7 Gambar 5.22. Hasilnya kemudian dimasukkan ke dalam variabel '*buzzer_followings_mean*'. Sedangkan untuk menghitung rata-rata jumlah *following* dari akun pengguna normal, fungsi **mean()** digunakan pada variabel '*df_normal*' kolom '*followings*'. Hasilnya kemudian dimasukkan ke dalam variabel '*normal_followings_mean*'. Proses ini ditunjukkan pada baris 8. Untuk menghitung variansi jumlah *following* dari akun *buzzer*, maka variabel '*df_buzzer*' kolom '*followings*' dihitung variansinya dengan menggunakan fungsi **var()** seperti yang ditunjukkan pada baris 10. Hasilnya kemudian dimasukkan ke dalam variabel '*buzzer_followings_var*'. Sedangkan untuk menghitung variansi jumlah *following* dari akun pengguna normal, fungsi **var()** digunakan pada variabel '*df_normal*' kolom '*followings*'. Hasilnya kemudian dimasukkan ke dalam variabel '*normal_followings_var*'. Proses ini ditunjukkan pada baris 11.

Untuk menghitung rata-rata jumlah pengikut (*follower*) dari akun *buzzer*, maka variabel '*df_buzzer*' kolom '*followers*' dihitung *mean*/rata-ratanya dengan menggunakan fungsi **mean()** seperti yang ditunjukkan pada baris 13 Gambar 5.22. Hasilnya kemudian dimasukkan ke dalam variabel '*buzzer_followers_mean*'. Sedangkan untuk menghitung rata-rata jumlah *follower* dari akun pengguna normal, fungsi **mean()** digunakan pada variabel '*df_normal*' kolom '*followers*'. Hasilnya kemudian dimasukkan ke dalam variabel '*normal_followers_mean*'. Proses ini ditunjukkan pada baris 14. Untuk menghitung variansi jumlah *follower*

dari akun *buzzer*, maka variabel '*df_buzzer*' kolom '*followers*' dihitung variansinya dengan menggunakan fungsi `var()` yang ditunjukkan pada baris 16. Hasilnya kemudian dimasukkan ke dalam variabel '*buzzer_followers_var*'. Sedangkan untuk untuk menghitung variansi jumlah *follower* dari akun pengguna normal, fungsi `var()` digunakan pada variabel '*df_normal*' kolom '*followers*'. Hasilnya kemudian dimasukkan ke dalam variabel '*normal_followers_var*'. Proses ini ditunjukkan pada baris 17.

```
1 buzzer_age_mean = df_buzzer["age"].mean()
2 normal_age_mean = df_normal["age"].mean()
3
4 buzzer_age_var = df_buzzer["age"].var()
5 normal_age_var = df_normal["age"].var()
6
7 buzzer_followings_mean = df_buzzer["followings"].mean()
8 normal_followings_mean = df_normal["followings"].mean()
9
10 buzzer_followings_var = df_buzzer["followings"].var()
11 normal_followings_var = df_normal["followings"].var()
12
13 buzzer_followers_mean = df_buzzer["followers"].mean()
14 normal_followers_mean = df_normal["followers"].mean()
15
16 buzzer_followers_var = df_buzzer["followers"].var()
17 normal_followers_var = df_normal["followers"].var()
```

Gambar 5.22: Bagian program untuk menghitung *mean* dan variansi untuk masing-masing fitur

Fungsi `gaussian()` berguna untuk menghitung fungsi *Gaussian* berdasarkan persamaan (4.13) seperti yang ditunjukkan oleh baris 2 Gambar 5.23. Hasilnya kemudian dimasukkan ke dalam variabel '*gauss*' dan dikembalikan melalui *statement return* pada baris 3.

```
1 from math import e, sqrt, pi
2
3 def gaussian(s,x,m):
4     gauss = 1/(sqrt(2*pi)*s)*e**(-0.5*(float(x-m)/s)**2)
5     return gauss
```

Gambar 5.23: Bagian program untuk menghitung fungsi *Gaussian*

Fungsi **classify()** digunakan untuk menghitung probabilitas suatu akun apakah sebuah *buzzer* atau pengguna normal. Untuk menghitung probabilitas suatu akun adalah *buzzer*, digunakan fungsi **gaussian()** dengan parameter '*buzzer_age_var*', '*age*', '*buzzer_age_mean*' lalu dikalikan dengan fungsi **gaussian()** berparameter '*buzzer_followings_var*', '*followings*', '*buzzer_followings_mean*' dan dikalikan lagi dengan fungsi **gaussian()** berparameter '*buzzer_followers_var*', '*followers*', '*buzzer_followers_mean*'. Hasilnya kemudian dimasukkan ke dalam variabel '*pbuzzer*' seperti yang ditunjukkan pada baris 4 Gambar 5.24. Untuk menghitung probabilitas suatu akun adalah pengguna normal, digunakan fungsi **gaussian()** dengan parameter '*normal_age_var*', '*age*', '*normal_age_mean*' lalu dikalikan dengan fungsi **gaussian()** berparameter '*normal_followings_var*', '*followings*', '*normal_followings_mean*' dan dikalikan lagi dengan fungsi **gaussian()** berparameter '*normal_followers_var*', '*followers*', '*normal_followers_mean*'. Hasilnya kemudian dimasukkan ke dalam variabel '*pnormal*' seperti yang ditunjukkan pada baris 13. Jika probabilitas suatu akun adalah *buzzer* lebih besar dibandingkan probabilitas suatu akun adalah pengguna normal, maka akun tersebut akan diklasifikasikan sebagai *buzzer*. Proses ini ditunjukkan pada baris 22. Akun yang terdeteksi sebagai *buzzer*, *username*nya dimasukkan ke dalam variabel array '*buzzers*' untuk selanjutnya digunakan dalam proses penghapusan tweet *buzzer*.

```

1 buzzers = []
2
3 def classify(username, age, followings, followers):
4     pbuzzer = gaussian(buzzer_age_var,
5                         age,
6                         buzzer_age_mean) *
7     gaussian(buzzer_followings_var,
8              followings,
9              buzzer_followings_mean) *
10    gaussian(buzzer_followers_var,
11             followers,
12             buzzer_followers_mean)
13    pnormal = gaussian(normal_age_var,
14                       age,
15                       normal_age_mean) *
16    gaussian(normal_followings_var,
17             followings,
18             normal_followings_mean) *
19    gaussian(normal_followers_var,
20             followers,
21             normal_followers_mean)
22    if pbuzzer >= pnormal:
23        print(str(username) + " : Buzzer")
24        buzzers.append(username)
25    else:
26        print(str(username) + " : Normal user")

```

Gambar 5.24: Bagian program untuk melakukan klasifikasi *buzzer*

Pada proses pengklasifikasian *buzzer*, data tes diimpor ke dalam variabel Pandas DataFrame '*df_test*' seperti yang ditunjukkan pada baris 1 Gambar 5.25. Untuk setiap elemen pada variabel '*df_test*', pada setiap iterasi perulangannya, fungsi **classify()** dengan parameter '*username*', '*age*', '*followings*', '*followers*' digunakan untuk menentukan kelas masing-masing akun. Hal ini ditunjukkan pada baris 3 dan 4.

```

1 df_test = pd.read_csv('buzz_test.csv')
2
3 for index, row in df_test.iterrows():
4     classify(row['username'],
5             row['age'],
6             row['followings'],
7             row['followers'])

```

Gambar 5.25: Bagian utama program klasifikasi *buzzer*

Variabel array ‘*buzzers*’ kemudian disimpan dalam suatu file CSV agar dapat digunakan dalam proses *buzzers’ tweets removal* seperti yang ditunjukkan pada Gambar 5.26.

```
1 with open('buzzers_list.csv', 'w', encoding="utf-8") as file:
2     for buzzer in buzzers:
3         file.write(buzzer)
4         file.write('\n')
```

Gambar 5.26: Bagian program untuk menyimpan daftar buzzer yang terklasifikasi

5.2.5 Penghapusan Tweet Buzzer

Dalam proses penghapusan tweet buzzer, daftar akun yang terklasifikasi sebagai *buzzer*, diimpor ke dalam variabel Pandas DataFrame bernama ‘*buzzers_list*’ seperti yang ditunjukkan pada baris 3 Gambar 5.27. Data tes yang akan dihapus tweet buzzernya diimpor dan dimasukkan ke dalam variabel ‘*test_tweets*’. Proses ini ditunjukkan oleh baris 4. Data latih yang akan dihapus tweet buzzernya juga diimpor dan dimasukkan ke dalam variabel ‘*train_tweets*’ seperti yang ditunjukkan oleh baris 5.

```
1 import pandas as pd
2
3 buzzers_list = pd.read_csv('buzzers_list.csv', header=None)
4 test_tweets = pd.read_csv('test_tweets.csv', header=None)
5 train_tweets = pd.read_csv('train_tweets.csv', header=None)
```

Gambar 5.27: Bagian program untuk mengimpor data dalam proses penghapusan tweet *buzzer*

Untuk setiap elemen pada variabel ‘*buzzers_list*’, dilakukan perulangan di mana pada setiap iterasinya apabila *username* pada variabel ‘*test_tweets*’ tidak sama dengan *username* pada variabel ‘*buzzers_list*’, maka data tersebut tetap disimpan pada variabel ‘*test_tweets*’. Hal ini ditunjukkan pada Gambar 5.28 baris 2. Masih dalam perulangan sebelumnya, apabila *username* pada variabel ‘*train_tweets*’ tidak sama dengan *username* pada variabel ‘*buzzers_list*’, maka data tersebut tetap disimpan pada variabel ‘*train_tweets*’ seperti yang ditunjukkan pada baris 3.

```

1 for index, row in buzzers_list.iterrows():
2     test_tweets = test_tweets[test_tweets[0] != row[0]]
3     train_tweets = train_tweets[train_tweets[0] != row[0]]

```

Gambar 5.28: Bagian program untuk menghapus tweet-tweet yang dihasilkan oleh akun *buzzer*

Pada Gambar 5.29, data latih dan data tes yang berisikan tweet-tweet yang bersih dari *buzzer* disimpan ke dalam suatu file CSV agar dapat digunakan pada proses berikutnya.

```

1 with open('removed_buzzers_test_tweets.csv', 'w', encoding="utf-8") as
file:
2     for index, row in test_tweets.iterrows():
3         file.write(row[1])
4         file.write('\n')
5
6 with open('removed_buzzers_train_tweets.csv', 'w', encoding="utf-8") as
file:
7     for index, row in train_tweets.iterrows():
8         file.write(row[1])
9         file.write('\n')

```

Gambar 5.29: Bagian program untuk menyimpan tweet-tweet yang bersih dari *buzzer*

5.2.6 Pengujian Model Klasifikasi Buzzer

Pada proses pengujian model klasifikasi *buzzer*, digunakan *10 fold cross validation* untuk menghitung performa model yang digunakan. Hasil perhitungan performa nantinya berupa *confusion matrix*, nilai akurasi, nilai presisi, nilai *recall*, dan nilai *F1 score*. Pertama-tama, *library* pandas dan kode klasifikasi *buzzer* yang telah dibuat sebelumnya diimpor ke dalam program. Setelah itu, *file* CSV data latih yang digunakan diimpor ke dalam variabel DataFrame '*dataset*' seperti yang ditunjukkan pada baris 4 Gambar 5.30. Untuk setiap elemen pada kolom '*age*', '*followings*', dan '*followers*' pada variabel '*dataset*' dilakukan normalisasi agar pada proses perhitungan probabilitas menggunakan fungsi Gaussian, data pada kolom yang memiliki nilai kecil mampu mengimbangi data pada kolom yang memiliki nilai lebih besar. Normalisasi dihitung dengan melakukan proses pengurangan pada data di suatu baris dan kolom dengan nilai minimal pada kolom

tersebut. Hasilnya kemudian dibagi dengan hasil pengurangan antara nilai maksimum kolom tersebut dengan nilai minimalnya. Proses ini dinamakan *Min-Max Normalization* dan ditunjukkan pada baris 5, 6, dan 7. Selanjutnya pada baris 8, kumpulan data yang sudah dinormalisasi tersebut diacak dan dimasukkan ke dalam variabel `'dataset_shuffled'`.

```
1 import pandas as pd
2 import buzzer
3
4 dataset = pd.read_csv('buzz_sample_data_80.csv')
5 dataset['age'] = (dataset['age'] - dataset['age'].min()) /
  (dataset['age'].max() - dataset['age'].min())
6 dataset['followings'] = (dataset['followings'] -
  dataset['followings'].min()) / (dataset['followings'].max() -
  dataset['followings'].min())
7 dataset['followers'] = (dataset['followers'] -
  dataset['followers'].min()) / (dataset['followers'].max() -
  dataset['followers'].min())
8 dataset_shuffled = dataset.sample(frac=1).reset_index(drop=True)
```

Gambar 5.30: Bagian program untuk melakukan proses normalisasi data buzzer pada saat proses 10-fold cross validation

Selanjutnya pada baris 1 Gambar 5.31, didefinisikan fungsi `kfold()` dengan parameter `'k'` di mana `'k'` adalah jumlah *fold* pada proses *cross validation*. Panjang *dataset* dihitung dengan menggunakan fungsi `len()` pada variabel `'dataset_shuffled'` seperti yang ditunjukkan pada baris 2. Sedangkan banyaknya data tes pada setiap *fold* dihitung dengan membagi panjang *dataset* dengan jumlah *fold*. Hasilnya kemudian dimasukkan ke dalam variabel `'window_size'` ditunjukkan pada baris 3.

Pada baris 8 Gambar 5.31 terjadi proses perulangan sebanyak panjang *dataset* dan setiap nilai `'window_size'` di mana indeks iterasinya ditampung dalam variabel `'i'`. Data tes yang digunakan adalah variabel `'dataset_shuffled'` yang dipotong mulai dari baris ke `'i'` hingga ke baris `'i+window_size'` sedangkan data latih yang digunakan adalah semua data pada variabel `'dataset_shuffled'` yang tidak terdapat pada data tes. Proses ini ditunjukkan pada baris 9 dan 10. Proses klasifikasi dilakukan dengan memanggil kode klasifikasi *buzzer* yang sebelumnya telah diubah

ke bentuk fungsi. Fungsi yang digunakan adalah fungsi `buzzer.classify()` seperti yang ditunjukkan pada baris 12. Jika hasil klasifikasi menghasilkan kelas *'buzzer'* dan data tesnya juga memiliki kelas *'buzzer'*, maka nilai *True Positive* yang diwakili oleh variabel *'TP'* ditambah satu seperti yang ditunjukkan pada baris 15 dan 16. Jika hasil klasifikasi menghasilkan kelas *'normal'* dan data tesnya juga memiliki kelas *'normal'*, maka nilai *True Negative* yang diwakili oleh variabel *'TN'* ditambah satu. Hal ini ditunjukkan pada baris 17 dan 18. Jika hasil klasifikasi menghasilkan kelas *'normal'* tetapi data tesnya memiliki kelas *'buzzer'*, maka nilai *False Negative* yang diwakili oleh variabel *'FN'* ditambah satu seperti yang ditunjukkan pada baris 19 dan 20. Jika hasil klasifikasi menghasilkan kelas *'buzzer'* tetapi data tesnya memiliki kelas *'normal'*, maka nilai *False Positive* yang diwakili oleh variabel *'FP'* ditambah satu. Proses ini ditunjukkan pada baris 21 dan 22. Keempat variabel *'TP'*, *'TN'*, *'FN'*, dan *'FP'* kemudian dikembalikan dengan menggunakan *statement return*.

```

1 def kfold (k):
2     dataset_length = len(dataset_shuffled)
3     window_size = int(dataset_length/k)
4     TP = 0
5     TN = 0
6     FN = 0
7     FP = 0
8     for i in range(0,dataset_length>window_size):
9         test_data =
dataset_shuffled[i:i+window_size].reset_index(drop=True)
10         train_data =
dataset_shuffled[~dataset_shuffled.isin(test_data)].dropna()
11
12         results = buzzer.classify(train_data,test_data)
13
14         for index, row in test_data.iterrows():
15             if row["class"] == results[index] == "buzzer":
16                 TP+=1
17             elif row["class"] == results[index] == "normal":
18                 TN+=1
19             elif row["class"] == "buzzer" and results[index] ==
"normal":
20                 FN+=1
21             elif row["class"] == "normal" and results[index] ==
"buzzer":
22                 FP+=1
23
24         return TP, TN, FN, FP

```

Gambar 5.31: Bagian program untuk melakukan proses *k-fold cross validation* pada klasifikasi *buzzer*

Fungsi `kfold()` dipanggil dengan parameter '10' di mana untuk setiap hasil kembalannya dimasukkan ke dalam variabel 'TP', 'TN', 'FN', dan 'FP' yang masing-masing mewakili nilai *True Positive*, *True Negative*, *False Negative*, dan *False Positive*. Pada baris 3 Gambar 5.32, keempat variabel tersebut dimasukkan ke dalam variabel DataFrame '*confusion_matrix*' agar dapat ditampilkan dalam *confusion matrix* yang berbentuk tabel. Variabel '*confusion_matrix*' tersebut kemudian dicetak seperti yang ditunjukkan pada baris 6. Akurasi dihitung dengan membagi penjumlahan antara *True Positive* dan *True Negative* dengan hasil penjumlahan keempat variabel *confusion matrix*. Proses ini ditunjukkan pada baris 8. Untuk menghitung nilai presisi, *True Positive* dibagi dengan hasil penjumlahan antara *True Positive* dan *False Positive* seperti yang ditunjukkan pada baris 11.

Dikarenakan dalam menghitung nilai presisi ada kemungkinan terjadi pembagian dengan nol, maka agar tidak terjadi *error*, hal tersebut diatasi dengan menggunakan *exception* seperti yang ditunjukkan pada baris 12. Jika terjadi pembagian dengan nol, maka nilai presisi dianggap sebagai tak terhingga. Hal serupa juga dilakukan untuk menghitung nilai *recall* dan *F1 score*. Nilai *recall* dihitung dengan membagi *True Positive* dengan hasil penjumlahan antara *True Positive* dan *False Negative* sedangkan nilai *F1 score* dihitung dengan melakukan perkalian antara nilai presisi dan nilai *recall* dengan 2. Hasilnya kemudian dibagi dengan hasil penjumlahan antara presisi dan *recall*. Proses perhitungan nilai *recall* dan *F1 score* ditunjukkan pada baris 16 dan 21. Hasil perhitungan tersebut kemudian dicetak agar dapat ditampilkan pada pengguna.

```

1 TP,TN,FN,FP = kfold(10)
2
3 confusion_matrix = pd.DataFrame([[ 'buzzer', TP, FP], [ 'normal',
FN ,TN]], columns=[ '(predicted)', 'buzzer', 'normal'])
4 confusion_matrix = confusion_matrix.set_index('(predicted)')
5
6 print(confusion_matrix)
7
8 accuracy = (TP+TN)/(TP+TN+FN+FP)
9
10 try:
11     precision = TP/(TP+FP)
12 except ZeroDivisionError:
13     precision = float('Inf')
14
15 try:
16     recall = TP/(TP+FN)
17 except ZeroDivisionError:
18     recall = float('Inf')
19
20 try:
21     flscore = (2*precision*recall)/(precision+recall)
22 except ZeroDivisionError:
23     flscore = float('Inf')
24
25 print('Akurasi : '+str(accuracy*100)+"%")
26 print('Presisi : '+str(precision*100)+"%")
27 print('Recall : '+str(recall*100)+"%")
28 print('F1 Score : '+str(flscore*100)+"%")

```

Gambar 5.32: Bagian utama program pengujian model klasifikasi *buzzer*

5.2.7 Pengujian Model Klasifikasi Sentimen

Pada proses pengujian model klasifikasi sentimen, digunakan *10 fold cross validation* untuk menghitung performa model yang digunakan. Hasil perhitungan performa nantinya berupa *confusion matrix* dan nilai akurasi. Pertama-tama, *library* pandas dan kode klasifikasi sentimen yang telah dibuat sebelumnya diimpor ke dalam program. Setelah itu, *file* CSV data latih yang digunakan diimpor ke dalam variabel DataFrame '*dataset*' seperti yang ditunjukkan pada baris 4 Gambar 5.33. Pada baris 5, variabel '*dataset*' tersebut diacak dan dimasukkan ke dalam variabel baru yang bernama '*dataset_shuffled*'. Selanjutnya, variabel '*dataset_shuffled*' dibagi berdasarkan kolomnya menjadi dua variabel baru, yaitu variabel '*tweet*' dan variabel '*classlabel*'.

```
1 import pandas as pd
2 import multinomialnb
3
4 dataset = pd.read_csv('anies-trainset.csv', header=None)
5 dataset_shuffled = dataset.sample(frac=1).reset_index(drop=True)
6 tweet = dataset_shuffled.loc[:, [0]]
7 classlabel = dataset_shuffled.loc[:, [1]]
```

Gambar 5.33: Bagian program untuk melakukan pembagian data menjadi bagian tweet dan bagian label

Pada baris 6 Gambar 5.34 terjadi proses perulangan sebanyak panjang dataset dan setiap nilai '*window_size*' di mana indeks iterasinya ditampung dalam variabel '*i*'. Data tes yang digunakan adalah variabel '*dataset_shuffled*' yang dipotong mulai dari baris ke '*i*' hingga ke baris '*i+window_size*' sedangkan data latih yang digunakan adalah semua data pada variabel '*dataset_shuffled*' yang tidak terdapat pada data tes. Proses ini ditunjukkan pada baris 9 dan 10. Proses klasifikasi dilakukan dengan memanggil kode klasifikasi sentimen yang sebelumnya telah diubah ke bentuk fungsi. Fungsi yang digunakan adalah fungsi ***multinomialnb.classify()*** seperti yang ditunjukkan pada baris 16. Jika hasil klasifikasi menghasilkan kelas '*pos*' dan data tesnya juga memiliki kelas '*pos*', maka nilai variabel '*A*' ditambah satu seperti yang ditunjukkan pada baris 19 dan 20. Jika hasil klasifikasi menghasilkan kelas '*net*' dan data tesnya juga memiliki

kelas 'net', maka nilai variabel '*B*' ditambah satu. Hal ini ditunjukkan pada baris 21 dan 22. Jika hasil klasifikasi menghasilkan kelas 'neg' dan data tesnya juga memiliki kelas 'neg', maka nilai variabel '*C*' ditambah satu seperti yang ditunjukkan pada baris 23 dan 24. Jika hasil klasifikasi menghasilkan kelas 'pos' tetapi data tesnya memiliki kelas 'net', maka nilai variabel '*D*' ditambah satu seperti yang ditunjukkan pada baris 25 dan 26. Jika hasil klasifikasi menghasilkan kelas 'pos' tetapi data tesnya memiliki kelas 'neg', maka nilai variabel '*E*' ditambah satu seperti yang ditunjukkan pada baris 27 dan 28. Jika hasil klasifikasi menghasilkan kelas 'net' tetapi data tesnya memiliki kelas 'pos', maka nilai variabel '*F*' ditambah satu seperti yang ditunjukkan pada baris 29 dan 30. Jika hasil klasifikasi menghasilkan kelas 'net' tetapi data tesnya memiliki kelas 'neg', maka nilai variabel '*G*' ditambah satu seperti yang ditunjukkan pada baris 31 dan 32. Jika hasil klasifikasi menghasilkan kelas 'neg' tetapi data tesnya memiliki kelas 'pos', maka nilai variabel '*H*' ditambah satu seperti yang ditunjukkan pada baris 33 dan 34. Jika hasil klasifikasi menghasilkan kelas 'neg' tetapi data tesnya memiliki kelas 'net', maka nilai variabel '*I*' ditambah satu seperti yang ditunjukkan pada baris 35 dan 36. Kesembilan variabel '*A*', '*B*', '*C*', '*D*', '*E*', '*F*', '*G*', '*H*', dan '*I*' kemudian dikembalikan dengan menggunakan *statement return*.


```

1 def kfold (k):
2     dataset_length = len(dataset_shuffled)
3     window_size = int(dataset_length/k)
4     A,B,C,D,E,F,G,H,I = 0,0,0,0,0,0,0,0,0
5
6     for i in range(0,dataset_length>window_size):
7         test_data_tweet = tweet[i:i+window_size].reset_index(drop=True)
8         test_data_label =
classlabel[i:i+window_size].reset_index(drop=True)
9         train_data_tweet = tweet[~tweet.isin(test_data_tweet)].dropna()
10        train_data_label =
classlabel[~classlabel.isin(test_data_label)].dropna()
11        test_data_tweet2 = test_data_tweet[0].values.tolist()
12        test_data_label2 = test_data_label[1].values.tolist()
13        train_data_tweet2 = train_data_tweet[0].values.tolist()
14        train_data_label2 = train_data_label[1].values.tolist()
15
16        results =
multinomialnb.classify(train_data_tweet2,train_data_label2,test_data_tweet2
)
17
18        for index, row in test_data_tweet.iterrows():
19            if test_data_label2[index] == results[index] == "pos":
20                A+=1
21            elif test_data_label2[index] == results[index] == "net":
22                B+=1
23            elif test_data_label2[index] == results[index] == "neg":
24                C+=1
25            elif test_data_label2[index] == "pos" and results[index] ==
"net":
26                D+=1
27            elif test_data_label2[index] == "pos" and results[index] ==
"net":
28                E+=1
29            elif test_data_label2[index] == "net" and results[index] ==
"pos":
30                F+=1
31            elif test_data_label2[index] == "net" and results[index] ==
"neg":
32                G+=1
33            elif test_data_label2[index] == "neg" and results[index] ==
"pos":
34                H+=1
35            elif test_data_label2[index] == "neg" and results[index] ==
"net":
36                I+=1
37
38        return A,B,C,D,E,F,G,H,I

```

Gambar 5.34: Bagian program untuk melakukan proses *k-fold cross validation* pada klasifikasi sentimen

Fungsi `kfold()` dipanggil dengan parameter '10' di mana untuk setiap hasil kembaliannya dimasukkan ke dalam variabel 'A', 'B', 'C', dan 'FP' yang masing-masing mewakili sembilan nilai yang terdapat pada *confusion matrix*. Pada baris 3 Gambar 5.35, kesembilan variabel tersebut dimasukkan ke dalam variabel DataFrame '*confusion_matrix*' agar dapat ditampilkan dalam *confusion matrix* yang berbentuk tabel. Variabel '*confusion_matrix*' tersebut kemudian dicetak seperti yang ditunjukkan pada baris 6. Akurasi dihitung dengan membagi penjumlahan antara A, B dan C dengan hasil penjumlahan kesembilan variabel *confusion matrix*. Proses ini ditunjukkan pada baris 8. Hasil perhitungan tersebut kemudian dicetak agar dapat ditampilkan pada pengguna.

```
1 A,B,C,D,E,F,G,H,I = kfold(10)
2
3 confusion_matrix = pd.DataFrame([[ 'pos', A, D, E], [ 'net', F, B, G],
4 [ 'neg', H, I, C]], columns=[ ' (predicted)', 'pos', 'net', 'neg'])
5 confusion_matrix = confusion_matrix.set_index(' (predicted)')
6
7
8 print(confusion_matrix)
9
10 accuracy = (A+B+C) / (A+B+C+D+E+F+G+H+I)
11 print('Akurasi : '+str(accuracy*100)+"%")
```

Gambar 5.35: Bagian utama program pengujian model klasifikasi sentimen

5.2.8 Evaluasi Metode Prediksi Hasil Pilkada

Untuk melihat seberapa baik metode prediksi hasil pilkada yang dilakukan, perlu dilakukan evaluasi yakni dengan menghitung tingkat kesalahan dari hasil yang didapat dengan hasil resmi dari KPUD DKI Jakarta. Oleh karena itu, pertama-tama, dilakukan proses perhitungan persentase tweet yang mendukung untuk masing-masing pasangan calon. Proses perhitungan persentase tweet yang mendukung untuk pasangan calon nomor urut dua ditunjukkan pada baris 2 Gambar 5.35, sedangkan persentase tweet yang mendukung untuk pasangan calon nomor urut tiga ditunjukkan pada baris 3. Persentase tweet yang mendukung untuk masing-masing pasangan calon yang telah didapat, kemudian dievaluasi dengan menghitung tingkat kesalahannya terhadap hasil resmi dari KPUD DKI Jakarta.

Metode perhitungan kesalahan yang digunakan adalah Mean Absolute Error. Proses perhitungan Mean Absolute Error ditunjukkan pada baris 4. Hasilnya kemudian dicetak seperti yang ditunjukkan pada baris 5.

```
1 def mae(dua,tiga):  
2     persentase_dua = dua/(dua+tiga)*100  
3     persentase_tiga = tiga/(dua+tiga)*100  
4     mae = (abs(42.05-persentase_dua)+abs(57.95-persentase_tiga))/2  
5     print(mae)
```

Gambar 5.36: Bagian program untuk menghitung persentase suara dan Mean Absolute Error

BAB VI

HASIL DAN PEMBAHASAN

6.1 Hasil Pengujian Klasifikasi Buzzer

Pengujian dilakukan menggunakan *10 cross fold validation* dengan jumlah data sebanyak 1466 buah ditunjukkan pada Gambar 6.1. Seperti yang terlihat pada *confusion matrix*, dari total data sebanyak 1466 akun, 337 akun *buzzer* semuanya terklasifikasi sebagai *buzzer*, 285 akun normal terklasifikasi sebagai *buzzer*, sebanyak 844 akun normal tetap terklasifikasi sebagai normal, dan tidak ada akun *buzzer* yang terklasifikasi sebagai normal. Akurasi rata-rata dari setiap *fold* yang telah dilakukan adalah 80,56%. Nilai presisi yang didapat adalah 54,18% dengan nilai *recall* sebesar 100% dan nilai *F1 Score* sebesar 70,28%. Hal ini berarti model mampu mengklasifikasi 80,56% dari seluruh akun secara benar. Persentase akun yang diidentifikasi dengan benar sebagai milik kelas ‘*buzzer*’ di antara semua akun yang diklasifikasikan oleh pengklasifikasi bahwa mereka termasuk dalam kelas ‘*buzzer*’ adalah sebesar 54,18%. Hal ini berarti ada sekitar 45,82% data dengan kelas ‘normal’ tetapi diklasifikasikan sebagai *buzzer*. Persentase kasus yang diidentifikasi dengan benar sebagai milik kelas ‘*buzzer*’ di antara semua kasus yang benar-benar termasuk dalam kelas ‘*buzzer*’ adalah sebesar 100%. Hal ini berarti model mampu mengklasifikasikan seluruh akun *buzzer* dengan benar (tidak ada satupun akun *buzzer* yang terklasifikasi sebagai normal).

```
IPython console
Console 1/A

In [112]: runfile('C:/Users/felix/Documents/Python Scripts/TugasAkhir/kfold.py',
wdir='C:/Users/felix/Documents/Python Scripts/TugasAkhir')
Reloaded modules: buzzer

      buzzer  normal
(predicted)
buzzer      337    285
normal         0    844
Akurasi : 80.55934515688949%
Presisi : 54.18006430868167%
Recall : 100.0%
F1 Score : 70.281543274244%

In [113]: |

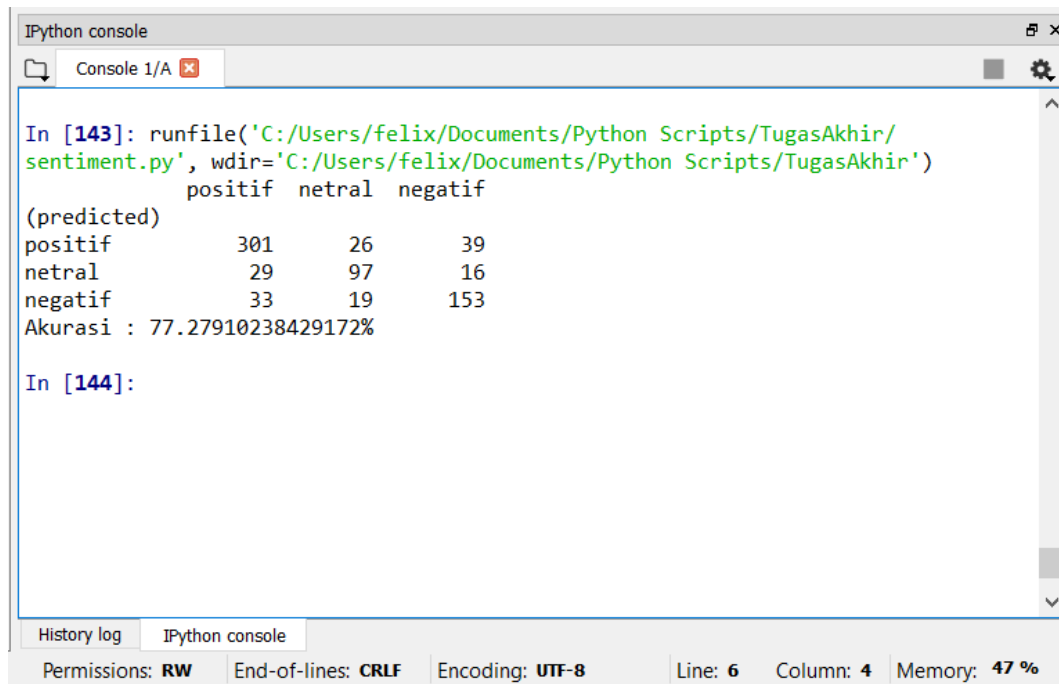
History log | IPython console
Permissions: RW | End-of-lines: CRLF | Encoding: UTF-8 | Line: 67 | Column: 15 | Memory: 46 %
```

Gambar 6.1: Hasil pengujian klasifikasi *buzzer* menggunakan *10-fold cross validation*

6.2 Hasil Pengujian Klasifikasi Sentimen

6.2.1 Hasil Pengujian Klasifikasi Sentimen Untuk Pasangan Ahok-Djarot

Pengujian dilakukan menggunakan *10 cross fold validation* dengan jumlah data sebanyak 713 buah ditunjukkan pada Gambar 6.2. Seperti yang terlihat pada *confusion matrix*, dari total data sebanyak 713 tweet, 301 tweet bersentimen positif terklasifikasi tetap sebagai positif, 29 tweet bersentimen positif terklasifikasi sebagai netral, 33 tweet bersentimen positif terklasifikasi sebagai negatif, 26 tweet bersentimen netral terklasifikasi sebagai positif, 97 tweet bersentimen netral tetap terklasifikasi sebagai netral, 19 tweet bersentimen netral terklasifikasi sebagai negatif. 39 tweet bersentimen negatif terklasifikasi sebagai positif, 16 tweet bersentimen negatif terklasifikasi sebagai netral, dan sebanyak 153 tweet bersentimen negatif tetap terklasifikasi sebagai negatif. Akurasi rata-rata dari setiap fold yang telah dilakukan adalah 77,28%. Hal ini berarti model mampu mengklasifikasi 77,28% dari seluruh data tweet secara benar.



```
In [143]: runfile('C:/Users/felix/Documents/Python Scripts/TugasAkhir/
sentiment.py', wdir='C:/Users/felix/Documents/Python Scripts/TugasAkhir')
          positif netral negatif
(predicted)
positif      301      26      39
netral        29      97      16
negatif       33      19     153
Akurasi : 77.27910238429172%

In [144]:
```

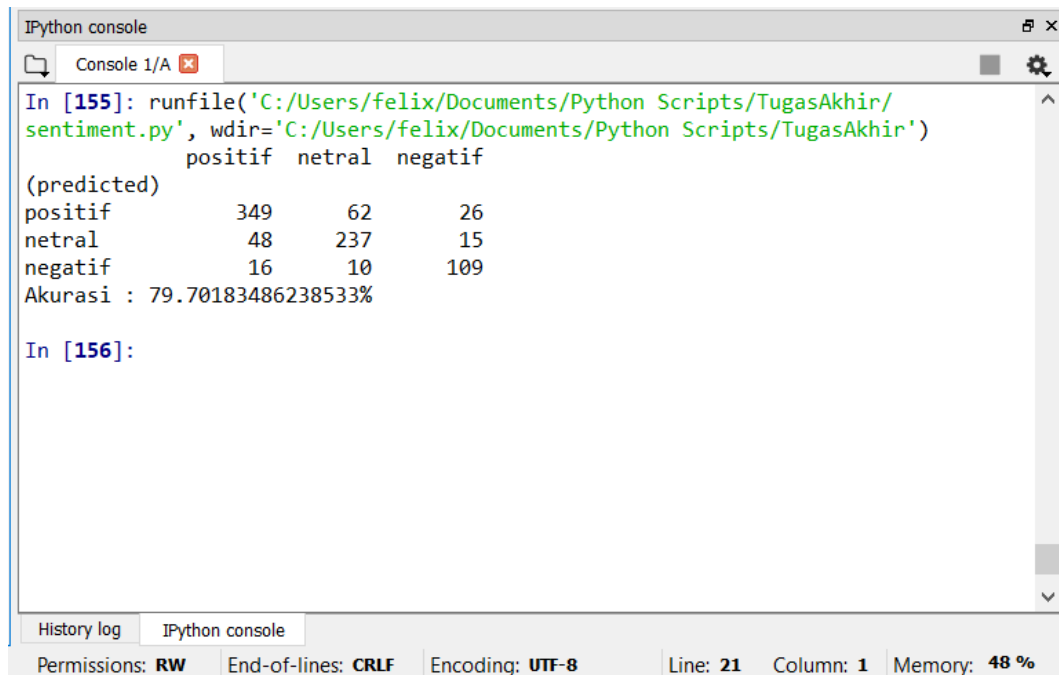
History log | IPython console

Permissions: **RW** | End-of-lines: **CRLF** | Encoding: **UTF-8** | Line: 6 | Column: 4 | Memory: 47 %

Gambar 6.2: Hasil pengujian klasifikasi sentimen pasangan Ahok-Djarot menggunakan *10-fold cross validation*

6.2.2 Hasil Pengujian Klasifikasi Sentimen Untuk Pasangan Anies-Sandi

Pengujian dilakukan menggunakan *10 cross fold validation* dengan jumlah data sebanyak 713 buah ditunjukkan pada Gambar 6.3. Seperti yang terlihat pada *confusion matrix*, dari total data sebanyak 872 tweet, 349 tweet bersentimen positif terklasifikasi tetap sebagai positif, 62 tweet bersentimen positif terklasifikasi sebagai netral, 26 tweet bersentimen positif terklasifikasi sebagai negatif, 48 tweet bersentimen netral terklasifikasi sebagai positif, 237 tweet bersentimen netral tetap terklasifikasi sebagai netral, 15 tweet bersentimen netral terklasifikasi sebagai negatif. 16 tweet bersentimen negatif terklasifikasi sebagai positif, 10 tweet bersentimen negatif terklasifikasi sebagai netral, dan sebanyak 109 tweet bersentimen negatif tetap terklasifikasi sebagai negatif. Akurasi rata-rata dari setiap fold yang telah dilakukan adalah 79,72%. Hal ini berarti model mampu mengklasifikasi 79,72% dari seluruh data tweet secara benar.



```
In [155]: runfile('C:/Users/felix/Documents/Python Scripts/TugasAkhir/sentiment.py', wdir='C:/Users/felix/Documents/Python Scripts/TugasAkhir')
          positif netral negatif
(predicted)
positif      349      62      26
netral       48     237      15
negatif      16      10     109
Akurasi : 79.70183486238533%

In [156]:
```

Gambar 6.3: Hasil pengujian klasifikasi sentimen pasangan Anies-Sandi menggunakan *10-fold cross validation*

Bila dibandingkan dengan nilai akurasi hasil pengujian klasifikasi sentimen untuk pasangan Ahok-Djarot, nilai akurasi hasil pengujian klasifikasi sentimen untuk pasangan Anies-Sandi lebih besar sekitar 2,45%. Hal ini dapat disebabkan karena jumlah data latih yang lebih besar bila dibandingkan dengan data latih untuk pasangan Ahok-Djarot. Jumlah data latih yang besar berakibat pada besarnya jumlah fitur yang digunakan dalam proses pelatihan model sehingga berpengaruh terhadap performa akurasi.

6.3 Hasil Klasifikasi Buzzer Pada Data Tes

Hasil klasifikasi *buzzer* pada data tes ditunjukkan pada Gambar 6.4. Hasil tersebut kemudian disimpan ke dalam suatu file CSV untuk digunakan dalam proses penghapusan tweet *buzzer*.

```

IPython console
Console 1/A
flyfluffy : Normal user
Yaanse : Normal user
ricosiswandi : Normal user
Sandi_becpe66 : Buzzer
eradotid : Buzzer
bungalilyloca : Normal user
witapril28 : Normal user
otan21 : Buzzer
selliyamanto : Buzzer
ElfriedaBatya : Buzzer
hermansaksono : Normal user
DasarKita : Buzzer
meeneel : Buzzer
ShoovdorBatbold : Normal user
klinik_fortuna : Buzzer
EdSamuel8 : Buzzer
PangeranBiru212 : Normal user
maulidineamira : Normal user
Casifamosoo : Buzzer
History log IPython console
Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 55 Column: 115 Memory: 49 %

```

Gambar 6.4: Contoh hasil klasifikasi *buzzer* pada data tes

Dari hasil yang didapat, diketahui bahwa dari total 3421 akun pada data tes, 924 akun terklasifikasi sebagai *buzzer* sedangkan 2497 akun terklasifikasi sebagai pengguna normal. Hal ini ditunjukkan pada Tabel 6.1.

Tabel 6.1 : Jumlah masing-masing kelas yang dihasilkan dalam proses klasifikasi *buzzer* pada data tes

Kelas	Jumlah
Buzzer	924
Normal	2497

6.4 Hasil Klasifikasi Sentimen Pada Data Tes

6.4.1 Hasil Klasifikasi Sentimen Pasangan Ahok-Djarot Pada Data Tes

Hasil klasifikasi sentimen untuk pasangan Ahok-Djarot pada data tes ditunjukkan pada Gambar 6.5. Hasil klasifikasi tersebut kemudian dihitung jumlahnya untuk masing-masing sentimen ditunjukkan pada Tabel 6.2.


```

IPython console
Console 1/A
1648 || class: positive
1649 || class: positive
1650 || class: positive
1651 || class: negative
1652 || class: positive
1653 || class: neutral
1654 || class: neutral
1655 || class: positive
1656 || class: negative
1657 || class: negative
1658 || class: positive
1659 || class: negative
1660 || class: neutral
1661 || class: negative
1662 || class: positive
1663 || class: positive
POS = 837 | NEG = 345 | NET = 481
In [114]:
History log IPython console
Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 43 Column: 24 Memory: 42 %

```

Gambar 6.5: Contoh hasil klasifikasi sentimen pasangan Ahok-Djarot pada data tes

Tabel 6.2 : Jumlah masing-masing kelas yang dihasilkan dalam proses klasifikasi sentimen pasangan Ahok-Djarot pada data tes

Kelas	Jumlah
Positif	837
Netral	481
Negatif	345

6.4.2 Hasil Klasifikasi Sentimen Pasangan Anies-Sandi Pada Data Tes

Hasil klasifikasi sentimen untuk pasangan Anies-Sandi pada data tes ditunjukkan pada Gambar 6.6. Hasil klasifikasi tersebut kemudian dihitung jumlahnya untuk masing-masing sentimen ditunjukkan pada Tabel 6.3.

```

IPython console
Console 1/A
2018 | class: positive
2019 | class: positive
2020 | class: positive
2021 | class: negative
2022 | class: positive
2023 | class: neutral
2024 | class: neutral
2025 | class: positive
2026 | class: negative
2027 | class: negative
2028 | class: positive
2029 | class: negative
2030 | class: neutral
2031 | class: negative
2032 | class: positive
2033 | class: positive
POS = 1017 | NEG = 322 | NET = 694
In [136]:
History log | IPython console
Permissions: RW | End-of-lines: CRLF | Encoding: UTF-8 | Line: 26 | Column: 1 | Memory: 42 %

```

Gambar 6.6: Contoh hasil klasifikasi sentimen pasangan Anies-Sandi pada data tes

Tabel 6.3 : Jumlah masing-masing kelas yang dihasilkan dalam proses klasifikasi sentimen pasangan Ahok-Djarot pada data tes

Kelas	Jumlah
Positif	1017
Netral	694
Negatif	322

6.5 Hasil Penghapusan Tweet Buzzer

Dari 924 akun *buzzer* yang terklasifikasi pada data tes dan 337 akun *buzzer* pada data latih, bila dijumlahkan maka akan menghasilkan jumlah *buzzer* sebanyak 1261 akun. Dalam proses penghapusan tweet *buzzer*, seluruh tweet yang dihasilkan oleh 1261 akun *buzzer* tersebut dihapus. Total terdapat 1329 tweet yang dihapus dalam proses penghapusan tweet *buzzer* untuk seluruh data tweet baik data latih maupun data tes, di mana 640 tweet untuk pasangan Ahok-Djarot dan 689 tweet untuk pasangan Anies-Sandi. Rincian jumlah tweet yang dihasilkan oleh *buzzer* untuk masing-masing pasangan calon dan sentimen dapat dilihat pada Tabel 6.4.

Tabel 6.4 : Jumlah tweet yang dihasilkan oleh *buzzer* untuk masing-masing pasangan calon dan sentimen pada seluruh data

Pasangan Calon	Kelas		
	Positif	Netral	Negatif
Ahok-Djarot	352	143	145
Anies-Sandi	335	245	109
Total	687	388	254

Jumlah data latih dan data tes yang telah digabungkan untuk masing-masing pasangan calon dan sentimen ditunjukkan pada Tabel 6.5. Sedangkan, Jumlah tweet untuk masing-masing pasangan calon dan sentimen pada seluruh data sesudah dilakukan proses *buzzer detection* ditunjukkan pada Tabel 6.6.

Tabel 6.5 : Jumlah tweet untuk masing-masing pasangan caklon dan sentimen pada seluruh data

Pasangan Calon	Kelas		
	Positif	Netral	Negatif
Ahok-Djarot	1196	629	551
Anies-Sandi	1453	992	460
Total	2649	1621	1011

Tabel 6.6 : Jumlah tweet untuk masing-masing pasangan calon dan sentimen pada seluruh data sesudah dilakukan proses *buzzer detection*

Pasangan Calon	Kelas		
	Positif	Netral	Negatif
Ahok-Djarot	844	486	406
Anies-Sandi	1118	747	351
Total	1962	1233	757

6.6 Hasil dan Analisis Prediksi Pilkada DKI Jakarta 2017

Dari Tabel 6.6, dapat dilihat bahwa proses *buzzer detection* mengurangi jumlah data sebesar 25,17%, dari jumlah data yang semula sebanyak 5281 data

menjadi 3952 data. Jumlah data yang berkurang untuk masing-masing sentimen ditunjukkan pada Tabel 6.7, sedangkan rincian untuk masing-masing pasangan calon ditunjukkan pada Tabel 6.8.

Tabel 6.7 : Persentase tweet yang dihasilkan oleh *buzzer* untuk masing-masing kelas sentimen

Kelas	Persentase pengurangan
Positif	25,93%
Netral	24,06%
Negatif	25,12%

Dapat dilihat bahwa persentase sentimen terbesar yang berkurang akibat proses *buzzer detection* adalah sentimen positif, diikuti oleh sentimen negatif, hingga yang terkecil yakni sentimen netral. Hal ini berarti, secara umum, *buzzer* lebih banyak mengirimkan tweet-tweet bersentimen positif dibanding sentimen lainnya. Sedangkan, tweet bersentimen negatif berada dalam urutan kedua sebagai tweet yang paling banyak dikirimkan oleh *buzzer*.

Tabel 6.8 : Persentase tweet yang dihasilkan oleh *buzzer* untuk masing-masing pasangan calon

	Positif	Netral	Negatif
Ahok-Djarot	29,43%	22,63%	26,32%
Anies-Sandi	23,06%	24,7%	23,7%

Dari Tabel 6.8, dapat dilihat bahwa di antara tiga sentimen yang ada pada tweet-tweet yang berkaitan dengan Ahok-Djarot, jumlah sentimen positif yang dihasilkan oleh *buzzer* adalah yang terbesar yakni sebesar 29,43%. Sedangkan, sentimen negatif berada pada urutan kedua dengan persentase sebesar 26,32%. Hal ini berbeda dengan pasangan Anies-Sandi yang mana jumlah sentimen negatif yang dihasilkan oleh *buzzer* lebih besar dibanding sentimen positifnya. Akan tetapi, persentase sentimen positif dan negatif yang dihasilkan oleh *buzzer* pada pasangan Anies-Sandi jauh lebih kecil bila dibandingkan dengan pasangan Ahok-Djarot.

Artinya, untuk setiap tweet bersentimen positif dan negatif yang berkaitan dengan masing-masing pasangan calon, lebih banyak tweet yang dihasilkan oleh *buzzer* untuk pasangan Ahok-Djarot dibanding dengan pasangan Anies-Sandi.

Jumlah tweet yang telah diklasifikasikan sentimennya untuk masing-masing pasangan calon digunakan dalam menghitung prediksi hasil pilkada. Perhitungan prediksi hasil pilkada dilakukan dengan membandingkan jumlah tweet yang mendukung masing-masing pasangan calon, misalnya jumlah tweet bersentimen positif pasangan Ahok-Djarot dibandingkan dengan jumlah tweet bersentimen positif pasangan Anies-Sandi, kemudian dihitung persentasenya. Karena kandidat dalam pilkada ini hanya ada dua, selain hanya menggunakan tweet positif, penelitian ini menggunakan asumsi, bahwa sentimen negatif terhadap suatu kandidat berarti tweet tersebut mendukung kandidat lainnya (positif terhadap kandidat lainnya). Asumsi ini juga digunakan pada penelitian Gayo-Avello (2011) dan Prasetyo (2014).

Penelitian ini membandingkan tiga metode berdasarkan sentimen tweet yang digunakan, yakni hanya menggunakan tweet bersentimen positif saja, hanya menggunakan tweet bersentimen negatif saja, serta menggunakan tweet bersentimen positif dan negatif. Hal ini dimaksudkan untuk melihat sentimen manakah yang memiliki pengaruh paling besar terhadap hasil prediksi pilkada. Selain itu, dibandingkan juga hasil yang didapat jika data yang digunakan melalui proses *buzzer detection* dengan yang tidak. Hal ini dimaksudkan untuk melihat seberapa besar pengaruh *buzzer* di media sosial dalam mempengaruhi hasil prediksi pilkada serta menganalisa sentimen apa saja yang banyak dihasilkan oleh para *buzzer* untuk masing-masing pasangan calon.

Untuk menghitung seberapa efektif data sentimen digunakan dalam menentukan pemenang pilkada, diperlukan perhitungan Mean Absolute Error dengan membandingkan persentase hasil prediksi pilkada dengan hasil resmi dari KPU DKI Jakarta. Jumlah suara sah Pilkada DKI Jakarta 2017 pada putaran kedua dapat dilihat pada Tabel 6.9.

Tabel 6.9 : Jumlah suara sah Pilkada DKI Jakarta 2017 putaran kedua (KPU DKI Jakarta, 2017)

Nomor Urut Pasangan Calon	Jumlah Suara	Persentase
Dua (Ahok-Djarot)	2.351.245	42,05%
Tiga (Anies-Sandi)	3.240.332	57,95%

Hasil perhitungan Mean Absolute Error yang didapat dari penelitian ini kemudian dibandingkan dengan hasil survei lembaga-lembaga independen seperti yang ditunjukkan pada Tabel 6.10. Hal ini dimaksudkan untuk melihat seberapa baik metode prediksi hasil pilkada menggunakan data Twitter dalam yang dilakukan dalam penelitian ini dibandingkan dengan metode survei konvensional.

Tabel 6.10 : Tabel perbandingan Mean Absolute Error antara hasil survei independen dengan metode berbasis data Twitter yang dilakukan pada penelitian ini

Lembaga Survei	Tanggal Survei	Nomor Urut Pasangan Calon		MAE
		Dua (Ahok-Djarot)	Tiga (Anies-Sandi)	
Institusi Survei Independen				
Median	21–27 Februari 2017	46,16%	53,84%	4,11%
LSI Denny JA	27 Februari – 3 Maret 2017	44,9%	55,1%	2,85%
SMRC	31 Maret – 5 April 2017	49,47%	50,53%	7.42%
Median	1–6 April 2017	46,62%	53,38%	4,57%
LSI Denny JA	7–9 April 2017	45,38%	54,62%	3,33%
Charta Politika	7–12 April 2017	51,36%	48,64%	9,31%
Indikator	12–14 April 2017	49,58%	50,42%	7,53%
Median	13–14 April 2017	49,01%	50,99%	6,96%
Penelitian Ini				

Metode	Nomor Urut Pasangan Calon		MAE
	Dua (Ahok-Djarot)	Tiga (Anies-Sandi)	
Data bersentimen positif tanpa <i>buzzer detection</i>	1196 45,15%	1453 54,85%	3,1%
Data bersentimen positif dengan <i>buzzer detection</i>	844 43,02%	1118 56,98%	0,97%
Data bersentimen negatif tanpa <i>buzzer detection</i>	460 45,5%	551 54,5%	3,45%
Data bersentimen negatif dengan <i>buzzer detection</i>	351 46,37%	406 53,63%	4,32%
Data bersentimen positif dan negatif tanpa <i>buzzer detection</i>	1656 45,25%	2004 54,75%	3,2%
Data bersentimen positif dan negatif dengan <i>buzzer detection</i>	1195 43,95%	1524 56,05%	1,9%

Dari Tabel 6.10, dapat dilihat bahwa metode terbaik adalah dengan menggunakan tweet bersentimen positif disertai proses *buzzer detection* yang menghasilkan nilai MAE sebesar 0.97% dan yang terburuk adalah survei Charta Politika pada tanggal 7-12 April 2017 dengan nilai MAE sebesar 9,31%. Secara umum, rata-rata nilai MAE yang dihasilkan oleh penelitian ini jauh lebih kecil bila dibandingkan dengan rata-rata nilai MAE metode survei konvensional, sehingga bisa dikatakan bahwa metode ini lebih baik dibandingkan dengan metode survei konvensional.

Dapat dilihat pada Tabel 6.10, bahwa teknik *buzzer detection* mampu menurunkan tingkat kesalahan prediksi hasil pilkada dengan rata-rata penurunan sebesar 0,85%. Walaupun secara rata-rata menurunkan tingkat kesalahan, akan tetapi penggunaan *buzzer detection* malah meningkatkan nilai MAE untuk data yang bersentimen negatif. Hal ini dapat disebabkan karena banyaknya tweet bersentimen negatif yang ditujukan oleh *buzzer* kepada pasangan Ahok-Djarot dibanding kepada pasangan Anies-Sandi, sehingga . Selain itu, dapat disimpulkan pula bahwa penggunaan hanya sentimen positif dalam prediksi hasil pilkada

mampu menghasilkan performa lebih baik dibanding dua metode lainnya dengan tingkat kesalahan rata-rata hanya sebesar 2,04%, lebih kecil bila dibandingkan dengan dua metode lainnya.

BAB VII

PENUTUP

7.1 Kesimpulan

Berdasarkan penelitian yang sudah dilakukan, dapat ditarik kesimpulan sebagai berikut:

1. Metode prediksi hasil pilkada dengan melakukan analisa sentimen tweet pada Twitter mampu memprediksi hasil Pilkada DKI Jakarta 2017 dengan benar dan dengan tingkat kesalahan yang lebih kecil dibandingkan dengan metode survei konvensional.
2. Teknik *buzzer detection* mampu menurunkan tingkat kesalahan prediksi hasil pilkada dengan rata-rata penurunan sebesar 0,85%.
3. Penggunaan hanya sentimen positif dalam prediksi hasil pilkada lebih baik dibanding dua metode lainnya dengan tingkat kesalahan rata-rata sebesar 2,04%.
4. Kombinasi penggunaan hanya sentimen positif dengan teknik *buzzer detection* mampu menghasilkan prediksi terbaik dengan tingkat kesalahan sebesar 0,97%.
5. Akun *buzzer* lebih banyak menghasilkan tweet bersentimen positif untuk pasangan calon yang didukungnya dibandingkan tweet bersentimen negatif untuk pasangan calon yang ditentangnya.
6. Implementasi algoritma Multinomial Naive Bayes yang dilakukan untuk mengklasifikasi sentimen pada data Twitter berbahasa Indonesia sudah bekerja dengan baik dengan rata-rata nilai akurasi sebesar 78,5%.
7. Implementasi algoritma Gaussian Naive Bayes yang dilakukan untuk mengklasifikasi akun *buzzer* pada data Twitter sudah bekerja dengan baik dengan nilai akurasi sebesar 80,56%.

7.2 Saran

Saran yang dapat diberikan untuk penelitian selanjutnya adalah sebagai berikut:

1. Menambahkan jumlah fitur pada proses klasifikasi *buzzer* guna meningkatkan performa klasifikasi.
2. Melakukan *user profiling* terhadap pendukung masing-masing pasangan calon agar dapat dianalisis komposisi demografi pendukung masing-masing pasangan calon.
3. Menggunakan metode *pre-processing* teks yang lebih baik atau algoritma klasifikasi yang berbeda dengan harapan performa klasifikasi sentimen yang dihasilkan bisa lebih baik..

REFERENSI

- Analytics, P., 2009. *Twitter Study*. <https://pearanalytics.com/blog/2009/twitter-study-reveals-interesting-results-40-percent-pointless-babble>, Agustus 2009, diakses 11 Mei 2017.
- Beiletes, C., Salzer R. dan Sergio V., 2013. *Validation of Soft Classification Models using Partial Class Memberships: An Extended Concept of Sensitivity & Co. applied to Grading of Astrocytoma Tissues*, Chemom. Intell. Lab. Syst.
- Bermingham, A. dan Alan F. S., 2011. *On Using Twitter to Monitor Political Sentiment and Predict Election Results*, Proceedings of the Workshop on Sentiment Analysis where AI meets Psychology (SAAIP), IJCNLP 2011.
- Bustamante, C., Leonardo G. dan Rogelio S., 2006. *Comparing Fuzzy Naive Bayes and Gaussian Naive Bayes for Decision Making in RoboCup 3D*, 5th Mexican International Conference on Artificial Intelligence Proceedings.
- Chakrabarti, S., 2006. *Data Mining Curriculum: A Proposal (Version 1.0)*, ACM SIGKDD Curriculum.
- Cornfield, M., 2008. *Yes, it did make a difference*, <http://takingnote.tcf.org/2008/06/yes-it-did-make.html>, Juni 2008, diakses 11 Mei 2017.
- Deshwal, A. dan Sudhir K. S., 2016. *Twitter Sentiment Analysis using Various Classification Algorithms*, 2016 5th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions).
- Fawcett, T., 2006. *An introduction to ROC analysis*, Pattern Recognition Letters 27 (2006) 861–874.
- Ficamos, P., Yan L. dan Weiyi C., 2017. *A Naive Bayes and Maximum Entropy approach to Sentiment Analysis: Capturing Domain-Specific Data in Weibo*, 2017 IEEE International Conference on Big Data and Smart Computing (BigComp).
- Gayo-Avello D. M., 2011. *Don't turn social media into another 'Literary Digest' poll*. Communications of the ACM.
- Gayo-Avello D. M., 2011. *Limits of electoral predictions using twitter*. ICWSM.
- Gheware, S., 2014. *Data Mining Task, Tools, Techniques and Applications*, International Journal of Advanced Research in Computer and Communication Engineering. Vol. 3, Issue 10.
- Hand, D. J. dan Keming Y., 2001. *Idiot's Bayes — not so stupid after all?*, International Statistical Review.

- Huang, J., Jingjing L. dan Charles X. L., 2003. *Comparing Naive Bayes, Decision Trees, and SVM with AUC and Accuracy*, 3rd IEEE International Conference on Data Mining.
- Huber, J. dan Florian H., 2005. *Systematic mispricing in experimental markets evidence from political stock markets*, 10th Annual Workshop on Economic Heterogeneous Interacting Agents.
- Ibrahim, M. et al, 2015. *Buzzer Detection and Sentiment Analysis for Predicting Presidential Election Results in A Twitter Nation*, 2015 IEEE 15th International Conference on Data Mining Workshops.
- John, G. H. dan Pat L., 1995. *Estimating continuous distributions in Bayesian classifiers*, UAI'95 Proceedings of the Eleventh conference on Uncertainty in artificial intelligence.
- Jurafsky, D. dan James H. M., 2016. *Speech and Language Processing*, Prentice-Hall, Inc.
- Manning, C. D., Prabhakar R. dan Hinrich S., 2008. *An Introduction to Information Retrieval*, Cambridge University Press.
- Media, K., 2017. *Teganjal Aturan, "Buzzer" Kampanye Saat Masa Tenang Tak Bisa Ditindak*, <http://nasional.kompas.com/read/2017/02/07/21343741/teganjal.aturan.buzzer.kampanye.saar.masa.tenang.tak.bisa.ditindak>, 7 Februari 2017, diakses 10 Mei 2017.
- Mitchell, T., 1997. *Machine Learning*, McGraw-Hill.
- Pang, B dan Lillian L., 2008. *Opinion mining and sentiment analysis*, Foundations and Trends in Information Retrieval Vol 2.
- Prasetyo, N. D., 2014. *Tweet-based election prediction*, M. Sc. Thesis, Delft University of Technology Netherland.
- Ramteke, Jyoti et al., 2016. *Election result prediction using Twitter sentiment analysis*, 2016 International Conference on Inventive Computation Technologies (ICICT)
- Sokolova, M. dan Guy L. 2009. *A systematic analysis of performance measures for classification tasks*, Information Processing and Management, 45, p. 427-437.
- Tumasjan, Andranik et al., 2010. *Predicting elections with twitter: What 140 characters reveal about political sentiment*, International AAAI Conference on Web and Social Media (ICWSM).
- V, Umadevi, 2014. *Sentiment Analysis Using Weka*, International Journal of Engineering Trends and Technology (IJETT) Volume 18 Number 4

- Wicaksono, A. J., Suyoto dan Pranowo, 2016. *A Proposed Method for Predicting US Presidential Election by Analyzing Sentiment in Social Media*, 2016 2nd International Conference on Science in Information Technology (ICSITech).
- Witten, D. M., 2011. *Classification And Clustering Of Sequencing Data Using A Poisson Model*, The Annals of Applied Statistics 2011, Vol. 5, No. 4, 2493–2518.
- Zhang, H., 2004. *The Optimality of Naive Bayes*, Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference.
- Zhang, Y., Huaiyu W. dan Lei C., 2012. *Some New Deformation Formulas about Variance and Covariance*, Proceedings of 4th International Conference on Modelling, Identification and Control.

LAMPIRAN

A. Tabel Kata *Stopwords*

No	Kata
1	saya
2	aku
3	gue
4	kamu
5	kau
6	dia
7	ia
8	kami
9	kalian
10	beliau
11	elo
12	lo
13	loe
14	lu
15	elu
16	ini
17	itu
18	mereka
19	yang
20	apa
21	kapan
22	kemana
23	dimana
24	kenapa
25	mengapa
26	siapa
27	bagaimana
28	gimana
29	telah
30	suatu
31	karena
32	dan
33	atau
34	tapi
35	tetapi
36	dari
37	ke
38	di
39	hingga
40	selama

No	Kata
41	pun
42	sehingga
43	untuk
44	dengan
45	antara
46	sebelum
47	sesudah
48	lagi
49	maka
50	jika
51	sekali
52	semua
53	lebih
54	kurang
55	paling
56	beberapa
57	bukan
58	hanya
59	bila
60	sangat
61	sama
62	harus
63	sekarang
64	kemarin
65	yaitu
66	seperti
67	bak
68	yg
69	dalam
70	pada
71	oleh
72	sebab
73	memang
74	tak
75	tidak
76	ya
77	iya
78	emang
79	berapa
80	begitu

No	Kata
81	juga
82	begini
83	gini
84	gitu
85	sana
86	sini
87	yakni
88	misalnya
89	sendiri
90	justru
91	tersebut
92	merupakan
93	adalah
94	amat
95	terlalu
96	sekian
97	demikian
98	menjadi
99	kepada
100	makin
101	semakin
102	kata
103	ujar
104	bagai
105	bagaikan
106	melainkan
107	padahal
108	sedang
109	sedangkan
110	sekedar
111	sekadar
112	&
113	ketika
114	namun
115	bisa
116	saja
117	beserta
118	telanjur
119	belum
120	sudah

No	Kata
121	sempat
122	akhirnya

No	Kata
123	nyaris
124	atas