**Laporan Akhir**

**Analisis Sentimen Masyarakat terhadap Tim Nasional Indonesia pada Piala AFF 2020 Menggunakan Algoritma K-Nearest Neighbors**

****

**Oleh Kelompok 2:**

1. Mochammad Rizki Aji Santoso (200411100086)
2. Baihaki (200411100181)
3. Muhammad Ilham Anggis Bangkit Pamungkas (200411100197)

**Dosen Pengampu :**

Dr. Fika Hastarita Rachman, S.T., M.Eng..

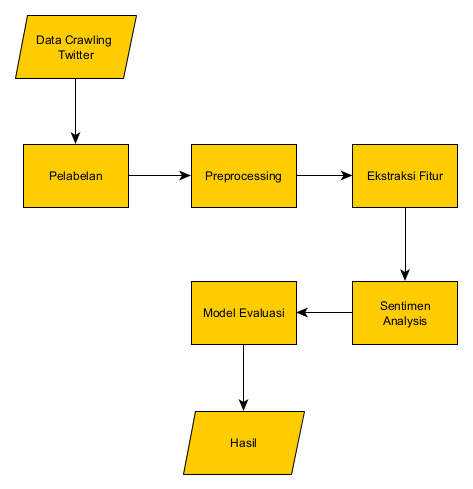
**PROGRAM STUDI TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK**

**UNIVERSITAS TRUNOJOYO MADURA**

**2023**

**Arsitektur Sistem**



**Copy Code Sesuai Arsitektur Sistem**

1. Import file #pssi dari github

| data1 = pd.read\_csv('https://raw.githubusercontent.com/rizkyluxszerr/pba/main/pssi.csv') |
| --- |

1. Import file #timnasindonesia dari github

| data2 = pd.read\_csv('https://raw.githubusercontent.com/rizkyluxszerr/pba/main/timnasindonesia.csv') |
| --- |

1. Import file #pssi dari github

| data3 = pd.read\_csv('https://raw.githubusercontent.com/rizkyluxszerr/pba/main/pialaaff2020.csv') |
| --- |

1. Penggabungan 3 file

| gabung = pd.concat([data1, data2, data3], axis=0)  gabung |
| --- |

1. Pelabelan

| import nltk  #melakukan berbagai operasi pada data teks, seperti tokenisasi, stemming, lemmatisasi, analisis sentimen, klasifikasi teks, dan lain sebagainya  nltk.download('vader\_lexicon')  #untuk mengunduh dataset yang digunakan untuk analisis sentimen menggunakan VADER  from nltk.sentiment import SentimentIntensityAnalyzer  #mengimpor kelas SentimentIntensityAnalyzer dari modul sentiment  #digunakan untuk melakukan analisis sentimen pada data teks  # Menginisialisasi sentiment analyzer  sia = SentimentIntensityAnalyzer()  # Membuat fungsi untuk memberikan label pada setiap teks tweet  def label\_tweet(tweet):  score = sia.polarity\_scores(tweet)  if score['compound'] >= 0.05:  return 1 # tweet bernilai positif  elif score['compound'] <= -0.05:  return 2 # tweet bernilai negatif  else:  return 0 # tweet bernilai netral  # Menerapkan fungsi label\_tweet pada setiap teks tweet dan menambahkan kolom label pada dataframe  gabung['label'] = gabung['text'].apply(lambda x: label\_tweet(x))  # Menampilkan 5 baris pertama dari dataframe  # print(df)  gabung |
| --- |

1. Preprocessing

| !pip install Sastrawi #menginstal package Sastrawi, untuk melakukan proses stemming pada teks berbahasa Indonesia.  import Sastrawi # mengimpor seluruh modul Sastrawi  import numpy as np #ntuk melakukan operasi pada array atau matriks  from sklearn.feature\_extraction.text import CountVectorizer # ekstraksi fitur dari teks dengan cara menghitung frekuensi kemunculan setiap kata  import string # berguna dalam manipulasi string  import re #untuk mengelola dan memanipulasi string menggunakan ekspresi reguler  nltk.download('stopwords') #tidak memiliki makna yang signifikan |
| --- |

* Hapus user

| def remove\_pattern(input\_txt, pattern):      r = re.findall(pattern, input\_txt)      for i in r:          input\_txt = re.sub(i, '', input\_txt)      return input\_txt  df['remove\_user']=np.vectorize(remove\_pattern)(df['text'], "@[\w]\*")  df |
| --- |

* Hapus Http

| def remove(tweet):  #remove angka  tweet = re.sub('[0-9]+', '', tweet)  # remove stock market tickers like $GE  tweet = re.sub(r'\$\w\*', '', tweet)  # remove old style retweet text "RT"  tweet = re.sub(r'^RT[\s]+', '', tweet)  # remove hashtags  # only removing the hash # sign from the word  tweet = re.sub(r'#', '', tweet)  return tweet  df['remove\_http'] = df['remove\_user'].apply(lambda x:remove(x))  # df.sort\_values("remove\_http", inplace = True)  df.drop\_duplicates(subset ="remove\_http", keep = 'first', inplace = True) |
| --- |

* Import modul stopwords

| #import stopword  from nltk.corpus import stopwords  stopwords\_indonesia = stopwords.words('indonesian') |
| --- |

* Import sastrawi dan nltk untuk tokenizer

| #import sastrawi  from Sastrawi.Stemmer.StemmerFactory import StemmerFactory  factory = StemmerFactory()  stemmer = factory.create\_stemmer()  #tokenize  from nltk.tokenize import TweetTokenizer    # Happy Emoticons  emoticons\_happy = set([  ':-)', ':)', ';)', ':o)', ':]', ':3', ':c)', ':>', '=]', '8)', '=)', ':}',  ':^)', ':-D', ':D', '8-D', '8D', 'x-D', 'xD', 'X-D', 'XD', '=-D', '=D',  '=-3', '=3', ':-))', ":'-)", ":')", ':', ':^', '>:P', ':-P', ':P', 'X-P',  'x-p', 'xp', 'XP', ':-p', ':p', '=p', ':-b', ':b', '>:)', '>;)', '>:-)',  '<3'  ])  # Sad Emoticons  emoticons\_sad = set([  ':L', ':-/', '>:/', ':S', '>:[', ':@', ':-(', ':[', ':-||', '=L', ':<',  ':-[', ':-<', '=\\', '=/', '>:(', ':(', '>.<', ":'-(", ":'(", ':\\', ':-c',  ':c', ':{', '>:\\', ';('  ])  # all emoticons (happy + sad)  emoticons = emoticons\_happy.union(emoticons\_sad) |
| --- |

* Tweet Cleaning

| def clean\_tweets(tweet):  # remove stock market tickers like $GE  tweet = re.sub(r'\$\w\*', '', tweet)    # remove old style retweet text "RT"  tweet = re.sub(r'^RT[\s]+', '', tweet)    # remove hyperlinks  tweet = re.sub(r'https?:\/\/.[\r\n]', '', tweet)    # remove hashtags  # only removing the hash # sign from the word  tweet = re.sub(r'#', '', tweet)    #remove coma  tweet = re.sub(r',','',tweet)    #remove angka  tweet = re.sub('[0-9]+', '', tweet)    # tokenize tweets  tokenizer = TweetTokenizer(preserve\_case=False, strip\_handles=True, reduce\_len=True)  tweet\_tokens = tokenizer.tokenize(tweet)    tweets\_clean = []  for word in tweet\_tokens:  if (word not in stopwords\_indonesia and # remove stopwords  word not in emoticons and # remove emoticons  word not in string.punctuation): # remove punctuation  #tweets\_clean.append(word)  stem\_word = stemmer.stem(word) # stemming word  tweets\_clean.append(stem\_word)  return tweets\_clean  df['tweet\_clean'] = df['remove\_http'].apply(lambda x: clean\_tweets(x)) |
| --- |

Fungsi untuk menghapus kolom []

| #remove punct  def remove\_punct(text):  text = " ".join([char for char in text if char not in string.punctuation])  return text  df['Tweet'] = df['tweet\_clean'].apply(lambda x: remove\_punct(x)) |
| --- |

1. Ekstraksi fitur menggunakan tf idf

| # Import fungsi tf-idf  From sklearn.feature\_extraction.text import TfidfVectorizer,CountVectorizer |
| --- |

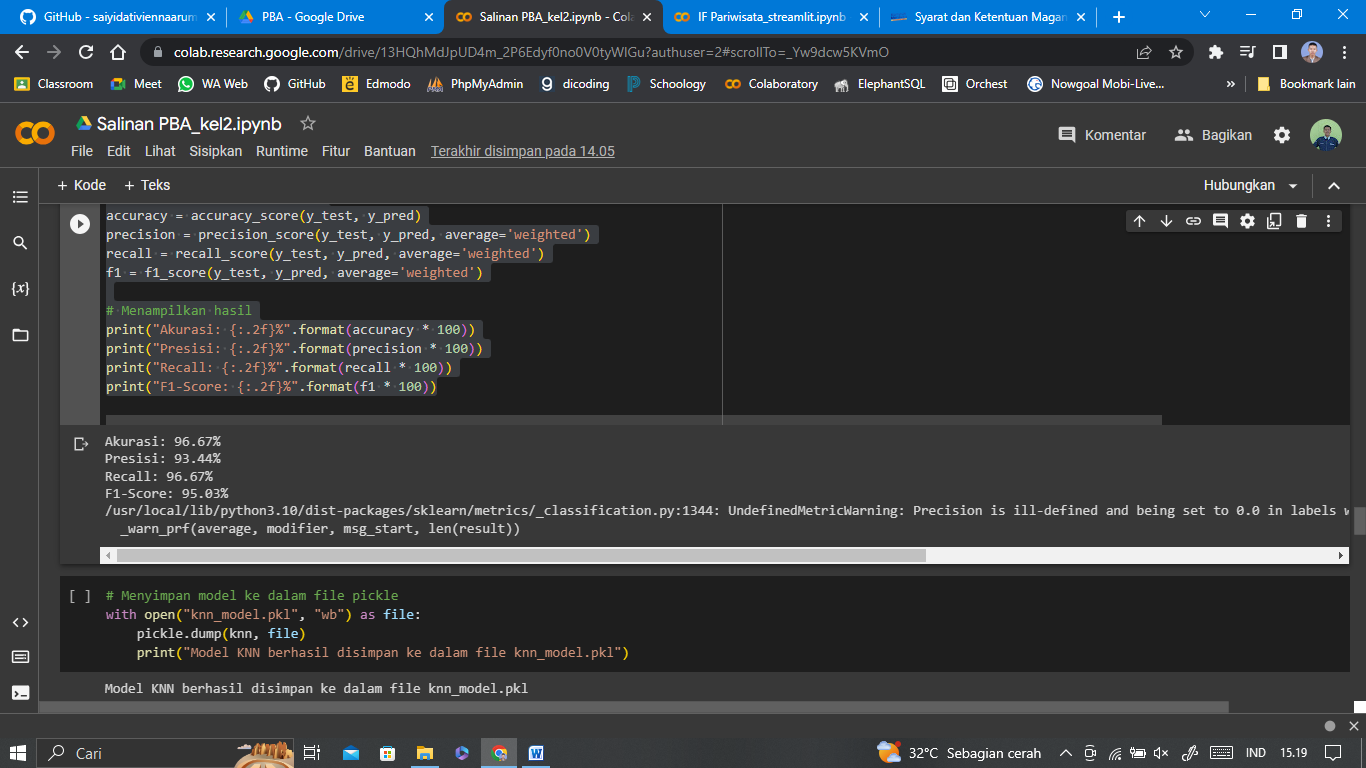
| # Inisialisasi objek TfidfVectorizer  tfidf\_vectorizer = TfidfVectorizer()  # Proses ekstraksi fitur TF-IDF  # Metode ini akan menghitung bobot TF-IDF untuk setiap kata dalam teks dan menghasilkan matriks TF-IDF yang mewakili teks tersebut.  # pd.DataFrame. toarray() digunakan untuk mengubah matriks sparse menjadi matriks denser, dan columns diatur menggunakan metode get\_feature\_names\_out()  # dari objek TfidfVectorizer. Kolom-kolom dataframe akan menjadi fitur-fitur TF-IDF yang diekstraksi, dan nilai-nilai dalam matriks TF-IDF akan menjadi  # nilai-nilai dalam dataframe.  tfidf\_matrix = tfidf\_vectorizer.fit\_transform(df['Tweet'])  # Membuat dataframe dari matriks TF-IDF  tfidf\_df = pd.DataFrame(tfidf\_matrix.toarray(), columns=tfidf\_vectorizer.get\_feature\_names\_out()) |
| --- |

| # Menyimpan vektorizer ke dalam file pickle  with open("vectorizer.pkl", "wb") as file:  pickle.dump(tfidf\_vectorizer, file)  print("Vektorizer berhasil disimpan ke dalam file vectorizer.pkl") |
| --- |

1. Modelling dengan KNN

| from sklearn.feature\_extraction.text import TfidfVectorizer  from sklearn.neighbors import KNeighborsClassifier  from sklearn.model\_selection import train\_test\_split  from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score  # Memisahkan fitur (tweet) dan label (sentimen) menjadi variabel X dan y.  # Kemudian, kita membagi dataset menjadi data pelatihan dan data pengujian menggunakan train\_test\_split dari scikit-learn.  # X = df['Tweet']  X = tfidf\_df  y = df['label']  # Membagi dataset menjadi data pelatihan dan data pengujian  # test\_size: Ukuran yang ingin Anda alokasikan untuk data testing, misalnya 0.2 untuk 20% data testing dan 80% data training.  # random\_state: Nilai yang ditentukan agar pemisahan data menjadi data training dan data testing dapat direproduksi dengan cara yang sama jika dikembalikan pada waktu yang berbeda.  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.1, random\_state=42)  # Membuat objek k-NN classifier dengan k=5  # untuk menentukan k=5 dalam k-NN.  knn = KNeighborsClassifier(n\_neighbors=3, metric='euclidean')  # Melatih model k-NN  # melatih model menggunakan data pelatihan dengan fit().  # knn.fit(X\_train\_vectorized, y\_train)  knn.fit(X\_train, y\_train)  # Melakukan prediksi pada data pengujian  y\_pred = knn.predict(X\_test)  # Menghitung metrik evaluasi  accuracy = accuracy\_score(y\_test, y\_pred)  precision = precision\_score(y\_test, y\_pred, average='weighted')  recall = recall\_score(y\_test, y\_pred, average='weighted')  f1 = f1\_score(y\_test, y\_pred, average='weighted')  # Menampilkan hasil  print("Akurasi: {:.2f}%".format(accuracy \* 100))  print("Presisi: {:.2f}%".format(precision \* 100))  print("Recall: {:.2f}%".format(recall \* 100))  print("F1-Score: {:.2f}%".format(f1 \* 100)) |
| --- |

1. Model Evaluasi



**Hasil Skenario Uji Coba**

Dalam penelitian ini memiliki 5 skenario uji coba pemodelan KNN, yaitu dengan percobaan mengubah nilai dari test size (ukuran yang ingin dialokasikan untuk data testing), random state (nilai yang ditentukan agar pemisahan data dapat direproduksi dengan cara yang sama jika kembali pada waktu berbeda) , dan n tetangga (jumlah tetangga terdekat yang akan digunakan dalam proses).

Test Size : 0,1 | Random State : 42 | n tetangga : 3

| Evaluasi | Hasil |
| --- | --- |
| Akurasi | 96.67% |
| Presisi | 93.44% |
| Recall | 96.67% |
| F1-Score | 95.03% |

Test Size : 0,2 | Random State : 42 | n tetangga : 2

| Evaluasi | Hasil |
| --- | --- |
| Akurasi | 91.53% |
| Presisi | 83.77% |
| Recall | 91.53% |
| F1-Score | 87.48% |

Test Size : 0,4 | Random State : 40 | n tetangga : 4

| Evaluasi | Hasil |
| --- | --- |
| Akurasi | 94.02% |
| Presisi | 89.96% |
| Recall | 94.02% |
| F1-Score | 91.95% |

Test Size : 0,3 | Random State : 40 | n tetangga : 5

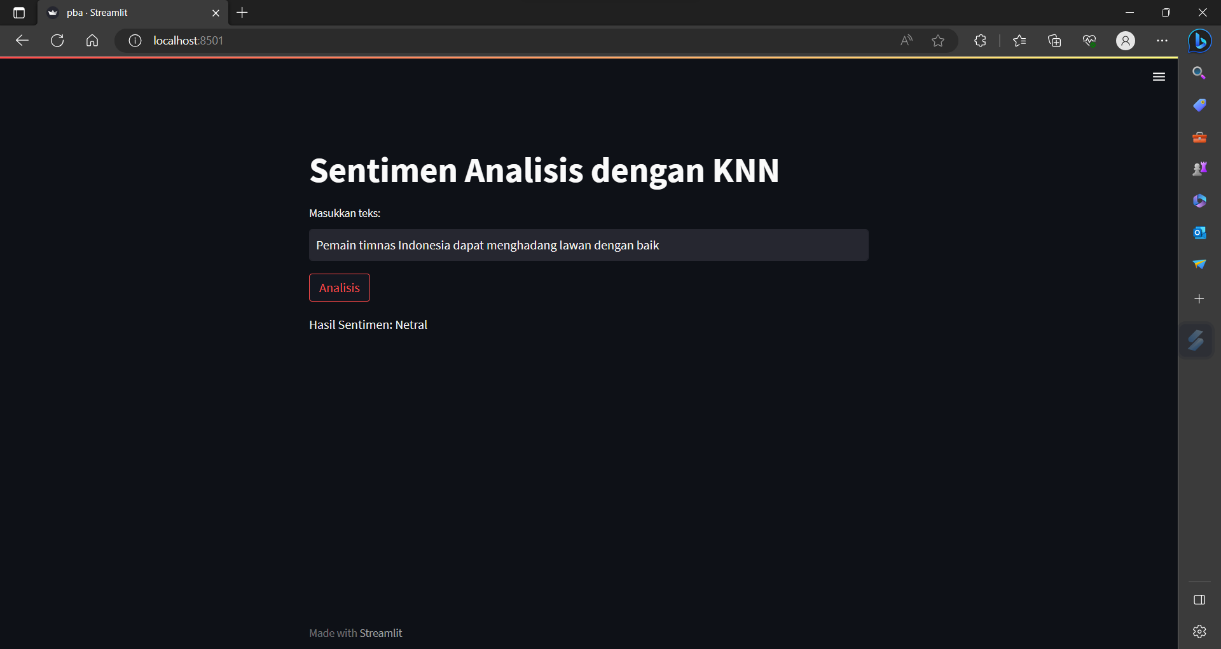
| Evaluasi | Hasil |
| --- | --- |
| Akurasi | 93.18% |
| Presisi | 88.90% |
| Recall | 93.18% |
| F1-Score | 90.99% |

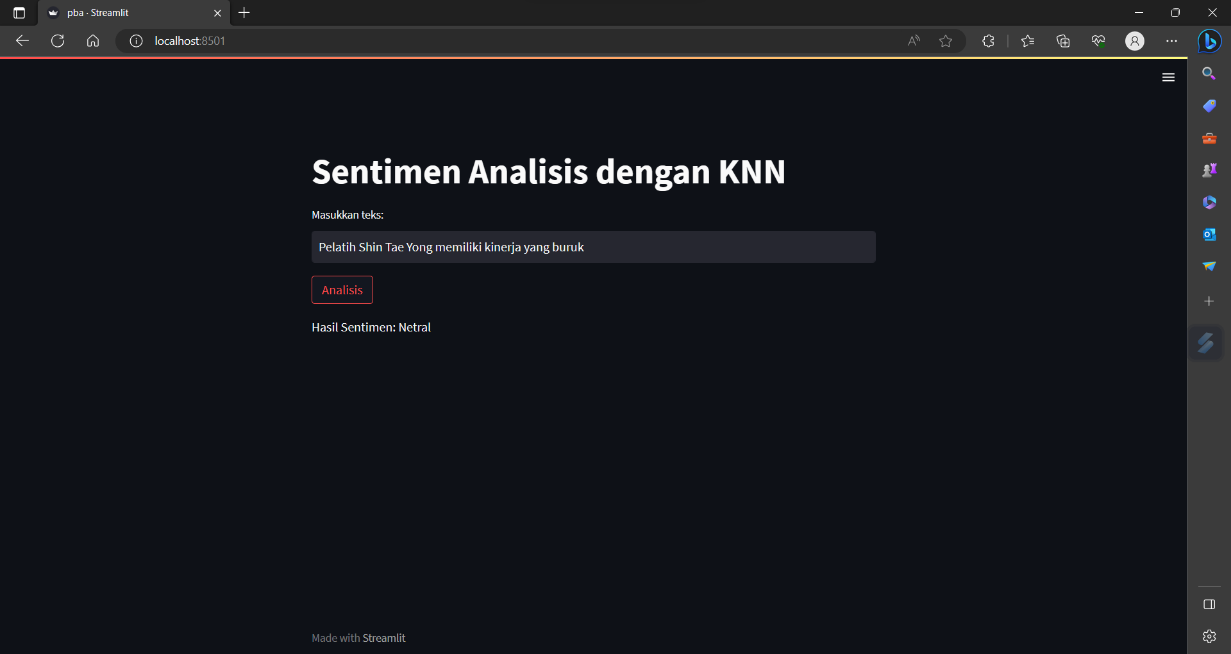
Test Size : 0,3 | Random State : 45 | n tetangga : 2

| Evaluasi | Hasil |
| --- | --- |
| Akurasi | 88.64% |
| Presisi | 84.35% |
| Recall | 88.64% |
| F1-Score | 85.95% |

* Akurasi : Mengukur sejauh mana model klasifikasi memberikan prediksi yang benar secara keseluruhan. Ini dihitung dengan membagi jumlah prediksi yang benar dengan total jumlah sampel. Meskipun akurasi penting, metrik ini dapat menjadi bias jika terjadi ketidakseimbangan kelas dalam dataset.
* Presisi : Mengukur sejauh mana prediksi positif yang diberikan oleh model benar. Ini adalah rasio jumlah predikasi positif yang benar terhadap total prediksi positif. Presisi memberikan informasi tentang seberapa banyak yang diprediksi positif benar dari semua kasus yang diprediksi positif.
* Recall : Mengukur sejauh mana model dapat menemukan semua kasus positif yang ada. Ini adalah rasio dari jumlah prediksi positif yang benar terhadap total jumlah kasus positif yang sebenarnya. Recall memberikan informasi tentang seberapa banyak kasus positif yang dapat ditemukan oleh model.
* F1 Score : Ukuran yang menggabungkan presisi dan recall menjadi satu angka tunggal. Ini adalah harmonic mean antara presesi dan recall. Berguna ketika terjadi ketidakseimbangan antar kelas positif dan negative dalam dataset, karena mempertimbangakan keduanya dalam perhitungannya.

**Implementasi Interface**

****

****

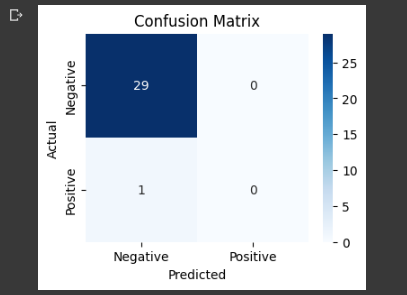
Dalam hasil pada interface, bisa dilihat bahwa sentimen yang didapat pada semua inputan adalah netral. Hal ini disebabkan pada saat pelabelan yang dilakukan oleh sistem coding tidak begitu akurat atau sama jika dilakukan pelabelan secara manual.

**Analisa Uji Coba**

* **Menampilkan true negative dan false positive**

Dari hasil uji coba scenario sebelumnya maka diperoleh hasil akurasi terbaik  dari scenario dengan menggunakan data oversampling dengan split data  testing sebesar 10%, random state 42 dan jumlah tetangga terdekat sebanyak 3.

| import matplotlib.pyplot as plt  import seaborn as sns  from sklearn.metrics import confusion\_matrix  cm = confusion\_matrix(y\_test, y\_pred)  plt.figure(figsize=(4, 3))  sns.heatmap(cm, annot=True, cmap='Blues', fmt='d', xticklabels=['Negative', 'Positive'], yticklabels=['Negative', 'Positive'])  plt.xlabel('Predicted')  plt.ylabel('Actual')  plt.title('Confusion Matrix')  plt.show() |
| --- |

****

* **Saran Pengembangan Selanjutnya**

Saran untuk pengembangan sistem, pada metode pelabelan terdapat kerancuan dimana hasil yang didapat kebanyakan bernilai netral. Jadi, hal tersebut dapat membuat output yang didapatkan pasti tidak sesuai dengan kata atau kalimat yang diinputkan. Hal ini dapat diatasi dengan menggunakan metode untuk pelabelan yang lain dengan tingkat akurasi yang tinggi, sehingga dapat memberikan output yang sesuai ketika dijalankan nantinya.