**코로나19 사태 전후로**

**주가와 변동 요인간의 상관관계분석**

1조 김경률 최현석 성영진

**목차**

**I. 서론**

1. 연구 배경

2. 데이터 및 변수

2-1. 데이터 출처

2-2. 독립변수와 종속변수 설명

**II.본론**

1. KOSDAQ 종목 5개, KOSPI 종목 5개 추출
2. 다중회귀 분석

2-1. 코로나 발생 이후 KOSPI 다중회귀분석

2-2. 코로나 발생 이후 KOSDAQ 다중회귀분석

2-3. 코로나 발생 이전 10년 동안의 KOSPI 다중회귀분석

**III. 결론**

**I. 서론.**

1. 연구배경

중국에서 발생한 코로나19가 전 세계로 확산되고 국제유가의 급락이 동반되면서 코로나19 사태는 새로운 국면으로 접어들었다. 코로나19가 글로벌 증시의 공통위험요인으로 작용함에 따라 주요국 증시는 매우 동조화된 하락 양상을 보였고 있으며 국내 상장주식의 주가하락도 대외요인에 대한 민감도에 따라 결정되는 모습이다. 통제하기 어려운 글로벌 위험요인에 직면한 한국 주식시장이 당분간 높은 변동성에 노출되는 것은 불가피한 것으로 보인다.



한국 주식시장은 코로나19 발생 이후 약 20% 하락하였고 주요국에 비해 하락폭이 상대적으로 작다. 팬데믹 6개월의 세계증시 움직임은 '제조업이 탄탄한 나라가 경제위기에 강하다'는 통설을 재확인시켜 주고 있다. 이는 앞으로 경기 반등국면이 전개될 경우의 증시반응도 제조강국에서 가장 크고 빠르게 나타날 것임을 시사한다. 제조업과 기술기업을 보유한 한국에서는 코로나 이후 ' V자형' 경기 회복을 보여주고 있다.

한국에서 ‘V자’ 반등 한 주식과 뉴스(빈도), 금리, 환율, 유가, 다우 , 나스닥 요인에 대한 국내 상장주식의 민감도를 업종별로 측정하겠다. 코로나19 발생 이후의 주가변화와의 관계를 살펴본 후 이들 요인의 국내증시에 대한 영향력을 확인하겠다. 추가적으로 코로나19 발생 이전 10년동안 한국에서 ‘V자’ 반등 한 주식과 뉴스(빈도), 금리, 환율, 유가, 다우 , 나스닥 요인이 어떠한 상관이 있는지 알아보겠다.

1. 데이터 및 변수

2-1. 데이터 출처

* KOSPI & KOSDAQ 종목 데이터

변수 : Date(날짜), Adj Close(조정 종가)

출처 : KRX, 야후 파이낸스 API

(https://marketdata.krx.co.kr/mdi#document=13020101)

* 환율 데이터

변수 : Date(날짜), dollar(기준환율)

출처 : 우리은행 외환센터

(https://spot.wooribank.com/pot/Dream?withyou=FXXRT0014)

* 금리 데이터

변수 : Date(날짜), interest\_rate(콜금리, 1일 전체거래)

출처 : 한국은행 경제통계시스템 (http://ecos.bok.or.kr/flex/EasySearch.jsp?langGubun=K&topCode=060Y001)

* 나스닥지수 데이터

변수 : Date(날짜), NASDAQ(나스닥 종가)

출처 : FinanceDataReader 데이터 수집 라이브러리

* 다우지수 데이터

변수 : Date(날짜), DOW(다우 종가)

출처 : FinanceDataReader 데이터 수집 라이브러리

* 유가 데이터

변수 : Date(날짜), WTI(서부텍사스유)

출처 : 한국석유공사

(http://www.petronet.co.kr/main2.jsp)

2-2. 독립변수와 종속변수

Index : Date

종속변수 : KOSDAQ 종목 Close(종가) 데이터,

KOSPI 종목 Adj Close(종가) 데이터

(3월 22일 기준 7월 22일에 가장 크게 증가한 KOSDAQ 종목5개,KOSPI 종목 5개)

독립변수 : dollar(기준환율), interest\_rate(콜금리, 1일 전체거래),

NASDAQ(나스닥 지수), DOW(다우 지수), WTI(서부텍사스유)

**II.본론**

1. KOSDAQ 종목 5개, KOSPI 종목 5개 추출

|  |
| --- |
| # 3월 20일 대폭락 이후 7월22일 기준 회복량이 가장 높은 KOSPI종목 Top5  less\_list = list()  max\_val = 0  save\_idx = list()  save\_per = list()  for idx in range(len(tmp)):  try:  a = tmp[idx].loc['2020-03-20', 'Adj Close']  b = tmp[idx].loc['2020-07-22', 'Adj Close']  per = (b-a)/a\*100  save\_idx.append(idx)  save\_per.append(per)  except KeyError:  print("에러가난 인덱스는 : ", idx)  per\_df = pd.DataFrame(save\_per, index = save\_idx)  per\_df.sort\_values(by=0, ascending=False, inplace=True) |

(해석)

가장 큰 회복량을 보여준 종목의 index 는 1위 421(삼성중공우),2위 538(신풍제약), 3위 897(두산퓨얼셀), 4위 275(녹십자홀딩스2우), 5위 353(일양약품우) 5종목이다.

하지만 3월 20일 종가를 기준으로 7월 22일 종가를 비교하였을때 가장 높은 회복량을 보여준 탑 5개 종목의 차트를 분석해보니 공통점은 3월 20일 대폭락장 이전과 이후에 가격의 변동이 없다가 호재의 영향으로만 높은 회복율을 띄었다. 우리가 원하는 조건의 종목이 아니라 다음 순위를 살펴보았다. 코로나로 인해 3월 20일에 저점을 찍고 ‘V’자 반등을 한 종목을 찾았다.

* KOSPI



(해석)

3월 22일 기준 ‘V자’ 반등한 KOSPI종목 중 7월 22일 종가가 가장 높은 비율로 증가한 상위 5개 종목을 추출하였다.

* KOSDAQ



(해석)

3월 22일 기준 ‘V자’ 반등한 KOSDAQ종목 중 7월 22일 종가가 가장 높은 비율로 증가한 상위 5개 종목을 추출하였다.

1. 데이터 전처리

**‘Date’인덱스 datetime64를 object타입으로 변경**

|  |
| --- |
| for i in range(7):  tmp3 = pd.Series (top10\_10years[i].index)  date\_str = tmp3.apply( lambda x:str(x)[:10] )  top10\_10years[i]['Date'] = date\_str.values  top10\_10years[i].set\_index(['Date'], inplace=True) |

**종목들의 날짜 교집합 구하기 해당날짜에 해당하는 종목 추출**

|  |
| --- |
| # 날짜를 리스트로 교집합 구하기  date = top10\_10years[0].index.unique() & top10\_10years[1].index.unique()& top10\_10years[2].index.unique()& top10\_10years[3].index.unique()& top10\_10years[4].index.unique()& top10\_10years[5].index.unique()& top10\_10years[6].index.unique()  top10 = list()  for i in range(len(top10\_10years)):  top10.append(top10\_10years[i].loc[date, :]) |

**환율데이터 불러오기, 결측치 제거**

|  |
| --- |
| path = '/content/drive/My Drive/데이터분석 프로젝트/data/krw\_usd.xlsx'  tmp\_dollar = pd.read\_excel(path, encoding='utf-8')  dollar = tmp\_dollar[['ÀÏÀÚ', '±âÁØÈ¯À²']]  dollar.columns = ['Date', 'dollar']  dollar.sort\_values('Date', ascending=True, inplace=True)  tmp = dollar.index  dollar['Date'] = tmp  dollar['Date'] = dollar['Date'].apply(lambda x : x.replace('.', '-'))  dollar.set\_index(['Date'], inplace=True)  dollar.dropna(axis=0, inplace=True) |

환율데이터를 불러오니 인코딩에 실패하여 알아 볼 수 없는 문자로 컬럼이 구성되어있었다. 컬럼명을 변경해주고 Date컬럼을 오름차순으로 정렬하고 종목 날짜 형식과 같은 형식으로 맞춰주고 Date를 index로 설정하고 결측치를 제거하였다.

**금리데이터 불러오기, 결측치 제거, 날짜형식 맞추기**

|  |
| --- |
| **path = '/content/drive/My Drive/데이터분석 프로젝트/data/interest\_rate\_10years.csv'**  **interest = pd.read\_csv(path, encoding='utf-8')**  **interest.columns=['Date', 'interest\_rate']**  **interest.dropna(axis=0, inplace=True)**  **interest.drop([0, 1, 2], inplace=True)**  **interest["Date"] = interest["Date"].apply(lambda x: x.replace('/', '-'))**  **interest.set\_index(['Date'], inplace=True)** |

**유가데이터를 불러오기, 결측치 제거, 날짜형식 맞추기**

|  |
| --- |
| **# 유가데이터는 1년단위로만 데이터를 가져올 수 있어서 10개의csv파일을 합쳐줘야 한다**  **path = '/content/drive/My Drive/데이터분석 프로젝트/data/유가'**  **col = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11']**  **tmp2 = list()**  **for i in range(11):**  **tmp2.append(pd.read\_csv(f'{path}/{col[i]}.csv', encoding='utf-8'))**  **for i in range(11):**  **del tmp2[i]['유종']**  **tmp2[i].columns = ['Date', 'WTI']**  **tmp2[i]['Date'] = tmp2[i]['Date'].apply(lambda x : f'2010-{x}')**  **tmp2[i]['Date'] = tmp2[i]['Date'].apply(lambda x : x.replace('\xa0', ''))**  **oil\_10years = pd.concat([tmp2[0], tmp2[1], tmp2[2], tmp2[3], tmp2[4], tmp2[5], tmp2[6], tmp2[7], tmp2[8], tmp2[9], tmp2[10]])**  **oil\_10years.set\_index(['Date'], inplace=True)** |

유가 데이터는 1년단위로 조회 다운가능하여 11개 파일은 다운받고 11개 파일을 열어서 날짜 전처리한 후에 11개 데이터 프레임을 pd.concat()함수를 이용해서 10년치 데이터를 합쳐준다.

**다우 지수, 나스닥 지수 데이터 불러오기, 결측치 제거, 날짜형식 맞추기**

|  |
| --- |
| **pip install -U finance-datareader**  **import FinanceDataReader as fdr**  **ixic = fdr.DataReader('IXIC', '2010-02-14', '2020-02-13')**  **ixic = ixic[['Close']]**  **dji = fdr.DataReader('DJI', '2010-02-14', '2020-02-13')**  **dji = dji[['Close']]**  **ixic.columns = ['NASDAQ']**  **dji.columns = ['DOW']**  **tmp3 = pd.Series (ixic.index)**  **date\_str = tmp3.apply( lambda x:str(x)[:10] )**  **ixic['Date'] = date\_str.values**  **ixic.set\_index(['Date'], inplace=True)**  **tmp3 = pd.Series (dji.index)**  **date\_str = tmp3.apply( lambda x:str(x)[:10] )**  **dji['Date'] = date\_str.values**  **dji.set\_index(['Date'], inplace=True)** |

FinanceDataReader라이브러리를 사용하여 다우, 나스닥 지수를 불러온다.

불러온 데이터중 ‘Close’컬럼만 뽑아낸다. ‘Date’인덱스의 날짜형식을 맞춰준다.

**종목, 유가, 환율, 금리, 다우, 나스닥 지수 데이터의 날짜 교집합을 찾는다.**

|  |
| --- |
| **date1 = date & dollar.index.unique() & interest.index.unique() & oil\_10years.index.unique() & dji.index.unique() & ixic.index.unique()** |

**찾은 공통된 날짜에 해당하는 행들만 뽑아서 데이터 프레임을 생성**

|  |
| --- |
| **interest = interest.loc[date1, :]**  **oil\_10years = oil\_10years.loc[date1, :]**  **dji = dji.loc[date1, :]**  **dollar = dollar.loc[date1, :]**  **ixic = ixic.loc[date1, :]**  **top7 = list()**  **for i in range(len(top10\_10years)):**  **top7.append(top10\_10years[i].loc[date1, :])**  **top7\_variable = list()**  **for i in range(len(top7)):**  **temp1 = pd.merge(top7[i], dollar, left\_index=True, right\_index=True)**  **temp2 = pd.merge(temp1, oil\_10years, left\_index=True, right\_index=True)**  **temp3 = pd.merge(temp2, ixic, left\_index=True, right\_index=True)**  **temp4 = pd.merge(temp3, dji, left\_index=True, right\_index=True)**  **top7\_variable.append(pd.merge(temp4, interest, left\_index=True, right\_index=True))** |

공통된 날짜의 행들만 뽑아내고 뽑아낸 데이터들을 merge를 통하여 하나의 데이터 프레임으로 생성

**컬럼들의 데이터타입을 확인하니 ‘WTI’, ‘interest\_rate’컬럼이 object로 되어있어서 float64형식으로 변경 및 결측치 제거**

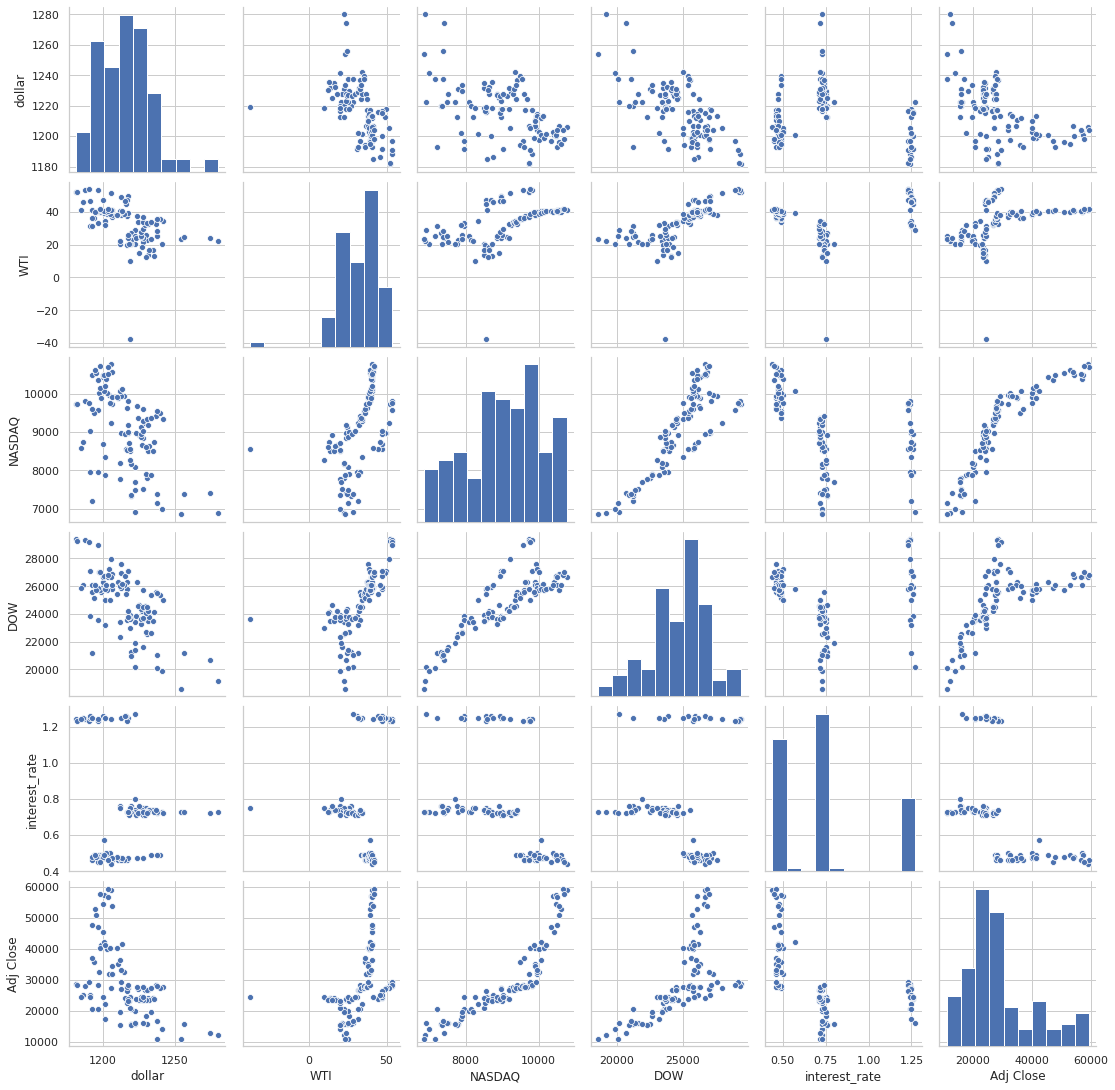
|  |
| --- |
| for i in range(1, len(top7\_variable)):  top7\_variable[i]['WTI'] = top7\_variable[i]['WTI'].apply(lambda x : float(x))  top7\_variable[i]['interest\_rate'] = top7\_variable[i]['interest\_rate'].apply(lambda x : float(x))  for i in range(len(top7\_variable)):  top7\_variable[i].dropna(axis=0, inplace=True) |

1. 다중회귀분석

3-1. 코로나 발생 이후 KOSPI 상위 5개 종목의 종가와 독립변수들간에 다중회귀분석

**산포행렬 그리기**

|  |
| --- |
| import numpy as np  import pandas as pd  import statsmodels.api as sm  import matplotlib.pyplot as plt  import seaborn as sns  #데이터 불러오기  path = '/content/top10\_variable.csv'  top10\_variable = pd.read\_csv( path, encoding = 'utf-8')  #‘Date’를 인덱스로 지정  top10\_variable.set\_index( 'Date', inplace=True)  #kospi지수 top 5을 종목별로 나누기  variable = list()  for i in range(5):  variable.append(top10\_variable[106\*i:(i+1)\*106])  #사용할 columns만 채택  cols = ['dollar', 'WTI', 'NASDAQ', 'DOW', 'interest\_rate', 'Adj Close']  #산포행렬그리기  sns.set( style='whitegrid' )  sns.pairplot( variable[0][ cols ], height=2.5 )  plt.show() |



(해석)

다중회귀 분석을 하였을 때 Adj. R-squared가 높으면 환율, 유가, 나스닥, 다우, 금리가 KOSPI 종목의 종가를 많이 설명할 수 있다.

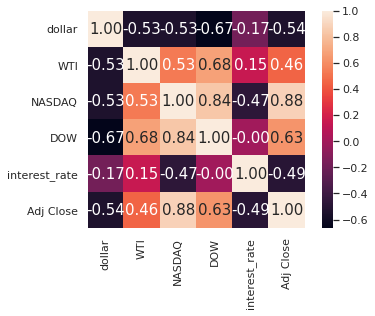
KOSPI 상위 5개 종목중에서 Adj. R-squared 가장 높은 ‘일진다이아’의 산포행렬을 그려보았다.

위의 그래프를 보면 ‘dollar’와 ‘Adj Close’는 음의 관계를 가지는것을 알 수 있다. ‘WTI’과 ‘interest\_rate(금리)’는 ‘Adj Close’와 낮은 관계를 보여주고 있다.

또한, ‘NASDAQ’과 ‘DOW’는 ‘Adj Close’는 강한 양의 관계를 보여준다.

**히트맵 그리기**

|  |
| --- |
| #상관분석하기 전에 상관분석할 columns들 확인하고 변경  corr\_tmp = np.corrcoef( variable[0][ cols ].values )  corr\_tmp.shape  variable[0][ cols ].values.T.shape  corr\_tmp = np.corrcoef( variable[0][ cols ].values.T )  #상관계수를 히트맵으로 확인  sns.heatmap(corr\_tmp, cbar=True, annot=True, square=True, fmt='.2f',  annot\_kws={'size':15}, yticklabels=cols, xticklabels=cols )  plt.show() |



(해석)

독립변수들과 종속변수간의 상관계수를 구하고 히트맵을 그려보았다.

히트맵을 보면 ‘dollar’와 ‘Adj Close’는 -0.54로 어느정도의 음의 관계가 있다. ‘WTI’와 ‘Adj Close’는 0.46으로 적절한 양의 관계를 보여주고 있다. 또한, ‘NASDAQ’과 ‘Adj Close’는 0.88로 강한 양의 관계가 있다.

‘DOW’와 ‘Adj Close’는 0.63으로 양의 관계를 보여준다. 또한, ‘interest\_rate’와 ‘Adj Close’는 -0.49로 어느정도의 음의 관계를 보여주고 있다.

**OLS 회귀 분석 결과**

|  |
| --- |
| #사용할 독립변수 리스트  columns = variable[0].columns[3:8]  #독립변수 데이터를 df으로 만들기  target\_df = variable[0][ columns ]  #종속변수를 Series로 만들기  y\_data = variable[0][ 'Adj Close' ]  #원데이터를 7 : 3으로 훈련용 데이터와 테스트 데이터로 나누고 난수 42로 실험환경을 통제  X\_train, X\_test, y\_train, y\_test = train\_test\_split( target\_df, y\_data, train\_size=0.7,  random\_state=42)  #augmentation:수식을 간단하게 구성하기 위해, 0이 아닌 상수값을 부여  X\_train\_added = sm.add\_constant(X\_train)  #독립변수 전체 적용 => OLS 회귀분석  full\_model = sm.OLS( y\_train, X\_train\_added)  full\_model\_res = full\_model.fit()  print( full\_model\_res.summary() ) |

OLS Regression Results

==============================================================================

Dep. Variable: Adj Close R-squared: 0.865

Model: OLS Adj. R-squared: 0.855

Method: Least Squares F-statistic: 86.96

Date: Wed, 29 Jul 2020 Prob (F-statistic): 3.66e-28

Time: 10:00:52 Log-Likelihood: -727.63

No. Observations: 74 AIC: 1467.

Df Residuals: 68 BIC: 1481.

Df Model: 5

Covariance Type: nonrobust

=================================================================================

coef std err t P>|t| [0.025 0.975]

---------------------------------------------------------------------------------

const 1.368e+05 6.71e+04 2.041 0.045 3029.253 2.71e+05

dollar -135.8584 48.553 -2.798 0.007 -232.744 -38.972

WTI 74.4755 56.635 1.315 0.193 -38.537 187.488

NASDAQ 16.9396 2.034 8.328 0.000 12.881 20.999

DOW -4.1674 0.782 -5.331 0.000 -5.727 -2.608

interest\_rate 6598.4883 4346.865 1.518 0.134 -2075.547 1.53e+04

==============================================================================

Omnibus: 6.288 Durbin-Watson: 2.015

Prob(Omnibus): 0.043 Jarque-Bera (JB): 6.238

Skew: 0.711 Prob(JB): 0.0442

Kurtosis: 2.943 Cond. No. 3.26e+06

==============================================================================

pH -0.3192 0.227 -1.404 0.161 -0.766 0.127

sulphates 0.8128 0.135 6.007 0.000 0.547 1.078

alcohol 0.2920 0.032 9.268 0.000 0.230 0.354

======================================================================

Omnibus: 29.060 Durbin-Watson: 2.001

Prob(Omnibus): 0.000 Jarque-Bera (JB): 50.192

Skew: -0.193 Prob(JB): 1.26e-11

Kurtosis: 3.963 Cond. No. 1.13e+05

======================================================================

(해석)

OLS Regression Results에서 Adj. R-squared를 보면 환율, 유가, 나스닥, 다우, 금리가 KOSPI 종가를 86.5%를 설명할 수 있다. 그리고 p값이 0.05가 넘는 ‘WTI’와 ‘interest\_rate’는 우연으로 KOSPI 종목의 종가를 설명할 수 있는 확률이 높은 것을 알 수 있다.

**순열과 조합**

|  |
| --- |
| from itertools import permutations, combinations |

**4개의 조합**

|  |
| --- |
| four\_cols = [ item for item in list( combinations( columns, 4 ) ) ]  len(three\_cols) |

**R제곱값이 가장 크고 다중공선성이 가장 작은 독립변수들의 조합**

|  |
| --- |
| four\_cols = [ item for item in list( combinations( columns, 4 ) ) ]  print( f'{"제거된컬럼":20} {"R제곱값":10} {"다중공선성":10}' )  print( '-'\*50 )  # r제곱이 가장 컸을대  best\_r\_idx = -1  best\_rqsrd = -1  best\_vif = -1  best\_n = 0  best\_v\_idx2 = -1  best\_rqsrd2 = -1  best\_vif2 = 10  best\_n2 = 0  for n in range(2, len(columns)):  four\_cols = [ item for item in list( combinations( columns, n ) ) ]  for idx, col in enumerate(four\_cols):  m\_model = sm.OLS( y\_train, sm.add\_constant( X\_train[ list(col) ] ) )  m\_model\_res = m\_model.fit()  vif = 1/(1-m\_model\_res.rsquared)  # 현재 r제곱이, 현재 최고 R제곱보다 크면 갱신  if m\_model\_res.rsquared > best\_rqsrd:  best\_rqsrd = m\_model\_res.rsquared  best\_r\_idx = idx  best\_vif = vif  best\_n = n  # 현재 vif가 현제 최고 vif값보다 작으면 갱신  if vif < best\_vif2:  best\_vif2 = vif  best\_v\_idx2 = idx  best\_rqsrd2= m\_model\_res.rsquared  best\_n2 = n  # 결과  print( "R제곱 기준", best\_r\_idx, best\_rqsrd, best\_vif, best\_n ) |

R제곱 기준 3 0.8613190530571184 7.210795873869172 4

(해석)

독립변수 조합들 중에서 R제곱이 가장 큰 조합의 인덱스는 3이고, R제곱은 0.86이므로 86% 설명할 수 있고, 조합의 개수는 4개이다.

**four\_cols에서 독립변수 조합**

|  |
| --- |
| print(four\_cols[ best\_r\_idx ]) |

('dollar', 'WTI', 'NASDAQ', 'DOW')

(해석)

R제곱 기준에서 best\_n1이 4인 독립변수 의 조합은 'dollar', 'WTI', 'NASDAQ', 'DOW' 인 것을 알 수 있다.

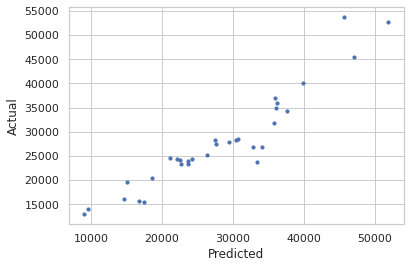
**현재 가장 좋은 점수를 받은 모델을 생성 -> 신규 데이터를 넣어서 예측**

|  |
| --- |
| target\_cols = list(four\_cols[ best\_r\_idx ])  y\_pred = m\_model\_res.predict( sm.add\_constant(X\_test[target\_cols]) ) |

**모델 학습과 예측을 통해 나온 R제곱값간의 비교**

|  |
| --- |
| m\_model = sm.OLS( y\_train, sm.add\_constant(X\_train[list(four\_cols[ best\_r\_idx ]) ] ) )  m\_model\_res = m\_model.fit()  m\_model\_res.rsquared  from sklearn.metrics import r2\_score  print("성능평가", r2\_score( y\_test, y\_pred ), m\_model\_res.rsquared)  plt.scatter(y\_pred, y\_test, s=10)  plt.xlabel("Predicted")  plt.ylabel("Actual")  plt.show() |

성능평가 0.8775682751443273 0.8613190530571184



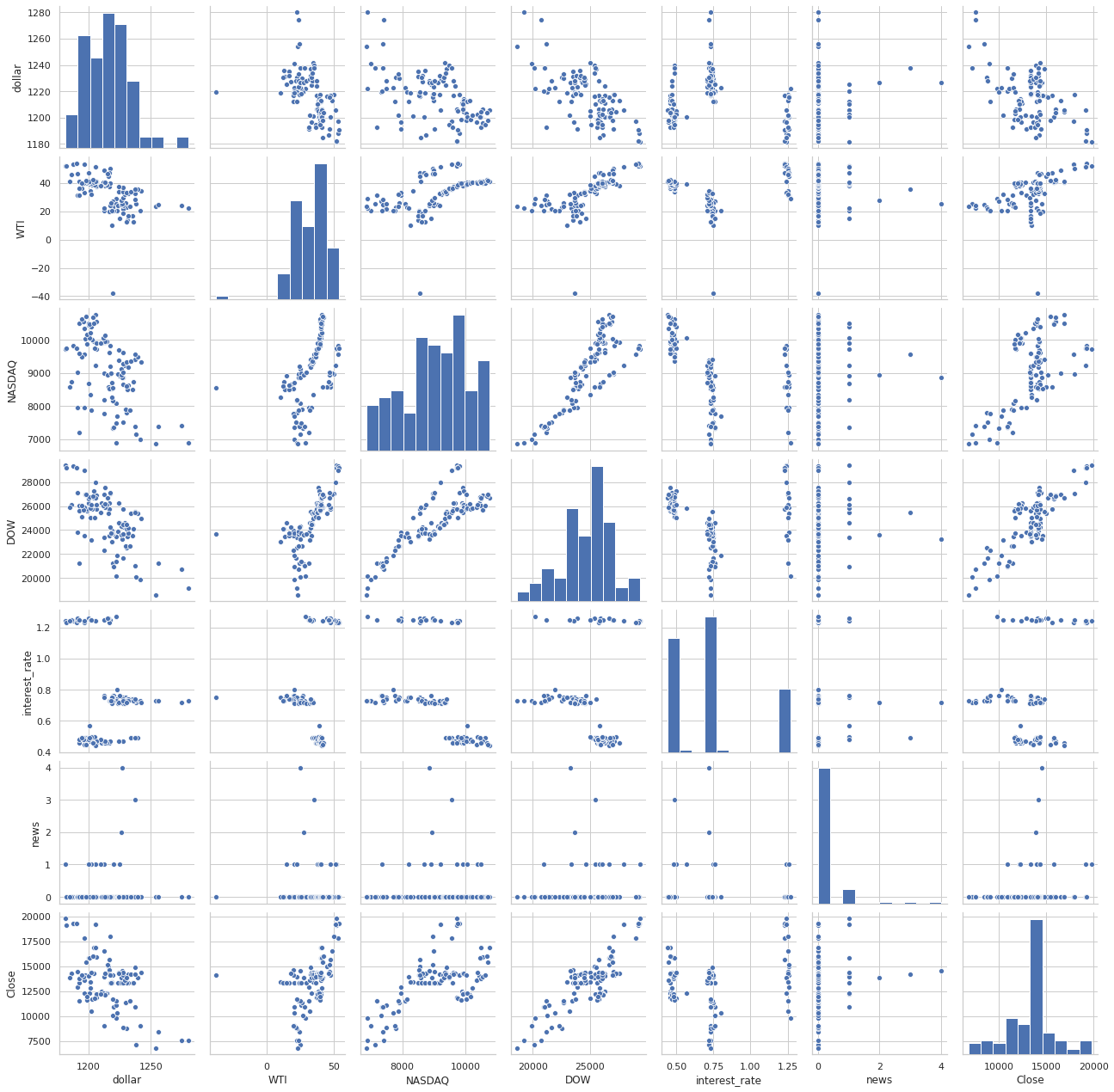
(해석)

예측결과와 y\_test결과 값의 성능평가의 정확도는 87%이며, 그래프에서 x축은 예측값이며 y축은 실제값인데 오차범위가 크지않은것을 확인 할 수 있다.

3-2. 코로나 발생 이후 KOSDAQ 다중회귀분석

**산포행렬 그리기**

|  |
| --- |
| import numpy as np  import pandas as pd  import statsmodels.api as sm  import matplotlib.pyplot as plt  import seaborn as sns  #데이터 불러오기  path = '/content/kosdaq\_top10AddNews.csv'  top10\_variable = pd.read\_csv( path, encoding = 'utf-8')  del top10\_variable['Unnamed: 0']  #kosdaqi지수 top 5을 종목별로 나누기  variable = list()  for i in range(5):  variable.append(top10\_variable[106\*i:(i+1)\*106])  #사용할 columns만 채택  cols = ['dollar', 'WTI', 'NASDAQ', 'DOW', 'interest\_rate','news', 'Close']  #산포행렬그리기  sns.set( style='whitegrid' )  sns.pairplot( variable[0][ cols ], height=2.5 )  plt.show() |



(해석)

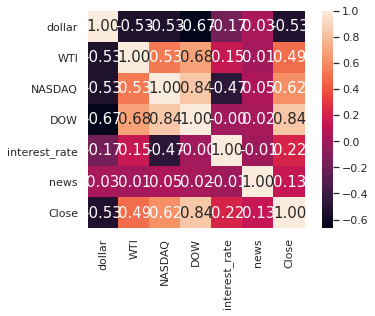
kosdaq을 분석할 때는 관련 news의 개수를 독립변수에 추가하여 다중회귀분석을 해보겠다.

kosdaq 지수 top 5 중에서 Adj. R-squared 가장 높은 ‘키이스트’의 산포행렬을 그려보았다.

위의 그래프를 보면 ‘dollar’와 ‘Adj Close’는 음의 관계를 가지는것을 알 수 있다. ‘WTI’, ‘interest\_rate’, ‘news’는 ‘Adj Close’와 낮은 관계를 보여주고 있다. 또한, ‘NASDAQ’과 ‘DOW’는 ‘Adj Close’는 강한 양의 관계를 보여준다.

**히트맵 그리기**

|  |
| --- |
| #상관분석하기 전에 상관분석할 columns들 확인하고 변경  corr\_tmp = np.corrcoef( variable[0][ cols ].values )  corr\_tmp.shape  variable[0][ cols ].values.T.shape  corr\_tmp = np.corrcoef( variable[0][ cols ].values.T )  #상관계수를 히트맵으로 확인  sns.heatmap(corr\_tmp, cbar=True, annot=True, square=True, fmt='.2f',  annot\_kws={'size':15}, yticklabels=cols, xticklabels=cols )  plt.show() |



(해석)

독립변수들과 종속변수간의 상관계수를 구하고 히트맵을 그려보았다.

히트맵을 보면 ‘dollar’와 ‘Adj Close’는 -0.53로 어느정도의 음의 관계가 있다. ‘WTI’와 ‘Adj Close’는 0.49으로 적절한 양의 관계를 보여주고 있다. 또한, ‘NASDAQ’과 ‘Adj Close’는 0.62로 양의 관계가 있다.

‘DOW’와 ‘Adj Close’는 0.84로 강한 양의 관계를 보여준다.

또한, ‘interest\_rate’와 ‘Adj Close’는 0.22로 어느정도의 약한 양의 관계를 보여주고 있다. 마지막으로 ‘news’와 ‘Adj Close’는 0.13.으로 연관성이 없다고 볼 수 있다.

**OLS 회귀 분석 결과**

|  |
| --- |
| #사용할 독립변수 리스트  columns = variable[0].columns[3:9]  #독립변수 데이터를 df으로 만들기  target\_df = variable[0][ columns ]  #종속변수를 Series로 만들기  y\_data = variable[0][ 'Close' ]  #원데이터를 7 : 3으로 훈련용 데이터와 테스트 데이터로 나누고 난수 42로 실험환경을 통제  X\_train, X\_test, y\_train, y\_test = train\_test\_split( target\_df, y\_data, train\_size=0.7,  random\_state=42)  #augmentation:수식을 간단하게 구성하기 위해, 0이 아닌 상수값을 부여  X\_train\_added = sm.add\_constant(X\_train)  #독립변수 전체 적용 => OLS 회귀분석  full\_model = sm.OLS( y\_train, X\_train\_added)  full\_model\_res = full\_model.fit()  print( full\_model\_res.summary() ) |

OLS Regression Results

==============================================================================

Dep. Variable: Close R-squared: 0.827

Model: OLS Adj. R-squared: 0.812

Method: Least Squares F-statistic: 53.40

Date: Thu, 30 Jul 2020 Prob (F-statistic): 1.25e-23

Time: 01:13:46 Log-Likelihood: -621.30

No. Observations: 74 AIC: 1257.

Df Residuals: 67 BIC: 1273.

Df Model: 6

Covariance Type: nonrobust

=================================================================================

coef std err t P>|t| [0.025 0.975]

---------------------------------------------------------------------------------

const -6.159e+04 1.61e+04 -3.814 0.000 -9.38e+04 -2.94e+04

dollar 32.9977 11.698 2.821 0.006 9.649 56.347

WTI -42.6601 13.566 -3.145 0.002 -69.738 -15.583

NASDAQ 1.2628 0.492 2.566 0.013 0.281 2.245

DOW 0.8448 0.188 4.486 0.000 0.469 1.221

interest\_rate 5127.6768 1045.493 4.905 0.000 3040.864 7214.490

news 460.9873 199.099 2.315 0.024 63.583 858.391

==============================================================================

Omnibus: 0.573 Durbin-Watson: 1.630

Prob(Omnibus): 0.751 Jarque-Bera (JB): 0.714

Skew: 0.163 Prob(JB): 0.700

Kurtosis: 2.646 Cond. No. 3.27e+06

==============================================================================

(해석)

OLS Regression Results에서 Adj. R-squared를 보면 환율, 유가, 나스닥, 다우, 금리, news가 KOSDAQ 종가를 82.7%를 설명할 수 있다.

**순열과 조합**

|  |
| --- |
| from itertools import permutations, combinations |

**4개의 조합**

|  |
| --- |
| four\_cols = [ item for item in list( combinations( columns, 4 ) ) ]  len(three\_cols) |

**R제곱값이 가장 크고 다중공선성이 가장 작은 독립변수들의 조합**

|  |
| --- |
| four\_cols = [ item for item in list( combinations( columns, 4 ) ) ]  print( f'{"제거된컬럼":20} {"R제곱값":10} {"다중공선성":10}' )  print( '-'\*50 )  # r제곱이 가장 컸을대  best\_r\_idx = -1  best\_rqsrd = -1  best\_vif = -1  best\_n = 0  best\_v\_idx2 = -1  best\_rqsrd2 = -1  best\_vif2 = 10  best\_n2 = 0  for n in range(2, len(columns)):  four\_cols = [ item for item in list( combinations( columns, n ) ) ]  for idx, col in enumerate(four\_cols):  m\_model = sm.OLS( y\_train, sm.add\_constant( X\_train[ list(col) ] ) )  m\_model\_res = m\_model.fit()  vif = 1/(1-m\_model\_res.rsquared)  if m\_model\_res.rsquared > best\_rqsrd:# 현재 r제곱이, 현재 최고 R제곱보다 크면 갱신  best\_rqsrd = m\_model\_res.rsquared  best\_r\_idx = idx  best\_vif = vif  best\_n = n    if vif < best\_vif2:# 현재 vif가 현제 최고 vif값보다 작으면 갱신  best\_vif2 = vif  best\_v\_idx2 = idx  best\_rqsrd2= m\_model\_res.rsquared  best\_n2 = n  # 결과  print( "R제곱 기준", best\_r\_idx, best\_rqsrd, best\_vif, best\_n ) |

R제곱 기준 0 0.813208607416722 5.353565740745605 5

(해석)

독립변수 조합들 중에서 R제곱이 가장 큰 조합의 인덱스는 0이고, R제곱은 0.81이므로 81% 설명할 수 있고, 조합의 개수는 5개이다.

**best\_n1이 4이므로 마지막에 세팅된 조합 four\_cols에서 독립변수 조합 확인**

|  |
| --- |
| print(four\_cols[ best\_r\_idx]) |

('dollar', 'WTI', 'NASDAQ', 'DOW', 'interest\_rate')

(해석)

R제곱 기준에서 best\_n1이 5인 독립변수 의 조합은 'dollar', 'WTI', 'NASDAQ', 'DOW', 'interest\_rate' 인 것을 알 수 있다.

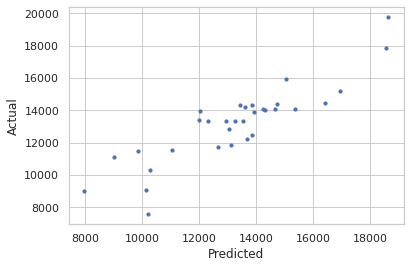
**현재 가장 좋은 점수를 받은 모델을 생성 -> 신규 데이터를 넣어서 예측**

|  |
| --- |
| target\_cols = list(four\_cols[ best\_r\_idx ])  y\_pred = m\_model\_res.predict( sm.add\_constant(X\_test[target\_cols]) ) |

**모델 학습과 예측을 통해 나온 R제곱값간의 비교**

|  |
| --- |
| m\_model = sm.OLS( y\_train, sm.add\_constant(X\_train[list(four\_cols[ best\_r\_idx ]) ] ) )  m\_model\_res = m\_model.fit()  m\_model\_res.rsquared  from sklearn.metrics import r2\_score  print("성능평가", r2\_score( y\_test, y\_pred ), m\_model\_res.rsquared)  plt.scatter(y\_pred, y\_test, s=10)  plt.xlabel("Predicted")  plt.ylabel("Actual")  plt.show() |

성능평가 0.7555902913917003 0.813208607416722



(해석)

예측결과와 y\_test결과 값의 성능평가의 정확도는 75%이며, 그래프에서 x축은 예측값이며 y축은 실제값인데 오차범위가 조금 있는 것을 확인 할 수 있다.

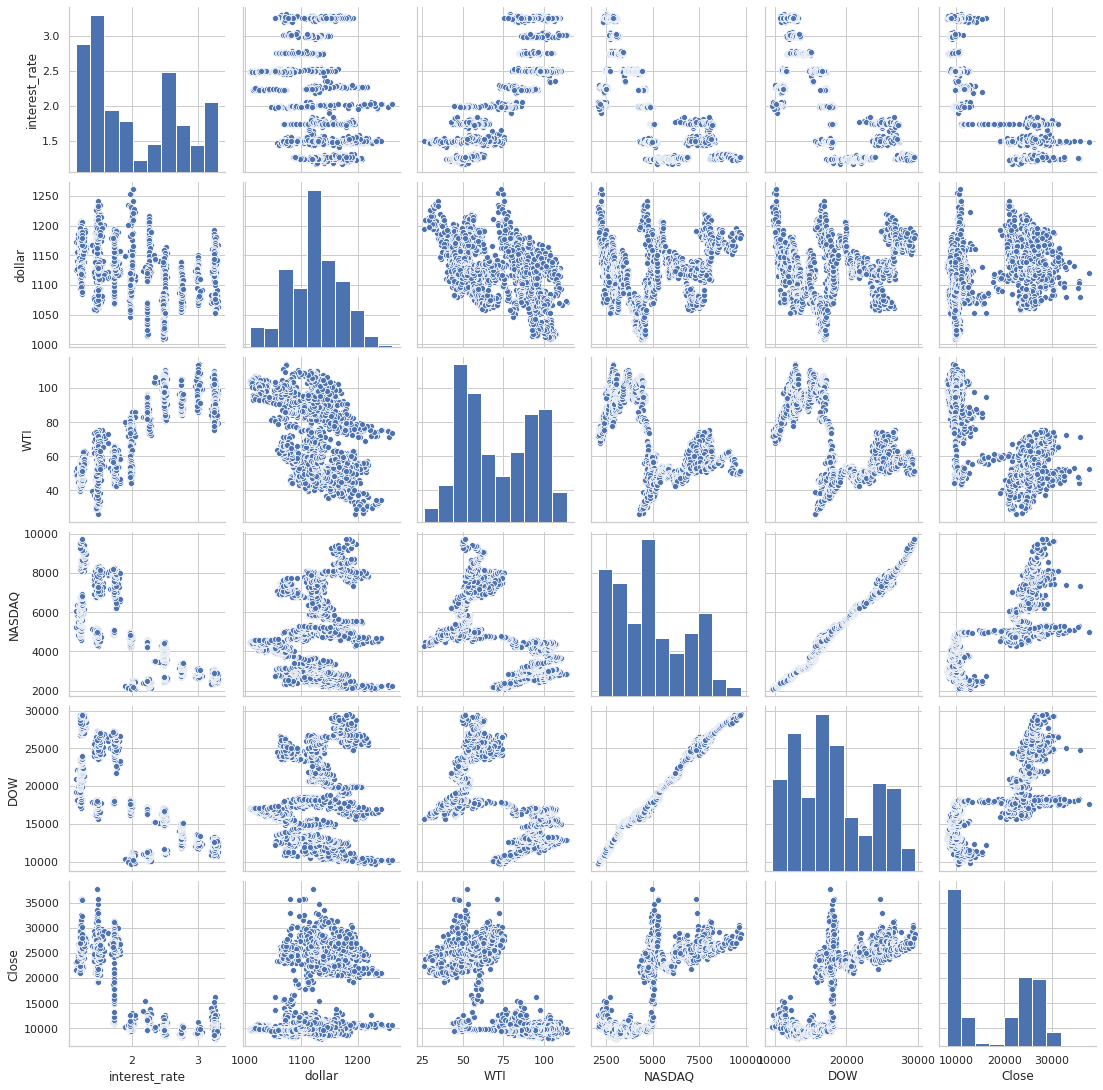
3-3. 코로나 발생 이전 10년 동안의 KOSPI 다중회귀분석

**코로나 발생 이후 V자 반등한 top10개 중 7개 종목의 10년 전 KOSPI 지수**

|  |
| --- |
| start\_date = '2010-02-14'#'2010-01-01'  end\_date = '2020-02-13'  code = ['081000.ks', '026890.ks', '298040.ks', '001065.ks', '144620.ks', '000725.ks', '047400.ks', '000157.ks', '005360.ks', '192650.ks']  tmp = list()  for i in range(len(top10)):  TRADE\_CODE = code[i]  tmp1 = data.get\_data\_yahoo( TRADE\_CODE, start\_date, end\_date)  if len(tmp1) > 2300:  tmp.append(tmp1)  print(i)    # 일진다이아 081000  # 디피씨 026890  # JW중외제약우 001065  # 현대건설우 000725  # 유니온머티리얼 047400  # 두산2우B 000157  # 모나미 005360  top10\_10years = list()  for i in range(len(tmp)):  top10\_10years.append(tmp[i][['Close']]) |

**산포행렬그리기**

|  |
| --- |
| sns.set(style='whitegrid')  sns.pairplot(top7\_variable[i][cols], height=2.5)  plt.show() |



(해석)

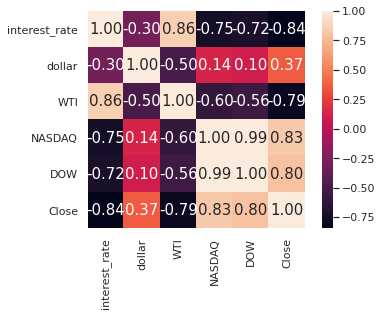
코로나 발생 이후 V자 반등한 top10개 중 7개 종목의 10년 전 KOSPI 지수와 독립변수인 환율, 유가, 나스닥, 다우, 금리와 다중회귀분석을 해보겠다.

KOSPI 상위 5개 종목중에서 Adj. R-squared 가장 높은 ‘JW중외제약우 ’의 산포행렬을 그려보았다.

위의 그래프를 보면 ‘interest\_rate’는 ‘Adj Close’와 음의 관계를 보여주고 있다. 또한, ‘NASDAQ’과 ‘DOW’는 ‘Adj Close’는 양의 관계를 보여준다.

**상관계수 구하고 히트맵 그리기**

|  |
| --- |
| corr\_tmp = np.corrcoef(top7\_variable[i][cols].values.T)  sns.heatmap(corr\_tmp, cbar=True, annot=True, square=True, fmt='.2f',  annot\_kws={'size':15}, yticklabels=cols, xticklabels=cols)  plt.show() |



(해석)

독립변수들과 종속변수간의 상관계수를 구하고 히트맵을 그려보았다.

히트맵을 보면 ‘interest\_rate’와 ‘Adj Close’는 -0.84로 높은 음의 관계가 있다. ‘dollar’와 ‘Adj Close’는 0.37로 낮은 관계를 보여준다.

‘WTI’와 ‘Adj Close’는 -0.79로 높은 음의 관계를 보여주고 있다.

또한, ‘NASDAQ’과 ‘Adj Close’는 0.8이 넘는 강한 양의 관계를 보여주고 있다.

**OLS 회귀분석 결과**

|  |
| --- |
| target\_df = top7\_variable[i][columns]  y\_data = top7\_variable[i]['Close']  X\_train, X\_test, y\_train, y\_test = train\_test\_split(target\_df, y\_data, train\_size=0.7,  random\_state=42)  X\_train.shape, X\_test.shape, y\_train.shape, y\_test.shape  X\_train\_added = sm.add\_constant(X\_train)  full\_model= sm.OLS(y\_train, X\_train\_added)  full\_model\_res = full\_model.fit()  print(full\_model\_res.summary()) |

OLS Regression Results

==============================================================================

Dep. Variable: Close R-squared: 0.838

Model: OLS Adj. R-squared: 0.838

Method: Least Squares F-statistic: 1646.

Date: Thu, 30 Jul 2020 Prob (F-statistic): 0.00

Time: 05:19:15 Log-Likelihood: -15117.

No. Observations: 1593 AIC: 3.025e+04

Df Residuals: 1587 BIC: 3.028e+04

Df Model: 5

Covariance Type: nonrobust

=================================================================================

coef std err t P>|t| [0.025 0.975]

---------------------------------------------------------------------------------

const 6810.3047 2854.143 2.386 0.017 1212.017 1.24e+04

dollar 15.0603 2.160 6.971 0.000 10.823 19.298

WTI -83.0089 8.245 -10.068 0.000 -99.181 -66.837

NASDAQ 4.8586 0.505 9.616 0.000 3.868 5.850

DOW -1.0182 0.178 -5.716 0.000 -1.368 -0.669

interest\_rate -2729.6771 295.212 -9.247 0.000 -3308.723 -2150.631

==============================================================================

Omnibus: 85.952 Durbin-Watson: 1.945

Prob(Omnibus): 0.000 Jarque-Bera (JB): 287.043

Skew: -0.156 Prob(JB): 4.67e-63

Kurtosis: 5.056 Cond. No. 6.96e+05

==============================================================================

(해석)

OLS Regression Results에서 Adj. R-squared를 보면 환율, 유가, 나스닥, 다우, 금리가 KOSPI 종가를 83.8%를 설명할 수 있다.

**R제곱이 큰 조합과 VIF가 작은 조합**

|  |
| --- |
| # r제곱이 가장 컸을때  best\_r\_idx = -1  best\_rsqrd = -1  best\_vif = -1  best\_n1 = 0  best\_v\_idx = -1  best\_rsqrd2 = -1  best\_vif2 = 10  best\_n2 = 0  for n in range(2, len(columns)):  five\_cols = [item for item in list(combinations(columns, n ))]  for idx, col in enumerate(five\_cols):  m\_model = sm.OLS( y\_train, sm.add\_constant(X\_train[list(col)]))  m\_model\_res = m\_model.fit()  vif = 1/(1-m\_model\_res.rsquared)  if m\_model\_res.rsquared > best\_rsqrd:  best\_rsqrd = m\_model\_res.rsquared  best\_r\_idx = idx  best\_vif = vif  best\_n1 = n  if vif < best\_vif2:  best\_v\_idx = idx  best\_rsqrd2 = m\_model\_res.rsquared  best\_vif2 = vif  best\_n2 = n    # 결과  print("R제곱 기준", best\_r\_idx, best\_rsqrd, best\_vif, best\_n1) |

R제곱 기준 1 0.834989206719781 6.0602096391465405 4

(해석)

독립변수 조합들 중에서 R제곱이 가장 큰 조합의 인덱스는 1이고, R제곱은 0.83이므로 83% 설명할 수 있고, 조합의 개수는 4개이다.

**best\_n1이 4이므로 마지막에 세팅된 조합 five\_cols에서 독립변수 조합 확인**

|  |
| --- |
| print("최고의 조합 : ", five\_cols[best\_r\_idx]) |

최고의 조합 : ('dollar', 'WTI', 'NASDAQ', 'interest\_rate')

(해석)

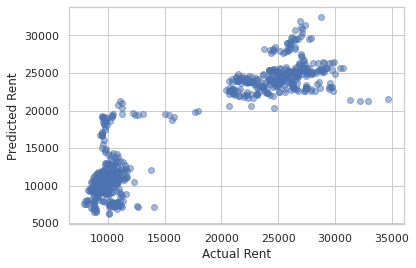
R제곱 기준에서 best\_n1이 4인 독립변수 의 조합은 'dollar', 'WTI', 'NASDAQ', 'interest\_rate' 인 것을 알 수 있다.

**현재 가장 좋은 점수를 받은 모델을 생성 -> 신규 데이터를 넣어서 예측**

**모델 학습과 예측을 통해 나온 R제곱값간의 비교**

|  |
| --- |
| m\_model = sm.OLS( y\_train, sm.add\_constant(X\_train[list(five\_cols[best\_r\_idx])]))  m\_model\_res = m\_model.fit()  target\_cols = list(five\_cols[best\_r\_idx])  y\_pred = m\_model\_res.predict( sm.add\_constant(X\_test[target\_cols]))  print('성능평가 결과 : ', r2\_score(y\_test, y\_pred))  plt.scatter(y\_test, y\_pred, alpha=0.5)  plt.show() |

성능평가 0.849554687271689 0.834989206719781



(해석)

예측결과와 y\_test결과 값의 성능평가의 정확도는 85%이며, 그래프에서 x축은 실제값이며 y축은 예측값인데 오차범위가 크지않은것을 확인 할 수 있다.

**III. 결론**

분석결과 코로나 이후에는 달러가격이 상승하면 주가는 하락하고 달러가격이 하락하면 주가가 상승하는 반비례 관계를 보여주고 있으며 나스닥 지수와 다우 지수와는 비례관계를 나타내고 있다.

코로나 이전에는 금리와 유가가 상승하면 주가는 하락하고 금리와 유가가 하락하면 주가는 상승하는 반비례 관계를 보여주고 있고 나스닥 지수와 다우 지수와는 비례 관계를 나타내고 있다.

코로나와 같은 악재상황 발생여부와 관계없이 우리나라 주가는 다우 지수와 나스닥 지수에 강한 상관관계를 가지고 있기때문에 투자를 할때 참고할만한 요소이며

코로나와 같은 악재상황이 발생했을때는 환율을 어느정도 고려하여 투자를 하면 좋을 것이다.

악재상황이 아닌 상황일때는 유가, 금리와 강한 반비례 관계를 가지고 있기 떄문에 투자를 할때 유가, 금리를 고려하여 투자를 하면 좋을 것이다.