MODEL PEMBELAJARAN DAN LAPORAN AKHIR
PROJECT-BASED LEARNING
MATA KULIAH BIG DATA
KELAS B

“Klasifikasi dan Analisis Sentimen Teks Ulasan Makanan pada Amazon menggunakan PySpark”

DISUSUN OLEH KELOMPOK “IV” :

1. ALYSSA AMORITA AZZAH (21083010057) - KETUA
2. EDINA ALANA NABILA (21083010022) - ANGGOTA
3. YASMIN ULAYYA (21083010033) - ANGGOTA
4. SHENY EKA OKTAVIANI (21083010037) - ANGGOTA
5. FIQIH PAVITA A (21083010042) - ANGGOTA

DOSEN PENGAMPU:

TRESNA MAULANA FAHRUDIN, S.ST., MT (199305012022031007)
KARTIKA MAULIDA HINDRAYANI, S.KOM, M.KOM (199209092022032009)

PROGRAM STUDI SAINS DATA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS PEMBANGUNAN NASIONAL “VETERAN”
JAWA TIMUR

2023

MODEL PEMBELAJARAN

1. Buatlah kelompok (1 kelompok sekitar 2-3 mahasiswa), untuk menentukan ide dan topik proyek Big Data menggunakan software Apache pilihan berikut ini:
    a. Apache Mahout
    b. Apache Ambari
    c. Apache Sqoop
    d. Apache HBASE
    e. Apache Kafka
    f. Cassandra
    g. Elastic Search
    h. Kibana
    i. Apache nutch
    j. MongoDB

    Software yang diusulkan tidak terbatas pada area bidang di atas, kelompok boleh mengusulkan penggunaan software Big Data lainnya selama dalam ruang lingkup Big Data.

    Penilaian ditekankan pada bagaimana kelompok menentukan dan memilih ide dan topik permasalahan yang diselesaikan serta menggunakan software Big Data yang telah direkomendasikan oleh kelompok.

    Beberapa contoh Proyek Big Data di antaranya:

    1. Data Engineering Project Building a Spotify ETL Pipeline using Python, Tautan: https://blog.devgenius.io/data-engineering-project-2-building-spotify-etl-using-python-and-airflow-432dd8e4ffa3
    2. Scheduling Streaming Insert ke BigQuery Menggunakan Serverless Option, Tautan: https://medium.com/google-cloud-indonesia/scheduling-streaming-insert-ke-bigquery-menggunakan-serverless-option-3e54b4825a22

3. How to use Airflow for Data Engineering pipelines in Google Cloud Platform, Tautan: https://medium.com/google-cloud/how-to-use-airflow-for-data-engineering-pipelines-in-gcp-22964481a3db
4. Stock Market Real-Time Data Analysis Using Kafka, Tautan: https://www.youtube.com/watch?v=KerNf0NANMo, https://github.com/nizmarcoding/Stock-Market-Real-Time-Data-Analysis-Using-Kafka ,
5. YouTube Data Analysis (End-To-End Data Engineering Project), Tautan: https://www.youtube.com/watch?v=yZKJFKu49Dk , https://medium.com/@darshilp/end-to-end-data-engineering-project-youtube-analytics-1c7eebf0d31
6. Twitter Data Pipeline using Airflow, Tautan: https://www.youtube.com/watch?v=q8q3OFFfY6c , https://github.com/darshilparmar/twitter-airflow-data-engineering-project

Output dari *project-based learning* juga adalah laporan proyek sesuai format yang ditentukan dan file pendukung proyek.

2. Dokumentasi *project-based learning* didokumentasikan ke dalam laporan yang mencakup:
    - Halaman Depan
    - Kata Pengantar
    - Latar Belakang beserta subbabnya
    - Tinjauan Pustaka beserta subbabnya
    - Metodologi Penelitian beserta subbabnya
    - Hasil dan Pembahasan beserta subbabnya
    - Kesimpulan
    - Lampiran

3. Video dokumentasi proyek yang menarik dan diunggah di platform Youtube (Lampiran)

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

1. BAB I: PENDAHULUAN

Bab 1 mencakup penjelasan terkait latar belakang, permasalahan, tujuan, dan manfaat dari proyek yang diusulkan. Pada bab ini memberikan gambaran tentang topik permasalahan yang diangkat, tingkat urgensinya, dan dampak yang akan diperoleh dari proyek yang diusulkan.

1.1 Latar Belakang

Uraikan latar belakang dari proyek yang diusulkan secara deduktif dimulai dari topik permasalahan secara umum yang sedang diteliti hingga mengerucut ke objek permasalahan yang akan diselesaikan.

E-commerce menjadi hal yang sangat populer dan mendominasi aktivitas perdagangan di era digital saat ini. Dengan kemajuan teknologi dan peningkatan akses internet, e-commerce telah mengubah cara orang berbelanja dan berbisnis secara drastis. Salah satu e-commerce terbesar dan dikenali oleh banyak orang dari berbagai Negara adalah amazon. Amazon awalnya hanya berfokus pada penjualan buku secara online. Namun, seiring berjalannya waktu, Amazon telah berekspansi menjadi platform yang menyediakan berbagai produk dan layanan, termasuk elektronik, pakaian, alat rumah tangga, layanan cloud computing, hingga makanan. Melalui inovasi dan strategi ekspansi yang agresif, Amazon terus berkembang dan menjadi salah satu perusahaan terbesar dan paling berpengaruh di dunia. Pada tahun ini, amazon berhasil mencatat pertumbuhan pendapatan sebesar 9 persen pada kuartal pertama dari tahun sebelumnya, yaitu laba sebesar US$3,2 miliar atau Rp46,9 triliun. Hal ini menunjukkan bahwa area utama bisnis Amazon terus tumbuh, dan perusahaan dapat mempertahankan momentum pertumbuhan.

Dengan berkembangnya sektor e-commerce, Amazon telah berhasil memperluas kehadirannya dalam pasar makanan dengan menawarkan berbagai macam produk makanan dan minuman kepada pelanggan di seluruh dunia. Amazon menyediakan berbagai macam produk makanan dari berbagai merek dan penjual, sehingga pelanggan dapat menemukan makanan dan minuman dalam kategori yang

luas, termasuk makanan ringan, makanan kering, produk segar, minuman, bahan makanan, dan banyak lagi. Namun, seiring dengan pertumbuhan dan ekspansi Amazon di bidang makanan, ulasan makanan dari pelanggan juga memainkan peran penting dalam membantu pembeli membuat keputusan pembelian yang lebih baik. Ulasan yang baik dapat membantu meningkatkan reputasi merek, meningkatkan penjualan, dan membangun hubungan jangka panjang antara penjual dan pelanggan. Namun, ulasan makanan dapat subjektif karena preferensi dan selera masing-masing orang berbeda. Oleh karena itu, penting bagi pembeli untuk membaca ulasan dengan bijak dan mempertimbangkan berbagai ulasan yang ada sebelum membuat keputusan pembelian.

Oleh karena pentingnya ulasan makanan dari pelanggan di platform e-commerce Amazon, kami melakukan pengembangan Analisis Sentimen dan klasifikasi teks ulasan makanan menggunakan PySpark. Hal ini bertujuan untuk mengekstraksi sentimen atau perasaan yang terkandung dalam ulasan makanan, baik itu positif, negatif, atau netral. Sehingga dapat mengidentifikasi ekspresi emosi dan penilaian yang diberikan oleh pelanggan dalam ulasan mereka. Analisis Sentimen ini berguna bagi calon pembeli yang ingin mendapatkan gambaran umum tentang pengalaman pelanggan sebelum mereka membeli produk makanan tertentu. Selain itu, kami juga melakukan klasifikasi teks pada ulasan makanan yang dapat mengklasifikasikan ulasan makanan ke dalam kategori yang relevan, seperti rasa, kualitas, pengiriman, layanan pelanggan, dan lainnya. Pendekatan ini membantu dalam mengorganisir ulasan makanan ke dalam topik-topik yang spesifik, sehingga memudahkan pembeli untuk mencari ulasan yang paling relevan dengan minat atau kebutuhan mereka.

1.2 Permasalahan

Identifikasi permasalahan dari proyek yang diteliti dalam bentuk poin-poin permasalahan secara tepat dan jelas.

1. Sulitnya melakukan analisis sentimen dan klasifikasi teks untuk mengidentifikasi ekspresi emosi dari ulasan pelanggan
2. Belum tersedianya analisis sentimen dan klasifikasi teks ulasan pelanggan pada Amazon e-commerce

1.3 Tujuan

Identifikasi tujuan dari proyek yang diteliti dalam bentuk poin-poin tujuan secara tepat dan jelas sesuai dengan permasalahan yang diangkat serta mengusulkan metode atau pendekatan sebagai solusi alternatif.

1. Melakukan analisis sentimen dan klasifikasi teks untuk mengidentifikasi ekspresi emosi dari ulasan pelanggan menggunakan PySpark
2. Tersedianya analisis sentimen dan klasifikasi teks ulasan pelanggan pada Amazon e-commerce

1.4 Manfaat

Uraikan manfaat yang akan didapat dari proyek yang diusulkan, baik bagi pengembangan ilmu pengetahuan, maupun pihak-pihak terkait.

1.4.1 Manfaat bagi masyarakat

Penelitian ini memberikan manfaat bagi masyarakat dalam memperoleh informasi yang lebih akurat dan relevan tentang produk makanan yang mereka pertimbangkan untuk dibeli. Dengan adanya analisis sentimen, pembeli dapat mengetahui pandangan pelanggan lain tentang produk tersebut, sehingga mereka dapat membuat keputusan pembelian yang lebih cerdas dan menghindari produk yang mungkin tidak memenuhi harapan mereka.

1.4.2 Manfaat bagi industri

Dengan memahami ulasan pelanggan, penelitian ini dapat membantu identifikasi tren konsumen, preferensi pasar, dan kebutuhan yang belum terpenuhi. Informasi ini dapat digunakan oleh produsen makanan untuk melakukan inovasi, mengembangkan produk yang lebih sesuai dengan keinginan pelanggan, dan meningkatkan daya saing industri makanan secara global. Dengan demikian, penelitian ini memiliki potensi untuk mendorong pertumbuhan dan perkembangan industri makanan yang lebih baik, memberikan pilihan yang lebih baik bagi konsumen, dan menciptakan lapangan kerja yang lebih banyak dalam sektor ini.

1.4.3 Manfaat bagi Universitas

Penelitian ini berkontribusi terhadap pengetahuan akademik dengan mengembangkan pemahaman yang lebih baik tentang analisis sentimen dan teknik klasifikasi teks dalam konteks ulasan makanan. Dengan mengaplikasikan metode dan algoritma yang relevan, penelitian ini dapat memperluas pemahaman kita tentang bagaimana ulasan makanan dapat dianalisis secara efektif. Selain itu, penelitian ini juga membuka peluang untuk inovasi dalam penggunaan teknologi PySpark dalam analisis data besar.

1.4.4 Manfaat bagi pengembangan ilmu

Penelitian ini memperkaya pengetahuan tentang analisis ulasan makanan dan membantu mengembangkan metode yang lebih baik dalam memahami konteks dan sentimen dalam teks. Penelitian ini juga berperan dalam mengembangkan model klasifikasi teks yang akurat dan efisien. Selain itu, penelitian ini juga memberikan kontribusi dalam pengembangan ilmu analisis sentimen. Dengan menganalisis sentimen yang terkandung dalam ulasan makanan, penelitian ini memperkaya pemahaman tentang evaluasi dan perasaan pelanggan terhadap produk makanan. Ini berpotensi diterapkan dalam berbagai konteks, termasuk analisis opini pelanggan dan analisis sosial media.

2. BAB II: TINJAUAN PUSTAKA

2.1 Teori Penunjang

2.1.1 E-Commerce

E-Commerce (Electronic Commerce) merupakan proses transaksi jual dan beli barang dan jasa melalui alat elektronik. Menurut Shim et al. (2000) dalam Suyanto (2003) e-commerce adalah konsep baru yang bisa digambarkan sebagai proses jual-beli barang atau jasa pada World Wide Web Internet. E-Commerce mencakup distribusi, penjualan, pembelian, marketing, hingga service dari suatu produk yang terjadi dalam suatu sistem jaringan komputer. Saat ini, e-commerce dapat menawarkan berbagai toko di mana pembeli dapat mengakses berbagai macam produk, melakukan pemesanan, kemudian memilih metode pengiriman dan pembayaran secara online.

2.1.2 Analisis Sentimen

Analisis sentimen merupakan suatu metode yang dilakukan untuk memproses data tekstual baik dokumen atau opini sehingga dapat mengartikan apakah sentimen tersebut memiliki makna positif atau negatif. Tujuan dari dilakukannya analisis sentimen adalah untuk mengetahui cara orang lain dalam merespons dan beropini pada suatu topik. Metode analisis sentimen dapat menggunakan algoritma dan teknik machine learning untuk mengklasifikasikan teks ke dalam kategori yang sesuai.

2.1.3 Klasifikasi Teks

Klasifikasi teks adalah salah satu pengaplikasian Text Mining dalam pengelompokkan data (Li, *et al.,* 2016). Klasifikasi teks merupakan suatu proses mengelompokkan data ke dalam kelas atau kategori yang sebelumnya telah ditentukan agar dapat digunakan untuk memprediksi kelas dari data-data yang kelasnya belum diketahui. Klasifikasi teks dibagi menjadi dua jenis, yaitu *supervised* dan *unsupervised*. Klasifikasi *supervised* digunakan pada teks yang telah memiliki kelas pada *data training*. Sedangkan klasifikasi *unsupervised* tidak menggunakan label kelas pada *data training* untuk menganalisis hubungan antara dua kata.

2.1.4 Apache Spark

Apache Spark adalah teknologi perangkat lunak analisis terpadu super cepat yang dirancang untuk memproses data yang berjumlah besar. Fitur utama dari Apache Spark adalah komputasi klaster pada memory yang meningkatkan kecepatan dari pemrosesan aplikasi. Apache Spark dirancang untuk mengetahui dan mengatasi kelemahan dari sistem pemrosesan data tradisional agar data dapat diolah dengan cepat dan efisien.

2.2 Penelitian Terkait

2.2.1 *Sentiment Classification for Amazon Fine Foods Reviews Using PySpark* (T. R. Aravindan dkk, 2023)

Penelitian ini bertujuan untuk melakukan sentimen analisis pada data ulasan makanan pada situs Amazon dalam skala besar menggunakan Apache Spark, Support Vector Machine (SVM), Naive Bayes, dan Regresi Logistik. Dari

penelitian tersebut didapatkan hasil bahwa SVM jauh lebih baik daripada Naive Bayes dan Regresi Logistik. Selain itu didapatkan bahwa analisis sentimen melalui *Big Data Framework* lebih efisien dibandingkan dengan analisis sentimen tanpa *MapReduce Framework.*

2.2.2 Sistem Analisis Sentimen pada Ulasan Produk Menggunakan Metode Naive Bayes (Billy Gunawan dkk, 2018)

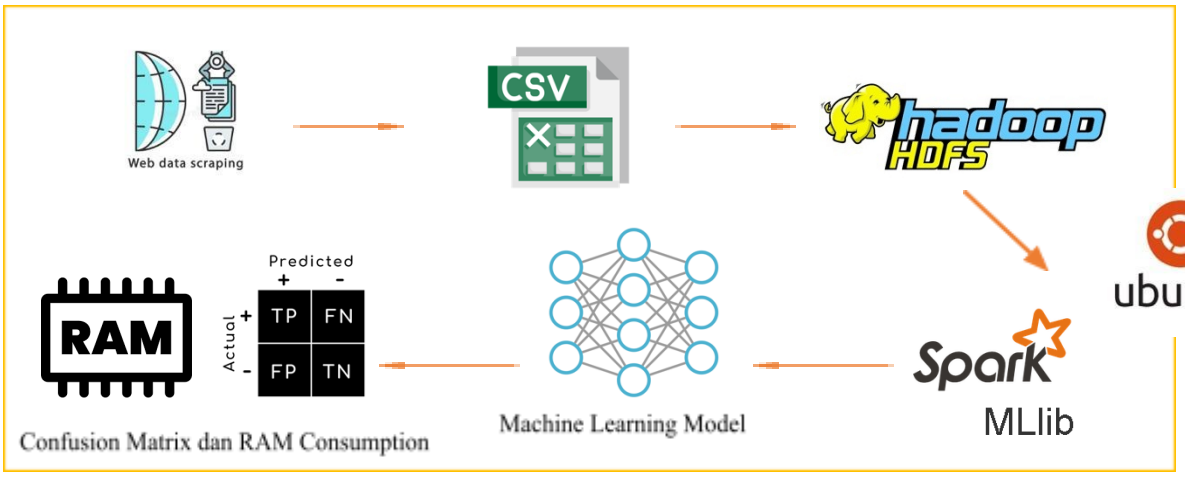
Penelitian ini dilakukan untuk memperoleh informasi sentimen dari ulasan produk online Bahasa Indonesia dengan melakukan proses analisis otomatis. Data diklasifikasikan dengan metode Naive Bayes menjadi 5 kelas, kemudian dievaluasi menggunakan *confusion matrix*. Hasil menunjukkan bahwa pada pengujian 5 kelas hasil terbaik terdapat pada *splitting data* 90% data latih dan 10% data uji dengan nilai akurasi sebesar 59,33%, *recall* sebesar 58,33%, dan *precision* sebesar 59,33%.

2.2.3 Pemanfaatan Spark untuk Analisis Sentimen Mengenai Netralitas Berita dalam Membahas Pemilu Presiden 2019 Menggunakan Metode *Naïve Bayes Classifier* (Reza Aprilliana Fauzi dkk, 2021)

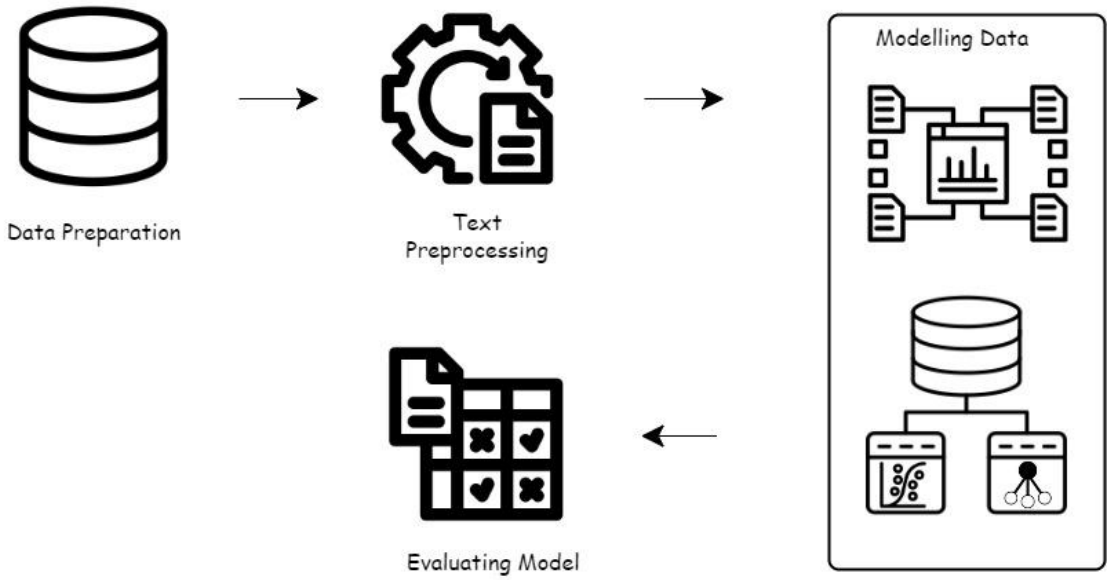
Penelitian ini melakukan analisis sentimen terhadap berita dari berbagai portal berita di Indonesia yang membahas tentang Pemilihan Umum (Pemilu) tahun 2019 menggunakan metode *Naive Bayes Classifier*. Penelitian ini memanfaatkan *Resilient Distributed Dataset* (RDD) yang dimiliki oleh Spark dalam mengklasifikasikan sentimen berita sebagai tipe data. Hasil penelitian menunjukkan bahwa nilai evaluasi terbaik terdapat pada pengujian menggunakan *K-Fold Cross Validation* dengan nilai k = 100. Pada *fold* ke-8 didapatkan hasil dengan nilai akurasi sebesar 100%, *precision* sebesar 100%, *recall* sebesar 100%, dan *F-Measure* sebesar 100%.

3. BAB III: METODOLOGI PENELITIAN

Bab III mencakup rancangan desain sistem dari proyek yang diusulkan mulai dari input-proses-output. Rancangan desain sistem dapat direpresentasikan dalam notasi *flowchart* maupun diagram secara umum. Dalam bab III ini juga mencakup **subbab** tiap-tiap proses pada desain sistem yang dirancang.



Gambar 1. Contoh Desain sistem yang diusulkan dalam proyek



3.1 Data Preparation

Data Preparation adalah proses persiapan data sebelum dilakukan analisis atau penggunaan dalam pemodelan. Tujuan dari Data Preparation adalah untuk mengoptimalkan kualitas data dengan membersihkan, mengubah, dan memformat data agar sesuai dengan kebutuhan analisis atau model yang akan digunakan. Data Preparation memainkan peran yang krusial dalam memastikan bahwa data yang digunakan dalam analisis atau model memiliki kualitas yang baik, sehingga dapat menghasilkan hasil yang akurat dan dapat diandalkan.

3.1.1 Dataset

Dataset ini terdiri dari ulasan makanan yang baik dari Amazon. Data ini mencakup periode lebih dari 10 tahun, termasuk semua ~500.000 ulasan hingga Oktober 2012. Ulasan

tersebut mencakup informasi produk dan pengguna, peringkat, dan ulasan teks biasa. Ini juga mencakup ulasan dari semua kategori lain di Amazon. Dataset ini berisi lebih dari setengah juta ulasan makanan dari Amazon yang mencakup lebih dari 10 tahun, dengan informasi tentang pengguna, produk, peringkat, dan ulasan teks. Terdapat juga beberapa pengguna yang memberikan lebih dari 50 ulasan dalam dataset ini.

3.1.2 Menentukan Sentimen Positif dan Negatif

Menetapkan sentimen positif dan negatif adalah proses untuk mengidentifikasi dan menganalisis emosi atau pendapat yang terkandung dalam suatu teks. Tujuan utama adalah untuk menentukan apakah teks tersebut mengandung sentimen yang positif, negatif, atau netral. Sentimen positif mencerminkan perasaan senang, puas, atau setuju, sedangkan sentimen negatif mencerminkan perasaan marah, kecewa, atau tidak setuju. Dengan memahami sentimen yang terkandung dalam teks, kita dapat memperoleh wawasan yang berharga untuk menginformasikan keputusan bisnis, pengembangan produk, atau pemahaman umum tentang opini publik.

3.1.3 Memfilter ulasan netral

Memfilter ulasan netral adalah proses menyaring atau mengidentifikasi ulasan yang tidak memiliki sentimen yang kuat atau jelas. Ulasan netral seringkali tidak mengekspresikan perasaan yang tegas baik positif maupun negatif terhadap suatu produk, layanan, atau pengalaman. Dalam konteks analisis sentimen, memfilter ulasan netral dapat membantu kita fokus pada ulasan yang memiliki opini yang lebih kuat, sehingga dapat mendapatkan pemahaman yang lebih baik tentang preferensi, kepuasan, atau kekecewaan.

3.1.4 Menetapkan peringkat biner

Menetapkan peringkat biner sebagai variabel target dengan menggunakan angka 1 untuk menyatakan sentimen positif dan angka 0 untuk menyatakan sentimen negatif. Tujuan utamanya adalah untuk mengklasifikasikan teks atau data dengan mengidentifikasi apakah mereka mengandung pandangan atau opini yang positif atau negatif. Dengan mengubah sentimen menjadi peringkat biner, metode ini memungkinkan pemodelan prediktif dan algoritma pembelajaran mesin untuk mengenali dan memproses sentimen dengan lebih efisien. Hasilnya, informasi dapat diekstraksi secara lebih terstruktur dan digunakan untuk tujuan seperti pemantauan brand, analisis ulasan produk, dan penelitian pasar.

3.2 Text Preprocessing

Text pre-processing adalah proses persiapan teks sebelum dianalisis menggunakan algoritma pemrosesan bahasa alami atau metode pembelajaran mesin. Tujuan utama dari pre-processing adalah mengubah teks mentah menjadi bentuk yang lebih terstruktur dan dapat dipahami oleh komputer. Beberapa langkah dalam pre-processing termasuk penghapusan tanda baca, tokenisasi (membagi teks menjadi kata-kata individu), mengubah semua huruf menjadi huruf kecil atau huruf besar, menghapus kata-kata yang tidak penting (stop words), serta mengatasi masalah kata-kata yang sama tetapi ditulis dengan cara yang berbeda (stemming atau lemmatisasi). Dengan melakukan pre-processing, data teks menjadi lebih siap untuk dianalisis dan menghasilkan hasil yang lebih akurat dan konsisten.

3.2.1 Membuat fungsi UDF

Membuat fungsi UDF untuk pemrosesan teks untuk membersihkan teks agar lebih mudah diproses dan dianalisis. Pertama, teks akan dikonversi ke huruf kecil untuk memastikan konsistensi dalam analisis. Selanjutnya, tanda baca seperti titik, koma,

tanda tanya, dan lainnya akan dihapus karena tidak memberikan kontribusi signifikan dalam pemrosesan teks. Kata-kata alfanumerik, yaitu kata-kata yang terdiri dari huruf dan angka, juga akan dihapus karena cenderung tidak relevan dalam analisis. Terakhir, stop words, yang merupakan kata-kata umum seperti "dan", "atau", dan "yang", akan dihapus karena tidak memberikan makna yang signifikan dalam pemrosesan teks. Dengan menggunakan fungsi UDF ini, proses pemrosesan teks menjadi lebih efektif dan fokus pada informasi yang lebih relevan.

3.2.2 POS tagging

POS tagging adalah metode yang digunakan untuk mengidentifikasi dan menandai kelas kata dalam sebuah kalimat, seperti verba, nomina, adjektiva, dan lainnya. Dengan menerapkan POS tagging, kita dapat memperoleh informasi yang lebih detail tentang struktur dan komposisi kalimat-kalimat dalam review makanan di Amazon. Hal ini memungkinkan kita untuk melakukan analisis sentimen yang lebih akurat dan memahami konteks serta makna dari setiap kata dalam teks yang diklasifikasikan.

3.2.3 Text Lemmatization

Text Lemmatization digunakan untuk mengoptimalkan pemrosesan teks. Proses ini melibatkan mengubah kata-kata menjadi bentuk dasar mereka yang disebut lemma. Dengan demikian, kata-kata dengan bentuk yang berbeda seperti bentuk kata kerja, kata benda, atau kata sifat dapat direpresentasikan dengan bentuk yang sama. Teknik ini membantu dalam mengurangi dimensi kata-kata dalam teks dan meningkatkan akurasi dan efisiensi dalam menganalisis sentimen dan melakukan klasifikasi teks review makanan.

3.3 Modelling Data

Pemodelan data adalah proses pengorganisasian, pemrosesan, dan analisis data untuk menghasilkan pemahaman yang lebih baik tentang pola, hubungan, dan informasi yang terkandung dalam data tersebut. Pemodelan data melibatkan penggunaan metode statistik, teknik pengolahan data, dan algoritma untuk menggali wawasan berharga dari data yang ada. Tujuan utama pemodelan data adalah untuk mengidentifikasi tren, membuat prediksi, mengoptimalkan keputusan, dan memberikan pemahaman yang mendalam tentang fenomena yang terjadi di dalam data. Dalam konteks analisis sentimen dan klasifikasi teks review maknan di Amazon, pemodelan data digunakan untuk memahami dan mengklasifikasikan ulasan pengguna berdasarkan sentimen, makna, dan konten yang terkait.

3.3.1 Membuat Dataset Final dengan kolom apt

Pembuatan dataset final dengan kolom apt adalah proses mengumpulkan dan menyusun data yang relevan ke dalam satu set data yang lengkap dan akurat. Tujuan dari pembuatan dataset ini adalah untuk memberikan informasi yang lebih lengkap dan terperinci, sehingga dapat digunakan untuk analisis, pemodelan, atau pengambilan keputusan.

3.3.2 Membagi data training dan data testing

Membagi data menjadi dua, yaitu data pelatihan (training data) dan data uji (test data), serta melakukan beberapa langkah pra-pemrosesan pada kedua set data tersebut. Tujuannya adalah untuk mempersiapkan data yang akan digunakan dalam

pembelajaran mesin. Data yang telah dibagi menjadi training data dan test data akan diubah formatnya sehingga sesuai dengan kebutuhan pemodelan.

3.3.3 Tokenisasi dan membuat matriks TF-IDF

Melakukan tokenisasi pada set data training dan membuat matriks TF-IDF menggunakan metode HashingTF. Untuk mempersiapkan data teks pada set pelatihan agar dapat digunakan dalam pemodelan atau analisis lebih lanjut dengan metode TF-IDF. Metode TF-IDF digunakan untuk mengekstraksi fitur-fitur penting dari teks dengan memberikan bobot yang lebih tinggi pada kata-kata tertentu. Dengan melakukan tokenisasi dan menghasilkan matriks TF-IDF, dapat membantu dalam mempersiapkan data teks agar dapat digunakan dalam pemodelan dan analisis berikutnya.

3.3.4 Model Klasifikasi

Pada penelitian kali ini model klasifikasi yang digunakan adalah regresi logistic dan naive bayes. Regresi logistik adalah teknik yang penting di bidang kecerdasan buatan dan *machine learning* (AI/ML). Model ML adalah program perangkat lunak yang dapat Anda latih untuk melakukan tugas pemrosesan data rumit tanpa campur tangan manusia. Model ML yang dibangun menggunakan regresi logistik membantu organisasi mendapatkan wawasan yang dapat ditindaklanjuti dari data bisnis mereka. Mereka dapat menggunakan wawasan ini untuk analisis prediktif untuk mengurangi biaya operasional, meningkatkan efisiensi, dan menskalakan dengan lebih cepat. Misalnya, bisnis dapat mengungkap pola yang meningkatkan retensi karyawan atau mengarah pada desain produk yang lebih menguntungkan.

Naive Bayes adalah metode yang cocok untuk klasifikasi biner dan multiclass. Metode yang juga dikenal sebagai Naive Bayes Classifier ini menerapkan teknik supervised klasifikasi objek di masa depan dengan menetapkan label kelas ke instance/catatan menggunakan probabilitas bersyarat. Probabilitas bersyarat adalah ukuran peluang suatu peristiwa yang terjadi berdasarkan peristiwa lain yang telah (dengan asumsi, praduga, pernyataan, atau terbukti) terjadi. model machine learning yang diterapkan pada Naive Bayes menggunakan teorema Bayes yang dirumuskan sebagai berikut:

$$P(A \mid B) = P(B \mid A)P(A)P(B)$$

Keterangan:

$P(A \mid B)$: Probabilitas A terjadi dengan bukti bahwa B telah terjadi (probabilitas superior)

$P(B \mid A)$: Probabilitas B terjadi dengan bukti bahwa A telah terjadi

$P(A)$: Peluang terjadinya A

$P(B)$: Peluang terjadinya B

3.4 Evaluasi Model

Evaluasi performa dengan membandingkan prediksi model dengan label sebenarnya pada data pengujian. Akurasi dihitung sebagai jumlah prediksi yang benar dibagi dengan jumlah total sampel dalam dataset pengujian. Dengan membandingkan akurasi dari kedua model yang digunakan, kita dapat mengidentifikasi model yang memiliki kinerja terbaik dalam melakukan klasifikasi pada dataset. Melalui evaluasi ini, sehingga dapat membuat keputusan yang lebih baik dan memilih model yang paling sesuai dengan tujuan dan kebutuhan analisis.

4. BAB IV: HASIL DAN PEMBAHASAN

Data Preparation

1. Dataset

   Dataset dalam studi kasus memiliki 568454 baris dan 10 kolom. Dataset ini terdiri dari ulasan makanan yang baik dari Amazon. Ulasan tersebut mencakup informasi produk dan pengguna, peringkat, dan ulasan teks biasa. Ini juga mencakup ulasan dari semua kategori lain di Amazon. Sebelum mulai melakukan analisis, terlebih dahulu dibutuhkan untuk import library.

4.1 Import library yang diperlukan

```
!pip install pyspark
```

```
import pyspark

from pyspark import SparkConf
from pyspark import SparkContext
from pyspark import HiveContext
from pyspark.sql.functions import *
from pyspark.sql.types import StringType
from pyspark.sql.functions import udf

import re
import string
import nltk
from nltk.stem.wordnet import WordNetLemmatizer
```

```
from nltk.corpus import stopwords
from nltk import pos_tag

%matplotlib inline
import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
import numpy as np
import string

from pyspark import SparkContext
from pyspark import SparkConf
from pyspark import SQLContext

from pyspark import HiveContext
```

Gambar x. Potongan script Import Library

Kode di atas adalah contoh kode yang digunakan untuk menginstal library PySpark dan mengimpor modul dan fungsi yang diperlukan untuk melakukan pemrosesan teks menggunakan PySpark. Berikut adalah penjelasan untuk setiap bagian kode:

!pip install pyspark: Ini adalah sintaks magic pada lingkungan notebook Python seperti Jupyter Notebook atau Google Colab. Sintaks ini digunakan untuk menginstal library PySpark menggunakan pip.

import pyspark: Ini mengimpor modul pyspark yang diperlukan untuk menggunakan PySpark.
from pyspark import SparkConf: Ini mengimpor kelas SparkConf yang digunakan untuk mengkonfigurasi Spark.
from pyspark import SparkContext: Ini mengimpor kelas SparkContext yang merupakan pintu masuk utama untuk berinteraksi dengan Spark.
from pyspark import HiveContext: Ini mengimpor kelas HiveContext yang digunakan untuk berinteraksi dengan Apache Hive.
from pyspark.sql.functions import *: Ini mengimpor semua fungsi dari modul pyspark.sql.functions. Fungsi ini digunakan untuk melakukan operasi pada DataFrame.

from pyspark.sql.types import StringType: Ini mengimpor kelas StringType yang digunakan untuk menentukan tipe data kolom dalam DataFrame.

from pyspark.sql.functions import udf: Ini mengimpor fungsi udf (user-defined function) yang digunakan untuk membuat fungsi kustom yang dapat diterapkan pada kolom DataFrame.

import re: Ini mengimpor modul re (regular expression) yang digunakan untuk manipulasi dan pencocokan pola teks.

import string: Ini mengimpor modul string yang berisi kumpulan konstanta karakter seperti huruf kecil, huruf besar, dan tanda baca.

import nltk: Ini mengimpor modul nltk (Natural Language Toolkit) yang digunakan untuk pemrosesan bahasa alami.

from nltk.stem.wordnet import WordNetLemmatizer: Ini mengimpor kelas WordNetLemmatizer dari modul nltk.stem.wordnet yang digunakan untuk melakukan lemmatisasi kata dalam bahasa Inggris.

from nltk.corpus import stopwords: Ini mengimpor modul stopwords dari nltk.corpus yang berisi daftar kata-kata yang umumnya dianggap tidak berarti dalam analisis teks dan sering dihapus.

from nltk import pos_tag: Ini mengimpor fungsi pos_tag dari modul nltk yang digunakan untuk melakukan penandaan pos (part-of-speech) kata-kata dalam kalimat.

%matplotlib inline: Ini adalah sintaks magic yang digunakan untuk menampilkan plot matplotlib di dalam notebook.

import pandas as pd: Ini mengimpor modul pandas sebagai pd yang digunakan untuk manipulasi dan analisis data.

import numpy as np: Ini mengimpor modul numpy sebagai np yang digunakan untuk operasi numerik.

import matplotlib.pyplot as plt: Ini mengimpor modul pyplot dari matplotlib sebagai plt yang digunakan untuk membuat plot dan visualisasi data.

from pyspark import SparkContext: Ini mengimpor kelas SparkContext yang merupakan pintu masuk utama untuk berinteraksi dengan Spark.

from pyspark import SQLContext: Ini mengimpor kelas SQLContext yang digunakan untuk berinteraksi dengan DataFrame dalam PySpark.

```
sc = SparkContext.getOrCreate(SparkConf().setMaster("local[*]"))
sqlContext=HiveContext(sc)
```

4.2 Memuat dataset

```
df1 =sqlContext.read.format('com.databricks.spark.csv')\
.options(header='true',inferschema='true')\
.load("/content/Reviews.csv")
```

Gambar x. Potongan script memuat dataset

Pada Script di atas, dataset dengan format .csv yang telah kita unduh sebelumnya akan kita muat kedalam dataframe dengan variabel "df1". DataFrame ini nantinya dapat digunakan untuk melakukan manipulasi, analisis, dan pemrosesan data menggunakan Apache Spark.

4.3 Data preparation

```
df1.count()
df1.show()
df=df1
```

(menghitung jumlah data)

Pada script di atas, dataset kita hitung ada berapa banyak baris di dalamya. Ternyata dataset yang digunakan mempunyai data sebanyak 568454. Lalu juga variabel yang awalnya "df" diubah menjadi "df"

Sebelum lanjut untuk melakukan analisis, perlu untuk memeriksa apakah ada missing value dan menanganinya. Missing value adalah hilangnya beberapa data dari suatu dataset. Biasanya missing value terjadi akibat kesalahan pengisian data, kesalahan pengukuran, atau sifat alami data yang tidak diketahui atau tidak dapat diukur.

```
variable_name="Text"
# Menghitung jumlah missing value pada variabel tertentu
missing_count = df.select(variable_name) \
    .where(col(variable_name).isNull() | isnan(variable_name)) \
    .count()

# Menampilkan jumlah missing value
print("Jumlah missing value pada variabel", variable_name, "adalah",
missing_count)
```

(memeriksa missing value)

Guna menghindari bias, kesalahan estimasi, atau hasil yang tidak akurat yang berpengaruh besar terhadap performa algoritma, maka butuh dilakukan penghapusan

atau pengisian missing value. Pada proyek ini, kami menemukan adanya 10 missing value setelah dilakukan pengecekan terhadap nilai null pada kolom “Text”.

```
df = df.dropna()
df.count()
```

(menghapus missing value)

Pada script di atas, missing value yang ditemukan kami tangani dengan menghapuskannya, dikarenakan jika mengisinya, datanya bertipe string atau kalimat yang tidak bisa digantikan dengan kalimat lain secara acak. Penghapusan missing value membuat adanya perubahan jumlah data. Data yang akan dianalisis kini hanya berjumlah 568444.

```
#mengecek tipe data
df.printSchema()
```

(cek tipe data)

Pada script di atas, kita memeriksa apakah tipe-tipe data dalam setiap kolom sudah benar. Karena tipe data yang tidak tepat juga bisa mempengaruhi hasil analisis.

```
from pyspark.sql.functions import col
from pyspark.sql.types import IntegerType
# Mengubah tipe data kolom "score" menjadi integer
df = df.withColumn("Score", col("Score").cast(IntegerType()))
```

(mengubah tipe data)

Karena pada kolom “Score” masih bertipe string, maka harus diubah ke tipe interger. Pengubahan tersebut dilakukan karena skor ini memang berbentuk numerik bukan kalimat.

```
import pyspark.sql.functions as f

df=df.filter((f.col('Score')!=3))
df.count()
```

(menghitung score untuk pelabelan)

```
from pyspark.sql.functions import udf
from pyspark.sql.types import StringType

# Definisikan fungsi untuk menentukan label sentimen
def get_sentiment_label(score):
    if score > 3:
        return "positif"
    elif score < 3:
        return "negatif"
    else:
        return "netral"

# Daftarkan fungsi sebagai User Defined Function (UDF) pada PySpark
```

```
sentiment_udf = udf(get_sentiment_label, StringType())

# Menerapkan UDF pada kolom "score" dalam DataFrame
df = df.withColumn("sentiment", sentiment_udf("Score"))
df.show(4)
```

(menentukan label sentimen)

Kode script tersebut untuk menentukan label sentimen berdasarkan skornya, dengan kriteria atau ambang batas yang menentukan label positif, negatif, atau netral. Disini skor ada pada skala 1-5. Skala 1-2 menunjukkan label negatif, skala 3 netral, dan 4-5 berlabel positif.

Mengubah kolom sentimen menjadi biner dengan nilai 0 dan 1 untuk menyederhanakan representasi sentimen menjadi dua kategori utama

```
def toBinary(score):
    if score >= 3: return 1
    else: return 0
udfScoretoBinary=udf(toBinary, StringType())

df = df.withColumn("Target", udfScoretoBinary("Score"))
print(df.show(4))
print(df.count())
```

(mengubah kolom sentimen menjadi biner)

Dalam pendekatan biner, sentimen negatif direpresentasikan dengan angka 0. Sehingga skor dengan skala 1-2 akan menjadi angka 0. Sedangkan sentimen positif dalam pendekatan biner biasanya direpresentasikan dengan angka 1. Sehingga skor dengan skala 3-5 akan diatur menjadi angka 1.

4.4 Preprocessing

```
import nltk
nltk.download('stopwords')
```

(Menginstall stopwords)

Sebelum tahap pemodelan data dilakukan preprocessing untuk menghilangkan permasalahan yang dapat mengganggu proses analisis.

```
#TEXT Pre-processing

##COnvert to lower
from pyspark.sql.functions import udf
from pyspark.sql.types import *

def lower(text):
    return text.lower()

lower_udf =udf(lower,StringType())
```

```
##Remove nonAscii
def strip_non_ascii(data_str):
#''' Returns the string without non ASCII characters'''
    stripped = (c for c in data_str if 0 < ord(c) < 127)
    return ''.join(stripped)
# setup pyspark udf function
strip_non_ascii_udf = udf(strip_non_ascii, StringType())

##FIx abbreviations
def fix_abbreviation(data_str):
    data_str = data_str.lower()
    data_str = re.sub(r'\bthats\b', 'that is', data_str)
    data_str = re.sub(r'\bive\b', 'i have', data_str)
    data_str = re.sub(r'\bim\b', 'i am', data_str)
    data_str = re.sub(r'\bya\b', 'yeah', data_str)
    data_str = re.sub(r'\bcant\b', 'can not', data_str)
    data_str = re.sub(r'\bdont\b', 'do not', data_str)
    data_str = re.sub(r'\bwont\b', 'will not', data_str)
    data_str = re.sub(r'\bid\b', 'i would', data_str)
    data_str = re.sub(r'wtf', 'what the fuck', data_str)
    data_str = re.sub(r'\bwth\b', 'what the hell', data_str)
    data_str = re.sub(r'\br\b', 'are', data_str)
    data_str = re.sub(r'\bu\b', 'you', data_str)
    data_str = re.sub(r'\bk\b', 'OK', data_str)
    data_str = re.sub(r'\bsux\b', 'sucks', data_str)
    data_str = re.sub(r'\bno+\b', 'no', data_str)
    data_str = re.sub(r'\bcoo+\b', 'cool', data_str)
    data_str = re.sub(r'rt\b', '', data_str)
    data_str = data_str.strip()
    return data_str

##Remove punctuations mentions and alphanumeric characters
def remove_features(data_str):
# compile regex
    url_re = re.compile('https?://(www.)?\w+\.\w+(/\w+)*/?')
    punc_re = re.compile('[%s]' % re.escape(string.punctuation))
    num_re = re.compile('(\\d+)')
    mention_re = re.compile('@(\w+)')
    alpha_num_re = re.compile("^[a-z0-9_.]+$")
# convert to lowercase
    data_str = data_str.lower()
# remove hyperlinks
    data_str = url_re.sub(' ', data_str)
# remove @mentions
    data_str = mention_re.sub(' ', data_str)
# remove puncuation
    data_str = punc_re.sub(' ', data_str)
```

```
# remove numeric 'words'
    data_str = num_re.sub(' ', data_str)
# remove non a-z 0-9 characters and words shorter than 1 characters
    list_pos = 0
    cleaned_str = ''
    for word in data_str.split():
        if list_pos == 0:
            if alpha_num_re.match(word):
                cleaned_str = word
            else:
                cleaned_str = ' '
        else:
            if alpha_num_re.match(word):
                cleaned_str = cleaned_str + ' ' + word
            else:
                cleaned_str += ' '
        list_pos += 1
# remove unwanted space, *.split() will automatically split on
# whitespace and discard duplicates, the " ".join() joins the
# resulting list into one string.
    return " ".join(cleaned_str.split())
# setup pyspark udf function

##Remove stop words
def stopwords_delete(word_list):
        from nltk.corpus import stopwords
        filtered_words=[]
        return word_list

# Part-of-Speech Tagging
def tag_and_remove(data_str):
    cleaned_str = ' '
# noun tags
    nn_tags = ['NN', 'NNP', 'NNP', 'NNPS', 'NNS']
# adjectives
    jj_tags = ['JJ', 'JJR', 'JJS']
# verbs
    vb_tags = ['VB', 'VBD', 'VBG', 'VBN', 'VBP', 'VBZ']
    nltk_tags = nn_tags + jj_tags + vb_tags
# break string into 'words'
    text = data_str.split()
# tag the text and keep only those with the right tags
    tagged_text = pos_tag(text)
    for tagged_word in tagged_text:
        if tagged_word[1] in nltk_tags:
            cleaned_str += tagged_word[0] + ' '
    return cleaned_str
```

```
##Lemmatization
def lemmatize(data_str):
# expects a string
    list_pos = 0
    cleaned_str = ''
    lmtzr = WordNetLemmatizer()
    text = data_str.split()
    tagged_words = pos_tag(text)
    for word in tagged_words:
        if 'v' in word[1].lower():
            lemma = lmtzr.lemmatize(word[0], pos='v')
        else:
            lemma = lmtzr.lemmatize(word[0], pos='n')
        if list_pos == 0:
            cleaned_str = lemma
        else:
            cleaned_str = cleaned_str + ' ' + lemma
        list_pos += 1
    return cleaned_str
```

(Text preprocessing)

Dari script diatas dilakukan beberapa langkah

```
lower_udf =udf(lower,StringType())
strip_non_ascii_udf = udf(strip_non_ascii, StringType())
fix_abbreviation_udf = udf(fix_abbreviation, StringType())
remove_features_udf = udf(remove_features, StringType())
remove_stops_udf = udf(stopwords_delete, StringType())
tag_and_remove_udf = udf(tag_and_remove, StringType())
lemmatize_udf = udf(lemmatize, StringType())
```

(deklarasi User-Defined Function (UDF) menggunakan PySpark)

Pada script di atas, fungsi `lower` ini digunakan untuk mengonversi teks menjadi huruf kecil (lowercase). Hal ini dilakukan agar tidak ada perbedaan dalam pengolahan kata, jadi kata-kata dengan huruf besar dan kecil dianggap sama.

Fungsi `strip_non_ascii` digunakan untuk menghapus karakter non-ASCII dari teks. Karena seringkali karakter non-ASCII tidak relevan dalam pemrosesan teks, sehingga lebih baik dihapus.

Untuk memperbaiki beberapa singkatan atau kata-kata umum yang sering digunakan dalam bahasa sehari-hari menggunakan fungsi `fix_abbreviation`. Misalnya, mengganti "thats" menjadi "that is", "ive" menjadi "i have", dan sebagainya. Hal itu dilakukan agar memperoleh representasi teks yang lebih standar dan mudah dipahami.

Fungsi `stopwords_delete` ini seharusnya menghapus stopwords (kata-kata umum yang tidak memberikan banyak informasi) dari teks. Namun, dalam implementasi tersebut, fungsi ini tidak melakukan apa pun dan hanya mengembalikan daftar kata tanpa perubahan.

Fungsi `tag_and_remove` untuk melakukan part-of-speech (POS) tagging pada teks dan menghapus kata-kata yang tidak termasuk dalam kategori tertentu, seperti kata benda (noun), kata sifat (adjective), dan kata kerja (verb). Tujuannya adalah untuk mempertahankan hanya kata-kata yang dianggap penting dalam konteks analisis teks tertentu.

`lemmatize` digunakan untuk melakukan lemmatisasi pada teks, yaitu mengubah kata-kata menjadi bentuk dasar atau kata dasar. Dalam fungsi ini, lemmatisasi dilakukan berdasarkan jenis kata (part-of-speech) menggunakan WordNetLemmatizer. Misalnya, kata kerja (verb) akan diubah menjadi bentuk dasar infinitif, sedangkan kata benda (noun) akan diubah menjadi bentuk dasar tunggal.

```
df = df.withColumn("lower_text",lower_udf(df["Text"]))
df = df.withColumn("text_non_asci",fix_abbreviation_udf(df["lower_text"]))
df = df.withColumn("fixed_abbrev",fix_abbreviation_udf(df["text_non_asci"]))
df = df.withColumn('removed_features',remove_features_udf(df['fixed_abbrev']))
df.show(5,True)
```

(Text preprocessing pada UDF)

Script di atas dimaksudkan untuk menambahkan kolom dari variabel-variabel yang berisi fungsi yang telah dideklarasikan pada script sebelumnya.

```
df_no_stop_words = df.withColumn("stopwords_delete",
remove_stops_udf("removed_features")).select('Text','stopwords_delete','Target')
df_no_stop_words.show(5)
```

(Menghapus stopwords)

Jadi pada script di atas, stopwords atau kata-kata umum yang tidak memberikan banyak informasi dari kolom "Text" akan dihapuskan. Lalu, dataframe membuat kolom baru yaitu "stopwords_delete", dan menyisakan kolom 'Text','stopwords_delete','Target', sedangkan lainnya dihapuskan.

```
df_pos_tagging=df_no_stop_words.withColumn("tag_and_remove_pos",
tag_and_remove_udf("stopwords_delete")).select('Text','tag_and_remove_pos','Targe
t')
```

(Memberi tag dan menghapus kata tanpa tag)

Setiap nilai pada kolom "stopwords_delete" akan diproses menggunakan fungsi tag_and_remove_udf untuk melakukan tagging (pemberian tag part-of-speech) dan menghapus kata-kata yang tidak memiliki tag yang diinginkan.

```
import nltk
nltk.download('averaged_perceptron_tagger')
```
(unduh dataset yang berisi model POS tagger)

Dengan mengunduh dataset ini, kita dapat menggunakan fungsi tagging berbasis perceptron rata-rata dalam NLTK untuk melakukan penandaan kelas kata (part-of-speech tagging) pada teks dalam bahasa Inggris.

```
#Tokenizing the document
from pyspark.ml.feature import Tokenizer
tokenizer = Tokenizer(inputCol="tag_and_remove_pos", outputCol="words")
wordsDataFrame = tokenizer.transform(df_pos_tagging)
for words_label in wordsDataFrame.select("words", "Target").take(3):
    print(words_label)

df_text =
df.withColumn("text_lower",lower_udf(df["Text"])).select('text_lower','Target')
```
(Proses Tokenisasi)

Jadi, maksud dari script di atas adalah melakukan tokenisasi pada teks dokumen menggunakan Tokenizer dan membuat dataframe baru yang berisi teks yang telah dikonversi menjadi huruf kecil dan kolom "Target".

```
from pyspark.ml.feature import StopWordsRemover
remover = StopWordsRemover(inputCol="words", outputCol="words_filtered")
wordsDataFrame1 =
remover.transform(wordsDataFrame).select("Target","words_filtered")
wordsDataFrame1.show(2)
```
(Menghapus stopwords dari hasil tokenisasi)

Dengan menggunakan StopWordsRemover, script di atas dapat menghapus kata-kata yang dianggap stop words dari teks yang telah ditokenisasi, sehingga memungkinkan untuk fokus pada kata-kata yang lebih informatif dalam analisis teks.

```
import nltk
nltk.download('wordnet')
```
(Unduh modul wordnet)

Setelah unduhan modul wordnet selesai, kita dapat menggunakan modul wordnet untuk melakukan lemmatization pada teks dalam bahasa Inggris.

```
from pyspark.sql.functions import monotonically_increasing_id
# Create Unique ID
df_text_lemma = df_text_lemma.withColumn("uid", monotonically_increasing_id())
df_text_lemma.show(4)
```
(Membuat Id unik)

Dengan menambahkan kolom "uid" yang berisi ID unik pada DataFrame, Anda dapat menggunakan ID ini sebagai referensi unik untuk setiap baris dalam proses-proses berikutnya.

4.5 Modelling Data

```
data = df_text_lemma.select('uid', 'lemmatized_text','Target')
#data=wordsDataFrame2
data.show(4)
```

(Preparing data for modelling)

membuat variabel baru bernama data yang menyimpan kolom 'uid', 'lemmatized_text', 'target' berdasarkan dataframe df_text_lemma yang sebelumnya sudah diolah. lalu menampilkan 4 data teratas dengan perintah show()

```
(trainingData, testData) = data.randomSplit([0.7, 0.3])
```

(Membagi data menjadi data training dan data testing)

data dibagi dengan perbandingan 70 untuk data testing dan 30 untuk data training. data dibagi menggunakan randomsplit

```
# Caching the RDD for training
trainingData
#Renaming features for modeling
training = trainingData.selectExpr("lemmatized_text as text", "Target
as label")
training = training.withColumn("label",
training["label"].cast(DoubleType()))

testData
#Renaming features for modeling
test = testData.selectExpr("lemmatized_text as text", "Target as
label")
test = test.withColumn("label", test["label"].cast(DoubleType()))
```

pada data training dan data testing, Dilakukan penamaan ulang pada kolom lemmatized_text menjadi text dan kolom label menjadi target. tujuannya untuk mempermudah pemanggilan nama. lalu selanjutnya merubah tipe data pada kolom label menjadi doubletype

```
from pyspark.ml.feature import HashingTF
from pyspark.ml.feature import IDF

tokenizer = Tokenizer(inputCol="text", outputCol="words")
hashingTF = HashingTF(inputCol=tokenizer.getOutputCol(),
outputCol="hashing")
idf = IDF(inputCol=hashingTF.getOutputCol(), outputCol="features")
```

mengenerate proses tokenizer pada kolom text dan output kolom menjadi words. mengenerate proses hashingtf dari kolom words menjadi kolom hashing, lalu yang terakhir adalah mengenerate proses IDF pada kolom hashing menjadi kolom features.

'Tokenizer' ; membagi teks menjadi kata-kata terpisah

'HashingTF': mengubah kumpulan fitur data menjadi fitur vektor yang dapat dilakukan pemodelan.

'IDF' : menghitung nilai nilai IDF yang dihasilkan dari hashingtf

```
from pyspark.ml import Pipeline
from pyspark.ml.classification import LogisticRegression
import time

start_time = time.time()
lr = LogisticRegression(maxIter=10, regParam=0.01)
pipeline = Pipeline(stages=[tokenizer, hashingTF, idf, lr])
# Training the model
model = pipeline.fit(training)
end_time = time.time()

training_time = end_time - start_time
print("Waktu pelatihan model: ", training_time, " detik")
#Predicing Output
prediction = model.transform(test)
prediction.select("label", "prediction").show(10,False)
```

(Melakukan pemodelan logistic regression)

Menginisiasi pemodelan logistic regression menjadi lr. lalu membuat pipeline untuk menggabungkan transformasi model klasifikasi ke dalam satu kesatuan dan berurutan. membuat variabel model yang berisi pipeline untuk data training. selanjutnya melakukan prediksi pada data testing dan menampilkan kolom label dan prediction,

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
evaluator =
MulticlassClassificationEvaluator(predictionCol="prediction")
evaluator.evaluate(prediction)
```

(Evaluasi logistic regression)

Melakukan import kelas Multiclassclassification untuk mengevaluasi model klasifikasi pada data yang telah diprediksi. melakukan evaluasi model dengan memanggil model evaluate dan menggunakan data 'prediction' sebagai input.

```
from pyspark.ml.classification import NaiveBayes
from pyspark.ml import Pipeline
```

```
nb = NaiveBayes(smoothing=1.0, modelType="multinomial")
pipeline_nb = Pipeline(stages=[tokenizer, hashingTF, idf, nb])
# Training the model
model_nb = pipeline_nb.fit(training)
#Predicing Output
prediction_nb = model_nb.transform(test)
```

(Melakukan pemodelan Naive Bayes)

Menginisiasi pemodelan naive bayes menjadi nb. lalu membuat pipeline untuk menggabungkan transformasi model klasifikasi ke dalam satu kesatuan dan berurutan. membuat variabel model yang berisi pipeline untuk data training. selanjutnya melakukan prediksi pada data testing dan menampilkan kolom label dan prediction,

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
evaluator_nb =
MulticlassClassificationEvaluator(predictionCol="prediction")
evaluator_nb.evaluate(prediction_nb)
```

(Evaluasi naive bayes)

Melakukan import kelas Multiclassclassification untuk mengevaluasi model klasifikasi pada data yang telah diprediksi. melakukan evaluasi model dengan memanggil model evaluate dan menggunakan data 'prediction' sebagai input.

Penyajian tabel juga dapat ditambahkan di dalam laporan sebagai berikut:

Tabel 1. Contoh Hasil Performa Klasifikasi Pada PySpark

No.	Pemodelan	Ukuran Data	Akurasi	Waktu
1.	Logistic Regression	70:30	91.5%	1 Jam 40 Menit 21 Detik
2.	Naive Bayes	70:30	87.8%	2 jam 2 menit 2 detik

5. BAB V: KESIMPULAN

Hasil klasifikasi dengan algoritma Regresi Logistik memiliki akurasi sebesar 91.5%. Sedangkan akurasi dari algoritma Naive Bayes sebesar 87.8%. berdasarkan lama waktu pemodelannya, metode Regresi Logistik lebih cepat dengan waktu eksekusi 1 jam 40 menit 21 detik. sedangkan naive bayes cenderung lebih lama dengan waktu eksekusi 2 jam 2 menit 2 detik. Artinya berdasarkan tingkat keberhasilan dalam memprediksi dengan benar kelas atau nilai yang sebenarnya dari data yang diuji

dan berdasarkan lama waktu eksekusi, Regresi Logistik adalah algoritma yang lebih tepat untuk digunakan.

6. DAFTAR PUSTAKA

Daftar pustaka menyajikan referensi yang digunakan dan dikutip di dalam bab dan subbab sebelumnya menggunakan *style* APA.

Laras, A. (2023, April 29). *Laba Amazon Melonjak Rp46,9 triliun USAI PHK Ribuan Karyawan*. Bisnis.com.

Suyanto, M. (2003). Strategi Periklanan pada E- Commerce Perusahaan Top Dunia. Yogyakarta : Andi.

Id. (1980). *ID*. Amazon. https://aws.amazon.com/id/what-is/logistic-regression/

Team, A., & Bunga. (2022, July 17). *Naive Bayes, Metode Klasifikasi Algoritma Yang Efektif*. Algoritma. https://algorit.ma/blog/naive-bayes-2022/

Febriyanti, N. F., Handoyo, E., & Adi Soetrisno, Y. A. (2021). SISTEM IMPORTING DAN PROCESSING DATA INSTRUMEN AKREDITASI BERBASIS PYSPARK DAN MYSQL. Transient: Jurnal Ilmiah Teknik Elektro, 10(1), 92-97. https://doi.org/10.14710/transient.v10i1.92-97

7. LAMPIRAN

   Pada bagian lampiran dapat ditambahkan beberapa set data dan tambahan lampiran penunjang yang membantu dalam proses pengerjaan proyek.