JNATIA Volume 4, Nomor 2, Februari 2026 p-ISSN: 2986-3929
Jurnal Nasional Teknologi Informasi dan Aplikasinya e-ISSN: 3032-1948

Klasifikasi Berita pada Sektor Ekonomi Indonesia dari Bisnis.com Menggunakan SVM dan TF-IDF

I Ketut Tangkas Agus Sucita, Made Agung Raharja

Program Studi Informatika, Fakultas Matematika dan Ilmu Pengetahuan Alam,
Universitas Udayana
Jalan Raya Kampus UNUD, Bukit Jimbaran, Kuta Selatan, Badung, Bali, Indonesia
sucita.2308561022@stundent.unud.ac.id
made.agung@unud.ac.id

Abstract

The growth of digital news in Indonesia has seen a significant surge, especially in the economic sector. However, the abundance of information makes it difficult for users to find relevant news. This study aims to classify economic news using the Support Vector Machine (SVM) method and Term Frequency-Inverse Document Frequency (TF-IDF). The process begins with collecting economic news data, preprocessing the text, feature extraction using TF-IDF, and classification using SVM. Evaluation results show that the model achieved a training accuracy of 81.33%, a test accuracy of 76.57% , and an average 5-fold cross-validation accuracy of 77.3%. This indicates high accuracy and stable generalization in classifying news into appropriate categories. This study is expected to assist readers and information systems in efficiently filtering news content.

Keywords: *SVM, TF-IDF, classification teks, economy news, machine learning*

1. Pendahuluan

Pertumbuhan teknologi digital yang pesat telah membawa perubahan signifikan dalam cara masyarakat mengakses dan mengonsumsi informasi, khususnya melalui media daring (online). Setiap harinya, ribuan berita baru dipublikasikan dan disebarkan melalui berbagai portal, media sosial, serta platform digital lainnya. Banyaknya informasi ini memberikan peluang besar dalam memperluas wawasan masyarakat, namun di sisi lain juga menimbulkan masalah overload informasi. Pembaca seringkali kesulitan untuk memilah mana informasi yang relevan, akurat, dan sesuai dengan minat atau kebutuhannya [1]. Kehadiran sistem klasifikasi berita otomatis menjadi salah satu solusi penting untuk membantu pengguna dalam menyaring dan mendapatkan konten yang tepat secara efisien. Sistem ini mampu mengelompokkan berita ke dalam kategori tertentu, sehingga mempermudah pengguna dalam menemukan topik yang diinginkan serta meningkatkan pengalaman membaca yang lebih terarah. Salah satu pendekatan yang banyak digunakan dalam sistem klasifikasi teks adalah algoritma Support Vector Machine (SVM). SVM dikenal sangat efektif dalam menangani data berdimensi tinggi seperti teks, serta mampu menghasilkan hyperplane optimal yang memisahkan kelas dengan margin maksimal [2], [3]. Berbagai penelitian sebelumnya telah berhasil mengimplementasikan SVM pada klasifikasi berita. Nanda et al. [1] menggunakan SVM dengan kernel linear dan berhasil mencapai akurasi hingga 88% dalam klasifikasi berita umum.

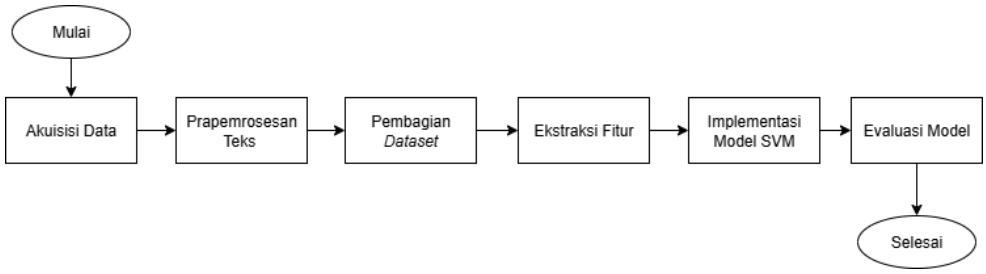
Penelitian oleh Gunawan dan Santoso [2] memanfaatkan kombinasi Doc2Vec dan SVM untuk multilabel classification pada berita bahasa Indonesia, dengan akurasi diatas 90%. Sedangkan Sudianto et al. [3] membandingkan performa SVM dengan Multi-Layer Perceptron (MLP) dalam klasifikasi topik berita, dan SVM menunjukkan performa yang kompetitif serta lebih stabil pada data yang beragam. Meskipun telah banyak penelitian dilakukan, penerapan klasifikasi otomatis yang fokus pada sektor ekonomi di Indonesia masih terbatas. Padahal, sektor ekonomi merupakan salah satu topik berita yang sangat penting dan sering menjadi perhatian masyarakat, baik untuk kepentingan bisnis, investasi, maupun kebijakan publik. Informasi ekonomi yang cepat, akurat, dan terklasifikasi dengan baik dapat membantu para pelaku ekonomi dalam

p-ISSN: 2986-3929
e-ISSN: 3032-1948

pengambilan keputusan yang lebih tepat. Dalam penelitian ini, dikembangkan model klasifikasi berita sektor ekonomi Indonesia menggunakan algoritma SVM dengan metode pembobotan Term Frequency-Inverse Document Frequency (TF-IDF). Model dirancang menggunakan pendekatan preprocessing teks yang menyeluruh, strategi One-vs-Rest untuk menangani multi-kelas, serta evaluasi model yang komprehensif melalui metrik akurasi, precision, recall, F1-score, confusion matrix, hingga cross-validation.

2. Metode Penelitian

Penelitian ini terdiri dari beberapa tahap utama, mulai dari akuisisi data, preprocessing teks, ekstraksi fitur, pembagian dataset, pembangunan model SVM, hingga evaluasi performa. Tahapan dilakukan secara sistematis untuk menghasilkan model klasifikasi berita yang akurat dan stabil.



Gambar 1. Alur Metode Penelitian

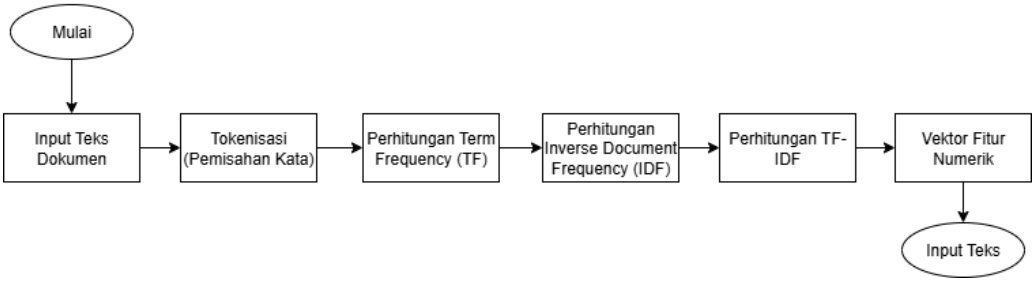
2.1. Dasar Teori

a. Term Frequency-Inverse Document Frequency (TF-IDF)

TF-IDF adalah metode pembobotan kata yang digunakan untuk menentukan pentingnya sebuah kata dalam sebuah dokumen. Ada dua komponen dalam perhitungan nilai TF – IDF, yaitu TF (Term Frequency) dan IDF (Inverse Document Frequency). TF menentukan pentingnya suatu kata relatif terhadap kemunculannya dalam suatu dokumen, sedangkan IDF menentukan suatu kata penting dalam suatu dokumen jika tidak sering muncul di dokumen lain [8]. Bobot dihitung dengan formula:

$$TF - IDF\,(t,d) = tft,d \times log(\frac{N}{dft}) \quad (1)$$

(di mana tft,d adalah frekuensi kata t di dokumen d, N total dokumen, dan dft jumlah dokumen mengandung t).

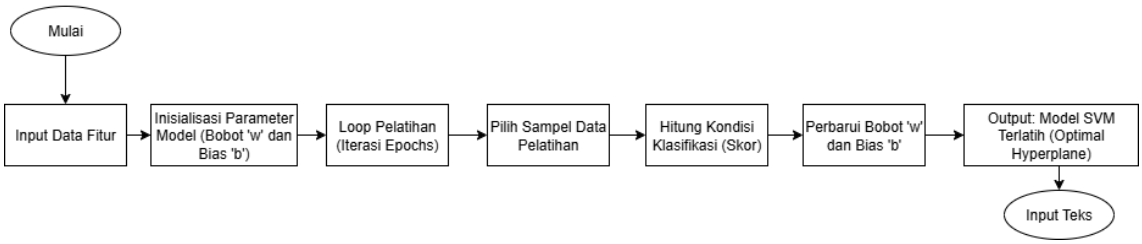


Gambar 2. AlurTerm Frequency-Inverse Document Frequency (TF-IDF)

p-ISSN: 2986-3929
e-ISSN: 3032-1948

b. Support Vector Machine (SVM)

SVM adalah salah satu metode klasifikasi dalam pembelajaran mesin (supervised learning) yang menggunakan model hasil pelatihan untuk memprediksi kelas. Algoritma ini memiliki prinsip mencari *hyperplane* yang memiliki *margin* terbesar [7]



Gambar 3. Alur Support Vector Machine (SVM)

2.2. Persiapan dan Akuisisi Data

Dataset diperoleh dari situs berita bisnis.com tahun 2016, dengan total 18.490 berita dalam bahasa Indonesia. Dataset ini terdiri dari beberapa atribut: kategori, waktu, judul, gambar, link, dan isi berita. Pada tahap awal, kolom waktu, gambar, dan link dihapus karena tidak relevan. Dataset kemudian diperiksa untuk mendeteksi dan menghapus data kosong.

```
16
17  # Hapus data kosong
18  print("\nData kosong sebelum dibersihkan:\n", df.isnull().sum())
19  df.dropna(inplace=True)
20  print("\nDimensi setelah menghapus NaN:", df.shape)
21
22  # Cek distribusi kategori
23  print(df['Kategori'].value_counts())
```

Gambar 4. Proses Akuisisi Data

Setelah *dataset* berhasil diakuisisi dan dibersihkan dari data yang tidak relevan maupun kosong, tahapan selanjutnya yang krusial, yaitu

2.3. Preprocessing Teks

Proses ini bertujuan untuk menstandarisasi dan membersihkan teks dari berbagai "noise" atau elemen yang tidak diperlukan, sehingga representasi fitur yang dihasilkan menjadi lebih akurat.

- Lowercasing: Semua huruf dalam teks dikonversi menjadi huruf kecil untuk memastikan konsistensi.
- Cleaning Teks: Proses ini menghilangkan berbagai elemen yang dapat mengganggu analisis, seperti tag HTML, tanda baca, angka, spasi ganda, serta referensi angka dalam kurung siku.
- Stopword Removal: Setelah teks dibersihkan, kata-kata umum yang memiliki frekuensi tinggi namun minim makna informatif (seperti 'yang', 'dan', 'adalah' dalam ahasa Indonesia) dihilangkan.
- Lemmatization: Tahap ini bertanggung jawab untuk mengubah kata-kata berinfleksi menjadi bentuk dasarnya (lemma). Sebagai contoh, kata 'mengurangi', 'pengurangan', dan 'dikurangi' akan direduksi menjadi 'kurang'. Proses ini memanfaatkan WordNetLemmatizer dari NLTK dengan bantuan penentuan part-of-speech untuk akurasi yang lebih baik.

p-ISSN: 2986-3929
e-ISSN: 3032-1948

```python
# Preprocess
def preprocess(text):
    text = text.lower()
    text = text.strip()
    text = re.compile(r'<.*?>').sub('', text)  # hapus HTML tag
    text = re.compile(r'[%s]' % re.escape(string.punctuation)).sub(' ', text)  # hapus
    text = re.sub(r'\s+', ' ', text)  # hilangkan spasi berlebih
    text = re.sub(r'\[[0-9]*\]', ' ', text)  # hapus referensi [1], [2], dst
    text = re.sub(r'[^\w\s]', ' ', text)  # hapus karakter selain huruf/angka
    text = re.sub(r'\d+', ' ', text)  # hapus angka
    text = re.sub(r'\s+', ' ', text)  # hilangkan spasi ganda
    return text

# Stopword Removal
stopword_list = stopwords.words('indonesian')
stop_words = set(stopwords.words('indonesian'))
```

Gambar 5. Proses Preprocessing Teks pada Kolom Berita

Setelah seluruh tahapan pra pemrosesan teks selesai dilakukan, akan menghasilkan data yang bersih dan siap untuk diolah lebih lanjut, langkah berikutnya adalah Pembagian Dataset.

2.4. Pembagian Dataset

Dataset yang sudah dibersihkan dibagi menjadi data latih (80%) dan data uji (20%) menggunakan train_test_split. Untuk mengatasi ketidakseimbangan data, dilakukan proses undersampling sehingga tiap kategori memiliki jumlah data sama, yaitu 1.384 data.

- Undersampling dan Penggabungan Kategori Kecil: Ini dilakukan untuk menyeimbangkan distribusi data.
- Label Encoding dan train_test_split: Ini adalah bagian di mana data dibagi dan label dikodekan.

```python

# --- Proses Undersampling dan Penggabungan Kategori Kecil ---
MIN_COUNT = 1000
UNDERSAMPLE_COUNT = 1384

# Mengidentifikasi kategori dengan jumlah data di atas MIN_COUNT
kategori_counts = df['Kategori'].value_counts()
kategori_ribuan = kategori_counts[kategori_counts >= MIN_COUNT].index.tolist()

# Menggabungkan kategori kecil ke 'Lainnya'
df['Kategori'] = df['Kategori'].apply(lambda x: x if x in kategori_ribuan else 'Lainnya')

# Memilih kategori final untuk undersampling
kategori_final = df['Kategori'].value_counts()[df['Kategori'].value_counts() >= UNDERSAMPLE_COUNT].index.tolist()
df_final = df[df['Kategori'].isin(kategori_final)]

# Melakukan Undersampling
df_undersampled = pd.DataFrame()
for kategori in kategori_final:
    df_kategori = df_final[df_final['Kategori'] == kategori]
    df_sampled = resample(df_kategori, replace=False, n_samples=UNDERSAMPLE_COUNT, random_state=42)
    df_undersampled = pd.concat([df_undersampled, df_sampled])
df_undersampled.reset_index(drop=True, inplace=True)

# Menampilkan distribusi data setelah penggabungan dan undersampling
print("Distribusi data setelah penggabungan dan undersampling:\n", df_undersampled['Kategori'].value_counts())

# --- Proses Label Encoding dan Train-Test Split ---
le = LabelEncoder()
df_undersampled['label_encoded'] = le.fit_transform(df_undersampled['Kategori'])

X = df_undersampled['Isi']
y = df_undersampled['label_encoded']

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

Gambar 6. Penyeimbangan Dataset dan Pembagian Data

Setelah proses penyeimbangan dan pembagian dataset, setiap kategori berita memiliki jumlah data yang seragam, yaitu 1.384 data, yang bertujuan untuk mengurangi potensi bias model terhadap kategori dengan jumlah data yang lebih besar., langkah esensial selanjutnya adalah Ekstraksi Fitur.

p-ISSN: 2986-3929
e-ISSN: 3032-1948

2.5. Ekstraksi Fitur

Pada tahap ini, teks yang telah melalui pra pemrosesan diubah menjadi representasi numerik yang dapat diproses oleh algoritma *machine learning*.

- Metode TF-IDF: Proses ekstraksi fitur dilakukan menggunakan Term Frequency-Inverse Document Frequency (TF-IDF). TfidfVectorizer dari sklearn.feature_extraction.text digunakan untuk menghitung bobot kata, merefleksikan seberapa penting sebuah kata dalam dokumen relatif terhadap seluruh korpus.

- Pembatasan Fitur: TfidfVectorizer dilatih pada data pelatihan (X_train). Meskipun awalnya dapat menghasilkan puluhan ribu fitur (kata unik), jumlah fitur kemudian direduksi menjadi maksimal 1500 fitur (max_features=1500) yang paling relevan. Reduksi ini bertujuan untuk mengurangi dimensi data, mempercepat komputasi, dan menghindari overfitting tanpa mengurangi informasi penting secara signifikan.

- Transformasi Data: Setelah vectorizer dilatih, ia digunakan untuk mengubah data pelatihan (X_train) dan data pengujian (X_test) menjadi matriks TF-IDF. Matriks ini awalnya dalam format sparse dan kemudian dikonversi menjadi dense NumPy array (.toarray()) untuk kompatibilitas dengan implementasi SVM manual selanjutnya.

```
# Inisialisasi dan latih TfidfVectorizer tanpa pembatasan fitur (untuk melihat total fitur)
tfidf_full = TfidfVectorizer()
tfidf_full.fit(X_train)
all_features = tfidf_full.get_feature_names_out()
print(f"Jumlah total fitur (sebelum reduksi): {len(all_features)}")

# Inisialisasi dan latih TfidfVectorizer dengan pembatasan max_features=1500
tfidf_limited = TfidfVectorizer(max_features=1500)
tfidf_limited.fit(X_train)
reduced_features = tfidf_limited.get_feature_names_out()
print(f"Jumlah fitur setelah reduksi (max_features=1500): {len(reduced_features)}")

# Transformasi data menjadi matriks TF-IDF (sparse) dan konversi ke dense array
X_train_limited = tfidf_limited.transform(X_train)
X_test_limited = tfidf_limited.transform(X_test)
X_train_limited_arr = X_train_limited.toarray()
X_test_limited_arr = X_test_limited.toarray()
```

Gambar 7. Ekstraksi Fitur Teks Menggunakan TF-IDF dengan max features=1500

Setelah representasi numerik data diperoleh melalui ekstraksi fitur, tahapan selanjutnya adalah Implementasi Model SVM.

2.6. Implementasi Model SVM

Pada tahapan Implementasi Model SVM, model Support Vector Machine (SVM) dibangun dan dilatih secara manual untuk klasifikasi berita multi-kelas. Proses ini melibatkan beberapa sub-tahapan penting:

a. Normalisasi Fitur Manual:

Fitur-fitur data, yang merupakan representasi numerik dari teks (matriks TF-IDF), dinormalisasi sebelum pelatihan. Normalisasi dilakukan dengan mengurangi rata-rata dari setiap fitur dan membaginya dengan standar deviasi, menggunakan fungsi normalize yang diimplementasikan secara kustom. Tujuan normalisasi adalah untuk memastikan semua fitur memiliki skala yang seragam, sehingga mencegah fitur dengan rentang nilai yang lebih besar mendominasi perhitungan jarak dan optimasi model.

p-ISSN: 2986-3929
e-ISSN: 3032-1948

```python
# Normalisasi
def normalize(X):
    mean = np.mean(X, axis=0)
    std = np.std(X, axis=0)
    std[std == 0] = 1  # Hindari pembagian nol
    return (X - mean) / std

# Hinge Loss (untuk referensi)
def hinge_loss(w, b, X, y, C):
    distances = 1 - y * (np.dot(X, w) + b)
    distances = np.maximum(0, distances)
    hinge_loss = C * np.sum(distances)
    return 0.5 * np.dot(w, w) + hinge_loss
```

Gambar 8. Normalisasi Fitur Manual

b. Implementasi Fungsi Pelatihan SVM Manual (svm_train):

Pelatihan dilakukan menggunakan pendekatan gradient descent untuk meminimalkan hinge loss. Bobot (w) dan bias (b) model diperbarui secara iteratif selama sejumlah epoch yang ditentukan. Pembaruan ini bergantung pada learning rate (lr), parameter regularisasi (C), dan kondisi klasifikasi setiap sampel pelatihan.

```python
# SVM Training
def svm_train(X, y, lr=0.0001, C=1, epochs=1000):
    n_samples, n_features = X.shape
    w = np.zeros(n_features)
    b = 0

    for epoch in range(epochs):
        for i in range(n_samples):
            condition = y[i] * (np.dot(X[i], w) + b)
            if condition >= 1:
                w -= lr * 2 * w
            else:
                w -= lr * (2 * w - C * y[i] * X[i])
                b += lr * C * y[i]
    return w, b
```

Gambar 9. Pelatihan SVM Manual

c. Strategi One-vs-Rest (OvR) untuk Klasifikasi Multi Kelas:

Strategi One-vs-Rest (OvR) diterapkan untuk mengatasi masalah klasifikasi dengan delapan kategori berita yang berbeda. Setiap model dilatih untuk membedakan satu kelas tertentu (dilabeli sebagai 1) dari semua kelas lainnya (dilabeli sebagai -1).

```python
# One-vs-Rest Training
def train_ovr_svm(X_train, y_train, num_classes, lr=0.0001, C=1, epochs=1000):
    models = []
    for k in range(num_classes):
        print(f"Training class {k} vs rest")
        y_binary = np.where(y_train == k, 1, -1)
        w, b = svm_train(X_train, y_binary, lr, C, epochs)
        models.append((w, b))
    return models

# One-vs-Rest Prediction
def predict_ovr_svm(X, models):
    scores = np.array([np.dot(X, w) + b for w, b in models])
    return np.argmax(scores, axis=0)
```

Gambar 10. Strategi One-vs-Rest (OvR)

p-ISSN: 2986-3929
e-ISSN: 3032-1948

d. Penyetelan Hyperparameter (Grid Search):

Proses grid search diimplementasikan melalui fungsi grid_search untuk menemukan kombinasi hyperparameter optimal (lr, C, epochs). Fungsi ini secara sistematis menguji berbagai kombinasi nilai hyperparameter yang telah ditentukan sebelumnya. Untuk setiap kombinasi, model dilatih dan akurasi pada data pelatihan dievaluasi. Kombinasi hyperparameter yang menghasilkan akurasi pelatihan tertinggi disimpan sebagai parameter terbaik model. Dalam penelitian ini, hyperparameter terbaik yang ditemukan adalah learning rate 0.0001, parameter C=1, dan 1000 epoch, dengan akurasi pelatihan tertinggi mencapai 81.30%

```python
#4. Hyperparameter Tuning
def grid_search(X_train, y_train):
    lr_list = [0.001, 0.0001]
    C_list = [0.1, 1]
    epochs_list = [500, 1000]

    best_acc = 0
    best_params = {}
    best_model = None
    num_classes = len(np.unique(y_train))

    for lr in lr_list:
        for C in C_list:
            for epochs in epochs_list:
                print(f"Training with lr={lr}, C={C}, epochs={epochs}")
                models = train_ovr_svm(X_train, y_train, num_classes, lr, C, epochs)
                y_pred = predict_ovr_svm(X_train, models)
                acc = accuracy(y_train, y_pred)
                print(f"Akurasi: {acc:.4f}")

                if acc > best_acc:
                    best_acc = acc
                    best_params = {'lr': lr, 'C': C, 'epochs': epochs}
                    best_model = models
```

Gambar 11. Penyetelan Hyperparameter

Setelah hyperparameter model ditentukan melalui proses grid search dan model terbaik berhasil dilatih, tahapan selanjutnya adalah Evaluasi Model untuk mengukur kinerja dan kemampuan generalisasi model pada data yang belum pernah dilihat sebelumnya.

2.7. Evaluasi Model

Pada tahap Evaluasi Model, kinerja model klasifikasi yang telah dilatih diukur secara komprehensif menggunakan data uji. Evaluasi ini mencakup beberapa metrik dan metode untuk memberikan gambaran lengkap tentang performa model:

- Waktu Prediksi: Mengukur durasi yang dibutuhkan model untuk melakukan prediksi pada seluruh set data uji, mengindikasikan efisiensi komputasi model.
- Akurasi (Accuracy): Menghitung proporsi prediksi yang benar dari total data uji.
- Laporan Klasifikasi (Classification Report): Menyediakan metrik Precision, Recall, dan F1-score untuk setiap kategori berita dan rata-ratanya, memberikan detail performa per kelas.
- Matriks Konfusi (Confusion Matrix): Visualisasi tabular yang menunjukkan jumlah prediksi benar dan salah untuk setiap kelas, membantu mengidentifikasi misklasifikasi antar kategori.

p-ISSN: 2986-3929
e-ISSN: 3032-1948

- Validasi Silang (Cross-Validation): Dilakukan K-Fold Cross-Validation (dengan k=5) secara manual untuk mengukur stabilitas dan kemampuan generalisasi model terhadap berbagai subset data, dengan melaporkan rata-rata akurasi dan deviasi standar.

3. Hasil dan Diskusi

3.1. Hasil Preprocessing dan Penyeimbangan Data

Tahap preprocessing berhasil membersihkan teks dari tanda baca, angka, stopwords, dan mengubah kata ke bentuk dasar. Dari total 18.490 data awal, setelah penggabungan kategori minoritas (<1.000 data) ke dalam "Lainnya" dan dilakukan undersampling, diperoleh 1.384 data per kategori.

```
Distribusi data setelah penggabungan dan undersampling:
 Kategori
Transportasi & Logistik    1384
Lainnya                    1384
Ekonomi                    1384
Agribisnis                 1384
Jasa & Niaga               1384
Energi & Tambang           1384
Infrastruktur              1384
Manufaktur                 1384
Name: count, dtype: int64
```

Gambar 12. Distribusi Data Setelah Preprocessing dan Undersampling

3.2. Hasil Ekstraksi Fitur

Metode TF-IDF menghasilkan representasi numerik dari teks yang merefleksikan kepentingan relatif kata dalam korpus. Sebanyak 1.500 fitur dengan bobot tertinggi dipertahankan.

```
Jumlah total fitur (sebelum reduksi): 48615
Jumlah fitur setelah reduksi (max_features=1500): 1500
```

Gambar 10. Hasil ekstraksi fitur

3.3. Hasil Pelatihan Model

Model SVM dengan konfigurasi terbaik (learning rate = 0.0001, C = 1, epochs = 500) menghasilkan akurasi pelatihan sebesar 81.33% dan akurasi data uji sebesar 76.57%.

p-ISSN: 2986-3929
e-ISSN: 3032-1948

```
Training with lr=0.0001, C=1, epochs=500
Training class 0 vs rest
Training class 1 vs rest
Training class 2 vs rest
Training class 3 vs rest
Training class 4 vs rest
Training class 5 vs rest
Training class 6 vs rest
Training class 7 vs rest
Akurasi: 0.8128
Training with lr=0.0001, C=1, epochs=1000
Training class 0 vs rest
Training class 1 vs rest
Training class 2 vs rest
Training class 3 vs rest
Training class 4 vs rest
```

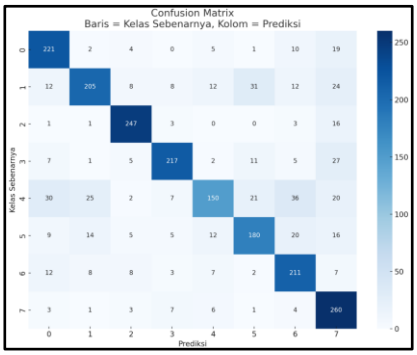
```
Training with lr=0.0001, C=1, epochs=500
Training class 0 vs rest
Training class 1 vs rest
Training class 2 vs rest
Training class 3 vs rest
Training class 4 vs rest
Training class 5 vs rest
Training class 6 vs rest
Training class 7 vs rest
Akurasi: 0.8128
Training with lr=0.0001, C=1, epochs=1000
Training class 0 vs rest
Training class 1 vs rest
Training class 2 vs rest
Training class 3 vs rest
Training class 4 vs rest
Training class 5 vs rest
Training class 6 vs rest
Training class 7 vs rest
Akurasi: 0.8130

Best Accuracy: 0.8130292424071356
Best Hyperparameters: {'lr': 0.0001, 'C': 1, 'epochs': 1000}
```

Gambar 13. Hasil Pelatihan Model

3.4. Hasil Confusion Matrix

Dari confusion matrix, terlihat bahwa model dapat mengenali kategori "Energi & Tambang" dan "Transportasi & Logistik" dengan baik (nilai diagonal tinggi). Sebaliknya, kategori "Jasa & Niaga" cenderung memiliki prediksi yang tertukar dengan "Ekonomi", menunjukkan adanya kesamaan konteks kata dalam berita.

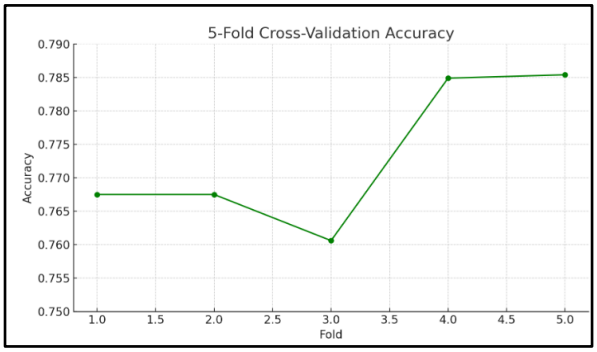



```
Confusion Matrix:
Baris = Kelas Sebenarnya, Kolom = Prediksi
     0    1    2    3    4    5    6    7
0  221    2    4    0    5    1   10   19
1   12  205    8    8   12   31   12   24
2    1    1  247    3    0    0    3   16
3    7    1    5  217    2   11    5   27
4   30   25    2    7  150   21   36   20
5    9   14    5    5   12  180   20   16
6   12    8    8    3    7    2  211    7
7    3    1    3    7    6    1    4  260
```

Gambar 14. Hasil Confusion Matrix

3.5. Hasil Cross Validation

Cross validation 5-fold menghasilkan rata-rata akurasi 77.3%, menunjukkan model memiliki stabilitas yang baik saat diuji pada subset data berbeda.




```
Final Result:
Scores per Fold : ['0.7578', '0.7679', '0.7504', '0.7854', '0.7738']
Mean Accuracy   : 0.767
Std. Deviation  : ±0.012
```

Gambar 15. Hasil Cross Validation

3.6. Diskusi

Hasil menunjukkan bahwa kombinasi TF-IDF dan SVM efektif untuk klasifikasi berita ekonomi berbahasa Indonesia. Model tetap stabil di berbagai skenario uji, meskipun ada kesalahan pada kategori yang mirip seperti "Ekonomi" dan "Jasa & Niaga". Undersampling membantu menyeimbangkan data, sementara pembatasan fitur TF-IDF (1.500 kata) mengurangi overfitting. Implementasi SVM manual membuka peluang eksplorasi parameter, dan model akhir berhasil dideploy dengan antarmuka Streamlit untuk prediksi teks secara langsung.

4. Kesimpulan

Berdasarkan hasil penelitian, dapat disimpulkan bahwa model klasifikasi berita sektor ekonomi Indonesia menggunakan algoritma Support Vector Machine (SVM) dengan metode pembobotan TF-IDF mampu menghasilkan performa yang baik. Model berhasil mencapai akurasi pengujian sebesar 76,57% dan akurasi cross validation rata-rata sebesar 77,3%, menunjukkan kemampuan generalisasi yang cukup stabil. Penggunaan strategi One-vs-Rest memungkinkan model menangani multi-kelas dengan efektif, meskipun masih ditemukan beberapa kesalahan klasifikasi pada kategori yang memiliki kemiripan konteks, seperti "Ekonomi" dan "Jasa & Niaga". Hal ini menunjukkan perlunya eksplorasi lebih lanjut pada teknik feature selection dan penyeimbangan data. Hasil penelitian ini menunjukkan potensi besar penggunaan SVM dan TF-IDF dalam mengelompokkan berita ekonomi secara otomatis, sehingga membantu pembaca dan platform media dalam menyajikan informasi yang lebih relevan. Disarankan untuk memperluas dataset dengan variasi waktu dan sumber, serta membandingkan performa model dengan algoritma lain seperti Random Forest atau deep learning. Upaya ini diharapkan dapat meningkatkan akurasi dan memperluas penerapan sistem klasifikasi otomatis di berbagai sektor berita lainnya.

Daftar Pustaka

[1] R. Nanda, E. Haerani, S. K. Gusti, and S. Ramadhani, “Klasifikasi Berita Menggunakan Metode Support Vector Machine,” *Jurnal Nasional Komputasi dan Teknologi Informasi*, vol. 5, no. 2, pp. 269–278, 2022..

[2] K. I. Gunawan and J. Santoso, “Multilabel Text Classification Menggunakan SVM dan Doc2Vec Classification pada Dokumen Berita Bahasa Indonesia,” *Journal of Information System, Graphics, Hospitality and Technology*, vol. 3, no. 01, pp. 29–38, 2021.

[3] S. Sudianto, A. D. Sripamuji, I. R. Ramadhanti, R. R. Amalia, J. Saputra, and B. Prihatnowo, “Penerapan Algoritma Support Vector Machine Dan Multi-Layer Perceptron Pada Klasifikasi Topik Berita,” Jurnal Nasional Pendidikan Teknik Informatika: JANAPATI, vol. 11, no. 2, pp. 84–91, 2022.

[4] I. Apriani, Y. Sibaroni, and I. Palupi, “Perbandingan Pembobotan Fitur TF-IDF dan TF-ABS Dalam Klasifikasi Berita Online Menggunakan Support Vector Machine (SVM),” e-Proceeding of Engineering, vol. 10, no. 3, pp. 3652–3663, 2023.

[5] F. Taufiqurrahman, S. A. Faraby, and M. D. Purbolaksono, “Klasifikasi Teks Multi Label pada Hadis Terjemahan Bahasa Indonesia Menggunakan Chi-Square dan SVM,” e-Proceeding of Engineering, vol. 8, no. 5, pp. 10650–10659, 2021.

[6] E. L. A. Suprayogi and Y. D. L. Supriana, “Pendeteksian Berita Hoax Menggunakan Metode Support Vector Machine dan Pembobotan Term Frequency Inverse Document Frequency,” e-Proceeding of Engineering, vol. 9, no. 1, pp. 58–65, 2022.

[7] K. Putria and M. A. Raharja, “Implementasi Algoritma Support Vector Machine dalam Klasifikasi Deteksi Depresi dari Postingan pada Media Sosial,” *Jurnal Nasional Teknologi Informasi dan Aplikasinya*, vol. 2, no. 1, pp. 193–202, 2023.

[8] I. K. Dwiprayoga and M. A. Raharja, “Komparasi Ekstraksi Fitur BoW dan TF-IDF untuk Klasifikasi SMS Menggunakan Naive Bayes,” *Jurnal Nasional Teknologi Informasi dan Aplikasinya*, vol. 3, no. 2, pp. 247–254, 2025.