졸업과제 중간 보고서

멀티모달 기반 스팸 필터링 플랫폼 개발

지도교수	최윤호
팀명	멀티모발
이름	**201824534** 윤상호
	201824636 이강우
	202055651 조재홍

[목차]

1. 과제 목표

본 졸업과제는 멀티모달 딥러닝/머신러닝 기반 스팸 필터링 플랫폼 개발을 목표로한다.

- 수집 텍스트 스팸 데이터(kaggle, 메일함), 생성 텍스트 스팸 데이터를 모달리티로 활용한다.
- 해당 모달리티들을 기반으로 스팸 필터링 모델을 학습시키고 학습된 모델로 스팸메일 여부를 판별한다.
- 시각화 인터페이스를 통해 결과(스팸메일 여부, 필터링 된 이유 등 스팸 분류에 유의미한 정보)를 확인한다.
- 새로운 유형의 스팸 메일에 대한 대처에 유연한 스팸 필터링 모델을 개발한다.

2. 제약사항 분석

2.1 멀티모달 활용 제약사항

(1) 수집 데이터

- 스팸이미지의 데이터가 공개 데이터 저장소나 인터넷 상에 업로드 되어 있는 것이 거의 없어서 이미지 데이터 셋 수집에 어려움이 있다.
- 한글로 이루어진 텍스트 스팸메일 데이터가 공개 데이터 저장소나 인터넷 상에 업로드 되어 있는 것이 거의 없어서 데이터 셋 수집에 어려움이 있다.

(2) 생성 데이터

- 생성형 AI서비스는 스팸메일 생성과 같은 불법적인 행위를 제공하지 않기 때문에 일반적인 방법으로 생성 데이터 셋을 형성하는데 어려움이 있다.

(3) 이미지 데이터

- 영어로 공개된 스팸 이미지 데이터 셋이 거의 없어서 활용하는데 어려움이 있다.

2.2 데이터 수집 및 전처리 제약사항

(1) 데이터 수집

- 스팸 이메일의 내용과 작동 방식이 시간에 따라 변하므로 지금 작동하는 기법이 이러한 스팸 이메일의 구조와 내용의 변화로 인해 가까운 미래에 쓸모없어질 수 있는 Concept Drift 현상을 방지하기 위해 Kaggle 데이터 저장소에서 다양한 시점의 데이터를 수집하려 하였으나 동일한 데이터 셋을 서로 다른 시점에 업로드 한 경우가 많다.
- 공개 데이터 저장소에서 수집한 영문 텍스트 스팸메일 데이터 셋의 인코딩 과정에서 특수문자가 깨지는 현상이 발생했다.
- Kaggle을 제외한 다른 경로로 데이터를 수집하고자 했는데, kaggle 외에는 우리가 사용할 수 있는 메일형태로 업로드되어 있는 것이 거의 없었다.

(2) 데이터 전처리

- 기존에 계획한 전처리 방법은 텍스트를 토큰화하여 불용어 처리 한 뒤 빈도수가 높은 단어들만 임베딩 벡터로 만들어서 모델학습에 사용하려 하였으나, 해당 방식으로 진행하면 규칙기반의 스팸 필터링 모델과 동일한 특징(입력 feature가 고정되어 새로운 유형의 데이터에 대한 대처가 어려움)을 가지게 되어 딥러닝/머신러닝 기반의 스팸 필터링 모델의 특징이 소실되는 문제점이 발생했다.
- content-based 모델을 선정했기에, 메일 내용 그 자체를 입력으로 넣는것 외에도 스팸 메일을 필터링 하는데에 중요하다고 생각되는 feature 값들을 선정하는데 있어서 입력 feature를 7개만 테스트한다고 가정하더라도 5040가지의 경우의 수를 검토하여 성능을 비교해야하는 현실적인 제약사항이 있다.

2.3 모델 선정 및 학습 제약사항

(1) MLP(Multi-Layer Perceptron)

- MLP 모델의 구축과정에서 은닉층의 수, 뉴런 개수, 학습률, 활성화 함수 등 다양한 하이퍼파라미터가 존재하는데 최적의 성능을 위해 모든 경우의 수를 다 실험하기에는 시간적인 제약사항이 있다.

(2) LSTM(Long Short Term Memory)

- 본 과제에서는 Keras의 Function API인 LSTM 모델을 활용하였다. 이 때 학습에 중요한 영향을 미치는 파라미터로 활성화 함수, 손실 함수, 최적화 함수 부분이 있는데 성능 비교를 위해 검증 해보기에는 총 17가지 파라미터를 조합해야하는데 이는 시간적으로 비효율적이고, 그 결과를 모두 비교하는 것이 문제가 있다.

2.4 웹 브라우저**(GUI)**설계 및 **DB**연동 제약사항

(1) PyScript

- 원래 파이썬과 JS의 호환이 가능한 PyScript를 이용해 Python의 외부 라이브러리인 Matplotlib과 Seaborn을 이용해 분석 데이터를 각종 그래프로 시각화해 출력할 예정이었지만, 파이썬의 코드가 웹상에서 실행되는 시간이 소요되어 실시간 반영의 측면에서 JS 외부 라이브러리인 Chart.js로 대체되었다.

(2) AWS (Amazon Web Service)

- Enron 학습 데이터로 모델을 학습시키는 시간이 소요되기 때문에, 개인 서버를 이용해 AWS 인스턴스를 생성해 Python 코드를 서버에 올려 프로젝트를 시연할 때, 인스턴스를 실행해 학습 시간을 기다릴 필요 없이 즉각적인 결괏값을 출력하길 원했지만, AWS는 온디맨드 방식으로 코드의 실행시간만큼 요금이 책정되어 비용적인 문제로 사용하지 못하고 웹 스토리지를 사용하는 대신, phpMyAdmin과 같은 DB를 공유해 실시간 연동을 구현하는 것으로 대체되었다.

- 추후에 프로젝트가 완성된 후에 웹 서비스를 로컬이 아닌 사용자들이 각자의 핸드폰이나 노트북에서 접속해 볼 수 있게 추가로 AWS를 이용해 볼 수도 있겠다.

(3) 실시간 연동

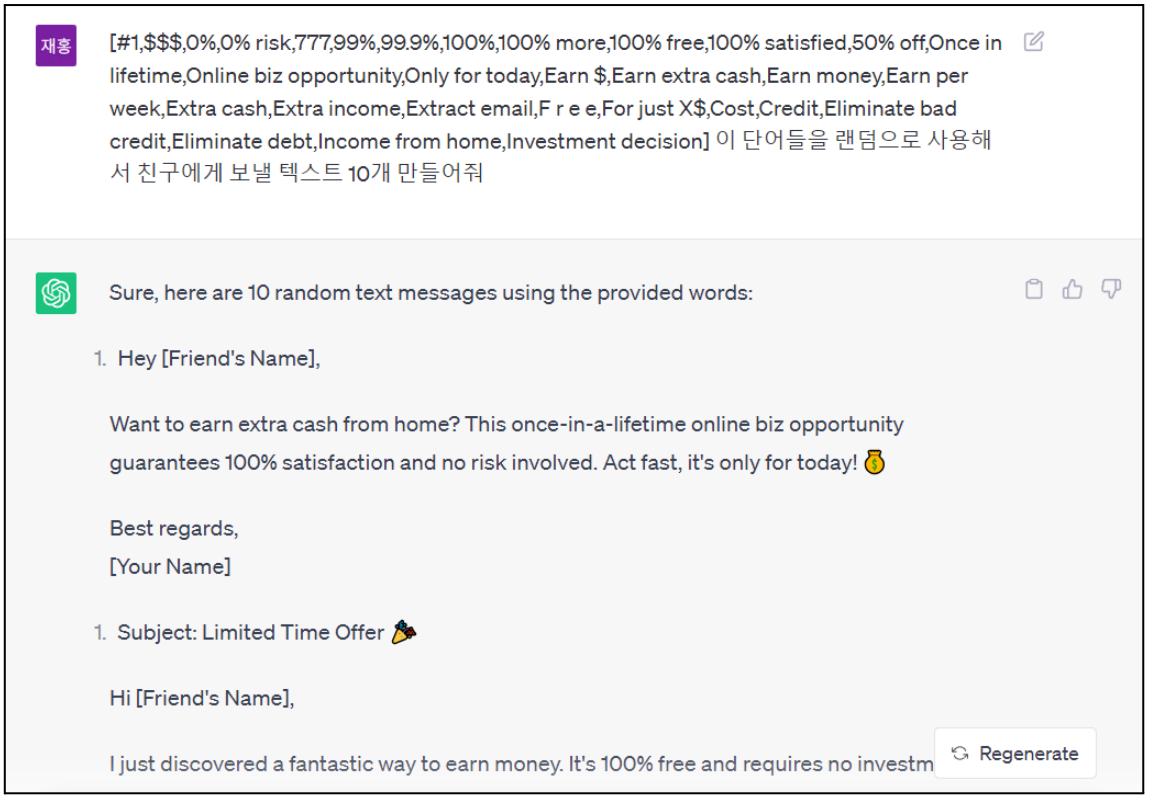
- 웹에서는 DB에 실시간으로 접근하여 DB 에서 필요한 내용을 자동으로 가져올 수 있지만, Google Colab 및 Jupyter notebook에서 DB의 내용을 자동으로 불러와서 구현하지 못하고 직접 셀을 실행시켜주어야 하는 제약사항이 있다. Flask나 Django와 같은 것들을 활용하여 웹에서 파이썬을 실행할 수 있는 방법과 같은 해결책을 모색중이다.

3. 설계 상세화 및 변경사항

3.1 멀티모달 활용

수집 텍스트 스팸 데이터(kaggle, 메일함), 생성 텍스트 스팸 데이터, 수집 이미지 스팸 데이터를 모달리티로 사용하려 하였으나 이미지 스팸 데이터 수집과정에서의 제약사항으로 인해 텍스트 스팸 데이터로 우선 모델 학습을 완료한 뒤 추후에 수집 이미지 스팸 데이터를 수집하는 방향으로 진행할 예정이다. 그리고 기존에 영문과 한글 텍스트 스팸 데이터를 모두 모델학습에 사용하려 하였으나 한글 텍스트 스팸 데이터 수집과정에서의 제약사항으로 인해 한글 텍스트 스팸 데이터는 모델학습에 사용하지 않기로 결정하였다.

생성형 AI서비스로 텍스트 스팸 데이터를 생성을 시도하면, 불법적인 행위에 서비스를 제공하지 않는다며 서비스를 거부하는 제약사항이 존재한다. 이러한 제약사항을 우회하기 위해 스팸 키워드들을 부여하고 해당 키워드를 랜덤하게 사용해서 텍스트를 만들어달라는 방식으로 스팸메일을 생성하였다.



[그림 1] Chat GPT를 사용하여 생성한 스팸메일 예시

3.2 데이터 수집 및 전처리

Concept Drift 현상을 고려하여 다양한 시점의 텍스트 스팸 데이터 셋을 Kaggle 공개 데이터 저장소에서 수집하였으나, 중복되는 데이터 셋이 매우 많았고 최종적으로 수집한 데이터의 수가 약 6000개에 불과하였다. 6000여개의 데이터로 학습을 진행하기에는 그 수가 매우 부족하다고 생각하여 약 6만 개의 이메일 데이터를 가진 Enron E-mail Dataset을 인공신경망 모델 학습에 사용했고, 수집한 6000여개의 데이터는 학습된 모델의 테스트 셋으로 사용하였다. Kaggle 수집 데이터 셋 외에도 본 졸업과제 구성원 3명의 각자의 구글, 네이버 메일함에 있는 메일을 수집하여 테스트 셋으로 사용하였다.

기존에 계획한 텍스트 메일 전처리 방법(입력 feature을 미리 선정해놓는 방법)으로 모델을 학습하면 규칙기반의 스팸 필터링 모델과 동일한 특징을 가지게 되어 새로운 방식으로 생성한 스팸메일에 대응하지 못하는 문제점이 발생했다. 이의 해결방안으로 메일 내용 그 자체를 입력으로 사용하고 학습 모델이 알아서 결과에 영향을 미치는 중요한 feature들을

판단하도록 TF-IDF(Term Frequency-Inverse Document Frequency, 이하 TF-IDF) Vectorize 기법을 사용하였다. TF-IDF 방식은 단어의 빈도와 역 문서 빈도(문서의 빈도에 특정 식을 취하는 것)를 사용하여 메일 내용내의 토큰화 된 단어들마다 중요한 정도에 따라서 가중치를 생성하여 모델 학습에 사용하였다.

```
Out[17]: array([[0.        , 4.90361785, 0.        , ..., 0.        , 0.        ,
                 0.        ],
                [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
                 0.        ],
                [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
                 0.        ],
                ...,
                [0.        , 6.17627678, 0.        , ..., 0.        , 0.        ,
                 0.        ],
                [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
                 0.        ],
                [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
                 0.        ]])
```

[그림 2] 가중치가 부여된 텍스트 예시

추가로 본 과제에서 구현한 TF-IDF 방식의 모델은 규칙 기반 스팸 필터링 방식과 달리 단일 입력 모델인데, 더욱 정확한 성능 비교를 위해서 TF-IDF 방식에 입력 feature들을 추가한 다중 입력모델 또한 검증해보았다. 스팸 여부에 영향을 끼칠 수 있는 feature로 특수문자, 숫자, URL, 대문자, 공백의 개수를 선정했고, 정규표현식을 활용하여 각 feature 값들을 구해주었다.

Out[2]:

	text	special_char	number_count	url_count	upper_count	blank_count	label
0	John Malowney would like us to calculate our p...	6	0	0	8	33	ham
1	MR.WANG QIN.\n\nDAH SING BANK LTD.\n\n19 DES V...	69	9	2	159	486	spam
2	\n\n REVISION #1\n\n\n\n\n\n Friday Night ...	120	60	0	110	416	ham
3	URL: http://www.newsisfree.com/click/-5,855353...	14	3	1	10	21	ham
4	Martin Nutraceutica|s Commences Expansion in G...	268	30	0	179	1035	spam
...	...	...	...	...	...	...	...
56447	This is a multi-part message in MIME format.\n...	498	125	2	617	321	spam
56448	glossed august stall marianne wilful \n\nb...	1655	366	1	643	549	spam
56449	Please plan to attend a meeting on Thursday, A...	23	11	0	15	49	ham
56450	Michelle: \n\n\n\nok we are all set up. We a...	33	7	0	25	216	ham
56451	This is a multi-part message in MIME format.\n...	809	2970	4	10159	64	spam

56452 rows × 7 columns

[그림 3] 추가로 생성한 입력 feature를 포함한 enron 데이터 셋

3.3 모델 선정**/**구축

<하이퍼 파라미터**(Hyper Parameter)>**

본 졸업과제에서 사용한 MLP, LSTM의 하이퍼 파라미터는 경우의 수가 너무 많아 성능비교를 위한 전체 검증은 불가능하였다. 따라서 각 하이퍼 파라미터의 특성에 근거하여 선정하였다.

본 졸업과제는 binary classification(이진 분류)를 기반으로 한다. 이진 분류의 경우 출력 노드가 1개이고, 이 node에서는 0~1 사이의 값을 가지면 마지막에 cast(0.5 이상이면 1, 미만이면 0)를 통해 1 또는 0의 값을 결과로 받을 수 있다. 따라서 활성화 함수로는 이진 분류에 적절한 함수인 시그모이드 함수를 선정하였다. 또한 손실 함수는 결과가 0또는 1인 이진 분류 문제에서 사용되는 binary_crossentropy를 선정하였다. 마지막으로 최적화 함수로는 데이터를 학습할 때 학습 방향과, 스텝 사이즈를 적절하게 결정해주는 Adam 방식을 선정하였다.

<MLP(Multi-Layer Perceptron)>

MLP는 가장 기본적인 인공신경망의 한 형태로 여러개의 퍼셉트론 뉴런을 여러 층으로 쌓은 다층신경망 구조이다. 입력층, 은닉층, 출력층으로 나누어져있다. MLP는 인공신경망 종류 중 범주형 데이터 분류에 가장 적합하다고 알려져 있기 때문에 선정하였다.

학습에 사용되는 입력피처의 개수에 따라 모델의 구조에 약간의 변화가 있으며, MLP 단일입력모델은 아래와 같은 형태를 띄도록 구성하였다.

```
In [27]: def get_simple_model():
             model = Sequential()
             model.add(Dense(512, activation='relu', input_shape=(num_max,)))
             model.add(Dropout(0.5))
             model.add(Dense(256, activation='relu'))
             model.add(Dropout(0.5))
             model.add(Dense(1, activation='sigmoid'))
             model.summary()
             model.compile(loss='binary_crossentropy',
                           optimizer='adam',
                           metrics=['acc',keras.metrics.binary_accuracy])
             print('compile done')
             return model
```

[그림 4] MLP 단일 입력 모델 구조

MLP 다중입력모델은 아래와 같은 구조를 갖도록 설계하였다.

```
In [20]: def get_simple_model():
             # 은닉층과 출력층 설정
             hidden_layer = Dense(64, activation='relu')(merged_input)
             output_layer = Dense(1, activation='sigmoid', name='output')(hidden_layer)

             # 모델 생성
             model = Model(inputs=[text_input, special_char_input, number_count_input, url_count_input, upper_count_input, blank_count_input], outpu

             # 모델 요약
             model.summary()

             # 모델 컴파일
             model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
             print('compile done')
             return model
```

[그림 5] MLP 다중 입력 모델 구조

<LSTM(Long Short Term Memory)>

LSTM은 각 입력들이 독립적인 MLP와 달리, 현재에 입력에서 이전의 입력을 고려하는 방식이다. 또한 RNN에서 발전된 형태인 모델인 만큼 데이터의 길이가 길어질수록 초기 타임 스텝의 정보가 사라지는 단점을 보완하였다. 이러한 특징들 덕분에 LSTM은 주로 순차적인 데이터를 학습, 처리, 분류하는데 주로 사용되고 감성 분석, 언어 모델링, 음성 인식, 비디오 분석등에 응용되고 있다. 하지만, content-based 기반의 학습의 대표적인 모델이고 자연어 처리에도 주로 사용되는 모델이다. 또한 스팸 필터링에도 잘 사용되는 모델이라 MLP와 성능 비교를 위해 선정하였다.

LSTM에 사용되는 데이터는 [Data_size, Time_steps, Features]의 형태인 3차원 배열의 형태로 맞춰주어야 한다. 따라서 아래와 같이 데이터 dimension을 바꿔주었다. 다중 입력 LSTM도 3차원 배열의 형태로 맞춰주는 기본 구조는 아래와 같다.

```
X_train_t = X_train.reshape(X_train.shape[0],1,num_max)

y_train_t = y_train.reshape(y_train.shape[0],1,1)

X_val_t = X_val.reshape(X_val.shape[0],1,num_max)

y_val_t = y_val.reshape(y_val.shape[0],1,1)
```

[그림 6] LSTM 입력을 위한 3차원 데이터로 변경하는 부분.

LSTM 단일 입력 모델의 구조는 아래와 같다.

```
def get_lstm_model():
    # layer를 순차적으로 쌓기위한 시퀀스 선언.
    lstm_model = Sequential()
    # LSTM을 512차원 출력
    # 시퀀스 출력은 True(마지막 아웃풋 시퀀스의 출력만 반환), False는 모두 반환.
    # input_shape(data_size, time_steps, features) 의 3차원 array
    # time_steps은 네트워크에서 사용할 시간 단위. 즉, 과거의 몇개의 데이터를 볼 것인지 결정.
    lstm_model.add(LSTM(256, input_shape=(1,num_max), return_sequences=True))
    '''오버피팅이 되는 것을 방지하기위해 모든 뉴런으로 학습하지 않고, 무작위로 학습을 할 뉴런을 정해서 학습을 진행하는 것이다.
    과적합 방지를 위해 훈련 중에 삭제해야 하는 입력 단위의 비율은 0.3'''
    lstm_model.add(Dropout(0.3))
    lstm_model.add(LSTM(512, return_sequences=True))
    lstm_model.add(Dropout(0.3))
    lstm_model.add(LSTM(256))
    lstm_model.add(Dense(256))
    lstm_model.add(Dropout(0.3))
    lstm_model.add(Dense(1))
    lstm_model.add(Activation('sigmoid'))
    #-----------------------------------------------
    # 은닉층을 사용하려면 LSTM하나더 add하면 된다.
    # lstm_model.add(LSTM(256, return_sequences=True))
    #-----------------------------------------------
    lstm_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

    return lstm_model
```

[그림 7] LSTM 단일 입력 모델 구조

LSTM 다중 입력 모델은, 4 feature와 6 feature를 검증해보았는데 둘 다 기본적인 구조는 아래와 같다.

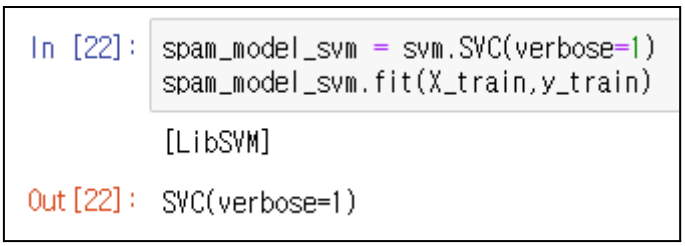
```
def get_6_input_model():
    special_char_input = layers.Input(shape=(1,1))
    number_input = layers.Input(shape=(1,1))
    url_input = layers.Input(shape=(1,1))
    upper_input = layers.Input(shape=(1,1))
    blank_input = layers.Input(shape=(1,1))
    text_input = layers.Input(shape=(1,num_max))

    special_char_output = layers.LSTM(64, dropout=0.3, recurrent_dropout=0.3)(special_char_input)
    number_output = layers.LSTM(64, dropout=0.3, recurrent_dropout=0.3)(number_input)
    url_output = layers.LSTM(64, dropout=0.3, recurrent_dropout=0.3)(url_input)
    upper_output = layers.LSTM(64, dropout=0.3, recurrent_dropout=0.3)(upper_input)
    blank_output = layers.LSTM(64, dropout=0.3, recurrent_dropout=0.3)(blank_input)
    text_output = layers.LSTM(64, dropout=0.3, recurrent_dropout=0.3)(text_input)

    special_model = Model(inputs=special_char_input, outputs=special_char_output)
    number_model = Model(inputs=number_input, outputs=number_output)
    url_model = Model(inputs=url_input, outputs=url_output)
    upper_model = Model(inputs=upper_input, outputs=upper_output)
    blank_model = Model(inputs=blank_input, outputs=blank_output)
    text_model = Model(inputs=text_input, outputs=text_output)

    concatenated = layers.concatenate([special_char_output, number_output, url_output, upper_output,blank_output, text_output])
    concatenated = layers.Dense(32, activation='sigmoid')(concatenated)
    concatenated = layers.BatchNormalization()(concatenated)
    concatenated_output = layers.Dense(1, activation='sigmoid')(concatenated)

    concatenated_model = Model([special_char_input, number_input, url_input, upper_input, blank_input, text_input], concatenated_output)

    concatenated_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

    return concatenated_model
```

[그림 8] LSTM 다중 입력 모델 구조

<SVM(Support Vector Machine)>

SVM은 데이터 분류에 전반적으로 많이 사용되는 모델이다보니 스팸 필터링에 많이 사용되고 그 성능이 검증된 알고리즘이다. 하지만, Spammer가 스팸 필터링을 우회하기 위하여 스팸 메일에서 스팸 키워드의 철자를 의도적으로 잘못 표기하거나 다른 언어로 표기하는 등 다양한 방식의 우회기법에 대처하지 못한다는 단점이 있다. 따라서 기존의 SVM과 본 졸업과제에서 활용하는 학습 모델인 인공신경망과의 비교를 위해서 선정하였다.

Scikit-learn 라이브러리에서 제공하는 SVM모델을 사용하였으며 MLP나 LSTM에 비해 학습속도가 매우 느렸다.



[그림 9] SVM모델 구조

3.4 웹 브라우저(GUI)설계 및 DB연동

(1) 클라이언트 사이드**(Client - Side)**

- HTML
- CSS
- JavaScript
- Chart.js

클라이언트 사이드에서는 웹페이지의 통일성을 위해서 같은 포맷의 header section footer로 구성하였다. 웹 요소들의 출력이나 소멸에 사용되는 애니메이션도 통일하였다. 변경 사항은 앞서 말했듯이, 실행시간의 문제로 PyScript에서 Chart.js로 변경되었다.

(2) 서버 사이드**(Server - Side)**

- PHP
- XAMPP (Apache, phpMyAdmin)

서버 사이드에서는 AWS(Amazon Web Service) 대신 PHP와 XAMPP안에 있는 Apache와 phpMyAdmin을 이용해 다른 플랫폼인 Google Colab(Python)과 웹페이지의 데이터 연동을 실현하였다.

4. 구성원별 진척도(나머지는 알아서 추가)

이름	진척도
윤상호	스팸 필터링 모듈(LSTM,SVM) 개발 및 성능 비교, 전처리(중복 제거, 토큰화, 다중입력모델 검증을 위한 feature 5개 추가 생성), LSTM에 사용할 데이터 변환(3차원화), 데이터 수집, 앞으로 학습에 활용할 생성 데이터 생성 진행 중
조재홍	스팸 필터링 모듈(MLP,SVM) 개발 및 성능 비교, 전처리(중복 제거, 토큰화, 다중입력모델 검증을 위한 feature 5개 추가 생성), 데이터 수집, 앞으로 학습에 활용할 생성 데이터 생성 진행 중
이강우	클라이언트 사이드(HTML, JavaScript, CSS)로 클라이언트가 이용할 수 있는 웹페이지 구현, 서버 사이드(XAMPP, PHP)를 이용해 DB를 구축해 서버와 웹, AI 모델과의 데이터 연동, 데이터 수집, 앞으로 학습에 활용할 생성 데이터 생성 진행 중

5. 보고 시점까지의 과제 수행 내용 및 중간 결과

5.1 데이터 수집 및 전처리

아래와 같이 Enron e-mail dataset을 인공신경망 모델학습에 사용하기 위해 수집하였다.

Out [2] :

	text	label
0	John Malowney would like us to calculate our p...	ham
1	MR.WANG QIN.\n\nDAH SING BANK LTD.\n\n19 DES V...	spam
2	\n\n REVISION #1\n\n\n\n\n\n\n Friday Night ...	ham
3	URL: http://www.newsisfree.com/click/-5,855353...	ham
4	Martin Nutraceutica|s Commences Expansion in G...	spam
...	...	...
56447	This is a multi-part message in MIME format.\n...	spam
56448	glossed august stall marianne wilful \n\nb...	spam
56449	Please plan to attend a meeting on Thursday, A...	ham
56450	Michelle: \n\n\n\nok we are all set up. We a...	ham
56451	This is a multi-part message in MIME format.\n...	spam

56452 rows × 2 columns

[그림 10] 수집한 Enron e-mail dataset

아래 3개의 데이터 셋은 학습된 모델의 성능평가를 위해 수집/생성하였다. 생성 데이터 셋의 경우 나중에 학습을 위해 개수를 많이 늘릴 예정이다. 현재는 학습된 모델의 성능 테스트 용도로만 사용하기에 개수가 많지 않다.

Out [2] :

	text	label
0	\n--- begin forwarded text\nDate: Sun, 11 Aug ...	ham
1	Well, from the looks of things, I can import a...	ham
2	Not exactly new bits, but I enjoyed seeing The...	ham
3	Once upon a time, Daniel wrote :> > And yes, ...	ham
4	Hello fork, Whilst digitally meandering aroun...	ham
...	...	...
6088	looking for property in spain ?\nlooking for\...	spam
6089	[ilug] re : popular . biz and . com extensi...	spam
6090	re : wall street micro news report\nhomeland ...	spam
6091	pleasure your women - size does matter !\nexp...	spam
6092	mail server\ndear projecthoneypot @ projectho...	spam

6093 rows × 2 columns

[그림 11] Kaggle에서 수집한 e-mail dataset

Out [4] :

	text	label
0	★ 23 people won first place last week / 1.1 bi...	spam
1	[D-4] [Last week, too!]! ★ 2.3 billion/11 firs...	spam
2	Reliable On: Line Jung: Product Ratio: As: All...	spam
3	Come before it's too late B: Milbae, Song1+1 i...	spam
4	You can pull your wrinkles without any burden ...	spam
...	...	...
164	There is no plan to deal with multimodal in th...	ham
165	[Plan] 1. Items to be displayed on the benchma...	ham
166	I'm planning to plan and proceed like this, an...	ham
167	The information is a mistake, and we will take...	ham
168	It is not "free of transportation" but "free o...	ham

169 rows × 2 columns

[그림 12] 개인 메일함(Naver, Google)에서 수집한 e-mail dataset

Out [3] :

	text	label
0	Hey there! Exclusive offer: Access now for 100...	spam
1	Urgent! Limited time deal: Get started now and...	spam
2	Attention required: Claim your prize today and...	spam
3	Act now! Exclusive discount: Buy direct and sa...	spam
4	Access right away! Get out of debt NOW with ou...	spam
...	...	...
234	Wishing you a wonderful and productive day.	ham
235	Your friendship is a true blessing in my life.	ham
236	Take care and stay safe, my dear friend.	ham
237	Thanks for being the amazing person you are!	ham
238	Grateful for our friendship. You mean a lot to...	ham

239 rows × 2 columns

[그림 13] 생성형 AI서비스(Bard, Chat GPT)로 생성한 e-mail dataset

인공신경망 모델의 입력변수를 다양화하기 위해 아래 그림과 같은 전처리 과정을 거쳐 특수문자, 숫자, URL, 대문자, 공백의 개수를 카운팅하여 총 5개의 피처를 추가한다.

각각 1 feature(text) / 4 feature(text, special_char, number_count, url_count) / 6 feature(text, special_char, number_count, url_count, upper_count, blank_count) 로 나누어서 테스트한다.

특수문자 카운팅

In [4]:
```
# 정규표현식 적용을 위한 라이브러리 호출
import re
# 정규표현식을 이용하여 특수문자의 수를 세는 함수 정의
def count_special_chars(text):
    special_chars = re.findall(r'[^ㅟㅟs]', text)
    return len(special_chars)
```

In [5]:
```
# 'special_num' 열에 특수문자의 수 계산하여 추가
df['special_char'] = df['text'].apply(count_special_chars)
df
```

Out [5] :

	text	label	special_char
0	John Malowney would like us to calculate our p...	ham	6
1	MR.WANG QIN.\n\nDAH SING BANK LTD.\n\n19 DES V...	spam	69
2	\n\n REVISION #1\n\n\n\n\n\n Friday Night ...	ham	120
3	URL: http://www.newsisfree.com/click/-5,855353...	ham	14
4	Martin Nutraceutica|s Commences Expansion in G...	spam	268
...	...	...	...
56447	This is a multi-part message in MIME format.\n...	spam	498
56448	glossed august stall marianne wilful \n\nb...	spam	1655
56449	Please plan to attend a meeting on Thursday, A...	ham	23
56450	Michelle: \n\n\n\nok we are all set up. We a...	ham	33
56451	This is a multi-part message in MIME format.\n...	spam	809

56452 rows × 3 columns

[그림 14] special_char 칼럼(특수문자 개수)를 추가하는 과정

숫자 카운팅

In [6]:
```
# 숫자 개수 카운팅 함수 정의
def count_numbers(text):
    pattern = r'\d+'  # 숫자를 찾기 위한 정규 표현식
    numbers = re.findall(pattern, text)
    count = len(numbers)
    return count
```

In [7]:
```
df['number_count'] = df['text'].apply(count_numbers)
df
```

Out[7]:

	text	label	special_char	number_count
0	John Malowney would like us to calculate our p...	ham	6	0
1	MR.WANG QIN.\n\nDAH SING BANK LTD.\n\n19 DES V...	spam	69	9
2	\n\n REVISION #1\n\n\n\n\n\n\n Friday Night ...	ham	120	60
3	URL: http://www.newsisfree.com/click/-5,855353...	ham	14	3
4	Martin Nutraceutica|s Commences Expansion in G...	spam	268	30
...	...	...	...	...
56447	This is a multi-part message in MIME format.\n...	spam	498	125
56448	glossed august stall marianne wilful \n\nb...	spam	1655	366
56449	Please plan to attend a meeting on Thursday, A...	ham	23	11
56450	Michelle: \n\n\n\nok we are all set up. We a...	ham	33	7
56451	This is a multi-part message in MIME format.\n...	spam	809	2970

56452 rows × 4 columns

[그림 15] number_count 칼럼(숫자 개수)를 추가하는 과정

문장 토큰화

In [8]:
```
# 토큰화를 위한 토크나이저 라이브러리 호출
from nltk.tokenize import TweetTokenizer
```

In [9]:
```
# example
tweet_tokenizer = TweetTokenizer()
print(tweet_tokenizer.tokenize("This is a coool #dummysmiley: :-) : -P <3 :) www.spamurl.com"))
```

```
['This', 'is', 'a', 'coool', '#dummysmiley', ':', ':-)', ':', '-', 'P', '<3', ':)', 'www.spamurl.com']
```

In [10]:
```
# 토큰화
df['tokens'] = df['text'].apply(tweet_tokenizer.tokenize)
df
```

Out[10]:

	text	label	special_char	number_count	tokens
0	John Malowney would like us to calculate our p...	ham	6	0	[John, Malowney, would, like, us, to, calculat...
1	MR.WANG QIN.\n\nDAH SING BANK LTD.\n\n19 DES V...	spam	69	9	[MR.WANG, QIN, ., DAH, SING, BANK, LTD, ., 19,...
2	\n\n REVISION #1\n\n\n\n\n\n\n Friday Night ...	ham	120	60	[REVISION, #, 1, Friday, Night, -, 6:45, pm, ...
3	URL: http://www.newsisfree.com/click/-5,855353...	ham	14	3	[URL, :, http://www.newsisfree.com/click/-5,85...
4	Martin Nutraceutica|s Commences Expansion in G...	spam	268	30	[Martin, Nutraceutica, |, s, Commences, Expans...
...	...	...	...	...	...
56447	This is a multi-part message in MIME format.\n...	spam	498	125	[This, is, a, multi-part, message, in, MIME, f...
56448	glossed august stall marianne wilful \n\nb...	spam	1655	366	[glossed, august, stall, marianne, wilful, bom...
56449	Please plan to attend a meeting on Thursday, A...	ham	23	11	[Please, plan, to, attend, a, meeting, on, Thu...
56450	Michelle: \n\n\n\nok we are all set up. We a...	ham	33	7	[Michelle, :, ok, we, are, all, set, up, ., We...
56451	This is a multi-part message in MIME format.\n...	spam	809	2970	[This, is, a, multi-part, message, in, MIME, f...

56452 rows × 5 columns

[그림 16] URL카운팅을 위해 text를 TweetTokenizer를 이용하여 토큰화하는 과정

URL 카운팅

In [11]:
```
# 문장 내에 URL 카운팅하는 함수 정의
def url_count(tokens):
    count = 0
    for token in tokens:
        if re.match(r"(http₩S+|www₩.₩S+)", token):
            count = count+1
    return count
```

In [12]:
```
df['url_count'] = df['tokens'].apply(url_count)
df
```

Out[12]:

	text	label	special_char	number_count	tokens	url_count
0	John Malowney would like us to calculate our p...	ham	6	0	[John, Malowney, would, like, us, to, calculat...	0
1	MR.WANG QIN.\n\nDAH SING BANK LTD.\n\n19 DES V...	spam	69	9	[MR.WANG, QIN, ., DAH, SING, BANK, LTD, ., 19,...	2
2	\n\n REVISION #1\n\n\n\n\n\n Friday Night ...	ham	120	60	[REVISION, #, 1, Friday, Night, -, -6:45, pm, ...	0
3	URL: http://www.newsisfree.com/click/-5,855353...	ham	14	3	[URL, :, http://www.newsisfree.com/click/-5,85...	1
4	Martin Nutraceutica|s Commences Expansion in G...	spam	268	30	[Martin, Nutraceutica, |, s, Commences, Expans...	0
...	...	...	...	...	...	...
56447	This is a multi-part message in MIME format.\n...	spam	498	125	[This, is, a, multi-part, message, in, MIME, f...	2
56448	glossed august stall marianne wilful \n\nb...	spam	1655	366	[glossed, august, stall, marianne, wilful, bom...	1
56449	Please plan to attend a meeting on Thursday, A...	ham	23	11	[Please, plan, to, attend, a, meeting, on, Thu...	0
56450	Michelle: \n\n\n\nok we are all set up. We a...	ham	33	7	[Michelle, :, ok, we, are, all, set, up, ., We...	0
56451	This is a multi-part message in MIME format.\n...	spam	809	2970	[This, is, a, multi-part, message, in, MIME, f...	4

56452 rows × 6 columns

[그림 17] url_count 칼럼(URL 개수)를 추가하는 과정

대문자 카운팅

In [14]:
```
# 대문자의 개수를 카운팅하는 함수를 정의
def count_upper(text):
    return len(re.findall(r'[A-Z]', text))
```

In [15]:
```
df['upper_count'] = df['text'].apply(count_upper)
df
```

Out[15]:

	text	label	special_char	number_count	tokens	url_count	upper_count
0	John Malowney would like us to calculate our p...	ham	6	0	[John, Malowney, would, like, us, to, calculat...	0	8
1	MR.WANG QIN.\n\nDAH SING BANK LTD.\n\n19 DES V...	spam	69	9	[MR.WANG, QIN, ., DAH, SING, BANK, LTD, ., 19,...	2	159
2	\n\n REVISION #1\n\n\n\n\n\n Friday Night ...	ham	120	60	[REVISION, #, 1, Friday, Night, -, -6:45, pm, ...	0	110
3	URL: http://www.newsisfree.com/click/-5,855353...	ham	14	3	[URL, :, http://www.newsisfree.com/click/-5,85...	1	10
4	Martin Nutraceutica|s Commences Expansion in G...	spam	268	30	[Martin, Nutraceutica, |, s, Commences, Expans...	0	179
...	...	...	...	...	...	...	...
56447	This is a multi-part message in MIME format.\n...	spam	498	125	[This, is, a, multi-part, message, in, MIME, f...	2	617
56448	glossed august stall marianne wilful \n\nb...	spam	1655	366	[glossed, august, stall, marianne, wilful, bom...	1	643
56449	Please plan to attend a meeting on Thursday, A...	ham	23	11	[Please, plan, to, attend, a, meeting, on, Thu...	0	15
56450	Michelle: \n\n\n\nok we are all set up. We a...	ham	33	7	[Michelle, :, ok, we, are, all, set, up, ., We...	0	25
56451	This is a multi-part message in MIME format.\n...	spam	809	2970	[This, is, a, multi-part, message, in, MIME, f...	4	10159

56452 rows × 7 columns

[그림 18] upper_count 칼럼(대문자 개수)를 추가하는 과정

공백 카운팅

In [16]:
```
# 공백의 개수를 카운팅하는 함수를 정의합니다.
def count_blank(text):
    return text.count(' ')
```

In [17]:
```
df['blank_count'] = df['text'].apply(count_blank)
df
```

Out[17]:

	text	label	special_char	number_count	tokens	url_count	upper_count	blank_count
0	John Malowney would like us to calculate our p...	ham	6	0	[John, Malowney, would, like, us, to, calculat...	0	8	33
1	MR.WANG QIN.\n\nDAH SING BANK LTD.\n\n19 DES V...	spam	69	9	[MR.WANG, QIN, ., DAH, SING, BANK, LTD, ., 19,...	2	159	486
2	\n\n REVISION #1\n\n\n\n\n\n Friday Night ...	ham	120	60	[REVISION, #, 1, Friday, Night, -, -6:45, pm, ...	0	110	416
3	URL: http://www.newsisfree.com/click/-5,855353...	ham	14	3	[URL, :, http://www.newsisfree.com/click/-5,85...	1	10	21
4	Martin Nutraceutica|s Commences Expansion in G...	spam	268	30	[Martin, Nutraceutica, |, s, Commences, Expans...	0	179	1035
...	...	...	...	...	...	...	...	...
56447	This is a multi-part message in MIME format.\n...	spam	498	125	[This, is, a, multi-part, message, in, MIME, f...	2	617	321
56448	glossed august stall marianne wilful \n\nb...	spam	1655	366	[glossed, august, stall, marianne, wilful, bom...	1	643	549
56449	Please plan to attend a meeting on Thursday, A...	ham	23	11	[Please, plan, to, attend, a, meeting, on, Thu...	0	15	49
56450	Michelle: \n\n\n\nok we are all set up. We a...	ham	33	7	[Michelle, :, ok, we, are, all, set, up, ., We...	0	25	216
56451	This is a multi-part message in MIME format.\n...	spam	809	2970	[This, is, a, multi-part, message, in, MIME, f...	4	10159	64

56452 rows × 8 columns

[그림 19] blank_count 칼럼(공백 개수)를 추가하는 과정

위의 전처리 과정을 거쳐 Enron e-mail 데이터 셋, Kaggle 수집 데이터 셋, 개인 메일함 수집 데이터 셋, 생성형 AI 생성 데이터 셋을 6개의 독립변수와 1개의 종속변수를 가진 학습 및 테스트 용 데이터 셋으로 수정하였다.

5.2 모델 학습 및 성능 비교

<MLP/LSTM 학습 공통 부분>

전처리가 완료된 데이터 셋에서 text를 토큰화하고 TF-IDF Vectorization을 수행하여 토큰의 가중치를 구한다. 입력피처를 text하나만 사용한다면 해당 가중치를 입력으로 모델학습을 진행한다.

```
Out[17]: array([[0.        , 4.90361785, 0.        , ..., 0.        , 0.        ,
                 0.        ],
                [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
                 0.        ],
                [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
                 0.        ],
                ...,
                [0.        , 6.17627678, 0.        , ..., 0.        , 0.        ,
                 0.        ],
                [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
                 0.        ],
                [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
                 0.        ]])
```

[그림 20] 모델 학습용 Enron 데이터 셋의 text 가중치

또한 추가로 열심히 학습한 것을 잃지 않고, 언제든 훈련을 중단할 수 있도록 모델 체크포인트를 사용하였다. 모델 체크포인트를 활용하면 매 에포크마다 네트워크 노드의 가중치를 파일에 저장할 수 있다. 이는 우리가 학습 시킨것에 대한 가중치를 잃어버리지 않고, 손실 값을 만족하면 신경망 작동을 중단할 수 있도록 한다. 만약, 모델 체크포인트가 없다면 모든 에포크를 학습할 때까지 기다려야 할 것이다.

```
# define checkpointer
checkpointer = ModelCheckpoint(filepath=model_save_path,
                               verbose=1,
                               save_best_only=True)


# define tensorboard
tensorboard = TensorBoard(log_dir='./logs',
                          histogram_freq=0,
                          write_graph=True,
                          write_images=True)
```

[그림 21] 모델 체크포인트

```
# get the compiled model
model = get_lstm_model()

# load history
# history=check_model(m,mat_texts,tags,epochs=10)
history=check_model2(model,X_train_t,y_train_t,X_val_t,y_val_t,epochs=10)
```

```
Epoch 1/10
617/618 [============================>.] - ETA: 0s - loss: 0.0418 - accuracy: 0.9830
Epoch 00001: val_loss improved from 0.02529 to 0.02080, saving model to checkpoints₩spam_detector_enron_model.h5
618/618 [==============================] - 36s 58ms/step - loss: 0.0418 - accuracy: 0.9830 - val_loss: 0.0208 - val_accuracy: 0.9933
Epoch 2/10
617/618 [============================>.] - ETA: 0s - loss: 0.0087 - accuracy: 0.9968
Epoch 00002: val_loss did not improve from 0.02080
618/618 [==============================] - 29s 47ms/step - loss: 0.0087 - accuracy: 0.9968 - val_loss: 0.0301 - val_accuracy: 0.9915
Epoch 3/10
618/618 [==============================] - ETA: 0s - loss: 0.0048 - accuracy: 0.9982
Epoch 00003: val_loss did not improve from 0.02080
618/618 [==============================] - 28s 46ms/step - loss: 0.0048 - accuracy: 0.9982 - val_loss: 0.0247 - val_accuracy: 0.9940
Epoch 4/10
618/618 [==============================] - ETA: 0s - loss: 0.0038 - accuracy: 0.9988
```

[그림 22] 모델 학습 진행 모습

<MLP 단일 입력 모델 학습과정>

MLP 단일 입력 모델은 은닉층은 2개의 Layer를 가지고 각 Layer의 노드 개수는 512, 256개이며 활성화함수는 Relu를 사용하였다. 출력층은 1개의 노드, 활성화함수는 Sigmoid함수를 사용하여 마무리하였다.

```
In [27]: def get_simple_model():
             model = Sequential()
             model.add(Dense(512, activation='relu', input_shape=(num_max,)))
             model.add(Dropout(0.5))
             model.add(Dense(256, activation='relu'))
             model.add(Dropout(0.5))
             model.add(Dense(1, activation='sigmoid'))
             model.summary()
             model.compile(loss='binary_crossentropy',
                           optimizer='adam',
                           metrics=['acc',keras.metrics.binary_accuracy])
             print('compile done')
             return model
```

[그림 23] MLP모델(단일입력) 구조

<MLP 다중 입력 모델 학습과정>

전처리과정에서 추가했던 변수들을 입력피처로 추가하여 모델을 학습한다면 Fuctional Api를 사용하여 인공신경망에 다중입력이 가능하도록 각 입력피처들을 하나로 concatenate하여 모델을 학습한다.

```
In [14]: special_char = np.array(df['special_char'])
         number_count = np.array(df['number_count'])
         url_count = np.array(df['url_count'])
         upper_count = np.array(df['upper_count'])
         blank_count = np.array(df['blank_count'])

         # 텍스트 입력 레이어
         text_input = Input(shape=(num_max,), name='text_input')

         # 특수문자, 숫자 개수, URL 개수 입력 레이어
         special_char_input = Input(shape=(1,), name='special_char_input')
         number_count_input = Input(shape=(1,), name='number_count_input')
         url_count_input = Input(shape=(1,), name='url_count_input')
         upper_count_input = Input(shape=(1,), name='upper_count_input')
         blank_count_input = Input(shape=(1,), name='blank_count_input')

         # 텍스트 입력 레이어와 나머지 입력 레이어들을 결합
         merged_input = concatenate([text_input, special_char_input, number_count_input, url_count_input, upper_count_input, blank_count_input])
```

[그림 24] 모델 학습을 위해 각 입력피처들을 하나로 합치는 코드

MLP 다중 입력 모델은 은닉층의 노드 개수는 64개, 활성화함수는 Relu를 사용하였다. 출력층은 1개의 노드, 활성화함수는 Sigmoid함수를 사용하여 마무리하였다.

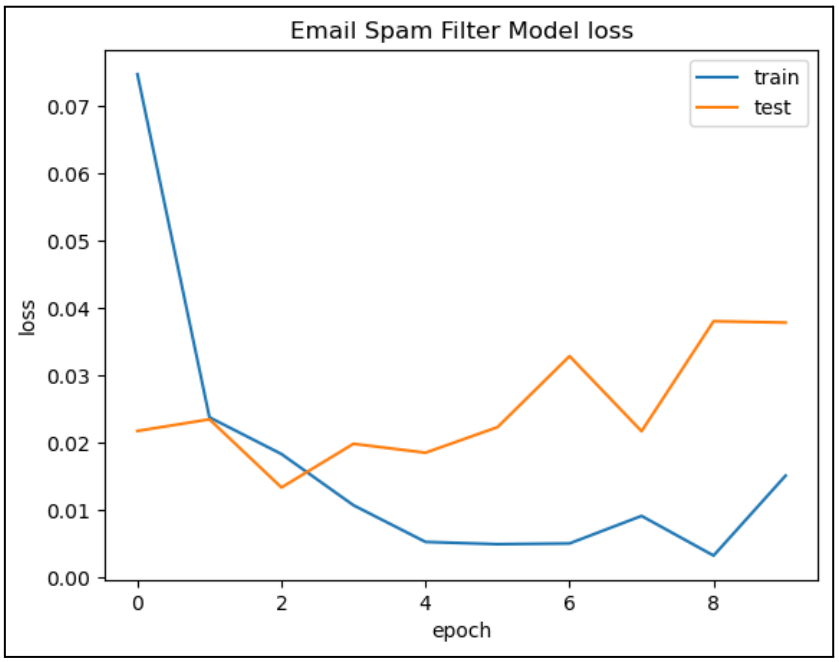
```
In [19]: def get_simple_model():
             # 은닉층과 출력층 설정
             hidden_layer = Dense(64, activation='relu')(merged_input)
             output_layer = Dense(1, activation='sigmoid', name='output')(hidden_layer)

             # 모델 생성
             model = Model(inputs=[text_input, special_char_input, number_count_input, url_count_input, upper_count_input, blank_count_input], outpu
             # 모델 요약
             model.summary()

             # 모델 컴파일
             model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
             print('compile done')
             return model
```

[그림 25] MLP모델(다중입력) 구조

앞서 만든 모델의 구조를 가지고 학습을 진행한다. 총 64개의 sample을 포함하는 네트워크를 통해 전파되는 각 batch에 대해 epoch 10(반복횟수)동안 네트워크를 학습시켰다. 해당 과정은 단일/다중 입력 모델 모두 동일하게 진행하였다.

```
def check_model2(model,x_train,y_train,x_val,y_val,epochs=10):
    history=model.fit(x_train,y_train,batch_size=64,
                      epochs=epochs,verbose=1,
                      shuffle=True,
                      validation_data=(x_val, y_val),
                      callbacks=[checkpointer, tensorboard]).history
    return history
```

[그림 26] MLP 모델 학습(fit)



[그림 27] MLP train/test loss plot

<LSTM 단일 입력 모델 학습과정>

LSTM 단일 입력 모델의 구조는 keras의 layer를 순차적으로 쌓기 위한 Sequential()을 사용했고, 총 4가지 유형의 layer를 사용한다. LSTM layer는 어떤 시퀀스를 입력으로 넣었을 때, 출력으로 또 다른 시퀀스 또는 행렬을 뽑아내는 순환신경망이다. dropout layer는 모델을 학습시킬 때 오버피팅(overfitting)이 되는 것을 방지하는 방법이다. 이 레이어는 학습할 때 모든 뉴런을 학습하지 않고, 학습할 뉴런을 무작위로 결정해서 진행하는 방법이다. Dense layer는 이전 레이어의 모든 뉴런과 결합된 형태의 레이어이다. Activation 레이어는 신경망이 노드의 출력을 계산하는데 사용할 활성화 함수를 선택하는 부분이다.

```
def get_lstm_model():
    lstm_model = Sequential()
    lstm_model.add(LSTM(256, input_shape=(1,num_max), return_sequences=True))
    lstm_model.add(Dropout(0.3))
    lstm_model.add(LSTM(512, return_sequences=True))
    lstm_model.add(Dropout(0.3))
    lstm_model.add(LSTM(256))
    lstm_model.add(Dense(256))
    lstm_model.add(Dropout(0.3))
    lstm_model.add(Dense(1))
    lstm_model.add(Activation('sigmoid'))

    lstm_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

    return lstm_model
```

[그림 28] LSTM 모델(단일 입력) 구조

LSTM은 (Data_size, time_steps, features) 의 3차원 배열의 형태로 데이터를 입력받으므로 우리가 가진 학습 데이터 셋을 3차원으로 변경시켜주어야 한다.

```
X_train_t = X_train.reshape(X_train.shape[0],1,num_max)

y_train_t = y_train.reshape(y_train.shape[0],1,1)

X_val_t = X_val.reshape(X_val.shape[0],1,num_max)

y_val_t = y_val.reshape(y_val.shape[0],1,1)
```

[그림 29] LSTM 모델(단일 입력) 학습데이터 셋을 3차원으로 변경하는 부분.

```
def check_model2(model,x_train,y_train,x_val,y_val,epochs=1):
    history=model.fit(x_train,y_train,batch_size=64,
                      epochs=epochs,verbose=1,
                      shuffle=True,
```

[그림 30] LSTM 모델(단일 입력) 학습(fit)

앞서 만든 모델의 구조를 가지고 학습을 시작한다. 기본적으로 첫번째 파라미터는 앞서 3차원으로 변경시켜준 입력 시퀀스 리스트이고, 두 번째 파라미터는 각 출력의 리스트이다. 본 졸업과제에서는 총 64개의 sample을 포함하는 네트워크를 통해 전파되는 각 batch에 대해 epoch 10(반복횟수)동안 네트워크를 학습시켰다.

<LSTM 다중 입력 모델 학습과정>

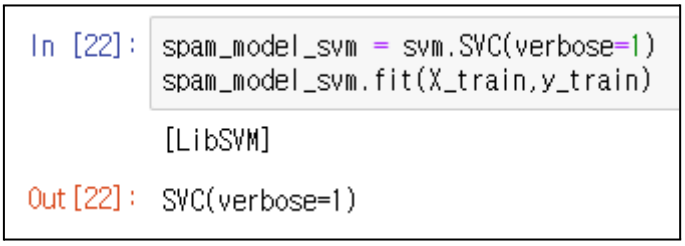
다중 입력의 경우 단일 입력과는 달리 layers.Input을 사용하여 각 feature에 대한 입력 shape을 정해놓고, 각 feature에 대한 6개의 LSTM 단일 입력 모델 구조를 구현한 뒤, 모든 LSTM 모델을 concatenate를 통해 결합하는 방식을 사용한다. 그 외에 나머지는 기본적으로 단일 입력 모델과 구조가 같다.(3차원으로 변경하는 부분이 피처수만큼 늘어난 것을 제외하면 대부분 같음)

```
def get_6_input_model():
    special_char_input = layers.Input(shape=(1,1))
    number_input = layers.Input(shape=(1,1))
    url_input = layers.Input(shape=(1,1))
    upper_input = layers.Input(shape=(1,1))
    blank_input = layers.Input(shape=(1,1))
    text_input = layers.Input(shape=(1,num_max))

    special_char_output = layers.LSTM(64, dropout=0.3, recurrent_dropout=0.3)(special_char_input)
    number_output = layers.LSTM(64, dropout=0.3, recurrent_dropout=0.3)(number_input)
    url_output = layers.LSTM(64, dropout=0.3, recurrent_dropout=0.3)(url_input)
    upper_output = layers.LSTM(64, dropout=0.3, recurrent_dropout=0.3)(upper_input)
    blank_output = layers.LSTM(64, dropout=0.3, recurrent_dropout=0.3)(blank_input)
    text_output = layers.LSTM(64, dropout=0.3, recurrent_dropout=0.3)(text_input)

    special_model = Model(inputs=special_char_input, outputs=special_char_output)
    number_model = Model(inputs=number_input, outputs=number_output)
    url_model = Model(inputs=url_input, outputs=url_output)
    upper_model = Model(inputs=upper_input, outputs=upper_output)
    blank_model = Model(inputs=blank_input, outputs=blank_output)
    text_model = Model(inputs=text_input, outputs=text_output)

    concatenated = layers.concatenate([special_char_output, number_output, url_output, upper_output,blank_output, text_output])
    concatenated = layers.Dense(32, activation='sigmoid')(concatenated)
    concatenated = layers.BatchNormalization()(concatenated)
    concatenated_output = layers.Dense(1, activation='sigmoid')(concatenated)

    concatenated_model = Model([special_char_input, number_input, url_input, upper_input, blank_input, text_input], concatenated_output)

    concatenated_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

    return concatenated_model
```

[그림 31] LSTM 모델(다중 입력) 구조

<SVM 학습과정>

아래와 같은 과정을 통해 모델을 학습 시키고 테스트 셋을 통해 다른 모델과 성능 비교 해본 결과, 개인 메일함을 통해 수집한 데이터와 생성형 AI서비스를 통해 생성한 데이터에 대해 SVM이 MLP와 LSTM에 비해 매우 낮은 성능을 보여서 다중입력모델 실험에서는 제외하였다.



[그림 32] SVM모델 학습과정

<각 모델 성능 비교>

성능 비교 결과를 보면, 빨간색으로 표시한 단일 입력(1개 피처) MLP와 다중 입력(6개 피처) LSTM 의 성능이 가장 좋게 나오고 있다. 성능 지표로는 Precision(정밀도)를 사용하였다.

1feature	SVM	LSTM	MLP
enron	0.94	0.98	0.99
collect	0.79	0.85	0.88
mailbox	0.13	0.67	0.7
create	0.25	0.7	0.68

4feature	LSTM	MLP
enron	0.99	0.98
collect	0.87	0.86
mailbox	0.63	0.66
create	0.68	0.7

6feature	LSTM	MLP
enron	0.99	0.98
collect	0.87	0.87
mailbox	0.65	0.63
create	0.78	0.74

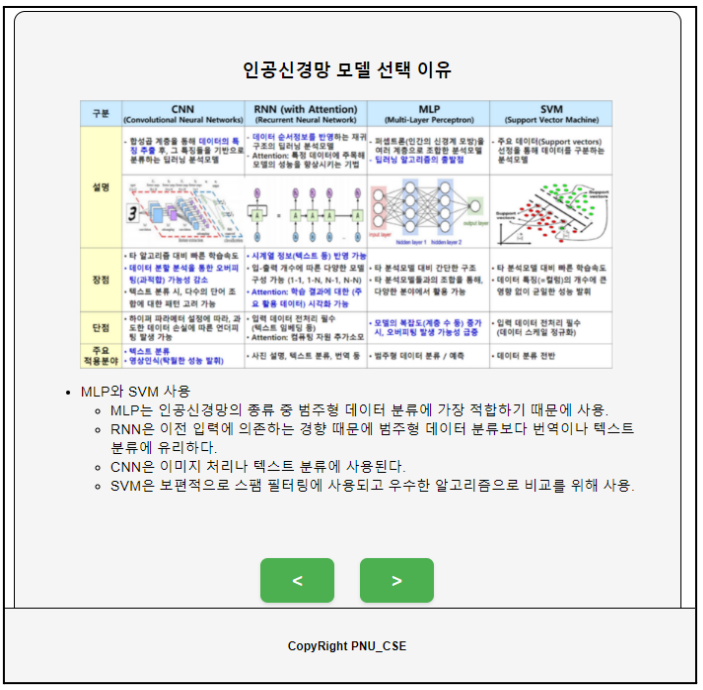
[그림 33] 각 모델의 입력피처 별 성능(정확도) 지표

5.3 Web & DataBase 연동

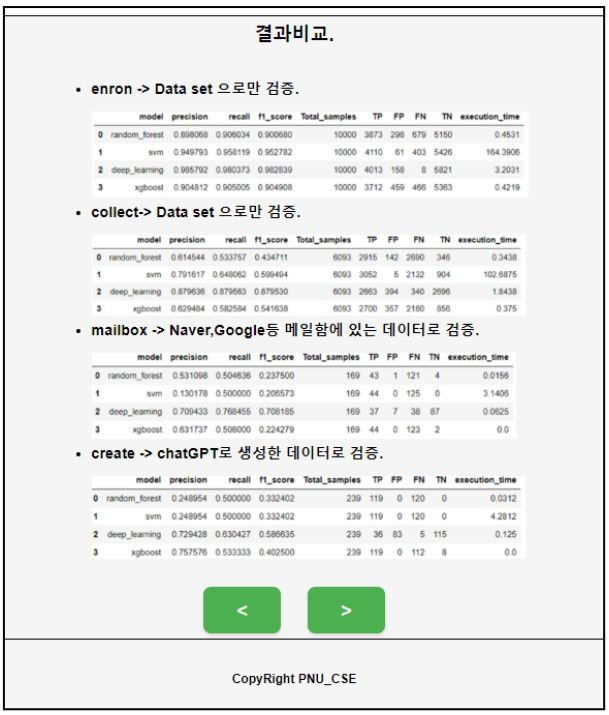
첫 화면은 머신러닝 소개, 메일 쓰기, DB 조회의 세 가지 분기점으로 나누어진다. 각자 클릭을 하면 그에 맞는 역할을 수행한다. 홈 버튼을 클릭 하면 항상 첫 페이지로 돌아온다.

[그림 34] First Page

머신러닝 소개 버튼을 클릭 시, 인공신경망 모델의 선택 이유, 수치로 뽑아낸 결과 등을 클라이언트는 버튼으로 클릭해 이동하며 기계학습의 간략한 설명을 볼 수 있다.



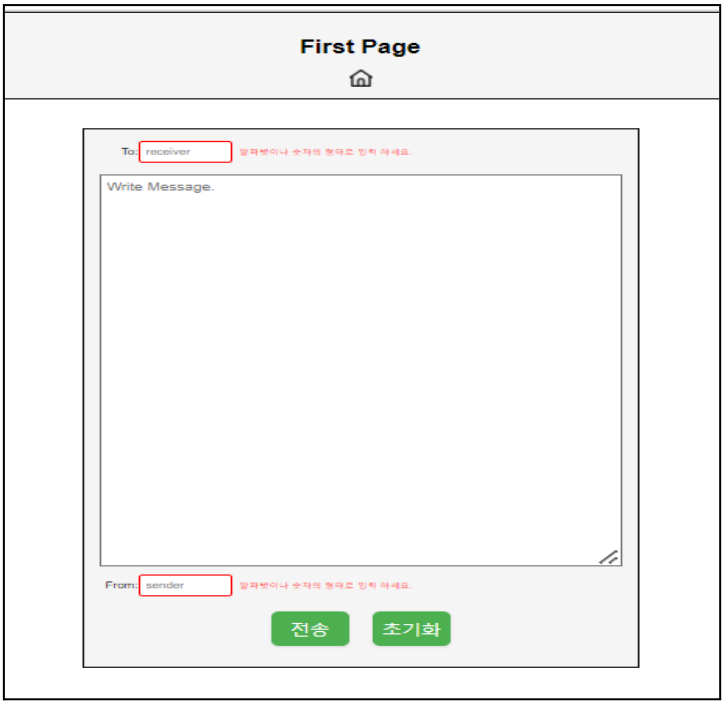
[그림 35] 인공신경망 모델 선택 이유 설명



[그림 36] Data Set 결과 비교

메일 쓰기 버튼을 클릭하면 모든 버튼이 사라지고 메일 형식의 텍스트가 나타난다.

공란없이 형식에 맞게 입력해야 하며 패턴에 맞게 입력하지 않으면 에러 메시지가 출력되고 DB에 입력되지 않는다.



[그림 37] 클라이언트 입력 메일 화면

DB의 Primary Key인 idx를 SELECT 문으로 가져와 집계함수 MAX를 이용해 가장 마지막의 IDX를 1 증가시켜 다음 입력할 메시지의 PK로 사용한다. 그 후 웹페이지에 POST 형식으로 클라이언트가 입력한 값들을 INSERT INTO 쿼리문으로 DB에 집어넣는 방식으로 구현하였다.

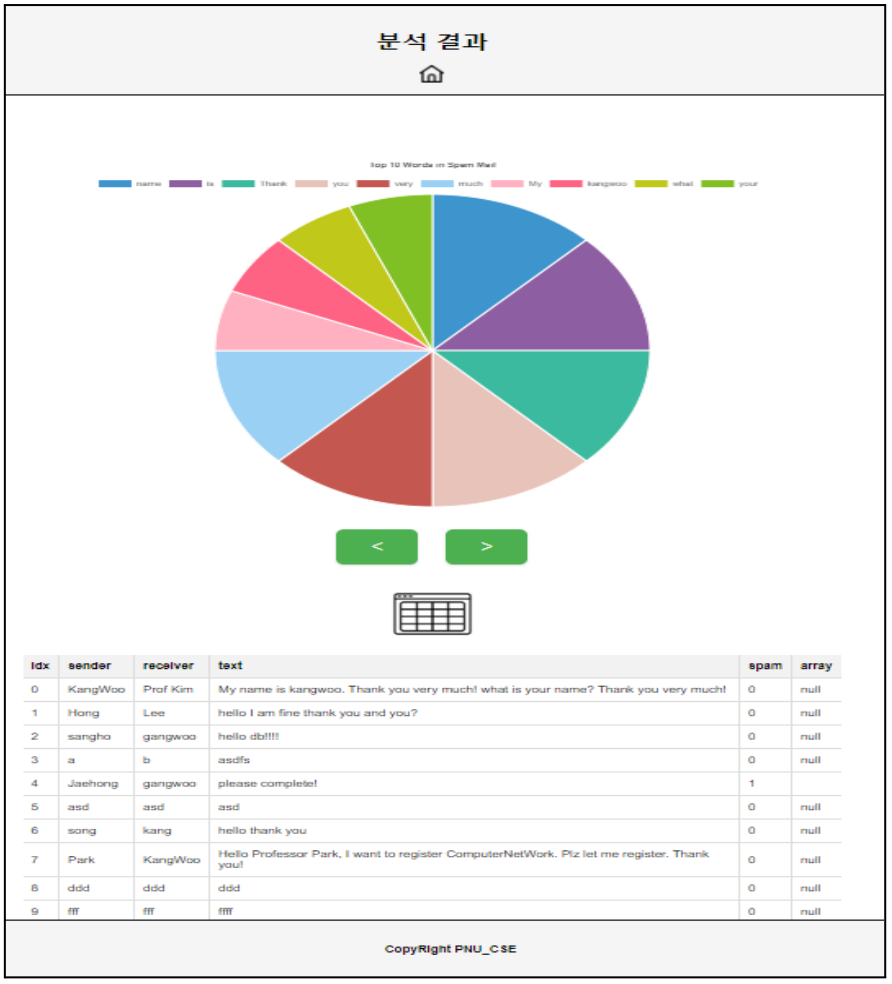
```php
$query = "SELECT MAX(idx) AS last_idx FROM mail";
$result = $database->query($query);
$row = $result->fetch_assoc();
$lastIdx = $row['last_idx'];
$newIdx = $lastIdx + 1;

$to = $_POST['To'];
$from = $_POST['From'];
$mailText = $_POST['mail'];
$result = mysqli_query($database, "INSERT INTO mail (idx,sender,receiver, text)
VALUES ('$newIdx','$from', '$to', '$mailText')");
mysqli_close($database);
```

[그림 38] 클라이언트 입력 정보 DB에 INSERT 쿼리 PHP 구현 코드

현재는 DB의 첫 번째 text 값을 가져와 가장 많이 나온 10개의 단어를 Chart.js를 이용해 Pie-Chart로 출력한다. 기계 학습한 data와 실시간 연동이 더 익숙해지면 AI 모델의 출력값을 받아 상위 10개의 스팸 키워드를 시각화할 예정이다. 밑에 있는 테이블 아이콘을 누르면 메시지에 사용된 전체 단어들을 테이블로 시각화한다.

```html
<canvas id="pie-chart" width="1000" height="600"></canvas>
<script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.5.0/Chart.min.js"></script>
<script>
new Chart(document.getElementById("pie-chart"), {
type: 'pie',
data: {
  labels: ky,
  datasets: [{
    label: "spam keywords",
    backgroundColor: ["#3e95cd", "#8e5ea2","#3cba9f","#e8c3b9","#c45850",'#9BD0F5','#FFB1
    data: dt
  }]
},
options: {
  title: {
    display: true,
    text: 'Top 10 Words in Spam Mail'
  }
}
```

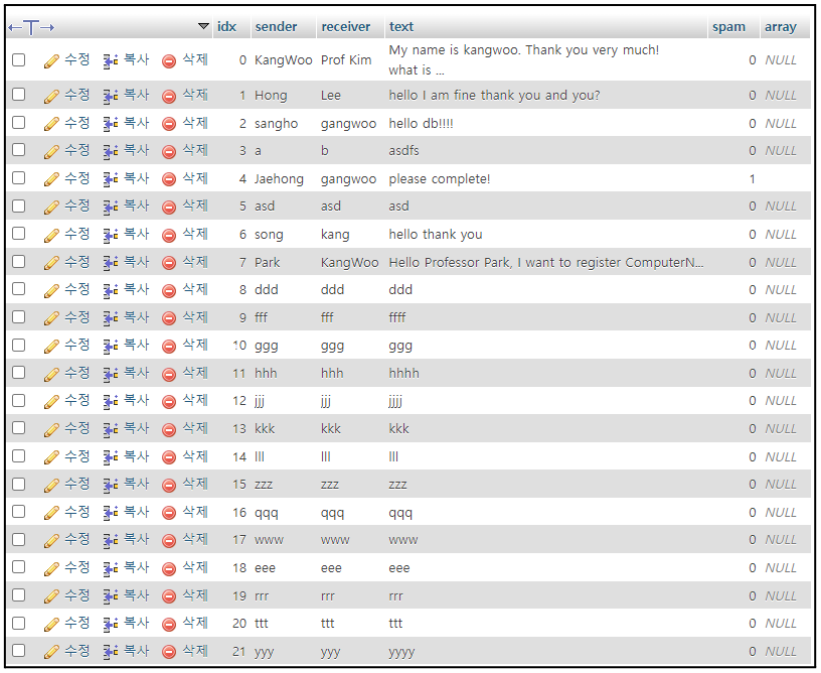
[그림 39] Chart.js를 이용한 그래프 시각화 JS 구현 코드



Idx	sender	receiver	text	spam	array
0	KangWoo	Prof Kim	My name is kangwoo. Thank you very much! what is your name? Thank you very much!	0	null
1	Hong	Lee	hello I am fine thank you and you?	0	null
2	sangho	gangwoo	hello db!!!!	0	null
3	a	b	asdfs	0	null
4	Jaehong	gangwoo	please complete!	1	
5	asd	asd	asd	0	null
6	song	kang	hello thank you	0	null
7	Park	KangWoo	Hello Professor Park, I want to register ComputerNetWork. Plz let me register. Thank you!	0	null
8	ddd	ddd	ddd	0	null
9	fff	fff	ffff	0	null

[그림 40] 클라이언트 입력 정보를 그래프와 테이블로 시각화

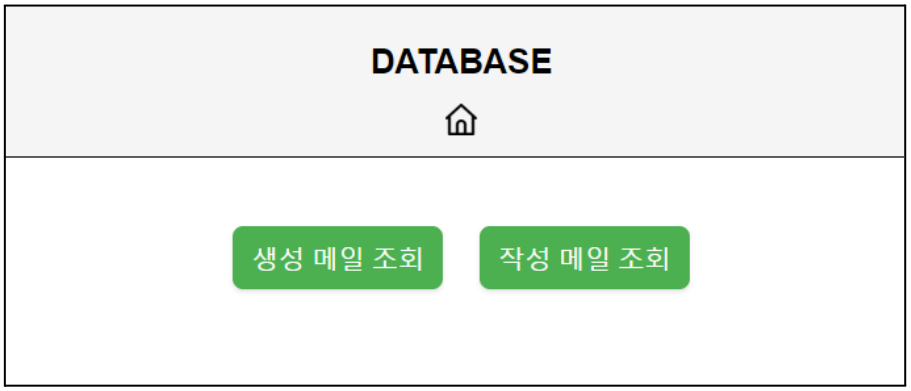
XAMPP를 이용해 Apache와 phpMyAdmin을 사용해 db를 구축하였으며, 컬럼으로는 idx, sender, receiver, text, spam, array가 들어가 있다. idx는 PK고 Unique한 값으로 중복이 허용되지 않는다. 혹시라도 특정 값을 가져와야 할 때를 대비해 만든 컬럼이고, sender는 송신자, receiver는 수신자, text는 메일 본문의 내용, spam은 int 값으로 1이면 spam 메일 0이면 ham 메일, array는 null이 허용되는 컬럼으로 기계학습의 출력값이 담길 예정이고, 최종적으로 이 값을 이용해 그래프를 시각화 할 예정이다.



	idx	sender	receiver	text	spam	array
수정 복사 삭제	0	KangWoo	Prof Kim	My name is kangwoo. Thank you very much! what is ...	0	NULL
수정 복사 삭제	1	Hong	Lee	hello I am fine thank you and you?	0	NULL
수정 복사 삭제	2	sangho	gangwoo	hello db!!!!	0	NULL
수정 복사 삭제	3	a	b	asdfs	0	NULL
수정 복사 삭제	4	Jaehong	gangwoo	please complete!	1	
수정 복사 삭제	5	asd	asd	asd	0	NULL
수정 복사 삭제	6	song	kang	hello thank you	0	NULL
수정 복사 삭제	7	Park	KangWoo	Hello Professor Park, I want to register ComputerN...	0	NULL
수정 복사 삭제	8	ddd	ddd	ddd	0	NULL
수정 복사 삭제	9	fff	fff	ffff	0	NULL
수정 복사 삭제	10	ggg	ggg	ggg	0	NULL
수정 복사 삭제	11	hhh	hhh	hhhh	0	NULL
수정 복사 삭제	12	jjj	jjj	jjjj	0	NULL
수정 복사 삭제	13	kkk	kkk	kkk	0	NULL
수정 복사 삭제	14	lll	lll	lll	0	NULL
수정 복사 삭제	15	zzz	zzz	zzz	0	NULL
수정 복사 삭제	16	qqq	qqq	qqq	0	NULL
수정 복사 삭제	17	www	www	www	0	NULL
수정 복사 삭제	18	eee	eee	eee	0	NULL
수정 복사 삭제	19	rrr	rrr	rrr	0	NULL
수정 복사 삭제	20	ttt	ttt	ttt	0	NULL
수정 복사 삭제	21	yyy	yyy	yyyy	0	NULL

[그림 41] phpMyAdmin db 정보

마지막 버튼인 DB조회를 누르면 DATABASE 페이지로 이동한다. 2개의 분기점이 있으며 검증용으로 각자 생성한 '생성 메일 조회', 클라이언트가 직접 작성한 '작성 메일 조회'의 2개의 테이블을 조회 할 수 있다. 현재는 테스트용으로 한개의 테이블만 구현되어 있지만, 최종적으로는 2개의 테이블을 사용할 예정이다.



[그림 42] DB 조회 화면

PHP로 가져온 data_arr을 json_encode를 이용해 JSON 객체로 받아 JS의 printTable 함수에서 table을 생성하여 웹상에 테이블을 생성한다. table의 요소를 클릭하면 메일 형식에 맞춰 조회해 볼 수 있다. 실시간 연동이 익숙해지면 AI 모델의 출력 값을 이용해

SPAM 키워드가 있다면 강조되어 출력되는 방식으로 구현 예정이다. OverFlow 속성이 적용되어 있어, 허용된 크기를 넘어가면 스크롤 바로 내리면서 조회할 수 있다.

```php
$data_arr = array();
$result = mysqli_query($database, "SELECT * FROM mail");
while($row = mysqli_fetch_assoc($result)){
  $data = array("idx"=> $row['idx'], "sender"=> $row['sender'], "receiver"=> $row['receiver'], "text"=>$row['text'],
  "spam" => $row['spam'], "array"=>$row['array']);
  $data_arr[] = $data;
}
```

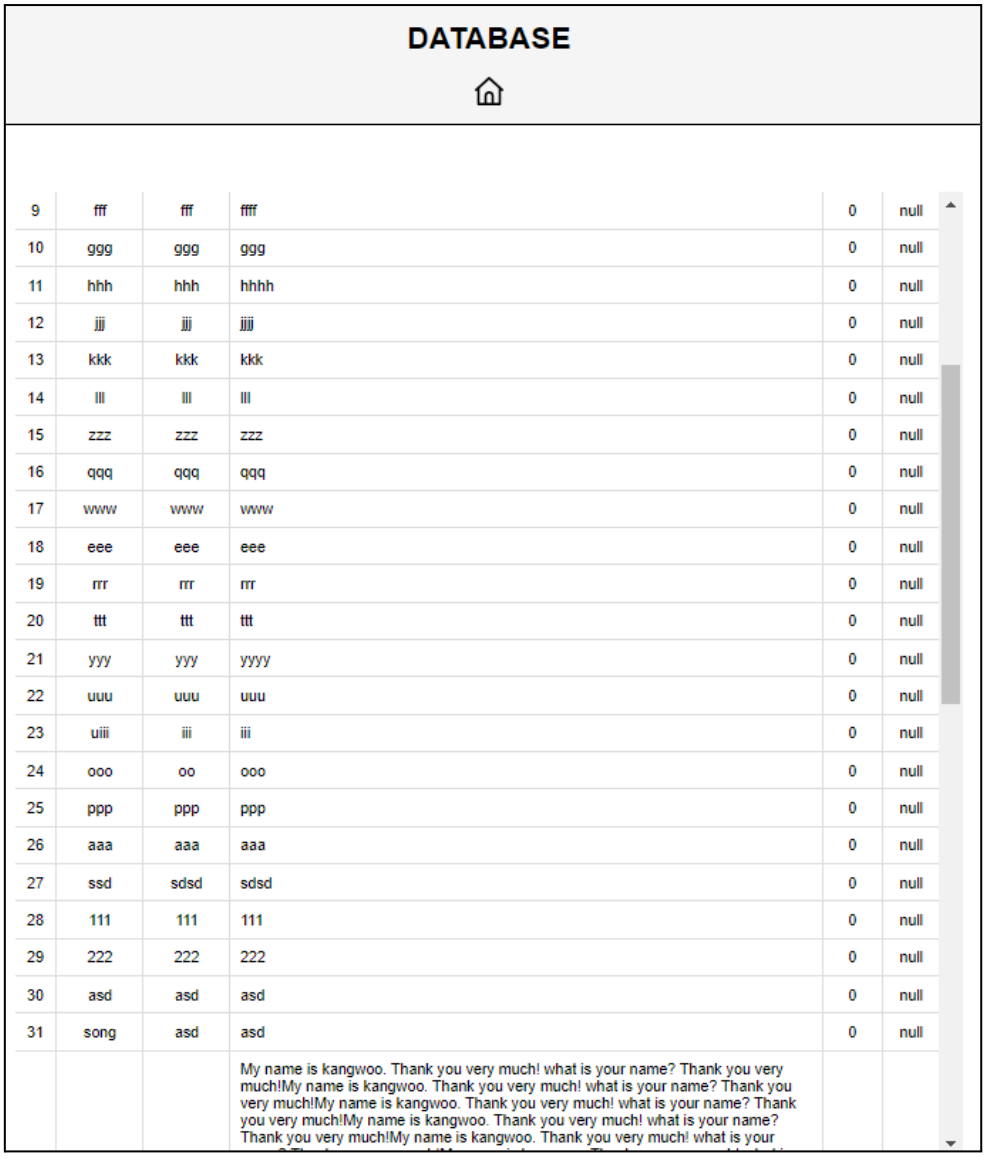
[그림 43] DB를 웹상으로 가져오는 SELECT 쿼리 PHP 구현 코드

```javascript
<script>
  var database = <?php echo json_encode($data_arr);?>;
  function printTable(){
    event.preventDefault();
    var st = "<table><thead><tr><th>idx</th><th>sender</th><th>receiver</th><th>text</th
      //add database element
      for (var i = 0; i < database.length; i++) {
        st += '<tr id="row-' + database[i].idx +'"  onclick="handleRowClick(this)">';
        st += '<td class="center-align">' + database[i].idx + '</td>';
        st += '<td class="center-align">' + database[i].sender + '</td>';
        st += '<td class="center-align">' + database[i].receiver + '</td>';
        st += '<td>' + database[i].text + '</td>';
        st += '<td class="center-align">' + database[i].spam + '</td>';
        st += '<td class="center-align">' + database[i].array + '</td>';
        st += '</tr>';
      }
      st+= "</tbody></table>";
      document.getElementById("dbTable").innerHTML = st;
  }
```

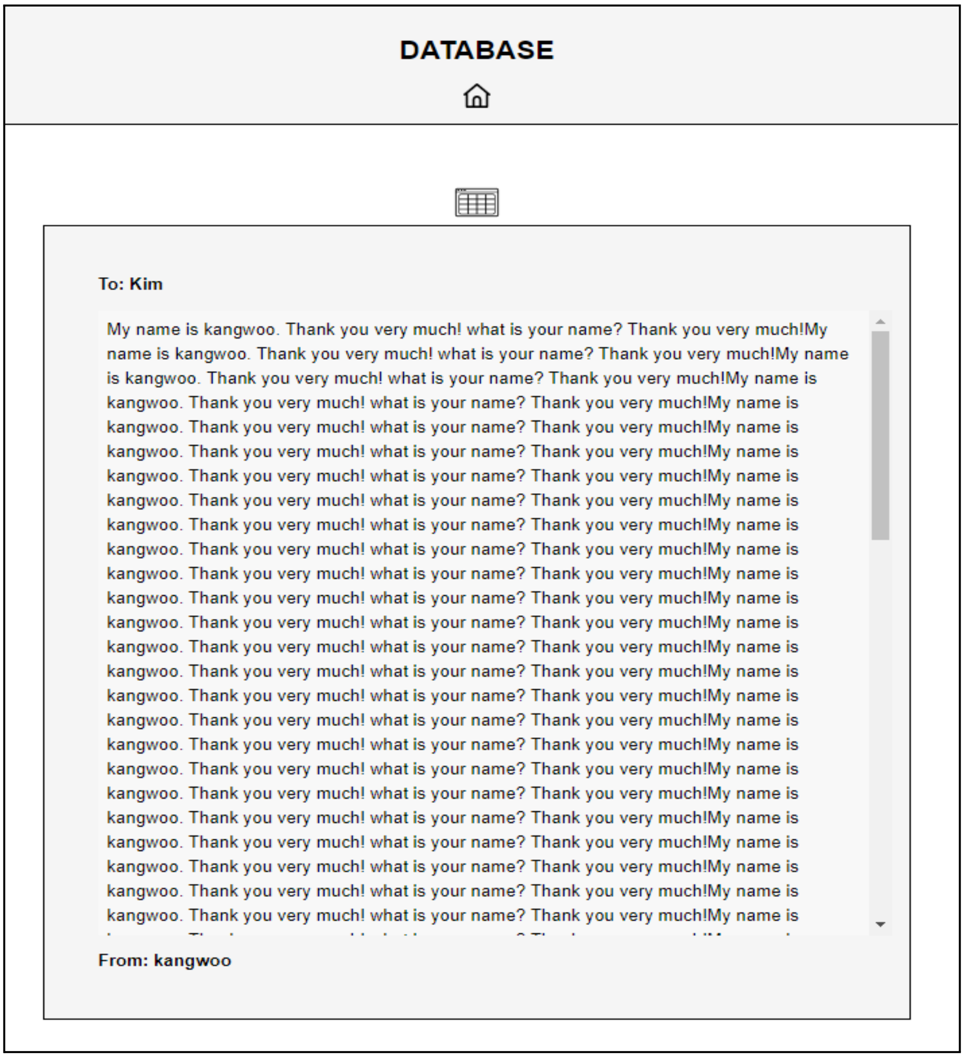
[그림 44] DB의 값들을 웹 페이지에 테이블로 시각화 하는 printTable함수

```javascript
function handleRowClick(row){
  var rowId = row.getAttribute("id");
  var idx = rowId.substring(4);
  var rowData = database.find(item => item.idx == idx);
  openDBMail(rowData);
}
```

[그림 45] 테이블의 행을 클릭 시 해당 행의 정보를 Mail출력 함수로 넘기는 handleRowClick 함수



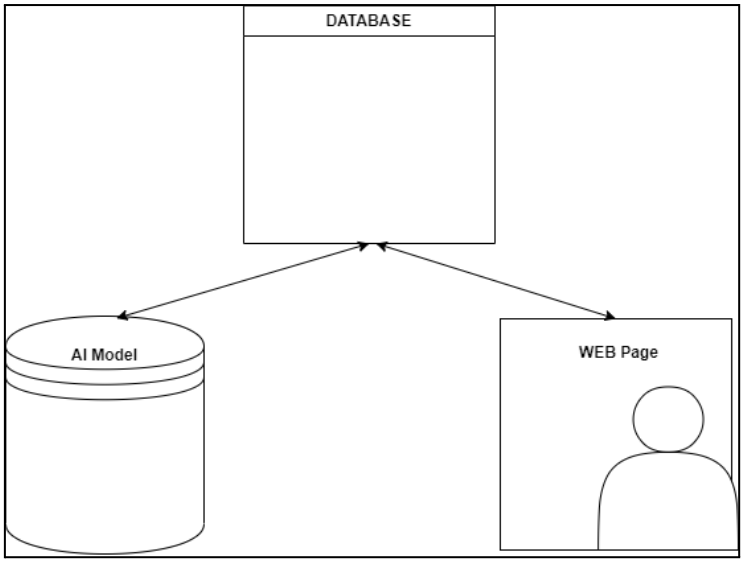
[그림 46] DB조회 웹페이지



[그림 47] DB요소 클릭 시 편지 형식으로 출력

기계학습이 끝난 모델에서 DB의 값을 가져오거나 클라이언트가 입력한 값을 input으로 받아 DB에 업로드 하기 위해 모든 권한을 허용받은 사용자를 생성하여 DB를 공유한다. GRANT ON TO 쿼리문을 이용해 DB를 같이 접근할 수 있으며, 이에 따라 개인 서버를 사용하지 않고 실시간 연동이 가능해진다.

[그림 48] 외부에서 접근을 허용받은 사용자 생성



[그림 49] 실시간 연동 구상도

6. 피드백 반영사항

- 원래 하려던 방법은 규칙 기반의 스팸 필터링 방식이었는데, 새로운 유형의 메일의 대처하고자 하는 본 졸업과제의 목적을 위해서 content-based 기반의 TF-IDF 방식을 활용한 모델을 선정하였다.

- kaggle 데이터 수집에 어려움이 있어 Enron Spam Dataset을 모델 학습에 사용하였다.

- 하이퍼 파라미터를 조정하여 Enron에 테스트 셋에 대한 정확도를 95 ~ 100% 까지 향상시켰다.

- 단순히 성능에 의존하는것이 아닌, 스팸 필터링에 사용할 각 모델의 이점을 이해하고 사용하였다.

- 성능을 향상시키기 위한 시도로 다양한 feature set을 생성하여 다중입력모델을 학습하여 각 모델의 성능을 비교실험 해보았다.

7. 결과

7.1 진행 상황 요약

(1) 모델 학습 및 검증을 위한 Enron e-mail dataset과 성능 비교실험을 위한 개인 메일함, Kaggle, 생성형 AI 서비스로 생성한 데이터 셋을 최종적으로 확보하였다.

(2) MLP, LSTM, SVM 모델에서의 학습을 위해 전처리과정을 거쳐 TF-IDF기반의 가중치 및 5개의 추가 입력 feature들을 생성하였다.

(3) 입력 feature 별로 MLP, LSTM, SVM 모델의 학습을 완료하였으며 각각의 성능을 비교하였다.

(4) 웹 페이지는 머신러닝의 간략한 소개, 클라이언트가 메일을 작성해 DB로 Insert하고 결과를 확인하는 기능을 구현하였고, DB조회는 조회기능과 클릭 시 메일 형식으로 조회 가능하다. 통일된 형식으로 맞추어 한 웹페이지가 이동하는 것처럼 보이게 하였다.

7.2 향후 진행 계획

(1) 생성형 AI 서비스를 이용하여 생성 스팸 데이터 셋을 추가로 생성할 예정이다.

(2) 현재, 1 Feature MLP와 6 Feature LSTM 중 하나를 선택해서 진행할 지, 둘 다 사용하여 성능비교를 할지 결정할 것이다.

(3) 추가로 생성한 데이터를 모델의 추가적인 학습에 사용하여 기존 모델과의 성능을 비교할 예정이다.

(4) 최종적으로 사용하게 될 모델에서 feature importance를 추출하여 웹 페이지에서 활용할 예정이다.

(5) 웹 페이지에서는 추가적인 chart.js의 그래프 모델을 이용해 이전에 비해 개선된 AI 모델의 성능을 사용자가 보기 쉽게 시각화하는 그래프를 추가할 예정이다.

(6) AI 모델의 출력값을 클릭할 필요 없이 실시간으로 웹페이지에 적용하기 위한 방법을 찾고 웹 에서는 비동기 요청인 AJAX나 주기적으로 서버에 요청을 보내는 Polling 방식을 이용해 구현할 예정이다.