# arima_model

February 23, 2023

## 1 Building an ARIMA Model for a Financial Dataset

In this notebook, you will build an ARIMA model for AAPL stock closing prices. The lab objectives are:

- Pull data from Google Cloud Storage into a Pandas dataframe
- Learn how to prepare raw stock closing data for an ARIMA model
- Apply the Dickey-Fuller test
- Build an ARIMA model using the statsmodels library

**Make sure you restart the Python kernel after executing the `pip install` command below!** After you restart the kernel you don't have to execute the command again.

[1]:
```
!pip install --user statsmodels
```

```
Requirement already satisfied: statsmodels in /opt/conda/lib/python3.7/site-
packages (0.13.5)
Requirement already satisfied: scipy>=1.3 in /opt/conda/lib/python3.7/site-
packages (from statsmodels) (1.7.3)
Requirement already satisfied: numpy>=1.17 in /opt/conda/lib/python3.7/site-
packages (from statsmodels) (1.21.6)
Requirement already satisfied: packaging>=21.3 in /opt/conda/lib/python3.7/site-
packages (from statsmodels) (23.0)
Requirement already satisfied: patsy>=0.5.2 in /opt/conda/lib/python3.7/site-
packages (from statsmodels) (0.5.3)
Requirement already satisfied: pandas>=0.25 in /opt/conda/lib/python3.7/site-
packages (from statsmodels) (1.3.5)
Requirement already satisfied: python-dateutil>=2.7.3 in
/opt/conda/lib/python3.7/site-packages (from pandas>=0.25->statsmodels) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /opt/conda/lib/python3.7/site-
packages (from pandas>=0.25->statsmodels) (2022.7.1)
Requirement already satisfied: six in /opt/conda/lib/python3.7/site-packages
(from patsy>=0.5.2->statsmodels) (1.16.0)
```

[2]:
```
%matplotlib inline

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

```python
import datetime

%config InlineBackend.figure_format = 'retina'
```

### 1.1 Import data from Google Clod Storage

In this section we'll read some ten years' worth of AAPL stock data into a Pandas dataframe. We want to modify the dataframe such that it represents a time series. This is achieved by setting the date as the index.

[3]:
```python
df = pd.read_csv('gs://cloud-training/ai4f/AAPL10Y.csv')

df['date'] = pd.to_datetime(df['date'])
df.sort_values('date', inplace=True)
df.set_index('date', inplace=True)

print(df.shape)

df.head()
```

```
(2517, 5)
```

[3]:

| | close | volume | open | high | low |
|---|---|---|---|---|---|
| date | | | | | |
| 2009-06-03 | 20.1357 | 140628992.0 | 20.0000 | 20.1586 | 19.8671 |
| 2009-06-04 | 20.5343 | 136628071.0 | 20.0186 | 20.5971 | 20.0057 |
| 2009-06-05 | 20.6671 | 157944127.0 | 20.7586 | 20.9143 | 20.4586 |
| 2009-06-08 | 20.5500 | 232466290.0 | 20.5457 | 20.6043 | 19.9186 |
| 2009-06-09 | 20.3886 | 168830811.0 | 20.5443 | 20.6514 | 20.0786 |

### 1.2 Prepare data for ARIMA

The first step in our preparation is to resample the data such that stock closing prices are aggregated on a weekly basis.

[4]:
```python
df_week = df.resample('w').mean()
df_week = df_week[['close']]
df_week.head()
```

[4]:

| | close |
|---|---|
| date | |
| 2009-06-07 | 20.445700 |
| 2009-06-14 | 20.106860 |
| 2009-06-21 | 19.525140 |
| 2009-06-28 | 19.711440 |
| 2009-07-05 | 20.258925 |

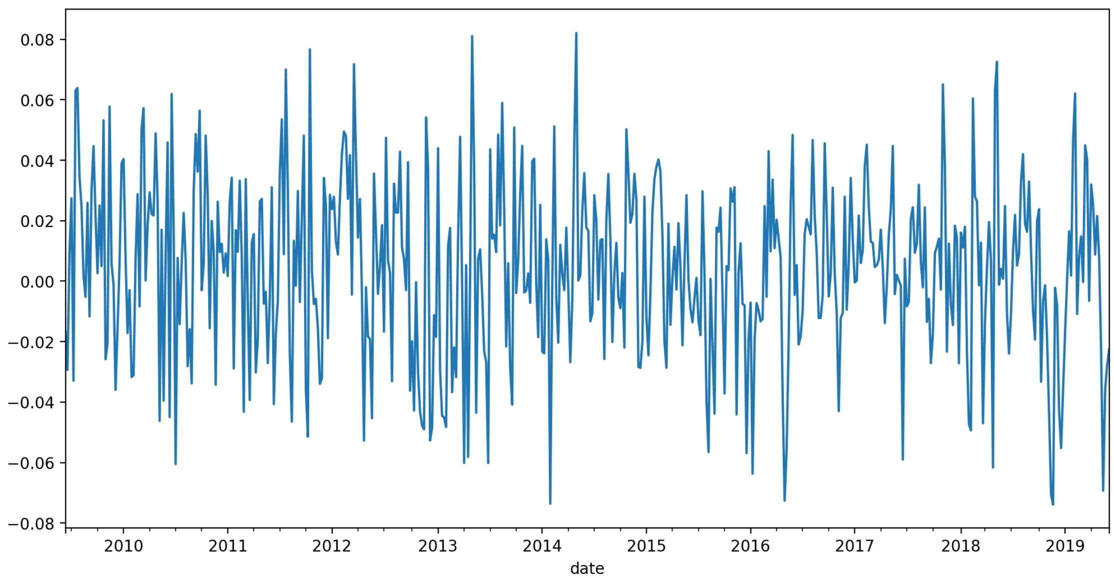
Let's create a column for weekly returns. Take the log to of the returns to normalize large fluctuations.

[5]:
```
df_week['weekly_ret'] = np.log(df_week['close']).diff()
df_week.head()
```

[5]:
```
                close  weekly_ret
date
2009-06-07  20.445700         NaN
2009-06-14  20.106860   -0.016712
2009-06-21  19.525140   -0.029358
2009-06-28  19.711440    0.009496
2009-07-05  20.258925    0.027396
```

[6]:
```
# drop null rows
df_week.dropna(inplace=True)
```

[7]:
```
df_week.weekly_ret.plot(kind='line', figsize=(12, 6));
```




[8]:
```
udiff = df_week.drop(['close'], axis=1)
udiff.head()
```

[8]:
```
            weekly_ret
date
2009-06-14   -0.016712
2009-06-21   -0.029358
2009-06-28    0.009496
2009-07-05    0.027396
```

```
2009-07-12   -0.032905
```

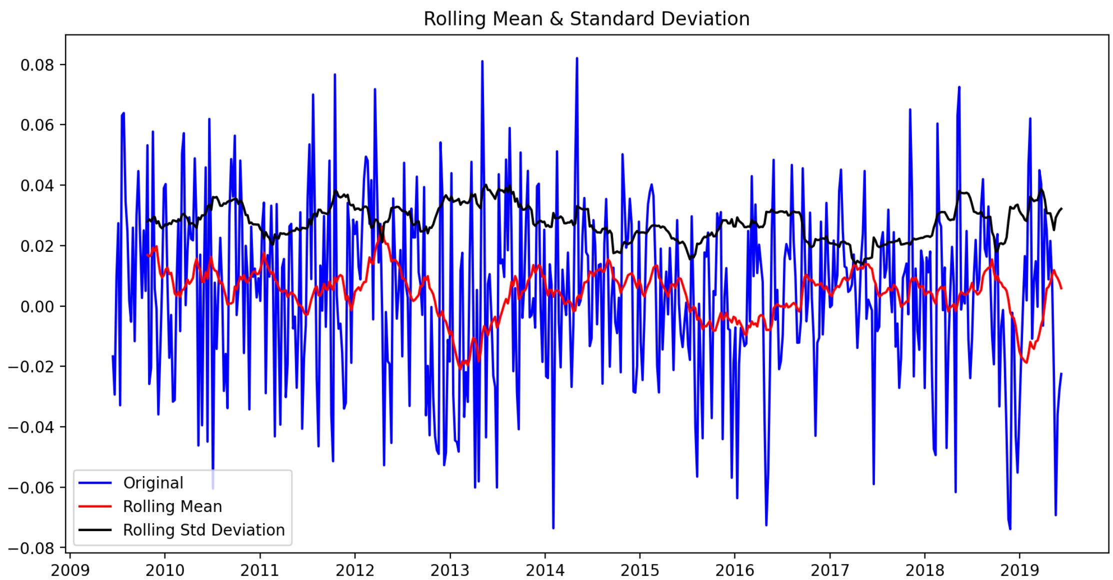
### 1.3 Test for stationarity of the udiff series

Time series are stationary if they do not contain trends or seasonal swings. The Dickey-Fuller test can be used to test for stationarity.

[9]:
```python
import statsmodels.api as sm
from statsmodels.tsa.stattools import adfuller
```

[10]:
```python
rolmean = udiff.rolling(20).mean()
rolstd = udiff.rolling(20).std()
```

[11]:
```python
plt.figure(figsize=(12, 6))
orig = plt.plot(udiff, color='blue', label='Original')
mean = plt.plot(rolmean, color='red', label='Rolling Mean')
std = plt.plot(rolstd, color='black', label = 'Rolling Std Deviation')
plt.title('Rolling Mean & Standard Deviation')
plt.legend(loc='best')
plt.show(block=False)
```




[12]:
```python
# Perform Dickey-Fuller test
dftest = sm.tsa.adfuller(udiff.weekly_ret, autolag='AIC')
dfoutput = pd.Series(dftest[0:4], index=['Test Statistic', 'p-value', '#Lags␣
 ↪Used', 'Number of Observations Used'])
for key, value in dftest[4].items():
    dfoutput['Critical Value ({0})'.format(key)] = value
```

```
dfoutput
```

[12]:
```
Test Statistic                -1.105002e+01
p-value                        5.107869e-20
#Lags Used                     2.000000e+00
Number of Observations Used    5.190000e+02
Critical Value (1%)           -3.443013e+00
Critical Value (5%)           -2.867125e+00
Critical Value (10%)          -2.569745e+00
dtype: float64
```

With a p-value $< 0.05$, we can reject the null hypotehsis. This data set is stationary.

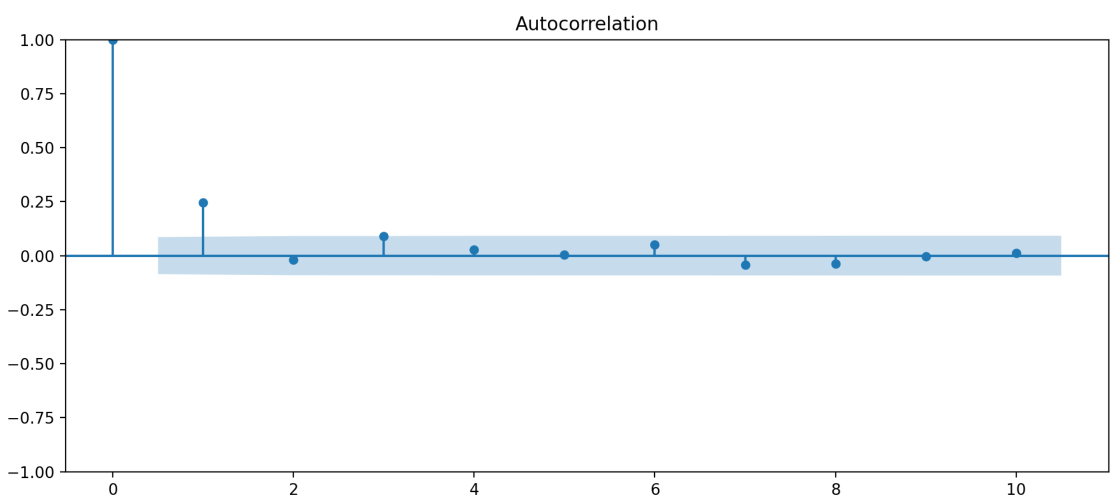
### 1.4 ACF and PACF Charts

Making autocorrelation and partial autocorrelation charts help us choose hyperparameters for the ARIMA model.

The ACF gives us a measure of how much each "y" value is correlated to the previous n "y" values prior.

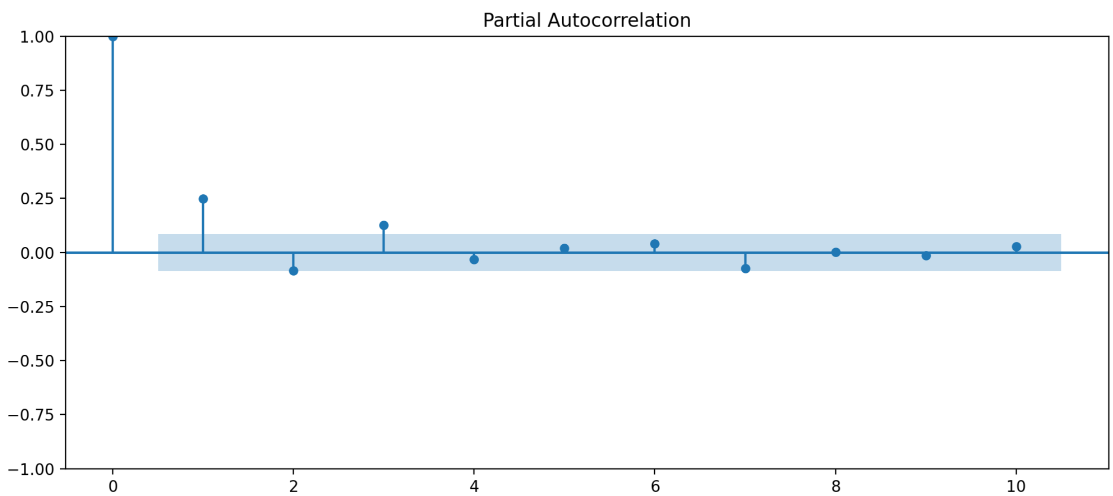
The PACF is the partial correlation function gives us (a sample of) the amount of correlation between two "y" values separated by n lags excluding the impact of all the "y" values in between them.

[13]:
```python
from statsmodels.graphics.tsaplots import plot_acf

# the autocorrelation chart provides just the correlation at increasing lags
fig, ax = plt.subplots(figsize=(12,5))
plot_acf(udiff.values, lags=10, ax=ax)
plt.show()
```



[14]:
```
from statsmodels.graphics.tsaplots import plot_pacf

fig, ax = plt.subplots(figsize=(12,5))
plot_pacf(udiff.values, lags=10, ax=ax)
plt.show()
```

```
/opt/conda/lib/python3.7/site-packages/statsmodels/graphics/tsaplots.py:353:
FutureWarning: The default method 'yw' can produce PACF values outside of the
[-1,1] interval. After 0.13, the default will change tounadjusted Yule-Walker
('ywm'). You can use this method now by setting method='ywm'.
  FutureWarning,
```




The table below summarizes the patterns of the ACF and PACF.

The above chart shows that reading PACF gives us a lag "p" = 3 and reading ACF gives us a lag "q" of 1. Let's Use Statsmodel's ARMA with those parameters to build a model. The way to evaluate the model is to look at AIC - see if it reduces or increases. The lower the AIC (i.e. the more negative it is), the better the model.

### 1.5 Build ARIMA Model

Since we differenced the weekly closing prices, we technically only need to build an ARMA model. The data has already been integrated and is stationary.

[15]:
```
from statsmodels.tsa.arima.model import ARIMA

# Notice that you have to use udiff - the differenced data rather than the
 ↪original data.
ar1 = ARIMA(udiff.values, order = (3, 0,1)).fit()
ar1.summary()
```

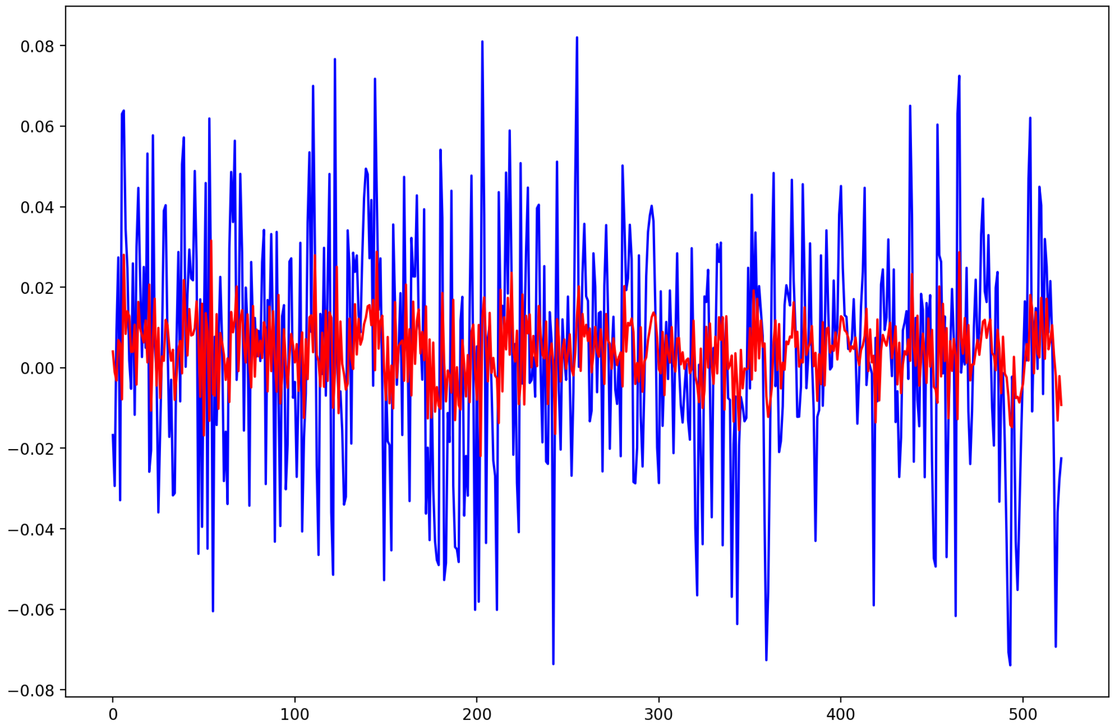
[15]:

```
<class 'statsmodels.iolib.summary.Summary'>
"""
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  522
Model:                 ARIMA(3, 0, 1)   Log Likelihood                1131.553
Date:                Thu, 23 Feb 2023   AIC                          -2251.105
Time:                        07:59:23   BIC                          -2225.559
Sample:                             0   HQIC                         -2241.100
                                - 522
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0040      0.002      2.345      0.019       0.001       0.007
ar.L1          0.1259      0.337      0.373      0.709      -0.536       0.787
ar.L2         -0.0770      0.101     -0.765      0.444      -0.274       0.120
ar.L3          0.1140      0.058      1.958      0.050      -0.000       0.228
ma.L1          0.1562      0.340      0.459      0.646      -0.510       0.822
sigma2         0.0008   4.79e-05     15.993      0.000       0.001       0.001
================================================================================
===
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):
0.00
Prob(Q):                              0.98   Prob(JB):
1.00
Heteroskedasticity (H):               0.75   Skew:
-0.01
Prob(H) (two-sided):                  0.06   Kurtosis:
3.00
================================================================================
===

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-
step).
"""
```

Our model doesn't do a good job predicting variance in the original data (peaks and valleys).

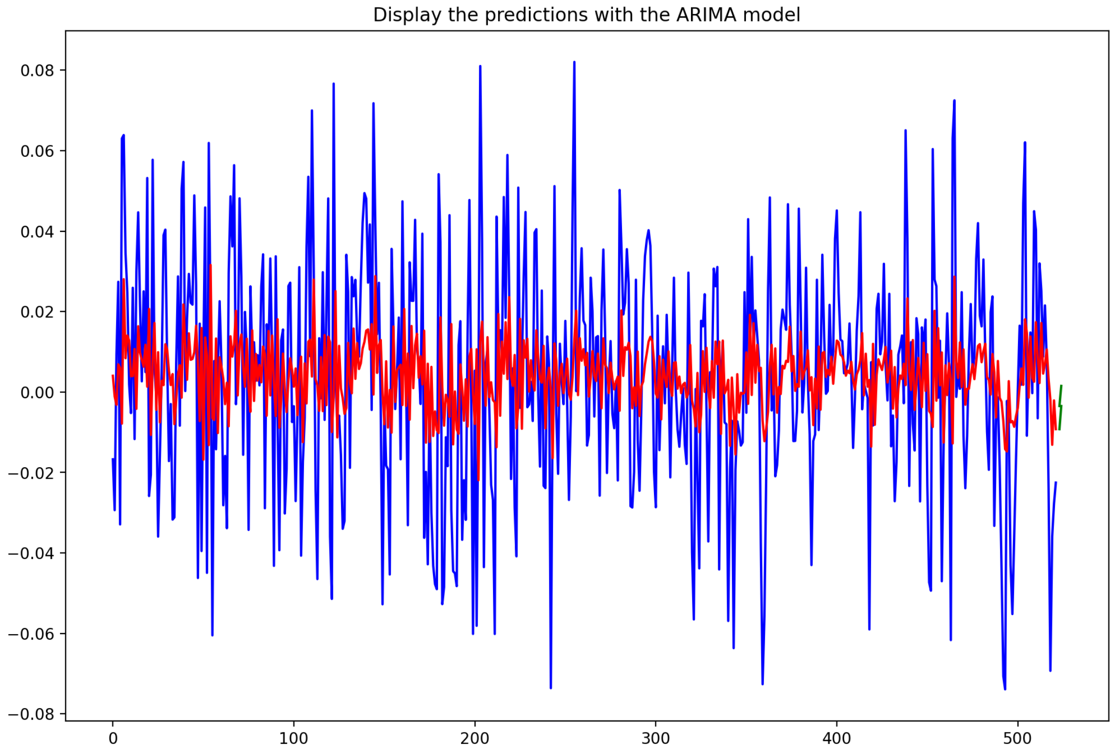
[16]:

```
plt.figure(figsize=(12, 8))
plt.plot(udiff.values, color='blue')
preds = ar1.fittedvalues
plt.plot(preds, color='red')
plt.show()
```




Let's make a forecast 2 weeks ahead:

[17]:
```
steps = 2

forecast = ar1.forecast(steps=steps)

plt.figure(figsize=(12, 8))
plt.plot(udiff.values, color='blue')

preds = ar1.fittedvalues
plt.plot(preds, color='red')

plt.plot(pd.DataFrame(np.array([preds[-1],forecast[0]]).T,index=range(len(udiff.
 ↪values)+1, len(udiff.values)+3)), color='green')
plt.plot(pd.DataFrame(forecast,index=range(len(udiff.values)+1, len(udiff.
 ↪values)+1+steps)), color='green')
plt.title('Display the predictions with the ARIMA model')
plt.show()
```




The forecast is not great but if you tune the hyper parameters some more, you might be able to reduce the errors.