APPLICATION OF NLP FOR E-COMMERCE REVIEW SENTIMENT CLASSIFICATION (TOKOPEDIA)

BY:RIFQI ARRAYAN

Profile

Hi! I am a fourth semester student at UPN Veteran Jakarta. I am a fast learner and have a strong interest in data analysis. My experiences in organizations and internships have equipped me with excellent teamwork and communication skills. I enjoy being involved in various projects in the IT sector, and I am constantly looking for opportunities to learn and improve my skills through professional organizations, and currently exploring the data field, especially data science and data analyst.

Background

In today's digital era, e-commerce platforms such as Tokopedia are the main place for people to conduct buying and selling transactions. Users often leave reviews of purchased products, both in positive and negative forms. These reviews have great potential to be analyzed to understand customer satisfaction, improve services, and provide better product recommendations. Therefore, sentiment analysis is one of the important approaches in evaluating public perception of products on the Tokopedia platform.

Problem and Goals

Problem

- How to process user review data from Tokopedia so that it can be used for sentiment analysis?
- What is the distribution of sentiment (positive vs. negative) in product reviews on Tokopedia?
- What method or technique is most suitable to perform sentiment classification from the text reviews?

Goals

- Conduct data exploration to understand review characteristics and sentiment distribution.
- Building a sentiment analysis model that is able to classify Tokopedia product reviews into positive and negative sentiments with high accuracy.
- Finding the optimal classification threshold so that precision and recall are balanced so that the model is more reliable.

Dataset Overview

This dataset contains a collection of product reviews given by Tokopedia users. Each review data contains a review text and a sentiment label indicating whether the review is positive or negative.

Dataset characteristics:

- Amount of data: 580 reviews

Key features:

- Review Text: Customer review text about the product.
- Sentiment Label: Sentiment category, usually 1 for positive and 0 for negative.
- Class distribution: Unbalanced data, with a much larger number of positive reviews (around 250) than negative reviews (around 20).
- Language: Indonesian, with possible variations in the use of words, slang, and informal expressions.

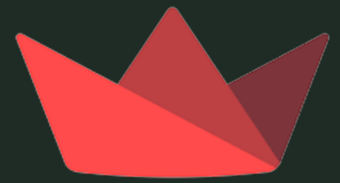
Tools

Google Colab

Microsoft Excel

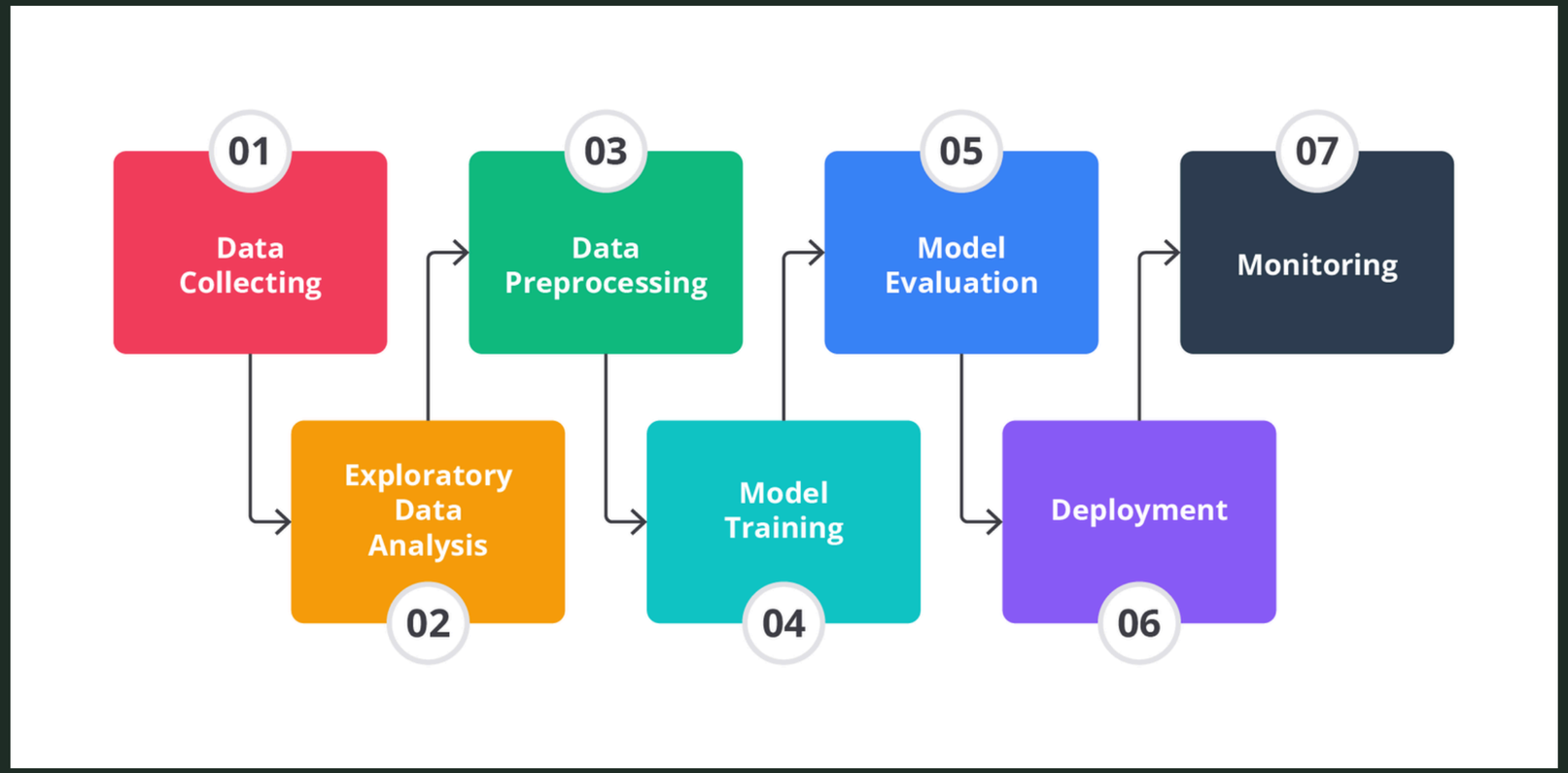
Python

Github



Streamlit

Data processing stages



DATA PREPARATION

Display of Data used

	Unnamed: 0	Nama_Produk	Akun	Ulasan_clean	label
0	1	Masker KF94 4 Ply Isi 10 Pcs Masker Medis Kese...	Stephanus	dua kali beli sayang lambat respon order pagi ...	0
1	2	Masker KF94 4 Ply Isi 10 Pcs Masker Medis Kese...	Yati	bagus kirim cepat	1
2	3	Masker KF94 4 Ply Isi 10 Pcs Masker Medis Kese...	Husin	barang sesuai pesan	1
3	4	Masker KF94 4 Ply Isi 10 Pcs Masker Medis Kese...	Febby	panas pakai	1
4	5	Masker KF94 4 Ply Isi 10 Pcs Masker Medis Kese...	Rahmat	barang sesuai minta bagus komunikasi jual beli...	0

EXPLANATORY DATA ANALYSIS

Check missing value,Data type and amount of data

	0
Unnamed: 0	0
Nama_Produk	0
Akun	0
Ulasan_clean	0
label	0

dtype: int64

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 580 entries, 0 to 579
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   Unnamed: 0    580 non-null    int64 
 1   Nama_Produk   580 non-null    object
 2   Akun          580 non-null    object
 3   Ulasan_clean  580 non-null    object
 4   label         580 non-null    int64 
dtypes: int64(2), object(3)
memory usage: 22.8+ KB
```

```
df.shape
```

```
(580, 5)
```

DATA PREPCESSING

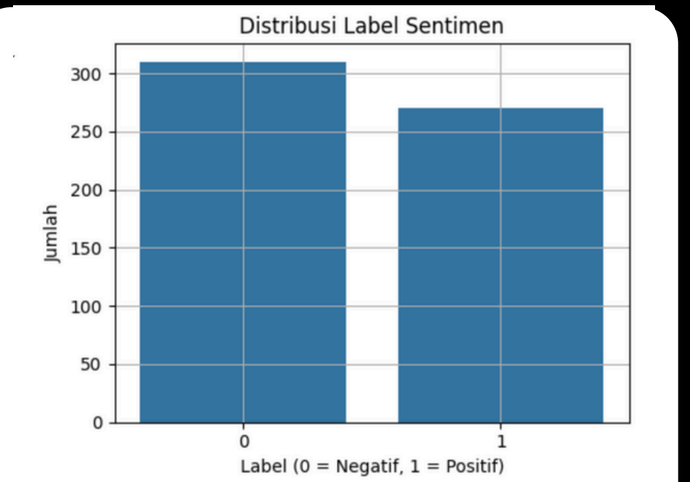
Delete unused columns

```
df.drop(columns=['Unnamed: 0', 'Nama_Produk', 'Akun'], inplace=True)
df.head()
```

	Ulasan_clean	label
0	dua kali beli sayang lambat respon order pagi ...	0
1	bagus kirim cepat	1
2	barang sesuai pesan	1
3	panas pakai	1
4	barang sesuai minta bagus komunikasi jual beli...	0

Delete columns that are not used, namely deleting Unnamed: 0, Product_name, and also Account columns.

View label distribution (positive and negative)



label	count
0	310
1	270

Looking at the distribution of labels (positive and negative) where negative labels totaled 310 while positive labels totaled 270.

Removes stop words and cleans data

```
stopwords_id = {
    'yang', 'untuk', 'dan', 'di', 'ke', 'dari', 'ini', 'itu', 'dengan', 'karena',
    'pada', 'saya', 'ada', 'tidak', 'bisa', 'jadi', 'sudah', 'akan', 'kalau',
    'dalam', 'adalah', 'atau', 'lebih', 'juga', 'oleh', 'sebagai'
}

def clean_text(text):
    text = str(text).lower()
    text = re.sub(r'[^a-z\s]', '', text)
    tokens = text.split()
    tokens = [word for word in tokens if word not in stopwords_id]
    return ' '.join(tokens)

df['Ulasan_clean'] = df['Ulasan_clean'].apply(clean_text)
```

It removes stopwords and also cleans the text from punctuation and unimportant characters so that the text results are cleaner and focus on important words only.

Delete duplicate spaces

```
def rewhitespace(text):
    corrected = str(text)
    corrected = re.sub(r"//t",r"\t", corrected)
    corrected = re.sub(r"( )\1+",r"\1", corrected)
    corrected = re.sub(r"(\n)\1+",r"\1", corrected)
    corrected = re.sub(r"(\r)\1+",r"\1", corrected)
    corrected = re.sub(r"(\t)\1+",r"\1", corrected)
    return corrected.strip(" ")

df['Ulasan_clean'] = df['Ulasan_clean'].apply(rewhitespace)
df.head()
```

remove duplicate spaces so that the model can predict accurately

Conduct tokenization

```
from nltk.tokenize import word_tokenize

def token(text):
  text = word_tokenize(text)
  return text

df['Ulasan_clean'] = df['Ulasan_clean'].apply(token)
df.head()
```

	Ulasan_clean	label
0	[dua, kali, beli, sayang, lambat, respon, orde...	0
1	[bagus, kirim, cepat]	1
2	[barang, sesuai, pesan]	1
3	[panas, pakai]	1
4	[barang, sesuai, minta, bagus, komunikasi, jua...	0

Perform Tokenization, which breaks the text into words so that each element can be analyzed.

Conduct Lemization

```
lemmatizer = WordNetLemmatizer()

def get_wordnet_pos(word):
    """Map POS tag to first character lemmatize() accepts"""
    tag = nltk.pos_tag([word])[0][1][0].upper()
    tag_dict = {
        'J': wordnet.ADJ,
        'N': wordnet.NOUN,
        'V': wordnet.VERB,
        'R': wordnet.ADV
    }
    return tag_dict.get(tag, wordnet.NOUN)


def lemmatize_review(Ulasan_clean):
    return [lemmatizer.lemmatize(word, get_wordnet_pos(word)) for word in Ulasan_clean]

df['Ulasan_clean'] = df['Ulasan_clean'].apply(lemmatize_review)
df.head()
```

Perform Lemization, which is to remove affixes

Conduct steamming

```python
from Sastrawi.Stemmer.StemmerFactory import StemmerFactory

# Buat objek stemmer
factory = StemmerFactory()
stemmer = factory.create_stemmer()

def stem_review(Ulasan_clean):
    return ' '.join([stemmer.stem(word) for word in Ulasan_clean])

df['Ulasan_clean'] = df['Ulasan_clean'].apply(stem_review)
df.head()
```

	Ulasan_clean	label
0	dua kali beli sayang lambat respon order pagi ...	0
1	bagus kirim cepat	1
2	barang sesuai pesan	1
3	panas pakai	1
4	barang sesuai minta bagus komunikasi jual beli...	0

Changing the word to the basic form, after doing the steamer, I merged the text into one unit because TF-IDF requires a format in the form of a string.

Model Training

TF-IDF

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer()
X = tfidf.fit_transform(df['Ulasan_clean'])
y = df['label']
```

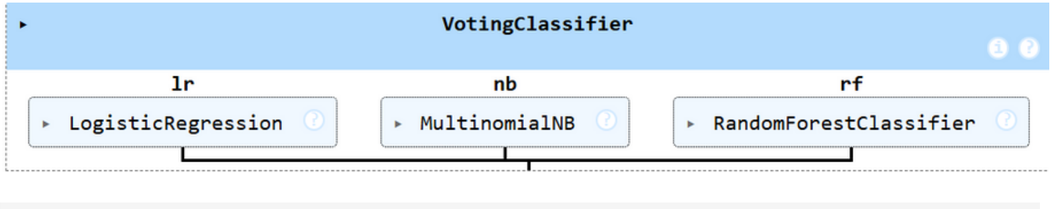
where the sentences contained in the Review_clean column will be converted into TF-IDF Score in the form of numbers that learn from the words that appear.
And Y as the target variable

Conduct splitting data

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=42)
```

Where 20% of the data is used for testing while 80% of the data is carried out training

Ensemble Learning



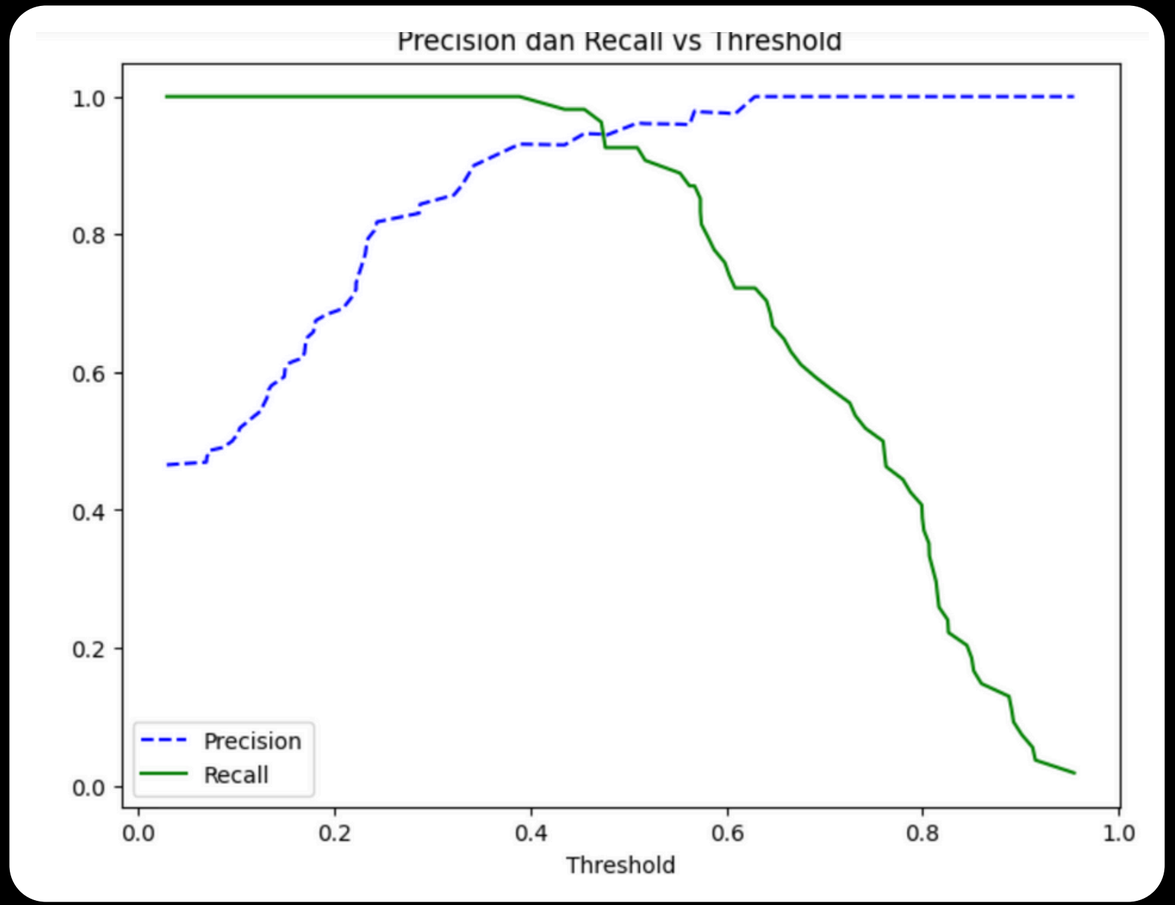

```python
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier, VotingClassifier

# Inisialisasi model voting classifier
model1 = LogisticRegression(max_iter=1000)
model2 = MultinomialNB()
model3 = RandomForestClassifier(n_estimators=100, random_state=42)

ensemble = VotingClassifier(
    estimators=[('lr', model1), ('nb', model2), ('rf', model3)],
    voting='soft' #menggabungkan probabilitas prediksi dari ketiga mo
)

#  Model ensembel dilatih
ensemble.fit(X_train, y_train)
```

Here we use 3 models namely:

- LogisticRegression (linear model)
- MultinomialNB (good for text data, especially bag-of-words/TF-IDF)
- RandomForestClassifier (ensemble tree model)

Model Evaluation

Precision,Recall VS Threshold



Classification report

```
Threshold terbaik berdasarkan F1-score: 0.39
F1-score tertinggi: 0.96

Classification Report dengan Threshold Optimal:
              precision    recall  f1-score   support

           0       1.00      0.94      0.97        62
           1       0.93      1.00      0.96        54

    accuracy                           0.97       116
   macro avg       0.97      0.97      0.97       116
weighted avg       0.97      0.97      0.97       116
```

Analyze Result

- Recall (green line) is high at low threshold → the model captures almost all positive data.
- Precision (dashed blue line) is high at high threshold → the model selects only highly confident positive data.
- The optimal point (threshold = 0.39) is the point where the F1-score (combined precision and recall) is the highest, which is 0.96.
- This means that at this point, the model is good enough at balancing the precision and completeness of its predictions.

Precision:

- For label 0 (negative) = 1.00 → All negative predictions are correct, no negative prediction errors.
- For label 1 (positive) = 0.93 → 93% of predicted positives are actually positive.

Recall (Coverage):

- For label 0 (negative) = 0.94 → Of all negative data, 94% were correctly recognised.
- For label 1 (positive) = 1.00 → All positive data was successfully recognised, no one was missed.

F1-score:

- Combination of precision and recall.
- Very high scores (0.96 and 0.97) → The model handles both classes very well.
- Overall accuracy: 0.97 or 97%
- Out of 116 data, only about 3 were misclassified.

C**onclusion**

- The optimal threshold of 0.39 resulted in a model with excellent and balanced performance.

Model:

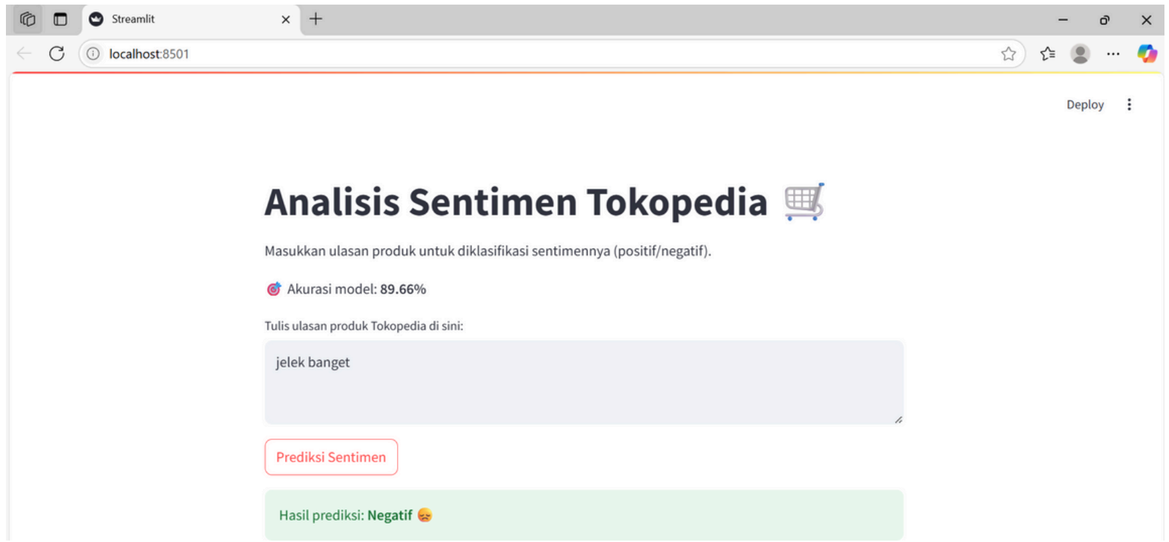
- Highly accurate in recognising negative reviews (not many false positives).
- Excellent at recognising positive reviews (no false negatives).
- The high F1-score (0.96) indicates that the model is suitable for sentiment analysis tasks as precision and recall are balanced.
- It is ideal for use in e-commerce platforms such as Tokopedia, as it can provide a good understanding of customer satisfaction or dissatisfaction.

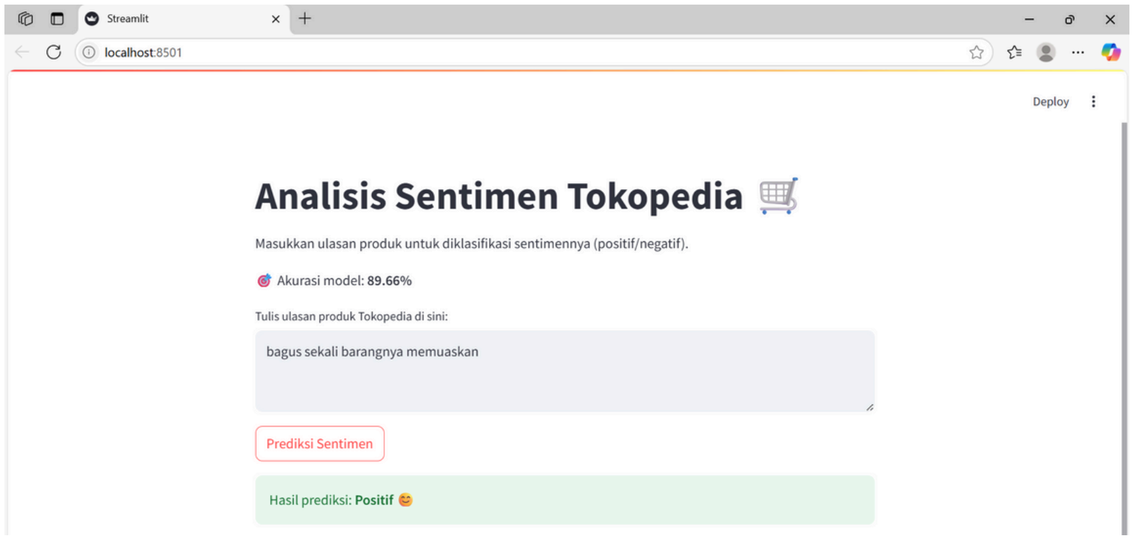
New Review Prediction

```python
# Fungsi prediksi baru
def prediksi_sentimen(teks):
    teks_bersih = clean_text(teks)
    tfidf_teks = tfidf.transform([teks_bersih])
    hasil = ensemble.predict(tfidf_teks)[0]
    return "Positif" if hasil == 1 else "Negatif"

# Contoh prediksi
print(prediksi_sentimen("Pengiriman cepat dan barang bagus banget!"))
print(prediksi_sentimen("Lubang baut tidak pas"))
```

```
Positif
Negatif
```

Deployment on Streamlit





It can be seen that it was deployed using Streamlit, where testing for positive and negative sentiment was carried out and it was proven that the model created could predict correctly

Conclusion

At the data collection and pre-processing stage, Tokopedia product review data is collected and cleaned from irrelevant elements. Columns such as Unnamed: 0, Product_Name, and Account were removed because they do not make a significant contribution to sentiment analysis. The preprocessing process includes removing stopwords (unimportant conjunctions), punctuation, and double spaces. Next, tokenization is carried out to break the sentence into words, as well as lemmatization and stemming to return the words to their basic form. The final results of this preprocessing are combined in a format that is ready to be used by TF-IDF based text representation methods.

At the modeling stage, the cleaned review data is represented using the TF-IDF Vectorizer. The data is then divided into 80% for training and 20% for testing. Three machine learning models are used for sentiment classification, namely Logistic Regression, Multinomial Naive Bayes, and Random Forest Classifier. These three models were chosen because of their good ability to handle text data and produce competitive performance.

In the evaluation stage, it was found that the optimal threshold was 0.39. At this point, the model achieves the best performance with an F1-score of up to 0.96, indicating an excellent balance between precision and recall. The model produces a precision of 93% for the positive class and 100% for the negative class, as well as a recall of 100% for the positive class and 94% for the negative class. Overall, the model was able to achieve an accuracy of 97%, indicating that the model is very reliable in classifying the sentiment of Tokopedia product reviews accurately and consistently.

VIEW IN GITHUB

THANK YOU