리눅스&하둡 분석 과제

1. Question

1) 베트남의 각 지방 도시들이 FDI (Foreign Direct Investment) 를 유치하는데 영향을 주는 요소들은 어떤 것이 있을까? (coef)
2) 검증된 요소들을 통해 베트남의 FDI유입을 예측해본다면 결과는 어떨까? (Liner regression)

Intro

베트남은 지난 10년간 ASEAN 지역에서 가장많은 FDI를 이끌어낸 나라 중 하나이며, 최근 5년간 top 3에 드는 FDI inflow를 유치하고 있다. 특히 경제개방을 시작한 이래 지금까지 베트남의 각 지방도시 63개 모두가 `적어도 1차례 이상`의 FDI를 유치한바가 있다.

베트남의 Vietnam Chamber of Commerce (VCCI) 와 미국의 USAID(United States Agency for International Development)는 매년 각 지방정부의 경쟁력 지수인 PCI(Provincial Competitiveness Index)를 발표하고 있다.

이러한 점에 착안하여 각 지방정부의 PCI지수가 FDI 유치에 어떠한 영향을 끼치는지 알아보고자 한다.

---

2. 데이터 구하기

분석을 위한 체계화된 데이터 셋이 존재하지 않기 때문에 각각의 데이터를 구해서 mysql을 통해 csv파일로 변환하는 작업을 거쳤다.

💡 1) ASEAN 국가들의 최근 10년 FDI 유치 금액

Resource : https://data.aseanstats.org/indicator/FDI.AMS.TOT.INF

💡 2) 베트남 지방정부 62개의 10년간 PCI 데이터

Resource : https://pcivietnam.vn/en 의 사이트에서 제공하는 연도별 dataset 사용

PCI데이터에는 종합 PCI점수를 계산하기 위해 총 10개의 sub-index를 계산한뒤 가중치를 적용하여 최종적으로 PCI 값을 산출한다.

이 과정을 maria db로 시행하였다.

분석을 위해 필요한 가장 중요한 자료로서 해당 자료에 대한 설명을 아래에 첨부한다.

👉 PCI 테이블에 들어가는 sub-index들

```
PCI : 아래의 10개의 데이터에 가중치를 적용하여 계산된 값. 각 지방정부의 경쟁력을 나타내며 100점 만점 기준

ENTRY_COST : 기업이 창업에 필요한 모든 면허를 등록하고 토지를 취득하고 수령하는 데 걸리는 시간, 필요한 면허의 수,
모든 면허와 허가를 취득하기 위해 필요한 난이도를 어떻게 인지하고 있는지 측정한 지표.

LAND_ACCESS : 기업들이 토지와 관련된 2가지 문제를 기준으로 측정. 즉 토지에 접근하기가 얼마나 쉬운가, 토지를 획득하면
종신 재직권이 보장되는가를 평가한 지표. (베트남은 사회주의국가로 토지에대한 '이용권' 만을 구매가능)

TRANSPARANCY: 기업이 사업을 운영하는 데 필요한 적절한 계획 및 법률 문서에 접근할 수 있는지 여부, 해당 문서를 공정하게
이용할 수 있는지 여부, 새로운 정책과 법률을 기업에 전달하고 예측 가능하게 구현하는지 여부, 도 웹 페이지의 사업 효용성에 대한 측정치.

TIME_COST: 지방 정부의 규제기관에 의해 검사가 시행될때 기업들이 얼마나 자주 & 오래 영업을 중단해야 하는지, 기업들이 행정절차를 따르는데 얼마나 많은
시간이 소모되는지를 측정한 것

INFORMAL_CHARGES: 기업들이 사업이외에 비공식적으로 사용되는 금액. 추가수수료가 사업 운영에 얼마나 부담이 되는지, 해당 지급으로 얼마나 원하는
'아웃풋'을 얻을 수 있는지 여부, 지방 공무원이 기업으로부터 임대료를 받기 위해 규정을 준수하는지에 대한 인식을 측정한 것

PROACTIVITY: 지방 정부가 민간기업을 지원하기위해 얼마나 적극적이고 창의적으로 정책을 시행하고, 새로운 정책을 입법하는지에 대한 기업들의 평가를 수치화 한 것.
```

```
BUSINESS_SUPPORT: 민간 부문 무역 촉진, 기업에 대한 규제 정보 제공, 사업 파트너 중매 및 기업에 대한 기술 서비스의 척도를 의미

LABOR_POLICY: 지방정부가 지역산업에 대한 직업훈련과 기술개발을 촉진하고 지역노동자 배치를 지원하기 위한 노력을 수치화한 것.

LAW: 기업이 지방정부의 법률기관을 분쟁 해결을 위한 유용한 수단으로 간주하는지, 공무원의 부패 행위에 대한 항소를 제기하는 수단으로 신뢰하는지 등
지방법률기관에 대한 기업들의 신뢰도.
```

💡 3) 베트남 지방정부 63개 지역의 10년간 FDI inflow 데이터

Resource : https://www.gso.gov.vn/en/investment-and-construction/
에서 제공하는 Statistical yearbook 2010 ~ 2019까지의 pdf 파일에서 'Foreign direct investment projects licensed in 20## by province' 부분의 도표에서 자료 발췌하여 csv 파일로 제작.

💡 4) 베트남 지방정부 63개의 10년간 Market size, population , labor quality 데이터

Resource : https://www.gso.gov.vn/en/px-web/?pxid=E0203-07&theme=Population and Employment (인구수)

Resource : https://www.gso.gov.vn/en/px-web/?pxid=E0802&theme=Trade%2C Price and Tourist (시장크기)

Resource : https://www.gso.gov.vn/en/px-web/?pxid=E0234&theme=Population and Employment (노동 인구)

```
MKS(Market Size) : 각 도시들에서 유통되고 있는 Goods and Services 의 retail prices(USD). 자금규모를 통해 각 지방의 시장크기를 나타낸다.

POP (Population) : 각 도시들의 총 인구수. 일반적으로 인구수는 시장크기, 노동력, 각종 infrastructure 의 발전정도와 연관이 깊다.

LAQU(Labor Quantity) : 각 도시별 15세이상 노동이 가능한 인구(=노동력)

LAQT(Labor Quality) : 각 도시별 훈련받은 노동가능 인구의 비율 (고급인력의 비중)
```

---

3. 마리아 db에 테이블로 생성(기본적인 데이터 테이블 생성)

```
생성할 데이터 종류 (2010~2019 기준)
1) ASEAN의 국가별 FDI inflow에 대한 데이터
2) 63개 지방도시별 FDI inflow 건수, 금액
3) 63개 지방도시별 PCI 지수(sub-index 19개) 에 대한 데이터(각 sub-index 에 가중치를 적용하여 계산)
4) 63개 지방도시별 시장크기, 인구수, 노동력, 노동품질 에 대한 데이터
```

테이블 입력에 앞서 PCI와 FDI 라는 단어가 들어간 모든 테이블을 DB에서 삭제

```
#PCI 테이블 삭제
set @table_name=null; # table_name을 null 값으로 변수 선언
select group_concat(table_schema,'.',table_name)
into @table_name from information_schema.tables #select 한 결과를 @table_name에 넣는다.
where table_schema = 'test_tables' and table_name like '%PCI%'; #table_name중에 PCI 를 포함한 것들 선택

#FDI 테이블 삭제
set @table_name=null; # table_name을 null 값으로 변수 선언
select group_concat(table_schema,'.',table_name)
into @table_name from information_schema.tables #select 한 결과를 @table_name에 넣는다.
where table_schema = 'test_tables' and table_name like '%FDI%'; #table_name중에 PCI 를 포함한 것들 선택
```

---

1) ASEAN 의 FDI inflow에 대한 데이터

💡 테이블 생성

```
drop table ASEAN_FDI;
create table ASEAN_FDI
(
YEAR INT(20),
COUNTRY VARCHAR(50),
FDI bigint(50) );
```

💡 생성한 테이블에 csv 파일 load

```
LOAD DATA LOCAL INFILE "/home/scott/hadoop_PJ/ASEAN_FDI_INFLOW.csv"
REPLACE
INTO TABLE PJ_01.ASEAN_FDI
fields TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(YEAR,COUNTRY, FDI);
```

💡 전체 데이터중 top5 국가들에 대한 데이터만 추려서 별도 테이블 생성

```
# 각 연도별 TOP5 데이터 담을 테이블 생성
DROP TABLE TOP5_ASEAN;
CREATE TABLE TOP5_ASEAN
(
YEAR int(50),
COUNTRY varchar(50),
FDI BIGINT(50),
RANK INT(10) );
```

💡 생성한 테이블에 데이터 담기

```
#데이터 담기
INSERT INTO TOP5_ASEAN
SELECT D.YEAR, D.COUNTRY, D.FDI, D.RANK
FROM (select YEAR, COUNTRY,FDI, DENSE_RANK() OVER (ORDER BY FDI DESC) AS RANK
  FROM ASEAN_FDI
    WHERE YEAR = 2010
    GROUP BY COUNTRY, YEAR
    ORDER BY RANK ASC)D
    WHERE RANK BETWEEN 1 AND 5
UNION ALL
SELECT D.YEAR, D.COUNTRY, D.FDI, D.RANK
FROM (select YEAR, COUNTRY,FDI, DENSE_RANK() OVER (ORDER BY FDI DESC) AS RANK
  FROM ASEAN_FDI
    WHERE YEAR = 2011
    GROUP BY COUNTRY, YEAR
    ORDER BY RANK ASC)D
    WHERE RANK BETWEEN 1 AND 5
UNION ALL
SELECT D.YEAR, D.COUNTRY, D.FDI, D.RANK
FROM (select YEAR, COUNTRY,FDI, DENSE_RANK() OVER (ORDER BY FDI DESC) AS RANK
  FROM ASEAN_FDI
    WHERE YEAR = 2012
    GROUP BY COUNTRY, YEAR
    ORDER BY RANK ASC)D
    WHERE RANK BETWEEN 1 AND 5
UNION ALL
```

```
SELECT D.YEAR, D.COUNTRY, D.FDI, D.RANK
FROM (select YEAR, COUNTRY,FDI, DENSE_RANK() OVER (ORDER BY FDI DESC) AS RANK
  FROM ASEAN_FDI
    WHERE YEAR = 2013
    GROUP BY COUNTRY, YEAR
    ORDER BY RANK ASC)D
    WHERE RANK BETWEEN 1 AND 5
UNION ALL
SELECT D.YEAR, D.COUNTRY, D.FDI, D.RANK
FROM (select YEAR, COUNTRY,FDI, DENSE_RANK() OVER (ORDER BY FDI DESC) AS RANK
  FROM ASEAN_FDI
    WHERE YEAR = 2014
    GROUP BY COUNTRY, YEAR
    ORDER BY RANK ASC)D
    WHERE RANK BETWEEN 1 AND 5
UNION ALL
SELECT D.YEAR, D.COUNTRY, D.FDI, D.RANK
FROM (select YEAR, COUNTRY,FDI, DENSE_RANK() OVER (ORDER BY FDI DESC) AS RANK
  FROM ASEAN_FDI
    WHERE YEAR = 2015
    GROUP BY COUNTRY, YEAR
    ORDER BY RANK ASC)D
    WHERE RANK BETWEEN 1 AND 5
UNION ALL
SELECT D.YEAR, D.COUNTRY, D.FDI, D.RANK
FROM (select YEAR, COUNTRY,FDI, DENSE_RANK() OVER (ORDER BY FDI DESC) AS RANK
  FROM ASEAN_FDI
    WHERE YEAR = 2016
    GROUP BY COUNTRY, YEAR
    ORDER BY RANK ASC)D
    WHERE RANK BETWEEN 1 AND 5
UNION ALL
SELECT D.YEAR, D.COUNTRY, D.FDI, D.RANK
FROM (select YEAR, COUNTRY,FDI, DENSE_RANK() OVER (ORDER BY FDI DESC) AS RANK
  FROM ASEAN_FDI
    WHERE YEAR = 2017
    GROUP BY COUNTRY, YEAR
    ORDER BY RANK ASC)D
    WHERE RANK BETWEEN 1 AND 5
UNION ALL
SELECT D.YEAR, D.COUNTRY, D.FDI, D.RANK
FROM (select YEAR, COUNTRY,FDI, DENSE_RANK() OVER (ORDER BY FDI DESC) AS RANK
  FROM ASEAN_FDI
    WHERE YEAR = 2018
    GROUP BY COUNTRY, YEAR
    ORDER BY RANK ASC)D
    WHERE RANK BETWEEN 1 AND 5
UNION ALL
SELECT D.YEAR, D.COUNTRY, D.FDI, D.RANK
FROM (select YEAR, COUNTRY,FDI, DENSE_RANK() OVER (ORDER BY FDI DESC) AS RANK
  FROM ASEAN_FDI
    WHERE YEAR = 2019
    GROUP BY COUNTRY, YEAR
    ORDER BY RANK ASC)D
    WHERE RANK BETWEEN 1 AND 5;

#출력하여 확인
SELECT * FROM TOP5_ASEAN;
```

2) 베트남 각 도시별 10년간 전체 FDI 프로젝트 건수 , 총 금액에 대한 테이블 생성

💡 각 연도별 베트남 각 지방의 FDI 총액 담을 테이블 생성

```
DROP TABLE FDI_BY_PROVINCE_2010;
create table FDI_BY_PROVINCE_2010
(PROVINCE varchar(50),
YEAR int(20),
FDI int(100)
);
```

각 생성한 테이블에 데이터 로드

```
LOAD DATA LOCAL INFILE "/home/scott/hadoop_PJ/2010_FDI_by_province.csv"
REPLACE
INTO TABLE PJ_01.FDI_BY_PROVINCE_2010
fields TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(PROVINCE, YEAR, FDI);
```

생성한 테이블 연결하여 담을 테이블 FDI 생성

```
DROP TABLE FDI;
create table FDI
(PROVINCE varchar(50),
YEAR int(20),
FDI int(100)
);
```

10년간 데이터 테이블 연결하여 FDI에 입력

```
INSERT INTO FDI
select *  from FDI_BY_PROVINCE_2010
UNION ALL
select *  from FDI_BY_PROVINCE_2011
UNION ALL
select *  from FDI_BY_PROVINCE_2012
UNION ALL
select *  from FDI_BY_PROVINCE_2013
UNION ALL
select *  from FDI_BY_PROVINCE_2014
UNION ALL
select *  from FDI_BY_PROVINCE_2015
UNION ALL
select *  from FDI_BY_PROVINCE_2016
UNION ALL
select *  from FDI_BY_PROVINCE_2017
UNION ALL
select *  from FDI_BY_PROVINCE_2018
UNION ALL
select *  from FDI_BY_PROVINCE_2019;
```


지난 10년간 베트남 각 지방의 총 FDI 유입건수, 총액 입력할 테이블 생성

```
drop table Accum_FDI;
create table Accum_FDI
(
PROVINCE VARCHAR(50),
Num_of_FDI INT(20),
FDI bigint(50) );
```


생성한 테이블에 csv 파일 load

```
LOAD DATA LOCAL INFILE "/home/scott/hadoop_PJ/accum_FDI.csv"
REPLACE
INTO TABLE PJ_01.Accum_FDI
fields TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(PROVINCE,Num_of_FDI, FDI);
```

3) 베트남 지방정부들의 10년치 PCI 데이터(sub-index 10개) + 총합 PCI점수를 담을 테이블 생성

💡 PCI의 SUB-INDEX에 가중치를 계산하기 위해, 각 INDEX별 가중점수가 입력된 테이블(wt_pci_value) 생성 및 CSV데이터 로드

```
create table wt_pci_value
(
chart varchar(20),
ENTRY_COST int(20),
LAND_ACCESS int(20),
TRANSPARANCY int(20),
TIME_COST  int(20),
INFORMAL_CHARGES int(20),
PROACTIVITY  int(20),
BUSINESS_SUPPORT   int(20),
LABOR_POLICY   int(20),
LAW  int(20),
TOTAL int(20) );
```

💡 DATA LOAD

```
LOAD DATA LOCAL INFILE "/home/scott/hadoop_PJ/weighted_values.csv"
REPLACE
INTO TABLE PJ_01.wt_pci_value
fields TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(chart,ENTRY_COST, LAND_ACCESS, TRANSPARANCY, TIME_COST, INFORMAL_CHARGES, PROACTIVITY, BUSINESS_SUPPORT, LABOR_POLICY, LAW, TO
```

💡 연도별로 PCI_BY_PROVINCE_#### ←##연도

CSV 파일이 연도별로 존재하여 각 데이터를 테이블에 LOAD한뒤 최종적으로 UNION ALL 하여 하나의 테이블로 합칠 예정

```
create table PCI_BY_PROVINCE_#### <-####(연도) 부분은 2010~2019로 수정하여 총 10개의 테이블 생성
(
PROVINCE varchar(20),
YEAR int(20),
ENTRY_COST float(20),
LAND_ACCESS float(20),
TRANSPARANCY float(20),
TIME_COST float(20),
INFORMAL_CHARGES float(20),
PROACTIVITY float(20),
BUSINESS_SUPPORT float(20),
LABOR_POLICY float(20),
LAW float(20) ) ;
```

💡 각각의 테이블에 CSV 데이터 삽입 (연도별로 총 10번 반복)

```
LOAD DATA LOCAL INFILE "/home/scott/hadoop_PJ/####_PCI_by_province.csv"
REPLACE
INTO TABLE PJ_01.PCI_BY_PROVINCE_####
```

```
fields TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(PROVINCE, YEAR, ENTRY_COST, LAND_ACCESS, TRANSPARANCY, TIME_COST, INFORMAL_CHARGES, PROACTIVITY, BUSINESS_SUPPORT,
 LABOR_POLICY, LAW  );
```

10개의 테이블 결과를 union all로 연결하여 담을 테이블 (PCI2 생성)

```
create table PCI2
(
PROVINCE varchar(20),
YEAR int(20),
ENTRY_COST float(20),
LAND_ACCESS float(20),
TRANSPARANCY float(20),
TIME_COST float(20),
INFORMAL_CHARGES float(20),
PROACTIVITY float(20),
BUSINESS_SUPPORT float(20),
LABOR_POLICY float(20),
LAW float(20) ) ;
```

연도별 테이블을 연결하여 PCI2에 삽입

```
INSERT INTO PCI2
select *  from PCI_BY_PROVINCE_2010
UNION ALL
select *  from PCI_BY_PROVINCE_2011
UNION ALL
select *  from PCI_BY_PROVINCE_2012
UNION ALL
select *  from PCI_BY_PROVINCE_2013
UNION ALL
select *  from PCI_BY_PROVINCE_2014
UNION ALL
select *  from PCI_BY_PROVINCE_2015
UNION ALL
select *  from PCI_BY_PROVINCE_2016
UNION ALL
select *  from PCI_BY_PROVINCE_2017
UNION ALL
select *  from PCI_BY_PROVINCE_2018
UNION ALL
select *  from PCI_BY_PROVINCE_2019;
```

PCI2와 wt_pci_value 의 데이터를 통해 가중치를 계산한 가중치를 담을 테이블 (PCI3) 생성 - 컬럼 pci2와 동일

```
create table PCI3
(
PROVINCE varchar(20),
YEAR int(20),
ENTRY_COST float(20),
LAND_ACCESS float(20),
TRANSPARANCY float(20),
TIME_COST float(20),
INFORMAL_CHARGES float(20),
PROACTIVITY float(20),
BUSINESS_SUPPORT float(20),
LABOR_POLICY float(20),
LAW float(20) ) ;
```

PCI2와 wt_pci_value 를 통해 가중치 계산하여 PCI3에 입력

```
INSERT INTO PCI3
SELECT P.PROVINCE, P.YEAR, (P.ENTRY_COST*wt.ENTRY_COST)/10 AS ENT_COST,
(P.LAND_ACCESS*wt.LAND_ACCESS)/10 AS LAND_ACCESS,
(P.TRANSPARANCY*wt.TRANSPARANCY)/10 AS TRANSPARANCY,
(P.TIME_COST*wt.TIME_COST)/10 AS TIME_COST,
(P.INFORMAL_CHARGES*wt.INFORMAL_CHARGES)/10 AS INFORMAL_CHARGES,
(P.PROACTIVITY*wt.PROACTIVITY)/10 AS PROACTIVITY,
(P.BUSINESS_SUPPORT*wt.BUSINESS_SUPPORT)/10 AS BUSINESS_SUPPORT,
```

```
(P.LABOR_POLICY*wt.LABOR_POLICY)/10 AS LABOR_POLICY,
(P.LAW*wt.LAW)/10 AS LAW
  FROM PCI2 P, wt_pci_value wt;
```

PCI2에 계산된 수치들의 합이 입력될 PCI 컬럼 추가

```
ALTER TABLE PCI2 ADD PCI FLOAT(50);
ALTER TABLE PCI2 ADD FDI FLOAT(50);
```

최종 결과를 담을 테이블 PCI 생성

```
DROP TABLE PCI;
create table PCI
(
PROVINCE varchar(20),
YEAR int(20),
PCI float(50),
FDI float(50),
MKS float(50),
POP float(50),
LAB float(20),
ENTRY_COST float(20),
LAND_ACCESS float(20),
TRANSPARANCY float(20),
TIME_COST float(20),
INFORMAL_CHARGES float(20),
PROACTIVITY float(20),
BUSINESS_SUPPORT float(20),
LABOR_POLICY float(20),
LAW float(20) ) ;
```

---

4) 각 지방도시별 시장관련 지수(Market_size, population, Labor_Quality)

💡 테이블 생성

```
DROP TABLE Eco_value;
create table Eco_value
(
PROVINCE varchar(20),
YEAR int(20),
MKS float(50),
POP float(50),
LAB float(20) ) ;
```

💡 DATA LOAD

```
LOAD DATA LOCAL INFILE "/home/scott/hadoop_PJ/Eco_value_by_province.csv"
REPLACE
INTO TABLE PJ_01.Eco_value
fields TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 LINES
(YEAR, PROVINCE,MKS, POP, LAB );
```

5) 위의 2)~4) 에서 만든 PCI, FDI, Economic 데이터 병합

1) 에서 만든 PCI에 FDI, MKS, POP, LAB 컬럼을 추가하고, 3)에서 만든 FDI테이블의 FDI, 4)에서 만든 경제지표들 추출하여 입력

각 SUB-INDEX의 값을 더하여 PCI컬럼으로서 PCI2테이블에 입력 + FDI 값도 합산하여 입력

```
INSERT INTO PCI
SELECT P.PROVINCE, P.YEAR,ROUND((P.ENTRY_COST+P.LAND_ACCESS+P.TRANSPARANCY+P.TIME_COST+P.INFORMAL_CHARGES+
P.PROACTIVITY+P.BUSINESS_SUPPORT+P.LABOR_POLICY+P.LAW),2) AS PCI, E.MKS, E.POP, E.LAB,F.FDI, P.ENTRY_COST, P.LAND_ACCESS, P.TRA
P.INFORMAL_CHARGES, P.PROACTIVITY, P.BUSINESS_SUPPORT, P.LABOR_POLICY, P.LAW
  FROM PCI3 P, FDI F, Eco_value E;
```

---

4. 파이썬에서 시각화 한다.

👉 시각화할 데이터 list

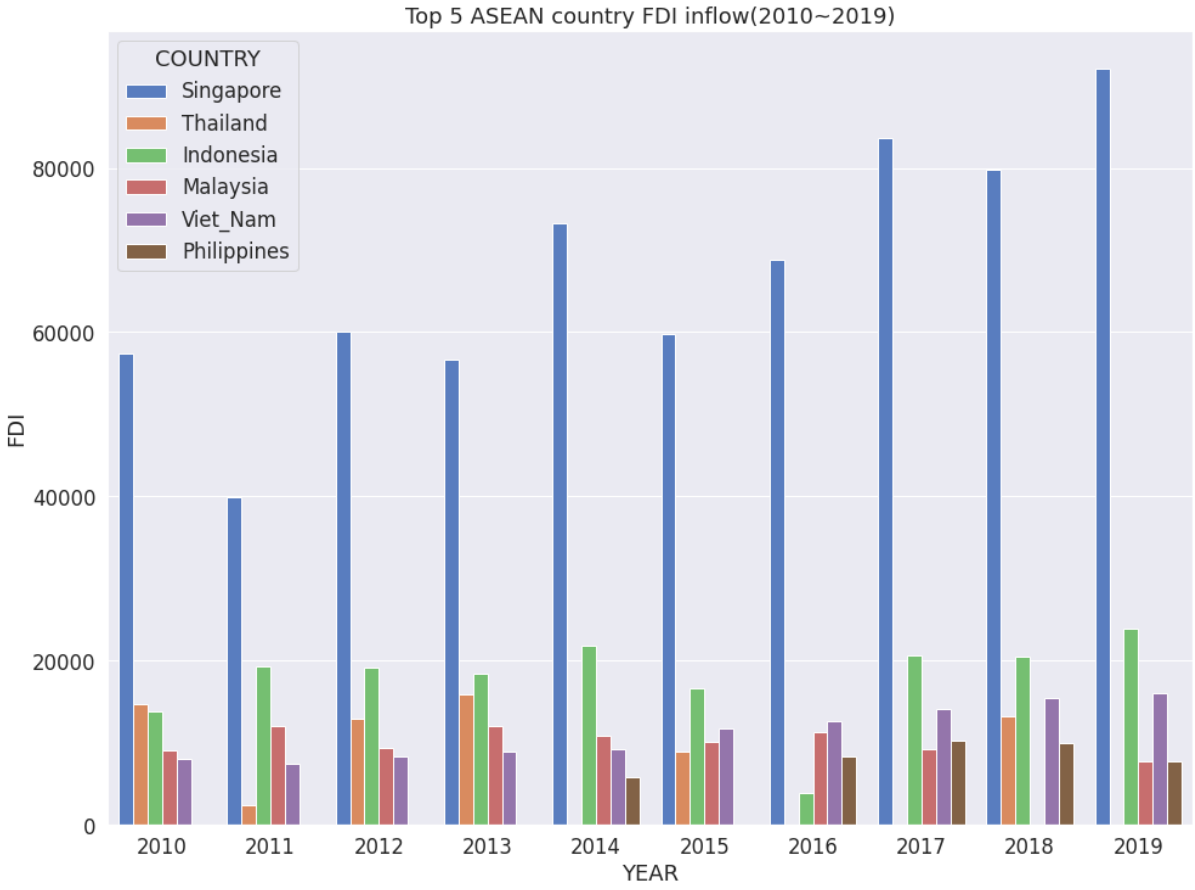
```
1. 10년간 ASEAN 국가들의 FDI 유입 정도 비교 그래프(BAR)
2. 10년간 베트남 지방정부의 FDI 총 유입액(Bar)
3. 각 지방별 FDI 와 요소들간의 상관관계파악(scatterplot) 및 피어슨계수로 유의미한 요소 파악
4. Line-regression 으로 검정 및 시각화
```

💡 1) 10년간 ASEAN 국가들의 FDI 유입 정도 그래프(BAR)

데이터 시각화

```
import os
import mysql.connector
import  pandas as  pd
import numpy as np


config = {
    "user": "root",
    "password": "1234",
    "host": "192.168.56.1", #local
    "database": "PJ_01", #Database name
    "port": "3306" #port는 최초 설치 시 입력한 값(기본값은 3306)
}

conn = mysql.connector.connect(**config)

# db select, insert, update, delete 작업 객체
cursor = conn.cursor()



# 데이터 불러오기
# 실행할 select 문 구성(위에서 작성한 sql 붙여넣기)
sql = """
        SELECT YEAR,COUNTRY,FDI FROM TOP5_ASEAN;
           """

# cursor 객체를 이용해서 수행한다.
cursor.execute(sql)
resultList = cursor.fetchall()  # tuple 이 들어있는 list


import seaborn as sns #seaborn import

df = pd.DataFrame(resultList)
```

```
df.columns = ['YEAR', 'COUNTRY','FDI']  #dataframe화한 데이터에 컬럼명 입력하기 [컬럼명1, 컬럼명2]

#print(df)  #결과 확인
sns.barplot(x='YEAR', y='FDI', hue='COUNTRY',data=df, palette="muted") #연도를 x축으로 hue옵션을 사용해 국가별로 묶어서 비교그래프 생성
plt.show()
```



👉 Number of FDI project by Province(2010~2019)

matplot 사용

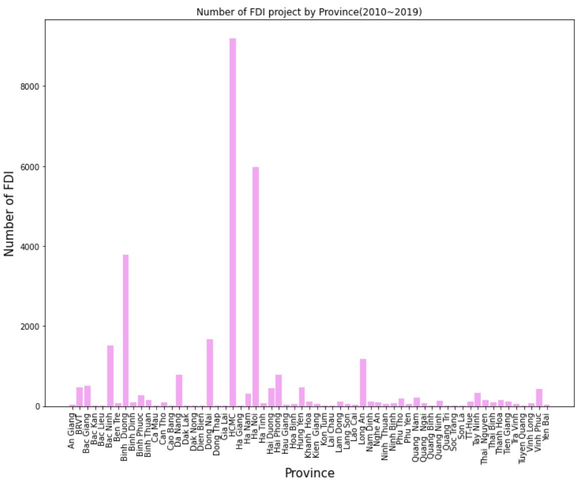
```
import mysql.connector
import  pandas as  pd
import numpy as np


config = {
    "user": "root",
    "password": "1234",
    "host": "192.168.56.1", #local
    "database": "PJ_01", #Database name
    "port": "3306" #port는 최초 설치 시 입력한 값(기본값은 3306)
}

conn = mysql.connector.connect(**config)

# db select, insert, update, delete 작업 객체
cursor = conn.cursor()



# 데이터 불러오기
# 실행할 select 문 구성(위에서 작성한 sql 붙여넣기)
sql = """ SELECT PROVINCE, Num_of_FDI FROM Accum_FDI;
          """
```

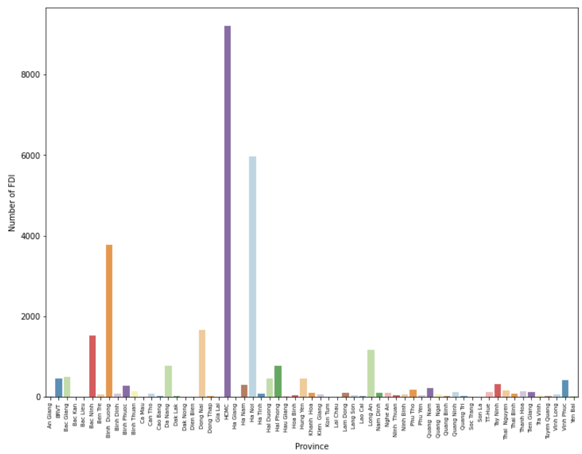
seaborn사용

```
import mysql.connector
import  pandas as  pd
import numpy as np


config = {
    "user": "root",
    "password": "1234",
    "host": "192.168.56.1", #local
    "database": "PJ_01", #Database name
    "port": "3306" #port는 최초 설치 시 입력한 값(기본값은 3306)
}

conn = mysql.connector.connect(**config)

# db select, insert, update, delete 작업 객체
cursor = conn.cursor()



# 데이터 불러오기
# 실행할 select 문 구성(위에서 작성한 sql 붙여넣기)
sql = """ SELECT PROVINCE, Num_of_FDI FROM Accum_FDI;
          """
```

```python
# cursor 객체를 이용해서 수행한다.
cursor.execute(sql)
resultList = cursor.fetchall()  # tuple 이 들어있는 list

df = pd.DataFrame(resultList)
df.columns = ['PROVINCE', 'Num_of_FDI']  #dataframe화한 데이터에 컬럼명
df.sort_values(by = 'PROVINCE', inplace =True)
#print(df)

#visualization
from matplotlib import pyplot as plt
plt.rc('font', family='NanumBarunGothic')
plt.rcParams["figure.figsize"] = (12, 9)

x = df['PROVINCE']
y = df['Num_of_FDI']


plt.title('Number of FDI project by Province(2010~2019)')
plt.ylabel('Number of FDI')
plt.xlabel('Province',fontsize=3)
plt.xticks(rotation=90)
plt.bar(x,y, align='center', alpha=0.7, color='purple')
```

```python
# cursor 객체를 이용해서 수행한다.
cursor.execute(sql)
resultList = cursor.fetchall()  # tuple 이 들어있는 list

df = pd.DataFrame(resultList)
df.columns = ['PROVINCE', 'Num_of_FDI']  #dataframe화한 데이터에
df.sort_values(by = 'PROVINCE', inplace =True)
#print(df)

#visualization
import seaborn as sns
from matplotlib import pyplot as plt
x = df['PROVINCE']
y = df['Num_of_FDI']


sns.barplot(x, y, alpha=0.8, palette='Paired')

plt.ylabel('Number of FDI')
plt.xlabel('Province',fontsize=10)
plt.xticks(rotation=90,fontsize=7)
plt.show()
```





⭐ Liner-regression 분석

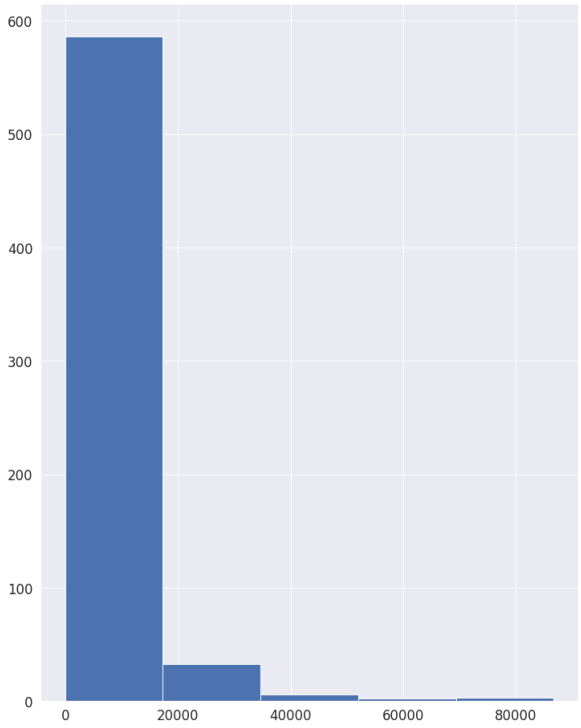
STEP 1. Data Import

```python
import pandas as pd
import numpy as np
import mysql.connector
import matplotlib.pyplot as plt
import seaborn as sns

import sys
sys.setrecursionlimit(2000) # 10000 is an example, try with different values

config = {
    "user": "root",
    "password": "1234",
    "host": "192.168.56.1", #local
    "database": "PJ_01", #Database name
    "port": "3306" #port는 최초 설치 시 입력한 값(기본값은 3306)
}

conn = mysql.connector.connect(**config)

# db select, insert, update, delete 작업 객체
cursor = conn.cursor()


# 실행할 select 문 구성(위에서 작성한 sql 붙여넣기)
```

```
sql = """ select PROVINCE,PCI,MKS, POP, LAQU,LAQT,ENTRY_COST, LAND_ACCESS, TRANSPARANCY
            , TIME_COST, INFORMAL_CHARGES, PROACTIVITY, BUSINESS_SUPPORT, LABOR_POLICY, LAW,  FDI from PCI
            GROUP BY PROVINCE, YEAR;
          """

# cursor 객체를 이용해서 수행한다.
cursor.execute(sql)

# select 된 결과 셋 얻어오기
resultList = cursor.fetchall()  # tuple 이 들어있는 list
df = pd.DataFrame(resultList, columns=['PROVINCE','PCI','MKS','POP','LAQU','LAQT','ENTRY_COST', 'LAND_ACCESS', 'TRANSPARANCY',
                                       'TIME_COST', 'INFORMAL_CHARGES','PROACTIVITY','BUSINESS_SUPPORT',
                                       'LABOR_POLICY','LAW','FDI'] )
```

STEP 2. Data exploration

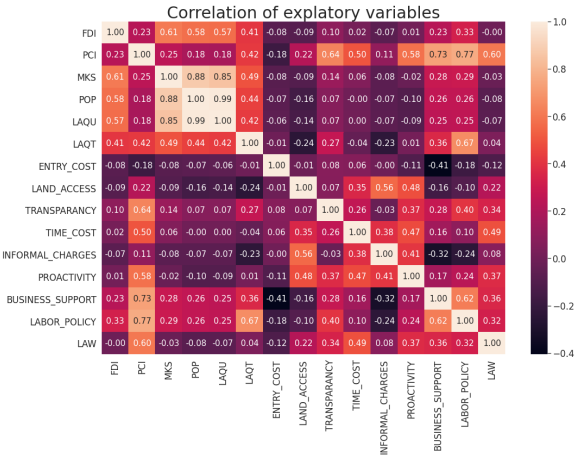
```
#데이터 기본정보 확인
print(df.shape) #(630, 11) #630개의 rows, 11개의 columns

#데이터의 NULL값 여부 (총 개수) 확인
print(df.isnull().sum() ) #모든 컬럼의 null값 0개로 확인

#각 컬럼별 데이터 타입확인
print(df.info() )

#target(예측할 데이터=종속변수) 데이터 확인
print(df['FDI'].describe() )

#target데이터의 분포 탐색
print( df['FDI'].hist(bins=5) )
```



전체 FDI volume의 분포는 대체로 USD 20000$ 이하에 분포 되어 있다. 이는 intro에서 살펴본 바와 같이 대부분의 투자가 대도시 위주로 집중되는 것이 주 원인으로 보여진다.

```
#설명변수(독립변수) 분포 탐색
numerical_columns=['PCI','MKS','POP','LAQU','LAQT','ENTRY_COST', 'LAND_ACCESS', 'TRANSPARANCY',
                                       'TIME_COST', 'INFORMAL_CHARGES','PROACTIVITY','BUSINESS_SUPPORT',
```

```
                                                'LABOR_POLICY','LAW']

fig = plt.figure(figsize = (20, 16))
ax = fig.gca()

df[numerical_columns].hist(ax=ax)
plt.show()

#변수들의 상관관계 탐색
#1 or -1 에가까울수록 강한 상관관계를 가진다
cols = ['FDI','PCI','MKS','POP','LAQU','LAQT','ENTRY_COST', 'LAND_ACCESS', 'TRANSPARANCY',
                                                'TIME_COST', 'INFORMAL_CHARGES','PROACTIVITY','BUSINESS_SUPPORT',
                                                'LABOR_POLICY','LAW']
corr = df[cols].corr(method = 'pearson')
fig = plt.figure(figsize = (16, 12))
ax = fig.gca()

sns.set(font_scale=1.5)
hm = sns.heatmap(corr.values,
                 annot=True,
                 fmt='.2f', #디자인 파라미터 #소수점 2번째 자리까지 출력
                 annot_kws={'size': 15}, #heatmap 각 칸의 사이즈
                 yticklabels=cols, #X축의 label 설정
                 xticklabels=cols, #Y축의 label 설정
                 ax=ax)
plt.tight_layout() #그래프가 좀더 보기좋게 출력
plt.show()
```



두 연속형 변수간의 상관관계는 `-1` 혹은 `1`에 가까울수록 강한 상관관계를 의미한다.

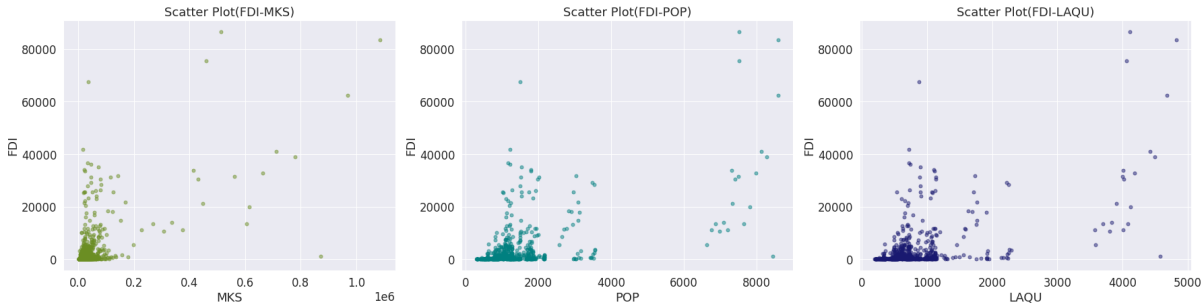
`Heatmap`을 통해 알수있는 것은 `Target` 변수인 `FDI`와 가장 상관관계가 높은 `feature` 은 `MKS`, `POP`, `LAQU` 이라는 것이다.

그외에 `PCI`, `BUSINESS_SUPPORT`, `LABOR_POLICY` 등이 상대적으로 연관성이 있을 것으로 보여진다.

전반적으로 `-1` 혹은 `1`에 많이 근접한 요소들은 없어 모든 요소들이 `FDI`유입에 강한 영향력을 끼치기는 어려울 것으로 예상된다.

```
#heatmap에서 밝혀낸 상관관계를 기반으로 주요 요소들과 종속변수의 관계 탐색

#설명변수와 종속변수의 관계탐색 (MKS,POP,LAQU)
plt.figure(figsize=(12,9))

# 1) MKS
plt.subplot(231)
plt.plot('MKS', 'FDI',
         data=df,
         linestyle='none',
         marker='o', #plot 형태 동그라미
         markersize=5,
         color='olivedrab',
         alpha=0.5)
plt.title('Scatter Plot(FDI-MKS)')
plt.xlabel('MKS')
plt.ylabel('FDI')


# 2) POP
plt.subplot(232)
plt.plot('POP', 'FDI',
         data=df,
         linestyle='none',
         marker='o', #plot 형태 동그라미
         markersize=5,
         color='teal',
         alpha=0.5)
plt.title('Scatter Plot(FDI-POP)')
```

```
plt.xlabel('POP')
plt.ylabel('FDI')


# 3) LAQU
plt.subplot(233)
plt.plot('LAQU', 'FDI',
         data=df,
         linestyle='none',
         marker='o', #plot 형태 동그라미
         markersize=5,
         color='midnightblue',
         alpha=0.5)
plt.title('Scatter Plot(FDI-LAQU)')
plt.xlabel('LAQU')
plt.ylabel('FDI')

plt.show()
```



heatmap의 결과에서도 드러났던 것과 마찬가지로 전체 요소중 가장 높은 상관관계를 보인 3개의 feature 모두 어느정도 양의 상관관계를 보이고는 있으나,
그 상관도가 높다고 보기는 어렵다.

STEP 3. Data analysys위한 데이터 전처리

최초로 적용한 모델의 성능평가를 위해 위에서 파악한 높은 상관관계를 가진 요소들에 대한 적용은 추후에 진행할 것이다.

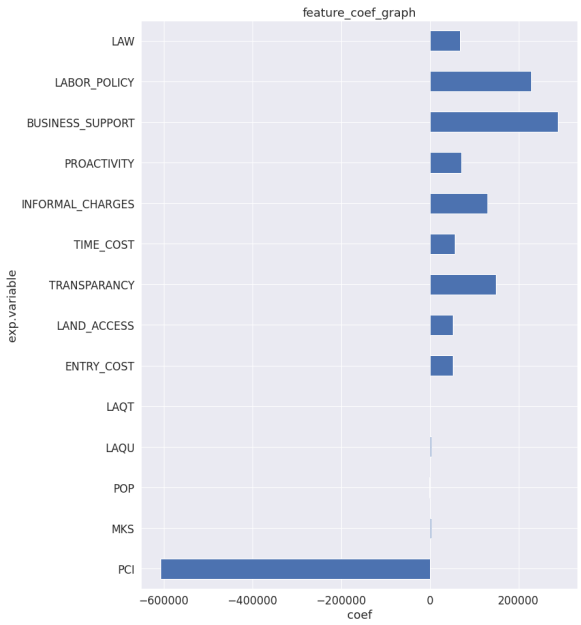
```
# 데이터 전처리1
#설명변수들 표준화하기(분포 표준화)
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler() #scaler 라는 ojbect 불러오기

#적용할 칼럼 명시하기
scale_columns=['PCI','MKS','POP','LAQU','LAQT','ENTRY_COST', 'LAND_ACCESS', 'TRANSPARANCY',
                                            'TIME_COST', 'INFORMAL_CHARGES','PROACTIVITY','BUSINESS_SUPPORT',
                                            'LABOR_POLICY','LAW']
df[scale_columns] = scaler.fit_transform(df[scale_columns]) #스케일링할 feature입력


# 데이터 전처리2
#데이터셋 분리 (train / test)
from sklearn import linear_model
from sklearn.model_selection import train_test_split
from math import sqrt

# dataset split to train/test
X = df[scale_columns]
y = df['FDI']

#train 20%, test 80%로 분리
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=(False),random_state=33)

#잘 나눠졌는지 확인
print( X_train.shape, X_test.shape ) #(504, 14)=80% (126, 14)=20% 로 잘 나누어져 있음.
```

STEP 4. Liner regression analysis

```
#모델 성과 평가를 자동으로 출력할 함수 생성
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import mean_absolute_error, mean_squared_error

my_predictions = {}

#결과물 출력할 색상의 종류 리스트 (출력할때마다 렌덤하게 색상 선정)
colors = ['r', 'c', 'm', 'y', 'k', 'khaki', 'teal', 'orchid', 'sandybrown',
          'greenyellow', 'dodgerblue', 'deepskyblue', 'rosybrown', 'firebrick',
          'deeppink', 'crimson', 'salmon', 'darkred', 'olivedrab', 'olive',
          'forestgreen', 'royalblue', 'indigo', 'navy', 'mediumpurple', 'chocolate',
          'gold', 'darkorange', 'seagreen', 'turquoise', 'steelblue', 'slategray',
          'peru', 'midnightblue', 'slateblue', 'dimgray', 'cadetblue', 'tomato'
         ]


def plot_predictions(name_, pred, actual): #분석모델이름, 예측값, 실제값
    df = pd.DataFrame({'prediction': pred, 'actual': y_test})
    df = df.sort_values(by='actual').reset_index(drop=True)

    plt.figure(figsize=(12, 9))
    plt.scatter(df.index, df['prediction'], marker='x', color='r')
    plt.scatter(df.index, df['actual'], alpha=0.7, marker='o', color='black')
    plt.title(name_, fontsize=15)
    plt.legend(['prediction', 'actual'], fontsize=12)
    plt.show()


#예측결과 그래프 & MSE 지표 출력할 함수 생성
def mse_eval(name_, pred, actual): #분석모델이름, 예측값, 실제값
    global predictions
    global colors

    plot_predictions(name_, pred, actual)

    mse = mean_squared_error(pred, actual)
    my_predictions[name_] = mse

    y_value = sorted(my_predictions.items(), key=lambda x: x[1], reverse=True)

    df = pd.DataFrame(y_value, columns=['model', 'mse'])
    print(df)
    min_ = df['mse'].min() - 10
    max_ = df['mse'].max() + 10

    length = len(df)

    plt.figure(figsize=(10, length))
    ax = plt.subplot()
    ax.set_yticks(np.arange(len(df)))
    ax.set_yticklabels(df['model'], fontsize=15)
    bars = ax.barh(np.arange(len(df)), df['mse'])

    for i, v in enumerate(df['mse']):
        idx = np.random.choice(len(colors))
        bars[i].set_color(colors[idx])
        ax.text(v + 2, i, str(round(v, 3)), color='k', fontsize=15, fontweight='bold')

    plt.title('MSE Error', fontsize=18)
    plt.xlim(min_, max_)

    plt.show()

#예측결과 그래프 & MAE 지표 출력할 함수 생성
def mae_eval(name_, pred, actual): #분석모델이름, 예측값, 실제값
    global predictions
    global colors

    plot_predictions(name_, pred, actual)

    mae = mean_absolute_error(pred, actual)
    my_predictions[name_] = mae

    y_value = sorted(my_predictions.items(), key=lambda x: x[1], reverse=True)

    df = pd.DataFrame(y_value, columns=['model', 'mae'])
    print(df)
    min_ = df['mae'].min() - 10
    max_ = df['mae'].max() + 10

    length = len(df)
```

```
    plt.figure(figsize=(10, length))
    ax = plt.subplot()
    ax.set_yticks(np.arange(len(df)))
    ax.set_yticklabels(df['model'], fontsize=15)
    bars = ax.barh(np.arange(len(df)), df['mae'])

    for i, v in enumerate(df['mae']):
        idx = np.random.choice(len(colors))
        bars[i].set_color(colors[idx])
        ax.text(v + 2, i, str(round(v, 3)), color='k', fontsize=15, fontweight='bold')

    plt.title('MAE Error', fontsize=18)
    plt.xlim(min_, max_)

    plt.show()


#예측결과 그래프 & RMSE 지표 출력할 함수 생성
def rmse_eval(name_, pred, actual): #분석모델이름, 예측값, 실제값
    global predictions
    global colors

    plot_predictions(name_, pred, actual)

    rmse = np.sqrt(mean_squared_error(pred, actual) )
    my_predictions[name_] = rmse

    y_value = sorted(my_predictions.items(), key=lambda x: x[1], reverse=True)

    df = pd.DataFrame(y_value, columns=['model', 'rmse'])
    print(df)
    min_ = df['rmse'].min() - 10
    max_ = df['rmse'].max() + 10

    length = len(df)

    plt.figure(figsize=(10, length))
    ax = plt.subplot()
    ax.set_yticks(np.arange(len(df)))
    ax.set_yticklabels(df['model'], fontsize=15)
    bars = ax.barh(np.arange(len(df)), df['rmse'])

    for i, v in enumerate(df['rmse']):
        idx = np.random.choice(len(colors))
        bars[i].set_color(colors[idx])
        ax.text(v + 2, i, str(round(v, 3)), color='k', fontsize=15, fontweight='bold')

    plt.title('RMSE Error', fontsize=18)
    plt.xlim(min_, max_)

    plt.show()

#모델 삭제 함수
def remove_model(name_): #분석모델이름
    global my_predictions
    try:
        del my_predictions[name_]
    except KeyError:
        return False
    return True



from sklearn import linear_model
#liner 회귀분석 모델 불러오기
lr = linear_model.LinearRegression()
model = lr.fit(X_train, y_train)

# print coef
print(lr.coef_)

#coef 시각화 하기
# figure 크기 지정
plt.rcParams['figure.figsize'] = [12, 16]

# graph 값 전처리
coefs = lr.coef_.tolist() #coef가 파이썬 list가 아니므로 변형
coefs_series = pd.Series(coefs) #판다스로 출력하기 쉽게 series 형태로 변형

# graph 정보지정
x_labels = scale_columns
ax = coefs_series.plot.barh() #다중 barplot 그리기
ax.set_title('feature_coef_graph') #타이틀 설정
ax.set_xlabel('coef') #x축 이름
ax.set_ylabel('exp.variable') #y축 이름
```

```
ax.set_yticklabels(x_labels)
plt.show()
```



첫번째 모델에서 가장 영향력이 큰 feature들은
PCI, BUSINESS_SUPPORT, LABOR_POLICY 등이 있음을 확인할 수 있다.

STEP 5. 학습 결과 해석

```
# 1) R2 score 계산

#학습 score = 0.4401251233806852
print(model.score(X_train, y_train))

#실제 score = 0.005335207829716415
print(model.score(X_test, y_test))


pred = model.predict(X_test)
mse_eval('LinearRegression', pred, y_test)
mae_eval('LinearRegression', pred, y_test)
rmse_eval('LinearRegression', pred, y_test)

# 설명함수의 유의성 검정
import statsmodels.api as sm

X_train = sm.add_constant(X_train)
model = sm.OLS(y_train, X_train).fit() #OLS 학습을 시켜준다
print( model.summary() ) #OLS는 X_train과 결과는 같지만, summary를 할 수 있다

#다중 공선성 검정
#vif 가 10보다 크면, 해당 feature는 다른 feature들과 상관관계가 높아 다중공선성이 발생한 feature이다.

from statsmodels.stats.outliers_influence import variance_inflation_factor

vif = pd.DataFrame()
vif["VIF Factor"] = [variance_inflation_factor(X_train.values, i) for i in range(X_train.shape[1])]
vif["features"] = X_train.columns
print(vif.round(1))
```

PCI, ENTRY_COST, LAND_ACCESS, TRANSPARANCY, TIME_COST, INFORMAL_CHARGES, PROACTIVITY, BUSINESS_SUPPORT, LABOR_POLICYLAW
등의 VIF 수치가 매우 높아 다중공선성이 발생한 것을 확인할 수 있었다.
특히 PCI의 경우 다른 SUB-INDEX 들을 합한 값이기 때문에 다른 feature에 상당히 강한 영향력을 주는 것으로 생각된다.

모델을 개선하기 위해 PCI를 feature 우선적으로 제거한뒤 다시한번 검정을 진행하였다.
그 결과 다른 PIC 관련 feature들의 vif 수치는 5이하로 하락하였으며, 최종적으로 POP, LAQU 의 vif 수치가 100이 넘어 학습에 방해를 하는 요소임을 확인
하였다.

STEP 6. 모델 개선

FDI와 feature간의 상관관계를 확인했던 heatmpa의 결과, OLS검정을 통해 확인한 결과, vif 값을 통해 검정한 결과등을 종합하여 FDI에 상대적으로 강한 영향을 주는 feature들을 선정하여 다시한번 학습을 진행하였다.
추가로 과적합 문제를 방지하기 위하여
Ridge, Lasso, ElsaticNEt 을 적용해보았으나 모두 Liner regression을 한것보다 수치가 높게 나와 적용하지 않았다.
개선한 모델의 전체 코드는 아래와 같다.

```
import pandas as pd
import numpy as np
import mysql.connector
import matplotlib.pyplot as plt
import seaborn as sns

import sys
sys.setrecursionlimit(2000) # 10000 is an example, try with different values


#STEP 1. Data Import

config = {
    "user": "root",
    "password": "1234",
    "host": "192.168.56.1", #local
    "database": "PJ_01", #Database name
    "port": "3306" #port는 최초 설치 시 입력한 값(기본값은 3306)
}

conn = mysql.connector.connect(**config)

# db select, insert, update, delete 작업 객체
cursor = conn.cursor()


# 실행할 select 문 구성(위에서 작성한 sql 붙여넣기)
sql = """ select PROVINCE,PCI,MKS, POP, LAQU,LAQT,ENTRY_COST, LAND_ACCESS, TRANSPARANCY
            , TIME_COST, INFORMAL_CHARGES, PROACTIVITY, BUSINESS_SUPPORT, LABOR_POLICY, LAW,  FDI from PCI
            GROUP BY PROVINCE, YEAR;
           """

# cursor 객체를 이용해서 수행한다.
cursor.execute(sql)

# select 된 결과 셋 얻어오기
resultList = cursor.fetchall()  # tuple 이 들어있는 list
df = pd.DataFrame(resultList, columns=['PROVINCE','PCI','MKS','POP','LAQU','LAQT','ENTRY_COST', 'LAND_ACCESS', 'TRANSPARANCY',
                                       'TIME_COST', 'INFORMAL_CHARGES','PROACTIVITY','BUSINESS_SUPPORT',
                                       'LABOR_POLICY','LAW','FDI'] )

#STEP 2. Data Exploration

#데이터 기본정보 확인
print(df.shape) #(630, 11)

#데이터의 NULL값 여부 (총 개수) 확인
print(df.isnull().sum() ) #모든 컬럼의 null값 0개로 확인

#각 컬럼별 데이터 타입확인
print(df.info() )

#target(예측할 데이터=종속변수) 데이터 확인
print(df['FDI'].describe() )

#target데이터의 분포 탐색
print( df['FDI'].hist(bins=5), plt.title('Histgram of FDI') )


#설명변수(독립변수) 분포 탐색
numerical_columns=['PCI','MKS','POP','LAQU','LAQT','ENTRY_COST', 'LAND_ACCESS', 'TRANSPARANCY',
                                       'TIME_COST', 'INFORMAL_CHARGES','PROACTIVITY','BUSINESS_SUPPORT',
                                       'LABOR_POLICY','LAW']

fig = plt.figure(figsize = (20, 16))
ax = fig.gca()

df[numerical_columns].hist(ax=ax)
plt.show()

#변수들의 상관관계 탐색
```

```
#1 or -1 에가까울수록 강한 상관관계를 가진다
cols = ['FDI','PCI','MKS','POP','LAQU','LAQT','ENTRY_COST', 'LAND_ACCESS', 'TRANSPARANCY',
                                                'TIME_COST', 'INFORMAL_CHARGES','PROACTIVITY','BUSINESS_SUPPORT',
                                                'LABOR_POLICY','LAW']
corr = df[cols].corr(method = 'pearson')
fig = plt.figure(figsize = (16, 12))
plt.title("Correlation of explatory variables",fontsize=30)
ax = fig.gca()

sns.set(font_scale=1.5)
hm = sns.heatmap(corr.values,
                 annot=True,
                 fmt='.2f', #디자인 파라미터 #소수점 2번째 자리까지 출력
                 annot_kws={'size': 15}, #heatmap 각 칸의 사이즈
                 yticklabels=cols, #X축의 label 설정
                 xticklabels=cols, #Y축의 label 설정
                 ax=ax)
plt.tight_layout() #그래프가 좀더 보기좋게 출력
plt.show()


#heatmap에서 밝혀낸 상관관계를 기반으로 주요 요소들과 종속변수의 관계 탐색

#설명변수와 종속변수의 관계탐색 (MKS,POP,LAQU)
plt.figure(figsize=(30,15))

# 1) MKS
plt.subplot(231)
plt.plot('MKS', 'FDI',
         data=df,
         linestyle='none',
         marker='o', #plot 형태 동그라미
         markersize=5,
         color='olivedrab',
         alpha=0.5)
plt.title('Scatter Plot(FDI-MKS)')
plt.xlabel('MKS')
plt.ylabel('FDI')


# 2) POP
plt.subplot(232)
plt.plot('POP', 'FDI',
         data=df,
         linestyle='none',
         marker='o', #plot 형태 동그라미
         markersize=5,
         color='teal',
         alpha=0.5)
plt.title('Scatter Plot(FDI-POP)')
plt.xlabel('POP')
plt.ylabel('FDI')


# 3) LAQU
plt.subplot(233)
plt.plot('LAQU', 'FDI',
         data=df,
         linestyle='none',
         marker='o', #plot 형태 동그라미
         markersize=5,
         color='midnightblue',
         alpha=0.5)
plt.title('Scatter Plot(FDI-LAQU)')
plt.xlabel('LAQU')
plt.ylabel('FDI')

plt.show()


#STEP 3.  Data analysys위한 데이터 전처리

# 데이터 전처리1
#설명변수들 표준화하기(분포 표준화)
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler() #scaler 라는 ojbect 불러오기

#적용할 설명변수 칼럼 명시하기
scale_columns=['MKS','POP','LAQT','LABOR_POLICY','BUSINESS_SUPPORT']
df[scale_columns] = scaler.fit_transform(df[scale_columns]) #스케일링할 feature입력




# 데이터 전처리2
#데이터셋 분리 (train / test)
from sklearn import linear_model
```

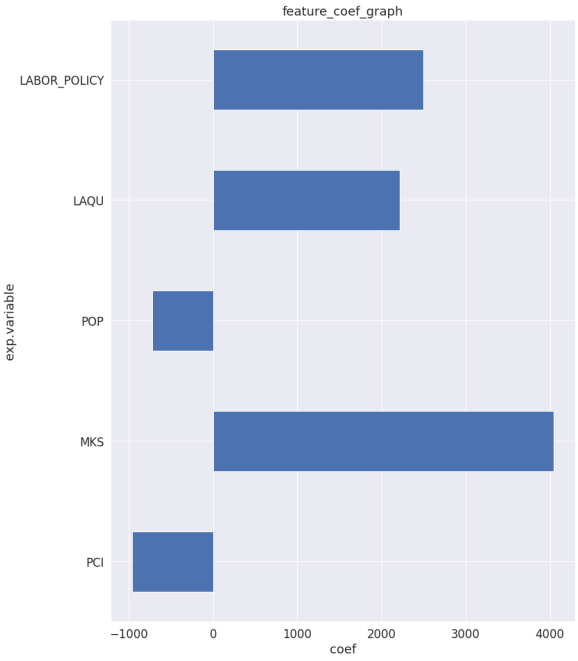
```
from sklearn.model_selection import train_test_split
from math import sqrt

# dataset split to train/test
X = df[scale_columns]
y = df['FDI']

#train 20%, test 80%로 분리
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=(False),random_state=33)

#잘 나눠졌는지 확인
print( X_train.shape, X_test.shape ) #(504, 14)=80% (126, 14)=20%



# STEP 4. 회귀분석 모델 학습
from sklearn.preprocessing import StandardScaler, MinMaxScaler, RobustScaler
std_scaler = StandardScaler()
std_scaled = std_scaler.fit_transform(X_train)
round(pd.DataFrame(std_scaled).describe(), 2)



#모델 성과 평가를 자동으로 출력할 함수 생성
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import mean_absolute_error, mean_squared_error

my_predictions = {}

#결과물 출력할 색상의 종류 리스트 (출력할때마다 렌덤하게 색상 선정)
colors = ['r', 'c', 'm', 'y', 'k', 'khaki', 'teal', 'orchid', 'sandybrown',
          'greenyellow', 'dodgerblue', 'deepskyblue', 'rosybrown', 'firebrick',
          'deeppink', 'crimson', 'salmon', 'darkred', 'olivedrab', 'olive',
          'forestgreen', 'royalblue', 'indigo', 'navy', 'mediumpurple', 'chocolate',
          'gold', 'darkorange', 'seagreen', 'turquoise', 'steelblue', 'slategray',
          'peru', 'midnightblue', 'slateblue', 'dimgray', 'cadetblue', 'tomato'
         ]

def plot_predictions(name_, pred, actual): #분석모델이름, 예측값, 실제값
    df = pd.DataFrame({'prediction': pred, 'actual': y_test})
    df = df.sort_values(by='actual').reset_index(drop=True)

    plt.figure(figsize=(12, 9))
    plt.scatter(df.index, df['prediction'], marker='x', color='r')
    plt.scatter(df.index, df['actual'], alpha=0.7, marker='o', color='black')
    plt.title(name_, fontsize=15)
    plt.legend(['prediction', 'actual'], fontsize=12)
    plt.show()


#예측결과 그래프 & MSE 지표 출력할 함수 생성
def mse_eval(name_, pred, actual): #분석모델이름, 예측값, 실제값
    global predictions
    global colors

    plot_predictions(name_, pred, actual)

    mse = mean_squared_error(pred, actual)
    my_predictions[name_] = mse

    y_value = sorted(my_predictions.items(), key=lambda x: x[1], reverse=True)

    df = pd.DataFrame(y_value, columns=['model', 'mse'])
    print(df)
    min_ = df['mse'].min() - 10
    max_ = df['mse'].max() + 10

    length = len(df)

    plt.figure(figsize=(10, length))
    ax = plt.subplot()
    ax.set_yticks(np.arange(len(df)))
    ax.set_yticklabels(df['model'], fontsize=15)
    bars = ax.barh(np.arange(len(df)), df['mse'])

    for i, v in enumerate(df['mse']):
        idx = np.random.choice(len(colors))
        bars[i].set_color(colors[idx])
        ax.text(v + 2, i, str(round(v, 3)), color='k', fontsize=15, fontweight='bold')

    plt.title('MSE Error', fontsize=18)
    plt.xlim(min_, max_)

    plt.show()

#예측결과 그래프 & MAE 지표 출력할 함수 생성
```

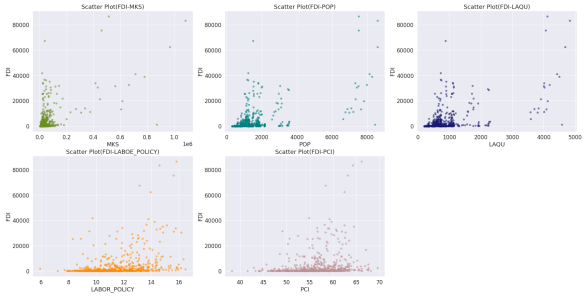
```
def mae_eval(name_, pred, actual): #분석모델이름, 예측값, 실제값
    global predictions
    global colors

    plot_predictions(name_, pred, actual)

    mae = mean_absolute_error(pred, actual)
    my_predictions[name_] = mae

    y_value = sorted(my_predictions.items(), key=lambda x: x[1], reverse=True)

    df = pd.DataFrame(y_value, columns=['model', 'mae'])
    print(df)
    min_ = df['mae'].min() - 10
    max_ = df['mae'].max() + 10

    length = len(df)

    plt.figure(figsize=(10, length))
    ax = plt.subplot()
    ax.set_yticks(np.arange(len(df)))
    ax.set_yticklabels(df['model'], fontsize=15)
    bars = ax.barh(np.arange(len(df)), df['mae'])

    for i, v in enumerate(df['mae']):
        idx = np.random.choice(len(colors))
        bars[i].set_color(colors[idx])
        ax.text(v + 2, i, str(round(v, 3)), color='k', fontsize=15, fontweight='bold')

    plt.title('MAE Error', fontsize=18)
    plt.xlim(min_, max_)

    plt.show()


#예측결과 그래프 & RMSE 지표 출력할 함수 생성
def rmse_eval(name_, pred, actual): #분석모델이름, 예측값, 실제값
    global predictions
    global colors

    plot_predictions(name_, pred, actual)

    rmse = np.sqrt(mean_squared_error(pred, actual) )
    my_predictions[name_] = rmse

    y_value = sorted(my_predictions.items(), key=lambda x: x[1], reverse=True)

    df = pd.DataFrame(y_value, columns=['model', 'rmse'])
    print(df)
    min_ = df['rmse'].min() - 10
    max_ = df['rmse'].max() + 10

    length = len(df)

    plt.figure(figsize=(10, length))
    ax = plt.subplot()
    ax.set_yticks(np.arange(len(df)))
    ax.set_yticklabels(df['model'], fontsize=15)
    bars = ax.barh(np.arange(len(df)), df['rmse'])

    for i, v in enumerate(df['rmse']):
        idx = np.random.choice(len(colors))
        bars[i].set_color(colors[idx])
        ax.text(v + 2, i, str(round(v, 3)), color='k', fontsize=15, fontweight='bold')

    plt.title('RMSE Error', fontsize=18)
    plt.xlim(min_, max_)

    plt.show()

#모델 삭제 함수
def remove_model(name_): #분석모델이름
    global my_predictions
    try:
        del my_predictions[name_]
    except KeyError:
        return False
    return True



from sklearn import linear_model
#liner 회귀분석 모델 불러오기
lr = linear_model.LinearRegression()
model = lr.fit(X_train, y_train)

# print coef
```

```
#print(lr.coef_)

#coef 시각화 하기
# figure 크기 지정
plt.rcParams['figure.figsize'] = [12, 16]

# graph 값 전처리
coefs = lr.coef_.tolist() #coef가 파이썬 list가 아니므로 변형
coefs_series = pd.Series(coefs) #판다스로 출력하기 쉽게 series 형태로 변형

# graph 정보지정
x_labels = scale_columns
ax = coefs_series.plot.barh() #다중 barplot 그리기
ax.set_title('feature_coef_graph') #타이틀 설정
ax.set_xlabel('coef') #x축 이름
ax.set_ylabel('exp.variable') #y축 이름
ax.set_yticklabels(x_labels)
plt.show()


#STEP 5. 학습 결과 해석

# 1) R2 score 계산

#학습 score = 0.4691181379020787
print(model.score(X_train, y_train))

#실제 score = -0.36674134727426155
print(model.score(X_test, y_test))


pred = model.predict(X_test)
mse_eval('LinearRegression', pred, y_test)
mae_eval('LinearRegression', pred, y_test)
rmse_eval('LinearRegression', pred, y_test)

from sklearn.linear_model import Ridge
# 값이 커질 수록 큰 규제
alphas = [100]
for alpha in alphas:
    ridge = Ridge(alpha=alpha)
    ridge.fit(X_train, y_train)
    pred = ridge.predict(X_test)
    mse_eval('Ridge(alpha={})'.format(alpha), pred, y_test)


# 설명함수의 유의성 검정
import statsmodels.api as sm

X_train = sm.add_constant(X_train)
model = sm.OLS(y_train, X_train).fit() #OLS 학습을 시켜준다
print( model.summary() ) #OLS는 X_train과 결과는 같지만, summary를 할 수 있다

#다중 공선성 검정
#vif 가 10보다 크면, 해당 feature는 다른 feature들과 상관관계가 높아 다중공선성이 발생한 feature이다.

from statsmodels.stats.outliers_influence import variance_inflation_factor

vif = pd.DataFrame()
vif["VIF Factor"] = [variance_inflation_factor(X_train.values, i) for i in range(X_train.shape[1])]
vif["features"] = X_train.columns
print(vif.round(1))
```

```
R2 Score
0.43238255966647454
0.045331175946832336


              model           mse
0  LinearRegression  3.469435e+07
              model          mae
0  LinearRegression  3402.613571
              model         rmse
0  LinearRegression  5890.190903
              model           mse
0  Ridge(alpha=100)  3.421923e+07

1  LinearRegression  5.890191e+03


                            OLS Regression Results
==============================================================================
Dep. Variable:                    FDI   R-squared:                       0.432
Model:                            OLS   Adj. R-squared:                  0.427
```

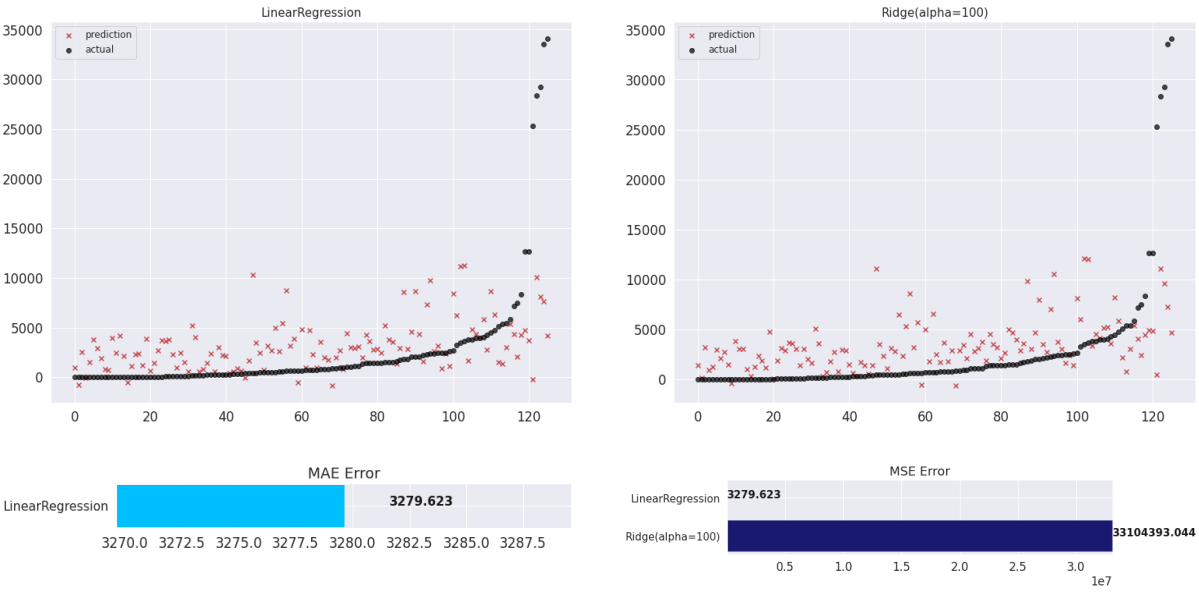
```
Method:                 Least Squares   F-statistic:                     75.87
Date:                Sat, 30 Jan 2021   Prob (F-statistic):           4.91e-59
Time:                        22:26:20   Log-Likelihood:                -5226.2
No. Observations:                 504   AIC:                         1.046e+04
Df Residuals:                     498   BIC:                         1.049e+04
Df Model:                           5
Covariance Type:            nonrobust
==========================================================================================
                             coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------
const                   4194.0426    346.767     12.095      0.000    3512.735    4875.350
MKS                     3720.6297    718.837      5.176      0.000    2308.303    5132.957
POP                     1714.2646    719.119      2.384      0.018     301.383    3127.146
LAQT                     457.3323    506.277      0.903      0.367    -537.371    1452.035
LABOR_POLICY            1965.9396    573.240      3.430      0.001     839.673    3092.206
BUSINESS_SUPPORT        -799.0030    457.991     -1.745      0.082   -1698.836     100.830
==============================================================================
Omnibus:                      365.718   Durbin-Watson:                   1.565
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             8503.112
Skew:                           2.853   Prob(JB):                         0.00
Kurtosis:                      22.296   Cond. No.                         5.14
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
   VIF Factor          features
0         1.0             const
1         5.3               MKS
2         5.1               POP
3         2.4              LAQT
4         2.8      LABOR_POLICY
5         1.8  BUSINESS_SUPPORT
```





feature 중에서
FDI의 증가에 가장 큰 영향을 끼치는 것은 MKS 인것으로 판단된다.
scatterplot을 보면 상대적으로 상관도가 가장 높은 변수들임에도 불구하고
FDI와는 약한 양의 상관관계를 가지고 있는 것으로 보여진다.

💡 모델 예측 결과



STEP 7 . 결과

OLS 분석을 통해 나온 결과를 보면 3개의 feature(MKS, POP, LABOR_POLICY) 가 FDI inflow에 상당한 영향을 끼친다는 것을 확인할 수 있었다.
즉, 해외 기업이 투자를 하고자 하는 지역의 시장크기가 얼마나 큰지, 거주 인구 (구매력, 노동력) 가 얼마나 많은지, 지방정부가 해당 지역의 고급노동력의 비중을 늘리기 위해 얼마나 노력하고 있는지 등에 대해 중요하게 생각하고 있다는 것을 확인할 수 있었다.

베트남의 산업구조상 대부분의 산업은 제조업에 집중되어 있고, 제조업을 주력으로 삼는 다수의 글로벌 기업들이 베트남을 주요 생산거점으로서 활용하고 있다는 점을 상기해 본다면, 해당 분석 결과가 어느정도 현실을 반영하고 있음을 알 수 있다.

아쉽게도 해당 모델의 R2 Score를 보면 테스트 단계에서는 0.432 로 약 40점대의 예측율을 보이는 반면, 실제 테스트에서는 1/10불과한 0.045라는 점수밖에 받지 못해 모델자체의 신뢰도는 상당히 낮은 것으로 판명된다.

실제로 OLS 검정을 통해 도출된 결과를 보아도 R-squre가 0.432에 불과하여 설명력이 부족함을 나타낸다.

해당 모델은 F-statistic 가 상당히 높고, 3개의 유의미한 feature 값이 발견되었다. 그러므로 해당모델에서 찾아낸 feature(x값) 들이 Y값(FDI)에 미치는 영향은 여전히 유의하지만, 추정된 회귀식을 신뢰하기는 힘들다는 결론에 다다랐다.

STEP 8 . 보완점

이 모델은 3가지 부분에서 보완점을 발견하였다.

첫째, 데이터의 절대적 양의 부족.
모델에 사용된 데이터는 베트남 63개 지방의 지난 10년간 누적된 데이터이다. 실제 베트남이 정부차원에서 적극적으로 FDI를 유치한것은 1986년부터 이며 PCI 데이터가 측정된 시점도 1990년대 초반부터이다. 그러므로 누적된 데이터를 추가적으로 더 모집할 수 있다면, 해당 모델의 설명력을 어느정도 증가시킬 수 있을 것이라 예상한다.

둘째, PCI 데이터에 대한 신뢰도 문제
PCI 데이터는 베트남 현지에서 운영중인 '기업인'에게 설문을 통하여 얻은 결과를 수치화하여 정리한 데이터이다. 이러한 데이터는 객관성이 상대적으로 결여되고, 설문 내용을 만드는 방법, 응답자의 성향 등 영향을 받을 수 있는 요인들이 여러가지 이기 때문에 데이터 자체의 신뢰도가 아주 높다고 하기는 어렵다.
그렇기 때문에, 지방정부의 행정적 부분에서의 feature를 수치화할 수 있는 별개의 feature가 더 추가된다면 feature의 객관성이 더욱 높아질 수 있을 것이다.

셋째, 경제적 요소(시장크기, 인구, 노동가능 인구, 노동품질) 등의 요소가 완벽하게 상호독립적인 요소가 아니었다. 이는 첫번째 모델을 검정하였을때의 VIF 수치를 통해 다중공선성에 문제가 있었음이 검증되었다.
좀 더 정확한 영향력을 측정하기 위해서는 FDI 유입에 영향을 끼칠 수 있는 다른 경제적인 요소들(전기 및 수도 사용량, 교통량, 토지개발지수 등) 을 추가한다면 더욱 좋은 결과를 얻을 수 있을 것이다.