Perbandingan Algoritma *Supervised Learning* untuk Klasifikasi Judul Skripsi Berdasarkan Bidang Dosen

http://dx.doi.org/10.28932/jutisi.v8i2.4960

Riwayat Artikel
Received: 18 Juni 2022 | *Final Revision*: 27 Juli 2022 | *Accepted*: 28 Juli 2022

Windha Mega Pradnya Duhita✉ #1, Muhammad Farhan Khoirulamri Darmawan#2, Lasmita Triana#3, Nurrofiqi Ankisqiantari#4

S1-Informatika, Universitas AMIKOM Yogyakarta
Jl. Ring Road Utara, Ngringin, Condongcatur, Kec. Depok, Kabupaten Sleman, Daerah Istimewa Yogyakarta 55281, Indonesia

[1]windha@amikom.ac.id
[2]muhammad.1608@students.amikom.ac.id
[3]lasmita.tr@students.amikom.ac.id
[4]nurrofiqi.2310@students.amikom.ac.id
✉Corresponding author: windha@amikom.ac.id

Abstrak **— Pada jenjang pendidikan khususnya S1, syarat kelulusan adalah menyelesaikan skripsi. Dalam melaksanakan skripsi, mahasiswa dibantu oleh dosen pembimbing yang akan mengarahkan dan juga menjadi tempat konsultasi. Masalahnya adalah beberapa mahasiswa tidak tahu topik apa yang harus mereka ambil untuk skripsi mereka. Dalam penelitian ini akan dibuat sistem klasifikasi judul skripsi berdasarkan bidang dosen. Dalam penelitian ini, data diambil dari Universitas AMIKOM Yogyakarta. Dengan topik yang sesuai dengan bidang dosen pembimbing, akan lebih mudah untuk mengarahkan mahasiswa dalam menyusun skripsi. Penelitian ini melakukan tahapan pelabelan, *preprocessing* teks, dan pembobotan kata atau disebut TF-IDF (*Term Frequency – Inverse Document Frequency*). Setelah itu, pemisahan data akan diklasifikasikan dengan algoritma *naive bayes classifier* (NBC), *K-Nearest Neighbor* (K-NN), dan *Support Vector Machine* (SVM). Kinerja dari ketiga algoritma tersebut dibandingkan untuk mengetahui kinerja dari algoritma tersebut baik. Hasil penelitian menunjukkan algoritma *Support Vector Machine* (SVM) berkinerja lebih baik dengan menghasilkan akurasi sebesar 89,24%, sedangkan algoritma *Naive Bayes Classifier* (NBC) menghasilkan akurasi sebesar 88,29%, dan algoritma *K-Nearest Neighbor* (KNN) dengan nilai k dari 18 menghasilkan akurasi 85,14%.**

Kata kunci— **Judul skripsi; Klasifikasi; K-NN; *Naïve Bayes*; SVM.**

Comparison of Supervised Learning Algorithm for Classification of Thesis Titles Based on Lecturer Fields

Abstract — ***At the level of education, especially S1, the graduation requirement is to complete a thesis. In carrying out the thesis, students are assisted by a college counselor who will direct and also a place for consultation. The problem is that some students don't know what topic they should take for their thesis. The research will create a thesis title classification system based on the field of***

lecturers. In this research, the data were taken from the University of AMIKOM Yogyakarta. With a topic that is in accordance with the college counselor's field, it will be easier to lead students in compiling a thesis. This study conducted stages of labeling, text preprocessing, and word weighting or called TF-IDF (Term Frequency – Inverse Document Frequency). After that, the data split will be classified with naive bayes classifier (NBC), K-Nearest Neighbor (K-NN), and Support Vector Machine (SVM) algorithms. The performance of the three algorithms is compared find out if the performance of the algorithm is good. The results showed the Support Vector Machine (SVM) algorithm performed better by producing an accuracy of 89.24%, while the Naive Bayes Classifier (NBC) algorithm produced an accuracy of 88.29%, and the K-Nearest Neighbor (KNN) algorithm with a k value of 18 produced an accuracy of 85.14%.

*Keywords—**Classification; K- NN; Naive Bayes; SVM; Thesis Title.***

I. PENDAHULUAN

Skripsi adalah salah satu syarat kelulusan bagi mahasiswa tingkat S1 sebagai tugas akhir di jenjang perkuliahan. Dalam proses pengerjaan skripsi, mahasiswa diarahkan, dibimbing, serta berkonsultasi dengan dosen pembimbing. Dosen pembimbing yang sesuai dengan bidang keahliannya akan memudahkan mahasiswa dalam pengerjaan skripsi. Jika judul skripsi mahasiswa tidak sesuai dengan bidang keahlian dosen pembimbing, maka mahasiswa akan kesulitan dalam melakukan bimbingan.

Untuk mempermudah dalam menentukan bidang dosen pembimbing yang sesuai dengan tema atau judul skripsi mahasiswa, maka perlu adanya program yang merekomendasikan bidang dosen berdasarkan judul skripsi. Program ini menghasilkan rekomendasi bidang dosen yang sesuai dan juga list nama dosen pembimbing yang sesuai dengan bidang tersebut. Pembuatan program ini menggunakan *text mining* yang akan membantu dalam mengelola data teks, dan mengklasifikasi data dengan membandingkan tiga algoritma yaitu *Naive Bayes Classifier* (NBC), *K-Nearest Neighbor* (K-NN), dan *Support Vector Machine* (SVM).

Sri Rahayu, Fatayat, Roni Salambue (2020), Hamdani Asril, Mustakim (2019), dan Fatayat, Riki Ario Nugroho (2021) melakukan penelitian mengenai rekomendasi dosen pembimbing dengan menggunakan proses *text preprocessing* yang terdiri dari *case folding, stop character removal, stopword, stemming,* dan pembobotan. Untuk penelitian Hamdani ini membandingkan algoritma *naïve bayes* dan K-NN. Hasil perbandingan tersebut algoritma K-NN lebih baik tingkat akurasinya daripada *naïve bayes*. Data yang digunakan dalam penelitian Fatayat dkk berjumlah 361 judul skripsi dengan 28 dosen pembimbing menggunakan algoritma *naïve bayes* [1] [2] [3].

Akromunnisa & Hidayat (2019), melakukan klasifikasi data abstrak skripsi Teknik informatika menggunakan K-Nearest Neighbor. Penelitian ini menggunakan abstrak Indonesia, inggris, dan data judul skripsi. Hasil yang didapat berupa klasifikasi data abstrak Indonesia lebih besar yaitu 100% tanpa proses *stemming* [4].

Pratama, Wihandika, & Ratnawati (2018), dalam penelitiannya melakukan sistem prediksi kelulusan mahasiswa. Pada penelitian tersebut terlihat bahwa keinginan mahasiswa lulus tepat waktu namun pada kenyataanya banyak yang tidak seperti itu, sehingga diperlukan penerapan sistem cerdas dalam prediksi kelulusan mahasiswa. Sistem ini membantu *user* dalam mengambil keputusan karena tingkat akurasinya mencapai 80.55% [5].

Hadiyanoor (2018), dan Chalida & Wahyudi (2019) membuat penerapan sistem cerdas untuk melakukan klasifikasi. Penelitian yang dilakukan oleh Hadiyanoor menggunakan algoritma SVM dan pengurutan ranking menggunakan metode *Cosine Similarity* dengan hasil akurasi 85% sedangkan penelitian yang dilakukan oleh Chalida & Wahyudi menggunakan algoritma NBC dan pembobotan menggunakan TF-IDF dengan data uji sejumlah 5055 dengan hasil paling besar *sentimen negatif* sebesar 35% [6] [7].

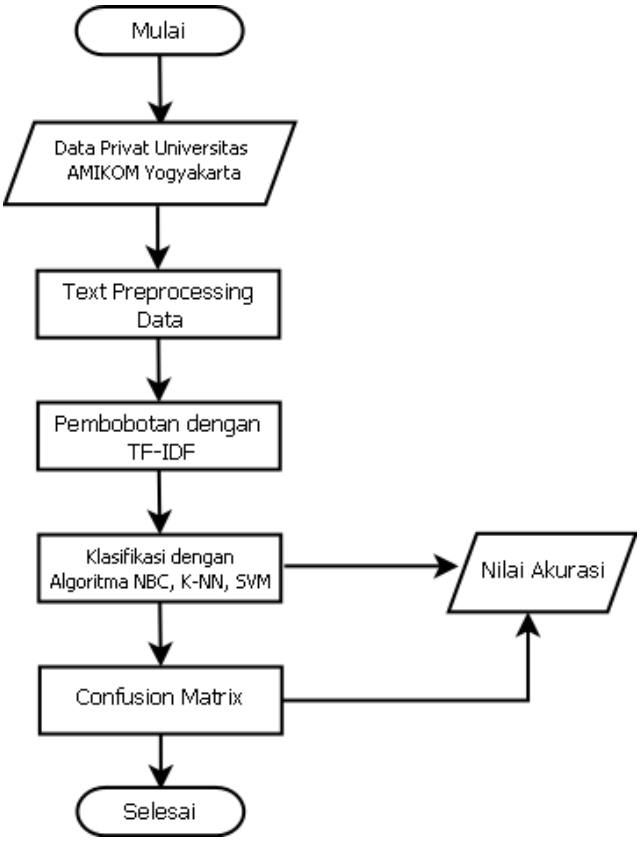
Ernawati & Wati (2018), dalam penelitiannya menganalisis *sentiment review* Agen Travel menggunakan algoritma K-Nearest Neighbor yang menggunakan 100 *review positif* dan 100 *review negatif* dengan enam kata yang berhubungan dengan *sentimen* yaitu: *Fast, Good, Great*, Buruk, *Cancel,* dan Tunggu. Dari penelitian yang telah dilakukan, didapatkan hasil akurasi sebesar 87% dan titik AUC 0,916 yang masuk dalam kategori *excellent* [8].

Patmi Kasih & Intan Nur Farida (2017) dalam penelitiannya membuat sebuah sistem dosen dengan metode perancangan dengan konsep *waterfall*. Hasil dari penelitian ini adalah untuk membangun sistem aplikasi rekomendasi pemilihan pembimbing tugas akhir berbentuk desktop dan mengimplementasikan metode *naive bayes* ke dalam aplikasi rekomendasi pemilihan pembimbing tugas akhir [9].

Berdasarkan penelitian-penelitian diatas menggunakan satu algoritma atau membandingkan dua algoritma yang berbeda. Pada penelitian ini menggunakan tiga algoritma yang akan dibandingkan yaitu NBC, K-NN, dan SVM. Ketiga algoritma tersebut merupakan *supervised learning* karena bisa mengolah dan mengelompokan data yang berlabel. Dari hasil pengukuran kinerja ketiga algoritma tersebut, algoritma yang mendapatkan nilai tertinggi yang diimplementasikan dalam sistem rekomendasi dosen pembimbing skripsi.

II. METODE PENELITIAN

Pada penelitian ini dilakukan dengan beberapa tahapan seperti pada Gambar 1 yaitu data yang diperoleh diolah dengan *text preprocessing*,. Selanjutnya melakukan pembobotan kata menggunakan TF-IDF. Hasil dari pembobotan kata kemudian dilakukan pengujian dataset menggunakan algoritma *Naive Bayes*, K-NN, dan SVM. Terakhir yaitu membandingkan tingkat nilai akurasi antara algoritma *Naive Bayes*, K-NN, dan SVM menggunakan *Confusion Matrix.*



Gambar 1. Alur Penelitian

A. Pengambilan Data

Data dalam penelitian ini merupakan data privat dari pihak instansi terkait yaitu Universitas Amikom Yogyakarta program studi Informatika. Data yang diperoleh sebesar 1598 data lalu menambahkan beberapa data baru untuk label "tidak ada bidang dosen". Data tersebut dilakukan labelisasi secara manual, pelabelan ini bertujuan untuk memudahkan dalam proses klasifikasi. Dalam penelitian ini melabeli bidang dosen yang sesuai tema skripsi. Bidang dosen program studi Informatika yaitu terdiri dari *Data Science* (0), *Multimedia* (1), *Networking* (2), *Software Engineering* (3), *Game* (4), dan Tidak Ada Bidang Dosen (5). Berikut ini gambar 2 merupakan implementasi dari pelabelan bidang dosen.

p-ISSN : 2443-2210
e-ISSN : 2443-2229

MELABAELI BIDANG DOSEN

- Data Science: 0,
- Multimedia: 1,
- Networking: 2,
- Software Engineering: 3,
- Game: 4,
- Tidak Ada Bidang Dosen : 5

```
In [7]: cleanup_nums = {"Bidang Dosen": {"Data Science": 0, "Multimedia": 1, "Networking": 2, "Software Engineering": 3, "Game": 4,
                                         }
        df = df.replace(cleanup_nums)
        df.head()
```

Out[7]:

	Judul Skripsi	Bidang Dosen
0	Menentukan Entitas dari Pemberitaan Media Dari...	0
1	Perbandingan Algoritma ACO dan PSO untuk Pemil...	0
2	Implementasi Kriptografi pada File Photo dan V...	0
3	Implementasi Jaringan Syaraf Tiruan Menggunaka...	0

Gambar 2. Implementasi Melabeli Bidang Dosen

B. *Text Processing*

1. *Case Folding*

Proses ini akan merubah data menjadi *lowercase* atau mengubah huruf kapital menjadi huruf kecil. Setelah pengumpulan dan pelabelan data, lalu penelitian ini melakukan pengecilan huruf pada data judul skripsi. Adapun implementasinya berikut pada Gambar 3.

Case Folding

lower case atau mengecilkan huruf

```
In [15]: teks_data_1=[]
         for teks in teks_data:
           teks_process = [teks.lower()]
           teks_data_1=teks_data_1+teks_process
         print(teks_data_1[0])
```

```
menentukan entitas dari pemberitaan media daring menggunakan hidden markov model untuk metode pos tagging
```

```
In [16]: teks_data_1
```

```
Out[16]: ['menentukan entitas dari pemberitaan media daring menggunakan hidden markov model untuk metode pos tagging',
          'perbandingan algoritma aco dan pso untuk pemilihan rute terpendek penjemputan panen kelapa sawit (studi kasus : pt djas
         a anugrah sejati)',
```

Gambar 3. Implementasi Proses *Case Folding*

2. *Tokenizing*

Pada tahapan ini akan dilakukan proses *tokenizing* yang menghilangkan angka, menghilangkan tanda baca (titik, koma, dan sebagainya), dan menghilangkan spasi yang tidak perlu. Berikut ini adalah gambar 4 pengimplementasian dari proses *tokenizing.*

menghilangkan angka dan link

```
In [17]: teks_data_2=[]
         for teks in teks_data_1:
           teks_process = [re.sub(r"http\S+", "", teks)]
           teks_processes = [re.sub(r'\d+', '', teks)]
           teks_data_2=teks_data_2+teks_processes
         print(teks_data_2[0])
```

menghilangkan tanda baca

```
In [19]: teks_data_3=[]
         for teks in teks_data_2:
           teks_process =[teks.translate(str.maketrans("","",string.punctuation))]
           teks_data_3=teks_data_3+teks_process
         print(teks_data_3[0])
```

menghilangkan spasi yang berlebihan

```
In [21]: teks_data_4=[]
         for teks in teks_data_3:
           teks_process = [teks.strip()]
           teks_data_4=teks_data_4+teks_process
         print(teks_data_4[0])

         menentukan entitas dari pemberitaan media daring menggunakan hidden markov model untuk metode pos tagging

In [22]: teks_data_4

Out[22]: ['menentukan entitas dari pemberitaan media daring menggunakan hidden markov model untuk metode pos tagging',
          'perbandingan algoritma aco dan pso untuk pemilihan rute terpendek penjemputan panen kelapa sawit studi kasus  pt djasa
         anugrah sejati',
```

Gambar 4. Implementasi Proses *Tokenizing*

3. *Stopword*

Proses *stopword* ini menghilangkan kata - kata yang tidak penting, seperti kata sambung, kata ganti, kata keterangan, dan kata sandang. Misalnya “dan”, “yang”, “ada”, “untuk”, dan lain - lain. Adapun gambar 5 merupakan implementasi dari proses *stopword* tersebut.

Stopword

menghilangkan kata-kata yang tidak penting seperti, kata ganti, kata keterangan, kata sambung, kata depan, dan kata sandang

```
In [23]: factory = StopWordRemoverFactory()
         stopword = factory.create_stop_word_remover()
         print(stopwords)

In [25]: from Sastrawi.StopWordRemover.StopWordRemoverFactory import StopWordRemoverFactory
         import pandas as pd

         factory = StopWordRemoverFactory()
         stopword = factory.create_stop_word_remover()
         teks_data_5=[]
         for i, kalimat in enumerate (df_new['Setelah Preprocessing']):
           stop = [stopword.remove(kalimat)]
           teks_data_5=teks_data_5+stop
         print(teks_data_5[0])

         menentukan entitas pemberitaan media daring menggunakan hidden markov model metode pos tagging

In [26]: teks_data_5

Out[26]: ['menentukan entitas pemberitaan media daring menggunakan hidden markov model metode pos tagging',
          'perbandingan algoritma aco pso pemilihan rute terpendek penjemputan panen kelapa sawit studi kasus  pt djasa anugrah se
         jati',
```

Gambar 5. Implementasi Proses *Stopword*

C. TF-IDF

TF-IDF (Term Frequency-Inverse Document Frequency) merupakan penghitungan bobot dari kata yang sering digunakan dalam text mining. Metode ini sangat efisien, mudah, dan memiliki hasil yang akurat. Pembobotan kata menggunakan TF-IDF dengan rumus :

Rumus TF = Pembobotan atau *weight* setiap kata (*term*) pada suatu dokumen berdasarkan jumlah kemunculannya dalam dokumen tersebut dengan bobot 1 dan 0 [10].

$$IDF(\omega) = \log\left(\frac{N}{DF(\omega)}\right) \quad (1)$$

$$TFIDF(\omega, d) = TF(\omega, d).IDF(\omega) \quad (2)$$

$TF\ IDF\ (\omega)$: bobot suatu kata dalam keseluruhan dokumen
ω : suatu kata
d : suatu dokumen
$TF(\omega, d)$: frekuensi kemunculan sebuah kata ω dalam dokumen
$IDF\ (\omega)$: invers *DF* dari kata ω
N : jumlah keseluruhan dokumen
$IDF\ (\omega)$: jumlah dokumen yang mengandung kata ω

Data yang sudah melalui *text preprocessing*, kemudian data teks diubah menjadi data vektor (angka) dengan proses TF-IDF. Berikut ini tabel 1 adalah sampel perhitungan TF-IDF.

TABEL 1
PERHITUNGAN TF-IDF

Term	IDF	TF-IDF = TF x IDF				
		D1	D2	D3	D4	D5
algoritma	0.698	0	0	0.698	0	0
pakar	0.221	0	0	0.221	0.221	0.221
yogyakarta	0.096	0.096	0	0.096	0.096	0.096

D. Algoritma Naïve Bayes, K-NN, SVM

1. Naive Bayes Classifier (NBC)

Algoritma NBC salah satu metode klasifikasi yang menggunakan metode probabilitas. Algoritma ini bisa digunakan untuk memprediksi masa depan menggunakan pengalaman yang lalu. Rumus dari *teorema bayes* adalah sebagai berikut [11]:

$$P\ (A \mid B) = \frac{P(B \mid A)\ .\ P(A)}{P(B)} \quad (3)$$

A , B : events
P (A|B) : probability of A given B is true
P (B|A) : probability of B given A is true
P (A) , P (B) : the independent probabilities of A and B

2. K-Nearest Neighbor (K-NN)

K-NN termasuk dalam kategori *supervised learning*, yang dapat menyelesaikan masalah *classification*. Algoritma ini melakukan klasifikasi dengan tetangga terdekat. K-NN termasuk algoritma *lazy learning* karena hanya sedikit belajar. Juga disebut *instance-based* karena K-NN ini tidak memerlukan model, tapi sebagai gantinya k-nn akan

membentuk hipotesis dari data training yang tersedia. Jika semakin banyak data yang digunakan maka hasil akhirnya akan semakin kompleks [12].
Adapun langkah-langkah dalam algoritma K-Nearest Neighbor, yaitu sebagai berikut [7]:

- Pilih *k* buah tetangga terdekat (*nearest neighbor*) secara acak,
- Petakan dataset ke ruang vektor
- Pisahkan dataset menjadi *data training* dan data uji
- Hitung jarak *d*, antara data uji dengan *data training*
- *Sorting d* dari yang terkecil hingga besar
- Ambil dan pisahkan data hasil *sorting* sebanyak *k*
- Amati *class* mayoritas
- Klasifikasikan data uji berdasarkan mayoritas

Jarak *Euclidean*

$$d(x_1, x_2) = \sqrt{\sum_{i=1}^{m}(x_{i1} - x_{i2})^2} \qquad (4)$$

Cosine Similarity, ini untuk menghitung kedekatan antara dokumen [13].

$$CosSim(x, dj) = \frac{\sum_{x=1}^{m} x_i dj_i}{\sqrt{\sum_{i=1}^{m} x_i^2} \cdot \sqrt{\sum_{i=1}^{m} dj_i^2}} \qquad (5)$$

x : dokumen uji
dj : dokumen latih
x_i dj_i: nilai bobot yang diberikan disetiap *term* pada dokumen

3. Support Vector Machine

SVM adalah salah satu metode dari *machine learning* untuk melakukan *Classification.* SVM pertama kali dikenalkan oleh Vapnik dkk pada tahun 1992. SVM merupakan *supervised learning* dan *binary classification*, untuk memaksimalkan *margin* antara dua *class* [12]. SVM memiliki tujuan yaitu menemukan optimal *hyperplane* (fungsi pemisah) yang memisahkan dua kelas, dengan memaksimalkan *margin* dua kelas. Margin sendiri merupakan jarak antar *hyperplane* dan *input vectors*(data) terdekat. Sedangkan *support vector* merupakan *Input Vector* yang bersentuhan dengan *boundary*. *Support vector* sendiri merupakan data yang paling luar dari masing-masing *class*.
Langkah-langkah dalam algoritma SVM sebagai berikut [14]:

- Menginisiasi nilai awal untuk nilai C
- Memasukan *data training* berdasarkan kemunculan *keyword* dalam satu judul
- Memasukan *dot product* dengan memasukan fungsi *kernel*.
- Menghitung Matriks
- Mencari nilai *error*
- Mencari nilai dari *delta alpha*
- Melakukan pengujian dengan menghitung *dot product* antara data uji dengan semua data latih.

Menghitung Matriks :

$$D_{ij} = y_i y_j \left(K(x_i . x_j) + \lambda^2\right) \qquad (6)$$

D_{ij} : elemen matriks ke ij
y_i : kelas data ke-i
y_j : kelas data ke-j
λ : batas teoritis
$K(x_i . x_j)$: kernel

Pada penelitian ini membandingkan empat jenis *kernel* yang berbeda yaitu *Linier*, *RBF*, *Polynomial*, dan *Radial* dengan hasil *kernel* terbaik pada penelitian ini merupakan *kernel Rbf* hal ini disebabkan karena *kernel Rbf* mampu menyelesaikan permasalahan *non-linier* dan dapat masuk kedalam dimensi baru sehingga data yang dihasilkan lebih akurat. Pada penelitian ini juga menentukan nilai C terbaik antara 1-5 dan dengan hasil nilai C terbaik yaitu 5 hal ini dipengaruhi karena nilai C pada pengujian ini memiliki *error* klasifikasi yang tidak berpengaruh pada margin. Nilai C lebih besar dari 0 itu penting karena menambah margin dan mengurangi jumlah *slack*. Pada penelitian ini melakukan pengujian nilai *gamma*, *hyperparameter gamma* ini untuk mengontrol jarak pengaruh titik latihan dan dicobakan antara 2-5 dengan hasil terbaik 2.

III. HASIL PEMBAHASAN

Hasil dan pembahasan pada penelitian kali ini, akan membandingkan hasil dari confusion matrix yang diperoleh dari ketiga Algoritma yang ada, yaitu NBC, K-NN, dan SVM.

A. Algoritma Naive Bayes Classifier

Klasifikasi algoritma ini menggunakan *library "sklearn.naive_bayes"* dengan *naive bayes* versi *multinomial*. Dalam algoritma ini membagi *data training* dan *data testing* dengan perbandingan 8:2. Berikut Gambar 6 adalah implementasi perhitungan akurasi di *Jupyter Notebook.*

MENCARI AKURASI NAIVE BAYES CLASSIFIER

```
[44]: # 1. import
      from sklearn.naive_bayes import MultinomialNB

      # 2. instantiate a Multinomial Naive Bayes model
      nb = MultinomialNB()
      %time nb.fit(Train_X_Tfidf, Train_Y)

      Wall time: 5 ms
[44]: MultinomialNB()

[45]: y_pred_class = nb.predict(Train_X_Tfidf)

[46]: # calculate accuracy of class predictions
      from sklearn import metrics

      print("Naive Bayes Score: {:.2f}%".format(nb.score(Train_X_Tfidf, Train_Y)*100))

      Naive Bayes Score: 88.29%
```

Gambar 6. Perhitungan Akurasi Algoritma NBC di *Jupyter Notebook*

Pengujian akurasi dilakukan dengan menggunakan *Confusion Matriks* dilakukan untuk mengetahui seberapa besar keberhasilan suatu sistem dalam melakukan klasifikasi. Dengan adanya *confusion matrix* ini bisa menghitung akurasi secara manual. Gambar 7 merupakan *script* dari *Confusion Matrix*.

CONFUSION MATRIX

```
[47]: from sklearn.metrics import confusion_matrix
      confusion_matrix(Train_Y, y_pred_class)

[47]: array([[292,  24,  15,   2,  19,   0],
             [ 29, 179,   5,   0,   5,   0],
             [  7,   1, 361,   1,   3,   0],
             [ 11,   2,   2,  53,  14,   0],
             [  5,   1,   1,   1, 409,   0],
             [  6,   0,   7,   0,  18,  56]], dtype=int64)
```

Gambar 7. *Confusion Matrix* Algoritma NBC

Perhitungan *confusion matrix multi-class* secara manual memakai rumus

Akurasi = $\frac{\sum TP}{Total\ Set\ Data} * 100$

$$\text{Akurasi} = \frac{292 + 179 + 361 + 53 + 409 + 56}{1529} * 100\% = 88.29\%$$

B. Algoritma K-Nearest Neighbor

Klasifikasi algoritma ini menggunakan *library "sklearn.neighbor".*Dalam algoritma ini membagi *data training* dan *data testing* dengan perbandingan 8:2, dan penelitian ini mengambil tetangga terdekat senilai k 18. Mengambil nilai k 18 karena menghasilkan nilai akurasi yang paling besar diantara nilai k yang lainnya. Akurasi yang didapat sebesar 85.14%. Berikut Gambar 8 adalah implementasi perhitungan akurasi di *Jupyter Notebook.*

Akurasi Algoritma K-NN

```
In [90]: # algoritma knn
         from sklearn.neighbors import KNeighborsClassifier
         knn = KNeighborsClassifier(n_neighbors = 18)  # n_neighbors means k
         knn.fit(Train_X_Tfidf, Train_Y)
         prediction = knn.predict(Test_X_Tfidf)

         print("{} NN Score: {:.2f}%".format(18, knn.score(Test_X_Tfidf, Test_Y)*100))

         18 NN Score: 85.14%
```

Gambar 8. Perhitungan Akurasi di *Jupyter Notebook*

Untuk menampilkan *confusion matrix* menggunakan *library "sklearn".* Dalam *confusion matrix* ini menggunakan TP *(true positive)* dan total dataset untuk menghitung akurasi manual. Gambar 9 merupakan *confusion matrix* algoritma K-NN, dan itu termasuk *confusion matrix multi-class.*

```
In [92]: from sklearn import metrics
         print(metrics.confusion_matrix(Test_Y,prediction))

         [[32  2  1  6  0  5]
          [ 0 87  1  3  2  0]
          [ 0  2 81  0  2  0]
          [ 4  9  5 56  1  2]
          [ 0  1  0  0 17  0]
          [ 0  0  1  1  0  2]]
```

Gambar 9. Confusion Matrix Algoritma K-NN

Menghitung *confusion matrix multi-class* menggunakan rumus 7.

$$\text{Akurasi} = \frac{\sum TP}{Total\ Set\ Data} * 100 \qquad (7)$$

Perhitungan manual akurasi :

$$\text{Akurasi} = \frac{32 + 87 + 81 + 56 + 17 + 2}{323} * 100\% = 85{,}1393\%$$

C. Algoritma Support Vector Machine

Pada penelitian ini menggunakan jenis klasifikasi *multiclass One Vs One Classifier* (OVO) dengan menggunakan *kernel RBF* karena teknik ini sering digunakan dalam pemecahan masalah *non linier* dan mampu memisahkan data lebih akurat sehingga dapat memberikan batas untuk menentukan nilai *c* dan *ɣ*. Berdasarkan waktu yang digunakan dalam melakukan klasifikasi, metode OVO lebih cepat dibandingkan *One Vs Rest* karena semakin besar datanya akan semakin membutuhkan banyak waktu. Berikut merupakan nilai akurasi algoritma SVM menggunakan *library* dari SVC dengan menggunakan perbandingan 8:2 untuk *data training* dan data testing yang menghasilkan nilai akurasi sebesar 89,24% . Gambar 10 adalah implementasi perhitungan akurasi di *Jupyter Notebook.*

```
In [41]:  1 SVM = SVC()
          2 SVM.fit(Train_X_Tfidf,Train_Y)
          3 predictions_SVM =SVM.predict(Test_X_Tfidf)
          4 print("SVM Accuracy Score : "),accuracy_score(predictions_SVM,Test_Y)*100

SVM Accuracy Score :

Out[41]: (None, 89.24418604651163)
```

Gambar 10. Perhitungan di Jupyter Notebook Algoritma SVM

Pengujian akurasi dengan *Confusion Matrix* dilakukan untuk mengetahui keberhasilan sistem dalam proses klasifikasi. Pada proses ini menggunakan *library* dari *Sklearn* dan *matplotplit* untuk mengetahui grafiknya supaya mudah dipahami. Gambar 11 merupakan implementasi *Confusion Matrix* Algoritma SVM.

```
In [59]:  1 from sklearn.metrics import confusion_matrix
          2 confusion_matrix(Test_Y,predictions_SVM)

Out[59]: array([[38,  1,  3,  5,  0,  0],
                [ 0, 79,  1,  2,  0,  0],
                [ 0,  1, 83,  2,  2,  0],
                [ 6,  5,  4, 62,  0,  0],
                [ 0,  0,  0,  2, 19,  0],
                [ 0,  1,  1,  1,  0, 26]], dtype=int64)
```

Gambar 11. Confusion Matrix Algoritma SVM

Perhitungan manual akurasi :

$$\text{Akurasi} = \frac{\sum TP}{Total\ Set\ Data} * 100$$

$$\text{Akurasi} = \frac{38 + 79 + 83 + 62 + 19 + 26}{343} * 100\% = 89.24\%$$

Dalam penelitian ini akan membandingkan kinerja dari ketiga algoritma tersebut. Hasil perhitungan akurasi, disajikan dalam tabel 2.

TABEL 2

PERBANDINGAN KETIGA ALGORITMA

Algoritma	**Akurasi**
NBC	88,29%
K-NN	85,14%
SVM	89,24%

Algoritma NBC ini menggunakan *Multinomial Naïve Bayes* karena fitur diambil dari distribusi sederhana sehingga lebih efisien dan lebih tepat untuk menghitung bentuk data *multiclass* yang ada. Akurasi yang diperoleh algoritma ini sebesar 88,29%.

Dalam algoritma *K-NN* nilai k itu menjadi pengaruh tingkat akurasi yang akan dihasilkan. Penelitian ini mengambil nilai k 18 dengan pembagian *data split* 80:20 (*training : testing*) sehingga menghasilkan akurasi sebesar 85,14%. Dengan banyaknya data yang dipelajari (*data training*), maka semakin akurat tingkat akurasinya.

Algoritma *SVM* ini menggunakan nilai C sebesar 5, hal ini dipengaruhi karena nilai C pada pengujian ini memiliki *error* klasifikasi yang tidak berpengaruh pada margin. Gamma sebesar 2 untuk mengontrol jarak pengaruh titik dan menggunakan *kernel RBF*. Proses klasifikasi ini terbukti mampu memberikan rekomendasi bidang dosen yang sesuai judul skripsi. Berdasarkan hasil dari pengujian yang dilakukan mampu memperoleh hasil akurasi sebesar 89.24%.

IV. SIMPULAN

Penelitian ini mengklasifikasi judul skripsi mahasiswa berdasarkan bidang dosen dengan membandingkan kinerja ketiga algoritma yaitu, *Naive Bayes Classifier* (NBC), *K-Nearest Neighbor* (K-NN), dan *Support Vector Machine* (SVM). Hasil pengujian menunjukkan algoritma NBC menghasilkan akurasi sebesar 88.29%, K-NN sebesar 85.14%, dan SVM sebesar 89.24%. Perbandingan kinerja tersebut menunjukkan bahwa algoritma SVM lebih baik kinerjanya dibandingkan dengan NBC dan K-NN. Kinerja algoritma SVM lebih baik dikarenakan SVM ini menggunakan teknik *kernel RBF* yang mana dapat digunakan dalam pemecahan masalah *non-linier* (jumlah pelabelan lebih dari dua) dan teknik tersebut memisahkan data lebih akurat. Algoritma SVM ini juga menggunakan metode *One vs One*, karena lebih cepat dalam pengklasifikasian data dalam jumlah yang banyak.

Saran untuk penelitian selanjutnya adalah bisa menambahkan dataset dengan jumlah yang lebih banyak. Dengan jumlah data yang banyak akan semakin optimal *system* dalam menganalisis data. Saat *text preprocessing* kata-kata berbahasa Inggris diterjemahkan, juga tambahkan fungsi dalam mengubah kata tidak baku menjadi kata baku sehingga data yang akan digunakan dapat lebih optimal dan bersih.

UCAPAN TERIMA KASIH

Ucapan terimakasih kepada Universitas Amikom Yogyakarta dan Prodi S1-Informatika yang telah membantu dan bersedia untuk memberikan dataset dalam penelitian ini.Terimakasih kepada dosen pembimbing, dan teman-teman yang sudah banyak membantu dalam pembuatan penelitian ini. Semoga sukses selalu.

DAFTAR PUSTAKA

[1] S. Rahayu, Fatayat, R. Salambue and A. Aminuddin, "Analisis Penentuan Dosen Pembimbing Skripsi Mahasiswa Menggunakan Naïve Bayes Classifier," [Online]. Available: https://repository.unri.ac.id/handle/123456789/10089. [Accessed 20 Juni 2022].

[2] H. Asril, Mustakim and I. Kamila, "I. K. Ham Klasifikasi Dokumen Tugas Akhir Berbasis Text Mining menggunakan Metode Naïve Bayes Classifier dan K-Nearest Neighbor," in *Seminar Nasional Teknologi Informasi, Komunikasi dan Industri (SNTIKI)*, Pekanbaru, 2019.

[3] Fatayat and R. A. Nugroho, "Analisa Penentuan Dosen Pembimbing Tugas Akhir Mahasiswa Menggunakan Naive Bayes Classifier," *Simtika*, vol. 4, no. 3, pp. 1-7, 2021.

[4] K. Akromunnisa and R. Hidayat, " Klasifikasi Dokumen Tugas Akhir (Skripsi) Menggunakan K-Nearest Neighbor," *JISKA (Jurnal Inform. Sunan Kalijaga)*, vol. 4, no. 1, pp. 69-75, 2019.

[5] A. Pratama, R. C. Wihandika and D. E. Ratnawati, "Implementasi algoritma support vector machine (SVM) untuk prediksi ketepatan waktu kelulusan mahasiswa," *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer*,, vol. 2, no. 4, p. 1704–1708, 2018.

[6] A. Hadiyanoor, "Klasifikasi Proposal Tugas Akhir Berdasarkan Bidang Keahlian Dan Peminatan Dosen Pembimbing Menggunakan Support Vector Machine Pada Jurusan Teknik Informatika Di Stmik Indonesia Banjarmasin," *Jurnal Ilmiah Teknologi Informasi dan Komputer Pranala*, vol. 13, no. 2, p. 57 – 128, 2018.

[7] M. Chalida and M. D. R. Wahyudi, "Analisis sentimen ujaran kebencian pemilihan presiden 2019 menggunakan algoritma Naïve Bayes," *Jnanaloka*, vol. 1, no. 1, pp. 25-33, 2020.

[8] S. Ernawati and R. Wati, "Penerapan Algoritma K-Nearest Neighbors Pada Analisis Sentimen Review Agen Travel," *Jurnal Khatulistiwa Informatika*, vol. 6, no. 1, pp. 64-69, 2018.

[9] P. Kasih and I. N. Farida, "Sistem Bantu Pemilihan Dosen Pembimbing Tugas Akhir Berdasarkan Kategori Pilihan dan Keahlian Dosen menggunakan Naïve Bayes," in *Seminar Nasional Teknologi Informasi, Komunikasi dan Aplikasinya (SNATIKA)*, Malang, 2017.

[10] R. Feldman and J. Sanger, The text mining handbook: Advanced approaches in analyzing unstructured data, Cambridge Univ. Press, 2007.

[11] M. H. Widianto, "Algoritma Naive Bayes," [Online]. Available: https://binus.ac.id/bandung/2019/12/algoritma-naive-bayes/. [Accessed 20 Juni 2022].

[12] R. Primartha, Algoritma Machine Learning, Bandung: Informatika, 2021.

[13] Y. Heryadi and T. Wahyono, Machine Learning: Konsep dan Implementasi., Yogyakarta: Gava Media, 2020.

[14] A. R. T. H. Ririd, A. W. Kurniawati and Y. Yunhasnawa, "Implementasi Metode Support Vector Machine Untuk Identifikasi Penyakit Daun Tanaman Kubis," *Jurnal Informatika Polinema*, vol. 4, no. 3, p. 181, 2018.