BAB V
HASIL DAN PEMBAHASAN

Bab ini menyajikan hasil pengembangan sistem yang telah dilakukan peneliti dan membahas pengembangan sistem dalam dua sub bab, yaitu hasil dan pembahasan.

5.1 Hasil

Bagian ini menyajikan hasil penelitian yang telah dicapai, meliputi perancangan model, proses klasifikasi, serta implementasi sistem yang telah dikembangkan.

5.1.1 Pra-proses Data dan Pembuatan Model

a) *Case Folding*

```
def case_folding(value):
  result = value.lower().strip()
  return result
```

Gambar 5. 1 Kode Penggalan *Case Folding*

Gambar 5.1 menunjukkan penggalan kode untuk tahap pertama pra-pemrosesan yaitu *case folding*. Nama fungsi yang didefinisikan adalah *case_folding* dan memiliki satu parameter dengan nama *value* yang digunakan untuk menampung argumen atau sinopsis yang akan diterapkan *case folding*. Argumen tersebut selanjutnya dimasukkan ke dalam variabel dengan nama *result*. *Output* yang dihasilkan dari fungsi ini adalah sinopsis yang telah diubah seluruh huruf kapitalnya menjadi huruf kecil sekaligus menghilangkan spasi pada awal dan akhir kalimat.

Tabel 5. 1 Hasil *Case Folding*

Input (Sebelum)	**Output (Sesudah)**
A gang of unemployed itinerant musicians play in the south of Stockholm. Then they get the chance to be an orchestra in a dance restaurant. It goes well until the female owner falls in love with one of them.	a gang of unemployed itinerant musicians play in the south of stockholm. then they get the chance to be an orchestra in a dance restaurant. it goes well until the female owner falls in love with one of them.

Tabel 5.1 menunjukkan perbandingan sinopsis sebelum dan sesudah penerapan *case folding*. Perbedaan yang jelas terlihat adalah seluruh kalimat berubah dari huruf besar menjadi huruf kecil.

b) *Cleaning*

```
def cleaning_synopsis(value):
  result = value.lower().strip()
  result = ' '.join(result.split())
  result = re.sub('[^a-zA-Z]', ' ', result)
  result = re.sub('\s[\s]+', ' ', result).strip()
  result = re.sub(r'@\S+', '', result)
  result = re.sub(r'http\S+', '', result)
  result = re.sub(r'pic.\S+', '', result)
  result = result.translate(str.maketrans('', '', string.punctuation))
  return result
```

Gambar 5. 2 Kode Penggalan *Cleaning*

Gambar 5.2 memperlihatkan penggalan kode untuk tahap kedua pra-pemrosesan yaitu *cleaning*. Nama fungsi yang didefinisikan adalah *cleaning_synopsis* dan memiliki satu parameter yang digunakan untuk menampung hasil pada tahap sebelumnya. Fungsi tersebut memiliki beberapa tahapan seperti memastikan bahwa setiap sinopsis hanya dipisahkan oleh satu spasi serta menghilangkan *hyperlinks* atau *url*, *hashtag*, angka, simbol retweet, emoji, dan tanda baca dengan menggunakan regex. Tahap ini memastikan bahwa data yang akan dikirimkan pada tahap selanjutnya hanya memiliki unsur elemen yang dibutuhkan saja.

Tabel 5. 2 Hasil *Cleaning*

Input (Sebelum)	**Output (Sesudah)**
a gang of unemployed itinerant musicians play in the south of stockholm. then they get the chance to be an orchestra in a dance restaurant. it goes well until the female owner falls in love with one of them.	a gang of unemployed itinerant musicians play in the south of stockholm then they get the chance to be an orchestra in a dance restaurant it goes well until the female owner falls in love with one of them

Tabel 5.2 menunjukkan perbandingan sinopsis sebelum dan sesudah penerapan *cleaning*. Perbedaan yang dapat dilihat adalah hilangnya tanda baca pada sinopsis.

c) Tokenisasi

```
def tokenize_synopsis(value):
  result = word_tokenize(value)
  return result
```

Gambar 5. 3 Kode Penggalan Tokenisasi

Gambar 5.3 memperlihatkan penggalan kode tahap ketiga pra-pemrosesan, yaitu tokenisasi. Nama fungsi pada tahap ini adalah *tokenize_synopsis* yang memiliki satu parameter untuk menampung hasil tahap sebelumnya. Tokenisasi diterapkan dengan memanfaatkan salah satu fungsi pada Pustaka NLTK yaitu *word_tokenize*. Fungsi tersebut akan memecah sinopsis menjadi kata-kata individual dan mengubahnya dari tipe data *string* menjadi *list*.

Tabel 5. 3 Hasil Tokenisasi

Input (Sebelum)	**Output (Sesudah)**
a gang of unemployed itinerant musicians play in the south of stockholm then they get the chance to be an orchestra in a dance restaurant it goes well until the female owner falls in love with one of them	['a', 'gang', 'of', 'unemployed', 'itinerant', 'musicians', 'play', 'in', 'the', 'south', 'of', 'stockholm', 'then', 'they', 'get', 'the', 'chance', 'to', 'be', 'an', 'orchestra', 'in', 'a', 'dance', 'restaurant', 'it', 'goes', 'well', 'until', 'the', 'female', 'owner', 'falls', 'in', 'love', 'with', 'one', 'of', 'them']

Tabel 5.3 menunjukkan perbandingan sinopsis sebelum dan sesudah penerapan tokenisasi. Kalimat yang berupa rangkaian kata dengan tipe data *string* diubah menjadi *list* atau daftar dengan jumlah token sesuai dengan jumlah kata yang ada pada sinopsis.

d) *Stopword Removal*

```
def stopwords_filtering(value):
  stopwords_list = set(stopwords.words('english'))
  filtered_words = []
  for synopsis in value:
    if synopsis not in stopwords_list:
      filtered_words.append(synopsis)
  return filtered_words
```

Gambar 5. 4 Kode Penggalan *Stopword Removal*

Gambar 5.4 memperlihatkan penggalan kode tahap keempat pra-pemrosesan yang diketahui sebagai *stopword removal*. Nama fungsi pada tahap ini adalah

filtering_stopwords dan memiliki satu parameter untuk menampung argumen atau hasil pada tahap sebelumnya. Daftar kata *stopword* bahasa inggris yang digunakan untuk menyaring sinopsis berasal dari *library* NLTK. Fungsi tersebut akan melakukan pengecekan terhadap sinopsis, apakah terdapat kata yang termasuk ke dalam daftar kata *stopword*. Jika tidak terdaftar, kata tersebut dimasukkan ke dalam *list* kata yang akan menjadi luaran.

Tabel 5. 4 Hasil *Stopword Removal*

Input (Sebelum)	**Output (Sesudah)**
['a', 'gang', 'of', 'unemployed', 'itinerant', 'musicians', 'play', 'in', 'the', 'south', 'of', 'stockholm', 'then', 'they', 'get', 'the', 'chance', 'to', 'be', 'an', 'orchestra', 'in', 'a', 'dance', 'restaurant', 'it', 'goes', 'well', 'until', 'the', 'female', 'owner', 'falls', 'in', 'love', 'with', 'one', 'of', 'them']	['gang', 'unemployed', 'itinerant', 'musicians', 'play', 'south', 'stockholm', 'get', 'chance', 'orchestra', 'dance', 'restaurant', 'goes', 'well', 'female', 'owner', 'falls', 'love', 'one']

Tabel 5.4 menunjukkan perbandingan sinopsis sebelum dan sesudah penerapan fungsi *stopword_removal*. Kata yang termasuk dalam daftar kata *stopword* akan dihilangkan agar mengurangi dimensi data. Adapun contoh kata yang termasuk ke dalam *stopword* yaitu *in, to, and, or, this*, dan kata lainnya yang tidak relevan.

e) *Stemming*

```
def stemming_synopsis(value):
  ps = PorterStemmer()
  return [ps.stem(word) for word in value]
```

Gambar 5. 5 Kode Penggalan *Stemming*

Gambar 5.5 menunjukkan penggalan kode tahap pra-pemrosesan ke-5 yaitu *stemming*. Nama fungsi pada tahap ini adalah *stemming_synopsis* dan memiliki satu parameter untuk menampung argumen atau hasil pada tahap sebelumnya. Pada tahap ini, setiap kata dalam daftar tersebut akan diubah menjadi bentuk dasarnya (*stem*) dengan menggunakan salah satu fungsi pada *library* NLTK yaitu *PorterStemmer*.

Tabel 5. 5 Hasil *Stemming*

Input (Sebelum)	**Output (Sesudah)**
['gang', 'unemployed', 'itinerant', 'musicians', 'play', 'south', 'stockholm', 'get', 'chance', 'orchestra', 'dance', 'restaurant', 'goes', 'well', 'female', 'owner', 'falls', 'love', 'one']	['gang', 'unemploy', 'itiner', 'musician', 'play', 'south', 'stockholm', 'get', 'chanc', 'orchestra', 'danc', 'restaur', 'goe', 'well', 'femal', 'owner', 'fall', 'love', 'one']

Tabel 5.5 menunjukkan perbandingan hasil sinopsis yang telah melalui tahap pra-pemrosesan akhir yaitu *stemming*. Kumpulan kata yang ada pada *list* ditransformasi menjadi kata dasar atau *root* dengan menghilangkan seluruh imbuhan seperti "-ed", "-ing", "-s" dan sebagainya. Perbedaan dapat dilihat dari perubahan kata *unemployed* menjadi *unemploy*, *musicians* menjadi *musician*, *falls* menjadi *fall*.

f) Pembuatan Model

```
model_bilstm=Sequential()
model_bilstm.add(Embedding(input_dim=max_words + 1, output_dim=32))
model_bilstm.add(Bidirectional(LSTM(units=32, return_sequences=True)))
model_bilstm.add(GlobalMaxPooling1D())
model_bilstm.add(Dense(units=10, activation='softmax'))
model_bilstm.compile(loss='categorical_crossentropy', optimizer="adam", metrics=['accuracy'])
print(model_bilstm.summary())
```

Gambar 5. 6 Kode Penggalan Pendefinisian Model

Gambar 5.6 menunjukkan penggalan kode untuk mendefinisikan model yang menggunakan arsitektur BiLSTM untuk mengklasifikasi genre film. Model tersebut dari lapisan *embedding* yang berperan untuk mengubah *input* teks menjadi representasi vektor numerik dan menyesuaikan ukuran dimensinya untuk dimasukkan ke *layer* BiLSTM. Pada penelitian ini, sebanyak 32 unit *layer* BiLSTM digunakan pada pendefinisian model. *Layer* GlobalMaxPooling1D berfungsi untuk mengambil nilai maksimum sepanjang dimensi dengan tujuan untuk mereduksi dimensi *input* data. *Layer dense* pada tahap ini menggunakan sebanyak 10 unit dan akan menjadi *output* pengolahan jaringan dengan menggunakan fungsi aktivasi yang disebut dengan *softmax*.

5.2 Pembahasan

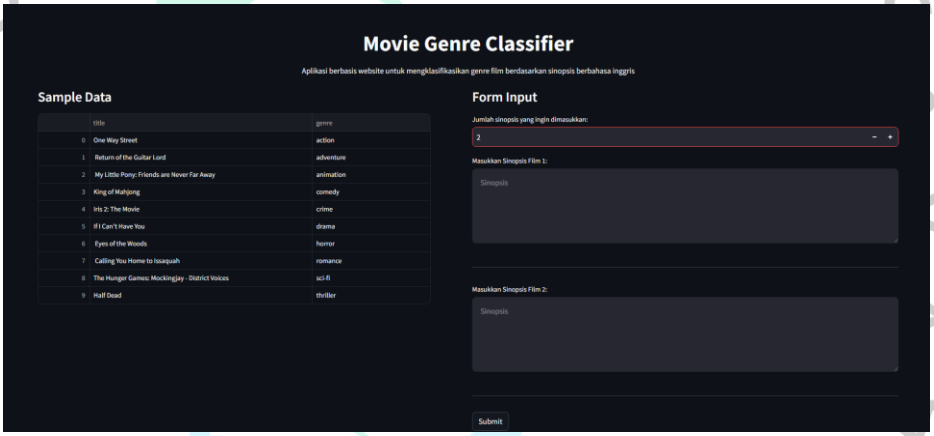
Sub bab ini akan focus membahas evaluasi terhadap aplikasi yang telah dibuat. Pengujian akan dilakukan menggunakan dua metode yang sudah

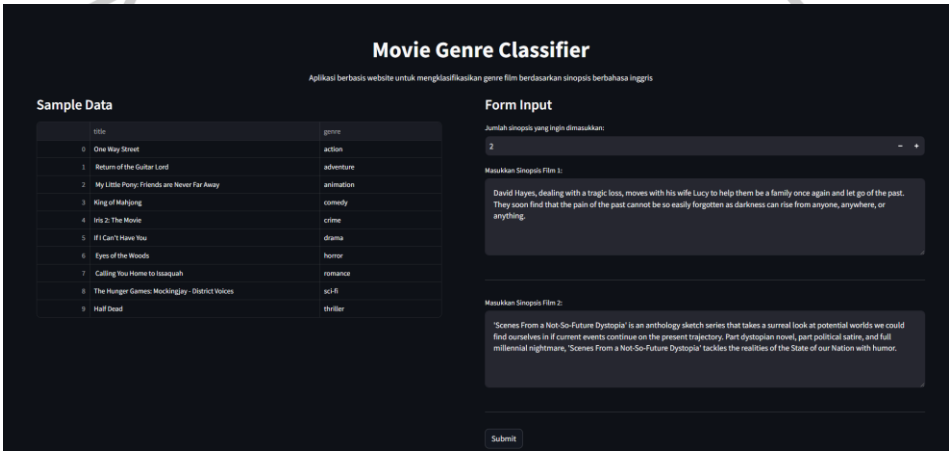
ditetapkan sebelumnya guna mengetahui kinerja dari aplikasi klasifikasi genre film berdasarkan sinopsis.

5.2.1 Pengujian *Black Box*

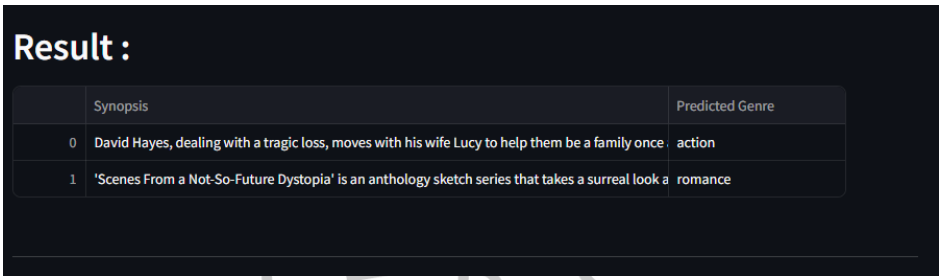
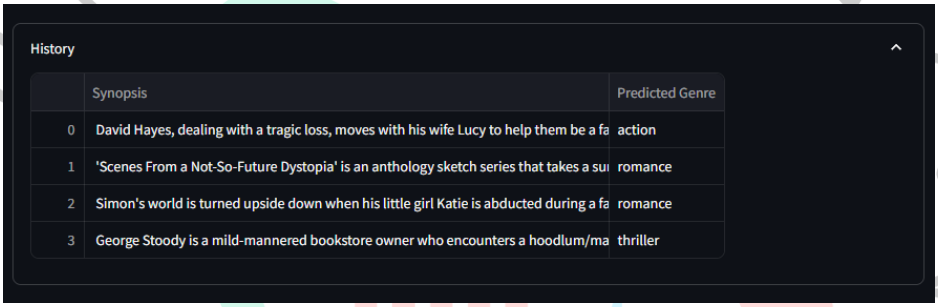
Pengujian *black box* akan memusatkan perhatian untuk mengevaluasi daya kerja fitur yang telah dibangun. Tujuannya adalah untuk melakukan verifikasi pada setiap fitur berfungsi agar sesuai dengan yang diharapkan.

Tabel 5. 6 Pengujian *Black Box*

<table>
<tr><th>No.</th><th>Skenario Pengujian</th><th>Hasil yang diharapkan</th></tr>
<tr><td>1.</td><td>User mendefinisikan jumlah sinopsis yang ingin dimasukkan</td><td>Sistem menampilkan form input sesuai dengan jumlah yang dimasukkan</td></tr>
<tr><td></td><td colspan="2">Hasil:
 

Kesimpulan: Sistem berhasil menyajikan form input sesuai jumlah yang dimasukkan</td></tr>
<tr><td>2.</td><td>User mengisi form input dengan sinopsis film</td><td>Sistem menampilkan isi form yang dimasukkan oleh user</td></tr>
<tr><td></td><td colspan="2">Hasil:
 

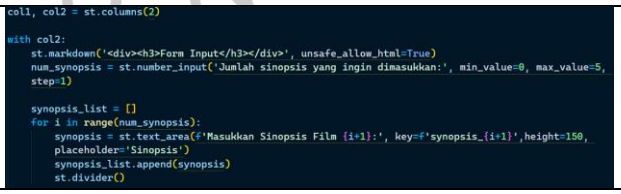
Kesimpulan: Sistem berhasil menyajikan isi dari form yang dimasukkan oleh user</td></tr>
</table>

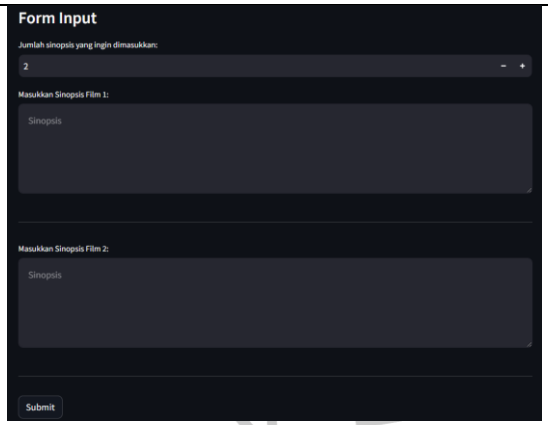
3.	*User* menekan tombol *submit* untuk memulai proses klasifikasi genre film	Sistem menampilkan tabel yang berisi sinopsis dan hasil klasifikasi
	 Kesimpulan: Sistem berhasil menyajikan tabel yang berisi sinopsis dan hasil klasifikasi	
4.	*User* menekan tombol history	Sistem menampilkan riwayat sinopsis yang telah dimasukkan sebelumnya
	Hasil:  Kesimpulan: Sistem berhasil menampilkan riwayat sinopsis yang telah dimasukkan sebelumnya	

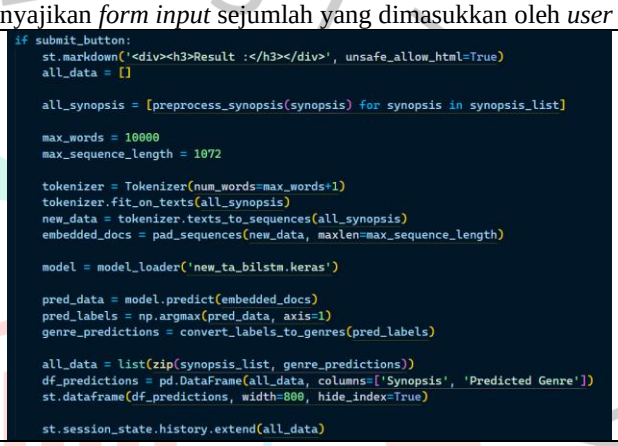
5.2.2 Pengujian *White Box*

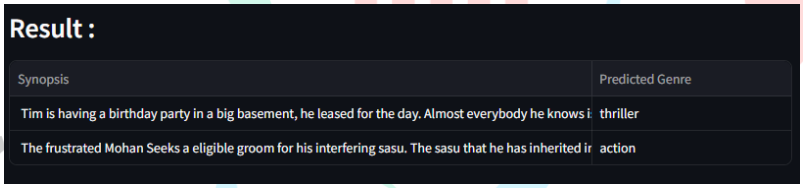
Pengujian *white box* akan memusatkan perhatian pada evaluasi struktur kode yang dikembangkan. Pengujian ini dilakukan untuk memverifikasi apakah kode program berjalan sesuai dengan yang diharapkan. Adapun hasil dari pengujian *white box*, ditunjukkan pada tabel di bawah ini.

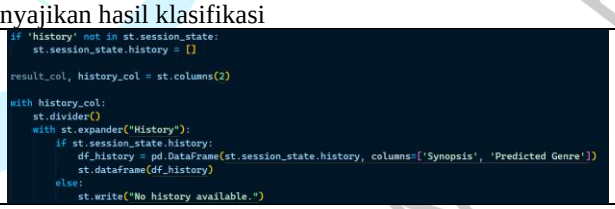
Tabel 5. 7 Pengujian *White Box*

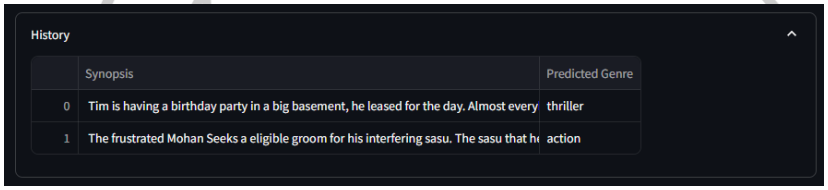
No.	Hasil yang diharapkan	Kode Program
1.	Sistem mampu menyajikan *form input* sesuai dengan jumlah yang dimasukkan	
	Hasil:	

<table>
<tr><td></td><td> 

Kesimpulan: Sistem berhasil menyajikan form input sejumlah yang dimasukkan oleh user</td></tr>
<tr><td>2.</td><td>Sistem mampu menyajikan hasil klasifikasi
 

Hasil:
 

Kesimpulan: Sistem berhasil menyajikan hasil klasifikasi</td></tr>
<tr><td>3.</td><td>Sistem mampu menyajikan riwayat atau history hasil klasifikasi
 

Hasil:
 

Kesimpulan: Sistem berhasil menyajikan riwayat hasil klasifikasi</td></tr>
</table>