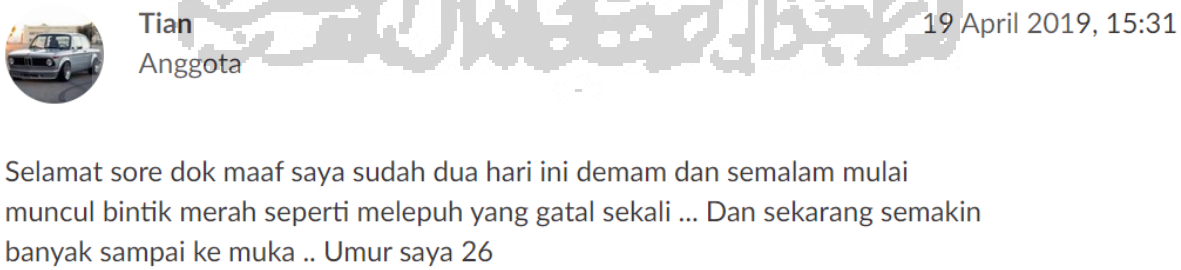
BAB IV
HASIL DAN PEMBAHASAN

4.1 Teks Klinis

Data yang digunakan pada teks klinis ini merupakan data yang telah ditulis secara manual oleh peneliti berdasarkan hasil rekaman suara dari pasien yang diambil langsung di Klinik Polifarma UII dan data dari alodokter.com. Adapun data untuk *training* terdiri dari 35 data dari dokter dan 106 dari alodokter.com. Berdasarkan data tersebut dilakukan pemetaan ke dalam komponen keluhan pasien (keluhan utama, onset, keluhan lain, keterangan, frekuensi serangan, sifat serangan, durasi, lokasi, perjalanan penyakit, riwayat pengobatan sebelumnya, dan akibat gangguan yang timbul) oleh dr. Rahma, dr. Alan, dr Ari Erna, dan dr. Dea yang berpraktik di Klinik Unisia Polifarma UII. Contoh teks keluhan dapat dilihat pada Gambar 4.1 dan Gambar 4.2. Gambar 4.1 merupakan teks keluhan pasien yang ditulis manual oleh peneliti berdasarkan hasil rekaman pasien yang diambil di Klinik Unisia Polifarma UII dan Gambar 4.2 merupakan teks keluhan pasien yang diambil dari *website* alodokter.com.

Radang tenggorokan dan pilek aja, sejak tadi malam. Cuma radangnya hilang timbul gitu sih dok. Belum berobat dok, buat minum sama makan susah menelan dok, apalagi kalo kering banget.

Gambar 4.1 Data Hasil Rekaman Pasien



Gambar 4.2 Data Alodokter

4.2 *Preprocessing*

Pada tahap ini dilakukan ekstraksi informasi, di mana kata yang tidak penting akan dihapus sehingga didapatkan hasil akhir yang diperlukan dalam klasifikasi pada keluhan pasien. Pada *preprocessing* ini dilakukan 11 urutan tahapan proses, yaitu:

a. *Case Folding*

Proses pada tahap ini dilakukan untuk mengubah teks menjadi huruf kecil semua agar dapat memudahkan proses tahapan berikutnya. Adapun kode program pada implementasi proses ini dapat dilihat pada Gambar 4.3.

```
33  # Format string as lowercase
34  def sentenceCaseString(string):
35      newString = string.lower()
36      newString = re.sub('([.,])', r' \1 ', newString)
37      return newString
38
```

Gambar 4.3 Kode Program *Case Folding*

Pada *case folding* ini menggunakan fungsi *lower()* yang akan mengubah teks menjadi huruf kecil dan *regular expression* atau re.sub(‘([.,])’, r’ \1 ‘, newString) yang digunakan untuk memberi spasi (‘ \1 ‘) setelah tanda titik (.) atau koma (,).

b. *Stopword* Tanda baca

Proses pada tahap ini dilakukan untuk menghilangkan tanda baca selain tanda baca titik (.), koma (,), dan hubung (-). Adapun kode program pada implementasi proses ini dapat dilihat pada Gambar 4.4.

```
35
36  def stopTandaBaca(string):
37      str = re.sub(r'([?!@#%$])', ' ', string)
38      return str
39
```

Gambar 4.4 Kode Program *Stopword* Tanda Baca

Pada tahap ini menggunakan fungsi *regular expression* (re.sub) untuk menghilangkan tanda baca seperti ‘?’, ‘!’, ‘@’, ‘#’, ‘%’, dan ‘$’, meskipun tanda baca tersebut terdiri dari satu atau lebih

c. Konversi Angka Menjadi Huruf

Proses pada tahap ini dilakukan untuk mengubah angka menjadi huruf. Tahap ini dilakukan karena semua angka tidak dimasukkan ke dalam kamus. Kode program pada konversi angka menjadi huruf ini dapat dilihat pada Gambar 4.5 dan Gambar 4.6.

```
133
134     def int_to_en(num):
135         d = { 0 : 'nol', 1 : 'satu', 2 : 'dua', 3 : 'tiga', 4 : 'empat', 5 : 'lima', 6 : 'enam', 7 : 'tujuh',
136               8 : 'delapan', 9 : 'sembilan', 10 : 'sepuluh', 11 : 'sebelas', 12 : 'dua belas', 13 : 'tiga belas',
137               14 : 'empat belas', 15 : 'lima belas', 16 : 'enam belas', 17 : 'tujuh belas', 18 : 'delapan belas',
138               19 : 'sembilan belas', 20 : 'dua puluh', 30 : 'tiga puluh', 40 : 'empat puluh', 50 : 'lima puluh',
139               60 : 'enam puluh', 70 : 'tujuh puluh', 80 : 'delapan puluh', 90 : 'sembilan puluh' }
140         k = 1000
141         m = k * 1000
142         b = m * 1000
143
144         assert(0 <= num)
145         if (num < 20):
146             return d[num]
147         if (num < 100):
148             if num % 10 == 0: return d[num]
149             else: return d[num // 10 * 10] + ' ' + d[num % 10]
150         if (num < k):...
153         raise AssertionError('num is too large: %s' % str(num))
154
```

Gambar 4.5 Kode Program Mengubah Angka

Pada Gambar 4.5 terdapat fungsi *int_to_en()*. Fungsi *int_to_en()* merupakan kode untuk mengubah angka menjadi huruf.

```
154
155     def convertNumber(sent):
156         data = []
157         for s in sent.split():
158             if s.isdigit():
159                 data.append(int_to_en(int(s)))
160             else:
161                 data.append(s)
162         return data
163
```

Gambar 4.6 Kode Program Konversi Angka Menjadi Huruf

Pada Gambar 4.6 terdapat fungsi *convertNumber()* yang merupakan kode untuk memanggil dari fungsi *int_to_en()* yang terdapat pada Gambar 4.5.

d. Normalisasi Singkatan & Padanan Kata

Proses pada tahap ini dilakukan untuk menormalisasikan kata dari singkatan dan padanan kata. Adapun kode program pada implementasi proses ini dapat dilihat pada Gambar 4.7.

```
38
39  def normalize(sent):
40      slang_word_raw = open('slang_word.txt').readlines
41      with open("slang_word.txt") as f:
42          slang_word_raw = f.readlines()
43          slang_word_raw = dict([i.split(",") for i in slang_word_raw])
44
45      slang_word_dict = []
46      for item in sent.split():
47          if item in slang_word_raw.keys():
48              slang_word_dict.append(slang_word_raw[item])
49          else:
50              slang_word_dict.append(item)
51      return " ".join(slang_word_dict)
52
```

Gambar 4.7 Kode Program Normalisasi

Pada normalisasi ini menggunakan kamus atau daftar kata yang terdapat pada (*'slang_word.txt'*), di mana dilakukan proses pengecekan tiap kata pada teks keluhan pasien dengan kamus. Daftar kata pada (*'slang_word.txt'*) untuk normalisasi dapat dilihat pada Lampiran A.

e. Normalisasi Pengulangan Kata yang Ditulis dengan Angka

Proses pada tahap ini dilakukan normalisasi pada kata singkatan yang ditulis dengan angka. Sebagai contoh "mual2" yang harusnya ditulis dengan "mual-mual". Adapun kode program pada implementasi proses ini dapat dilihat pada Gambar 4.8.

```
52
53  def singkatan(string):
54      token = []
55      for i in string.split():
56          kataBelakang = i[-1]
57          if kataBelakang == "2":
58              kataDepan = i[:-1]
59              token.append(kataDepan +"-"+ kataDepan)
60          else:
61              token.append(i)
62      benar = ' '.join(token)
63      return benar
64
```

Gambar 4.8 Kode Program untuk Pengulangan Kata yang Ditulis dengan Angka

Pada pengulangan kata ini digunakan aturan pada percabangan *if else*. Jika kata belakang pada sebuah kata terdapat angka "2" maka angka "2" tersebut dihapus lalu diganti dengan tanda hubung "-" dan kata tersebut diulang lagi.

f. Pemecahan Kalimat & Frasa/Klausa

Proses pada tahap ini dilakukan untuk memisahkan kalimat berdasarkan tanda baca titik (.) dan memisahkan kata berdasarkan tanda baca koma (,). Adapun kode program pada implementasi proses ini dapat dilihat pada Gambar 4.9.

```
64
65  def pemecahKalimat(string):
66      newKalimat = []
67      newKata = []
68      newKalimat = string.split(".")
69      for i in range(len(newKalimat)):
70          newKata = newKalimat[i].split(",")
71          newKalimat[i] = newKata
72      for i in range(len(newKalimat)):
73          newKalimat[i] = [kata.strip() for kata in newKalimat[i]]
74      return newKalimat
75
```

Gambar 4.9 Kode Program Pemecahan Kalimat & Frasa/Klausa

Pada proses pemecahan kalimat dan frasa/klausa ini menggunakan fungsi *range()* yang digunakan sebagai *counter* pada perulangan *for*, di mana *counter* ini merupakan variabel yang akan menyimpan nilai dari perulangan. Setiap kalimat dipisahkan berdasarkan tanda baca titik (.) dan disimpan pada fungsi *range()* agar kalimat dapat dilakukan pemisahan lagi berdasarkan tanda baca koma (,) yang disebut dengan frasa/klausa.

g. *Spell Correction*

Proses pada tahap ini dilakukan untuk memeriksa kesalahan pada penulisan kata. Adapun kode program pada implementasi proses ini dapat dilihat pada Gambar 4.10 dan Gambar 4.11. Gambar 4.10 merupakan kode program untuk membuka dan membaca file dari "typo.txt", sedangkan Gabar 4.11 merupakan kode program untuk *spell checking*.

```
5
6   WORDS = Counter(words(open('typo.txt').read()))
7
```

Gambar 4.10 Kode Program untuk Membaca *File*

```
75
76  def correctionWord(list_):
77      token2 = []
78      token3 = []
79      t = []
80      for string in list_:
81          for word in string:
82              token = []
83              for i in word.split():
84                  if i in word:
85                      hasil = sp.correction(i)
86                      token.append(hasil)
87                  else:
88                      pass
89              token2.append(" ".join(token))
90          token3.append(token2)
91          token2 = []
92      return token3
93
```

Gambar 4. 11 Kode Program untuk *Spell Checking*

Pada *spell correction* ini dilakukan pengecekan dengan kamus yang bernama *"typo.txt"*, dan menggunakan fungsi dari *spell checker* yaitu *correction*. Secara otomatis, kata per kata akan dicek berdasarkan perhitungan pada persamaan kata dari fungsi *correction*.

h. *Stopword Removal*

Stopword Removal merupakan proses untuk menghilangkan kata-kata yang tidak diperlukan dalam teks keluhan pasien. Kode program pada *stopword removal* dapat dilihat pada Gambar 4.12 dan Gambar 4.13.

```
12
13  # read stopword.txt
14  def readText(file):
15      with open(file) as f:
16          readText = f.read().splitlines()
17      return readText
18  readStopword = readText('stopword.txt')
19
```

Gambar 4.12 Read *File Stopword*

Gambar 4.12 merupakan kode untuk membaca dari *file* '*stopword.txt*' yang berisi kamus daftar kata yang akan dihapus. Fungsi *f.read()* merupakan kode untuk membaca *file*.

```
96  def stopword(list_):
97      token2 = []
98      token3 = []
99      token = []
100     token4 = []
101     for string in list_:
102         for word in string:
103             stop = []
104             word = re.sub(r'(umur)[\s\S]*(tahun|belas|puluh|satu|dua|tiga|empat|lima|enam|tujuh|delapan|sembilan|sepuluh)
105             word = re.sub("selamat malam|selamat pagi|selamat sore|selamat siang|pagi dok|siang dok|sore dok|malam dok|de
106             for i in word.split():
107                 token.append(i)
108                 if i in readStopword:
109                     if "saya" in i and len(word.split()) <=2:
110                         token4.append(i)
111                         if i in readKlas:
112                             pass
113                         else:
114                             token4.remove(i)
115                 else:
116                     if "nama" in i:
117                         break;
118                     stop.append(i)
119             token2.append(" ".join(stop))
120         token3.append(token2)
121         token2 = []
122     return token3
```

Gambar 4.13 Kode Program *Stopword Removal*

Gambar 4.13 merupakan kode untuk menghapus kata-kata yang tidak diperlukan dalam teks keluhan pasien. Pada baris ke-104, sebagai contoh apabila ada kata yang mengandung kata "umur" dan diikuti kata "tahun" atau "belas" maka akan dihapus. Pada baris ke-105, sebagai contoh apabila pada kalimat mengandung kata "selamat malam" atau "malam dok" maka akan dihapus. Jika kata pada kalimat tersebut mengandung kata yang ada pada *file* "*stopword.txt*" maka kata tersebut akan dihapus. Daftar kata pada (*'stopword.txt'*) untuk *stopword removal* dapat dilihat pada Lampiran B.

i. Tokenisasi

Pada tahap ini dilakukan pemisahan antar kata menjadi token. Kode program pada tokenisasi ini dapat dilihat pada Gambar 4.14. Pada kode program tersebut diakukan pemisahan antar kata yang telah dipisah menggunakan fungsi *split()*.

```
109
110     def tokenisasi(list_):
111         token2 = []
112         for string in list_:
113             for word in string:
114                 token = []
115                 for i in word.split():
116                     token.append(i)
117                 token2.append(token)
118         return token2
119
```

Gambar 4.14 Kode Program Tokenisasi

j. *N-Gram*

Pada tahap ini dilakukan pengelompokan kata berdasarkan *tri-gram*, *bi-gram*, dan *uni-gram*. Kode program pada *n-gram* ini dapat dilihat pada Gambar 4.15 dan Gambar 4.16.

```
204
205     with open("Unigram.txt")as uni:
206         unigram = uni.read().split()
207     with open("Bigram.txt")as bi:
208         bigram = bi.read().split("\n")
209     with open("Trigram.txt")as tri:
210         trigram = tri.read().split("\n")
211
```

Gambar 4.15 Kode Program untuk Membaca *File N-Gram*

Pada Gambar 4.15 merupakan kode yang digunakan untuk membuka dan membaca dari *file* “unigram.txt”, “bigram.txt”, dan “trigram.txt”.

```
215
216 def generate_unigram(words_list, n=1):
217     ngrams_list = []
218     for num in range(0, len(words_list)):
219         ngram = ' '.join(words_list[num:num + n])
220         ngrams_list.append(ngram)
221     return ngrams_list
222
223 def generate_bigram(words_list, n=2):
224     ngrams_list = []
225     for num in range(0, len(words_list)):
226         if num != len(words_list)-1:
227             ngram = ' '.join(words_list[num:num + n])
228             ngrams_list.append(ngram)
229     return ngrams_list
230
231 def generate_trigram(words_list, n=3):
232     ngrams_list = []
233     for num in range(0, len(words_list)):
234         if num != len(words_list) - 2 and num != len(words_list)-1:
235             ngram = ' '.join(words_list[num:num + n])
236             ngrams_list.append(ngram)
237     return ngrams_list
238
```

Gambar 4.16 Kode Program *N-Gram*

Pada Gambar 4.16 merupakan kode fungsi untuk mengelompokkan kata yang dicocokkan pada *file* "unigram.txt", "bigram.txt", dan "trigram.txt". Adapun pengurutannya yaitu dimulai dari *tri-gram* terlebih dahulu, lalu *bi-gram*, dan *uni-gram*. Jika terdapat token yang memiliki kecocokan pada *uni-gram*, *bi-gram*, dan *tri-gram*, maka token yang diambil untuk hasil akhirnya adalah token yang berada di *tri-gram*. Begitu juga jika terdapat token yang memiliki kecocokan pada *uni-gram* dan *bi-gram*, maka token yang diambil untuk hasil akhirnya adalah token yang berada di *bi-gram*. Adapun daftar kata *tri-gram*, *bi-gram*, dan *uni-gram* dapat dilihat pada Lampiran C, D, dan E.

k. Pengindeksan *Array*

Proses pada tahap ini dilakukan pengindeksan pada *array* untuk menentukan lokasi token/frasa/klausa. Kode program pada pengindeksan *array* dapat dilihat pada Gambar 4.17.

```
231
232     def cekLocationString(sentDump, list_):
233         pos_ = 0
234         kalimat = []
235         for i in sentDump:
236             a = ' '.join(i)
237             for j in list_:
238                 if re.search(j, a):
239                     kalimat.append([j, pos_])
240             pos_ = pos_ + 1
241         return kalimat
242
```

Gambar 4.17 Kode Program Pengindeksan *Array*

Pada Gambar 4.17 terdapat fungsi *cekLocationString()* di mana fungsi ini digunakan untuk memberikan indeks pada *array* agar dapat mengetahui lokasi token/frasa/klausa terdapat dalam kalimat ke berapa.

4.3 *Feature Extraction*

Setelah selesai tahap *preprocessing* lalu dilanjutkan dengan proses *feature extraction* dengan metode *tf-idf* yang akan memberikan bobot setiap token agar token dapat diklasifikasikan dengan SVM. Kode program pada *tf-idf* dapat dilihat pada Gambar 4.18.

```
count_vect = CountVectorizer()
X_train_counts = count_vect.fit_transform(source)
X_test_counts = count_vect.transform(dump)

tfidf_transformer = TfidfTransformer()
X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)
X_test_tfidf = tfidf_transformer.transform(X_test_counts)
```

Gambar 4.18 Kode Program untuk TF-IDF

Pada Gambar 4.18 terdapat CountVectorizer() yang digunakan untuk menghitung jumlah kata (*term*) dari data X_train dan X_test, di mana X_train merupakan data dari "source" dan X_test dari "dump". Lalu terdapat TfidfTransformer() yang digunakan untuk menghitung nilai idf dengan memanggil tfidf_transformer.transform() dari hasil perhitungan tf.

4.4 Klasifikasi SVM

Setelah selesai *tf-idf* lalu dilanjutkan dengan proses klasifikasi menggunakan metode pembelajaran SVM. Metode SVM merupakan metode pembelajaran mesin di mana digunakan pelabelan manual untuk menentukan kelas pada klasifikasi. Pelabelan ini disimpan pada *file* "label.txt". Contohnya adalah "dua hari,onset", yang artinya bahwa dua hari termasuk dalam kelas "onset". Untuk keluhan utama dan keluhan lain disimpan dalam kamus "klasifikasi.txt".

```
218
219     def readKlasifikasi(file):
220         with open(file) as f:
221             readKlasifikasi = f.read().splitlines()
222         return readKlasifikasi
223
224     def readLabel(file):
225         result = []
226         with open(file) as f:
227             read_ = f.read().splitlines()
228         for i in read_:
229             a = i.split(",")
230             result.append(a[0])
231         return result
232
```

Gambar 4.19 Kode Program untuk Membaca Klasifikasi dan Label

Pada Gambar 4.19 terdapat fungsi *readKlasifikasi()* dan *readLabel*(). Fungsi *readKlasifikasi()* digunakan untuk membuka dan membaca *file* dari "klasifikasi.txt". Sedangkan fungsi *readLabel()* digunakan untuk membuka dan membaca *file* dari "label.txt".

```
253
254     def klasifikasi_(list_):
255         keluhanUtama = []
256         keluhanLain = []
257         keluhanLainLain = []
258         i = list_[0][1]
259         print("list", list_)
260         for string in list_:
261             if string[0] in readKlas:
262                 if i == string[1]:
263                     keluhanUtama.append(string)
264                     i = i + 1
265                 else:
266                     keluhanLain.append(string)
267             else:
268                 keluhanLainLain.append(string)
269         return keluhanUtama, keluhanLain, keluhanLainLain
270
```

Gambar 4.20 Kode Program Klasifikasi

Gambar 4.20 merupakan fungsi *klasifikasi()* yang digunakan untuk membedakan keluhan dengan yang lain. Jika token terdapat dalam kamus "klasifikasi.txt" maka akan masuk ke dalam keluhan, jika tidak maka akan masuk ke dalam keluhan lain-lain. Keluhan lain-lain merupakan kelas klasifikasi selain dari keluhan utama dan keluhan lain, yaitu onset, keterangan, durasi, lokasi, sifat serangan, frekuensi serangan, perjalanan penyakit, riwayat pengobatan dan akibat gangguan yang timbul. Dari keluhan tersebut, kata pertama akan masuk ke keluhan utama, sedangkan kata selanjutnya akan masuk ke keluhan lain.

```
text_clf = SGDClassifier(loss='hinge', penalty='l2', alpha=1e-3, random_state=42, n_iter=5)
text_clf.fit(X_train_tfidf, target)

predicted = text_clf.predict(X_test_tfidf)
print("Prediksi : ", predicted)

for doc, pred in zip(datates, predicted):
    print(doc, pred)
```

Gambar 4.21 Kode Program SVM

Gambar 4.21 merupakan pengklasifikasi *linear* SVM dengan pelatihan SGDClassifier. SGD (*Stochastic Gradient Descent*) mendukung klasifikasi multi-kelas dengan

menggabungkan beberapa pengklasifikasi biner dalam skema "*one versus all*". Text_clf.fit() digunakan untuk melatih model SVM dengan X_train_tfidf. Text_clf_predict() digunakan untuk memprediksi label pada X_test_tfidf.

```
354
355     predictSVM(keluhanLainLain)
356     for i in keluhanUtama:
357         print(i, "keluhan utama")
358     for i in keluhanLain:
359         print(i, "keluhan lain")
360
```

Gambar 4.22 Kode Program Menampilkan Hasil Klasifikasi

Pada Gambar 4.22 dilakukan proses penampilan dari semua hasil klasifikasi dari keluhan selain keluhan utama dan keluhan lain, keluhan utama, serta keluhan lain di mana hasil yang keluar diurutkan berdasarkan indeks.

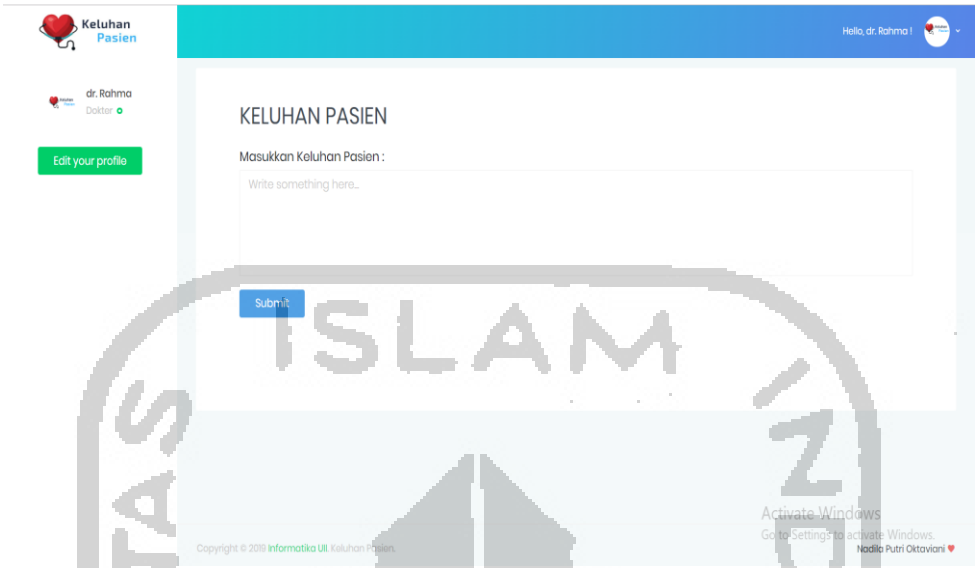
4.5 Tampilan Sistem

Untuk pemanggilan fungsi *python* yang akan ditampilkan pada web menggunakan Flask dan Jinja. Kode program untuk pemanggilan fungsi python dapat dilihat pada Gambar 4.23.

```
126
127 @app.route('/tampil', methods=['POST', 'GET'])
128 def tampil():
129     jumlah_kalimat=1
130     if request.method == 'POST':
131         data = request.form['data']
132         jumlah_kalimat, hasilPreprocessing, isiTabel = KeluhanPasien.tampilPreprocessing(data)
133         return render_template('index.html', hasil=hasilPreprocessing, isiTabel=isiTabel, kalimat=int(jumlah_kalimat), nama=session['dokter'], foto=session
134     return render_template('index.html', kalimat=int(jumlah_kalimat), nama=session['dokter'], foto=session['foto'])
135
```

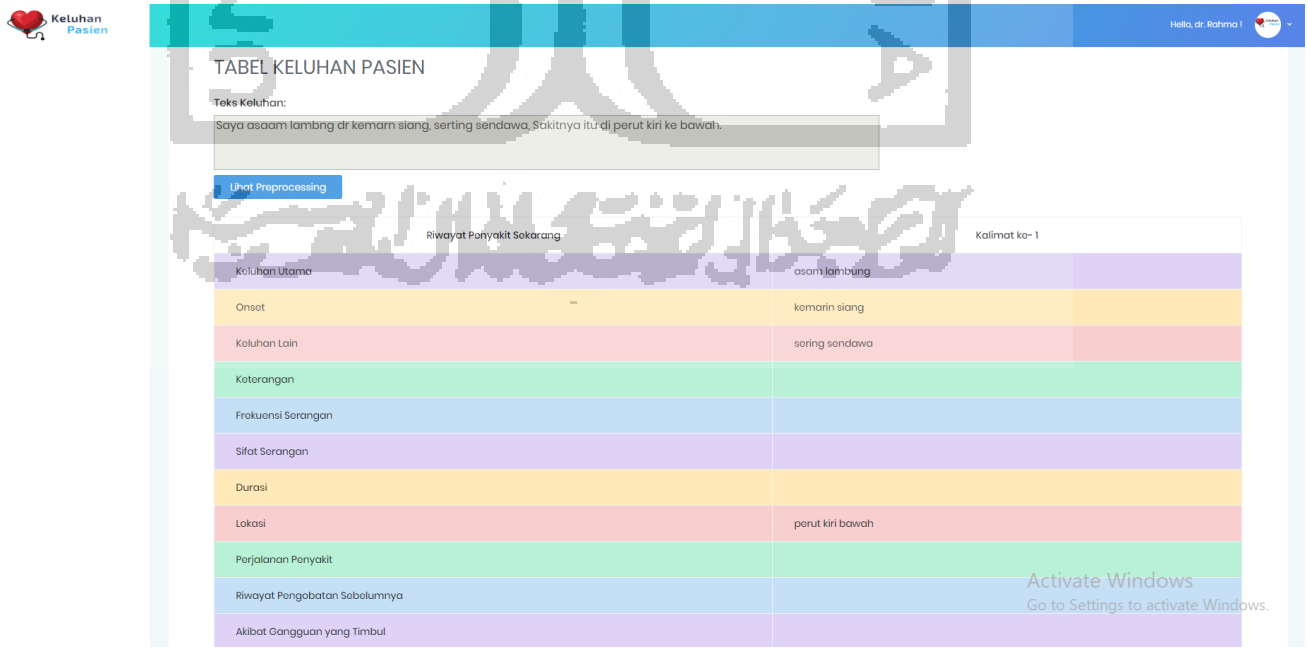
Gambar 4.23 Kode Program Pemanggilan Fungsi Python

Pada Gambar 4.23 terdapat fungsi tampil() yang digunakan untuk menampilkan proses pada Python ke halaman web. Data pada baris ke-131 merupakan teks keluhan pasien yang telah dimasukkan oleh dokter, di mana data tersebut diambil atau langsung masuk ke dalam *preprocessing* pada python. Jumlah_kalimat pada baris ke-132 digunakan untuk melihat kalimat pada teks keluhan pasien terdiri dari berapa kalimat, jumlah_kalimat ini akan digunakan untuk menampilkn berapa kolom pada tabel klasifikasi di web. hasilPreprocessing pada baris ke-132 digunakan untuk menampilkan semua hasil *preprocessing* dari python.



Gambar 4.24 Halaman Tampilan Masukan Keluhan Pasien

Gambar 4.24 merupakan halaman web pada tampilan masukan pada keluhan pasien yang belum di*submit*. *Form* dari masukan keluhan pasien tersebut sebagai contoh dimasukkan keluhan seperti "Saya asaam lambung dr kemarn siang, serting sendawa. Sakitnya itu di perut kiri ke bawah.". Lalu setelah di *submit*, hasilnya dapat dilihat pada Gambar 4.25.



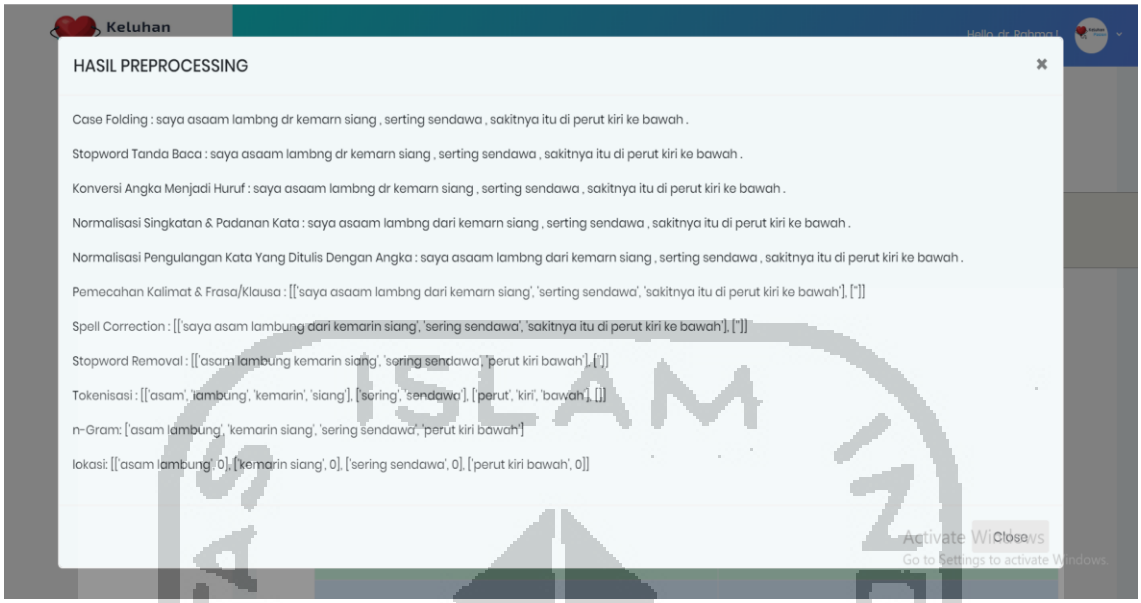
Gambar 4.25 Tampilan Hasil Pemetaan Keluhan Pasien

Gambar 4.25 merupakan halaman web pada keluhan pasien setelah di*submit.* Berdasarkan gambar tersebut terdapat tabel hasil klasifikasi. Adapun kode program untuk menampilkannya dapat dilihat pada Gambar 4.26.

```
<div class="table-responsive">
  <table class="table table-bordered">
    <thead>
      <tr>
        <th width="250px">
          Riwayat Penyakit Sekarang
        </th>
        {% for i in range(kalimat) %}
        <th width="250px">
          Kalimat ke- {{ i+1 }}
        </th>
        {% endfor %}
      </tr>
    </thead>
    <tbody>
      <tr class="table-info">
        <td>
          Keluhan Utama
        </td>
        {% for i in range(kalimat) %}
            <td>
            {% for keluhanUtama in isiTabel[0] %}
              {% if keluhanUtama[1]==i %}
              {{ keluhanUtama[0] }}
              {% endif %}
            {% endfor %}
             </td>
        {% endfor %}
      </tr>
```

Gambar 4.26 Kode Program untuk Menampilkan Tabel Klasifikasi

Gambar 4.26 merupakan kode program pada html untuk menampilkan tabel klasifikasi dengan menggunakan Jinja. Untuk kolom pada "Kalimat ke-" pada tabel ditentukan berdasarkan jumlah dari kalimat yang dimasukkan pada *form* masukan keluhan pasien. Lalu pada halaman tersebut terdapat tombol "Lihat *Preprocessing*". Jika tombol tersebut diklik, maka hasil tampilannya dapat dilihat pada Gambar 4.27.



Gambar 4.27 Tampilan dari Hasil *Preprocessing*

Gambar 4.27 merupakan halaman web pada tampilan hasil *preprocessing* setelah diklik tombol "Lihat *Preprocessing*". Adapun kode program untuk melihat hasil *preprocessing* dapat dilihat pada Gambar 4.28. Hasil preprocessing adalah hasil dari proses pemanggilan fungsi Python. Untuk melihat proses pemanggilan fungsi dari python ke web dapat dilihat pada Gambar 4.29.

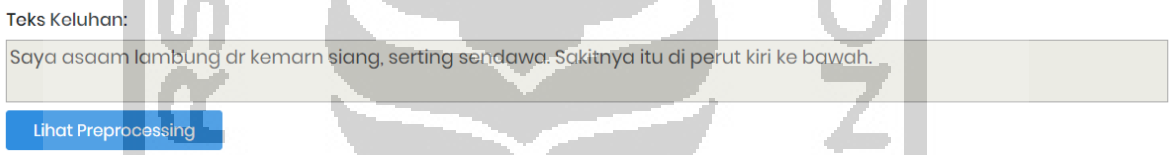
```
<!-- Modal -->
<div class="modal fade" id="ModalHasil" tabindex="-1" role="dialog" aria-labelledby="ModalHasilLabel">
  <div class="modal-dialog modal-dialog-centered modal-lg">
    <div class="modal-content">
      <div class="modal-header">
        <h4 class="modal-title" id="ModalHasilLabel">HASIL PREPROCESSING</h4>
        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
          <span aria-hidden="true">×</span>
        </button>
      </div>
      <div class="modal-body">
        {% for i in hasil%}
        <p> {{ i }}</p>
        {% endfor%}
      </div>

      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
      </div>
    </div>
  </div>
</div>
```

Gambar 4.28 Kode Program untuk Melihat *Hasil Preprocessing*

```
126
127 @app.route('/tampil', methods=['POST', 'GET'])
128 def tampil():
129     jumlah_kalimat=1
130     if request.method == 'POST':
131         data = request.form['data']
132         print(data)
133         jumlah_kalimat, hasilPreprocessing, isiTabel = KeluhanPasien.tampilPreprocessing(data)
134         return render_template('index.html',data=data, hasil=hasilPreprocessing, isiTabel=isiTabel, kalimat=int(jumlah_kalimat),
135     return render_template('index.html', data=data,kalimat=int(jumlah_kalimat), nama=session['dokter'], foto=session['foto'])
136
```

Gambar 4.29 Kode Program Pemanggilan Fungsi Python

Gambar 4.29 merupakan kode program utama untuk menampilkan semua proses. Proses ini dimulai dari masukan keluhan pasien pada web yang disimpan pada variabel "data". Variabel "data" ini dilempar ke fungsi Python agar dapat melakukan *preprocessing* dan menampilkan tabel klasifikasi pada keluhan pasien. Tampilan data keluhan pasien atau dalam hal ini yang ditampung dalam variabel "data" dapat dilihat pada Gambar 4.30.



Gambar 4.30 Tampilan Teks Keluhan

4.6 Pengujian

Pada tahap ini dilakukan pengujian sebanyak 20 data, di mana 10 data berasal dari dokter dan 10 data dari alodokter.com yang telah dipetakan oleh dokter. Akan tetapi dari data itu, hasil pemetaan dalam komponen keluhan pasien yang dilakukan oleh dokter sebanyak 6 data telah diubah ke bahasa medis. Sedangkan, tujuan dari penelitian ini adalah untuk mendokumentasikan keluhan pasien dari apa yang diucapkan oleh pasien (Aspects, 1996). Data dari dokter yang telah dipetakan dengan tetap menggunakan bahasa keluhan pasien sebanyak 14 data, sehingga data keluhan pasien ini digunakan dalam pengujian. Adapun data pemetaan keluhan pasien yang tidak digunakan dalam pengujian dapat dilihat pada Lampiran F.

Hasil pengujian ini dihitung berdasarkan:

$$\text{Tingkat Prediksi} = \frac{\text{Jumlah Prediksi antara Klasifikasi yang Dilakukan oleh Dokter dan Sistem}}{\text{Jumlah dari Klasifikasi Dokter}} \quad (1)$$

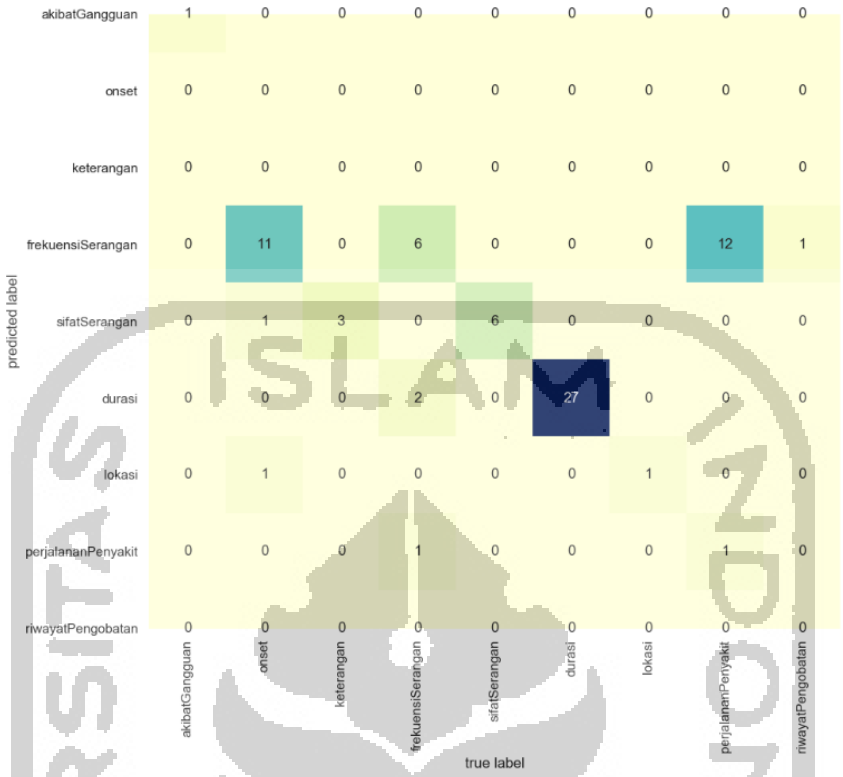
Hasil Akhir = Penjumlahan Tingkat Prediksi dari Semua Data Pengujian x 100% (2)

Banyaknya Data Pengujian

No	D/S	Kalimat	Kalimat ke-1											Hasil
			Keluhan Utama	Onset	Keluhan Lain	Ket	FS	SS	D	L	PP	RP	AG	
1	D	Batuk berdahak 3 hari disertai pilek, nyeri tenggorokan, belum diobati.	batuk berdahak	tiga hari	pilek, nyeri tenggorokan							belum diobati		4/5 = 0,8
	S	Batuk berdahak 3 hari disertai pilek, nyeri tenggorokan, belum diobati.	batuk berdahak	tiga hari	pilek nyeri tenggorokan	belum obati								

Gambar 4.31 Tabel untuk Melihat Kecocokan Pada Data

Gambar 4.30 merupakan contoh untuk melihat kecocokan pada klasifikasi. Dari contoh tersebut, terdapat 4 token yang sama dari 5 token yang dipetakan oleh dokter dengan perhitungan 4 /5 menjadi 0,8. Hasil tersebut dijumlahkan dengan hasil data lainnya, lalu dibagi dengan banyaknya data (14 data) dan dikali dengan 100%. Dari perhitungan tersebut, didapat hasil akhir sebesar 46% karena data tidak ada di kamus. Akan tetapi, setelah data ditambahkan pada kamus didapat hasil akhir sebesar 86%. Adapun tabel data dari hasil pengujian ini dapat dilihat pada lampiran G. Sedangkan hasil data pengujian setelah data dimasukkan ke dalam kamus dapat dilihat pada lampiran H.



Gambar 4.32 *Confusion Matrix*

Gambar 4.32 merupakan hasil dari pengujian menggunakan *confusion matrix*. Dari *matrix* tersebut dapat dihitung nilai *precision, recall, f1-score,* dan *support*. Hasil dari *precision, recall, f1-score,* dan *support* dapat dilihat pada Gambar 4.33.

accuracy : 56.7567568%

	precision	recall	f1-score	support
frekuensi	1.00	1.00	1.00	1
keluhanLain	0.00	0.00	0.00	13
keluhanUtama	0.00	0.00	0.00	3
keterangan	0.20	0.67	0.31	9
lokasi	0.60	1.00	0.75	6
onset	0.93	1.00	0.96	27
perjalananPenyakit	0.50	1.00	0.67	1
riwayatPengobatan	0.50	0.08	0.13	13
sifat	0.00	0.00	0.00	1
micro avg	0.57	0.57	0.57	74
macro avg	0.41	0.53	0.42	74
weighted avg	0.52	0.57	0.50	74

Gambar 4.33 *Precision, Recall, F1-score, Support*

Hasil *accuracy* dari pengujian ini sebesar 56,75%. Sedangkan *precision*, *recall*, dan *f1-score* diambil dari nilai *macro average*. Hasil dari *precision* sebesar 41%, *recall* sebesar 53%, dan *f1-score* sebesar 42%.