코로나 관련 정보 및 확진자 예측

오픈소스 SW 개론 팀 프로젝트 20조

16013090 김종빈
16010969 허인주
15011191 남규헌

1. 주제
……1) 개요
……2) 프로젝트 목적

2. 사용 툴

3. 프로젝트 내용
……1) Interface
……2) 코로나 확진자 학습 모델링
……3) 코로나 확진자 예측

4. 참고자료 및 데이터

5. 역할 및 기여도

6. 주차별계획

2. 사용 툴

-Python : 가독성이 높고 쉬운 문법을 지닌 프로그래밍 언어로 빠른 습득이 가능. 데이터 분석과 모델링을 다루는 통계학부터 딥러닝과 인공지능을 활용하는 의학에까지 다양한 분야에 두루 활용.

-Numpy : 수치 데이터를 다루는 파이썬 패키지. Numpy의 핵심이라고 불리는 다차원 행렬 자료구조인 ndarray를 통해 벡터 및 행렬을 사용하는 선형 대수 계산에서 주로 사용.

-Matplotlib : 데이터를 차트(chart)나 플롯(plot)으로 시각화하는 패키지. 데이터 분석 이전에 데이터 이해를 위한 시각화나, 데이터 분석 후에 결과를 시각화하기 위해서 사용된다.

-Keras : 케라스(Keras)는 파이썬으로 작성된 오픈 소스 신경망 라이브러리이다. MXNet, Deeplearning4j, 텐서플로, Microsoft Cognitive Toolkit 또는 Theano 위에서 수행할 수 있다.[2][3] 딥 신경망과의 빠른 실험을 가능케 하도록 설계되었으며 최소한의 모듈 방식의 확장 가능성에 초점을 둔다.

-Atom : Github에서 만든 에디터로 다양한 프로그래밍 언어의 편집기로 사용할 수 있도록 고안된 도구입니다.

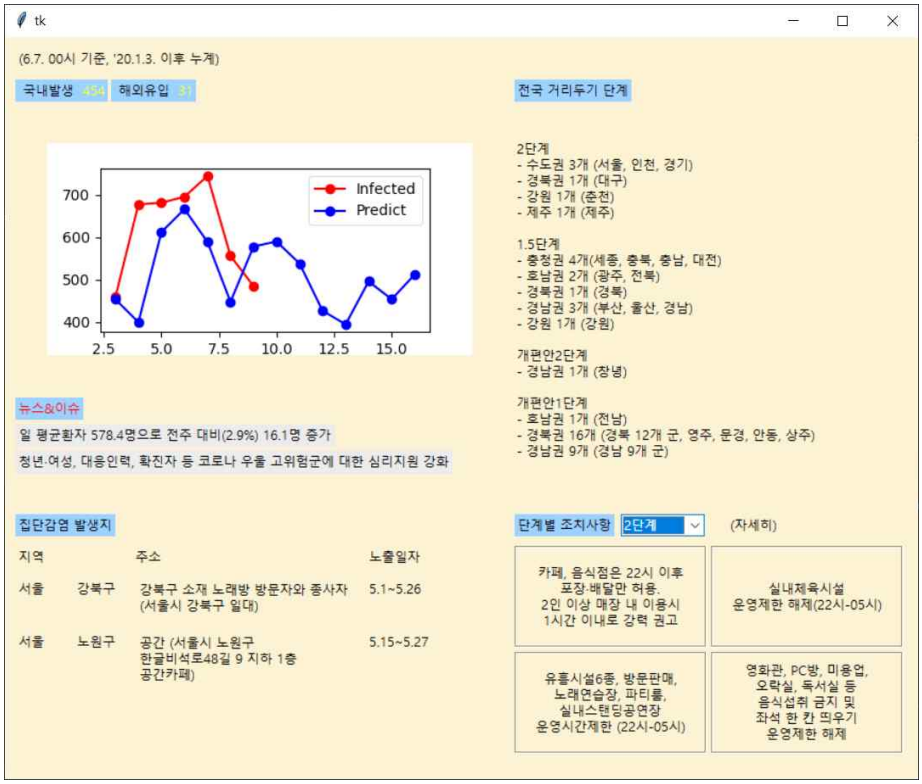
-Colaboratory(코랩) : 구글에서 교육과 과학 연구를 목적으로 개발한 클라우드 기반의 무료 Jupyter 노트북 개발 환경. 웹 브라우저를 통해 제어하고 실제 파이썬 코드 실행은 구글 클라우드의 가상 서버에서 이루어진다.

-Git, GitHub : 컴퓨터 파일의 변경사항을 추적하고 여러 명의 사용자들 간에 해당 파일들의 작업을 조율하기 위한 분산 버전 관리 시스템. 소프트웨어 개발에서 소스 코드 관리에 주로 사용되지만 어떠한 집합의 파일의 변경사항을 지속적으로 추적하기 위해 사용될 수 있다.

-Pandas : 판다스는 넘파이를 기반으로 만들어졌으며, 자체 데이터 타입인 시리즈(Series), 데이터프레임(DataFrame) 등을 제공하고 있는 데이터 조작 및 분석을 위한 소프트웨어 라이브러리로 Python 프로그래밍 언어로 작성되어 있습니다. 특히 수치 표와 시계열 을 조작하기 위한 데이터 구조와 연산을 제공합니다.

3. 프로젝트 내용

(1) Interface



Interface 주요 기능(추가)

1) 스크래핑을 통해 새로운 정보를 갱신

- 오늘의 확진자 정보 제공
- 최근 주요 뉴스 및 이슈를 알려주고 바로 기사를 확인 가능하게 함
- 집단감염 발생지를 알려줌
- 전국 거리두기 단계를 보여줌
- 단계별 조치사항을 보여줌

2) 모델을 통해 예측한 확진자를 그래프로써 나타냄

(2) 코로나 확진자 학습 모델링

1) 데이터 전처리

- csv 데이터

```
import pandas as pd
df = pd.read_csv('predict.csv')
df['Date'] = pd.to_datetime(df['Date'],format='%Y-%m-%d')
```

날짜, 검사자 수, spread, 사회적 거리두기 단계, 확진자 수

spread = 증상자*0.035+무증상자*0.008

	Date	Test_numb	spread	policy	Infected
2	2020-01-26	23	0.035	0	1
3	2020-01-27	9	0.035	0	1
4	2020-01-28	55	0	0	0
5	2020-01-29	71	0	0	0
6	2020-01-30	57	0	0	0
7	2020-01-31	57	0	0	0
8	2020-02-01	119	0.035	0	8
9	2020-02-02	55	0.035	0	3
10	2020-02-03	61	0	0	0
11	2020-02-04	116	0	0	1
12	2020-02-05	105	0.07	0	2
13	2020-02-06	166	0.035	0	5
14	2020-02-07	466	0	0	1
15	2020-02-08	745	0	0	0
16	2020-02-09	498	0.105	0	3
17	2020-02-10	512	0.035	0	0
18	2020-02-11	1214	0	0	1
19	2020-02-12	1299	0	0	0
20	2020-02-13	887	0	0	0

- 스케일링

각각의 데이터의 가중치를 맞추기 위해 스케일링 하였습니다.

```
from sklearn.preprocessing import MinMaxScaler

df.sort_index(ascending=False).reset_index(drop=True)

scaler = MinMaxScaler()
scale_cols = ['Test_number', 'spread', 'policy', 'Infected']
df_scaled = scaler.fit_transform(df[scale_cols])
df_scaled = pd.DataFrame(df_scaled)
df_scaled.columns = scale_cols
```

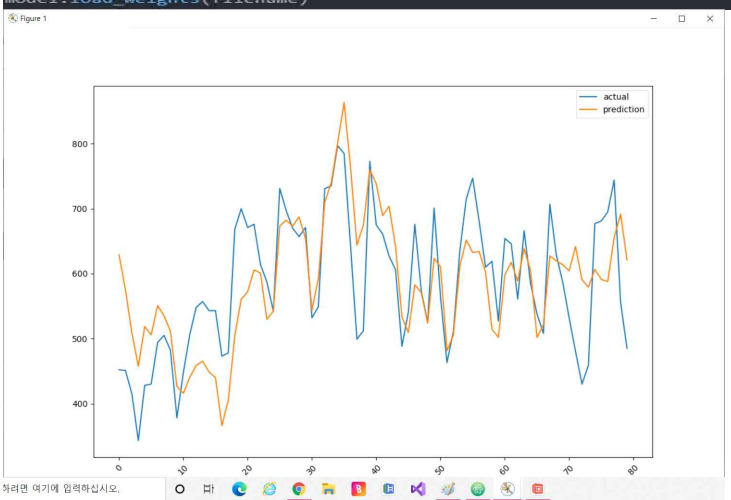
- 데이터 셋 분할

데이터를 학습할 Traning set과 검사할 Test set으로 나눔

```
def make_dataset(data, label, window_size=20):
    feature_list = []
    label_list = []
    for i in range(len(data) - window_size):
        feature_list.append(np.array(data.iloc[i:i+window_size]))
        label_list.append(np.array(label.iloc[i+window_size]))
    return np.array(feature_list), np.array(label_list)

def make_dataset2(data, window_size=20):
    label_list = []
    for i in range(len(data) - window_size):
        label_list.append(data[i:i+window_size])
    return np.array(label_list)

SIZE_RATE = 0.8
TEST_SIZE = int(len(df_scaled)*SIZE_RATE)
WINDOW_SIZE = 20

train = df_scaled[:TEST_SIZE]
test = df_scaled[TEST_SIZE:]
```

- LSTM 모델링

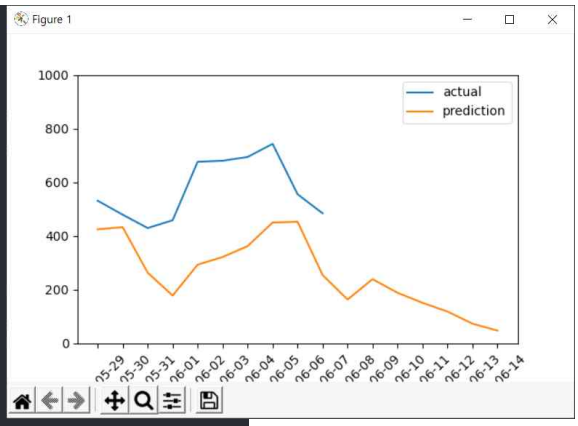
```
x_train, x_valid, y_train, y_valid = train_test_split(train_feature, train_label, test_size=0.2)
x_train.shape, x_valid.shape

test_feature = test[feature_cols]
test_label = test[label_cols]

test_feature.shape, test_label.shape

test_feature, test_label = make_dataset(test_feature, test_label, 20)
test_feature.shape, test_label.shape

model = Sequential()
model.add(LSTM(16,
               input_shape=(train_feature.shape[1], train_feature.shape[2]),
               activation='relu',
               return_sequences=False)
          )
model = Sequential()
model.add(LSTM(16,
               input_shape=(train_feature.shape[1], train_feature.shape[2]),
               activation='relu',
               return_sequences=False)
          )

model.add(Dense(1))

model.compile(loss='mean_squared_error', optimizer='adam')
early_stop = EarlyStopping(monitor='val_loss', patience=5)

model_path = 'model'
filename = os.path.join(model_path, 'tmp_checkpoint.h5')
checkpoint = ModelCheckpoint(filename, monitor='val_loss', verbose=1, save_best_only=True, mode='auto')

history = model.fit(x_train, y_train,
                                    epochs=200,
                                    batch_size=16,
                                    validation_data=(x_valid, y_valid),
                                    callbacks=[early_stop, checkpoint])


model.load_weights(filename)
```



LSTM 모델을 통해 확진자 데이터를 학습시킨 테스트 데이터를 나타냄

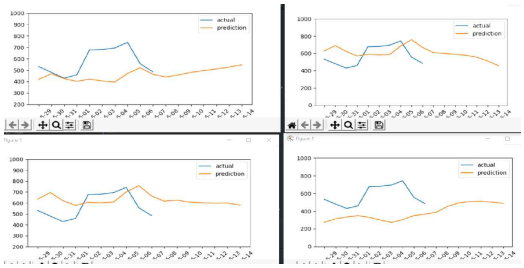
(3) 코로나 확진자 예측

```
pol = 2
t_p = np.array([[pol]])
t_p=t_p.astype(np.float)

for i in range(0,7):

    a = test_feature[[-1]]
    b = model.predict(a)

    t1_i = np.array(test[label_cols])
    t1_i=np.concatenate((t1_i, b))
    t1_i.shape
    t1_i = make_dataset2(t1_i, 20)
    t1_i.shape
    t1_i=t1_i[[-1]]

    t1_t = model2.predict(t1_i)
    t1_s = model3.predict(t1_i)

    arr = np.concatenate((t1_t, t1_s, t_p, b), axis=1)

    test = scaler.fit_transform(test[scale_cols])
    test = np.concatenate((test, arr))
    test = pd.DataFrame(test)
    test.columns = scale_cols

    test_feature = test[feature_cols]
    test_label = test[label_cols]

    test_feature.shape, test_label.shape

    test_feature, test_label = make_dataset(test_feature, test_label, 20)
    test_feature.shape, test_label.shape

pred = model.predict(test_feature)
pred.shape
```

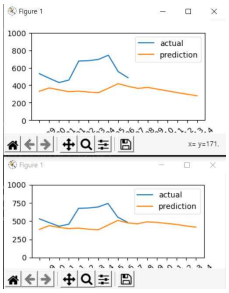


학습된 LSTM모델을 이용하여 내일의 확진자의 수를 예측하고 확진자를 통해 검사자 수와 spread값을 예측하는 모델을 학습시켜 내일의 확진자 수를 통해 내일의 검사자 수와 spread 값을 예측하였습니다.
policy(사회적 거리두기 단계)는 2로 고정하여 새로운 내일 예측 데이터 배열 arr를 만들어 기존 데이터에 추가하였으며 그렇게 추가된 데이터를 가지고 다시 그 다음날의 확진자 수를 예측하는 for문을 만들어 7일 후까지 예측하는 모델을 만들어 표현하였습니다.

사회적 거리두기 2단계



1단계



사회적 거리두기를 달리 설정하여 예측한 그래프 변화율 차이를 보여 현재 요구되는 사회적 거리두기를 제시할 수 있습니다.

(4) 참고자료 및 데이터

[경기도감염병관리지원단]

2020.01.26 부터 최근까지의 경기도 및 전국 코로나19 확진자 데이터를 csv파일로 저장

- "코로나19 경기도 확진환자" 로부터 경기도 확진환자의 증상/무증상 데이터 확보
- "코로나19 전국 현황" 으로부터 전국 검사자 및 확진자 수 데이터 확보
- 전국 데이터에는 없는 증상/무증상 수를 경기도의 증상/무증상 수 비율을 통해 전국 데이터에 적용하여 임의로 환산

http://www.gidcc.or.kr/%EC%BD%94%EB%A1%9C%EB%82%98covid-19-%ED%98%84%ED%99%A9/

[코로나바이러스감염증-19(COVID-19) 정식 홈페이지]

모델링 인풋 요소 중 하나인 거리두기 정책(단계)을 날짜별로 가져와 직접 csv에 추가

-코로나 관련 정보(일일 확진자 수, 뉴스&이슈, 집단 감염 발생지, 전국 거리두기 단계, 거리두기 단계별 조치사항)를 크롤링을 통해 수집

http://ncov.mohw.go.kr/

[LSTM을 이용한 삼성전자 주가 예측하기(블로그)]

LSTM을 통해 주가를 예측하는 것으로부터 아이디어 착안

- 주가를 예측할 때 필요한 요소인 ['시가', '고가', '저가', '거래량'] 대신 ['검사자 수', '확산정도(spread)', '정책'] 을 요소로 결과값 ['확진자 수'] 를 예측

https://coding-yoon.tistory.com/131

(5) 역할 및 기여도

조원	역할	기여도
김종빈	기획 / GUI / LSTM 모델링 / 코로나 정보 크롤링 / 발표	1/2
허인주	기획 / GUI / LSTM 모델링 / 데이터 전처리 / PPT	1/2
남규헌	기획	

(6) 주차별 계획

기간		세부 내용
1주차	4/6 – 4/12	프로젝트 관련 자료 수집
2주차	4/13 – 4/19	계획서 작성
3주차	4/20 – 4/26	계획서 발표 및 시험기간
4주차	4/27 – 5/3	파이썬 GUI틀 만들기
5주차	5/4 – 5/10	확진자 데이터 수집
6주차	5/11 – 5/17	확진자 수 예측 모델 만들기
7주차	5/18 – 5/24	에측 모델 GUI에 적용
8주차	5/25 – 5/31	최종 발표