4. IMPLEMENTASI

Pada bab ini akan dijelaskan mengenai implementasi sistem dari desain sistem yang telah dibahas pada bab sebelumnya. Pada Tabel 4.1 akan dijelaskan pemetaan segmen program secara singkat, termasuk nama proses dan keterangan setiap proses.

4.1. Import Library

Untuk melakukan penelitian ini maka dibutuhkan beberapa tools dan fungsi dari beberapa *library* dan *dependency*. *Tools* utama yang akan digunakan adalah Jupyter Lab untuk proses pembuatan model dan analisis dan Visual Studio Code. Sementara itu, *library* yang digunakan yaitu:

- Flask untuk membangun aplikasi website dengan bahasa pemrograman Python
- Sklearn untuk model Random Forest, SVM, metrics dan lainnya
- Tweepy untuk mengakses API dari Twitter
- Pickle untuk membantu menyimpan model
- Dan *library-library* lainnya

4.1.1. Twitter Authentication

Untuk dapat melakukan *scraping* data dari Twitter maka dibutuhkannya akun Twitter *developer.* Twitter developer dapat diakses di *link* berikut https://developer.twitter.com/en. Kemudian jika sudah punya akun bisa menekan tombol 'Sign Up' atau jika belum dapat menekan tombol 'Sign In' untuk mendaftarkan akun terlebih dahulu. Selanjutnya, untuk me-*generate* API *keys* dan token dapat dilakukan langkah-langkah berikut:

1. Navigasikan ke halaman "*Projects and Apps*".
2. Klik "*Keys and tokens*" di sebelah "*Settings*".
3. Klik tombol "*Regenerate*" di sebelah set kunci dan token.

Setelah API *keys* dan token sudah di *generate*, salin *bearer token, consumer key, consumer secret, access token, access token secret* dan tempel masing-masing token tersebut kedalam sebuah variabel berbeda. Token-token tersebut digunakan pada saat menyambungkan antara *library* 'tweepy' dengan Twitter *developer.* Pada segmen program 4.1 berisi contoh penulisan *code* untuk menyambungkan dengan API dari Twitter.

Segmen Program 4.1

Autentikasi Twitter dengan token dari Twitter *developer*

```
auth = tw.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
api = tw.API(auth, wait_on_rate_limit=True)
```

4.2. Implementasi Sistem

Desain sistem yang dibuat pada Bab 3 akan diimplementasikan pada subbab-subbab berikut dan dirincikan ke dalam beberapa segmen program yang daftar nya dapat dilihat pada Tabel 4.1.

Tabel 4.1

Pemetaan Segmen Program

Segmen Program	**Proses**	**Referensi**
4.2	Melakukan inisialisasi fungsi dan autentikasi Twitter	-
4.3	Melakukan *upload* dataset	-
4.4	Melakukan *preprocessing* data	Gambar 3.10 *Flowchart preprocessing* data Gambar 3.11 Flowchart preprocessing data untuk memproses satu akun Twitter
4.5	Melakukan proses LIWC	Gambar 3.12 Flowchart proses LIWC
4.6	Melakukan normalisasi data	-
4.7	Melakukan klasifikasi Random Forest	Gambar 3.13 Flowchart Random Forest untuk training Gambar 3.14 Flowchart Random Forest untuk trial

4.8	Melakukan perkalian antara hasil Random Forest dengan korelasi *learning style* (Siddiquei dan Khalid, 2018) dengan cara pertama	Gambar 3.18 Flowchart learning style cara pertama
4.9	Melakukan perkalian antara hasil Random Forest dengan korelasi *learning style* (Siddiquei dan Khalid, 2018) dengan cara kedua	Gambar 3.18 Flowchart learning style cara kedua
4.10	Tampilan *website* awal	-
4.11	Tampilan *website* saat menunjukkan hasil	-

4.2.1. Melakukan inisialisasi fungsi dan autentikasi Twitter

Segmen program 4.2 berisi inisialisasi fungsi, *library* LIWC dan autentikasi Twitter untuk bisa melakukan *scraping*. Fungsi-fungsi yang digunakan antara lain untuk menghapus emoji (fungsi deEmojify), mengganti kata-kata yang disingkat menjadi kata yang lengkap (fungsi decontracted), dan fungsi untuk *tokenize*. Untuk autentikasi Twitter, token yang asli tidak dapat dituliskan di dalam laporan ini karena token tersebut bersifat *private key* sehingga tidak bisa disebarluaskan.

Segmen Program 4.2
Melakukan inisialisasi fungsi dan variabel autentikasi Twitter

```
def deEmojify(inputString):
    regrex_pattern = re.compile(pattern = "["
       u"\U0001F600-\U0001F64F"  # emoticons
       u"\U0001F300-\U0001F5FF"  # symbols & pictographs
       u"\U0001F680-\U0001F6FF"  # transport & map symbols
       u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
       u"\U00002702-\U000027B0"
       u"\U000024C2-\U0001F251"
       u"\U0001F901-\U0001F9D0"
                          "]+", flags = re.UNICODE)
    return regrex_pattern.sub(r'',inputString)

def decontracted(phrase):
    phrase = re.sub(r"g.r.e", "graduate record examinations",
phrase)
    phrase = re.sub(r"till", "until", phrase)
```

```
    phrase = re.sub(r"r.i.p", "rest in peace", phrase)
    phrase = re.sub(r"xmas", "christmas", phrase)
    phrase = re.sub(r"x-mas", "christmas", phrase)
    phrase = re.sub(r"^ vw", "volkswagen", phrase)
    phrase = re.sub(r"^ ca", "california", phrase)
    phrase = re.sub(r"^ uk", "united kingdom", phrase)
    phrase = re.sub(r"^ ma", "massachusetts", phrase)
    phrase = re.sub(r"<3", " ", phrase)
    phrase = re.sub(r":3", " ", phrase)
    phrase = re.sub(r":d", " ", phrase)
    phrase = re.sub(r":p", " ", phrase)
    phrase = re.sub(r":o", " ", phrase)
    phrase = re.sub(r"PROPNAME", " ", phrase)
    phrase = re.sub("4r5e", "fuck", phrase)
    phrase = re.sub("5h1t", "fuck", phrase)
    phrase = re.sub("5hit", "fuck", phrase)
    phrase = re.sub("ass-fucker", "fuck", phrase)
    phrase = re.sub("assfucker", "fuck", phrase)
    phrase = re.sub("assfukka", "fuck", phrase)
    phrase = re.sub("asswhole", "fuck", phrase)
    phrase = re.sub("a_s_s", "fuck", phrase)
    phrase = re.sub("b!tch", "fuck", phrase)
    phrase = re.sub("b17ch", "fuck", phrase)
    phrase = re.sub("blow job", "fuck", phrase)
    phrase = re.sub("boiolas", "fuck", phrase)
    phrase = re.sub("bollok", "fuck", phrase)
    phrase = re.sub("boooobs", "fuck", phrase)
    phrase = re.sub("booooobs", "fuck", phrase)
    phrase = re.sub("boooooooobs", "fuck", phrase)
    phrase = re.sub("bunny fucker", "fuck", phrase)
    phrase = re.sub("buttmuch", "fuck", phrase)
    phrase = re.sub("c0cksucker", "fuck", phrase)
    phrase = re.sub("carpet muncher", "fuck", phrase)
    phrase = re.sub("cl1t", "fuck", phrase)
    phrase = re.sub("cockface", "fuck", phrase)
    phrase = re.sub("cockmunch", "fuck", phrase)
    phrase = re.sub("cockmuncher", "fuck", phrase)
    phrase = re.sub("cocksuka", "fuck", phrase)
    phrase = re.sub("cocksukka", "fuck", phrase)
    phrase = re.sub("cokmuncher", "fuck", phrase)
    phrase = re.sub("coksucka", "fuck", phrase)
    phrase = re.sub("cunillingus", "fuck", phrase)
    phrase = re.sub("cuntlick", "fuck", phrase)
    phrase = re.sub("cuntlicker", "fuck", phrase)
    phrase = re.sub("cuntlicking", "fuck", phrase)
    phrase = re.sub("cyalis", "fuck", phrase)
    phrase = re.sub("cyberfuc", "fuck", phrase)
    phrase = re.sub("cyberfuck", "fuck", phrase)
    phrase = re.sub("cyberfucked", "fuck", phrase)
    phrase = re.sub("cyberfucker", "fuck", phrase)
    phrase = re.sub("cyberfuckers", "fuck", phrase)
    phrase = re.sub("cyberfucking", "fuck", phrase)
```

```
    phrase = re.sub("dirsa", "fuck", phrase)
    phrase = re.sub("dlck", "fuck", phrase)
    phrase = re.sub("dog-fucker", "fuck", phrase)
    phrase = re.sub("donkeyribber", "fuck", phrase)
    phrase = re.sub("ejaculatings", "fuck", phrase)
    phrase = re.sub("ejakulate", "fuck", phrase)
    phrase = re.sub("f u c k", "fuck", phrase)
    phrase = re.sub("f u c k e r", "fuck", phrase)
    phrase = re.sub("f4nny", "fuck", phrase)
    phrase = re.sub("faggitt", "fuck", phrase)
    phrase = re.sub("faggs", "fuck", phrase)
    phrase = re.sub("fannyflaps", "fuck", phrase)
    phrase = re.sub("fannyfucker", "fuck", phrase)
    phrase = re.sub("fanyy", "fuck", phrase)
    phrase = re.sub("fingerfucker", "fuck", phrase)
    phrase = re.sub("fingerfuckers", "fuck", phrase)
    phrase = re.sub("fingerfucks", "fuck", phrase)
    phrase = re.sub("fistfuck", "fuck", phrase)
    phrase = re.sub("fistfucked", "fuck", phrase)
    phrase = re.sub("fistfucker", "fuck", phrase)
    phrase = re.sub("fistfuckers", "fuck", phrase)
    phrase = re.sub("fistfucking", "fuck", phrase)
    phrase = re.sub("fistfuckings", "fuck", phrase)
    phrase = re.sub("fistfucks", "fuck", phrase)
    phrase = re.sub("fuckingshitmotherfucker", "fuck",
phrase)
    phrase = re.sub("fuckwhit", "fuck", phrase)
    phrase = re.sub("fudge packer", "fuck", phrase)
    phrase = re.sub("fudgepacker", "fuck", phrase)
    phrase = re.sub("fukwhit", "fuck", phrase)
    phrase = re.sub("fukwit", "fuck", phrase)
    phrase = re.sub("fux0r", "fuck", phrase)
    phrase = re.sub("f_u_c_k", "fuck", phrase)
    phrase = re.sub("god-dam", "fuck", phrase)
    phrase = re.sub("kawk", "fuck", phrase)
    phrase = re.sub("knobead", "fuck", phrase)
    phrase = re.sub("knobed", "fuck", phrase)
    phrase = re.sub("knobend", "fuck", phrase)
    phrase = re.sub("knobjocky", "fuck", phrase)
    phrase = re.sub("knobjokey", "fuck", phrase)
    phrase = re.sub("kondum", "fuck", phrase)
    phrase = re.sub("kondums", "fuck", phrase)
    phrase = re.sub("kummer", "fuck", phrase)
    phrase = re.sub("kumming", "fuck", phrase)
    phrase = re.sub("kums", "fuck", phrase)
    phrase = re.sub("kunilingus", "fuck", phrase)
    phrase = re.sub("l3itch", "fuck", phrase)
    phrase = re.sub("m0f0", "fuck", phrase)
    phrase = re.sub("m0fo", "fuck", phrase)
    phrase = re.sub("m45terbate", "fuck", phrase)
    phrase = re.sub("ma5terb8", "fuck", phrase)
    phrase = re.sub("ma5terbate", "fuck", phrase)
```

```
    phrase = re.sub("master-bate", "fuck", phrase)
    phrase = re.sub("masterb8", "fuck", phrase)
    phrase = re.sub("masterbat3", "fuck", phrase)
    phrase = re.sub("masterbations", "fuck", phrase)
    phrase = re.sub("mof0", "fuck", phrase)
    phrase = re.sub("mothafuck", "fuck", phrase)
    phrase = re.sub("mothafuckaz", "fuck", phrase)
    phrase = re.sub("mothafucked", "fuck", phrase)
    phrase = re.sub("mothafucking", "fuck", phrase)
    phrase = re.sub("mothafuckings", "fuck", phrase)
    phrase = re.sub("mothafucks", "fuck", phrase)
    phrase = re.sub("mother fucker", "fuck", phrase)
    phrase = re.sub("motherfucked", "fuck", phrase)
    phrase = re.sub("motherfuckings", "fuck", phrase)
    phrase = re.sub("motherfuckka", "fuck", phrase)
    phrase = re.sub("motherfucks", "fuck", phrase)
    phrase = re.sub("muthafecker", "fuck", phrase)
    phrase = re.sub("muthafuckker", "fuck", phrase)
    phrase = re.sub("n1gga", "fuck", phrase)
    phrase = re.sub("n1gger", "fuck", phrase)
    phrase = re.sub("nigg3r", "fuck", phrase)
    phrase = re.sub("nigg4h", "fuck", phrase)
    phrase = re.sub("nob jokey", "fuck", phrase)
    phrase = re.sub("nobjocky", "fuck", phrase)
    phrase = re.sub("nobjokey", "fuck", phrase)
    phrase = re.sub("penisfucker", "fuck", phrase)
    phrase = re.sub("phuked", "fuck", phrase)
    phrase = re.sub("phuking", "fuck", phrase)
    phrase = re.sub("phukked", "fuck", phrase)
    phrase = re.sub("phukking", "fuck", phrase)
    phrase = re.sub("phuks", "fuck", phrase)
    phrase = re.sub("phuq", "fuck", phrase)
    phrase = re.sub("pigfucker", "fuck", phrase)
    phrase = re.sub("pimpis", "fuck", phrase)
    phrase = re.sub("pissflaps", "fuck", phrase)
    phrase = re.sub("rimjaw", "fuck", phrase)
    phrase = re.sub("s hit", "fuck", phrase)
    phrase = re.sub("scroat", "fuck", phrase)
    phrase = re.sub("sh!t", "fuck", phrase)
    phrase = re.sub("shitdick", "fuck", phrase)
    phrase = re.sub("shitfull", "fuck", phrase)
    phrase = re.sub("shitings", "fuck", phrase)
    phrase = re.sub("shittings", "fuck", phrase)
    phrase = re.sub("s_h_i_t", "fuck", phrase)
    phrase = re.sub("t1tt1e5", "fuck", phrase)
    phrase = re.sub("t1tties", "fuck", phrase)
    phrase = re.sub("teez", "fuck", phrase)
    phrase = re.sub("tittie5", "fuck", phrase)
    phrase = re.sub("tittiefucker", "fuck", phrase)
    phrase = re.sub("tittywank", "fuck", phrase)
    phrase = re.sub("tw4t", "fuck", phrase)
    phrase = re.sub("twathead", "fuck", phrase)
```

```
    phrase = re.sub("twunter", "fuck", phrase)
    phrase = re.sub("v14gra", "fuck", phrase)
    phrase = re.sub("v1gra", "fuck", phrase)
    phrase = re.sub("w00se", "fuck", phrase)
    phrase = re.sub("whoar", "fuck", phrase)

    return phrase

def tokenize(text):
    for match in re.finditer(r'\w+', text, re.UNICODE):
        yield match.group(0)

# Twitter auth variable
bearer_token= <private key>
consumer_secret= <private key>
access_token= <private key>
access_token_secret= <private key>
# LIWC Dictionary
parse, category_names =
liwc.load_token_parser('LIWC2007_English100131.dic')
```

4.2.2. Melakukan upload dataset

Segmen program 4.3 berisi tentang upload dataset Twitter yang disimpan dalam bentuk .csv dimana data ini digunakan untuk training model.

Segmen Program 4.3

Melakukan *upload* dataset

```
#Load dataset
file = "Dataset/mypersonality_edited.csv"
datatext = pd.read_csv(file, sep=',')
datatext.head()
```

4.2.3. Melakukan preprocessing data

Segmen program 4.4 berisi tentang data *tweet* yang sudah di upload diproses terlebih dahulu sebelum digunakan. Untuk *tokenization* dilakukan oleh fungsi word_tokenize yang ada di dalam *library* nltk.tokenize.

Segmen Program 4.4

Melakukan preprocessing data

```
df = pd.DataFrame()
```

```
df = datatext[['#AUTHID', 'STATUS', 'cEXT', 'cNEU', 'cAGR',
'cCON', 'cOPN']].copy()
df = df.rename(columns={'#AUTHID': 'id', 'STATUS': 'old'})

# replace n with 0 and y with 1 for cEXT, cNEU, cOPN, cCON,
cAGR
df['cEXT'] = df['cEXT'].replace('y','1')
df['cEXT'] = df['cEXT'].replace('n','0')
df['cNEU'] = df['cNEU'].replace('y','1')
df['cNEU'] = df['cNEU'].replace('n','0')
df['cOPN'] = df['cOPN'].replace('y','1')
df['cOPN'] = df['cOPN'].replace('n','0')
df['cCON'] = df['cCON'].replace('y','1')
df['cCON'] = df['cCON'].replace('n','0')
df['cAGR'] = df['cAGR'].replace('y','1')
df['cAGR'] = df['cAGR'].replace('n','0')

# change datatype -> all in object type
df = df.convert_dtypes()
df["cEXT"] = pd.to_numeric(df["cEXT"])
df['cNEU'] = pd.to_numeric(df["cNEU"])
df['cOPN'] = pd.to_numeric(df["cOPN"])
df['cCON'] = pd.to_numeric(df["cCON"])
df['cAGR'] = pd.to_numeric(df["cAGR"])
print(df.dtypes)

# case folding or lowering case
df["processed"] = df["old"].apply(lambda s: s.casefold())

# delete link
df['processed'] = df['processed'].apply(lambda s:
re.sub(r"(?:\@|https?\://)\S+", "", s))

# expanding contractions & delete emoji
temp = []
temp_2 = ''
for text in df['processed']:
#     temp.append(contractions.fix(text))
    temp_2 = contractions.fix(text)
    temp_2 = decontracted(temp_2)
    temp_2 = deEmojify(temp_2)
    temp.append(temp_2)
df['processed'] = temp

# Remove punctuation
temp = []
punc = '''\()-[]{};:'"\,<>/@#$%^&*_~\+'''
for words in df['processed']:
    for ele in words:
        if ele in punc:
            words = words.replace(ele, "")
        if ele == '\n':
```

```
            words = words.replace(ele, " ")
#     print(words)
    temp.append(words)
df['processed'] = temp

# Tokenize
temp = []
for text in df['processed']:
    temp.append(word_tokenize(text))

df['tokenize'] = temp
pd.set_option('display.max_colwidth', None)
df.head()
```

4.2.4. Melakukan proses LIWC

Di dalam segmen program 4.5 ini, menunjukkan bagaimana proses LIWC dilakukan setelah *preprocessing* data.

Segmen Program 4.5

Melakukan proses LIWC

```
df_size = int(len(df['tokenize']))
for size in range (df_size):
    df_result = Counter(category for token in
df['tokenize'][size] for category in parse(token))
    df.loc[size, 'achieve'] = df_result['achieve']
    df.loc[size, 'adverb'] = df_result['adverb']
    df.loc[size, 'affect'] = df_result['affect']
    df.loc[size, 'anger'] = df_result['anger']
    df.loc[size, 'anx'] = df_result['anx']
    df.loc[size, 'article'] = df_result['article']
    df.loc[size, 'assent'] = df_result['assent']
    df.loc[size, 'auxverb'] = df_result['auxverb']
    df.loc[size, 'bio'] = df_result['bio']
    df.loc[size, 'body'] = df_result['body']
    df.loc[size, 'cause'] = df_result['cause']
    df.loc[size, 'certain'] = df_result['certain']
    df.loc[size, 'cogmech'] = df_result['cogmech']
    df.loc[size, 'conj'] = df_result['conj']
    df.loc[size, 'death'] = df_result['death']
    df.loc[size, 'discrep'] = df_result['discrep']
    df.loc[size, 'excl'] = df_result['excl']
    df.loc[size, 'family'] = df_result['family']
    df.loc[size, 'feel'] = df_result['feel']
    df.loc[size, 'filler'] = df_result['filler']
    df.loc[size, 'friend'] = df_result['friend']
    df.loc[size, 'funct'] = df_result['funct']
    df.loc[size, 'future'] = df_result['future']
    df.loc[size, 'health'] = df_result['health']
```

```
    df.loc[size, 'hear'] = df_result['hear']
    df.loc[size, 'home'] = df_result['home']
    df.loc[size, 'humans'] = df_result['humans']
    df.loc[size, 'i'] = df_result['i']
    df.loc[size, 'incl'] = df_result['incl']
    df.loc[size, 'ingest'] = df_result['ingest']
    df.loc[size, 'inhib'] = df_result['inhib']
    df.loc[size, 'insight'] = df_result['insight']
    df.loc[size, 'ipron'] = df_result['ipron']
    df.loc[size, 'leisure'] = df_result['leisure']
    df.loc[size, 'money'] = df_result['money']
    df.loc[size, 'motion'] = df_result['motion']
    df.loc[size, 'negate'] = df_result['negate']
    df.loc[size, 'negemo'] = df_result['negemo']
    df.loc[size, 'nonfl'] = df_result['nonfl']
    df.loc[size, 'number'] = df_result['number']
    df.loc[size, 'past'] = df_result['past']
    df.loc[size, 'percept'] = df_result['percept']
    df.loc[size, 'posemo'] = df_result['posemo']
    df.loc[size, 'ppron'] = df_result['ppron']
    df.loc[size, 'preps'] = df_result['preps']
    df.loc[size, 'present'] = df_result['present']
    df.loc[size, 'pronoun'] = df_result['pronoun']
    df.loc[size, 'quant'] = df_result['quant']
    df.loc[size, 'relativ'] = df_result['relativ']
    df.loc[size, 'relig'] = df_result['relig']
    df.loc[size, 'sad'] = df_result['sad']
    df.loc[size, 'see'] = df_result['see']
    df.loc[size, 'sexual'] = df_result['sexual']
    df.loc[size, 'shehe'] = df_result['shehe']
    df.loc[size, 'social'] = df_result['social']
    df.loc[size, 'space'] = df_result['space']
    df.loc[size, 'swear'] = df_result['swear']
    df.loc[size, 'tentat'] = df_result['tentat']
    df.loc[size, 'they'] = df_result['they']
    df.loc[size, 'time'] = df_result['time']
    df.loc[size, 'verb'] = df_result['verb']
    df.loc[size, 'we'] = df_result['we']
    df.loc[size, 'work'] = df_result['work']
    df.loc[size, 'you'] = df_result['you']
df.describe()
```

4.2.5. Melakukan normalisasi data

Dalam segmen program 4.6 dilakukan normalisasi data. Terdapat dua langkah yang harus dilalui. Langkah pertama adalah mendapatkan panjang token setiap *record* data. Setelah itu, mengelompokkan data berdasarkan id dan setiap *feature* dari LIWC dan panjang token ditambahkan berdasarkan pengelompokannya. Kemudian membuat sebuah *DataFrame* baru

yang menampung id, panjang token yang sudah dijumlah berdasarkan id-nya, kemudian hasil normalisasi per baris dan hasil *Big 5 Personality* yang sudah ada di *DataFrame* sebelumnya.

Segmen Program 4.6

Melakukan normalisasi data

```
scaling_x = [len(token) for token in df['tokenize']]
df['len token'] = scaling_x
df_groupby = df.groupby('id', as_index=False ).agg({'len
token': 'sum', "achieve":"sum", "adverb":"sum",
"affect":"sum", "anger":"sum", "anx":"sum", "article":"sum",
"assent":"sum", "auxverb":"sum", "bio":"sum", "body":"sum",
"cause":"sum", "certain":"sum", "cogmech":"sum", "conj":"sum",
"death":"sum", "discrep":"sum", "excl":"sum", "family":"sum",
"feel":"sum", "filler":"sum", "friend":"sum", "funct":"sum",
"future":"sum", "health":"sum", "hear":"sum", "home":"sum",
"humans":"sum", "i":"sum", "incl":"sum", "ingest":"sum",
"inhib":"sum", "insight":"sum", "ipron":"sum",
"leisure":"sum", "money":"sum", "motion":"sum",
"negate":"sum", "negemo":"sum", "nonfl":"sum", "number":"sum",
"past":"sum", "percept":"sum", "posemo":"sum", "ppron":"sum",
"preps":"sum", "present":"sum", "pronoun":"sum",
"quant":"sum", "relativ":"sum", "relig":"sum", "sad":"sum",
"see":"sum", "sexual":"sum", "shehe":"sum", "social":"sum",
"space":"sum", "swear":"sum", "tentat":"sum", "they":"sum",
"time":"sum", "verb":"sum", "we":"sum", "work":"sum",
"you":"sum", 'cEXT':'mean', 'cNEU':'mean', 'cAGR':'mean',
'cCON':'mean', 'cOPN':'mean'})
df_groupby

# Creates pandas DataFrame.
df_normalize = pd.DataFrame()
df_normalize['id'] = df_groupby['id']
df_normalize['len token'] = df_groupby['len token']

df_normalize['achieve'] =
df_groupby['achieve']/df_normalize['len token']
df_normalize['adverb'] =
df_groupby['adverb']/df_normalize['len token']
df_normalize['affect'] =
df_groupby['affect']/df_normalize['len token']
df_normalize['anger'] = df_groupby['anger']/df_normalize['len
token']
df_normalize['anx'] = df_groupby['anx']/df_normalize['len
token']
df_normalize['article'] =
df_groupby['article']/df_normalize['len token']
df_normalize['assent'] =
df_groupby['assent']/df_normalize['len token']
df_normalize['auxverb'] =
```

```
df_groupby['auxverb']/df_normalize['len token']
df_normalize['bio'] = df_groupby['bio']/df_normalize['len
token']
df_normalize['body'] = df_groupby['body']/df_normalize['len
token']
df_normalize['cause'] = df_groupby['cause']/df_normalize['len
token']
df_normalize['certain'] =
df_groupby['certain']/df_normalize['len token']
df_normalize['cogmech'] =
df_groupby['cogmech']/df_normalize['len token']
df_normalize['conj'] = df_groupby['conj']/df_normalize['len
token']
df_normalize['death'] = df_groupby['death']/df_normalize['len
token']
df_normalize['discrep'] =
df_groupby['discrep']/df_normalize['len token']
df_normalize['excl'] = df_groupby['excl']/df_normalize['len
token']
df_normalize['family'] =
df_groupby['family']/df_normalize['len token']
df_normalize['feel'] = df_groupby['feel']/df_normalize['len
token']
df_normalize['filler'] =
df_groupby['filler']/df_normalize['len token']
df_normalize['friend'] =
df_groupby['friend']/df_normalize['len token']
df_normalize['funct'] = df_groupby['funct']/df_normalize['len
token']
df_normalize['future'] =
df_groupby['future']/df_normalize['len token']
df_normalize['health'] =
df_groupby['health']/df_normalize['len token']
df_normalize['hear'] = df_groupby['hear']/df_normalize['len
token']
df_normalize['home'] = df_groupby['home']/df_normalize['len
token']
df_normalize['humans'] =
df_groupby['humans']/df_normalize['len token']
df_normalize['i'] = df_groupby['i']/df_normalize['len token']
df_normalize['incl'] = df_groupby['incl']/df_normalize['len
token']
df_normalize['ingest'] =
df_groupby['ingest']/df_normalize['len token']
df_normalize['inhib'] = df_groupby['inhib']/df_normalize['len
token']
df_normalize['insight'] =
df_groupby['insight']/df_normalize['len token']
df_normalize['ipron'] = df_groupby['ipron']/df_normalize['len
token']
df_normalize['leisure'] =
df_groupby['leisure']/df_normalize['len token']
```

```
df_normalize['money'] = df_groupby['money']/df_normalize['len
token']
df_normalize['motion'] =
df_groupby['motion']/df_normalize['len token']
df_normalize['negate'] =
df_groupby['negate']/df_normalize['len token']
df_normalize['negemo'] =
df_groupby['negemo']/df_normalize['len token']
df_normalize['nonfl'] = df_groupby['nonfl']/df_normalize['len
token']
df_normalize['number'] =
df_groupby['number']/df_normalize['len token']
df_normalize['past'] = df_groupby['past']/df_normalize['len
token']
df_normalize['percept'] =
df_groupby['percept']/df_normalize['len token']
df_normalize['posemo'] =
df_groupby['posemo']/df_normalize['len token']
df_normalize['ppron'] = df_groupby['ppron']/df_normalize['len
token']
df_normalize['preps'] = df_groupby['preps']/df_normalize['len
token']
df_normalize['present'] =
df_groupby['present']/df_normalize['len token']
df_normalize['pronoun'] =
df_groupby['pronoun']/df_normalize['len token']
df_normalize['quant'] = df_groupby['quant']/df_normalize['len
token']
df_normalize['relativ'] =
df_groupby['relativ']/df_normalize['len token']
df_normalize['relig'] = df_groupby['relig']/df_normalize['len
token']
df_normalize['sad'] = df_groupby['sad'] /df_normalize['len
token']
df_normalize['see'] = df_groupby['see']/df_normalize['len
token']
df_normalize['sexual'] =
df_groupby['sexual']/df_normalize['len token']
df_normalize['shehe'] = df_groupby['shehe']/df_normalize['len
token']
df_normalize['social'] =
df_groupby['social']/df_normalize['len token']
df_normalize['space'] = df_groupby['space']/df_normalize['len
token']
df_normalize['swear'] = df_groupby['swear']/df_normalize['len
token']
df_normalize['tentat'] =
df_groupby['tentat']/df_normalize['len token']
df_normalize['they'] = df_groupby['they']/df_normalize['len
token']
df_normalize['time'] = df_groupby['time']/df_normalize['len
token']
```

```
df_normalize['verb'] = df_groupby['verb']/df_normalize['len
token']
df_normalize['we'] = df_groupby['we']/df_normalize['len
token']
df_normalize['work'] = df_groupby['work']/df_normalize['len
token']
df_normalize['you'] = df_groupby['you'] /df_normalize['len
token']

# add Big 5 Personality results to df_normalize dataframe
df_normalize['cEXT'] = df_groupby['cEXT']
df_normalize['cNEU'] = df_groupby['cNEU']
df_normalize['cAGR'] = df_groupby['cAGR']
df_normalize['cCON'] = df_groupby['cCON']
df_normalize['cOPN'] = df_groupby['cOPN']

# print the data
df_normalize
```

4.2.6. Melakukan klasifikasi Random Forest

Di dalam segmen program 4.7 berisi tentang klasifikasi dari *Random Forest* dari hasil normalisasi. Setiap *personality* dari *Big 5 Personality* akan dibuat modelnya masing-masing tapi untuk yang dituliskan pada segmen program 4.6 ini adalah hanya EXT atau kepribadian *extravert*. Hal ini dikarenakan penulisannya serupa hanya saja di penamaan variabelnya saja yang berbeda.

Segmen Program 4.7

Melakukan klasifikasi Random Forest

```
features = ["achieve", "adverb", "affect", "anger", "anx",
"article", "assent", "auxverb", "bio", "body", "cause",
"certain", "cogmech", "conj", "death", "discrep", "excl",
"family", "feel", "filler", "friend", "funct", "future",
"health", "hear", "home", "humans", "i", "incl", "ingest",
"inhib", "insight", "ipron", "leisure", "money", "motion",
"negate", "negemo", "nonfl", "number", "past", "percept",
"posemo", "ppron", "preps", "present", "pronoun", "quant",
"relativ", "relig", "sad", "see", "sexual", "shehe", "social",
"space", "swear", "tentat", "they", "time", "verb", "we",
"work", "you"]
x = df_normalize[features]
y = df_normalize['cEXT']
# Split dataset into training set and test set
x_train, x_test, y_train_ext, y_test_ext = train_test_split(x,
y, test_size=0.2, random_state = 1) # 80% training and 20%
test
```

```
rf_ext=BalancedRandomForestClassifier(random_state=1,
max_features='log2', min_samples_leaf= 5, max_depth= 50)
rf_ext.fit(x_train, y_train_ext)
train = rf_ext.predict(x_train)
test = rf_ext.predict(x_test)
print("Train accuracy: ", metrics.accuracy_score(y_train_ext,
train))
```

4.2.7. Melakukan perkalian antara hasil Random Forest dengan korelasi *learning style*

Pada segmen program 4.8 berisi perkalian antara hasil *Big 5 Personality Random Forest* dikalikan dengan korelasi *learning style* milik Siddiquei dan Khalid (2018). Terdapat dua cara yang dilakukan yang mengacu pada bab 3 subbab *flowchart learning style*.

Segmen Program 4.8

Melakukan perkalian antara hasil Random Forest dengan korelasi *learning style* dengan cara pertama (Siddiquei dan Khalid, 2018)

```
# correlation
const_active_ext = 0.228
const_active_opn = 0.234

const_reflective_ext = 0.236
const_reflective_opn = -0.243

const_sensing_agr = 0.261
const_sensing_con = 0.239

const_intuitive_agr = -0.268
const_intuitive_con = 0.247

const_visual_agr = 0.335
const_visual_opn = 0.376

const_verbal_agr = -0.335
const_verbal_opn = -0.376

const_seq_con = -0.041

const_global_con = 0.041

# multiply
def multiply_active(ext_pred, agr_pred, con_pred, neu_pred,
opn_pred):
    active_ext = const_active_ext * ext_pred
    active_agr = const_active_agr * agr_pred
```

```
    active_con = const_active_con * con_pred
    active_neu = const_active_neu * neu_pred
    active_opn = const_active_opn * opn_pred
    return (active_ext + active_agr + active_con + active_neu
+ active_opn)*0.11

def multiply_reflective(ext_pred, agr_pred, con_pred,
neu_pred, opn_pred):
    reflective_ext = const_reflective_ext * ext_pred
    reflective_agr = const_reflective_agr * agr_pred
    reflective_con = const_reflective_con * con_pred
    reflective_neu = const_reflective_neu * neu_pred
    reflective_opn = const_reflective_opn * opn_pred
    return (reflective_ext + reflective_agr + reflective_con +
reflective_neu + reflective_opn) * 0.11

def multiply_sensing(ext_pred, agr_pred, con_pred, neu_pred,
opn_pred):
    sensing_ext = const_sensing_ext * ext_pred
    sensing_agr = const_sensing_agr * agr_pred
    sensing_con = const_sensing_con * con_pred
    sensing_neu = const_sensing_neu * neu_pred
    sensing_opn = const_sensing_opn * opn_pred
    return (sensing_ext + sensing_agr + sensing_con +
sensing_neu + sensing_opn) * 0.11

def multiply_intuitive(ext_pred, agr_pred, con_pred, neu_pred,
opn_pred):
    intuitive_ext = const_intuitive_ext * ext_pred
    intuitive_agr = const_intuitive_agr * agr_pred
    intuitive_con = const_intuitive_con * con_pred
    intuitive_neu = const_intuitive_neu * neu_pred
    intuitive_opn = const_intuitive_opn * opn_pred
    return (intuitive_ext + intuitive_agr + intuitive_con +
intuitive_neu + intuitive_opn) * 0.11

def multiply_visual(ext_pred, agr_pred, con_pred, neu_pred,
opn_pred):
    visual_ext = const_visual_ext * ext_pred
    visual_agr = const_visual_agr * agr_pred
    visual_con = const_visual_con * con_pred
    visual_neu = const_visual_neu * neu_pred
    visual_opn = const_visual_opn * opn_pred
    return (visual_ext + visual_agr + visual_con + visual_neu
+ visual_opn) *0.11

def multiply_verbal(ext_pred, agr_pred, con_pred, neu_pred,
opn_pred):
    verbal_ext = const_verbal_ext * ext_pred
    verbal_agr = const_verbal_agr * agr_pred
    verbal_con = const_verbal_con * con_pred
    verbal_neu = const_verbal_neu * neu_pred
```

```
    verbal_opn = const_verbal_opn * opn_pred
    return (verbal_ext + verbal_agr + verbal_con + verbal_neu
+ verbal_opn) * 0.11

def multiply_seq(ext_pred, agr_pred, con_pred, neu_pred,
opn_pred):
    seq_ext = const_seq_ext * ext_pred
    seq_agr = const_seq_agr * agr_pred
    seq_con = const_seq_con * con_pred
    seq_neu = const_seq_neu * neu_pred
    seq_opn = const_seq_opn * opn_pred
    return (seq_ext + seq_agr + seq_con + seq_neu + seq_opn) *
0.11

def multiply_global(ext_pred, agr_pred, con_pred, neu_pred,
opn_pred):
    global_ext = const_global_ext * ext_pred
    global_agr = const_global_agr * agr_pred
    global_con = const_global_con * con_pred
    global_neu = const_global_neu * neu_pred
    global_opn = const_global_opn * opn_pred
    return (global_ext + global_agr + global_con + global_neu
+ global_opn) * 0.11


result_active = multiply_active(ext_pred, agr_pred, con_pred,
neu_pred, opn_pred)
    result_reflective = multiply_reflective(ext_pred,
agr_pred, con_pred, neu_pred, opn_pred)
    result_sensing = multiply_sensing(ext_pred, agr_pred,
con_pred, neu_pred, opn_pred)
    result_intuitive = multiply_intuitive(ext_pred, agr_pred,
con_pred, neu_pred, opn_pred)
    result_visual = multiply_visual(ext_pred, agr_pred,
con_pred, neu_pred, opn_pred)
    result_verbal = multiply_verbal(ext_pred, agr_pred,
con_pred, neu_pred, opn_pred)
    result_seq = multiply_seq(ext_pred, agr_pred, con_pred,
neu_pred, opn_pred)
    result_global = multiply_global(ext_pred, agr_pred,
con_pred, neu_pred, opn_pred)


results = {
        "active": result_active,
        "reflective": result_reflective,
        "sensing": result_sensing,
        "intuitive": result_intuitive,
        "visual": result_visual,
        "verbal": result_verbal,
        "seq": result_seq,
        "global": result_global
```

```
    }
results = sorted(results.items(), key=lambda x: x[1],
reverse=True)
print(results)
```

Segmen Program 4.9

Melakukan perkalian antara hasil Random Forest dengan korelasi *learning style* dengan cara kedua (Siddiquei dan Khalid, 2018)

```
result_active =
(const_active_ext/(const_active_ext+const_active_opn)*(ext_pre
d/50-1) +
const_active_opn/(const_active_ext+const_active_opn)*(opn_pred
/50-1))*11
result_reflective =
(const_reflective_ext/(const_reflective_ext-
const_reflective_opn)*(ext_pred/50-1) +
const_reflective_opn/(const_reflective_ext-
const_reflective_opn)*(opn_pred/50-1))*11
result_sensing =
(const_sensing_agr/(const_sensing_agr+const_sensing_con)*(agr_
pred/50-1) +
const_sensing_con/(const_sensing_agr+const_sensing_con)*(con_p
red/50-1))*11
result_intuitive = (const_intuitive_agr/(const_intuitive_agr-
const_intuitive_con)*(1-agr_pred/50) +
const_intuitive_con/(const_intuitive_con-
const_intuitive_agr)*(con_pred/50-1)) * 11
result_visual =
(const_visual_agr/(const_visual_agr+const_visual_opn)*(agr_pre
d/50-1) +
const_visual_opn/(const_visual_agr+const_visual_opn)*(opn_pred
/50-1))*11
result_verbal =
(const_verbal_agr/(const_verbal_agr+const_verbal_opn)*(1-
agr_pred/50) +
const_verbal_opn/(const_verbal_agr+const_verbal_opn)*(1-
opn_pred/50))*11
result_seq = (1 - neu_pred/50)*11
result_global = (neu_pred/50-1) * 11


results = {
        "active": result_active,
        "reflective": result_reflective,
        "sensing": result_sensing,
        "intuitive": result_intuitive,
        "visual": result_visual,
        "verbal": result_verbal,
```

```
        "seq": result_seq,
        "global": result_global
    }
results = sorted(results.items(), key=lambda x: x[1],
reverse=True)
print(results)
```

4.2.8. Website

Pada segmen program 4.10 berisi mengenai tampilan *website* awal dan segmen program 4.11 berisi mengenai tampilan *website* saat menunjukkan hasil.

Segmen Program 4.10

Tampilan *website* awal

```
{% extends 'base.html' %}

{% block head %}
<title>Undergraduate Project</title>
{% endblock %}

{% block body %}
<div class="container">
    <div class="row align-items-center mt-5">
        <div class="col-5">
            <form method="POST" method="POST" action="/tweet">
                <div class="form-group">
                    <label for="oneTweet">Your tweet :</label>
                    <textarea class="form-control"
name="oneTweet" id="oneTweet" rows="7"
                        placeholder="I'm looking forward to
meet you in person! See you soon"
                        style="font-size: 15px;"
required></textarea>
                    <input class="form-control" name="type"
id="type" value="tweet" style="font-size: 15px" hidden>
                </div>
                <div class="d-grid gap-2 mx-auto mt-3">
                    <input class="btn" type="submit"
value="Examine" style="color: white; background-color:
#2082c9;" >
                </div>
            </form>
        </div>
        <div class="col-2 text-center">
            <h5>or</h5>
        </div>
        <div class="col-5">
            <form method="POST" method="POST"
```

```
action="/account">
                <div class="form-group">
                    <label for="account_name">Your
account:</label>
                    <textarea class="form-control"
name="account_name" id="account_name" rows="7"
placeholder="BlessMeGod (without @)"
                        style="font-size: 15px;"
required></textarea>
                    <input class="form-control" name="type"
id="type" value="account" style="font-size: 15px" hidden>
                </div>
                <div class="d-grid gap-2 mx-auto mt-3">
                    <input class="btn" type="submit"
value="Scrape and examine" style="color: white; background-
color: #2082c9;" >
                </div>
            </form>
        </div>
    </div>
</div>
{% endblock %}
```

Segmen Program 4.11

Tampilan *website* saat menunjukkan hasil

```
{% extends 'base.html' %}

{% block head %}
<title>Result for {{ type }}</title>
{% endblock %}

{% block body %}
<div class="container mt-3">
  {% if type == 'tweet' %}
  <p>Your tweet:</p>
  {% else %}
  <p>Your account:</p>
  {% endif %}

  <div class="card">
    <div class="card-body">
      <i>"{{ tweet }}"</i>
    </div>
  </div>

  <p class="mt-5">Your result:</p>
  <div class="row">
    <div class="col">
      <h5>
        <center>BIG FIVE PERSONALITY</center>
```

```
      </h5>
      <div class="row mt-5">
        <div class="col text-end">
          Open
        </div>
        <div class="col">
          <div class="progress">
            <div class="progress-bar" role="progressbar"
style="width: {{ personality.open }}%" aria-valuemin="0"
             aria-
valuemax="100">{{ personality.open|round }}%</div>
          </div>
        </div>
        <div class="col"></div>
      </div>
      <div class="row mt-2">
        <div class="col text-end">
          Conscientious
        </div>
        <div class="col">
          <div class="progress">
            <div class="progress-bar" role="progressbar"
style="width:{{ personality.con }}%" aria-valuemin="0" aria-
valuemax="100">{{ personality.con|round }}%</div>
          </div>
        </div>
        <div class="col"></div>
      </div>
      <div class="row mt-2">
        <div class="col text-end">
          Extrovert
        </div>
        <div class="col">
          <div class="progress">
            <div class="progress-bar" role="progressbar"
style="width:{{ personality.ext }}%" aria-valuemin="0"
             aria-
valuemax="100">{{ personality.ext|round }}%</div>
          </div>
        </div>
        <div class="col"></div>
      </div>
      <div class="row mt-2">
        <div class="col text-end">
          Agreeable
        </div>
        <div class="col">
          <div class="progress">
            <div class="progress-bar" role="progressbar"
style="width:{{ personality.agr }}%" aria-valuemin="0"
             aria-
valuemax="100">{{ personality.agr|round }}%</div>
```

```
          </div>
        </div>
        <div class="col"></div>
      </div>
      <div class="row mt-2">
        <div class="col text-end">
          Neurotic
        </div>
        <div class="col">
          <div class="progress">
            <div class="progress-bar" role="progressbar"
style="width: {{ personality.neu }}%" aria-valuemin="0"
              aria-
valuemax="100">{{ personality.neu|round }}%</div>
          </div>
        </div>
        <div class="col"></div>
      </div>
    </div>
    <div class="col">
      <h5>
        <center>LEARNING STYLE</center>
      </h5>
      <div class="row">
        <img src="static/{{ learning }}.png">
      </div>
      <div class="row">
        <ul>
          <li>
            If your score for a dimension is 1 or 3, you are
fairly well balanced on the two categories of that dimension,
with only a mild preference for one or the other.
          </li>
          <li>
            If your score for a dimension is 5 or 7, you have
a moderate preference for one category of that dimension. You
may learn less easily in an environment that fails to address
that preference at least some of the time than you would in a
more balanced environment.
          </li>
          <li>
            If your score for a dimension is 9 or 11, you have
a strong preference for one category of that dimension. You
may have difficulty learning in an environment that fails to
address that preference at least some of the time.
          </li>
        </ul>
        <br>
      </div>
    </div>
  </div>
  <br><br></div>
```

```
{% endblock %}
```