# Golebooks: Chatbot *Book E-Commerce* Menggunakan Algoritma *Decision Tree*


Andika Risky Sururi
G6401211046
*Computer Science*
*IPB University*
Bogor, Indonesia
andikarisky@apps.ipb.ac.id

Antonio
G6401211111
*Computer Science*
*IPB University*
Bogor, Indonesia
rosaantonio@apps.ipb.ac.id

Farhan Nurohman
G6401211028
*Computer Science*
*IPB University*
Bogor, Indonesia
07032003farhan@apps.ipb.ac.id

Muhammad Ghifar Azka Nurhadi
G6401211108
*Computer Science*
*IPB University*
Bogor, Indonesia
ghifarazka@apps.ipb.ac.id

Rafif Rabbani
G6401211018
*Computer Science*
*IPB University*
Bogor, Indonesia
generasirabbani@apps.ipb.ac.id



***Abstrak*—Saat ini, penerapan kecerdasan buatan telah mencakup berbagai bidang, salah satunya adalah komersial dan retail. Pada bidang tersebut, salah satu teknologi yang sering digunakan adalah chatbot. Teknologi ini dapat membantu pelanggan e-commerce yang ingin bertanya mengenai harga maupun persoalan transaksi lainnya. Chatbot bekerja dengan membaca input teks dari pengguna, dan memberikan teks responnya. Dalam pengembangan chatbot, salah satu metode yang dominan digunakan adalah decision tree. Pada penelitian ini, dilakukan pengembangan sebuah chatbot bernama Golebooks, yang ditujukan untuk membantu pelanggan e-commerce buku. Golebooks dikembangkan menggunakan metode decision tree yang diimplementasikan dalam Python menggunakan bantuan pustaka scikit-learn dan Sastrawi. Terhadap hasil model yang didapatkan, dilakukan tuning untuk meningkatkan akurasi model. Hasilnya, ditemukan bahwa model awal memiliki akurasi yang lebih baik, sedangkan model hasil tuning mengalami overfitting.**



***Kata Kunci—book e-commerce, chatbot, decision tree, sastrawi, scikit-learn***


## Pendahuluan

Dalam era digital yang semakin berkembang, chatbot menjadi salah satu inovasi teknologi yang banyak digunakan sebagai media komunikasi dengan pengguna dalam berbagai bidang, termasuk layanan publik [6]. Chat robot atau Chatbot merupakan salah satu bentuk penerapan *Natural Language Processing* (NLP), yang merupakan salah satu bidang dalam *artificial intelligence* yang mempelajari komunikasi antara manusia dan komputer menggunakan bahasa alami [4]. Chatbot tidak hanya dapat memberikan informasi namun juga dapat mengutarakan ekspresi emosi. Teknologi ini mulai ada sejak tahun 1966 yang dibuat oleh Joseph Weizenbaum dan dinamai ELIZA, saat itu ELIZA digunakan sebagai seorang psikoterapis dalam berinteraksi dengan lawan bicara manusia. Pada 2019, sekitar 80% proses pemasaran memanfaatkan penggunaan teknologi chatbot untuk mendukung berbagai layanan bisnis mereka.

Terkhusus dalam pemanfaatan dalam bidang e-commerce, chatbot menjadi aset yang bernilai karena mampu menjawab pertanyaan, memberikan rekomendasi buku berdasarkan preferensi pengguna, dan membantu dalam proses transaksi. Pengaplikasian chatbot dalam e-commerce membuka potensi besar untuk meningkatkan pengalaman belanja online, memberikan informasi yang relevan, serta memfasilitasi proses tanya jawab yang lebih cepat dan efektif.

Salah satu metode yang mendominasi dalam membimbing alur percakapan chatbot adalah penggunaan decision tree. Decision tree adalah teknik untuk representasi data secara hierarkis. Decision tree menggunakan metode percabangan untuk menggambarkan setiap kemungkinan hasil dari suatu masalah. Struktur tree menunjukkan bagaimana satu pilihan mengarah ke yang berikutnya, dan penggunaan cabang menunjukkan bahwa setiap pilihan saling eksklusif. Sebuah pohon keputusan dapat digunakan untuk mengklasifikasi dan menemukan jawaban masalah yang kompleks [8].

Dalam proyek akhir ini kelompok kami memanfaatkan penggunaan decision tree dalam e-commerce toko buku yang kami berikan nama

Golebooks. Dengan adanya pemanfaatan chatbot ini harapannya toko dapat menyediakan jawaban instan atas pertanyaan pelanggan dengan efektif.

Penelitian ini bertujuan untuk mengevaluasi penggunaan decision tree dalam konteks pengembangan chatbot dengan berbagai variasi jumlah parameter pada model. Di samping itu, kami juga ingin mengidentifikasi strategi optimal dalam menyesuaikan parameter decision tree sehingga menghasilkan model yang tidak hanya memiliki tingkat akurasi yang tinggi, tetapi juga menunjukkan tingkat kesesuaian yang baik dengan berbagai variasi data masukan.

Pada penelitian ini, rumusan masalah yang ingin kami pecahkan adalah bagaimana penggunaan decision tree dengan variasi jumlah parameter mempengaruhi kinerja chatbot, terutama dalam hal akurasi dan potensi overfitting. Kami tertarik untuk membandingkan keberhasilan model awal chatbot dengan model setelah dilakukan tuning parameter. Melalui penelitian ini, kami berharap untuk mengidentifikasi strategi terbaik dalam menggunakan parameter decision tree untuk menghasilkan model chatbot yang responsif dan adaptif terhadap berbagai data masukan.

**Tinjauan Pustaka**

Decision tree adalah suatu teknik yang digunakan untuk menggambarkan data secara hierarkis. Metode percabangan digunakan dalam pohon keputusan untuk mengilustrasikan berbagai kemungkinan hasil dari suatu masalah. Struktur tree mencerminkan bagaimana satu pilihan membawa ke pilihan berikutnya, dan menunjukan bahwa penggunaan cabang setiap pilihan saling eksklusif.

Penggunaan metode decision tree dalam pengembangan chatbot untuk e-commerce menjadi fokus dalam penelitian ini. Sebelumnya, penelitian yang dilakukan oleh Jijo et al. [1] mencatat bahwa metode decision tree efektif digunakan untuk klasifikasi teks pada dataset dengan label yang tidak sedikit, seiring dengan hasil positif pada akurasi dan interpretabilitas model.

Dalam algoritma decision tree, terdapat proses pemilihan node root dan node-node setelahnya. Pemilihan label untuk menjadi node didasarkan pada *information gain,* di mana label yang memiliki nilai *gain* tertinggi akan dipilih. *Information gain* merupakan salah satu metrik yang digunakan untuk segmentasi dan sering disebut sebagai *mutual information* [1]. Salah satu cara untuk mengukur *information gain* adalah menggunakan *entropy*, yang dirumuskan sebagai berikut,

$$Gain(S, A) = Entropy(S) - \sum_{i=1}^{n} \frac{|S_i|}{|S|} * Entropy(S_i)$$

Gambar 1. Rumus Gain

di mana A adalah atribut S dengan n jumlah partisi pada A, dan *S*i adalah bagian dari S dengan nilai atribut ke-i. *Information gain* dapat diproses apabila kita telah menentukan *entropy* untuk setiap kasus pada label [1]. Sementara itu, entropy sendiri dirumuskan sebagai berikut,

$$Entropy(S) = -\sum_{i=1}^{n} p_i * \log_2 p_i$$

Gambar 2. Rumus Entropy

di mana pi adalah probabilitas label pada atribut ke-i.

Indeks GINI digunakan untuk mengevaluasi tingkat kemurnian kelas setelah dilakukan pemisahan berdasarkan suatu atribut. Tujuan dari pemisahan ini adalah untuk meningkatkan tingkat kemurnian pada set data yang dihasilkan dari proses tersebut. Semakin tinggi nilai Indeks GINI, semakin efektif pemilihan atribut dalam meningkatkan kemurnian kelas pada set data yang diproses. Dengan kata lain, atribut yang menghasilkan pemisahan dengan nilai Indeks GINI yang tinggi dianggap sebagai pilihan terbaik untuk meningkatkan kemurnian set data.

$$Gini(S) = 1 - \sum_{i=1}^{n} P_i^2$$

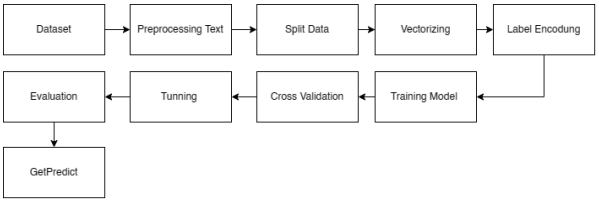
Gambar 3. Rumus Gini

Dalam konteks pemrosesan teks berbahasa Indonesia, penggunaan pustaka Sastrawi telah menjadi pilihan umum. Rosid et al. [5] mengemukakan penggunaan pustaka Sastrawi dalam tahap pra-pemrosesan dokumen menghasilkan hasil yang lebih baik dibandingkan dengan penelitian sebelumnya yang menggunakan algoritma Porter.

**Metode**

Penelitian ini menggunakan metode decision tree untuk mengembangkan chatbot e-commerce buku. Digunakan dua dataset berupa pertanyaan dan jawaban yang bersumber dari repositori berikut: https://github.com/ReinhartBW/ChatBot. Data yang pada awalnya masih berupa teks akan melalui tahap preprocessing, yakni penghapusan tanda baca, eliminasi stopword, dan stemming. Label pada dataset pertanyaan diubah menjadi format numerik melalui label encoding dan data selain label pada dataset pertanyaan diubah bentuknya menjadi dalam bentuk vektor dengan menggunakan *count vectorizing*.

Kemudian, dataset dibagi menjadi data pelatihan dan pengujian. Pembagian dilakukan dengan membagi dataset sebanyak 80% untuk melatih data dan 20% sisanya sebagai data pengujian. Tahap selanjutnya, dilakukan Cross validation yang berfungsi sebagai indikator apakah akurasi mengalami anomali seperti overfitting. Kemudian, proses tuning dilakukan menggunakan GridSearchCV. Hasil model yang optimal digunakan untuk memprediksi data uji, dan program dibuat untuk menghasilkan respon chatbot yang bervariasi dari dataset jawaban secara acak.




Gambar 4. Diagram Alur Metode

**Pembahasan**

Pengembangan chatbot, seperti yang disebutkan sebelumnya, menggunakan dua buah dataset, yakni pertanyaan serta jawaban. Pada dataset pertanyaan, masing-masing pertanyaan disertai dengan label yang menandakan kategori/*class* dari pertanyaan itu. Dataset pertanyaan terdiri dari 117 observasi dan 2 attribut dengan jumlah sebelas label *class*. Sementara itu, jumlah dataset jawaban terdiri dari 3 observasi dan 11 attribut. Berikut merupakan contoh dataset pertanyaan dan jawaban.

| | Question | Class |
|---|---|---|
| 0 | Berapa harga buku? | Book Price |
| 1 | Harga buku kena berapa? | Book Price |
| 2 | Buku dijual berapa? | Book Price |
| 3 | Bukunya berapa? | Book Price |
| 4 | Berapa bukunya? | Book Price |

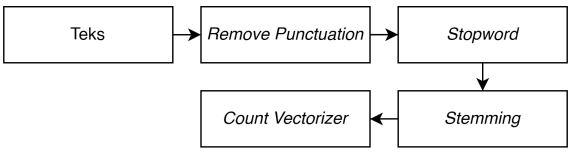
Gambar 5. Dataset Pertanyaan

| | Greeting | Morning Greeting | Noon Greeting | Night Greeting | Goods List | Book Price |
|---|---|---|---|---|---|---|
| 0 | Halo! Ada yang bisa saya bantu? | Selamat pagi! Ada yang bisa kami bantu? | Selamat siang! Ada yang bisa kami bantu? | Selamat malam! Ada yang bisa kami bantu? | Kami menjual beberapa barang seperti contohnya... | Kami menjual buku dengan harga 13000 per pak |
| 1 | Halo ! | Pagi juga ! Ada yang bisa saya bantu? | Siang juga ! Ada yang bisa saya bantu? | Malam juga ! Ada yang bisa Ada yang bisa saya ... | Toko ini menjual beberapa perlengkapan seperti... | Harga buku yang dijual di toko kami adalah seb... |

Gambar 6. Dataset jawaban

Tujuan dari pengolahan teks ini adalah untuk melakukan klasifikasi teks pertanyaan yang nantinya akan menghasilkan luaran berupa kelas label. Kemudian, label yang dihasilkan sebagai luaran tersebut akan digunakan untuk menghasilkan jawaban yang diambil secara random dari dataset jawaban berdasarkan nama kolom yang sesuai dengan label hasil klasifikasi teks.

Pada penelitian ini, digunakan beberapa pustaka, diantaranya pandas, sklearn, matplotlib, random, sastrawi, dan re. Sebelum masuk ke dalam proses modeling, data teks perlu dilakukan beberapa tahap pemrosesan terlebih dahulu. Tahap pertama dalam pemrosesan teks adalah *remove punctuation* yang ditujukan untuk menghapus simbol-simbol yang tidak diperlukan. Kemudian dilanjutkan tahap stopword yang berguna untuk menghapus kata-kata yang tidak relevan diikuti tahapan *stemming* untuk mengubah kata berimbuhan menjadi kata dasar, dan diakhiri dengan tahap vectorizing menggunakan teknik count vectorizer untuk mengubah bentuk teks menjadi bentuk vektor sehingga dapat diolah oleh komputer. Pemrosesan teks banyak menggunakan pustaka sastrawi karena memiliki fungsi pemrosesan teks yang telah disesuaikan dengan bahasa Indonesia. Berikut merupakan diagram alur dari pemrosesan teks yang dilakukan.




Gambar 7. Diagram Alur Preprocessing Teks

Setelah pemrosesan teks yang dilakukan pada dataset pertanyaan terkhusus pada kolom “Question”, tahap selanjutnya adalah melakukan proses *label encoding* untuk mengubah label pada kolom “Class” menjadi dalam bentuk numerik seperti pada gambar berikut.

```
[41] le = LabelEncoder()
     label_encoded = le.fit_transform(df.Class)
     print(label_encoded)

     [ 0  0  0  0  0  0  0  0  0  0  0  0  1  1  1  1  1  1  1  1  1  1  1
       1  2  2  2  2  2  2  2  2  2  2  3  3  3  3  3  3  3  3  3  3  3  4
       4  4  4  5  5  5  5  5  5  6  6  6  6  6  6  7  7  7  7  7  7  8  8  8
       8  8  8  8  8  9  9  9  9  9  9  9  9  9  9  9  9  9  9  9 10 10 10
      10 10 10 10 10 10 10 10 10]
```

Gambar 8. Hasil *Label Encoding*

Setelah dataset pertanyaan siap untuk digunakan untuk melakukan pelatihan model. Selanjutnya akan dilakukan proses splitting data menjadi X_train, X_test, y_train, y_test dengan persentase data *training* sebesar 0.80 dan data *testing* sebesar 0.20 serta menggunakan parameter stratify agar distribusi setiap label pada data *training* dan *testing* menjadi merata.

```
X_train,X_test, y_train,y_test = train_test_split(
    fitur, label_encoded,stratify=label_encoded,
    test_size = 0.2, random_state=42)
vectorizer = CountVectorizer()
vectorizer.fit(X_train)
X_train = vectorizer.transform(X_train)
X_test = vectorizer.transform(X_test)
```

Gambar 9. Proses *Splitting* Data dan *Vectorizing*

Selanjutnya, dilakukan proses modeling menggunakan model algoritma *decision tree* yang berasal dari pustaka sklearn dengan parameter max_dept bernilai 10 dan random _random state bernilai 42. Model ini menghasilkan akurasi dengan data testing sebesar 0.875 dan data uji training sebesar 0.935.

```
[82] from sklearn import tree
     model = tree.DecisionTreeClassifier(max_depth= 10, random_state = 42).fit(X_train,y_train)
     model.score(X_test,y_test), model.score(X_train,y_train)

     (0.875, 0.9354838709677419)
```

Gambar 10. Akurasi Model

Untuk memastikan bahwa model tidak mengalami overfitting, dilakukan teknik *cross validation* menggunakan pustaka sklearn. Dataset pertanyaan akan dibagi menjadi 4 sub bagian guna dilakukan proses pengujian. Hasil dari teknik *cross validation* menunjukan empat buah akurasi yakni 0.7941, 0.8727, 0.7324, 0.8257 dengan rata-ratanya sebesar 0.80627. Rata-rata hasil dari teknik *cross validation* yang dilakukan memiliki nilai yang tidak jauh berbeda dengan akurasi nilai model yang telah diuji sebelumnya sehingga dapat dikatakan bahwa model tersebut tidak mengalam*i overfitting*.

```
from sklearn import metrics
from sklearn.model_selection import cross_val_score
import numpy as np
fitur_vec = vectorizer.transform(fitur)
scores = cross_val_score(
model, fitur_vec, label_encoded, cv=4, scoring='f1_macro')

print(scores)
print(np.mean(scores))
```

```
[0.79417552 0.87272727 0.73246753 0.82571166]
0.8062704971795882
```

Gambar 11. Hasil *Cross Validation*

Meskipun memiliki akurasi yang cukup baik, model tersebut masih dapat dilakukan proses *tuning* untuk meningkatkan akurasinya dengan menggunakan beberapa kombinasi parameter tambahan lainnya. Proses tuning dilakukan dengan menggunakan *GridSearchCV. GridSearchCV* akan mencari kombinasi parameter untuk mendapatkan hasil akurasi yang terbaik sekaligus dilakukan proses *Cross Validation* untuk melakukan pencegahan model yang overfitting. Parameter yang akan dilakukan proses pencarian kombinasi terbaik antara lain parameter *criterion*, *max_depth*, *ccp_alpha*, *splitter*, dan *max_features*.

```
[96] parameter = {
         "criterion":['gini','entropy'],
         "max_depth": range(1,15),
         "ccp_alpha": [0.1, 0.01, 0.001],
         "splitter": ['best', 'random'],
         "max_features":['sqrt', 'log2']
     }
```

Gambar 12. Parameter pada Hyperparameter Tuning

Proses pencarian kombinasi parameter model terbaik menggunakan *GridSearchCV* mengunakan cv sebanyak 4, sehingga akan terbentuk 4 sub bagian pada teknik *cross validation* yang digunakan.

```
from sklearn.model_selection import GridSearchCV
grid = GridSearchCV(model, param_grid = parameter,
                    cv= 4, verbose = 1, n_jobs=-1)
grid.fit(X_train,y_train)

Fitting 4 folds for each of 336 candidates, totalling 1344 fits
▸ GridSearchCV
▸ estimator: DecisionTreeClassifier
  ▸ DecisionTreeClassifier
```

Gambar 13. *Hyperparameter Tuning*

Hasil dari *GridSearchCV* didapatkan nilai akurasi *train* dan *test* sebesar 0.93 dan 0.70 dengan parameter terbaiknya antara lain ccp_alpha: 0.01, criterion: 'entropy', max_depth: 14, max_features: 'log2', random_state: 42, splitter: 'random'.

```
[100] grid.best_estimator_

DecisionTreeClassifier(ccp_alpha=0.01, criterion='entropy', max_depth=14,
                       max_features='log2', random_state=42, splitter='random')
```

```
best = tree.DecisionTreeClassifier(ccp_alpha=0.01,criterion='entropy',
                                   max_depth=14, max_features='log2',
                                   random_state=42, splitter='random')
best.fit(X_train,y_train)
best.score(X_train,y_train), best.score(X_test, y_test)

(0.9354838709677419, 0.7083333333333334)
```

Gambar 14. Hasil *Hyperparameter Tuning*

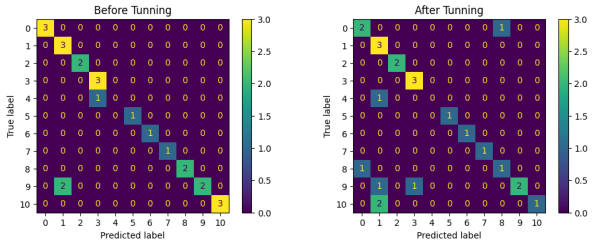
Dengan akurasi best score yang mengalami penurunan pada *testing*, ada kemungkinan terjadi overfitting pada model yang dilakukan *tuning*. Model yang overfitting ini tidak bagus karena tidak cukup baik untuk melakukan prediksi dengan menggunakan dataset yang sebelumnya belum pernah dipelajari. Untuk memastikan overfitting, dilakukan cross validation untuk model yang telah dilakukan tuning, dan didapatkan adanya nilai rata-rata akurasi yang sangat rendah dengan sebaran akurasi di masing-masing sub bagian tidak merata karena ada sub bagian yang memiliki akurasi sangat tinggi yakni 0.84 dan ada sub bagian yang memiliki akurasi sangat rendah yakni 0.450 sehingga ada potensi overfitting pada model yang telah dilakukan tuning.

```
scores_tun= cross_val_score(
best, fitur_vec, label_encoded, cv=4, scoring='f1_macro')
print(scores_tun)
print(np.mean(scores_tun))
```

```
[0.74350649 0.67412587 0.45064935 0.8459596 ]
0.6785603285603286
```

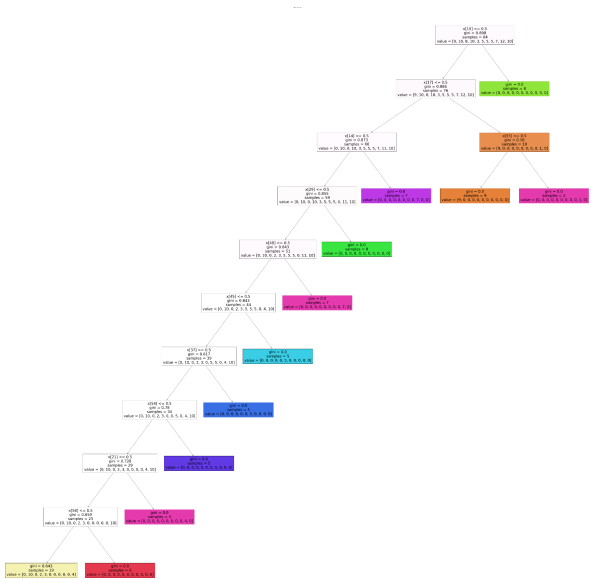
Gambar 15. Hasil *Cross Validation Hyperparameter Tuning*

Untuk memastikan model yang lebih baik digunakan, dilakukan perhitungan ketepatan model dalam memprediksi data testing dengan menggunakan confusion matrix berikut.




Gambar 16. *Confusion Matrix* pada data *testing*

Terlihat bahwa model yang sebelum dilakukan tuning memprediksi lebih baik dibandingkan model yang telah dilakukan tuning. Oleh karena itu, model yang akan digunakan pada chatbot ini akan menggunakan model yang pertama kali telah dilatih sebelum melakukan proses *tuning*. Berikut merupakan tree model yang dihasilkan atas proses *training* yang sebelumnya dilakukan



Gambar 17. Topologi *Tree*

Setelah mendapatkan model yang mampu melakukan prediksi kelas label dengan baik. Selanjutnya akan dilakukan proses prediksi data uji. Data uji merupakan data yang sebelumnya belum pernah dipelajari dan berbentuk pertanyaan yang dimasukkan ke dalam chatbot. Data uji akan dilakukan tahap pemrosesan teks terlebih dahulu dan diubah menjadi vector untuk dilakukan prediksi. Kemudian setelah prediksi dilakukan hasil yang dikeluarkan model akan dilakukan proses *inverse encoding* untuk mengembalikan representasi numerik menjadi bentuk string sesuai dengan label semula sebelum dilakukan proses encoding.

```
def get_predict(masukkan):
  masukkan = stem(stop_words(remove_punc(masukkan)))
  masukkan = vectorizer.transform([masukkan])
  predict = model.predict(masukkan)
  return le.inverse_transform(predict)[0]
```

Gambar 18. Fungsi untuk Prediksi Data baru

Agar chatbot mampu memberikan respon jawaban yang sesuai, dibuatlah program untuk mengambil respon yang berasal dari dataset jawaban berdasarkan kolom yang sesuai dengan hasil prediksi secara random.

```
[70] masukkan=""
     while masukkan != "stop":
       masukkan = input("User: ")
       if masukkan != "stop":
         print("Chatbot: ", random.choice(answer[get_predict(masukkan)]))
       else:
         print("Terima kasih sudah bertanya!:D")
```

Gambar 19. Program Utama Chatbot

Sebagai contoh, ketika diberikan data testing baru “Pagi”, akan dilakukan prediksi oleh model yang mengeluarkan output label “Morning Greeting”.

```
masukkan=""
while masukkan != "stop":
  masukkan = input("User: ")
  if masukkan != "stop":
    print("Chatbot: ", random.choice(answer[get_predict(masukkan)]))
  else:
    print("Terima kasih sudah bertanya!:D")

User: Pagi
Chatbot:  Pagi juga ! Ada yang bisa saya bantu?
User:
```

Gambar 20. Contoh Jawaban Chatbot

Kemudian, akan diambil nilai secara random dari kolom “Morning Greeting” yang menjadi output prediksi dari model yang dalam hal ini terpilih nilai “Pagi juga ! Ada yang bisa saya bantu?”

| | Greeting | Morning Greeting | Noon Greeting | Night Greeting |
|---|---|---|---|---|
| 0 | Halo! Ada yang bisa saya bantu? | Selamat pagi! Ada yang bisa kami bantu? | Selamat siang! Ada yang bisa kami bantu? | Selamat malam! Ada yang bisa kami bantu? |
| 1 | Halo ! | Pagi juga ! Ada yang bisa saya bantu? | Siang juga ! Ada yang bisa saya bantu? | Malam juga ! Ada yang bisa Ada yang bisa saya ... |
| 2 | Hai, ada yang bisa dibantu? | Selamat pagi juga ! Ada yang bisa dibantu? | Selamat siangi juga ! Ada yang bisa dibantu? | Selamat malam juga ! Ada yang bisa dibantu? |

Gambar 21. Gambar dataset Jawaban

## Kesimpulan

1. Setelah dilakukan, hyperparameter tuning, model awal dengan parameter max_depth: 10 dan random_state: 42 memiliki kemampuan yang lebih baik dalam memprediksi data secara general dibandingkan dengan akurasi model setelah dilakukan proses tuning yang didukung dengan hasil yang ditunjukkan pada confusion matrix yang menunjukkan model sebelum tuning lebih baik dalam memprediksi data baru (data testing) dibandingkan model setelah tuning.

2. Model hasil tuning mengalami overfitting yang ditunjukkan pada hasil cross-validation yang tidak konsisten (adanya akurasi pada salah satu sub bagian yang tinggi sedangkan pada sub lain rendah).

## Saran

Jumlah parameter yang digunakan untuk tuning perlu dikurangi sehingga tidak terlalu spesifik guna menghasilkan tree model yang *robust* untuk semua data masukkan yang belum pernah ditemui oleh model sebelumnya. Perlu juga digunakan metode lain untuk melakukan proses hyperparameter tuning selain GridSearchCV

## Daftar Pustaka


[1] B. T. Jijo, A. M. Abdulazeez, "Classification based on decision tree algorithm for machine learning," JASTT, vol. 2, no. 01, pp. 20-28, March 2021. doi:10.38094/jastt20165

[2] D. Adiwardana, M. T. Luong, D. R. So, J. Hall, N. Fiedel, R. Thoppilan, Z. Yang, A. Kulshreshtha, G. Nemade, Y. Lu, Q. V. Le, "Towards a human-like open-domain chatbot," arXiv, 2001.09977v3 (cs.CL), pp. 1-15, February 2020. doi:10.48550/arXiv.2001.09977

[3] I. W. Puspitasari, F. R. Rinawan, W. G. Purnama, H. Susiarno, A. I. Susanti, "Development of a chatbot for pregnant women on a posyandu application in indonesia: from qualitative approach to decision tree method," Informatics, vol. 9, no. 88, pp. 1-30, October 2022. doi:10.3390/informatics9040088

[4] J. Jia, "The study of the application of a keywords-based chatbot system on the teaching of foreign languages," arXiv:cs/0310018 [cs.CY], pp. 1-1, 2003. doi:10.48550/arXiv.cs/0310018

[5] M. A. Rosid, A. S. Fitrani, I. R. I. Astutik, N. I. Mulloh, H. A. Gozali, "Improving text preprocessing for student complaint document classification using sastrawi," IOP Conf. Series: Materials Science and Engineering, no. 874, pp. 1-6, 2020. doi:10.1088/1757-899X/874/1/012017

[6] M. Mustaqim, A. Gunawan, Y. B. Pratama, I. Zaliman, "Pengembangan chatbot layanan publik menggunakan machine learning dan natural languange processing," Journal of Information Technology and Society, vol. 1, no. 1, pp. 1-3, June 2023.

[7] P. D. L. Santoso, I. Riski, N. Kholik, M. R. Akbar, A. Saifudin, Yulianti, "Penerapan artificial intelligence dalam aplikasi chatbot sebagai media informasi dan pembelajaran mengenai kebudayaan bangsa," Jurnal Informatika Universitas Pamulang, vol.6, no. 3, pp. 579-589, November 2021. doi:10.32493/informatika.v6i3.11845

[8] P. D. Prasetyo, F. A. Yasril, L. M. A. Tachtiar, A. A. Sianturi, A. Hakim, M. Reyhan, "Prediksi kualitas wine menggunakan metode decision tree," IPB University, 2022.

[9] T. Castle-Green, S. Reeves, J. F. Fischer, B. Koleva, "Decision trees as sociotechnical objects in chatbot design," CUI, pp. 22-24, July 2020. doi:10.1145/3405755.3406133

[10] Y. Zhang, S. Sun, M. Galley, Y. C. Chen, C. Brockett, X. Gao, J. Gao, J. Liu, B. Dolan, "Dialogpt : large-scale generative pre-training for conversational response generation," arXiv. 1911.00536v3 (cs.CL), pp. 1-7, May 2020. doi:10.48550/arXiv.1911.00536