대학원 시계열 분석 기말레포트

시계열 분석을 통한 Walmart 판매 예측

202140191 엄태훈

1. 서론

순차적(Sequential) 데이터의 종류 중 하나인 시계열 데이터는 데이터가 시간에 따라 구성돼 있는 것을 말한다. 시계열 데이터는 과거의 데이터가 현재 또는 미래에 영향을 줄 수도 있고 계절에 따라 데이터의 패턴이 다르게 나타날 수 있기 때문에, 분석을 진행함에 있어 정상성(stationary) 만족, 계절성 제거 등 다양한 요소들을 고려해야 한다. 대표적인 시계열 데이터로는 일별 주식데이터, 연도별 경제성장률, 연도별 인구수 등이 있다. 이러한 시계열 데이터를 적절하게 분석하여 꽤나 정확한 예측을 할 수 있는 모델을 만들게 된다면 회사 성장을 미리 예측하여 주식을 미래보다 더 저렴한 가격에 매수하여 수익을 실현할 수 있고 미래의 경제 상황을 예측하여 문제 해결을 위한 사전정책을 수립하는 등 실생활에 굉장히 유용한 이익을 가져다줄 수 있다.

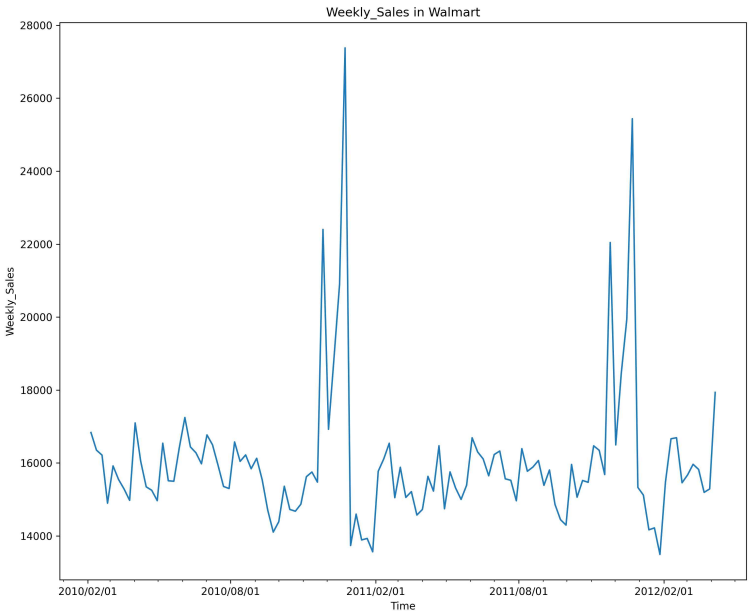
본 보고서에서는 마트의 주간 판매량 데이터를 기존 시계열 모형에 적합하여 미래의 주간 판매량을 예측해 보고자한다. 마트에서의 판매량 예측을 통해 필요한 물품의 재고를 사전에 파악하여 재고가 과도하게 쌓임으로써 발생할 수 있는 손실을 방지할 수 있다. 또한, 본 보고서에서는 적절한 데이터를 찾지 못하여 다루지는 못했지만, 품목별 판매량 데이터를 분석하여 마트의 품목별 판매량 예측을 할 수도 있다. 이를 통해, 가장 잘 팔리거나 잘 팔리지 않는 품목을 미리 예측할 수 있다. 이는 마트의 물품 위치를 정하고자 할 때 상당한 도움지표가 될 수 있고 이 또한 마트의 수익 증가를 이끌 수 있게 될 것이다.

2. 본론

2.1 데이터 설명 및 정상성 확인

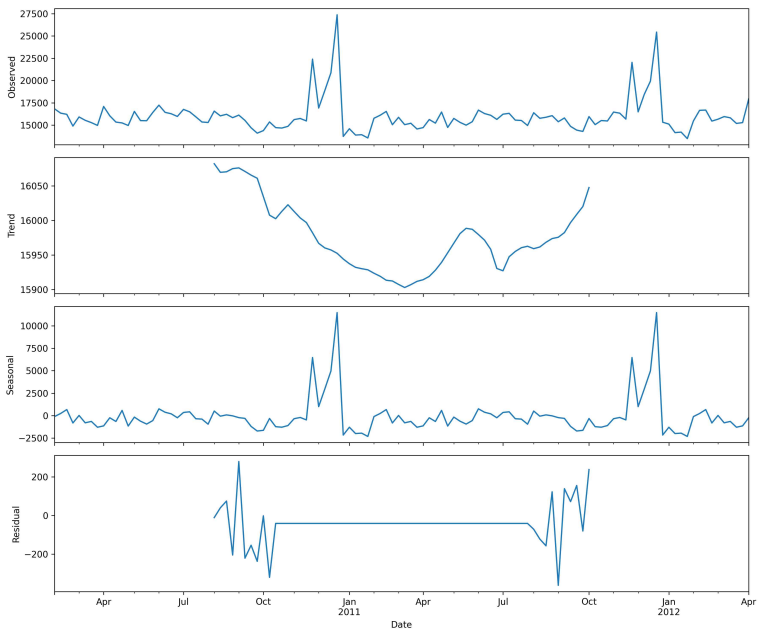
시계열 분석을 진행하기 위해 데이터는 Kaggle 사이트에 있는 Walmart Dataset을 사용하였다. Walmart는 미국의 Target과 더불어 가장 큰 대표적인 소매점 중 하나이다. 데이터의 개수는 총 421,570개이며, 2010년 2월 5일부터 2012년 10월 26일까지 약 2년 동안의 45개의 지점에 대한 주간 판매량(Weekly_Sales) 변수를 이용하여 분석을 진행하였다. 데이터 전처리 과정에서 45개 지점의 주간 판매량을 하나의 주간 판매량으로 통일하기 위해 주간 별 45개 지점의 평균을 각 주간 판매량 변수로 사용하였다. 또한, 데이터의 80%(2010년 2월 5일 ~ 2012년 4월 6일)를 시계열 모형 적합을 위한 훈련데이터로 사용하였고 나머지 20%(2012년 4월 13일 ~ 2012년 10월 26일)는 적합된 모형을 평가하기 위한 평가데이터로 사용하였다.

분석을 진행하기에 앞서, 데이터의 시간에 따른 주간 판매량 파악과 데이터가 정상성을 만족하는지 살펴보기 위해 그래프를 그려보았다.



[그림 1: Walmart 판매량의 2년 변화 추이]

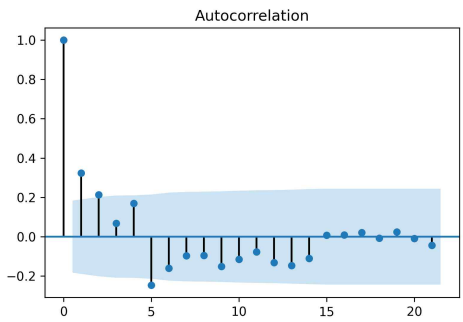
2010년 2월 5일 ~ 2012년 4월 6일의 45개 지점의 Walmart의 주간 판매량 평균 그래프를 그린 결과는 위와 같다. 시간에 따라 데이터가 일정하게 증가하거나 감소하지 않는 것을 볼 수 있고 특별한 계절성을 가지지 않는 것으로 보아 데이터가 정상성(Stationary)을 만족하는 것처럼 보인다. 또한, 특정 한 달간 판매량이 급격하게 늘어나는 곳을 두 군데 볼 수 있다. 확인 결과, 2010년 12월, 2011년 12월에 판매량이 급증한 것으로 확인 되었으며, 이는 크리스마스의 영향으로 인해 판매량이 증가한 것으로 보인다.



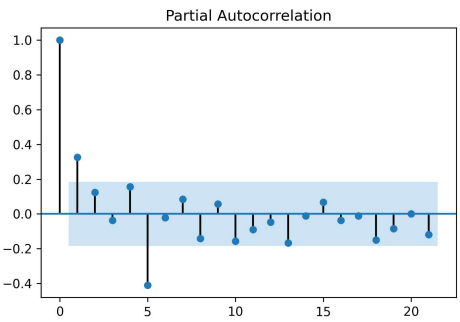
[그림 2: Walmart 판매량의 시계열 분해]

Walmart 데이터의 계절성을 보다 더 정확하게 확인하기 위해 시계열 분해를 진행해 보았다. 시계열 분해 결과, 특정 지점에서 급속히 증가하는 패턴이 나타나지만 반복적으로 일어나지 않으며, 나머지 구간에서도 특정한 패턴이 보이지 않아 계절성은 존재하지 않는다고 판단할 수 있다. 또한, 시계열 데이터가 꾸준히 증가하는지 감소하는지 나타내는 추세선 또한 특정한 규칙이 보이지 않고 잔차 또한 특별한 것이 보이지 않아 계절성은 제거하지 않고 분석을 진행하였다.

분석을 진행하기에 앞서, 시간별 그래프의 변화만으로 간단하게 정상성을 판단할 수 있지만 보다 더 정확한 정상성 판단을 위해 ACF, PACF 그래프와 단위근 검정을 실시하였다. ACF(Autocorrelation function)는 자기상관함수를 말하며, 현재 시점을 T라고 할 때, K이전의 시점(T-K)과의 상관관계를 나타내며, PACF(Partial autocorrelation function)은 T-K부터 T시점까지 범위내에 모든 시점을 고려하는 ACF와 달리, 외부변수를 제거하고 오직 T시점과 T-K시점만을 고려한 상관 관계를 말한다. ACF, PACF값이 급격하게가 아닌 완만하게 감소하게 되면 과거의 시점이 미래의 시점에 큰 영향을 미치고 있다는 신호이기 때문에 정상성을 만족하지 않는다는 결과를 얻을 수 있다. 단위근은 확률론의 데이터 검정에서 쓰이는 개념이다. 일반적으로 시계열 데이터는 일정한 규칙을 가정하기 때문에 매우 복잡한 형태를 갖는 시계열 데이터라도 단위근을 통해 어떻게든 단순화시킬 수 있을 것이라 생각할 수 있다. 단위근이 존재하면 정상 시계열이 아니라고 판단하게 된다. 단위근을 검정할 수 있는 방법 중 하나는 ADF-TEST로 시계열에서 단위근(unit root)가 존재하는지의 여부를 검정함으로써 시계열 데이터가 정상성을 만족하는지 확인하는 방법이다. 앞에 두 가지 방법을 이용하여 시계열 데이터의 정상성을 확인해 보았다.



[그림 3: Walmart의 ACF]



[그림 4: Walmart의 PACF]

Walmart 주간 판매량 데이터의 ACF, PACF를 그린 결과는 위와 같다. ACF, PACF 모두 값이 완만하게 감소하는 것이 아니라 시점 2에서부터 급격하게 감소하여 0으로 수렴하는 것을 볼 수 있다. 따라서, 주어진 데이터는 앞선 그래프에서도 확인할 수 있듯이 정상성을 만족하는 데이터임을 알 수 있다.

Augmented Dickey-Fuller test	
Test Statistic	-5.3113391203880544
P-value	5.18715388095781e-06
#Lags Used	4
Number of Observations Used	109
Critical Value (1%)	-3.49181775886872
Critical Value (5%)	-2.5811201893779985
Critical Value (10%)	-2.8884437992971588

[표 1: Walmart의 ADF-Test 결과]

Walmart 주간 판매량 데이터의 ADF-Test(단위근 검정)결과는 위와 같다. P-value가 약 5.19e-6으로 유의수준 0.05보다 매우 작은 값을 가지므로 귀무가설(H_0: 단위근이 존재한다. 즉, 정상 시계열이 아니다)를 기각한다. 따라서 주어진 데이터는 정상시계열 데이터임을 알 수 있으며, 별도의 차분이 필요하지 않은 것 또한 확인할 수 있다.

2.2 모형 적합

Walmart의 주간 판매량 예측을 위해 AR(2), MA(2), ARMA(2,2), ARMA(2,2)-GARCH(1,1) 모형을 적합해보기로 하였다. ADF-Test결과, 별도의 차분은 필요하지 않으므로 ARIMA 모형은 고려하지 않았다. 또한, 모형들의 차수를 2차로 고려한 이유는 ACF, PACF그래프를 그려보았을 때, 시점이 2일 때부터 값들이 급격하게 감소하기 시작하였기 때문에 2차 모형을 고려하였다.

첫 번째로, AR(2)모형에 데이터를 적합 시켜 보았다. AR(Autogressive models)모델은 현재 시점 T의 값을 나타내고자 할 때, 과거의 데이터를 통해 함수로 표현할 수 있는 모델을 말하며, AR(P) 모델을 수식으로 표현하면 아래와 같다.

$$x_t = \phi_0 + \phi_1 x_{t-1} + \phi_2 x_{t-2} + \ldots + \phi_p x_{t-p} + w_t,\ w_t \sim N(0, \sigma_w^2)$$

이러한 AR(2) 모델을 Walmart 데이터에 적합시킨 결과는 다음과 같다.

AR Model Results						
Dep. Variable	Weekly_Sales		No. Observations		114	
Model	AR(2)		Log Likelihood		-1034.992	
Method	css-mle		S.D. of innovations		2091.529	
AIC	2075.985		BIC		2084.193	
Sample	0		HQIC		2079.316	
==========						
==========						
	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1.Weekly_sales	0.5835	0.085	6.830	0.000	0.416	0.751
ar.L2.Weekly_sales	0.4110	0.086	4.798	0.000	0.243	0.579

[표 2: Walmart의 AR(2) 모형 결과]

AR 모형의 1차 계수와 2차 계수 모두 P-value가 0.000으로 유의수준 0.05보다 매우 작은 것을 확인할 수 있다. 따라서, AR(2)의 계수는 모두 유의한 것을 확인할 수 있다. 추정된 AR(2) 모형의 식은 아래와 같다.

$$x_t = 0.5835x_{t-1} + 0.4110x_{t-2} + w_t,\ w_t \sim N(0,\sigma_w^2)$$

두 번째로, MA(2)모형에 데이터를 적합 시켜 보았다. MA(Moving average models) 모델은 현재 시점 T의 값을 나타내고자 할 때, AR모형과 달리 과거의 오차항을 통해 함수로 표현할 수 있는 모델을 말하며, 이때, 각 오차항은 서로 독립이다. MA(Q) 모형을 수식으로 표현하면 아래와 같다.

$$x_t = w_t + \theta_1 w_{t-1} + \theta_2 w_{t-2} + ... + \theta_Q w_{t-Q},\ w_t \sim N(0,\sigma_w^2)$$

이러한 MA(2) 모형을 Walmart데이터에 적합시킨 결과는 다음과 같다.

MA Model Results						
Dep. Variable	Weekly_Sales		No. Observations		114	
Model	MA(2)		Log Likelihood		-1144.589	
Method	css-mle		S.D. of innovations		5336.422	
AIC	2295.178		BIC		2303.387	
Sample	0		HQIC		2298.510	
==========						
	coef	std err	z	P>|z|	[0.025	0.975]
ma.L1.Weekly_sales	1.4744	0.043	34.282	0.000	1.390	1.559
ma.L2.Weekly_sales	0.9999	0.043	23.289	0.000	0.916	1.084

[표 3: Walmart의 MA(2) 모형 결과]

MA(2)모형의 1차계수와 2차계수 모두 P-value가 0.000으로 유의수준 0.05보다 매우 작은 것을 확인할 수 있다. 따라서, MA(2)의 계수는 모두 유의한 것을 알 수 있다. MA(2) 모형의 추정된 식은 다음과 같다.

$$x_t = w_t + 1.4744w_{t-1} + 0.9999w_{t-2},\ w_t \sim N(0,\sigma^{2_w})$$

세 번째로, ARMA(2,2)모형에 데이터를 적합시켜 보았다. ARMA(Autoregressive Moving average Model)은 AR모형과 MA모형이 결합된 형태로, 현재 시점 T의 값을 나타내고자 할 때, 과거의 데이터 자료와 오차항을 모두 사용하여 함수로 표현하게 된다. ARMA(P,Q) 모형을 수식으로 표현하면 다음과 같다.

$$x_t = \theta_1 x_{t-1} + ...\theta_p x_{t-p} + w_t + \theta_1 w_{t-1} + ... + \theta_Q w_{t-Q}, \sim N(0,\sigma^{2_w})$$

이러한 ARMA(2,2) 모형을 Walmart데이터에 적합시킨 결과는 다음과 같다.

ARMA Model Results						
Dep. Variable	Weekly_Sales		No. Observations		114	
Model	ARMA(2,2)		Log Likelihood		-1029.622	
Method	css-mle		S.D. of innovations		1982.653	
AIC	2069.245		BIC		2082.926	
Sample	0		HQIC		2074.797	
==========						
==========						
	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1.Weekly_sales	0.1137	0.120	0.950	0.344	-0.121	0.348
ar.L2.Weekly_sales	0.8838	0.120	7.386	0.000	0.649	1.118
ma.L1.Weekly_sales	0.2590	0.163	1.588	0.115	-0.061	0.579
ma.L2.Weekly_sales	-0.4724	0.146	-3.234	0.002	-0.759	-0.186

[표 4: Walmart의 ARMA(2,2) 모형 결과]

ARMA(2,2)모형에서 AR의 1차계수와 MA의 1차계수의 P-value는 각각 0.344, 0.115로 유의수준 0.05보다 큰 것을 알 수 있고 AR의 2차계수와 MA의 2차계수의 P-value는 각각 0.000, 0.002로 유의수준 0.05보다 작은 것을 알 수 있다. 따라서, 1차 계수는 유의하지 않고 2차계수는 유의하게 나타나는 것을 확인할 수 있다. ARMA(2,2)모형의 추정된 식은 다음과 같다.

$$x_t = w_t + 0.1137x_{t-1} + 0.8838x_{t-2} + 0.2590w_{t-1} - 0.4724w_{t-2},\ w_t \sim N(0,\sigma^{2_w})$$

마지막으로, ARMA(2,2)-GARCH(1,1)모형을 적합시켜 보았다. ARMA-GARCH 모형은 데이터를 ARMA 모형에 적합시킨 후, 적합된 ARMA의 잔차를 GARCH 모형을 통해 예측하게 된다. 최종적으로는 ARMA에 적합된 평균과 ARMA의 잔차를 예측한 GARCH모형을 이용하여 미래를 예측하는 것이 목적이다. GARCH 모형은 AR모형에 조건부 이분산성이 포함된 ARCH 모형의 일반화된 모형이다. GARCH모형은 변동성을 고려하는 모형이기 때문에 금융분야의 시계열 분석을 할 때 꽤 유용하게 사용되는 모형이다. GARCH(1,1)모형을 수식으로 표현하면 다음과 같다.

$$Y_t = \sigma_t\epsilon_t,\ \sigma_t^2 = \alpha_0 + \alpha_1 Y_{t-1}^2 + B_1\sigma_{t-1}^2,\ \epsilon_t \sim IID(0,1) \text{ independent of } \sigma_t$$

앞서 Walmart에 데이터를 적합시킨 ARMA(2,2) 모델의 잔차를 GARCH(1,1)에 적합시킨 결과는 다음과 같다.

Constant Mean - GARCH Model Results			
Dep. Variable	None	R-squared	0.000
Mean Model	Constant Mean	Adj. R-squared	0.000
Vol Model	GARCH	Log-Likelihood	-1017.04
Distribution	Normal	Method	Maximum Likelihood
AIC	2042.08	BIC	2053.02
Df Residuals	113	DF Model	1
==========			

Mean Model						
	coef	std err	z	P>|t|	[0.025	0.975]
Mu	192.2391	407.069	0.475	0.635	-6.04e+2	9.92e+2
Volatility Model						
	coef	std err	t	P>|t|	[0.025	0.975]
omega	6.344e+5	6.349e+5	0.985	0.324	-6.28e+5	1.9e+6
alpha[1]	0.4617	1.883	0.245	0.806	-3.228	4.152
beta[1]	0.4324	1.060	0.408	0.683	-1.645	2.510

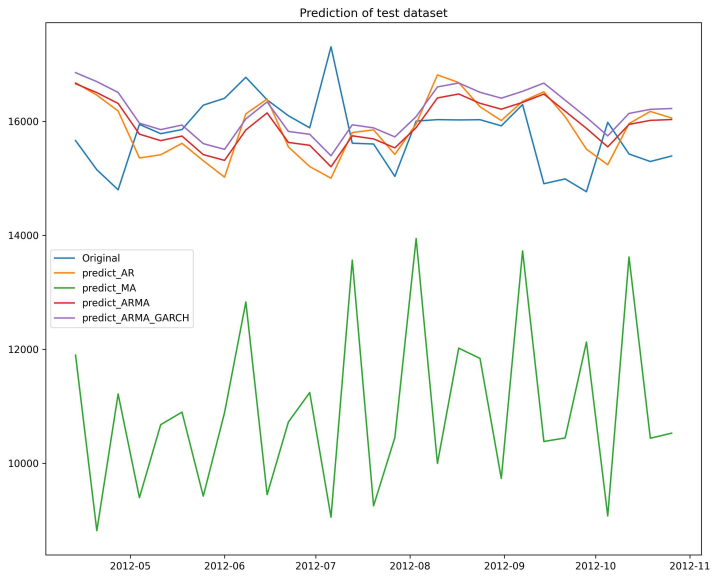
[표 5: Walmart의 ARMA(2,2)-GARCH(1,1) 결과]

모든 계수의 P-value가 유의수준 0.05보다 크기 때문에 유의하지 않은 것을 알 수 있다. ARMA(2,2)-GARCH(1,1)모형의 추정된 식은 다음과 같다.

$$Y_t = \sigma_t \epsilon_t,\ \sigma_t^2 = 6.344e5 + 0.4617\, Y_{t-1}^2 + 0.4324\sigma_{t-1}^2,\ \epsilon_t \sim IID(0,1) \text{ independent of } \sigma_t$$

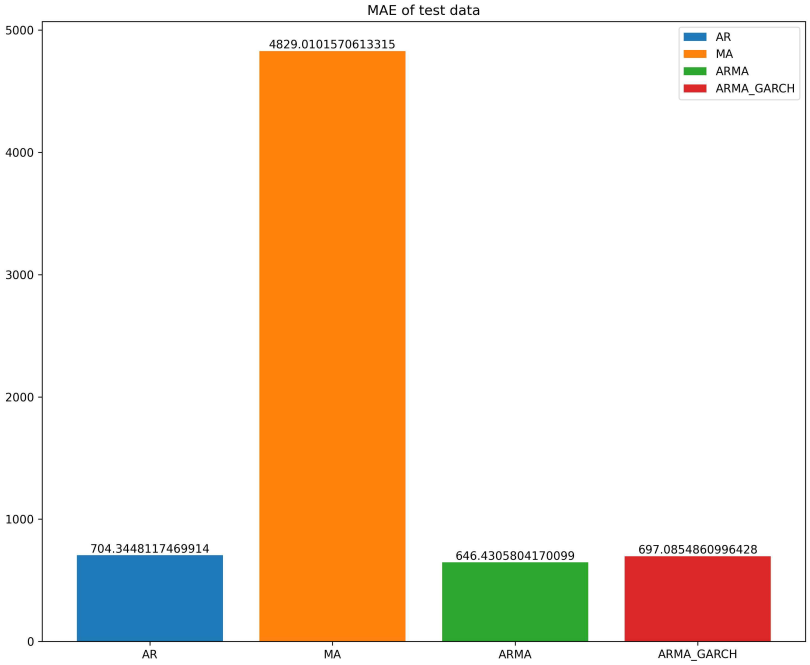
3. 결론

본론에서는 Walmart데이터를 AR(2), MA(2), ARMA(2,2), ARMA(2,2)-GARCH(1,1) 4가지 모형에 적합시킨 후 예측에 필요한 추정된 식을 얻었다. 추정된 식을 Test 데이터로 지정했던 2012년 4월 13일 ~ 2012년 10월 26일의 기간에 대해 판매량 예측을 진행한 결과는 다음과 같다.

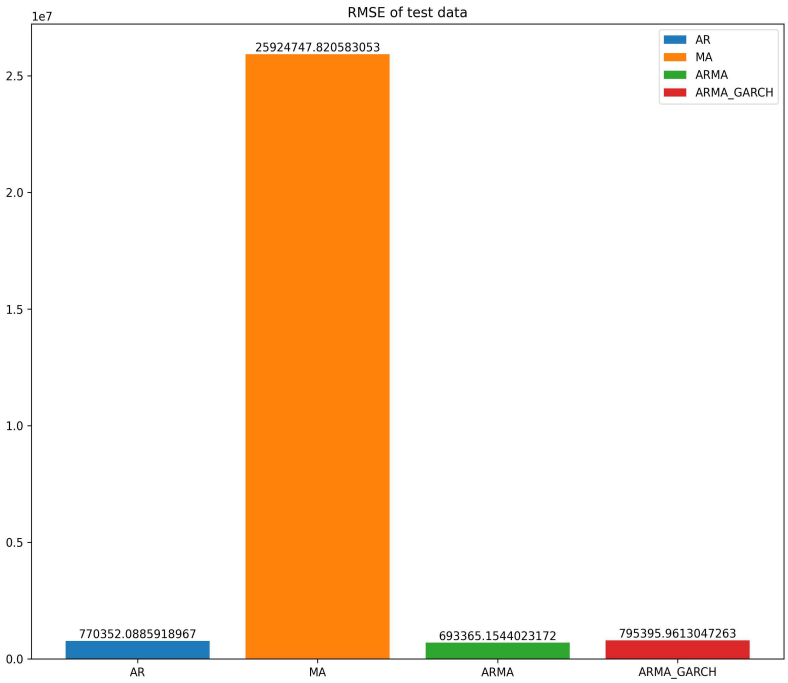


[그림 5: Test 데이터와 4개의 예측 모형 비교]

그림에서도 알 수 있듯이, 4개의 모형 중 MA(2) 모형은 기존의 Test데이터와 많이 떨어져있는 것을 볼 수 있다. 이를 통해, MA(2) 모형은 예측 성능이 많이 떨어져있는 것을 확인할 수 있다. 나머지 AR(2), ARMA(2,2), ARMA(2,2)-GARCH(1,1) 모형은 MA(2) 모형에 비해 Test데이터와 어느정도 가깝게 예측하는 것을 확인할 수 있다. 모형의 성능을 조금 더 자세히 비교하기 위해 MAE, RMSE 2개의 성능 지표를 비교해본 결과는 다음과 같다.



[그림 6: 4개 모형의 MAE 지표]



[그림 7: 4개 모형의 RMSE 지표]

4개 모형의 성능지표를 비교해본 결과, 앞선 모형비교 그래프에서 알 수 있듯이, MA(2) 모형의 MAE, RMSE가 월등히 높은 것을 확인할 수 있다. 3개의 모형중 ARMA(2,2) 모형이 MAE, RMSE가 제일 작아 가장 성능이 좋은 것을 확인할 수 있었으며, ARMA(2,2)-GARCH(1,1)모형의 MAE는 AR(2) 모형보다 작았지만, 오히려 RMSE는 ARMA(2,2)-GARCH(1,1)모형이 AR(2) 모형보다 더 높은 것을 보아 두 모형의 성능이 비슷한 것을 확인할 수 있었다.

본 보고서에서는 다양한 시계열 기법을 이용하여 마트의 판매량 예측을 진행해보고자 하였으며, 4개의 모형 중, ARMA(2,2) 모형이 가장 뛰어난 성능을 보인 것을 확인할 수 있었다. 그러나 가장 성능이 좋은 ARMA(2,2) 모형의 RMSE는 693,365로 상당히 높은 값을 가지는 것을 확인할 수 있었다. 성능이 가장 좋은 모형임에도 불구하고 RMSE가 높게 나온 가장 큰 이유중 하나로는 데이터의 크기가 작기 때문이라고 생각된다. 사람이 살아감에 있어 1,2년은 어느정도 긴 시간으로 생각될 수 있겠지만, 시계열 분석에서의 1,2년은 매우 작은 크기의 데이터이다. 또한, 본 보고서에서는 일간 데이터가 아닌 주간 데이터를 사용했기 때문에 이러한 영향을 더욱 크게 받은 것으로 보인다. 시계열 분석을 진행함에 있어 최소 5~10년 정도의 크기를 가지는 데이터를 사용하게 되면 성능향상에 도움이 될 것 이라고 생각한다.

4. 참고문헌

- ARMA-GARCH(https://drive.google.com/drive/folders/1b5VNeWKtfZG9wYpPkIBM2-4O-BXizmjX)
- Rfriend(https://rfriend.tistory.com/)

5. 코드

```
# 구글 드라이브 연결
from google.colab import drive
drive.mount('/content/drive')

# 필요패키지 로드
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
%matplotlib inline

from matplotlib.ticker import MultipleLocator, IndexLocator, FuncFormatter
from matplotlib.dates import MonthLocator, DateFormatter

from statsmodels.tsa.stattools import acf,pacf
import statsmodels.api as sm

from statsmodels.tsa.seasonal import seasonal_decompose

from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.arima_model import ARIMA

from sklearn import metrics
from arch import arch_model


# 데이터 전처리
train_df = pd.read_csv('/content/drive/MyDrive/대학원시계열_최종레포트/Time_Series/Final_Report/train.csv')
train_df = train_df.groupby('Date')[['Weekly_Sales']].mean()
train_df.insert(0,'Date',train_df.index)
train_df.index = range(0,len(train_df))
train_df['Date'] = pd.to_datetime(train_df['Date'])

# 훈련 평가 데이터셋 구분
ratio = int(len(train_df) *0.8)
test_df = train_df.iloc[ratio:,:]
test_df = test_df.reset_index(drop = True)
train_df = train_df.iloc[:ratio,:]

# 시계열 그래프
fig = plt.figure(figsize=(12,10))
ax = fig.add_subplot()
ax.plot(train_df['Date'],train_df['Weekly_Sales'])
ax.xaxis.set_major_locator(MonthLocator(interval=6))
ax.xaxis.set_major_formatter(DateFormatter('%Y/%m/%d'))
ax.xaxis.set_minor_locator(MonthLocator(interval=1))
plt.ylabel('Weekly_Sales')
plt.xlabel('Time')
plt.title('Weekly_Sales in Walmart')
plt.savefig('/content/drive/MyDrive/대학원시계열_최종레포트/Time_Series/Final_Report/Time-series.png',dpi=300)
plt.show()

# ACF
sm.graphics.tsa.plot_acf(np.array(train_df['Weekly_Sales']))
plt.savefig('/content/drive/MyDrive/대학원시계열_최종레포트/Time_Series/Final_Report/ACF.png',dpi=300)

#PACF
sm.graphics.tsa.plot_pacf(np.array(train_df['Weekly_Sales']))
plt.savefig('/content/drive/MyDrive/대학원시계열_최종레포트/Time_Series/Final_Report/PACF.png',dpi=300)

# 시계열 분해
seasonal_DF = train_df.set_index(train_df['Date'])
seasonal_result = seasonal_decompose(seasonal_DF['Weekly_Sales'],model='additive')
plt.rcParams['figure.figsize'] = [12, 10]
seasonal_result.plot()
```

```
plt.savefig('/content/drive/MyDrive/대학원시계열_최종레포트/Time_Series/Final_Report/Seasonal.png',dpi=300)
plt.show()

# ADF-TEST
adfuller(train_df['Weekly_Sales'], autolag="AIC")

#AR model
AR_model = ARIMA(train_df['Weekly_Sales'], order=(2,0,0))
AR_model_fit = AR_model.fit(trend='nc',full_output=True, disp=1)
print(AR_model_fit.summary())

# AR Model 예측
y_pred_AR = AR_model_fit.predict(1,len(test_df))

# AR Model 성능 지표
mae_AR = metrics.mean_absolute_error(test_df['Weekly_Sales'],y_pred_AR)
rmse_AR = metrics.mean_squared_error(test_df['Weekly_Sales'],y_pred_AR)
Error_AR = pd.DataFrame([[mae_AR,rmse_AR]],columns = ['MAE','RMSE'])

#MA Model
MA_model = ARIMA(train_df['Weekly_Sales'], order=(0,0,2))
MA_model_fit = MA_model.fit(trend='nc',full_output=True, disp=1)
print(MA_model_fit.summary())

#MA Model 예측
y_pred_MA = MA_model_fit.predict(1,len(test_df))

#MA Model 성능 지표
mae_MA = metrics.mean_absolute_error(test_df['Weekly_Sales'],y_pred_MA)
rmse_MA = metrics.mean_squared_error(test_df['Weekly_Sales'],y_pred_MA)
Error_MA = pd.DataFrame([[mae_MA,rmse_MA]],columns = ['MAE','RMSE'])

#ARMA model
from statsmodels.tsa.arima_model import ARIMA
ARMA_model = ARIMA(train_df['Weekly_Sales'], order=(2,0,2))
ARMA_model_fit = ARMA_model.fit(trend='nc',full_output=True, disp=1)
print(ARMA_model_fit.summary())

# ARMA Model 예측
y_pred_ARMA = ARMA_model_fit.predict(1,len(test_df))

# ARMA Model 성능지표
mae_ARMA = metrics.mean_absolute_error(test_df['Weekly_Sales'],y_pred_ARMA)
rmse_ARMA = metrics.mean_squared_error(test_df['Weekly_Sales'],y_pred_ARMA)
Error_ARMA = pd.DataFrame([[mae_ARMA,rmse_ARMA]],columns = ['MAE','RMSE'])

# ARMA-GARCH
arima_residuals = arima_model_fitted.resid
garch = arch_model(arima_residuals, p=1, q=1)
garch_fitted = garch.fit()
garch_fitted.summary()

#ARMA 잔차를 GARCH를 통해 예측
predicted_mu = arima_model_fitted.predict(1,len(test_df))
garch_forecast = garch_fitted.forecast(horizon=1)
predicted_et = garch_forecast.mean['h.1'].iloc[-1]
y_pred_ARMA_GARCH = predicted_mu + predicted_et

# ARMA-GARCH 성능지표
mae_ARMA_GARCH = metrics.mean_absolute_error(test_df['Weekly_Sales'],y_pred_ARMA_GARCH)
rmse_ARMA_GARCH = metrics.mean_squared_error(test_df['Weekly_Sales'],y_pred_ARMA_GARCH)
```

```
Error_ARMA_GARCH = pd.DataFrame([[mae_ARMA_GARCH,rmse_ARMA_GARCH]],columns = ['MAE','RMSE'])


#Test셋과 4개 모형 예측 비교
fig = plt.figure(figsize=(12,10))
ax = fig.add_subplot()
ax.plot(test_df['Date'],test_df['Weekly_Sales'], label = 'Original')
ax.plot(test_df['Date'],y_pred_AR, label = 'predict_AR')
ax.plot(test_df['Date'],y_pred_MA, label = 'predict_MA')
ax.plot(test_df['Date'],y_pred_ARMA, label = 'predict_ARMA')
ax.plot(test_df['Date'],y_pred_ARMA_GARCH, label = 'predict_ARMA_GARCH')
plt.title('Prediction of test dataset')
ax.legend()
plt.savefig('/content/drive/MyDrive/대학원시계열_최종레포트/Time_Series/Final_Report/Comparison.png',dpi=300)
plt.show()

# 성능지표 데이터 프레임 통일
Error_AR.index = ["AR"]
Error_MA.index = ["MA"]
Error_ARMA.index = ["ARMA"]
Error_ARMA_GARCH.index = ["ARMA_GARCH"]

Error_data = pd.DataFrame(columns=['MAE',"RMSE"])
Error_data = Error_data.append([Error_AR,Error_MA,Error_ARMA,Error_ARMA_GARCH])
# 모형 MAE 비교
fig = plt.figure(figsize=(12,10))
ax = fig.add_subplot()
ax.bar(Error_data.index[0],Error_data['MAE'][0],label=Error_data.index[0])
ax.bar(Error_data.index[1],Error_data['MAE'][1],label=Error_data.index[1])
ax.bar(Error_data.index[2],Error_data['MAE'][2],label=Error_data.index[2])
ax.bar(Error_data.index[3],Error_data['MAE'][3],label=Error_data.index[3])
for i, j in enumerate(Error_data.index):
    plt.text(Error_data.index[i], Error_data['MAE'][i],
             Error_data['MAE'][i],
             fontsize = 10,
             color='black',
             horizontalalignment='center',
             verticalalignment='bottom')
plt.title('MAE of test data')
ax.legend(loc='best')
plt.savefig('/content/drive/MyDrive/대학원시계열_최종레포트/Time_Series/Final_Report/MAE.png',dpi=300)
plt.show()

# 모형 RMSE 비교
fig = plt.figure(figsize=(12,10))
ax = fig.add_subplot()
ax.bar(Error_data.index[0],Error_data['RMSE'][0],label=Error_data.index[0])
ax.bar(Error_data.index[1],Error_data['RMSE'][1],label=Error_data.index[1])
ax.bar(Error_data.index[2],Error_data['RMSE'][2],label=Error_data.index[2])
ax.bar(Error_data.index[3],Error_data['RMSE'][3],label=Error_data.index[3])
for i, j in enumerate(Error_data.index):
    plt.text(Error_data.index[i], Error_data['RMSE'][i],
             Error_data['RMSE'][i],
             fontsize = 10,
             color='black',
             horizontalalignment='center',
             verticalalignment='bottom')
plt.title('RMSE of test data')
ax.legend(loc='best')
plt.savefig('/content/drive/MyDrive/대학원시계열_최종레포트/Time_Series/Final_Report/RMSE.png',dpi=300)
plt.show()
```