

분기별 부산항 물동량(Container) Data를 이용한 2019년도 2분기 경제성장률(GDP) 예측

통계학과 / 201646114 / 박준형

1. 프로젝트 목적 및 동기

A. 주제 선정 배경 - Main Theme { 경제성장률(GDP) }

A-1. 2019년 1분기 경제성장률 붕괴

['마이너스 성장' <1분기 -0.3%> 쇼크에 빠진 대한민국 경제](#) 한국경제 | [한국경제 2019/04/29 中]

- 1분기 경제성장률(-0.3%)이 발표됨과 동시에 경기침체 우려의 목소리 多

A-2. 2분기 이후 반등할 것이라는 정부의 발표

[이호승 기재차관, "GDP 2분기 이후 반등할 것..."](#) [연합뉴스 2019/04/29 中]

- 과연 확률적으로 얼마나 유의할까에 대한 의문 有

A-3. 정부의 발표대로 2분기 경제성장률이 반등한다면, 과연 얼마나?

실제로 한은이 올 상반기 경제 성장률 전망치를 2.3%로 발표했는데, 이를 위해선 2분기에 전

기 대비 1.5%를 성장해야 한다. 1분기에 대한 기저 효과를 감안해도 현 상황에서 이 정도의 성 [헤럴드경제 2019/04/25 中]

- 상반기 전망치를 충족시키기 위한 2분기 최소 성장률인 +1.5%가 가능할까에 대한 의문 有

B. 주제 선정 배경 - Sub Theme (부산항 Container)

B-1. 경제성장률은 수/출입과 밀접.

B-2. 부산항 물동량(Container) Data & GDP Data가 공공데이터로써 연도별로 잘 정리되어 있다.

- Data set을 기간별로 나누어 Time-series 분석이 가능하다.

B-3. 부산항 물동량(Container) Data를 이용하여 GDP 예측 알고리즘을 만들 수 있다.

- 알고리즘을 통해 A-2, A-3의 의문을 어느 정도 해소하리라 기대.

2. 프로젝트 개요

1) 개별파일의 구성 : GDP_Ft, Container_Ft, GDP_Container_Ft, time_Series

- 또한, 각 파일 내의 함수도 그 기능에 따라 나누어 함수를 구성하였다.
- 수정 및 보안을 편리하게 하고, 가독성을 높이기 위하여

ex) 개별파일의 구성



ex) 각 파일 내에서 기능에 따른 함수구성

```
#####---GDP Function---#####
# GDP_Data 전처리 함수
def GDP_Pre() :
    Data_GDP = pd.read_csv('gdp.csv')
    New_Data_GDP = Data_GDP.set_index('Y_Q')
    return New_Data_GDP

# 전체 GDP 출력 함수
def GDP_All(GC) :
    New_Data_GDP=GDP_Pre()
```

2) 각 파일의 여러 함수를 Main 파일에서 모듈로 불러와서 사용

- 필요에 따라 함수 간 호출, 파일 간 호출도 강행

ex) Main file에서 import

```
import GDP_Ft as GF
import Container_Ft as CF
import GDP_Container_Ft as GCF
import time_series as TS
```

3) 사용자의 Command에 따라 각 함수가 작동하며 필요한 정보를 입력받고, 출력함

ex)

```
특정 년도/분기의 Container를 확인하시겠습니까 (Y/N) ?n
1분기보다 2분기의 경제성장률이 좋았던 적이 있는지 그래프로 확인하시겠습니까 (Y/N) ?n
Time_Series 모델로 적합한 적합 모델의 Time_Series 그래프와 2019년도 2분기 예측값을 확인하시겠습니까 (Y/N) ?n
2008년(경제대공황)을 제외하고 적합한 적합 모델의 Time_Series 그래프와 2019년도 2분기 예측값을 확인하시겠습니까 (Y/N) ?n
```

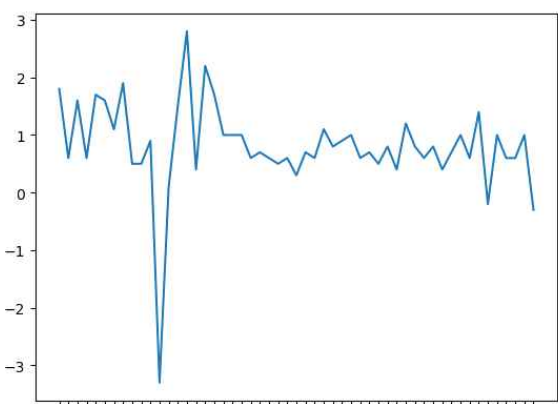
3. 프로젝트 구현 내용

3.1. 개발환경 (제목 2 스타일)

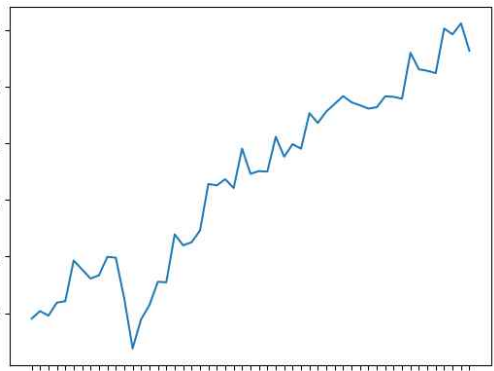
- Python (파이썬)
- Anaconda (아나콘다)
- Spyder 3 (스파이더 3)

3.2. 화면 구성도

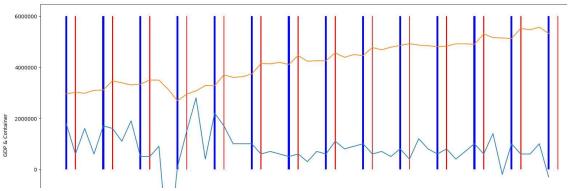
1) GDP Ft

<pre>2006_1분기부터 2019_1분기까지의 GDP를 확인하시겠습니까 (Y/N) ?y GDP Y_Q 2006_1 1.8 2006_2 0.6 2006_3 1.6 2006_4 0.6 2007_1 1.7 2007_2 1.6 2007_3 1.1 2018_4 1.0 2019_1 -0.3 특정 년도/분기의 GDP를 확인하시겠습니까 (Y/N) ?</pre>	
<p>1-(1)</p> <p>출력 물음에 대해 Y(or y)를 입력하면 전체 GDP를 출력 후 [다음 물음] 출력.</p>	<p>1-(2)</p> <p>출력 물음에 대해 Y(or y)를 입력하면 전체 GDP Graphic 출력 후 [다음 물음] 출력.</p>
<pre>2006_1분기부터 2019_1분기까지의 GDP를 확인하시겠습니까 (Y/N) ?h 잘못입력하셨습니다. 다시입력하세요 2006_1분기부터 2019_1분기까지의 GDP를 확인하시겠습니까 (Y/N) ?= 잘못입력하셨습니다. 다시입력하세요 2006_1분기부터 2019_1분기까지의 GDP를 확인하시겠습니까 (Y/N) ?g 잘못입력하셨습니다. 다시입력하세요 2006_1분기부터 2019_1분기까지의 GDP를 확인하시겠습니까 (Y/N) ?i</pre>	<pre>2006_1분기부터 2019_1분기까지의 GDP를 확인하시겠습니까 (Y/N) ?n 특정 년도/분기의 GDP를 확인하시겠습니까 (Y/N) ?y 연도_분기를 입력하시오(ex : '2012_3')2019_1 GDP -0.3 Name: 2019_1, dtype: float64</pre>
<p>1-(3)</p> <p>Y(y), N(n) 이외의 문자 입력 시 다시 입력하라는 메시지 출력</p>	<p>1-(4)</p> <p>출력 물음에 대해 N(or n)을 입력하면 [다음 물음] 출력. [다음 물음]은 사용자가 원하는 년도_분기를 입력받아 해당 GDP 출력</p>

2) GDP Ft

<pre>2006_1분기부터 2019_1분기까지의 Container를 확인하시겠습니까 (Y/N) ?y Container Y_Q 2006_1 2951069.0 2006_2 3017581.0 2018_4 5561434.0 2019_1 5319062.0 특정 년도/분기의 Container를 확인하시겠습니까 (Y/N) ?n</pre>	
<p>2-(1)</p> <p>출력 물음에 대해 Y(or y)를 입력하면 전체 Container를 출력 후 [다음 물음] 출력.</p>	<p>2-(2)</p> <p>출력 물음에 대해 Y(or y)를 입력하면 전체 Container Graphic 출력 후 [다음 물음] 출력.</p>
<pre>2006_1분기부터 2019_1분기까지의 Container를 확인하시겠습니까 (Y/N) ?d 잘못입력하셨습니다. 다시입력하세요 2006_1분기부터 2019_1분기까지의 Container를 확인하시겠습니까 (Y/N) ?3 잘못입력하셨습니다. 다시입력하세요</pre>	<pre>2006_1분기부터 2019_1분기까지의 Container를 확인하시겠습니까 (Y/N) ?n 특정 년도/분기의 Container를 확인하시겠습니까 (Y/N) ?y 년도_분기를 입력하시오(ex : '2012_3')2014_3 Container 4680962.0 Name: 2014_3, dtype: float64</pre>
<p>2-(3)</p> <p>Y(y), N(n) 이외의 문자 입력 시 다시 입력하라는 메시지 출력</p>	<p>2-(4)</p> <p>출력 물음에 대해 N(or n)을 입력하면 [다음 물음] 출력.</p>

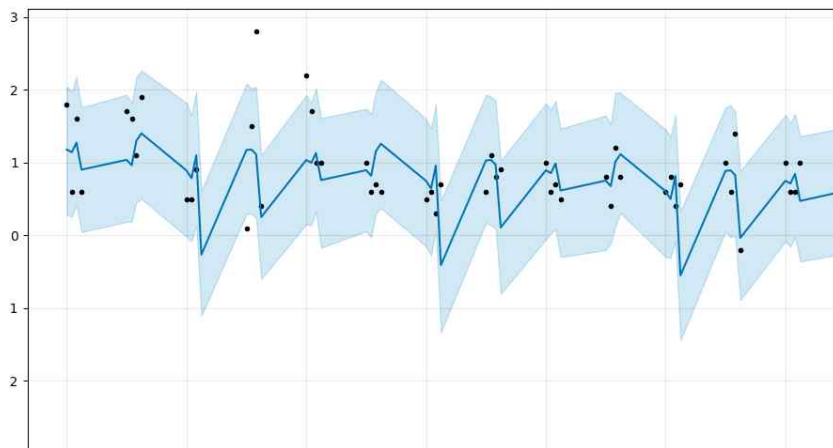
3) GDP_Container Ft & time_series

<pre>1분기보다 2분기의 경제성장률이 좋았던 적이 있는지 그래프로 확인하시겠습니까 (Y/N) ?y</pre>  <p>blue bar : 매년 1분기, red bar : 매년 2분기 blue line : GDP, red line : Container</p>	<pre>Time_Series 모델로 적합한 적합 모델의 Time_Series 그래프와 2019년도 2분기 예측값을 확인하시겠습니까 (Y/N) ?y 2019 2분기 예측 값(for 95% 신뢰도) = 83 0.532761 Name: yhat, dtype: float64 2019 2분기 예측 상한 값(for 95% 신뢰도) = 83 1.450642 Name: yhat_upper, dtype: float64 2019 2분기 예측 하한 값(for 95% 신뢰도) = 83 -0.385408 Name: yhat_lower, dtype: float64</pre>
<p>3-(1)</p> <p>출력 물음에 대해 Y(or y)를 입력하면 Graphic 출력 후 [다음 물음] 출력.</p>	<p>3-(2)</p> <p>출력 물음에 대해 Y(or y)를 입력하면 2019_2분기 GDP 예측값 출력 후 [다음 물음] 출력.</p>
<pre>2008년(경제대공황)을 제외하고 적합한 적합 모델의 Time_Series 2019 2분기 예측 값(for 95% 신뢰도) = 79 0.354715 Name: yhat, dtype: float64 2019 2분기 예측 상한 값(for 95% 신뢰도) = 79 1.020928 Name: yhat_upper, dtype: float64 2019 2분기 예측 하한 값(for 95% 신뢰도) = 79 -0.289056 Name: yhat_lower, dtype: float64</pre>	<p>칸이 작아서 아래에 첨부</p>
<p>3-(3)</p> <p>출력 물음에 대해 Y(or y)를 입력하면, 수정된 2019_2분기 GDP 예측값 출력 후 [다음 물음] 출력.</p>	<p>3-(4)</p>

[GDP 예측 Graphic]

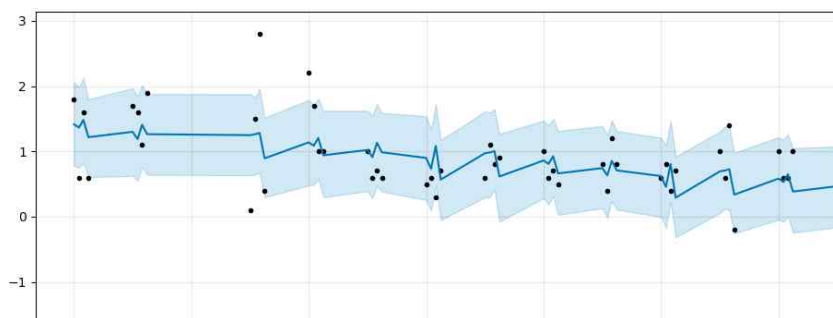
파란 실선: 예측 곡선, 하늘색 바탕: 예측 상한 및 하한, 검은 점: 실제 값

[수정 전]



```
2019 2분기 예측 값( for 95% 신뢰도 ) = 83    0.532761
Name: yhat, dtype: float64
2019 2분기 예측 상한 값( for 95% 신뢰도 ) = 83    1.450642
Name: yhat_upper, dtype: float64
2019 2분기 예측 하한 값( for 95% 신뢰도 ) = 83    -0.385408
Name: yhat_lower, dtype: float64
```

[수정 후] - 경제 대공황(2008년 Data 삭제 후)



```
2019 2분기 예측 값( for 95% 신뢰도 ) = 79    0.354715
Name: yhat, dtype: float64
2019 2분기 예측 상한 값( for 95% 신뢰도 ) = 79    1.020928
Name: yhat_upper, dtype: float64
2019 2분기 예측 하한 값( for 95% 신뢰도 ) = 79    -0.289056
Name: yhat_lower, dtype: float64
```

3.3. 기능 설명 (개별 File에 따른 함수를 나눠서 설명하겠습니다.)

[GDP_Ft]

표	내용구분	내용구분
import	import pandas as pd import matplotlib.pyplot as plt	
GDP 전처리	def GDP_Pre() : Data_GDP = pd.read_csv('gdp.csv') New_Data_GDP = Data_GDP.set_index('Y_Q') return New_Data_GDP	
전체 GDP 출력	def GDP_All(GC) : New_Data_GDP=GDP_Pre() if GC == 'Y' : print(New_Data_GDP) plt.plot(New_Data_GDP) elif GC == 'N' : return 1 else : print("잘못입력하셨습니다. 다시입력하세요") GC=input("2006_1분기부터 2019_1분기까지의 GDP를 확인하시겠습니까 (Y/N) ?").upper() GDP_All(GC)	
특정 GDP 출력	def GDP_Y_Q(GC): New_Data_GDP=GDP_Pre() if GC == 'Y' : Y_Q = input("연도_분기를 입력하시오(ex : '2012_3')") print(New_Data_GDP.loc[Y_Q]) return 1 elif GC == 'N' : return 1 else : print("잘못입력하셨습니다. 다시입력하세요") GDP_Confirm=input("특정 년도/분기의 GDP를 확인하시겠습니까 (Y/N) ?").upper() GDP_Y_Q(GDP_Confirm)	

[Container_Ft]

표	내용구분	내용구분
import	import pandas as pd import matplotlib.pyplot as plt	
Container 전처리	def Container_Pre() : # 필요한 칼럼만 불러오기 Data_Container = pd.read_csv('container.csv') Data_Container=Data_Container['TEU소계'] # 불필요한 Record 삭제 Data_Container=Data_Container.drop(0) i=1 while i <= 491 : Data_Container=Data_Container.drop(i) i = i + 37 # index 재정렬	

	<pre> Data_Container = Data_Container.reset_index(drop=True) # 필요한 Record만 추출 Data_Container=Data_Container.iloc[0:476:3] #index 재정렬 Data_Container = Data_Container.reset_index(drop=True) # 심표(.) 떼기 -> list로 반환됨 Data_Container = [float((item.replace(',',''))) for item in Data_Container] # 월별 Data를 분기별로 재정렬 i=0 while i <= 156 : Data_Container[i] = Data_Container[i] + Data_Container[i+1] + Data_Container[i+2] i = i + 3 # list를 DataFrame으로 만들기 Data_Container=pd.DataFrame(Data_Container) # 불필요한데이터 지우기 i=0 while i <= 156 : Data_Container.drop(i+1, inplace = True) Data_Container.drop(i+2, inplace = True) i = i + 3 # index 재정렬 Data_Container = Data_Container.reset_index(drop=True) # GDP Data에서 index로 쓸 columns 추출 Data_GDP = pd.read_csv('gdp.csv') Data_Container['Y_Q'] = Data_GDP['Y_Q'] # 추출한 columns을 index로 지정 Data_Container = Data_Container.set_index('Y_Q') New_Data_Container=Data_Container.rename(columns={0:'Container'}) return New_Data_Container </pre>	
전체 Container 출력	<pre> def Container_All(CC) : New_Data_Container=Container_Pre() if CC == 'Y' : print(New_Data_Container) plt.plot(New_Data_Container) elif CC == 'N' : return 1 else : print("잘못입력하셨습니다. 다시입력하세요") CC=input("2006_1분기부터 2019_1분기까지의 Container를 확인하시겠습니까 (Y/N) ?").upper() Container_All(CC) </pre>	
특정 Container 출력	<pre> def Container_Y_Q(CC): New_Data_Container=Container_Pre() if CC == 'Y' : Y_Q = input("연도_분기를 입력하시오(ex : '2012_3')") print(New_Data_Container.loc[Y_Q]) return 1 elif CC == 'N' : return 1 else : print("잘못입력하셨습니다. 다시입력하세요") </pre>	

	Container_Confirm=input("특정 년도/분기의 Container를 확인하시겠습니까 (Y/N) ?").upper() Container_Y_Q(Container_Confirm)	
--	--	--

[GDP_Container_Ft]

표	내용구분	내용구분
import	import pandas as pd import matplotlib.pyplot as plt import GDP_Ft as GF import Container_Ft as CF	
GDP & Container 하나로 병합	def GDP_Container_Pre(): GDP = GF.GDP_Pre() GDP_mult = 1000000*GF.GDP_Pre() Container = CF.Container_Pre() Data = pd.merge(GDP, Container, on='Y_Q') Data_mult = pd.merge(GDP_mult, Container, on='Y_Q') return Data, Data_mult	
Plot	def Data_Plot(PC): if PC == 'Y': Data, Data_mult = GDP_Container_Pre() plt.plot(Data_mult) plt.ylabel('년_분기') plt.ylabel('GDP & Container') xbar1 = ['2006_1','2007_1','2008_1','2009_1','2010_1','2011_1','2012_1', '2013_1','2014_1','2015_1','2016_1','2017_1','2018_1','2019_1'] ybar = [6000000] plt.bar(xbar1,ybar,width=0.2, color='blue') xbar2 = ['2006_2','2007_2','2008_2','2009_2','2010_2','2011_2','2012_2', '2013_2','2014_2','2015_2','2016_2','2017_2','2018_2','2019_2'] plt.bar(xbar2,ybar,width=0.1, color='red') elif PC == 'N': return 1 else: print("잘못입력하셨습니다. 다시입력하세요") Plot_Confrim = input("1분기보다 2분기의 경제성장률이 좋았던 적이 있는지 그래프로 확인하시겠습니까 (Y/N) ?").upper() Data_Plot(Plot_Confrim)	

[time_series]

표	내용구분	내용구분
import	import GDP_Container_Ft as GCF import pandas as pd import numpy as np import matplotlib.pyplot as plt from datetime import datetime from fbprophet import Prophet	
Time_Series 전처리	def Time_Series_Pre(): Data, temp = GCF.GDP_Container_Pre() dates = list(Data.index)	

	<pre> dates = [item.replace('_', '-') for item in dates] dates_index = pd.DatetimeIndex(dates) GDP=list(Data['GDP']) Col = ['GDP'] Time_Series = pd.DataFrame(data=GDP, index = dates_index, columns=Col) Time_Series['Container'] = list(Data['Container']) return Time_Series </pre>	
수정 전 예측	<pre> def All_Time(TS_Confrim) : if TS_Confrim == 'Y': Time_Series = Time_Series_Pre() # 시계열 예측을 위한 DataFrame으로 변환 Predict_GDP = pd.DataFrame({'ds':Time_Series.index, 'y':Time_Series['GDP']}) # Container Columns을 연도별로 GDP Columns에 적합 (Fitted) m=Prophet() m.fit(Predict_GDP) future = m.make_future_dataframe(periods=31) forecast = m.predict(future) forecast = forecast[forecast['ds'] == '2019-02-01'] print("2019 2분기 예측 값(for 95% 신뢰도) = ", forecast['yhat']) print("2019 2분기 예측 상한 값(for 95% 신뢰도) = ", forecast['yhat_upper']) print("2019 2분기 예측 하한 값(for 95% 신뢰도) = ", forecast['yhat_lower']) return Predict_GDP elif TS_Confrim == 'N' : return 1 else : print("잘못입력하셨습니다. 다시입력하세요") TS_Confrim = input("Time_Series 모델로 적합한 적합 모델의 Time_Series 그래프와 2019년도 2분기 예측값을 확인하시겠습니까 (Y/N) ?").upper() All_Time(TS_Confrim) </pre>	
수정 후 예측	<pre> def Del_Time(TS_Confrim) : if TS_Confrim == 'Y': Time_Series = Time_Series_Pre() # 2008년 1분기 Time_Series.drop(Time_Series.index[8], inplace = True) # 2008년 2분기 Time_Series.drop(Time_Series.index[8], inplace = True) # 2008년 3분기 Time_Series.drop(Time_Series.index[8], inplace = True) # 2008년 4분기 Time_Series.drop(Time_Series.index[8], inplace = True) # 시계열 예측을 위한 DataFrame으로 변환 Predict_GDP = pd.DataFrame({'ds':Time_Series.index, 'y':Time_Series['GDP']}) # Container Columns을 연도별로 GDP Columns에 적합 (Fitted) m=Prophet() m.fit(Predict_GDP) future = m.make_future_dataframe(periods=31) </pre>	

```
forecast = m.predict(future)
forecast = forecast[forecast['ds'] == '2019-02-01']
print("2019 2분기 예측 값( for 95% 신뢰도 ) = ", forecast['yhat'])
print("2019 2분기 예측 상한 값( for 95% 신뢰도 )
= ", forecast['yhat_upper'])
print("2019 2분기 예측 하한 값( for 95% 신뢰도 )
= ", forecast['yhat_lower'])

return Predict_GDP

elif TS_Confrim == 'N' :
    return 1
else :
    print("잘못입력하셨습니다. 다시입력하세요")
    TS_Confrim = input("2008년(경제대공황)을 제외하고 적합한 적합 모델의
Time_Series 그래프와 2019년도 2분기 예측값을 확인하시겠습니까
(Y/N ?").upper()
    Del_Time(TS_Confrim)
```

[Main]

표	내용구분	내용구분
import	import GDP_Ft as GF import Container_Ft as CF import GDP_Container_Ft as GCF import time_series as TS from fbprophet import Prophet	
GDP 확인	GDP_All=input("2006_1분기부터 2019_1분기까지의 GDP를 확인하시겠습니까 (Y/N) ?").upper() GF.GDP_All(GDP_All) GDP_Y_Q=input("특정 년도/분기의 GDP를 확인하시겠습니까 (Y/N) ?").upper() GF.GDP_Y_Q(GDP_Y_Q)	1- (1)(2) (3)(4)
Container 확인	Container_All=input("2006_1분기부터 2019_1분기까지의 Container를 확인하시겠습니까 (Y/N) ?").upper() CF.Container_All(Container_All) Container_Y_Q=input("특정 년도/분기의 Container를 확인하시겠습니까 (Y/N) ?").upper() CF.Container_Y_Q(Container_Y_Q)	2- (1) (2) (3) (4)
1&2분기 비교	Plot_Confrim = input("1분기보다 2분기의 경제성장률이 좋았던 적이 있는지 그래프로 확인하시겠습니까 (Y/N) ?").upper() GCF.Data_Plot(Plot_Confrim)	3- (1)
수정 전 Value & Graphic	TS_Confrim = input("Time_Series 모델로 적합한 적합 모델의 Time_Series 그래프와 2019년도 2분기 예측값을 확인하시겠습니까 (Y/N) ?").upper() if TS_Confrim == 'Y' : Predict_GDP = TS.All_Time(TS_Confrim) m=Prophet() m.fit(Predict_GDP) future = m.make_future_dataframe(periods=31) forecast = m.predict(future) m.plot(forecast)	3- (2) (4)
수정 후 Value & Graphic	TS_Confrim = input("2008년(경제대공황)을 제외하고 적합한 적합 모델의 Time_Series 그래프와 2019년도 2분기 예측값을 확인하시겠습니까 (Y/N) ?").upper() if TS_Confrim == 'Y' : Predict_GDP = TS.Del_Time(TS_Confrim) m=Prophet() m.fit(Predict_GDP) future = m.make_future_dataframe(periods=31) forecast = m.predict(future) m.plot(forecast)	3- (3) (4)

4. 프로젝트 개발 소감

이번 프로젝트를 하면서 가장 크게 느낀점은 프로젝트 개발순서에 관한 고찰입니다.

다른 학생들은 얼마의 시간이 걸렸는지 알 수 없지만, 저는 몇 번이고 수정하고, 또 수정하였고 결과적으로 중간쯤에 가서는 옳고 처음부터 다시 시작하였습니다.

처음 데이터를 python에 불러온 후 저는 무작정 키보드를 잡았습니다.

머릿속의 뒤엉킨 구상처럼 하려다 보니, 무엇을 함수로 만들어야 하고, 어떤 것을 모듈화시켜야 할지 막연했습니다. 또한, 그 결과 main 함수가 너무 복잡해지고 길어져 하루가 지나고 나면 제가 무엇을 코딩하였는지 알아보기조차 힘들었습니다. 하지만 막연함 속에서 계속 진행하였고 그 결과 프로젝트를 전면 수정하게 되었습니다. 수정할 때에 교수님의 말씀이 생각나서 하나하나 순서대로 a4 용지에 적어나가 보았습니다.

무엇을 file로 만들고 import 할지.

무엇을 function으로 만들고 사용할지.

그 결과 처음보다는 훨씬 좋은 결과물이 훨씬 짧은 시간에 효율적으로 만들어졌습니다.

이번 프로젝트를 하면서 제가 배우고, 느꼈던 가장 큰 것은 개발순서에 대한(혹은 구상한 것의 구체화)

것입니다. 내 머릿속의 뒤엉킨 생각과 알고리즘을 한 장의 종이에 일목요연하게 표현하는 것. 그것의 중요함을 느낄 수 있었던 프로젝트였습니다.

ps. 계절학기 재수강 때 만나요. 교수님. ♥