

Day 13

Text Analytics / Chatbot

XML (Extensible Markup Language)

- A text-based format that allows for the structuring of electronic documents and is not limited to a set of labels.
전자문서를 구조화하는 텍스트기반의 포맷. 텍스트를 만들어 사용할 수 있음

```
<?xml version="1.0" standalone="yes" ?>
- <shop location="Birmingham" size="Large">
  - <food>
    <Name>Apple</Name>
    <type>fruit</type>
    <cost>15</cost>
  </food>
  - <food>
    <Name>Carrot</Name>
    <type>vegetable</type>
    <cost>10</cost>
  </food>
</shop>
```

Jason

- A syntax for storing and exchanging data. 데이터를 저장하고 교환하는 문법
- Text, written with JavaScript object notation 자바스크립트 오브젝트기호를 사용
- Similar to python dictionary 파이썬 딕셔너리와 비슷

```
[  
  {  
    "Id": 0,  
    "FirstName": "string",  
    "LastName": "string",  
    "Name": "string",  
    "EmailAddress": "string",  
    "TerritoryId": 0  
  }  
]
```

Python dictionary

```
dict = {'Name': 'Hasan', 'Age': 20, 'Sex': 'Male'}  
  
dict['Age'] = 24; # Update existing entry  
dict['Address'] = "ABC"; # Add new entry  
dict.pop('Name', None) # Remove the existing
```

Convert from Python to JSON:

```
import json

x = {
    "name": "John",
    "age": 30,
    "city": "New York"
}

y = json.dumps(x)
print(y)
```

- You can convert a Python object into a JSON string by using `json.dumps()`.
제이슨 덤프즈로 파이썬
오브젝트를 제이슨 문자열로
만들수 있음
- The following types can convert into JSON strings: dict, list, tuple, string, int, float, True, False, None 딕셔너리,
리스트, 튜플 등을 다양한
오브젝트를 변환할수 있음

Convert from JSON to Python

```
import json

x =  '{"name":"John", "age":30, "city":"New York"}'
y = json.loads(x)
print(y)

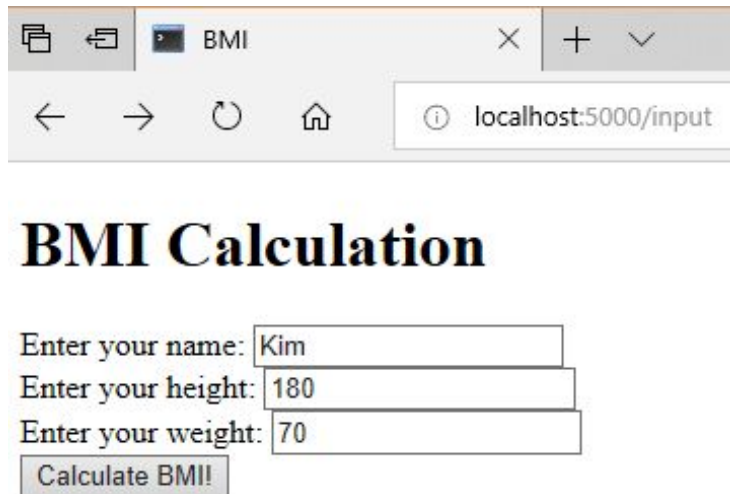
print(y["age"]) #Get the value of the "age" key
```

Exercise #13

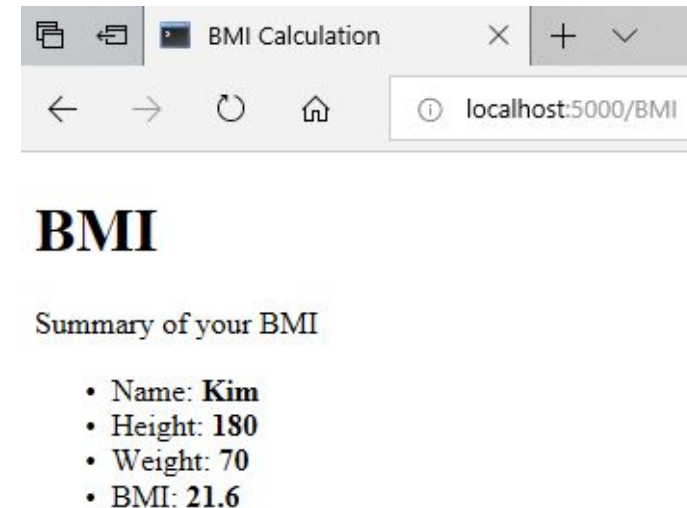
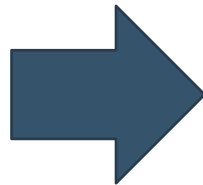
- Complete the Jason tutorial on https://www.w3schools.com/python/python_json.asp

Exercise #13

- Write a program that prompts the user to input their weight and height, and then outputs their BMI. To work out your BMI, divide your weight in kilograms by your height in meters squared ($BMI = w / h^2$). **사용자에게 몸무게, 키를 물어본 후 BMI를 출력하는 프로그램을 만드세요. BMI는 킬로그램 몸무게를 미터 키의 제곱으로 나누어서 계산하세요**



The screenshot shows a web browser window with the title 'BMI'. The address bar displays 'localhost:5000/input'. The page content includes the heading 'BMI Calculation' and three input fields: 'Enter your name:' with the value 'Kim', 'Enter your height:' with the value '180', and 'Enter your weight:' with the value '70'. Below these fields is a button labeled 'Calculate BMI!'.



The screenshot shows a web browser window with the title 'BMI Calculation'. The address bar displays 'localhost:5000/BMI'. The page content includes the heading 'BMI' and a section titled 'Summary of your BMI'. Below this section is a list of values: 'Name: Kim', 'Height: 180', 'Weight: 70', and 'BMI: 21.6'.

Exercise #13

```
<html>  
  <head>  
    { head content  
  </head>  
  <body>  
    { body content  
  </body>  
</html>
```

- Create the basic structure of HTML Document. **Html문서 기본틀 만들기**

Exercise #13

- Insert the following code inside the `<body>` tag

바디텍안에 다음
폼을 집어넣기

```
<form action=result method="post">

<p>

Enter your name:

<input type="text" name="name" /><br>

Enter your height:

<input type="text" name="height" /><br>

Enter your weight:

<input type="text" name="weight"><br>

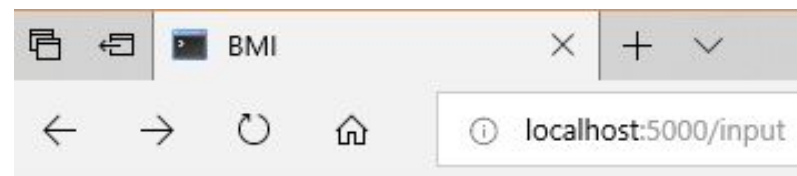
<input type="submit" value="Calculate BMI!" />

</p>

</form>
```

input.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
<title>BMI</title>
</head>
<body>
<h1>BMI Calculation</h1>
<form action=result method="post">
<p>
Enter your name: <input type="text" name="name" /><br />
Enter your height: <input type="text" name="height" /><br />
Enter your weight: <input type="text" name="weight"><br />
<input type="submit" value="Calculate BMI!" />
</p>
</form>
</body>
</html>
```



BMI Calculation

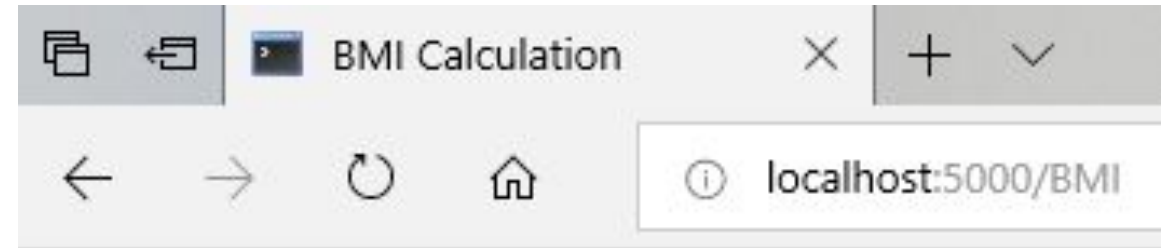
Enter your name:

Enter your height:

Enter your weight:

Exercise #13

- Create a html called result.html to display the result of BMI calculations **BMI계산 결과가 나타나는 result.html만들기**

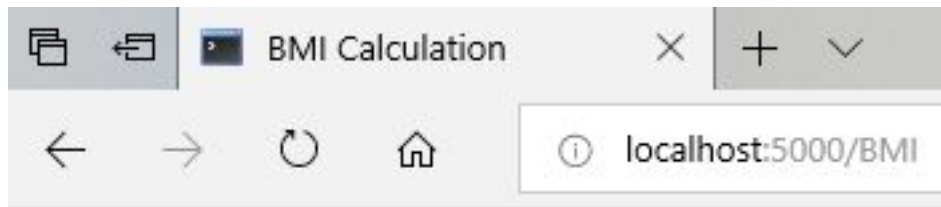


BMI

Summary of your BMI

- Name: **Kim**
- Height: **180**
- Weight: **70**
- BMI: **21.6**

Result.html



BMI

Summary of your BMI

- Name: **Kim**
- Height: **180**
- Weight: **70**
- BMI: **21.6**

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
<title>BMI Calculation</title>
</head>
<body>
<h1>BMI</h1>
<p>Summary of your BMI</p>
<ul>
<li>Name:<strong>{{name}}</strong></li>
<li>Height:<strong>{{height}}</strong></li>
<li>Weight:<strong>{{weight}}</strong></li>
<li>BMI:<strong>{{bmi}}</strong></li>
</ul>
</body>
</html>
```

Python Delimiters

- You can escape from HTML and execute the Python code using following tags
 - `{% ... %}` for Statements **html상에서 파이썬 명령어 실행**
 - `{{ ... }}` for Expressions to print to the output **결과물 출력**
 - `{# ... #}` for Comments not included in the output **주석**
 - `# ... ##` for Line Statements **한줄 명령어 실행**

Exercise #13

- Create a python file named app.py to requesting objects from the input page, calculate BMI, and passing parameters to the result page 입력페이지에서 오브젝트를 요청한후 BMI를 계산하고 그 결과를 결과페이지로 보내는 app.py 파일을 만드세요.

app.py

```
from flask import Flask  
app = Flask(__name__)
```

```
@app.route("/")  
def hello():  
    return "Hello World!"
```

```
if __name__ == "__main__":  
    app.run()
```

The route() function is a decorator, which tells the application which URL should call the associated function. 라우트와 함수를 연결하여 어플리케이션이 어디로 가야하는지 말해줌

Class name is equal to the name of the scope in which top-level code executes. 클래스의 이름이 스코프의 이름과 같을때 app이 실행

Rendering Template

```
from flask import Flask,  
render_template  
app = Flask(__name__)  
  
@app.route('/')  
def index():  
    return render_template('hello.html')  
  
if __name__ == '__main__':  
    app.run(debug = True)
```

hello.html

```
<!doctype html> <html>  
<body> <h1>Hello!</h1>  
</body>  
</html>
```


app.py

여러개의 값을 따로 따로 보내는 경우

```
from flask import Flask, render_template, request
app = Flask(__name__)

@app.route('/')
def input():
    return render_template('input.html')

@app.route('/BMI', methods=['POST', 'GET'])
def bmi():
    name = request.form.get('name')
    height = int(request.form.get('height'))
    weight = int(request.form.get('weight'))
    bmi = round(((weight/height)/height) * 10000, 2)
    return render_template("BMI.html", name = name, height=height,
weight=weight, bmi=bmi)

if __name__ == '__main__':
    app.run(debug = True)
```

Class name is equal to the name of the scope in which top-level code executes. 클래스이름이 탑레벨의 스코프이름과 같을때 실행

Don't need to restart manually for each change in the code 코드가 변할때마다 다시 시작할 필요없이 리로딩해줌

Requesting Objects and Passing Multiple Parameters 여러값을 한번에 보내는 경우

```
from flask import Flask, render_template, request
app = Flask(__name__)

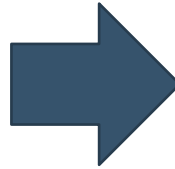
@app.route('/')
def student():
    return render_template('student.html')

@app.route('/result',methods = ['POST', 'GET'])
def result():
    if request.method == 'POST':
        result = request.form
        return render_template("result.html",result = result)

if __name__ == '__main__':
    app.run(debug = True)
```

Accepting Inputs and Displaying Parameters

```
<html>
  <body>
    <form action =
"http://localhost:5000/result" method
= "POST">
      <p>Name <input type = "text"
name = "Name" /></p>
      <p>Physics <input type =
"text" name = "Physics" /></p>
      <p>Chemistry <input type =
"text" name = "chemistry" /></p>
      <p>Maths <input type ="text"
name = "Mathematics" /></p>
      <p><input type = "submit"
value = "submit" /></p>
    </form>
  </body>
</html>
```



```
<!doctype html>
<html>
  <body>
    <table border = 1>
      {% for key, value in
result.items() %}
        <tr>
          <th> {{ key }} </th>
          <td> {{ value }} </td>
        </tr>
      {% endfor %}
    </table>
  </body>
</html>
```

Organizing Project

Starts up a development server and includes route definitions
서버를 실행, 라우팅정의

`app.py`

`config.py`

`requirements.txt`

`static/`

`templates/`

Contains most of the configuration variables that your app needs. 상태설정

Lists all of the Python packages that your app depends on. 파이썬 패키지의 리스트

CSS, JavaScript, images 스타일시트, 자바스크립트, 이미지들

Jinja2 templates and html files 템플릿과 html파일들

Passing Single Parameters and Displaying Parameters 한개값만 보내는 경우

```
from flask import Flask,
render_template

app = Flask(__name__)

@app.route('/hello/<user>')
def hello_name(user):

    return
    render_template('hello.html', name =
user)

if __name__ == '__main__':
    app.run(debug = True)
```

```
<!doctype html>

<html>

    <body>

        <h1>Hello {{ name }}!</h1>

    </body>

</html>
```

Exercise #13

- Complete the tutorials
 - https://www.tutorialspoint.com/flask/flask_templates.htm
 - https://www.tutorialspoint.com/flask/flask_sending_form_data_to_template.htm

Exercise #13

- Write a program for online counselor. It is okay that your counseling is not logically correct. Generate lists of answers and randomly give an advice for your customer. 온라인카운셀러 프로그램 만들기. 답을 리스트로 만든후 랜덤하게 고객에서 조언을 줌.

▪Output example

```
Good morning, I hope you are well today.  
What can I do for you?
```

```
>> My mother and I don't get along  
Why do you say that your mother and you don't get along
```

```
>> she always favors my sister  
You seem to think that she always favors your sister
```

```
>> my dad and I get along fine  
Can you explain why your dad and you get along fine
```

```
>> he helps me with my homework  
Please tell me more
```

```
>> quit  
Have a nice day!
```

Random Library

```
import random
```

```
random.randint(3, 9) #3에서 9사이에서 랜덤넘버
```

```
random.choice(["apple", "banana", "cherry"]) #한개를 랜덤하게 선택
```

```
random.sample(["apple", "banana", "cherry"], k=2) #2개를 랜덤하게 선택
```

```
random.random() #0에서 1사이의 랜덤넘버 (소수)
```

```
random.uniform(20, 60) #20과 60사이의 숫자를 공평하게 선택
```


Response Example

```
{
  "intents": [
    {
      "tag": "greeting",
      "patterns": ["Hi there", "How are you", "Is anyone there?", "Hey", "Hola", "Hello", "Good day"],
      "responses": ["Hello, thanks for asking", "Good to see you again", "Hi there, how can I help?"],
      "context": [""]
    },
    {
      "tag": "goodbye",
      "patterns": ["Bye", "See you later", "Goodbye", "Nice chatting to you, bye", "Till next time"],
      "responses": ["See you!", "Have a nice day", "Bye! Come back again soon."],
      "context": [""]
    },
    {
      "tag": "thanks",
      "patterns": ["Thanks", "Thank you", "That's helpful", "Awesome, thanks", "Thanks for helping me"],
      "responses": ["Happy to help!", "Any time!", "My pleasure"],
      "context": [""]
    },
    {
      "tag": "noanswer",
      "patterns": [],
      "responses": ["Sorry, can't understand you", "Please give me more info", "Not sure I understand"],
      "context": [""]
    }
  ]
}
```

Exercise #13

- The customer requests a program that computes a person's income tax. **고객의 세금을 계산해 주는 프로그램**

The tax laws are as follows:

- a flat tax rate of 20% **세금을 20프로**
- A \$10,000 standard deduction **기본공제 10,000불**
- An additional \$3,000 deduction for each dependents **부양자별 3000불 공제**

```
Enter the gross income: 150000.00
Enter the number of dependents: 3
The income tax is $26200.00
```

Exercise #13

Investment Program

Write a program that computes an investment report. 투자수익을 계산해 주는 프로그램

The inputs to this program are the following:

- An initial amount to be invested (a floating-point number) 초기 투자비용 (소수)
- A period of years (an integer) 기간 (정수)
- An interest rate (a percentage expressed as an integer) 이자율 (퍼센트)

Exercise #13

The program uses a simplified form of compound interest, in which the interest is computed once each year and added to the total amount invested. The output of the program is a report in tabular form that shows, for each year in the term of the investment, the year number, the initial balance in the account, the interest earned for that year, and the ending balance for that year. The columns of the table are suitably labeled with a header in the first row. Following the output of the table, the program prints the total amount of the investment balance and the total amount of interest earned for the period.

Output

The proposed user interface is shown in the figure below.

Enter the investment amount: *10000.00*

Enter the number of years: *5*

Enter the rate as a %: *5*

Year	Starting balance	Interest	Ending balance
1	10000.00	500.00	10500.00
2	10500.00	525.00	11025.00
3	11025.00	551.25	11576.25
4	11576.25	578.81	12155.06
5	12155.06	607.75	12762.82

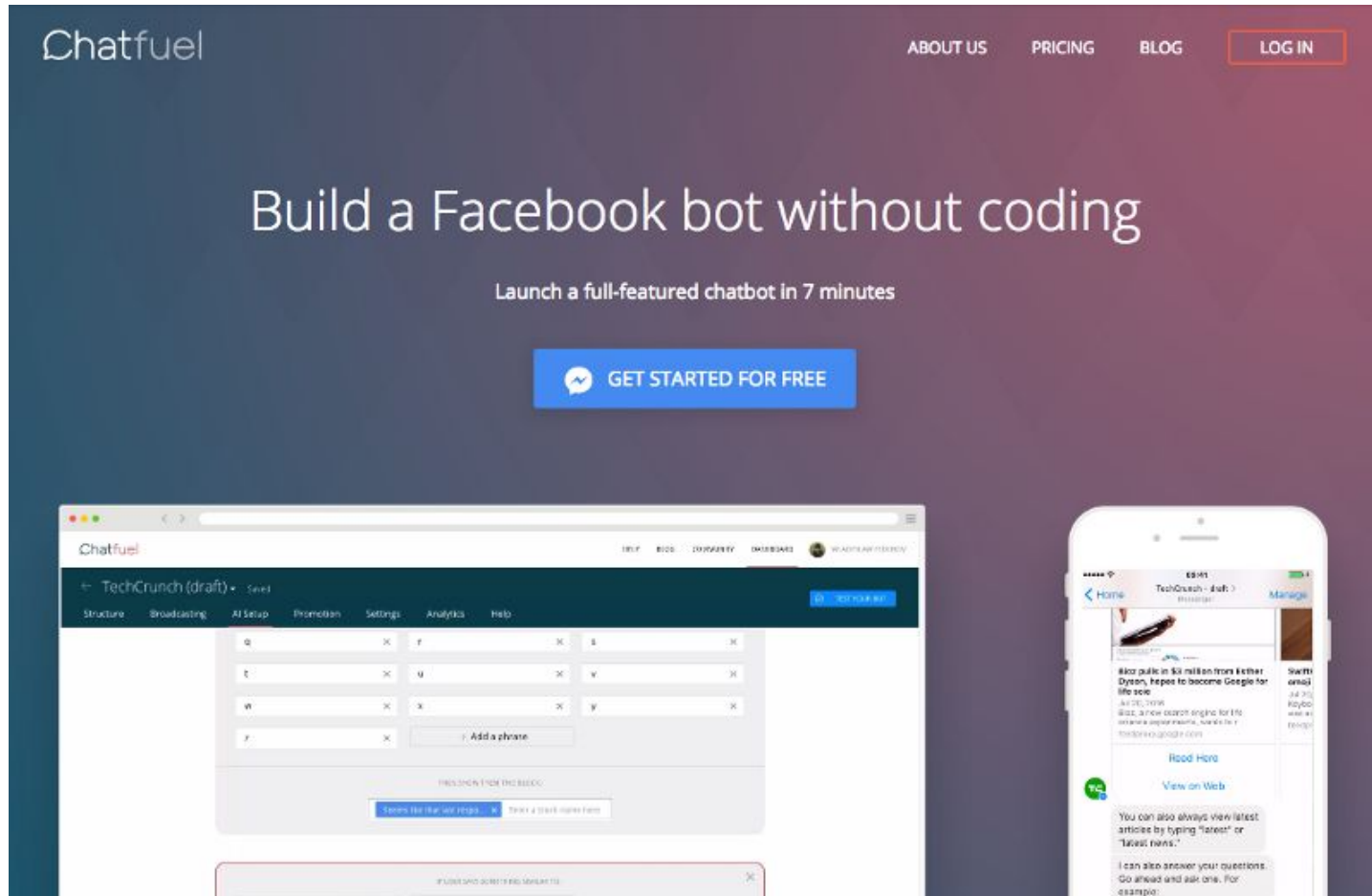
Ending balance: \$12762.82

Total interest earned: \$2762.82

Data Sets for Testing

Investment	Years	Rate
100.00	1	5
100.00	2	5
100.00	5	5
100.00	50	5
10000.00	1	5
10000.00	50	5
10000.00	50	20

Chatfuel



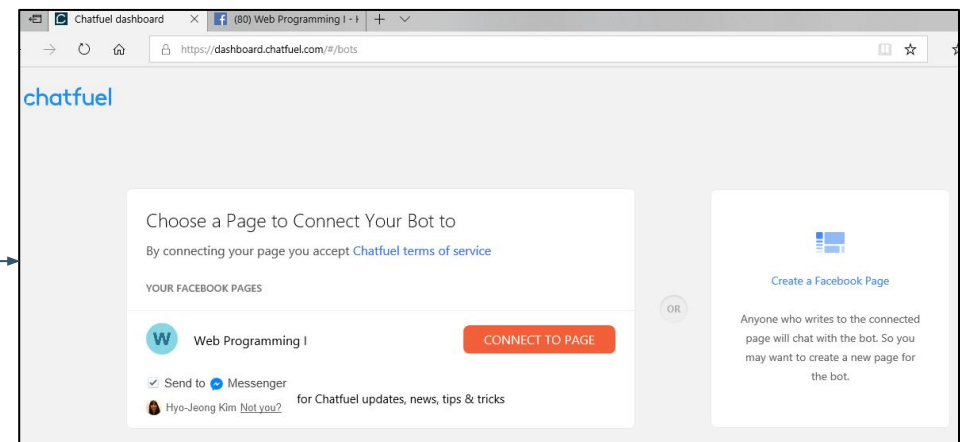
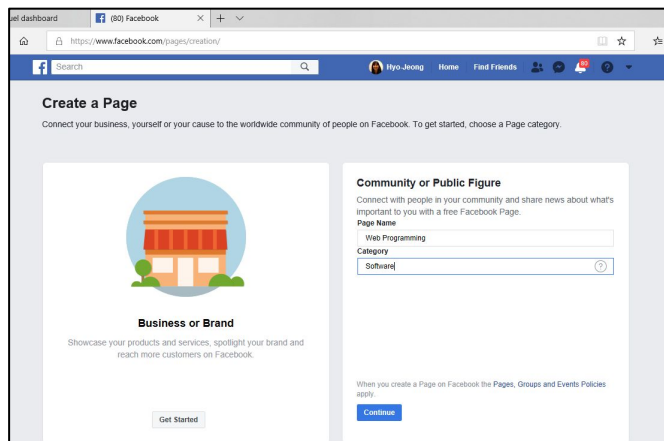
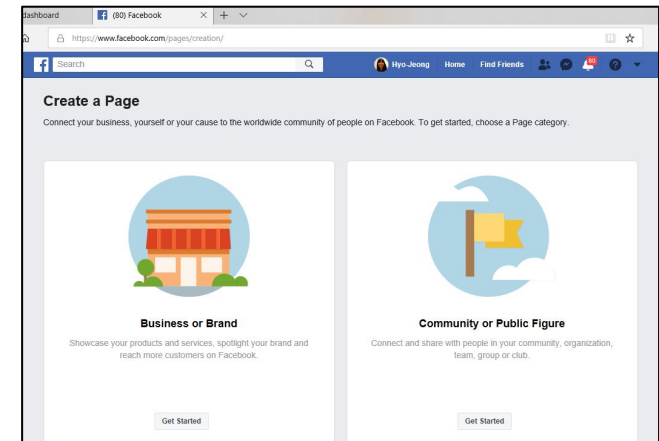
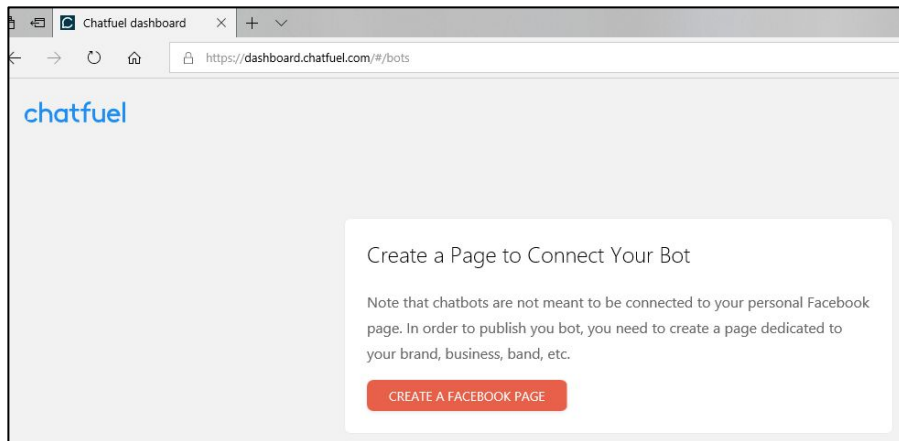
The image shows the Chatfuel website and its mobile app interface. The website has a dark blue header with the Chatfuel logo and navigation links: ABOUT US, PRICING, BLOG, and a LOG IN button. The main headline reads "Build a Facebook bot without coding" with a sub-headline "Launch a full-featured chatbot in 7 minutes". A prominent blue button says "GET STARTED FOR FREE". Below this, there are two screenshots: one of the Chatfuel web dashboard showing a "TechCrunch (draft)" chatbot configuration with a flowchart, and another of the Chatfuel mobile app showing a chatbot interface with a "TechCrunch" profile and a list of articles.

코딩없이
페이스북
메신저에 챗봇을
만들수 있음

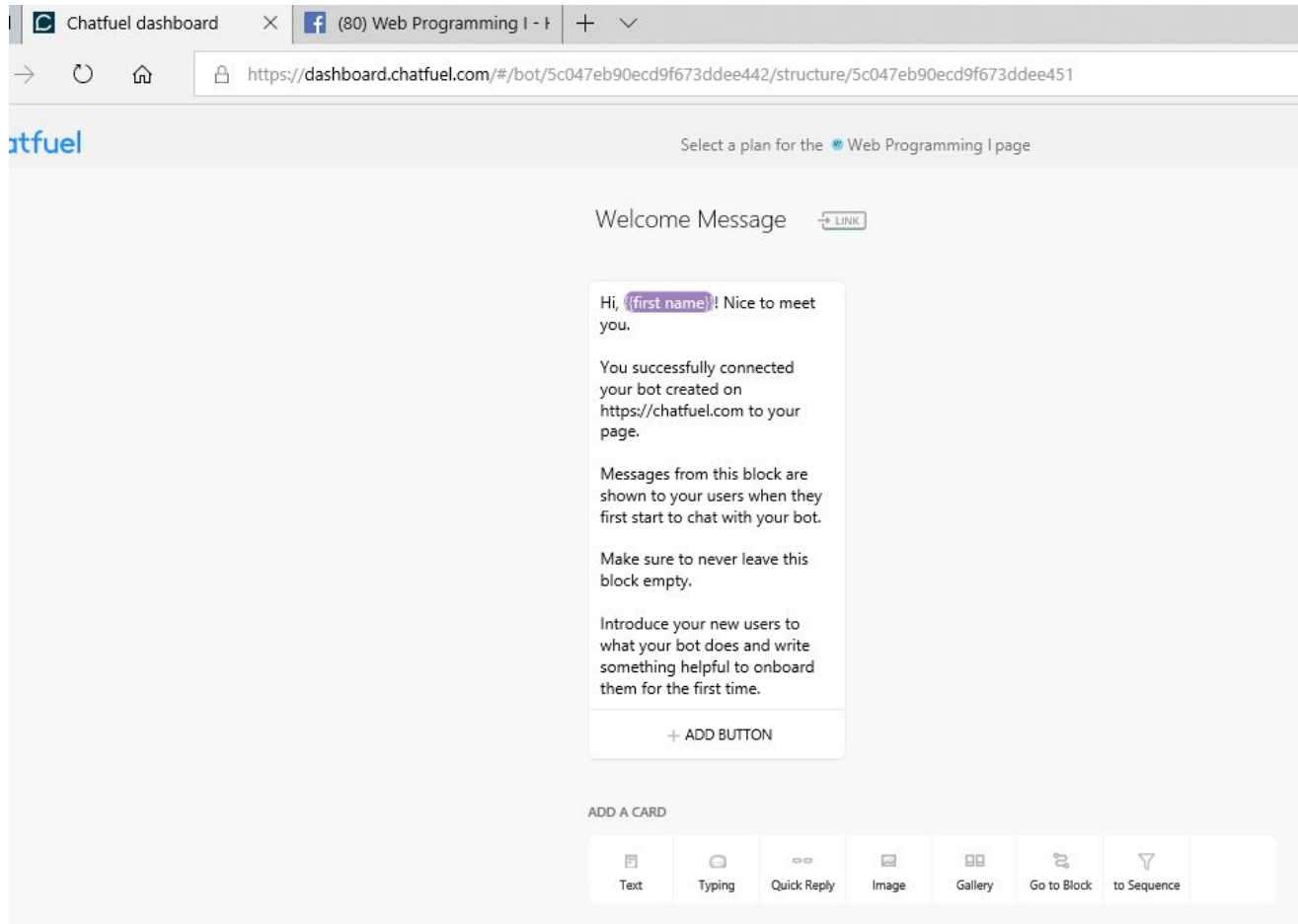
페이스북 메신저용 챗봇 만들기

- 페이스북 계정 생성
- 챗봇용 페이스북 페이지 1개 생성
- 페이스북 메신저 앱설치

Creating FaceBook Page

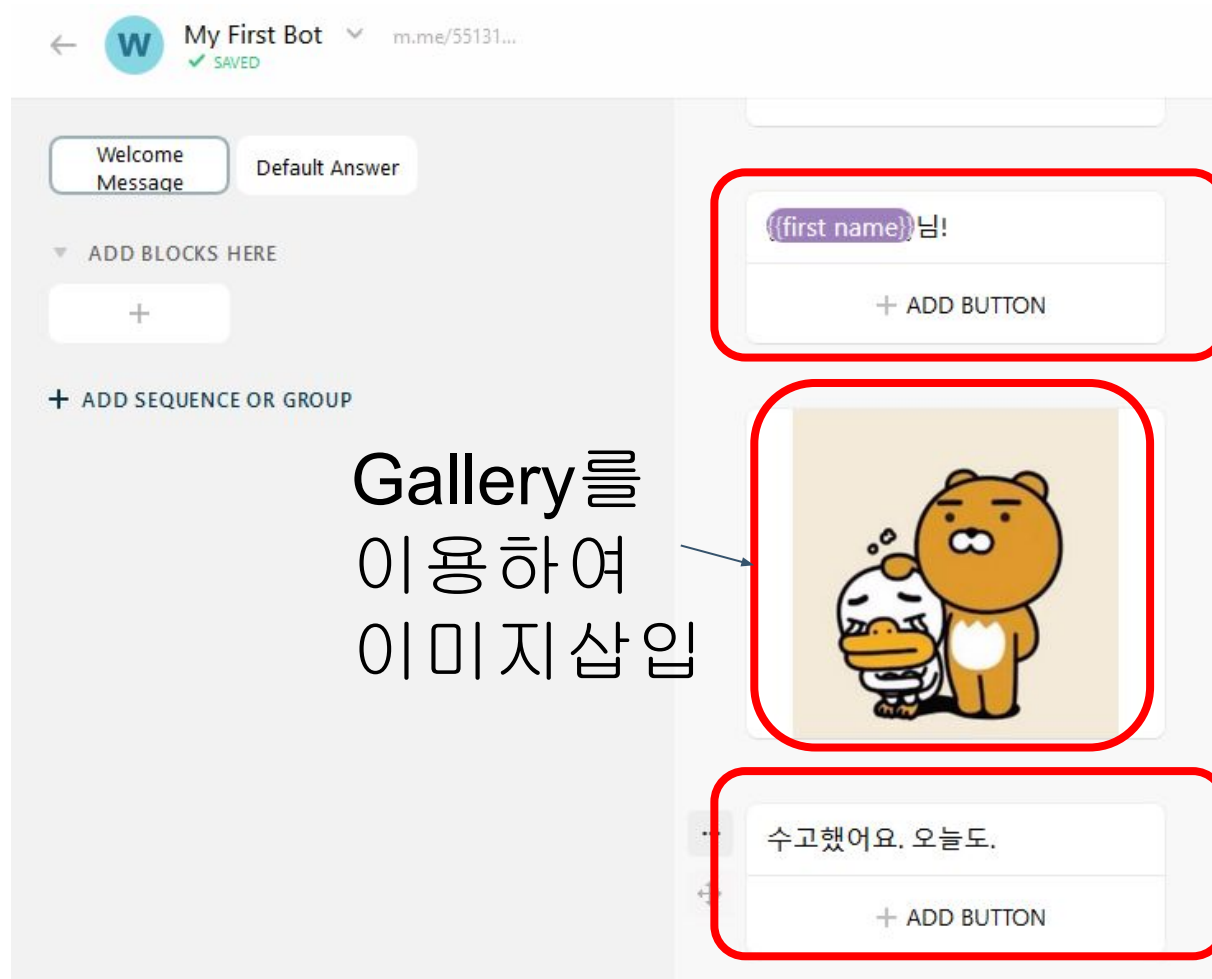


Welcome Message

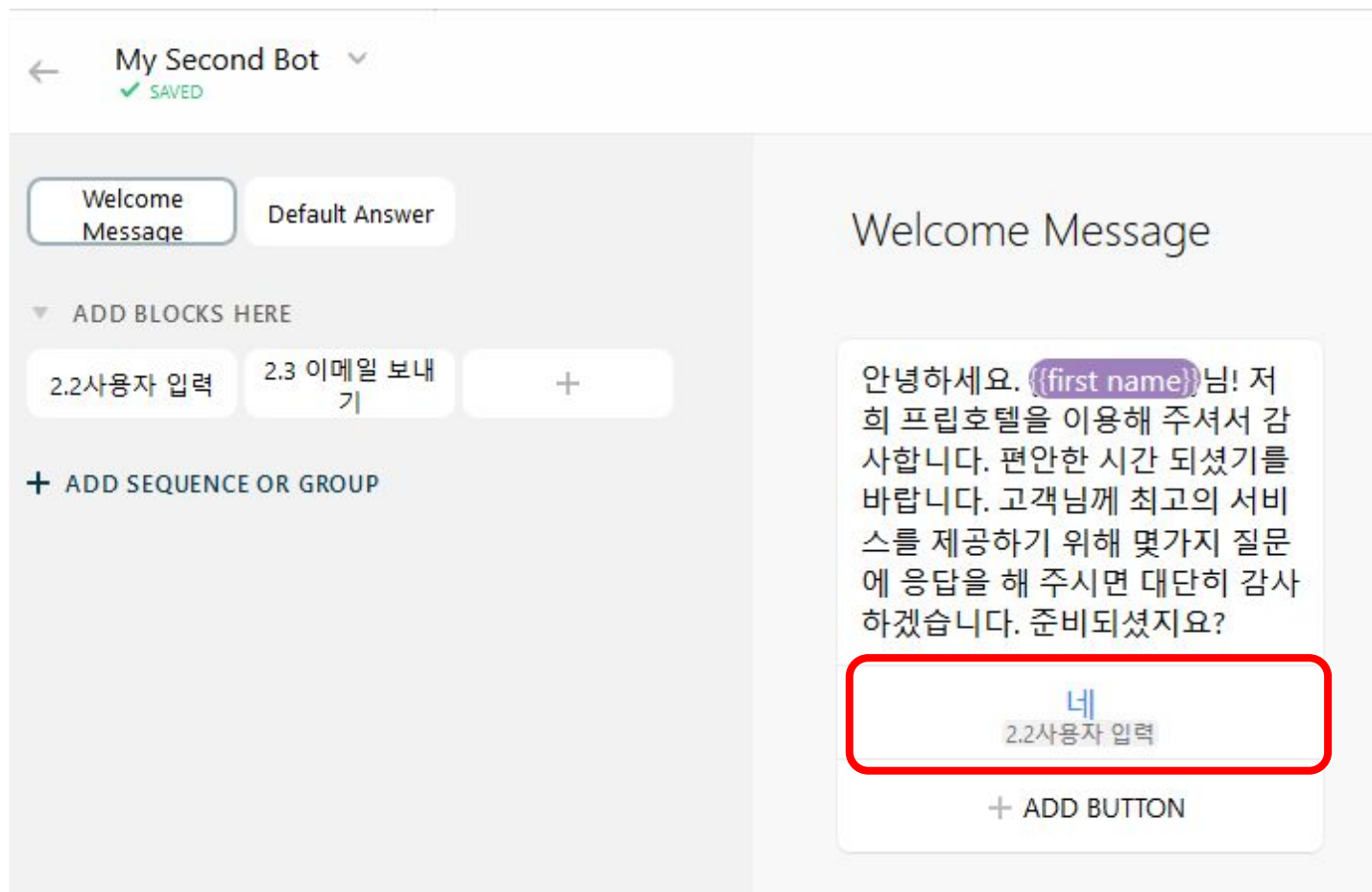


챗봇이랑 채팅시
처음뜨는 메세지

웰컴 메시지 만들기



Add Button



사용자 입력블락을
만들어서 버튼에 연결

User Input

← My Second Bot
SAVED

CONNECT TO FACEBOOK

TEST

Welcome Message

Default Answer

ADD BLOCKS HERE

2.2사용자 입력

2.3 이메일 보내기

+

ADD SEQUENCE OR GROUP

2.2사용자 입력

☰ User Input

?

Ask bot users questions and save their responses to user attributes. You can then utilize user attributes in broadcasting user filters, Go to Block plugin or export them via Send Email or JSON plugins.

MESSAGE TO USER *	VALIDATION	SAVE ANSWER TO ATTRIBUTE *
고객님이 머무시는 동안 저희가 제공해 드린 서비스가 전체적으로 어떠했나요?	None	{{feedback::review}} ✓
감사합니다. 저희가 제공한 서비스가 어떤 점을 더 개선하면 좋을까요? 구체적으로 말씀해 주시면 고맙겠습니다.	None	{{feedback::didnotlike}} ✓
감사합니다. 머무시는 기간 동안 가장 만족스러운 점이 있다면 어떤 것이 있었나요? 구체적으로 말씀해 주시면 고맙겠습니다.	None	{{feedback::likeit}} ✓

사용자의 대답을
변수에 저장

Send Email

← My Second Bot
✓ SAVED

CONNECT TO FACEBOOK

TEST T

Welcome Message

Default Answer

▼ ADD BLOCKS HERE

2.2사용자 입력

2.3 이메일 보내기

+

+ ADD SEQUENCE OR GROUP

✉ Send Email

Use this plugin to solicit user feedback or notifications, or export data from your chatbot. Note that you can't send emails to the bot's users with this plugin.

TITLE *

[CS봇]{{first name}} {{last name}} 님이 ✓

EMAIL ADDRESSES *

hyo-jeong.kim@hotmail.com ✓

EMAIL BODY *

CS봇에서 수집한 고객 설문 응답
=====

feedback:review = {{feedback::review}}

feedback:didnotlike = {{feedback::didnotlike}}

feedback:likeit = {{feedback::likeit}}

=====

저장했던
변수를
이용하여
이메일메세지
를 만든후
이메일보내기
블락을 만듬

Exercise #13

- Complete the chatbot tutorials on <https://medium.com/@feedbots/chatfuel-로-간단히-챗봇-만들기-1-71d009201e96> and <https://medium.com/@feedbots/chatfuel-로-이메일-저장하는-챗봇-만들기-9180cf7e0dcd>

Text Classification

N-Grams

- The process of combining the nearby words together for representation purposes where N represents the number of words to be combined together 옆에 있는 단어들을 함께 묶어주는 과정. N은 합칠 단어의 수
- Example
 - “Natural Language Processing is essential to Computer Science.”

- 1-gram or unigram
 - “Natural, Language, Processing, is, essential, to, Computer, Science”
- Bigram
 - “Natural Language, Language Processing, Processing is, is essential, essential to, to Computer, Computer Science”
- Trigram
 - “Natural Language Processing, Language Processing is, Processing is essential, is essential to, essential to Computer, to Computer Science”

N-Gram Code

```
vectorizer2 = CountVectorizer(analyzer='word',  
ngram_range=(2, 2))  
X2 = vectorizer2.fit_transform(corpus)  
print(vectorizer2.get_feature_names())  
print(X2.toarray())  
X2_df = pd.DataFrame(X2.toarray(),  
columns=vectorizer2.get_feature_names())  
X2_df.head(10)
```

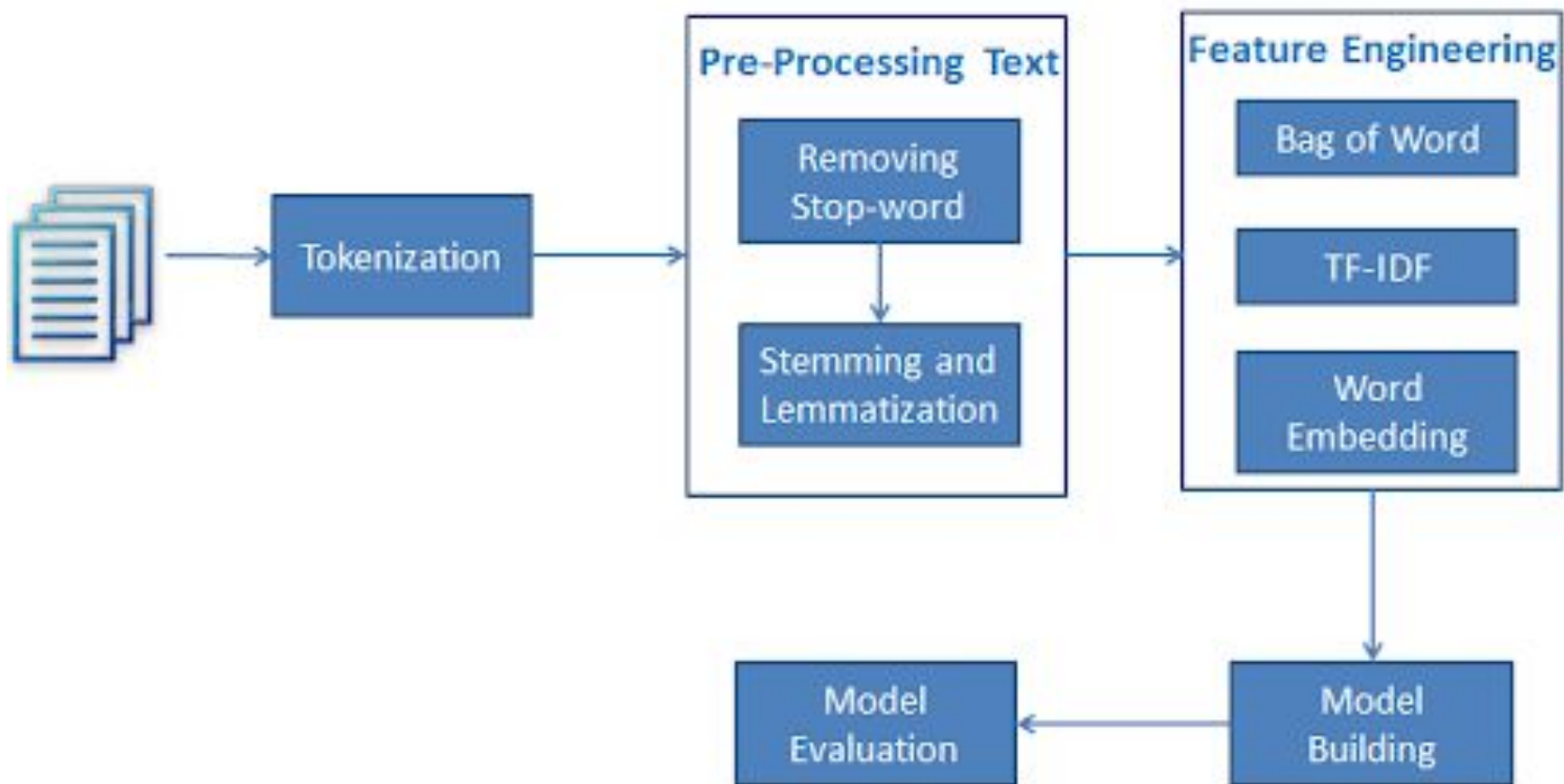
Stopwords and Lowercase

```
from sklearn.feature_extraction.text import  
CountVectorizer  
  
from nltk.tokenize import RegexpTokenizer  
  
token = RegexpTokenizer(r'[a-zA-Z0-9]+')  
  
cv =  
CountVectorizer(lowercase=True, stop_words='english', ngram  
_range = (1,1), tokenizer = token.tokenize)  
  
text_counts= cv.fit_transform(data['Phrase'])
```

RegexTokenizer

```
from nltk.tokenize import RegexTokenizer  
  
sentence = "Think and wonder, wonder and think."  
  
token = RegexTokenizer(r"\w+") #similar to tokens =  
re.split('\W+', text)  
  
new_words = token.tokenize(sentence)  
  
print(new_words)
```

Text Analytics Process



Word Embedding Example

```
docs = ['Well done!',  
        'Good work',  
        'Great effort',  
        'nice work',  
        'Excellent',  
        'Weak',  
        'Poor effort!',  
        'not good',  
        'poor work',  
        'Could have done better.']  
labels = np.array([1,1,1,1,1,0,0,0,0,0])
```

- One hot encoding (50 vocab size)
- Padding (4 maxlen, post)
- CNN model
 - Embedding (8 output vector size)
 - Flatten, 1 Dense with sigmoid
- Compile
- Fit (epochs = 50)
- Evaluate (accuracy)

Exercise #13

스팸분류 문제

- Download the SMS Spam Collection Data Set from UCI ML repository
스팸데이터셋을 다운로드
- Read and explore the dataset 데이터 읽고 탐색
- Count body_text by label and draw bar chart 스팸분류에 따른 body_text를 카운트해서 막대그래프

	label	body_text
0	ham	I've been searching for the right words to tha...
1	spam	Free entry in 2 a wkly comp to win FA Cup fina...
2	ham	Nah I don't think he goes to usf, he lives aro...
3	ham	Even my brother is not like to speak with me. ...
4	ham	I HAVE A DATE ON SUNDAY WITH WILL!!

Exercise #13

- X and y split (x – body_text, y-label, use only 50000 data!)
- Data preprocessing
 - Removing HTML tags and unwanted characters
 - Encoding (from `keras.preprocessing.text` import `one_hot`, use 20000 words)
 - Padding (from `keras.preprocessing.sequence` import `pad_sequences`, 400 post padding)

Removing HTML tags and Unwanted Characters

```
def clean_str(string):
    string = re.sub(r"\\", "", string)
    string = re.sub(r"\'", "", string)
    string = re.sub(r'\"', "", string)
    return string.strip().lower()

texts = []; labels = []

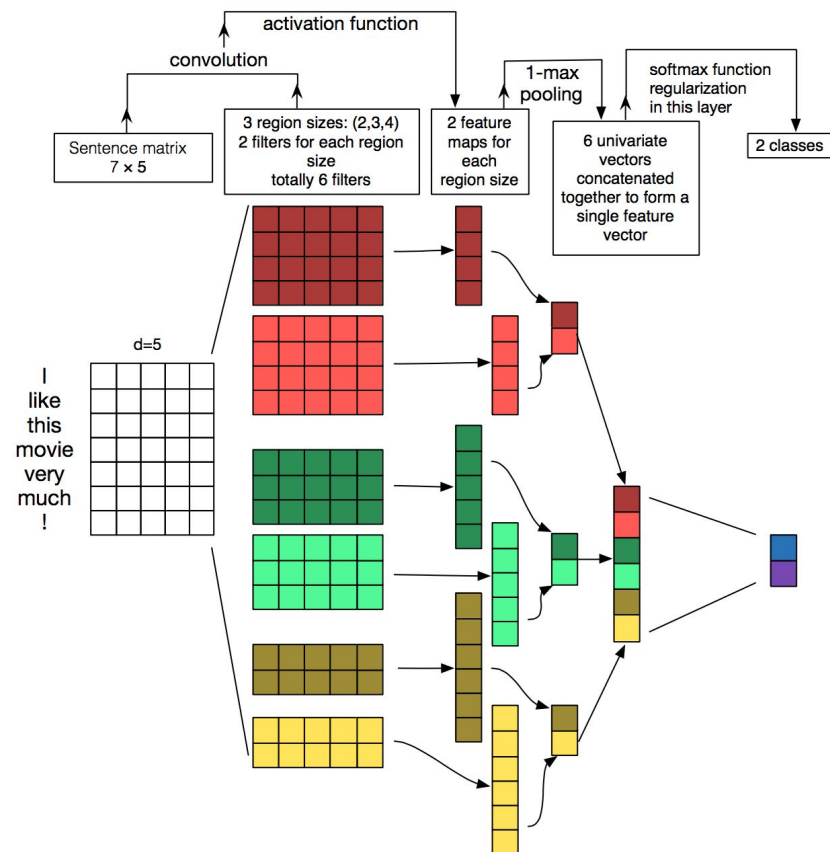
for i in range(df.message.shape[0]):
    text = BeautifulSoup(df.message[i])
    texts.append(clean_str(str(text.get_text().encode())))

for i in df['class']:
    labels.append(i)
```

Exercise #13

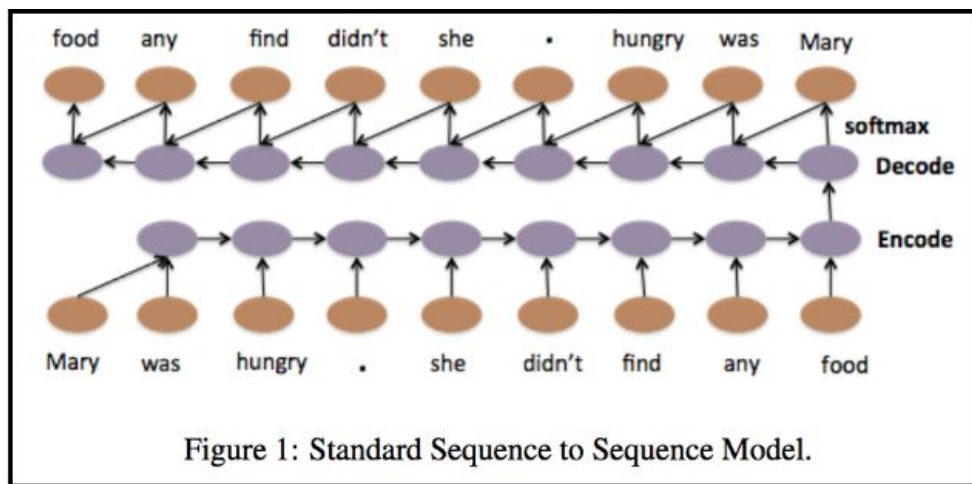
- Train and test split (.3 test set) 30프로의 테스트셋
- Model building
 - Embedding 50 output vectors, 1 Flatten, 1 Dense with sigmoid)
 - Embedding, Conv1D(128 units, 5 kernel, activation='relu'), MaxPooling1D(5), Conv1D(128 units, 5 kernel, activation='relu'), MaxPooling1D(5), Conv1D(128, 5, activation='relu'), MaxPooling1D(35), Flatten(), Dense(128, activation='relu'), Dense(1, activation='sigmoid')
 - Embedding, Bidirectional(LSTM(100)), Dense(1, activation='softmax')
- Compile (optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
- Evaluation (epochs=50)

Architecture of CNN Model



```
sequence_input = Input(shape=(MAX_SEQUENCE_LENGTH, ),
dtype='int32')
embedded_sequences = embedding_layer(sequence_input)
l_cov1= Conv1D(128, 5,
activation='relu')(embedded_sequences)
l_pool1 = MaxPooling1D(5)(l_cov1)
l_cov2 = Conv1D(128, 5, activation='relu')(l_pool1)
l_pool2 = MaxPooling1D(5)(l_cov2)
l_cov3 = Conv1D(128, 5, activation='relu')(l_pool2)
l_pool3 = MaxPooling1D(35)(l_cov3) # global max
pooling
l_flat = Flatten()(l_pool3)
l_dense = Dense(128, activation='relu')(l_flat)
preds = Dense(len(macronum),
activation='softmax')(l_dense)
```

Architecture of RNN Model



```
sequence_input =  
Input(shape=(MAX_SEQUENCE_LENGTH, ),  
dtype='int32')  
embedded_sequences =  
embedding_layer(sequence_input)  
l_lstm =  
Bidirectional(LSTM(100))(embedded_sequences)  
preds = Dense(len(macronum),  
activation='softmax')(l_lstm)  
model = Model(sequence_input, preds)  
model.compile(loss='categorical_crossentropy',  
optimizer='rmsprop',  
metrics=['acc'])
```

Exercise #13

- Model building with imdb

```
from keras.datasets import imdb
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=5000)
```

- Padding x_train, x_test (maxlen=400)

- CNN Model

- Embedding (embedding_dims = 50), Dropout(0.2), Flatten()
- Conv1D (250 filters, 3 kernel size, activation='relu'), MaxPooling1D()
- 250 Dense with activation='relu', Dropout(0.2)
- 1 Dense with sigmoid

- Compile and fit (epochs=2, batch=32)

- Predict and evaluation

Exercise #13

- Modeling building with yelp
 - X and y split
 - Encoding with Tokenizer (num_words=5000, vocab size = len(tokenizer.word_index) + 1) and padding (maxlen = 100, post)
 - Train and test split
 - Embedding (output_dim=50), Flatten, 10 Dense with relu, 1 Dense with sigmoid
 - Compile and fit (epochs = 10)
 - Evaluate (loss and accuracy)

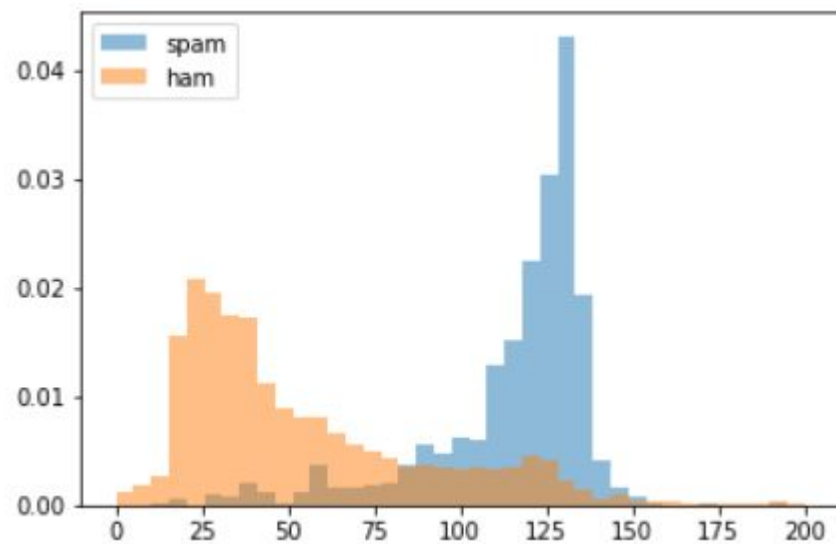
Exercise #13

스팸분류 문제

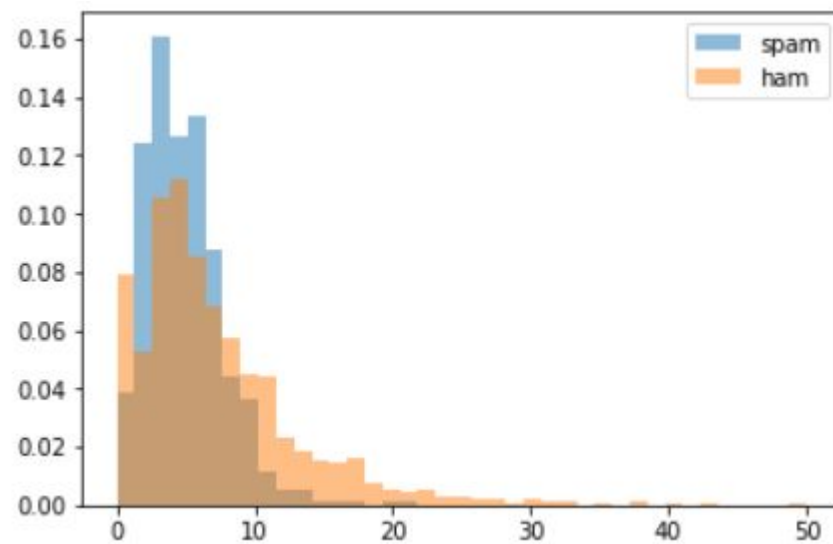
Preprocess data – punctuation, tokenize, stopwords, stemming, lemmatizer
전처리

- Vectorize data – bag of words, bi-grams, TF-IDF 단어의 벡터화
- Add new columns – length of text message, % of punctuation in the text
새로운 컬럼생성
- Draw the histogram of new columns respectively by labels 히스토그램

Visualizing Features of Text



message body length



% of punctuation

Exercise #13

스팸분류 문제

- To decide if the text is spam or not, build:
 - Bayesian Model
 - Random Forest Model
 - Logistic Regression Model
- Compare the results to that of CNN, RNN Models

Naive Bayes 나이트 베이지언

- A classification technique based on Bayes' Theorem with an assumption of independence among predictors.
독립변수들끼리 서로 독립적이라고 가정하는
베이지언 이론에 기초한 분류기술
- Easy to build and particularly useful for very large data sets
데이터가 많을때 유용

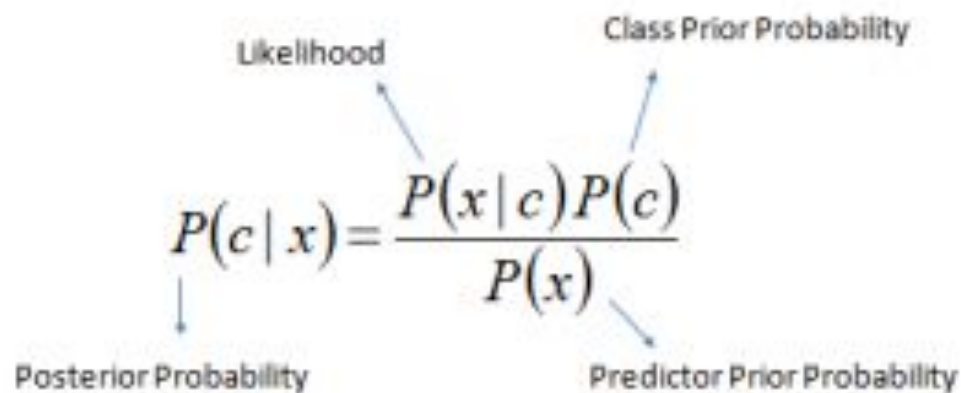
Naive Bayes 나이브 베이지언

- Outperforms even highly sophisticated classification methods
복잡한 분류문제에서 잘 작동함.
- Mostly used in text classification and with problems having multiple classes
클래스가 많은 텍스트분류 문제에 주로 사용됨
- Assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature
속성들끼리 서로 연관성이 없다고 가정함

Example

- A fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple 사과가 사과라고 인식되기 위해서는 빨간색, 둥근 모양, 3인치의 직경을 가져야 됩니다. 각 속성들은 서로 관련이 있고, 다른 속성에 연관되어 있지만, 각각의 속성은 이 과일이 사과라고 인식되는 확률에 독립적으로 영향을 미친다고 가정합니다.
- $P(\text{사과}|\text{빨강})$, $P(\text{사과}|\text{둥근모양})$, $P(\text{사과}|\text{3인치직경})$

Posterior Probability



The diagram shows the formula for Posterior Probability: $P(c|x) = \frac{P(x|c)P(c)}{P(x)}$. Arrows point from the following labels to the corresponding parts of the formula: 'Likelihood' points to $P(x|c)$, 'Class Prior Probability' points to $P(c)$, 'Posterior Probability' points to $P(c|x)$, and 'Predictor Prior Probability' points to $P(x)$.

$$P(c | X) = P(x_1 | c) \times P(x_2 | c) \times \cdots \times P(x_n | c) \times P(c)$$

- $P(c|x)$ is the posterior probability of *class* (c , *target*) given *predictor* (x , *attributes*).
주어진 조건에서 그 클래스가 될 확률
- $P(x|c)$ is the likelihood which is the probability of *predictor* given *class*. 그 클래스에서 그 조건이 나올 확률
- $P(c)$ is the prior probability of *class*. 그 클래스가 나왔던 확률
- $P(x)$ is the prior probability of *predictor*. 전에 그 조건이 나온 확률

Text Classification

- Works particularly well with natural language processing (NLP) problems 언어처리에 유용함

Text	Tag
"A great game"	Sports
"The election was over"	Not sports
"Very clean match"	Sports
"A clean but forgettable game"	Sports
"It was a close election"	Not sports



Word	P(word Sports)	P(word Not Sports)
a	$\frac{2+1}{11+14}$	$\frac{1+1}{9+14}$
very	$\frac{1+1}{11+14}$	$\frac{0+1}{9+14}$
close	$\frac{0+1}{11+14}$	$\frac{1+1}{9+14}$
game	$\frac{2+1}{11+14}$	$\frac{0+1}{9+14}$

$$\begin{aligned}
 &P(a|Sports) \times P(very|Sports) \times P(close|Sports) \times P(game|Sports) \times P(Sports) \\
 &= 2.76 \times 10^{-5} \\
 &= 0.0000276
 \end{aligned}$$

$$\begin{aligned}
 &P(a|Not Sports) \times P(very|Not Sports) \times P(close|Not Sports) \times P(game|Not Sports) \times P(Not Sports) \\
 &= 0.572 \times 10^{-5} \\
 &= 0.00000572
 \end{aligned}$$

$$P(sports|a\ very\ close\ game) = \frac{P(a\ very\ close\ game|sports) \times P(sports)}{P(a\ very\ close\ game)}$$

Algorithm Example

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency Table		
Weather	No	Yes
Overcast		4
Rainy	3	2
Sunny	2	3
Grand Total	5	9

Step 1: Convert the data set into a frequency table 현재있는 데이터로 빈도테이블을 만듦

Step 2: Create Likelihood table

확률로 전환하여 가능성테이블을 만듦

Likelihood table				
Weather	No	Yes		
Overcast		4	$=4/14$	0.29
Rainy	3	2	$=5/14$	0.36
Sunny	2	3	$=5/14$	0.36
All	5	9		
	$=5/14$	$=9/14$		
	0.36	0.64		

Step 3: Use Naive Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction 각 클래스에 후속확률을 구함. 높은 후속확률을 갖는 클래스가 예측값이 됨

Example

- Will players play if weather is sunny 해가 난 날 나가서 놀 확률은??
 - $P(\text{Yes} \mid \text{Sunny}) = P(\text{Sunny} \mid \text{Yes}) * P(\text{Yes}) / P(\text{Sunny})$
 - $P(\text{Sunny} \mid \text{Yes}) = 3/9 = 0.33$ 나가서 논 중에 해가 난 경우
 - $P(\text{Yes}) = 9/14 = 0.64$ 나가서 논 경우
 - $P(\text{Sunny}) = 5/14 = 0.36$ 해가 난 경우
 - $P(\text{Yes} \mid \text{Sunny}) = 0.33 * 0.64 / 0.36 = 0.60$, which has higher probability **60%의 확률이 있음**

Three Types of Naive Bayes Model

- Gaussian - used in classification and it assumes that features follow a normal distribution 변수들이 정규분포를 따를때
- Multinomial - used for discrete counts 별개의 것을 셀때
- Bernoulli - useful if your feature vectors are binary (i.e. zeros and ones) 속성이 두가지 값만 가질때

Importing Library

```
#Import Library of Gaussian Naive Bayes model  
from sklearn.naive_bayes import GaussianNB, BernoulliNB,  
MultinomialNB
```

Building GaussianNB Model

```
#Create a Gaussian Classifier
gnb = GaussianNB()

# Train the model using the training sets
gnb.fit(x, y)
```

Building BernoulliNB Model

```
#Create a BernoulliNB Classifier  
bnb = BernoulliNB()  
  
# Train the model using the training sets  
bnb.fit(x, y)
```

Building MultinomialNB Model

```
#Create a MultinomialNB Classifier  
mnb = MultinomialNB()  
  
# Train the model using the training sets  
mnb.fit(x, y)
```

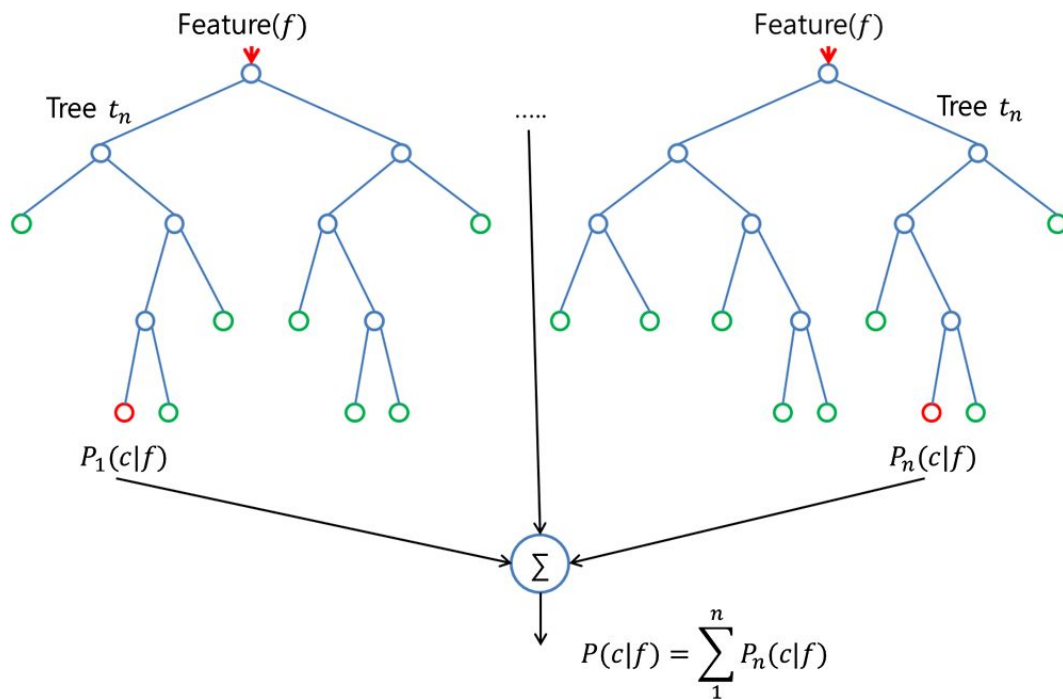
Predicting Output

```
#Predict Output  
y_pred = gnb.predict(x_test)  
print(y_pred)
```

Evaluating Output

```
accuracy_score(y, gnb.predict(x)) #train performance  
accuracy_score(y_test, y_pred) #test performance  
confusion_matrix(y_test, y_pred) #test performance  
print(classification_report(y_test, y_pred))
```

Random Forest



- A collection of classification tree
분류트리의 집합
- An ensemble tool which takes a subset of observations and a subset of variables to build a decision trees
관측치와 변수의 부분집합을 모아서 의사결정트리를 만드는 앙상블기법

Random Forest Model

- Builds multiple decision tree and amalgamate them together to get a more accurate and stable prediction

```
from sklearn.ensemble import RandomForestClassifier  
rfc = RandomForestClassifier(random_state=0)  
rfc.fit(x_train, y_train)  
rfc.predict(x_test)
```

n_estimators - number of
trees to be grown in the
forest 트리의 갯수

Model Evaluation

```
accuracy_score(y_test, rfc.predict(x_test)) #accuracy for  
test data
```

```
cross_val_score(rfc, x_train, y_train, cv=5,  
scoring='accuracy')
```

GridSearchCV

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

rf = RandomForestClassifier()
param = {'n_estimators': [10, 150, 300],
         'max_depth': [30, 60, 90, None]}

gs = GridSearchCV(rf, param, cv=5, n_jobs=-1)# n_jobs=-1 for parallelizing search
gs_fit = gs.fit(X_count_feat, data['label'])
pd.DataFrame(gs_fit.cv_results_).sort_values('mean_test_score', ascending=False).head()
```

Logistic Model

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression()
classifier.fit(x_train, y_train)
score = classifier.score(x_test, y_test)
print("Accuracy:", score)
```

Decision Tree

```
from sklearn.tree import DecisionTreeClassifier  
  
dtc = DecisionTreeClassifier(random_state=0)  
  
dtc.fit(x_train, y_train)  
  
dtc.predict(x_test)
```

You can also import DecisionTreeRegressor from sklearn.tree if you want to use a decision tree to predict a numerical target variable. 연속변수에 대한 예측인 경우는 리그레서를 사용하면 됨

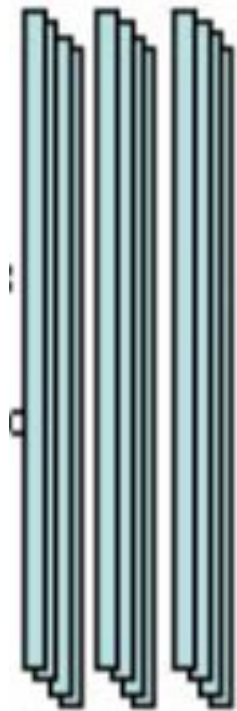
Model Evaluation for Decision Tree

```
from sklearn.metrics import accuracy_score  
accuracy_score(y_test, dtc.predict(x_test)) #9.0  
cross_val_score(dtc, x_train, y_train, cv=5,  
scoring='accuracy')
```

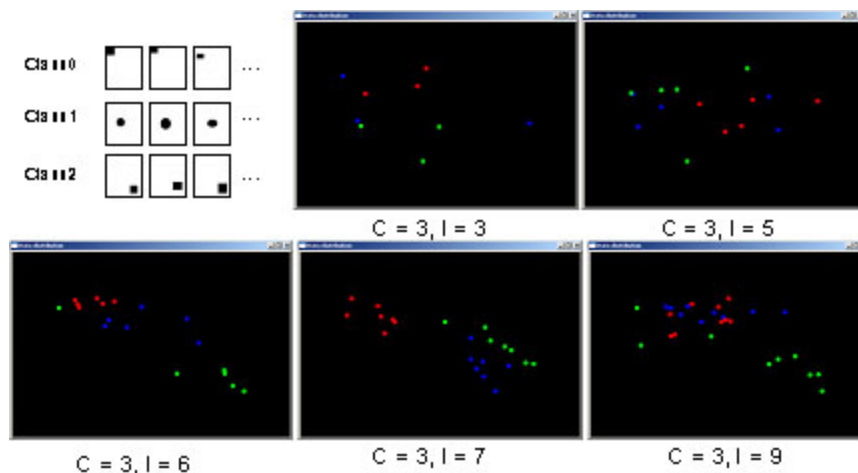
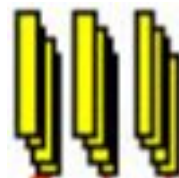
Linear Discriminant Analysis (LDA)

- A well-known scheme for feature extraction and dimension reduction
속성추출, 차원축소에 사용
- Widely used in high dimensional data such as face recognition and image retrieval
얼굴인지, 이미지검색

Training Set

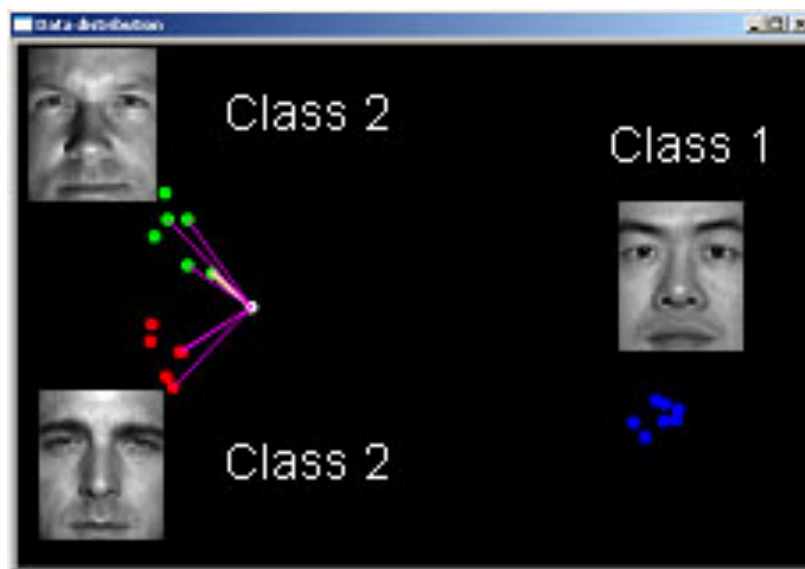


Dimension
Reduction



3-Class/ 2-Dimension Map

Testing Image



Applying Weighted K-Nearst Neighborhood

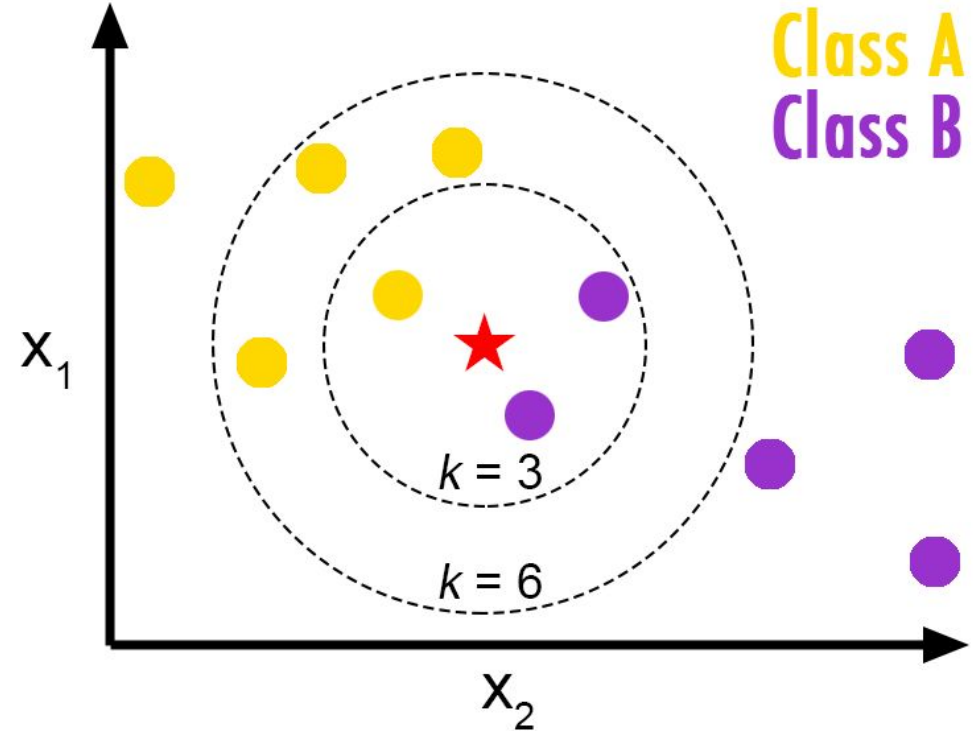
Code for LDA

- Find a linear combination of features that separates two or more classes of objects
두개 이상의 클래스들을 구별해주는 속성들의 조합

```
from sklearn.discriminant_analysis import  
LinearDiscriminantAnalysis  
lda = LinearDiscriminantAnalysis()  
lda.fit(x_train, y_train)
```

k-Nearest Neighbors (kNN)

- The case is assigned to the class that most common amongst its K nearest neighbors measured by a distance function
거리에 따라 가까운 k 개의 이웃들을 정하고 그 이웃들의 클래스를 따름
- The distance functions can be Euclidean, Manhattan, Minkowski (for continuous variables) and Hamming distance (for categorical variables)

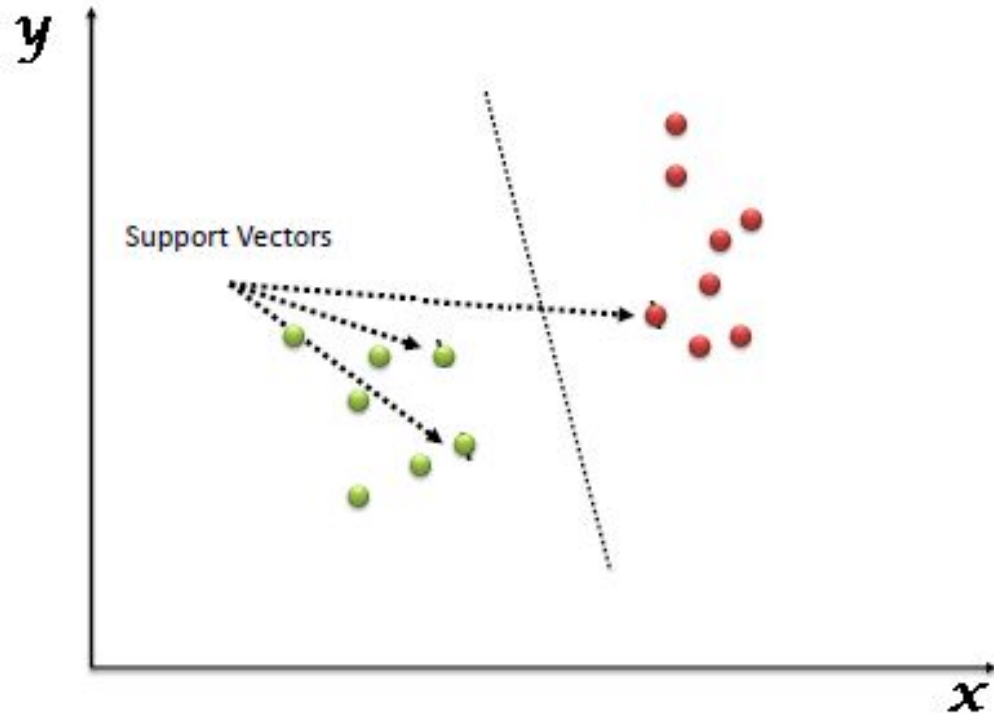


Code for kNN

- Used for both classification and regression problems. 분류문제, 회귀문제에 다 사용됨

```
from sklearn.neighbors import KNeighborsClassifier  
knn = KNeighborsClassifier()  
knn.fit(x_train, y_train)
```

Support Vector Machines (SVM)



- Plot each data item as a point in n -dimensional space where n is number of features you have.
변수들의 공간안에 모든 데이터포인트들을 그림
- Then, perform classification by finding the hyper-plane that maximize the distances between nearest data point (margin)
가장 가까운 데이터포인트들의 거리를 최대화시킬수 있는 하이퍼플레인을 찾아 분류하는 방법

Code for SVM

- A supervised machine learning algorithm which can be used for both classification or regression challenges

```
from sklearn.svm import SVC  
svm = SVC()  
svm.fit(x_train, y_train)
```

Model Evaluation Summary

- Compare the models to each other and select the most accurate.

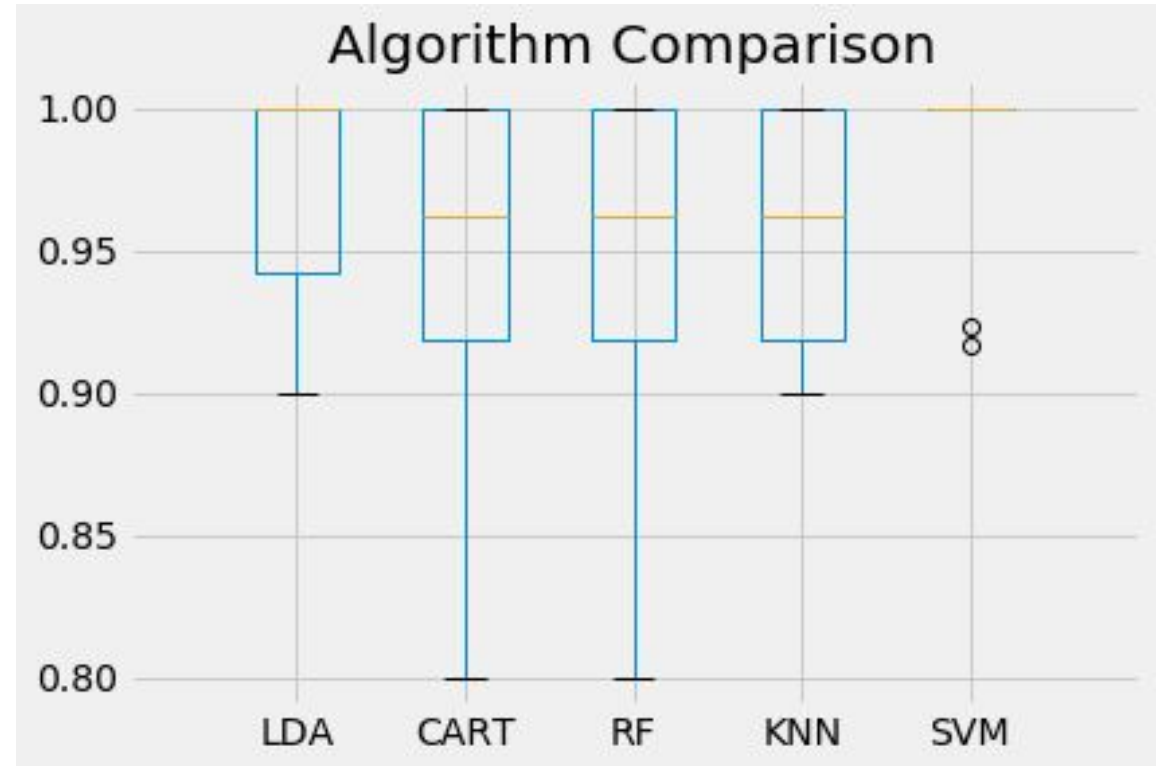
Model Accuracy Estimation

Model	CV Mean	CV Standard Deviation
lda	0.973974358974359	0.04011017863311351
dtree	0.9381002331002332	0.07315791398631202
rf	0.9471911421911422	0.06257433351640862
knn	0.9571911421911421	0.04326342995186461
svm	0.9839743589743589	0.032083317323697944

Algorithm Comparison

- Many samples achieving 100% accuracy.

```
plt.boxplot([lda_score, dtc_score,  
rfc_score,  
knn_score,  
svm_score])
```



Exercise #13

- Load the iris dataset and create a data frame named df (using data and target)
아이리스 데이터셋을 가져와서 데이터프레임을 만드세요
- Summarize the data frame 데이터 요약
 - head(20), shape, dtypes, describe(), df.groupby('Species').size()
- Data visualization 데이터 시각화
 - Univariate plot – boxplot, histogram
 - Multivariate plot – scatter_matrix

Exercise #13

- IV and DV Split 독립변수, 종속변수로 나누기
- Splitting Dataset (test size=.2) 훈련, 테스트 데이터로 나누기
- Linear Regression 회귀모델
 - Model evaluation (r2 score)
 - Cross validation (5 fold)
- Logistic Regression 로지스틱회귀
 - Model evaluation (accuracy score)
 - Cross validation (5 fold)

Exercise #13

- Decision Tree
 - Model evaluation (accuracy score)
 - Cross validation (5 fold)
 - Decision tree visualization (max depth=3)
- Random Forest
 - Model evaluation (accuracy score)
 - Cross validation (5 fold)

Variable Descriptions for Loan Prediction

	Variable	Description
non-numeric	Loan_ID	Unique Loan ID
	Gender	Male/ Female
	Married	Applicant married (Y/N)
	Dependents	Number of dependents
	Education	Applicant Education (Graduate/ Under Graduate)
	Self_Employed	Self employed (Y/N)
numeric	ApplicantIncome	Applicant income
	CoapplicantIncome	Coapplicant income
	LoanAmount	Loan amount in thousands
	Loan_Amount_Term	Term of loan in months
	Credit_History	credit history meets guidelines
	Property_Area	Urban/ Semi Urban/ Rural
	Loan_Status	Loan approved (Y/N)

Exercise #13

- Read loan.csv file and do some basic exploration

`shape, dtypes, describe...`

- Check missing values and fill them with mode or median values

`df.isna().sum() or df.`

`specific_column.value_counts(dropna=False)`

`df.specific_column.fillna(df.specific_column.mode()[0],
inplace=True)`

`df.specific_column.fillna(df.specific_column.median(),
inplace=True)`

Exercise #13

- Encode non-numeric values
 - Gender, Married □ get_dummies
 - df.rename(columns={'Male':'Gender'}, inplace=True)
 - df.replace({1:0, 0:1}, inplace=True)
 - Dependents, Education, Self_Employed, Property_Area, Loan_Status □ label encoder

Exercise #13

- Use loan prediction data and splitting Dataset (decide the test size)
- Logistic Regression
 - Model evaluation (r2 score)
 - Cross validation (decide the fold)
- Decision Tree
 - Model evaluation (r2 score, accuracy score)
 - Cross validation (decide the fold)
 - Decision tree visualization (decide the max depth)
- Random Forest
 - Model evaluation (r2 score, accuracy score)
 - Cross validation (decide the fold)