# Multivariate Modeling
# DATS 6450
# Instructor: Dr. Reza Jafari

# Term Project

**Changhao Ying**
**CY**
**04/15/2020**

# Abstract

This individual project uses time series dataset as analysis aim. The objective of this project is using the various model: Holt winter, Linear regression and ARMA to predict the dependent variable. Holt winter and ARMA models are both self-predictive which means I just need the dependent variable and can apply the model to it. In contrast, the regression model I used in this project includes two independent variables. I will compare these models in the end to choose the best one in this case.

# Introduction

In this project, I will use the data from Kaggle(https://www.kaggle.com/szrlee/stock-time-series-20050101-to-20171231#AAPL_2006-01-01_to_2018-01-01.csv). The dataset is about apple company's stock price change from 01/03/2006 to 12/29/2017. It contains 6 variables: high, open, low, close, volume and name. Open is the price of the stock at market open, high is highest price reached in the day, low is lowest price reached in the day, volume is the number of shares traded, name is apple. There is no missing value is this dataset.

I will use pycharm and python as tool to predict the highest price of the stock in the day by various methods. Firstly, I will change the dependent variable to stationary which means the mean and variance of the dataset is constant over time.

Then I will use Holt-Winter seasonal Method, Linear regression method, ARMA model to predict the price. In the linear regression method, I will pick the volume and open variables to be independent variable by feature selection.

In the end, I will compare the mean, variance, MSE, RMSE, Q value of the residuals to choose the best model to predict the highest price.

# Method and Theory

- Holt-Winter seasonal method comprises the forecast equation and three smoothing equations:
  1. Level $\ell_t$
  2. Trend $b_t$
  3. Seasonal $s_t$

$$
\begin{aligned}
\hat{y}_{t+h|t} &= \ell_t + hb_t + s_{t+h-m(k+1)} \\
\ell_t &= \alpha y_t + (1-\alpha)(\ell_{t-1} + b_{t-1}) \\
b_t &= \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1} \\
s_t &= \gamma(y_t - \ell_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m}
\end{aligned}
$$

where $k$ is the is the integer part of $\frac{h-1}{m}$, $m$ denotes the frequency of the seasonality. i.e. for quarterly data $m = 4$ and for monthly data $m = 12$ and $0 \leq \gamma \leq 1 - \alpha$

## Normal Equations for linear regression

- Let suppose the number of observations to be $T$. Then the multiple linear regression model can be written as system of linear equations:

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

- where :

$$\boldsymbol{Y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_T \end{pmatrix} \boldsymbol{X} = \begin{pmatrix} 1 & x_{1,1} & x_{2,1} & \cdots & x_{k,1} \\ 1 & x_{1,2} & x_{2,2} & \cdots & x_{k,2} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{1,T} & x_{2,T} & \cdots & x_{k,T} \end{pmatrix}$$

$$\boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_T \end{pmatrix} \boldsymbol{\epsilon} = \begin{pmatrix} \epsilon_0 \\ \epsilon_1 \\ \vdots \\ \epsilon_T \end{pmatrix}$$

$$\boxed{\hat{\beta} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{Y}}$$

This is a **least square estimator**.

It is also called **Normal equation**.

## Generalized partial autocorrelation function (GPAC)

- The PAC consider the case when $n_b = 0$. What about the case when $n_a \neq 0$ and $n_b \neq 0$?
- The generalized partial autocorrelation is used to estimated the order of ARMA model when $n_a \neq 0$ and $n_b \neq 0$.

$$\phi^{j}_{kk} = \frac{\begin{vmatrix} \hat{R}_y(j) & \hat{R}_y(j-1) & \dots & \hat{R}_y(j+1) \\ \hat{R}_y(j+1) & \hat{R}_y(j) & \dots & \hat{R}_y(j+2) \\ \vdots & \vdots & \vdots & \vdots \\ \hat{R}_y(j+k-1) & \hat{R}_y(j+k-2) & \dots & \hat{R}_y(j+k) \end{vmatrix}}{\begin{vmatrix} \hat{R}_y(j) & \hat{R}_y(j-1) & \dots & \hat{R}_y(j-k+1) \\ \hat{R}_y(j+1) & \hat{R}_y(j) & \dots & \hat{R}_y(j-k+2) \\ \vdots & \vdots & \vdots & \vdots \\ \hat{R}_y(j+k-1) & \hat{R}_y(j+k-2) & \dots & \hat{R}_y(j) \end{vmatrix}}$$

I will check the GPAC table of the dependent variable to determine the orders of the ARMA model. Autoregressive moving average (ARMA(na, nb)) models are the combination of AR(na) and MA(nb) models:

$$y(t) + a_1 y(t-1) + a_2 y(t-2) + \dots + a_{n_a} y(t-n_a) = \epsilon(t) + b_1\epsilon(t-1) + b_2\epsilon(t-2) + \dots + b_{n_b}\epsilon(t-n_b)$$

**Mean squared error**

$$MSE = mean(e_t^2)$$

**Root Mean squared error**

$$RMSE = \sqrt{mean(e_t^2)}$$

**Box-Pierce test**

- One such a test is "Box-Pierce test", based on the following statistics:

$$Q = T\sum_{k=1}^{h} r_k^2$$

- where h is the maximum lag being considered, T is the # of observation and $r_k$ is autocorrelation.

## Autocorrelation

- **Correlation** measures the extend of a linear relationship between two variables.
- **Auto-correlation** measures the linear relationship between lagged values of time series.
- The notation used for autocorrelation is $\tau_k$ which shoes the linear relationship between $y_t$ and $y_{t-k}$.
- $\tau_k$ for stationary processes is time invariant. It just depends on the lagged values of time series.

$$\boxed{\hat{\tau}_k = \frac{\sum_{t=k+1}^{T}(y_t - \overline{y})(y_{t-k} - \overline{y})}{\sum_{t=1}^{T}(y_t - \overline{y})^2}} \tag{1}$$

- where $y(t)$ is the observation at time $t$ and $\overline{y}$ is the sample mean of all observations.

Based on the Q value, MSE, mean, RMSE, variance of the models' residuals, I will pick the best model finally.

# Answer

## Preprocessing of the dataset:

Firstly, import the dataset into pycharm and take a look:

```
Out[4]:
         Date   Open   High    Low  Close     Volume  Name
0  2006-01-03  10.34  10.68  10.32  10.68  201853036  AAPL
1  2006-01-04  10.73  10.85  10.64  10.71  155225609  AAPL
2  2006-01-05  10.69  10.70  10.54  10.63  112396081  AAPL
3  2006-01-06  10.75  10.96  10.65  10.90  176139334  AAPL
4  2006-01-09  10.96  11.03  10.82  10.86  168861224  AAPL
```

Check if there is any missing values:

```
In[5]: def nan_checker(df):
  ...:     df_nan = pd.DataFrame([[var, df[var].isna().sum() / df.shape[0], df[var].dtype]
  ...:                            for var in df.columns if df[var].isna().sum() > 0],
  ...:                           columns=['var', 'proportion', 'dtype'])
  ...:     df_nan = df_nan.sort_values(by='proportion', ascending=False)
  ...:     return df_nan
  ...:df_nan = nan_checker(df)
  ...:df_nan.reset_index(drop=True)
  ...:#No missing value
  ...:
Out[5]:
Empty DataFrame
Columns: [var, proportion, dtype]
Index: []
```

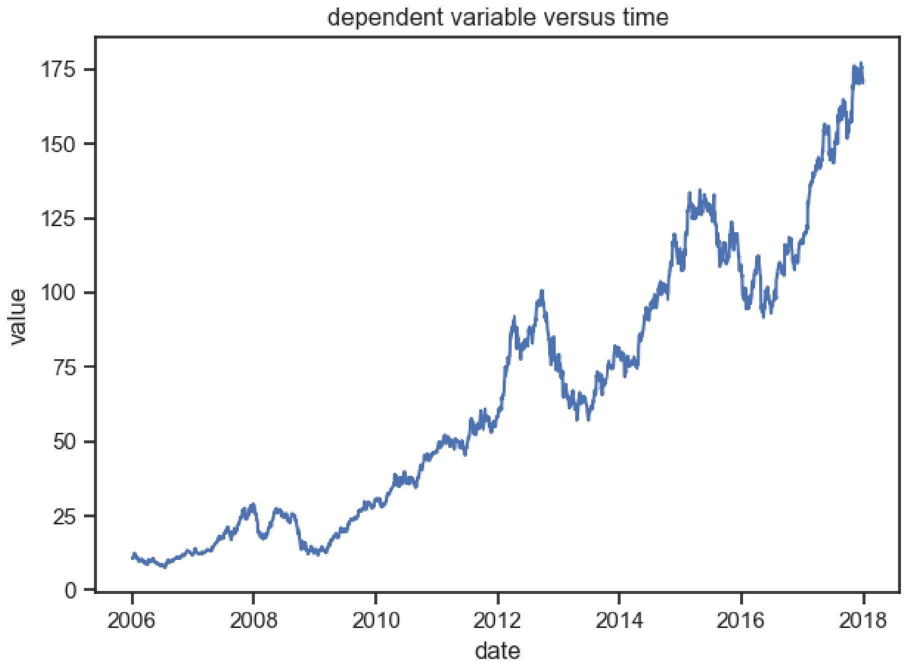
The result shows there is no NA values.

Change the index to be the date and delete the variable: name which is meaningless.

```
DatetimeIndex: 3019 entries, 2006-01-03 to 2017-12-29
Data columns (total 5 columns):
Open      3019 non-null float64
High      3019 non-null float64
Low       3019 non-null float64
Close     3019 non-null float64
Volume    3019 non-null int64
dtypes: float64(4), int64(1)
memory usage: 141.5 KB
```

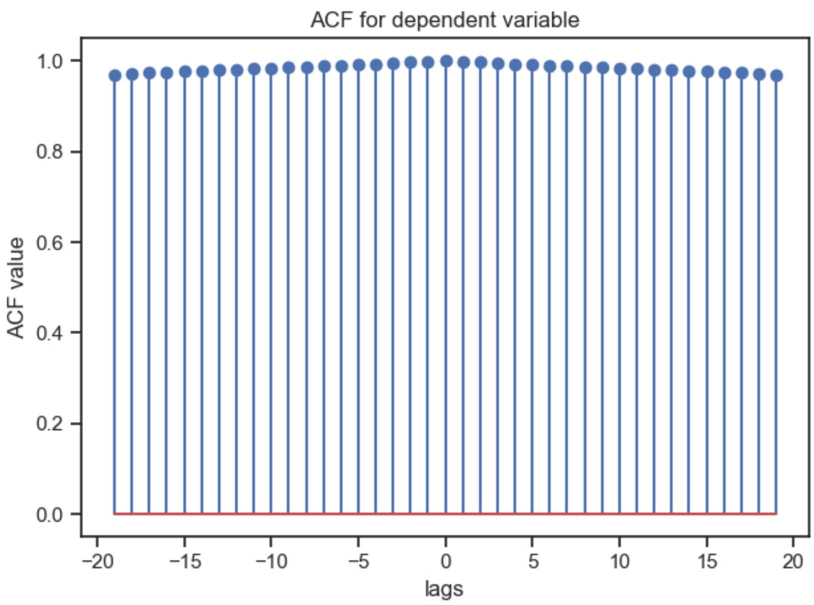
There are total 3019 rows of the dataset.

Plot the dependent variable over time:




We can see that the dependent variable is not stationary as its mean and variance changes over time.

ACF plot of the dependent variable:

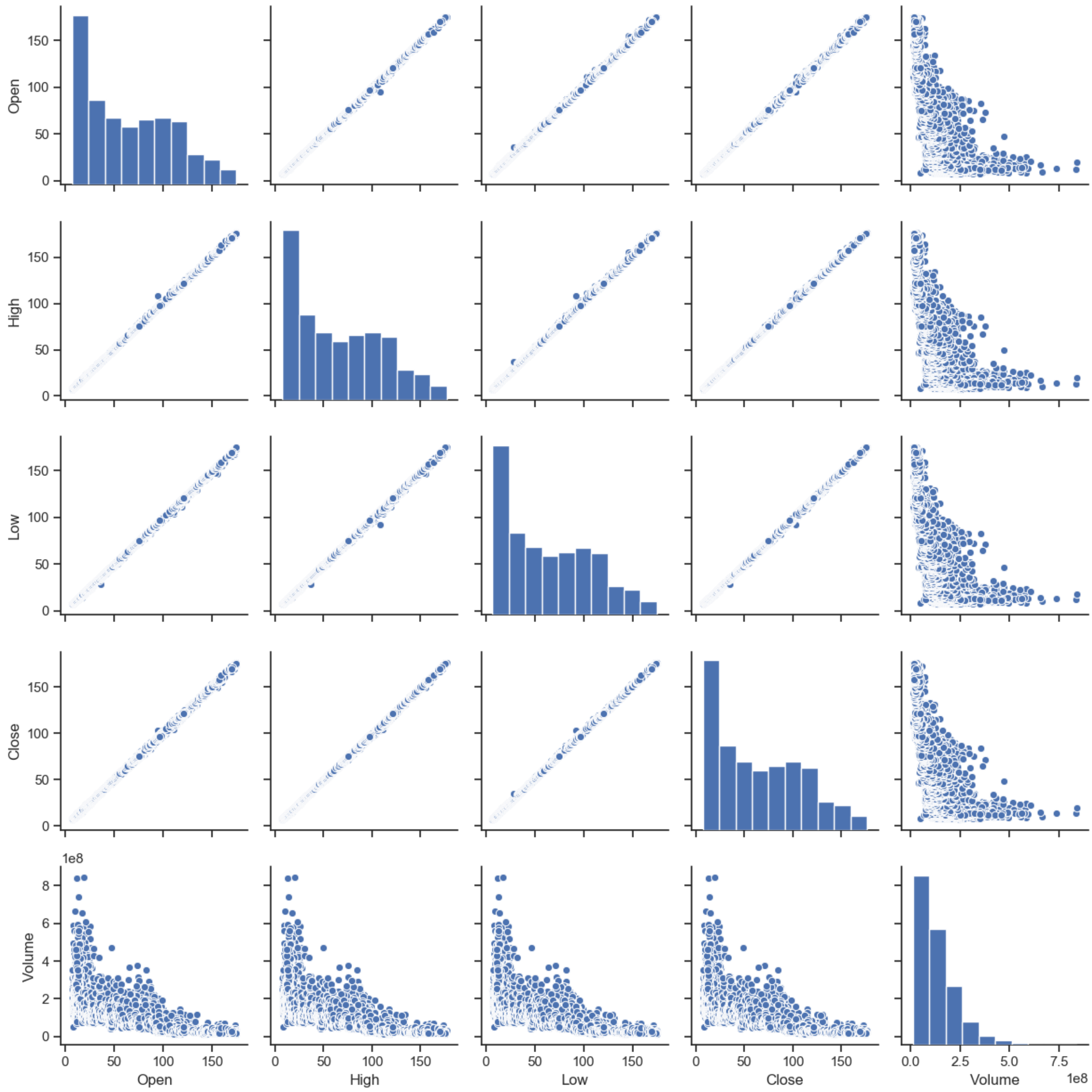


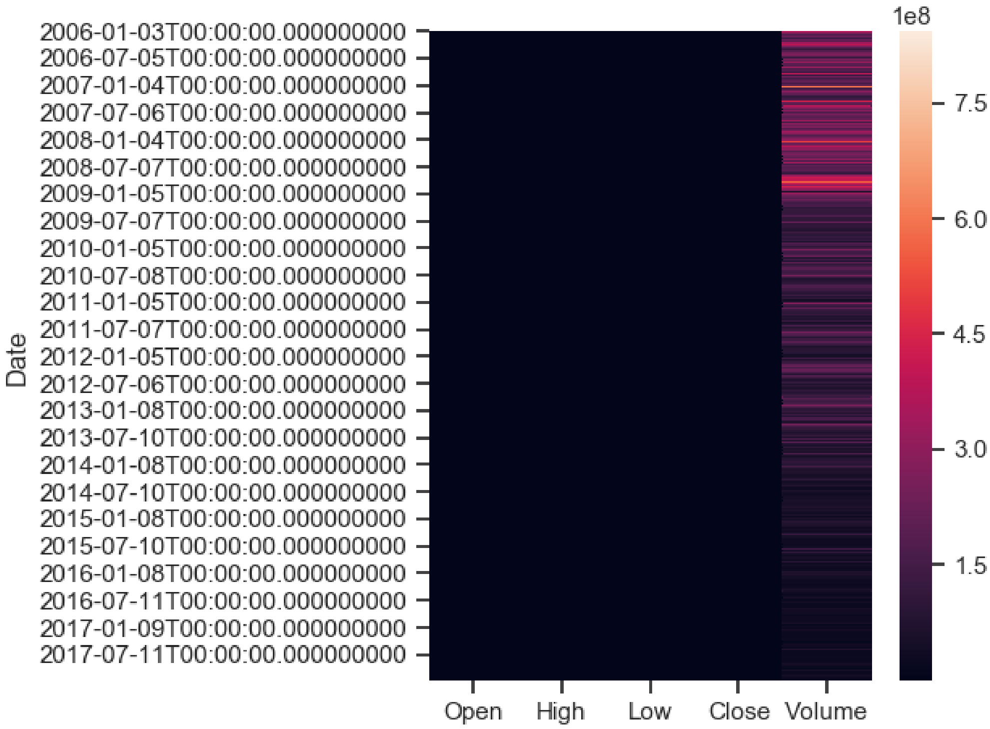
Apply ADF test to the dependent variable:

```
ADF Statistic: 0.679412
p-value: 0.989418
Critical Values:
    1%: -3.433
    5%: -2.863
    10%: -2.567
```

The p-value is 0.99 which is higher than 0.05 that means the dependent variable is not stationary.

Correlation Matrix of all variables:






From the correlation matrix of all variables, we can see that the open, high, low, close variables have almost linear correlation with each other. This means, if I choose over 2 variables of them as independent variables will cause multicollinearity. For this reason, I choose volume and open as independent variables.

Apply first difference method to the dependent variable:

- Differencing is performed by subtracting previous observation from the current observation.

$$\Delta y(t) = y(t) - y(t-1)$$

- This is called $1^{st}$ order difference.

```
ADF Statistic: -17.214917
p-value: 0.000000
Critical Values:
    1%: -3.433
    5%: -2.863
    10%: -2.567
```

The p-value of the ADF test after first difference is less than 0.05 so now it is stationary.

The highest price after first difference will be new dependent variable.

Because we can see that volume variable's quantity is much higher than other variables, so I divide this variable 1000000 as the unit of the volume will be million. Later, I will perform multiplicative decomposition and Holt winter method which require no negative and 0 values in the dependent variable so I delete these rows as well.

Out[16]:

| Date | first_high | Open | Volume |
|---|---|---|---|
| 2006-01-04 | 0.17 | 10.73 | 155.225609 |
| 2006-01-06 | 0.26 | 10.75 | 176.139334 |
| 2006-01-09 | 0.07 | 10.96 | 168.861224 |
| 2006-01-10 | 0.67 | 10.89 | 570.088246 |
| 2006-01-11 | 0.41 | 11.98 | 373.548882 |

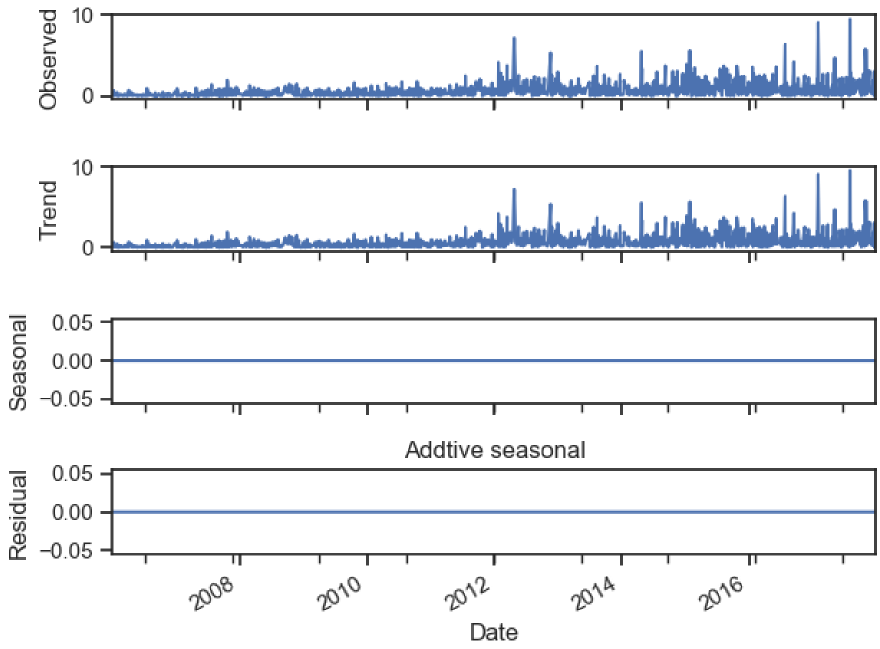
Now the new dataset has 1563 samples.

I divide the dataset into train(80%) and test set(20%). The x_test and x_train are the independent variables and y_test and y_train are the dependent variable.

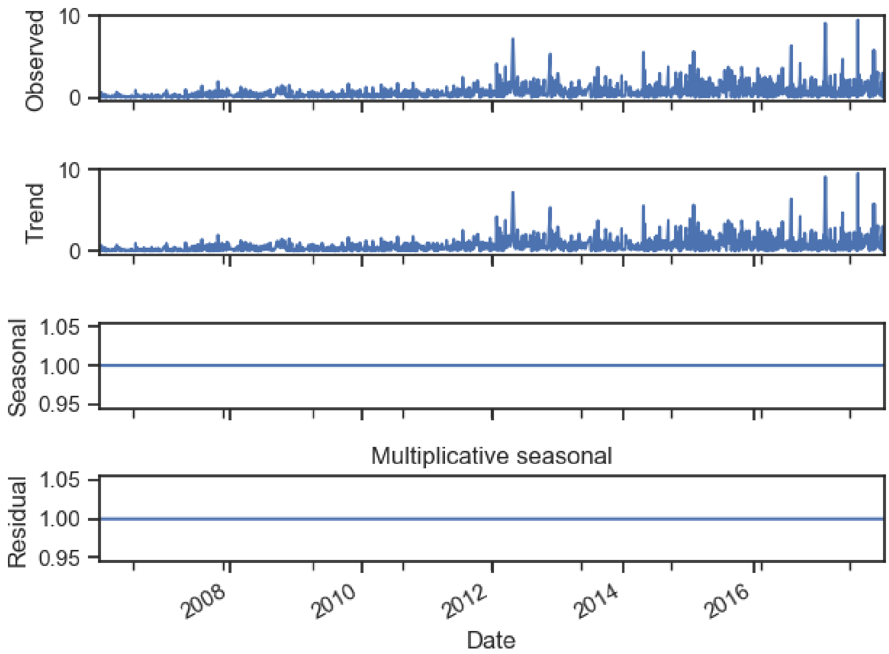
Up to now, the preprocess of the dataset is finished, and I can start model building.

# Time Series decomposition:

Additive method:




Multiplicative method:



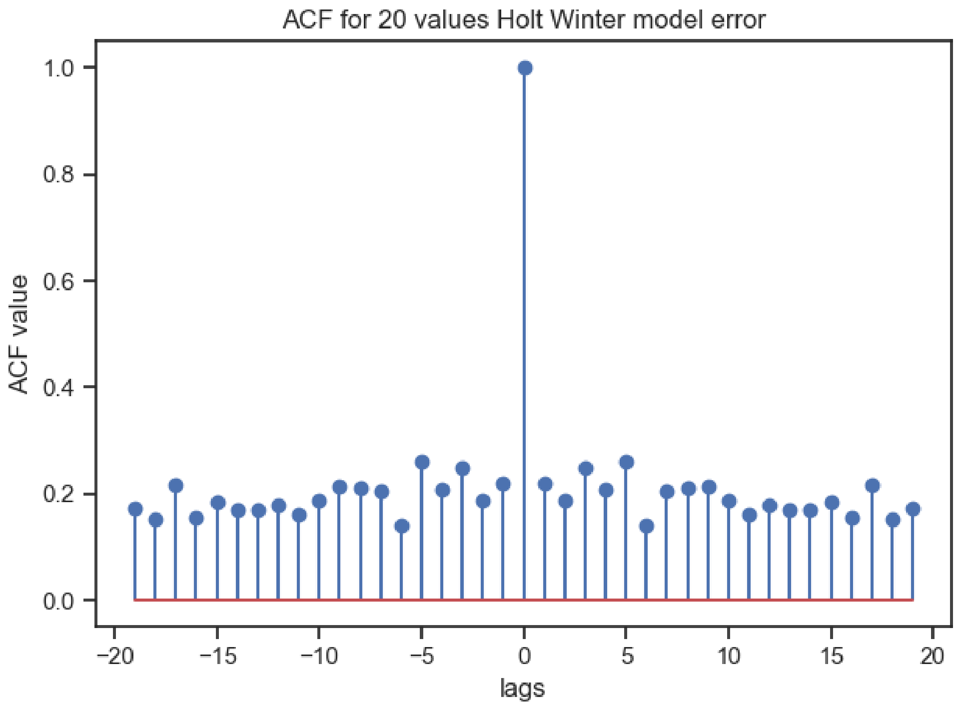

The residual of the multiplicative method is 1. So I choose additive method.

# Holt Winter Method:

Because the time comes over from 2006 to 2017, so I choose seasonal periods to be 4, the trend and season be additive method.

```
Out[21]:
            first_high  Holt_Winter
Date
2015-08-17        1.34     0.612581
2015-08-25        2.31     0.741204
2015-08-27        3.35     0.910489
2015-08-28        0.07     0.857961
2015-08-31        1.22     0.626184
```

Make a prediction using test set be holt winter method.




```
The Q value of Holt Winter model is: 224.60655583757844
The variance of Holt Winter model is: 1.9267451285309043
The mse of Holt Winter model is: 2.4870110987765988
The mean of Holt Winter model error is: -0.7485091651046729
RMSE of Holt Winter error is: 1.5770260298348278
```

# Linear regression:

As explained in the preprocessing part, I will use volume and open variables as independent variables.

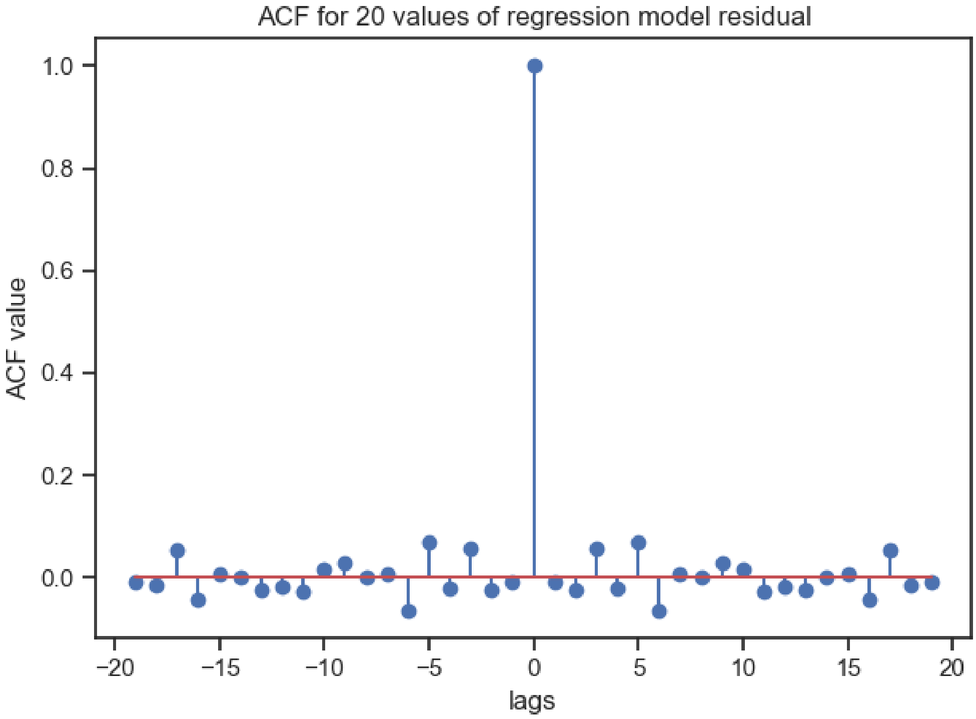
```
Dep. Variable:              first_high   R-squared:                       0.265
Model:                             OLS   Adj. R-squared:                  0.264
Method:                  Least Squares   F-statistic:                     224.5
Date:                 Sat, 18 Apr 2020   Prob (F-statistic):           5.46e-84
Time:                         23:45:14   Log-Likelihood:                -1079.3
No. Observations:                 1250   AIC:                             2165.
Df Residuals:                     1247   BIC:                             2180.
Df Model:                            2
Covariance Type:             nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
int           -0.4201      0.058     -7.280      0.000      -0.533      -0.307
Open           0.0125      0.001     21.112      0.000       0.011       0.014
Volume         0.0024      0.000     11.321      0.000       0.002       0.003
==============================================================================
Omnibus:                      883.674   Durbin-Watson:                   2.013
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            21921.694
Skew:                           2.946   Prob(JB):                         0.00
Kurtosis:                      22.651   Cond. No.                         651.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

The R squared value is about 0.265 and adjust R squared value is about 0.264. It means the model explains about 26% variation in the dependent variable. The p values of the 2 variable are about 0 which means they are all important in the t test.

The AIC value is about 2165 and BIC is about 2180. It tells me how well my model fits the dataset without overfitting it.

```
The Q value of regression is: 6.705997064870352
The variance of regression model error is: 1.4182817582340832
The mse of regression model error is: 1.4473389597765267
The mean of regression model error is: -0.17046173043367582
RMSE of regression mdel error is: 1.2030540136571286
```

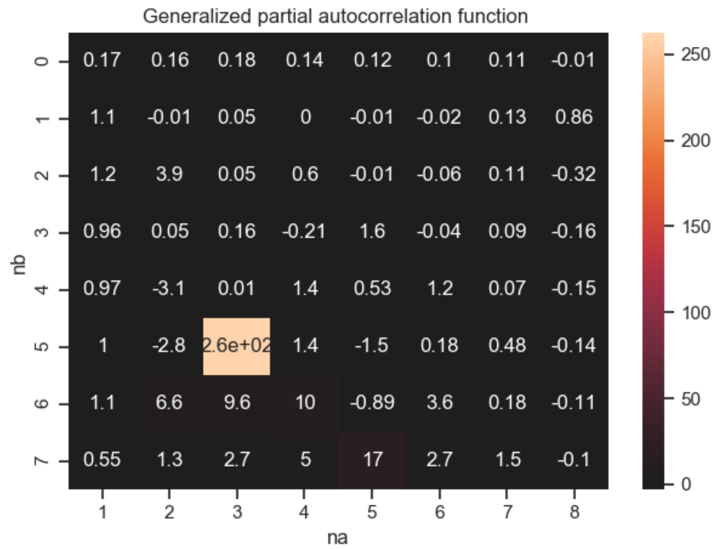



The residual of the regression model is not autocorrelated which is good.

# ARMA model:

Firstly, I need to use GPAC table to determine the order of ARMA model.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0.17 | 0.16 | 0.18 | 0.14 | 0.12 | 0.10 | 0.11 | -0.01 |
| 1 | 1.11 | -0.01 | 0.05 | 0.00 | -0.01 | -0.02 | 0.13 | 0.86 |
| 2 | 1.18 | 3.91 | 0.05 | 0.60 | -0.01 | -0.06 | 0.11 | -0.32 |
| 3 | 0.96 | 0.05 | 0.16 | -0.21 | 1.58 | -0.04 | 0.09 | -0.16 |
| 4 | 0.97 | -3.11 | 0.01 | 1.40 | 0.53 | 1.16 | 0.07 | -0.15 |
| 5 | 1.00 | -2.79 | 262.24 | 1.35 | -1.52 | 0.18 | 0.48 | -0.14 |
| 6 | 1.09 | 6.60 | 9.61 | 10.50 | -0.89 | 3.58 | 0.18 | -0.11 |
| 7 | 0.55 | 1.28 | 2.67 | 5.01 | 16.74 | 2.74 | 1.50 | -0.10 |




We can see that when na reaches 3, there is a high value in the GPAC table. So I will firstly take a try of **ARMA(1,1)** model to see if it passes the whiteness test and check for zero/pole cancellation. Then I will try **ARMA(2,1)** model to see if it passes the whiteness test and check for zero/pole cancellation. In the end, I will compare the results of the two models and pick the best one.

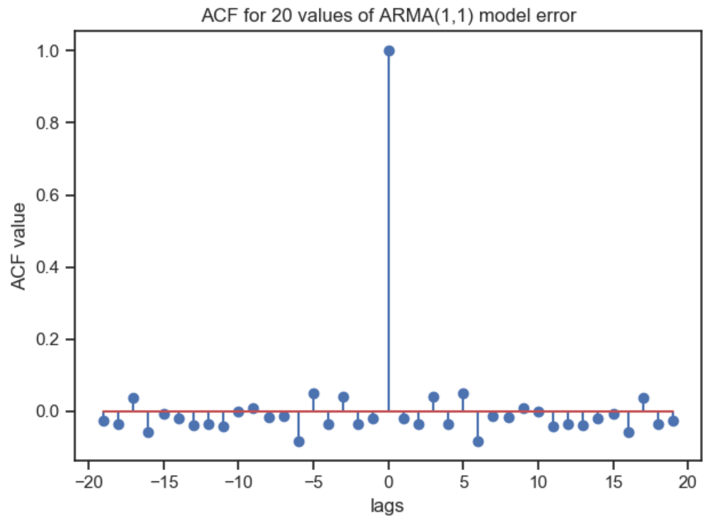
# ARMA(1,1)

```
                              ARMA Model Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 1250
Model:                     ARMA(1, 1)   Log Likelihood               -1155.160
Method:                       css-mle   S.D. of innovations              0.606
Date:                Sat, 18 Apr 2020   AIC                           2316.320
Time:                        23:55:15   BIC                           2331.712
Sample:                             0   HQIC                          2322.106

==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1.y        1.0000   9.66e-06   1.03e+05      0.000       1.000       1.000
ma.L1.y       -0.9722      0.006   -158.148      0.000      -0.984      -0.960
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1            1.0000           +0.0000j            1.0000            0.0000
MA.1            1.0286           +0.0000j            1.0286            0.0000
-----------------------------------------------------------------------------
```

```
The confidence interval is: [[ 0.99998106  1.00001894]
 [-0.9842359  -0.96013876]]
The covariance matrix is: [[ 9.33687257e-11 -5.33957657e-10]
 [-5.33957657e-10  3.77898260e-05]]
```

The standard deviation of the 2 parameters are both about 0.




The ACF plot of the residual shows that the residuals are not autocorrelated which means the model is good.

```
The Q value is: 8.499859943385617
In[40]: from scipy.stats import chi2
   ...:DOF=20-2
   ...:alfa=0.01
   ...:chi_critical=chi2.ppf(1-alfa,DOF)
   ...:if Q11<chi_critical:
   ...:    print("the residual is white")
   ...:else:
   ...:    print("Not white")
   ...:#pass
   ...:
the residual is white
```

The lags I used here is 20 and na=1, nb=1. The result passes the whiteness test.

**zero/pole cancellation:**

The parameter of AR model is 1 and MA model is about -1:

Y(t)=Y(t-1)+e(t)-e(t-1)

```
In[41]: #Pass zero/pole cancellation
   ...:np.roots([1,1])
Out[41]: array([-1.])
In[42]: #Pass zero/pole cancellation
   ...:np.roots([1,-1])
Out[42]: array([1.])
```

With np.roots, the two common roots are found, so I cancel them.

```
The Q value of ARMA(1,1) is: 8.499859943385617
The variance of ARMA(1,1) is: 1.4417277354732836
The mse of ARMA(1,1) is: 2.1177432566282977
The mean of ARMA(1,1) error is: 0.8222016304745535
RMSE of ARMA(1,1) error is: 1.4552468026518037
```

The mean and variance of ARMA(1,1) model is low(about 1) so this model is not biased.

# ARMA(2,1)

```
                              ARMA Model Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 1250
Model:                     ARMA(2, 1)   Log Likelihood               -1154.719
Method:                       css-mle   S.D. of innovations              0.606
Date:                Sun, 19 Apr 2020   AIC                           2317.438
Time:                        00:09:34   BIC                           2337.962
Sample:                             0   HQIC                          2325.154

==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1.y        0.9809   9.56e-06   1.03e+05      0.000       0.981       0.981
ar.L2.y        0.0191   9.75e-07   1.96e+04      0.000       0.019       0.019
ma.L1.y       -0.9712      0.006   -153.156      0.000      -0.984      -0.959
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1            1.0000           +0.0000j            1.0000            0.0000
AR.2          -52.4839           +0.0000j           52.4839            0.5000
MA.1            1.0296           +0.0000j            1.0296            0.0000
-----------------------------------------------------------------------------
```

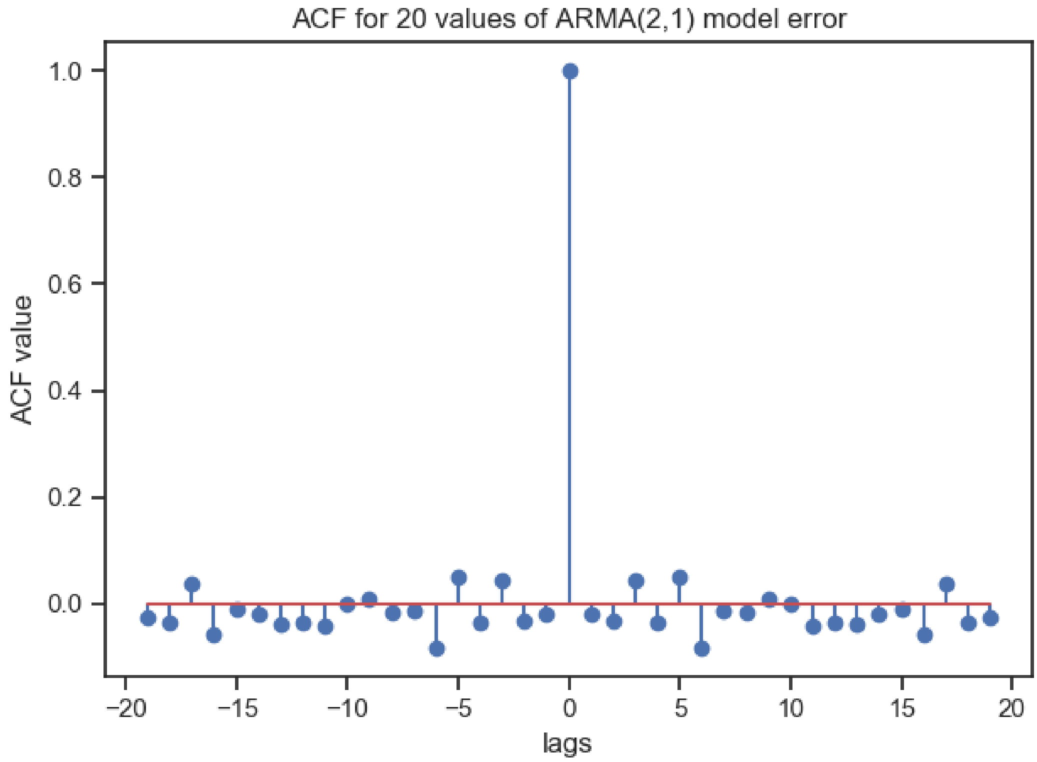
The model is: Y(t)=Y(t-1)+Y(t-2)+e(t)-e(t-1).

**zero/pole cancellation:**

```
In[45]: #Pass
   ...:np.roots([1,1,0])
Out[45]: array([-1.,  0.])
In[46]: #Pass
   ...:np.roots([1,-1,0])
Out[46]: array([1., 0.])
In[47]: #Pass
   ...:np.roots([1,0])
Out[47]: array([0.])
```

The three roots can be cancelled.

```
The confidence interval is: [[ 0.98092778  0.98096525]
 [ 0.01905157  0.01905539]
 [-0.98366435 -0.95880619]]
The covariance matrix is: [[ 9.13739855e-11 -1.20496169e-12 -5.08034583e-10]
 [-1.20496169e-12  9.49657848e-13  1.48660723e-12]
 [-5.08034583e-10  1.48660723e-12  4.02144340e-05]]
```




```
In[51]: acf.remove(acf[0])
   ...:acf1=np.array(acf)
   ...:Q21=len(error_21)*np.sum(acf1**2)
   ...:DOF=20-3
   ...:alfa=0.01
   ...:chi_critical=chi2.ppf(1-alfa,DOF)
   ...:if Q21<chi_critical:
   ...:    print("the residual is white")
   ...:else:
   ...:    print("Not white")
   ...:#Pass
   ...:
the residual is white
```

The lags here is still 20, na=2, nb=1, pass the whiteness test.

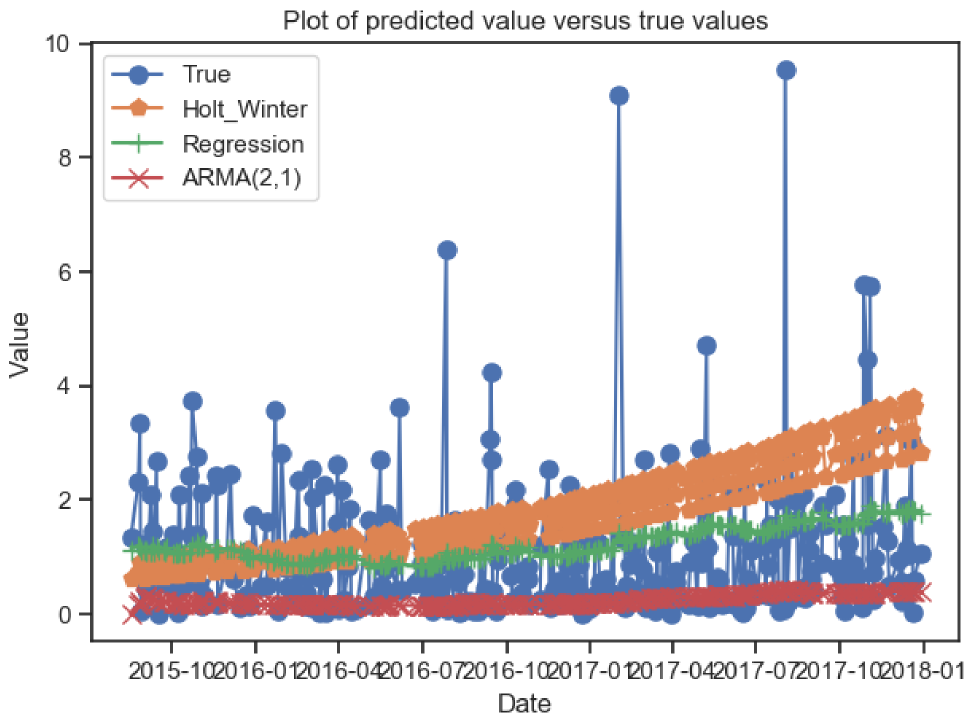
```
The Q value of ARMA(2,1) is: 8.514392530392273
The variance of ARMA(2,1) is: 1.442332811808308
The mse of ARMA(2,1) is: 2.1177775109469246
The mean of ARMA(2,1) error is: 0.8218544269751283
RMSE of ARMA(2,1) error is: 1.4552585718513822
```

The results of the 2 ARMA models are quite similar and both pass the whiteness test. I am inclined to ARMA(2,1) because it has more parameters.

# Final Model selection

Out[53]:

| | method | MSE | mean | variance | Q value | RMSE |
|---|---|---|---|---|---|---|
| 0 | Holt Winter | 2.487011 | -0.748509 | 1.926745 | 224.606556 | 1.577026 |
| 1 | Regression | 1.447339 | -0.170462 | 1.418282 | 6.705997 | 1.203054 |
| 2 | ARMA(2,1) | 2.117778 | 0.821854 | 1.442333 | 8.514393 | 1.455259 |




The q value of the holt winter model is high which means the residual is not autocorrelated. The mean, variance, RMSE, MSE and Q value of regression model and ARMA(2,1) model are quite similar.
From the plot of the predicted values versus the true value (test set), we can see that there are some outliers that the model does not predict. This does affect much I think because the value is not high(0-10) and this is acceptable.
The AMRA(2,1) model's prediction are all very low(almost 0) which is little deviated from the true values.
As a result, I will pick linear regression model finally.

# Summary and conclusion

In the end, I choose linear regression model as the best one. However, the R squared of this model is just about 0.265 which is not high. What's more, in the true world, we need much more variables to predict the stock price and this is a very complex process. In my linear regression model, the independent variables are just 2 which is not enough.

And in this project, I just try ARMA(2,1) and ARMA(1,1) models. Maybe with higher orders, ARMA model can perform better.

The good thing is that the mean, variance, MSE, RMSE, Q values of my models are all very low. This means the model is not biased and performs well.

# Appendix

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import statsmodels.api as sm
from scipy import signal
#%%
#Check the dataset
df=pd.read_csv('AAPL.csv')
df.head(5)
#%%
def nan_checker(df):
    df_nan = pd.DataFrame([[var, df[var].isna().sum() / df.shape[0],
df[var].dtype]
                           for var in df.columns if df[var].isna().sum() >
0],
                          columns=['var', 'proportion', 'dtype'])
    df_nan = df_nan.sort_values(by='proportion', ascending=False)
    return df_nan
df_nan = nan_checker(df)
df_nan.reset_index(drop=True)
#No missing value
#%%
df.Timestamp = pd.to_datetime(df.Date,format='%Y-%m-%d')
df.index = df.Timestamp
df.drop('Date',axis = 1, inplace = True)
df.drop('Name',axis = 1, inplace = True)
df.head(5)
#%%
df.info()
#%%
plt.plot(df['High'])
plt.xlabel('date')
plt.ylabel('value')
plt.title('dependent variable versus time')
plt.show()
#%%
def get_auto_corr(timeSeries,k):
    l = len(timeSeries)
    timeSeries1 = timeSeries[0:l-k]
    timeSeries2 = timeSeries[k:]
    timeSeries_mean = np.mean(timeSeries)
    timeSeries_var = np.array([i**2 for i in timeSeries-
timeSeries_mean]).sum()
```

```
    auto_corr = 0
    for i in range(l-k):
        temp = (timeSeries1[i]-timeSeries_mean)*(timeSeries2[i]-
timeSeries_mean)/timeSeries_var
        auto_corr = auto_corr + temp
    return auto_corr
#%%
dep=np.array(df['High'])
acf=[]
for i in range(20):
    acf.append(get_auto_corr(dep,i))
L1=np.arange(0,20,1)
L2=-L1[::-1]
x = np.concatenate((L2[0:-1], L1))
acf_reverse = acf[::-1]
ACF = np.concatenate ((acf_reverse[0:-1], acf))
plt.stem(x,ACF, use_line_collection=True, markerfmt = 'o')
plt.xlabel('lags')
plt.ylabel('ACF value')
plt.title('ACF for dependent variable')
plt.show()
#%%
from statsmodels.tsa.stattools import adfuller
stat =df['High'].values
result = adfuller(stat)
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
   print('\t%s: %.3f' % (key, value))
#%%
import seaborn as sns
sns.set(style="ticks")
sns.pairplot(df)
plt.show()
#%%
sns.heatmap(df)
plt.show()
#%%
df['first_high']=(df['High']-df['High'].shift(1)).dropna()
df=df.drop(df.index[0])
df.head()
#%%
stat =df['first_high'].values
```

```
result = adfuller(stat)
print('ADF Statistic: %f' % result[0])
print('p-value: %f' % result[1])
print('Critical Values:')
for key, value in result[4].items():
   print('\t%s: %.3f' % (key, value))
#%%
#get dependent and independent variable
df1=df[['first_high','Open','Volume']]
#df1['Volume']=df['Volume'].apply(lambda x: x/1000000)
#df1['Volume'].div(1000000)
df1.loc[:,'Volume']=df1.loc[:,'Volume'].div(1000000)
#df1['Volume'].round(2)
df2 = df1[df1['first_high'] >0]
df2.head()
#%%
df2.info()
#%%
from sklearn.model_selection import train_test_split
X=df2[['Open','Volume']]
Y=df2[['first_high']]
x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.2,shuffle=Fa
lse)
#%%
from statsmodels.tsa.seasonal import seasonal_decompose
result = seasonal_decompose(df2['first_high'], model='additive', freq=1)
result.plot()
plt.title('Addtive seasonal')
plt.show()
#%%
result1 = seasonal_decompose(df2['first_high'],
model='multiplicative',freq=1)
result1.plot()
plt.title('Multiplicative seasonal')
plt.show()
#%%
#Holt winter prediction
from statsmodels.tsa.api import ExponentialSmoothing
fit1 =ExponentialSmoothing(np.asarray(y_train['first_high']),
seasonal_periods=4, trend='add', seasonal='add').fit(use_boxcox=True)
y_test['Holt_Winter'] = fit1.forecast(len(y_test))
y_test.head()

#%%
```

```python
ttt=np.array(y_test['first_high'])
error_winter=ttt-y_test['Holt_Winter'].values
acf=[]
for i in range(20):
    acf.append(get_auto_corr(error_winter,i))
L1=np.arange(0,20,1)
L2=-L1[::-1]
x = np.concatenate((L2[0:-1], L1))
acf_reverse = acf[::-1]
ACF = np.concatenate ((acf_reverse[0:-1], acf))
plt.stem(x,ACF, use_line_collection=True, markerfmt = 'o')
plt.xlabel('lags')
plt.ylabel('ACF value')
plt.title('ACF for 20 values Holt Winter model error')
plt.show()
#%%
acf.remove(acf[0])
acf1=np.array(acf)
Q_winter=len(error_winter)*np.sum(acf1**2)
var_winter=np.var(error_winter)
mse_winter=np.mean(error_winter**2)
mean_winter=np.mean(error_winter)
rmse_winter=(mse_winter)**0.5
print("The Q value of Holt Winter model is:",Q_winter)
print("The variance of Holt Winter model is:",var_winter)
print("The mse of Holt Winter model is:",mse_winter)
print("The mean of Holt Winter model error is:",mean_winter)
print("RMSE of Holt Winter error is:",rmse_winter)
#%%
#Regression model
x_train.insert(0,"int",1)
x_test.insert(0,"int",1)
#%%
model=sm.OLS(y_train,x_train).fit()
print(model.summary())
#%%
y_test['Regression'] = model.predict(x_test)
error_reg=y_test['first_high'].values-y_test['Regression'].values

#%%
acf=[]
for i in range(20):
    acf.append(get_auto_corr(error_reg,i))
L1=np.arange(0,20,1)
```

```python
L2=-L1[::-1]
x = np.concatenate((L2[0:-1], L1))
acf_reverse = acf[::-1]
ACF = np.concatenate ((acf_reverse[0:-1], acf))
plt.stem(x,ACF, use_line_collection=True, markerfmt = 'o')
plt.xlabel('lags')
plt.ylabel('ACF value')
plt.title('ACF for 20 values of regression model residual')
plt.show()

#%%
acf.remove(acf[0])
acf1=np.array(acf)
Q_reg=len(error_reg)*np.sum(acf1**2)
var_reg=np.var(error_reg)
mse_reg=np.mean(error_reg**2)
mean_reg=np.mean(error_reg)
rmse_reg=(mse_reg)**0.5
print("The Q value of regression is:",Q_reg)
print("The variance of regression model error is:",np.var(error_reg))
print("The mse of regression model error is:",np.mean(error_reg**2))
print("The mean of regression model error is:",mean_reg)
print("RMSE of regression mdel error is:",rmse_reg)
#finish regression model

#%%
#ARMA model
#determin parameters
y=np.array(y_train['first_high'])
acf=[]
for i in range(100):
    acf.append(get_auto_corr(y,i+1))
ry=[np.var(y)]
for i in range(99):
    ry.append(acf[i+1]*np.var(y))
#%%
phi=[]
phi_1=[]
i=0
gpac = np.zeros(shape=(8, 7))
for j in range(0,8):
    for k in range(2,9):
        bottom = np.zeros(shape=(k, k))
        top = np.zeros(shape=(k, k))
```

```
        for m in range(k):
            for n in range(k):
                bottom[m][n]=ry[abs(j+m - n)]
            top[m][-1]=ry[abs(j+m+1)]
        i=i+1
        top[:,:k-1] = bottom[:,:k-1]
        phi.append(round((np.linalg.det(top) / np.linalg.det(bottom)),2))
    phi_1.append(round(ry[j + 1] / ry[j],2))
gpac=np.array(phi).reshape(8,7)
Phi1=pd.DataFrame(phi_1)
Gpac=pd.DataFrame(gpac)
GPAC = pd.concat([Phi1,Gpac], axis=1)
GPAC.columns=['1','2','3','4','5','6','7','8']
print(GPAC)
#%%
sns.heatmap(GPAC, center=0, annot=True)
plt.title("Generalized partial autocorrelation function ")
plt.xlabel("na")
plt.ylabel("nb")
plt.show()
#%%
#na=1,nb=1
model1=sm.tsa.ARMA(y,(1,1)).fit(trend='nc',disp=0)
print(model1.summary())
#%%
print("The confidence interval is:",model1.conf_int(alpha=0.05, cols=None))
print("The covariance matrix is:",model1.cov_params())
#%%
result = model1.predict(start=0,end=312)
true=np.array(y_test['first_high'])
error_11=true-result
y_test['ARMA11']=result
y_test.head()
#%%
acf=[]
for i in range(20):
    acf.append(get_auto_corr(error_11,i))
L1=np.arange(0,20,1)
L2=-L1[::-1]
x = np.concatenate((L2[0:-1], L1))
acf_reverse = acf[::-1]
ACF = np.concatenate ((acf_reverse[0:-1], acf))
plt.stem(x,ACF, use_line_collection=True, markerfmt = 'o')
plt.xlabel('lags')
```

```
plt.ylabel('ACF value')
plt.title('ACF for 20 values of ARMA(1,1) model error')
plt.show()
#%%
acf.remove(acf[0])
acf1=np.array(acf)
Q11=len(error_11)*np.sum(acf1**2)
print("The Q value is:",Q11)
#%%
from scipy.stats import chi2
DOF=20-2
alfa=0.01
chi_critical=chi2.ppf(1-alfa,DOF)
if Q11<chi_critical:
    print("the residual is white")
else:
    print("Not white")
#pass
#%%
#Pass zero/pole cancellation
np.roots([1,-1])
#%%
var_11=np.var(error_11)
mse_11=np.mean(error_11**2)
mean_11=np.mean(error_11)
rmse_11=(mse_11)**0.5
print("The Q value of ARMA(1,1) is:",Q11)
print("The variance of ARMA(1,1) is:",var_11)
print("The mse of ARMA(1,1) is:",mse_11)
print("The mean of ARMA(1,1) error is:",mean_11)
print("RMSE of ARMA(1,1) error is:",rmse_11)

#%%
#na=2,nb=1
model2=sm.tsa.ARMA(y,(2,1)).fit(trend='nc',disp=0)
print(model2.summary())
#%%
#Pass
np.roots([1,0])
#%%
print("The confidence interval is:",model2.conf_int(alpha=0.05, cols=None))
print("The covariance matrix is:",model2.cov_params())
#%%
result2 = model2.predict(start=0,end=312)
```

```python
#result2[0]=y_test['first_high'][0]
true2=np.array(y_test['first_high'])
error_21=true2-result2
y_test['ARMA21']=result2
y_test.head()
#%%
acf=[]
for i in range(20):
    acf.append(get_auto_corr(error_21,i))
L1=np.arange(0,20,1)
L2=-L1[::-1]
x = np.concatenate((L2[0:-1], L1))
acf_reverse = acf[::-1]
ACF = np.concatenate ((acf_reverse[0:-1], acf))
plt.stem(x,ACF, use_line_collection=True, markerfmt = 'o')
plt.xlabel('lags')
plt.ylabel('ACF value')
plt.title('ACF for 20 values of ARMA(2,1) model error')
plt.show()
#%%
acf.remove(acf[0])
acf1=np.array(acf)
Q21=len(error_21)*np.sum(acf1**2)
DOF=20-3
alfa=0.01
chi_critical=chi2.ppf(1-alfa,DOF)
if Q21<chi_critical:
    print("the residual is white")
else:
    print("Not white")
#Pass
#%%
var_21=np.var(error_21)
mse_21=np.mean(error_21**2)
mean_21=np.mean(error_21)
rmse_21=(mse_21)**0.5
print("The Q value of ARMA(2,1) is:",Q21)
print("The variance of ARMA(2,1) is:",var_21)
print("The mse of ARMA(2,1) is:",mse_21)
print("The mean of ARMA(2,1) error is:",mean_21)
print("RMSE of ARMA(2,1) error is:",rmse_21)

#%%
#Pick ARMA(2,1) finally
```

```
#Left holt winter, regression, ARMA(2,1) models finally
pd.set_option('display.width', 400)
pd.set_option('display.max_columns', 10)
data={'method':['Holt Winter','Regression','ARMA(2,1)'],
      'MSE':[mse_winter,mse_reg,mse_21],
      'mean':[mean_winter,mean_reg,mean_21],
      'variance':[var_winter,var_reg,var_21],
      'Q value':[Q_winter,Q_reg,Q21],
      'RMSE':[rmse_winter,rmse_reg,rmse_21]}
table=pd.DataFrame(data)
table
#%%
#plt.figure(figsize=(10,8))
plt.plot(y_test['first_high'], label='True',marker='o',markersize=8)
plt.plot(y_test['Holt_Winter'],
label='Holt_Winter',marker='p',markersize=8)
plt.plot(y_test['Regression'],label='Regression',marker='+',markersize=8)
plt.plot(y_test['ARMA21'],label='ARMA(2,1)',marker='x',markersize=8)
plt.xlabel('Date')
plt.ylabel('Value')
plt.title('Plot of predicted value versus true values')
plt.legend(loc='best')
plt.show()
```