**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**

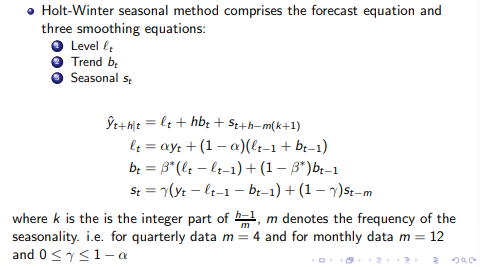
In this project, I will use the data from Kaggle(<https://www.kaggle.com/szrlee/stock-time-series-20050101-to-20171231#AAPL_2006-01-01_to_2018-01-01.csv>). The dataset is about apple company’s stock price change from 01/03/2006 to 12/29/2017. It contains 6 variables: high, open, low, close, volume and 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.

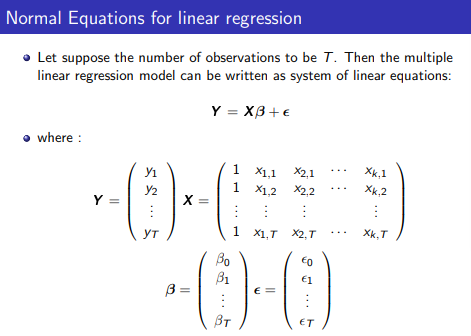
I will use pycharm and python as tool to predict the highest price of the stock in the day by various methods. Firstly, I will change the dependent variable to stationary which means the mean and variance of the dataset is constant over time.

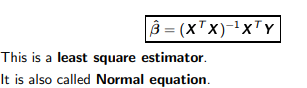
Then I will use Holt-Winter seasonal Method, Linear regression method, ARMA model to predict the price. In the linear regression method, I will pick the volume and open variables to be independent variable by feature selection.

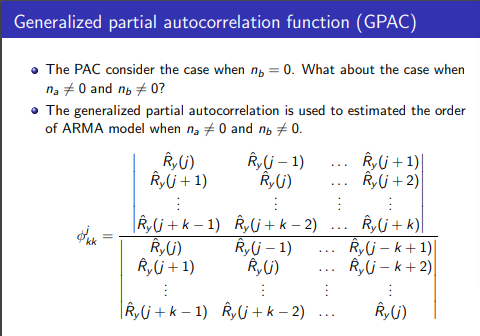
In the end, I will compare the mean, variance, MSE, RMSE, Q value of the residuals to choose the best model to predict the highest price.

**Method and Theory**

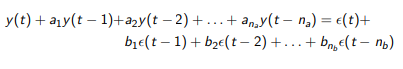


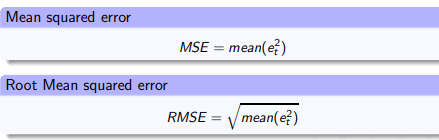


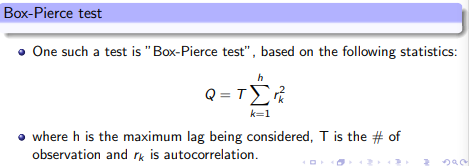


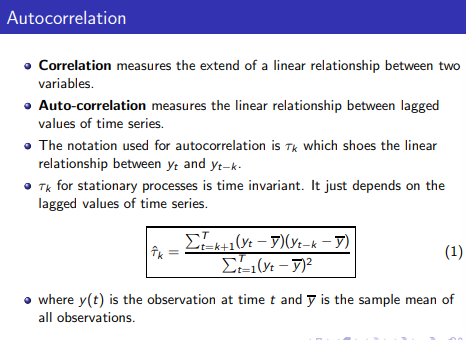


I will check the GPAC table of the dependent variable to determine the orders of the ARMA model. Autoregressive moving average (ARMA(na, nb)) models are the combination of AR(na) and MA(nb) models:







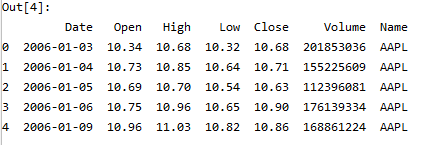


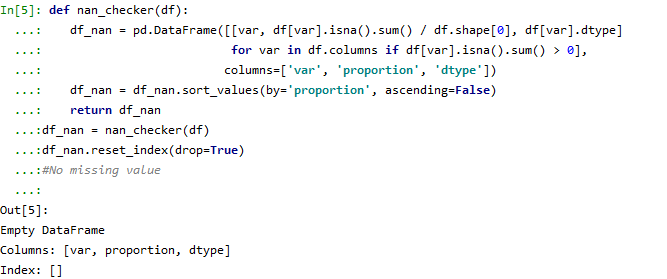
Based on the Q value, MSE, mean, RMSE, variance of the models’ residuals, I will pick the best model finally.

**Answer**

**Preprocessing of the dataset:**

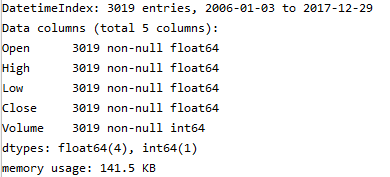
Firstly, import the dataset into pycharm and take a look:



Check if there is any missing values:

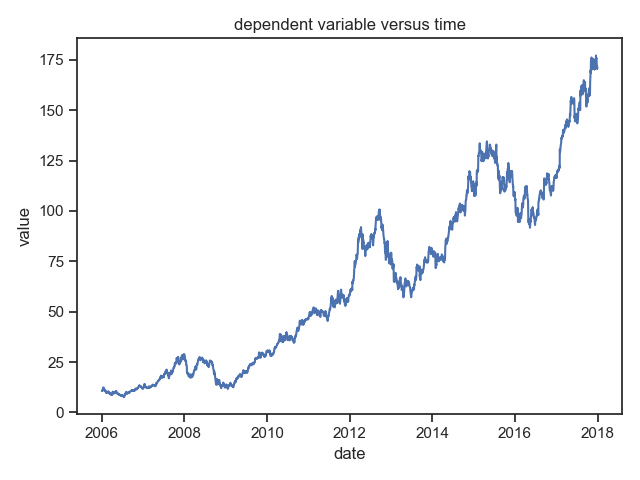
The result shows there is no NA values.

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

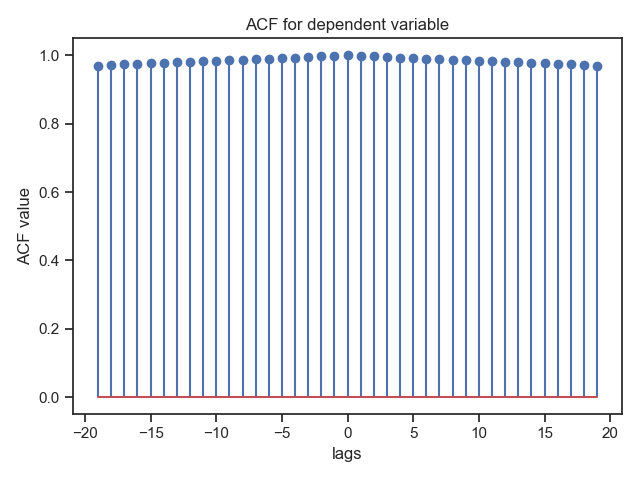


There are total 3019 rows of the dataset.

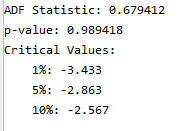
Plot the dependent variable over time:



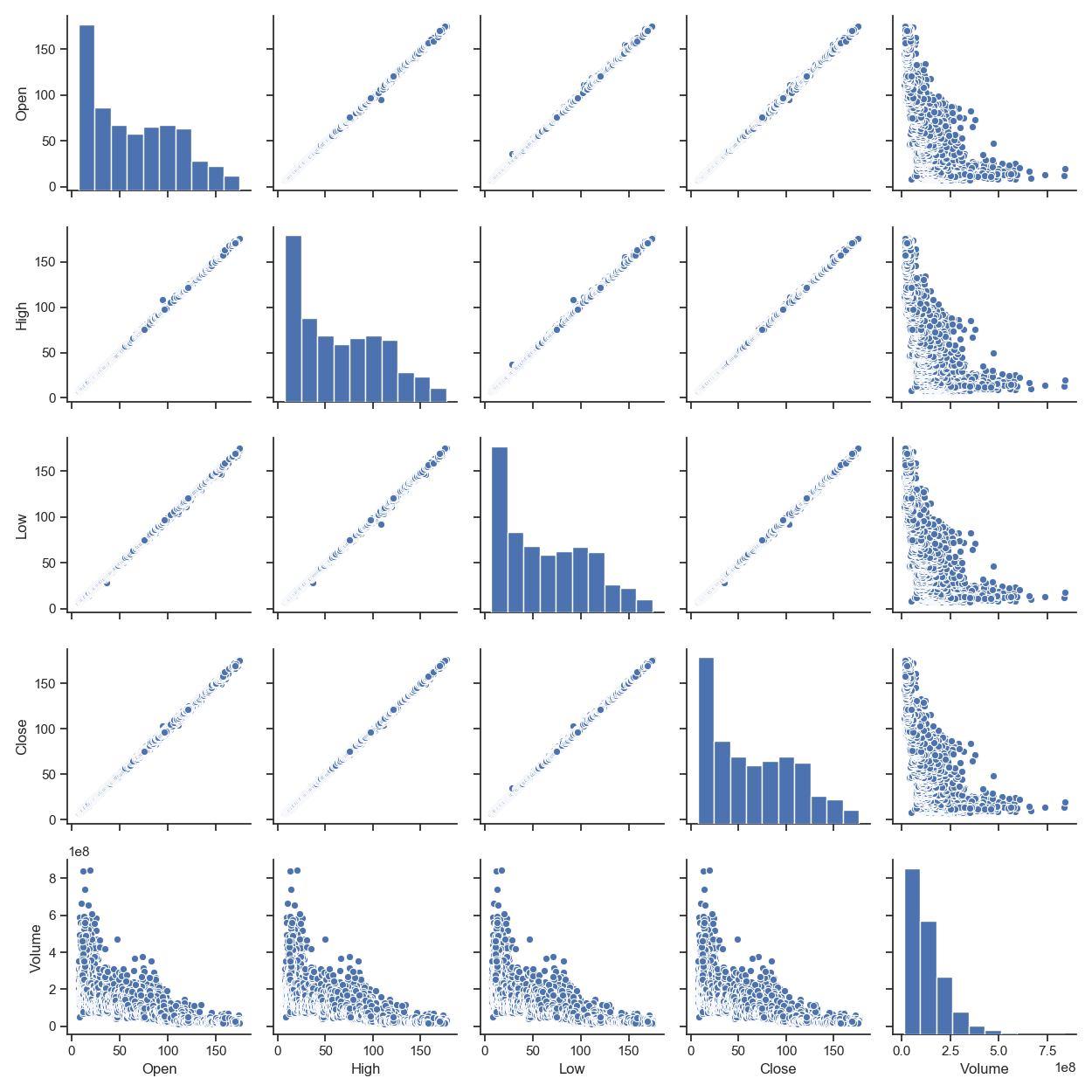
We can see that the dependent variable is not stationary as its mean and variance changes over time.

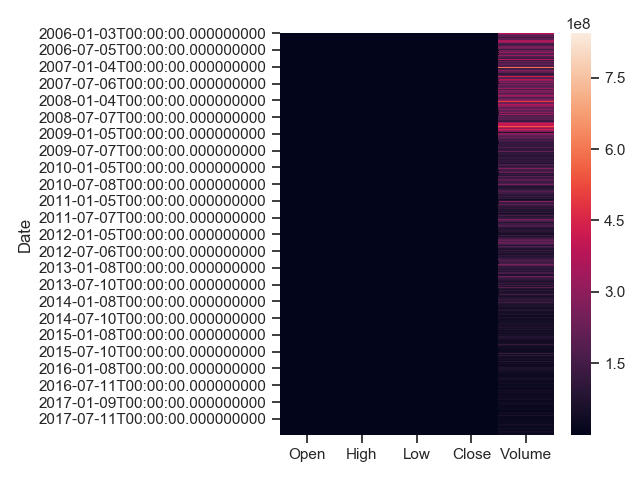
ACF plot of the dependent variable:

Apply ADF test to the dependent variable:

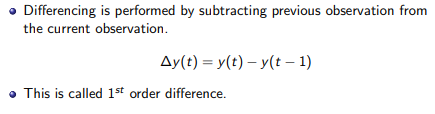
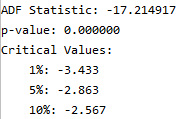


The p-value is 0.99 which is higher than 0.05 that means the dependent variable is not stationary.

Correlation Matrix of all variables:



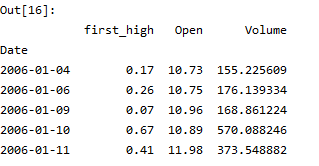
From the correlation matrix of all variables, we can see that the open, high, low, close variables have almost linear correlation with each other. This means, if I choose over 2 variables of them as independent variables will cause multicollinearity. For this reason, I choose volume and open as independent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

Apply first difference method to the dependent variable:

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.



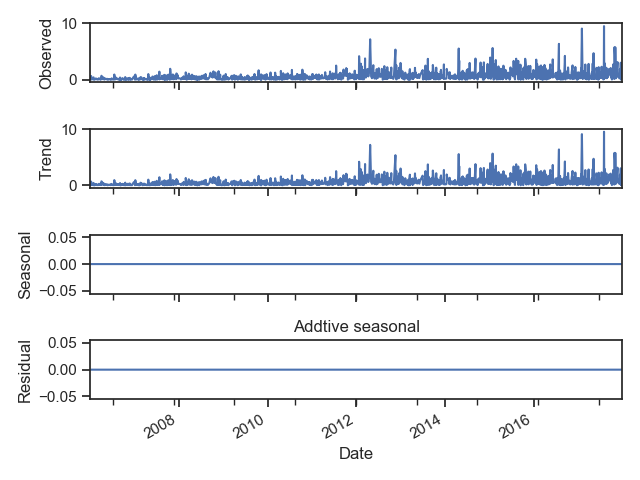
Now the new dataset has 1563 samples.

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.

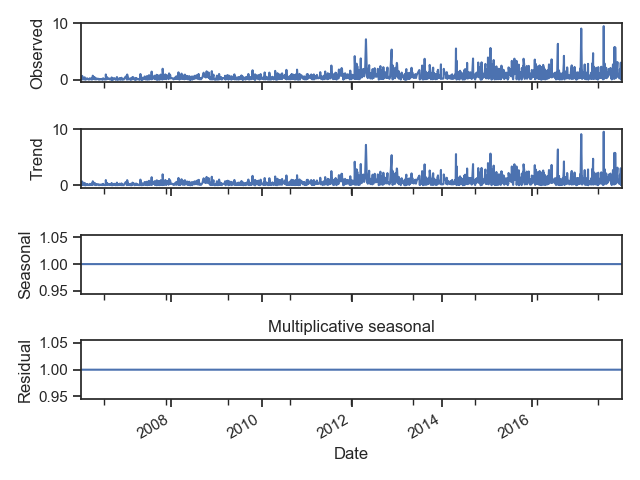
Up to now, the preprocess of the dataset is finished, and I can start model building.

**Time Series decomposition:**

Additive method:



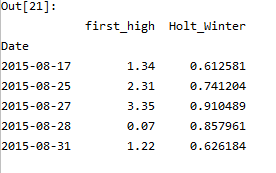
Multiplicative method:



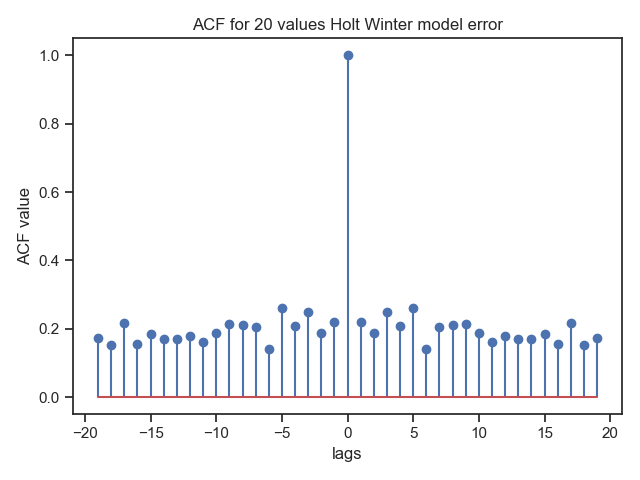
The residual of the multiplicative method is 1. So I choose additive method.

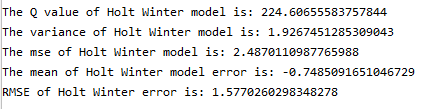
**Holt Winter Method:**

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

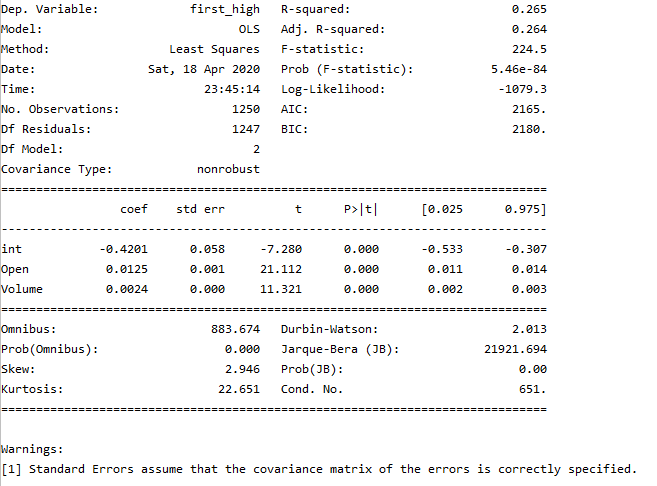


Make a prediction using test set be holt winter method.



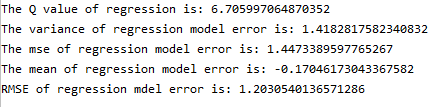


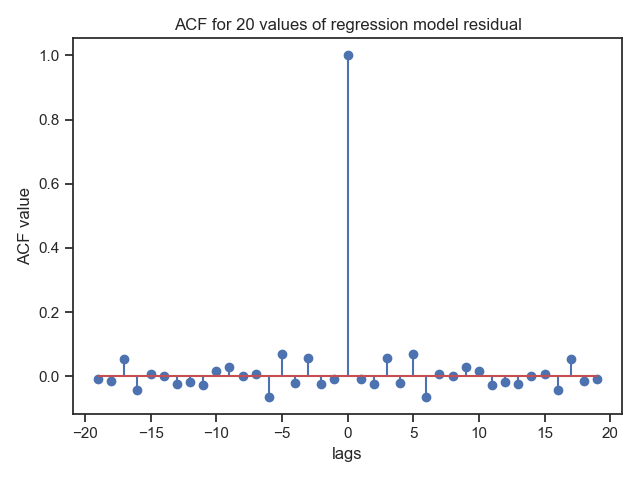
**Linear regression:**

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

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.

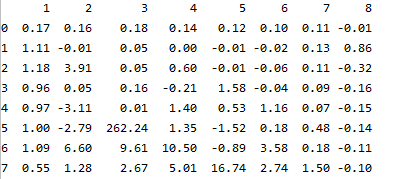


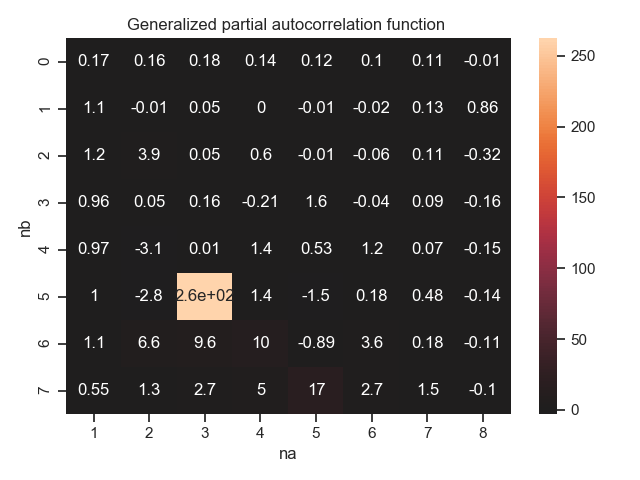


The residual of the regression model is not autocorrelated which is good.

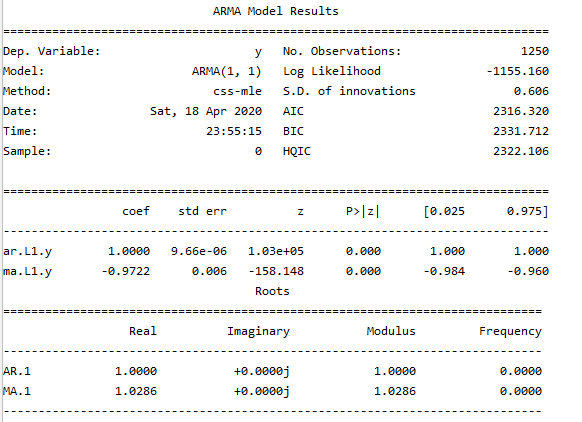
**ARMA model:**

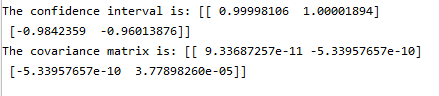
Firstly, I need to use GPAC table to determine the order of ARMA model.



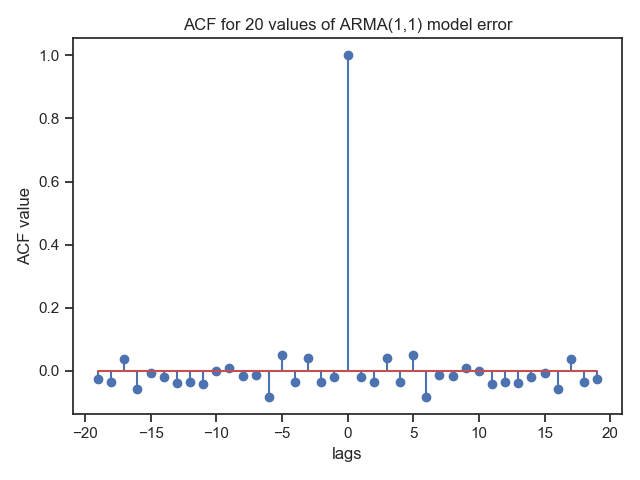


We can see that when na reaches 3, there is a high value in the GPAC table. So I will firstly take a try of **ARMA(1,1)** model to see if it passes the whiteness test and check for zero/pole cancellation. Then I will try **ARMA(2,1)** model to see if it passes the whiteness test and check for zero/pole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

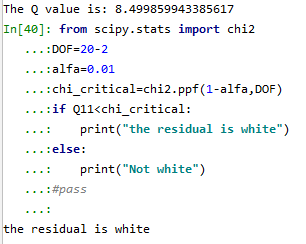
**ARMA(1,1)**



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



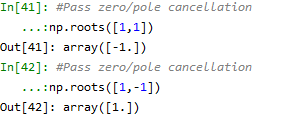
The ACF plot of the residual shows that the residuals are not autocorrelated which means the model is good.

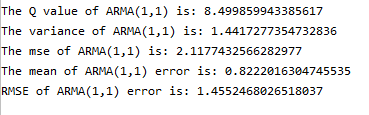
The lags I used here is 20 and na=1, nb=1. The result passes the whiteness test.

**zero/pole cancellation:**

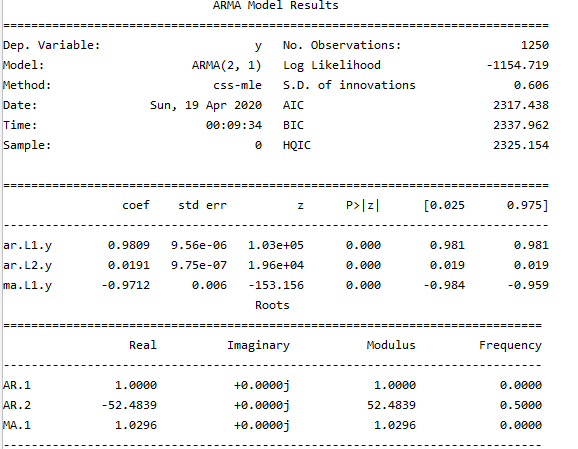
The parameter of AR model is 1 and MA model is about -1:

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

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

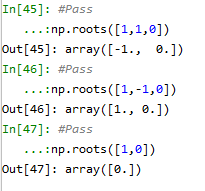


The mean and variance of ARMA(1,1) model is low(about 1) so this model is not biased.

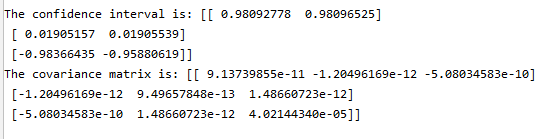
**ARMA(2,1)**

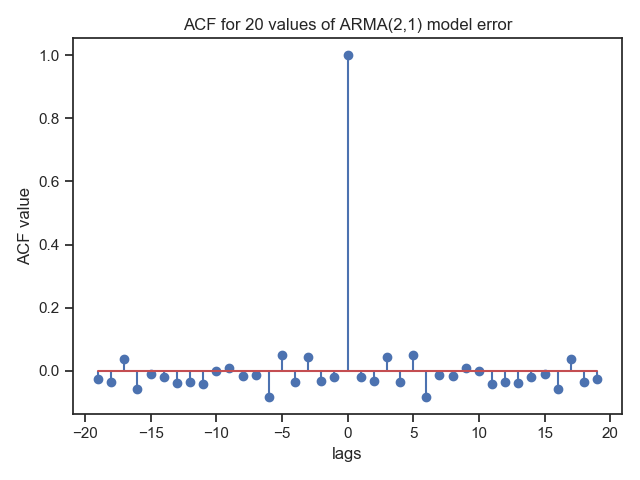
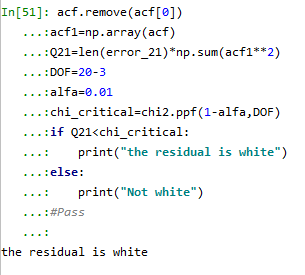
The model is: Y(t)=Y(t-1)+Y(t-2)+e(t)-e(t-1).

**zero/pole cancellation:**

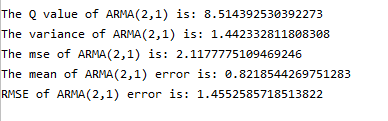


The three roots can be cancelled.



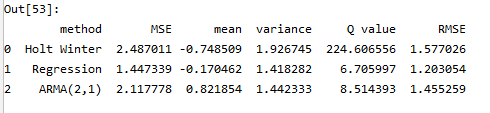


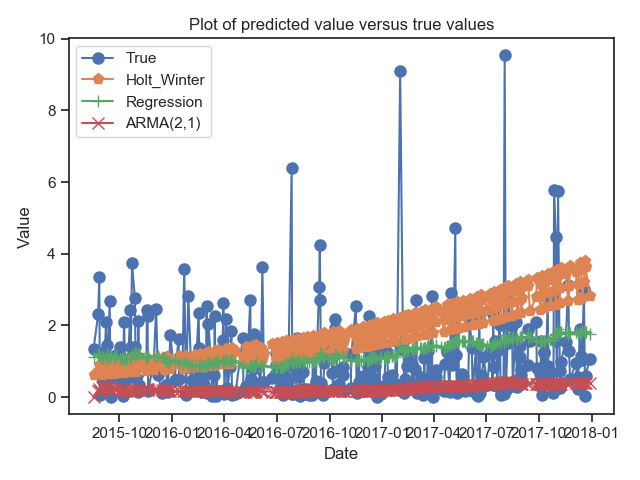
The lags here is still 20, na=2, nb=1, pass the whiteness test.



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**





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=**False**)  
*#%%***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()  
  
*#%%*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)  
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)  
*#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()