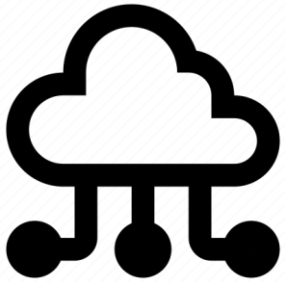


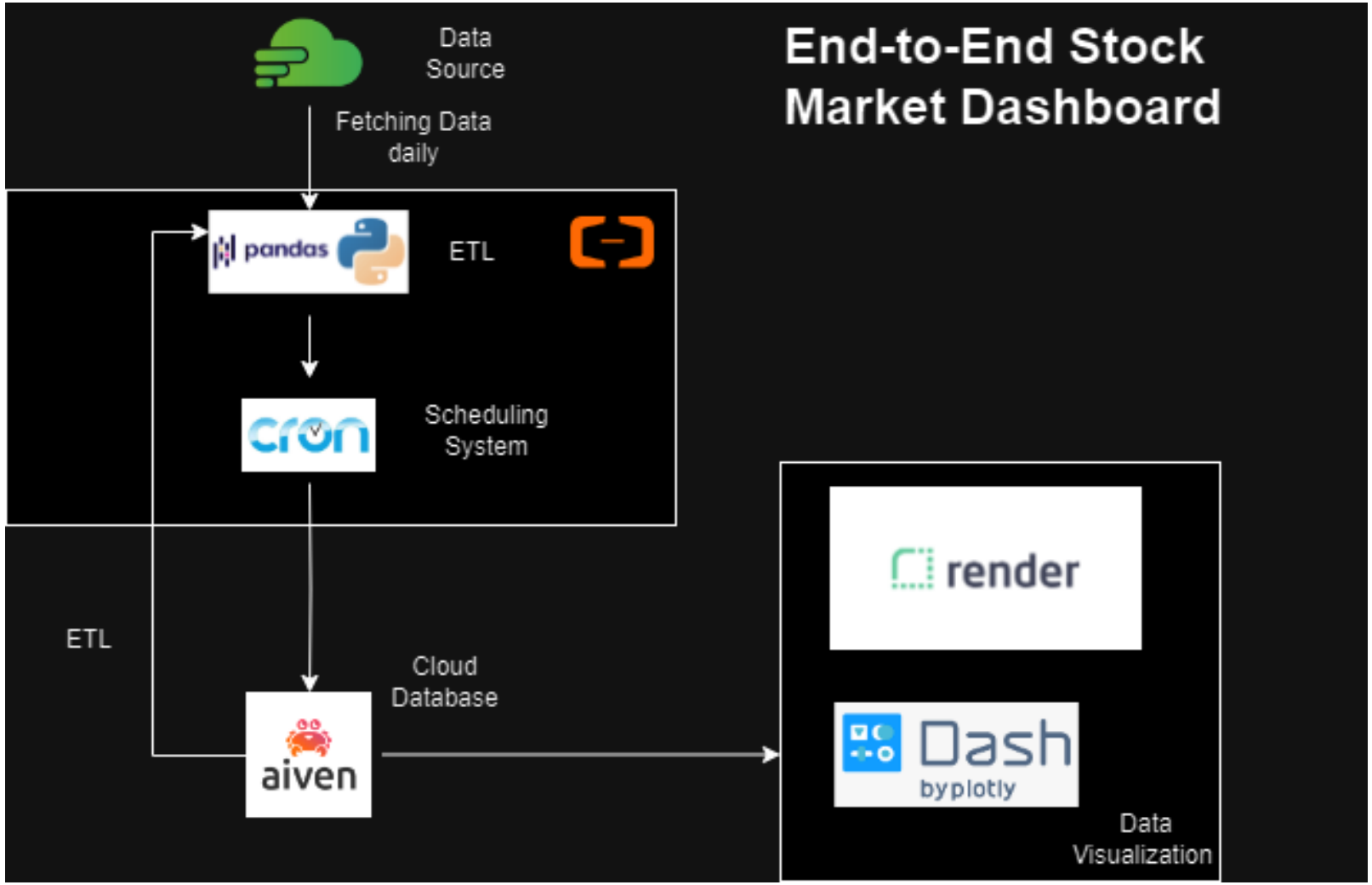
Building a Cloud Data Pipeline for Financial Analysis with SARIMAX

LINK TO DASHBOARD

Muhamad Tartila Sahid

mtartila@gmail.com

DIAGRAM


End-to-End Stock Market Dashboard
Data Source
Fetching Data daily
pandas
ETL
Scheduling System
ETL
Cloud Database
aiven
render
Dash
by plotly
Data Visualization

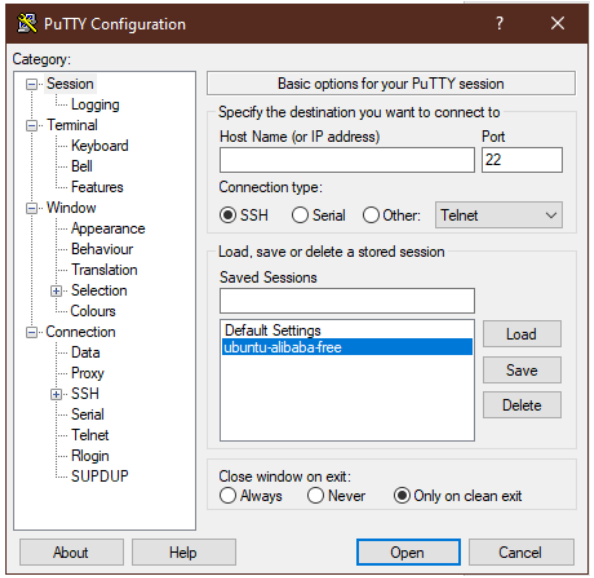
Backend Cloud Technicalities

EXTRACTION	**Alibaba ECS** instance fetches API data from **Finnhub** every 10 minutes every 8:30 PM – 3:00 AM (GMT+7)
TRANSFORMATION	Calculating the average of daily closing price once the market closed using **python**
STORAGE	Raw data and transformed data are stored within **Aiven Cloud DB**
SCORING	**SARIMAX** model is used for daily scoring, training data from May 2023 – May 2024
DASHBOARD	Results are uploaded to a **Plotly Dash dashboard** hosted on **Render**

System Information & API

Alibaba ECS :

1. **Zone** : Singapore C
2. **Instance type** : ecs.t5-lc1m1.small
3. **CPU and Memory** : 1 Core (vCPUs), 1 GiB
4. **Operating System**: Ubuntu 20.04 64 bit



Finnhub API :

1. Rate limit **60 calls** per seconds
2. Call method used : [Finnhub_Quote_Call]

Aiven DB :

1. **CPU : 1**
2. **RAM : 1 GB**
3. **Storage : 5 GB**
4. **Version : MySQL 8.0.30**

PIPELINE

Data Extraction - Scripts

- Fetching and storing data from finnhub API to Aiven DB : Extraction
- Data is fetched from 8.30 PM – 3 AM (GMT +7) for every 10 minutes daily : Scheduler

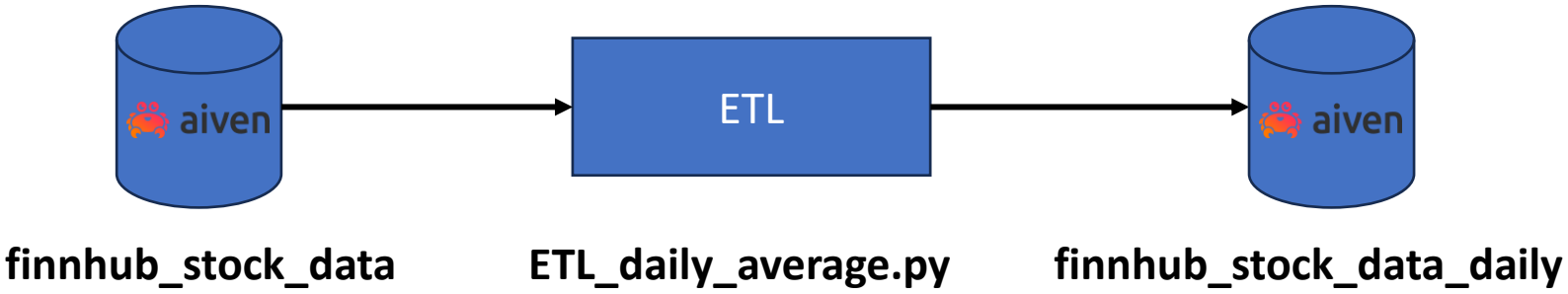
symbol	record_date	open	high	low	close	previous_	inserted_date
AAPL	6/7/2024 2:16	195.685	196.5	194.95	195.08	195.87	6/7/2024 2:16
AAPL	6/7/2024 2:26	195.685	196.5	194.95	195.25	195.87	6/7/2024 2:26
AAPL	6/7/2024 2:36	195.685	196.5	194.95	195.12	195.87	6/7/2024 2:36
AAPL	6/7/2024 2:46	195.685	196.5	194.95	195.035	195.87	6/7/2024 2:46
AAPL	6/7/2024 2:56	195.685	196.5	194.38	194.42	195.87	6/7/2024 2:56
AAPL	6/7/2024 20:36	195.06	195.75	194.55	194.67	194.48	6/7/2024 20:36
AAPL	6/7/2024 20:46	195.06	195.75	194.14	194.515	194.48	6/7/2024 20:46
AAPL	6/7/2024 20:56	195.06	195.75	194.14	194.58	194.48	6/7/2024 20:56
AAPL	6/7/2024 21:06	195.06	195.75	194.14	194.76	194.48	6/7/2024 21:06
AAPL	6/7/2024 21:16	195.06	195.75	194.14	194.956	194.48	6/7/2024 21:16
AAPL	6/7/2024 21:26	195.06	195.75	194.14	194.94	194.48	6/7/2024 21:26
AAPL	6/7/2024 21:36	195.06	195.75	194.14	195.03	194.48	6/7/2024 21:36
AAPL	6/7/2024 21:46	195.06	195.75	194.14	195.72	194.48	6/7/2024 21:46
AAPL	6/7/2024 21:56	195.06	195.97	194.14	195.685	194.48	6/7/2024 21:56
AAPL	6/7/2024 22:06	195.06	195.97	194.14	195.105	194.48	6/7/2024 22:06

```
 GNU nano 4.8                    /tmp/crontab.RuH71U/crontab
@reboot /root/my_python_project/finnhub_cron.sh
```

Data ETL - Scripts

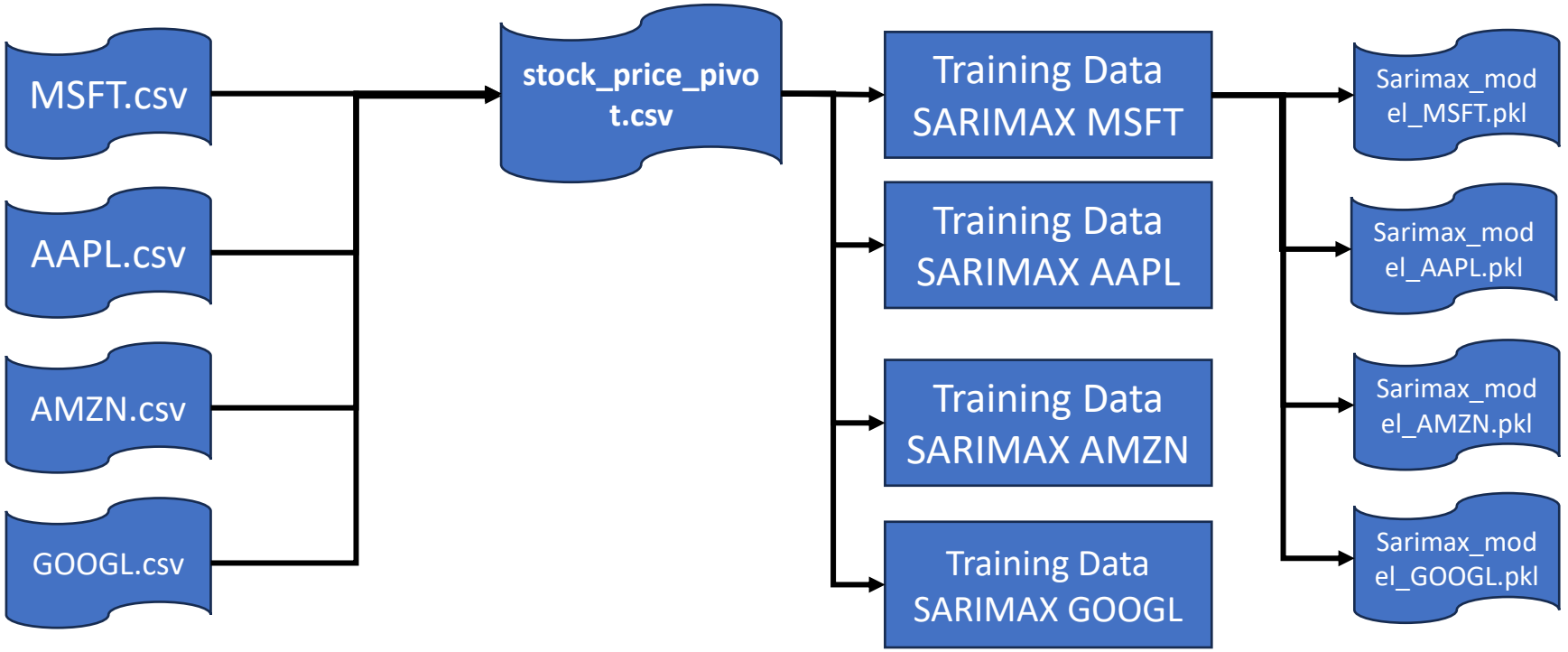
- Fetching and transforming data from table **finnhub_stock_data** to **finnhub_stock_data_daily** : ETL
- Job scheduler to run every **4 AM (GMT+7)** right after the stock market closes : ETL Scheduler

```
0 4  * * * /root/my_python_project/etl_cron.sh
```



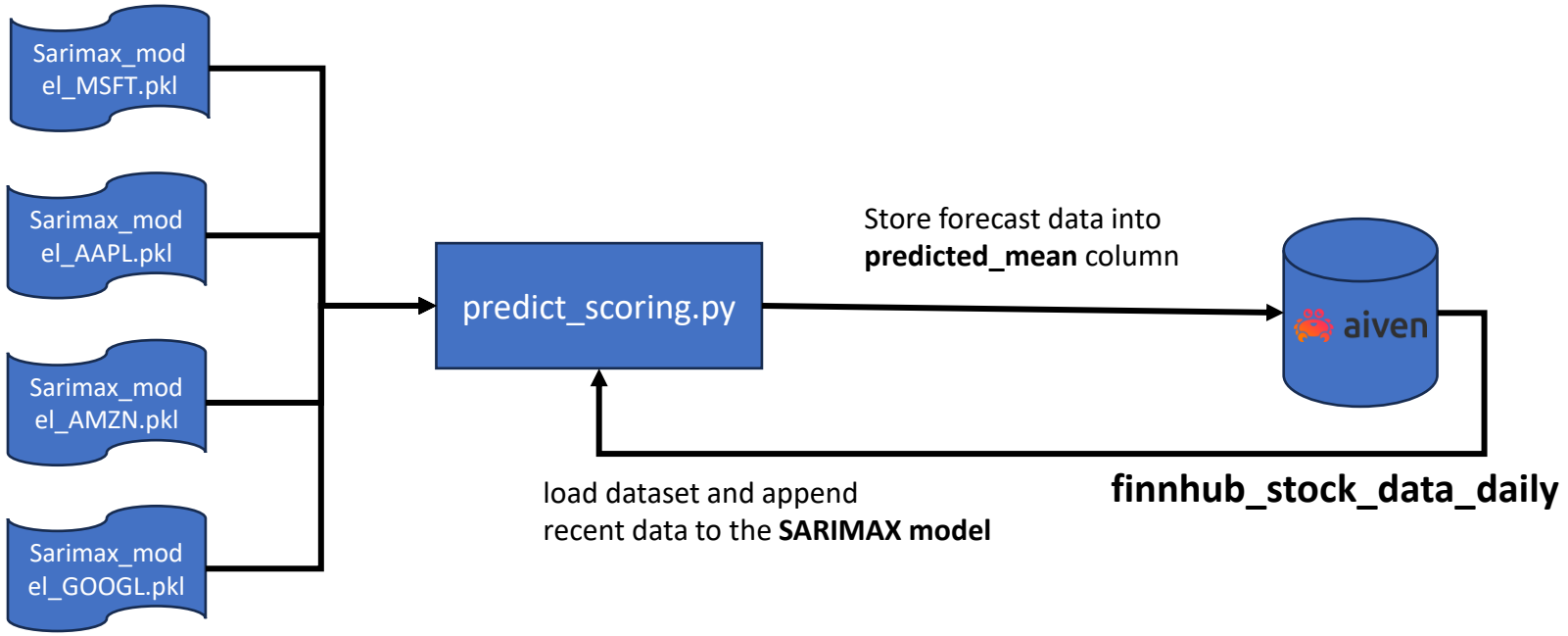
SARIMAX Model - Training

- Models were trained locally using **jupyter notebook** and saved with **joblib**
- Using datasets from yahoo finance from 9/6/2023 – 7/6/2024 : Dataset Creation
- Scripts for training each model : Model Training



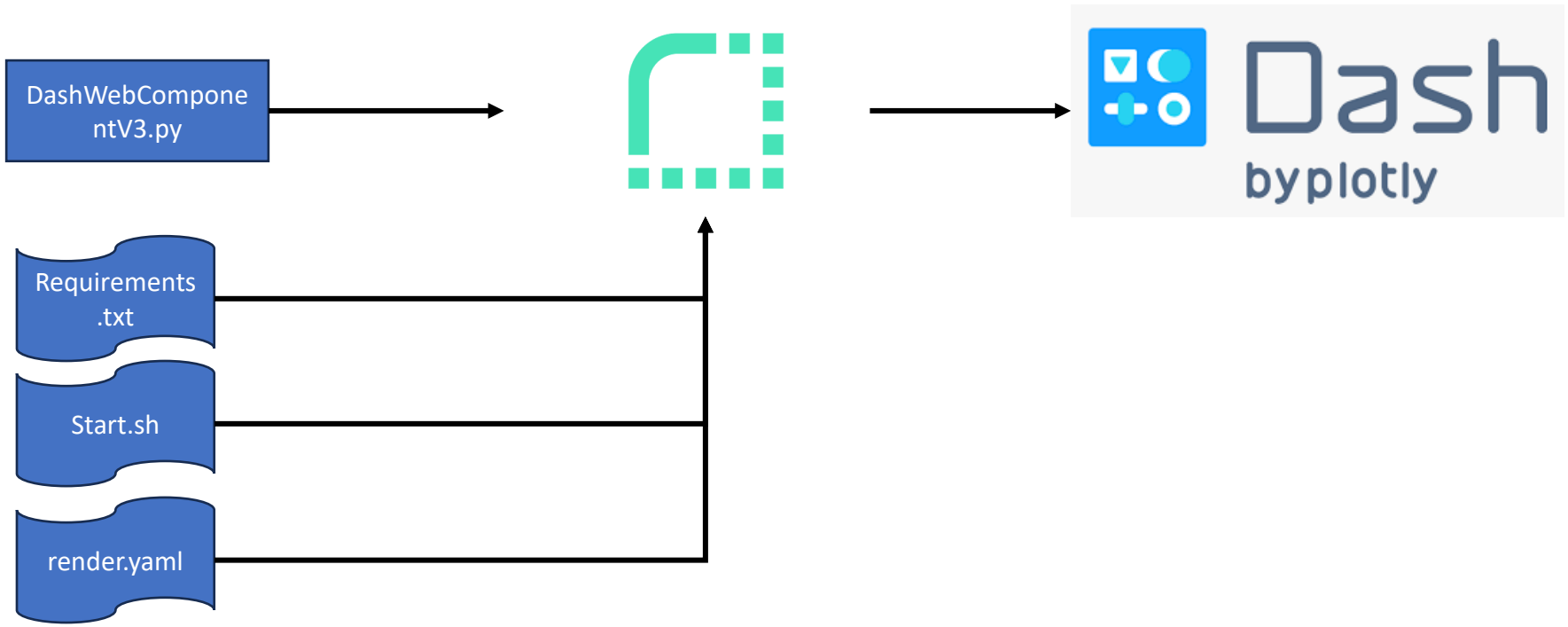
SARIMAX Model - Scoring

- Daily scoring to predict the next 5 business day forecast : Forecasting



Render Dashboard

- Required GITHUB to deploy dashboard : Render Deployment
- Python Script for dynamic dashboard : Plotly Dash Web Component



EDA

SARIMAX

- Statistical model that Combines autoregressive **(AR)**, Integration **(I)**, and moving average **(MA)** components with **S**easonal and e**X**ogenous factors **(p, d, q)(P, D, Q, s).**
- AR : takes past values to predict current values (p) to a linear regression

$$X_t = \varepsilon_t + \theta_1 \varepsilon_{t-1} + \ldots + \theta_q \varepsilon_{t-q}$$

- I : differencing of observations to make time series stationary so it can be predicted

$$Y'_t = Y_t - Y_{t-1}$$

- MA : takes past errors in predictions (q) to a linear regression

$$X_t = \varepsilon_t + \phi_1 X_{t-1} + \ldots + \phi_p X_{t-p}$$

- P *:* Number of seasonal autoregressive term (P = 1 indicates value from 12 months ago as one season back).
- D : Seasonal differencing substracts the value of the same period in previous season from current value.
- Q : Number of seasonal moving average terms.
- s : number of time steps per season.

$$Y_t = c + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} + \Theta(L)\varepsilon_t + \Phi(L^s)\varepsilon_{t-s} + \Theta_Q(L^s)\varepsilon_{t-Qs}$$

Augmented Dickey-Fuller (ADF)

- **Statistical** test used to determine whether a given time series is stationary
- **p-value**: The probability that the observed data would occur if the null hypothesis were true. A low p-value (typically < 0.05) indicates that we can reject the null hypothesis in favor of the alternative hypothesis, suggesting that the time series is stationary.
- With the p-value of several stocks below, it indicates the needs of differencing by one order to achieve stationarity (**I = 1).**

```
In [6]:
#ADF TEST TO CHECK FOR STATIONARITY OF A DATAFRAME

res = sm.tsa.adfuller(df['close_AAPL'])
print('p-value:{}'.format(res[1]))

p-value:0.2709793792635439
```

```
In [7]:
res = sm.tsa.adfuller(df['close_AAPL'].diff().dropna(),regression='c')
print('p-value:{}'.format(res[1]))

p-value:4.243421588274003e-27
```

AAPL

```
In [6]:
#ADF TEST TO CHECK FOR STATIONARITY OF A DATAFRAME

res = sm.tsa.adfuller(df['close_GOOGL'])
print('p-value:{}'.format(res[1]))

p-value:0.8722416494410326
```

```
In [7]:
res = sm.tsa.adfuller(df['close_GOOGL'].diff().dropna(),regression='c')
print('p-value:{}'.format(res[1]))

p-value:5.8307593346714404e-30
```

GOOGL

```
In [15]:
#ADF TEST TO CHECK FOR STATIONARITY OF A DATAFRAME

res = sm.tsa.adfuller(df['close_AMZN'])
print('p-value:{}'.format(res[1]))

p-value:0.8774311371539011
```

```
In [16]:
res = sm.tsa.adfuller(df['close_AMZN'].diff().dropna(),regression='c')
print('p-value:{}'.format(res[1]))

p-value:2.673751444536066e-26
```

AMZN

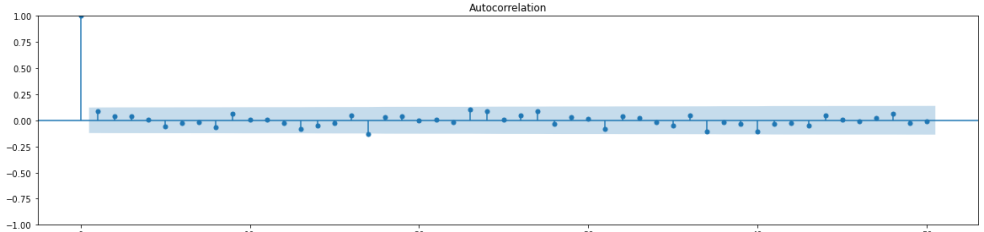
```
In [6]:
#ADF TEST TO CHECK FOR STATIONARITY OF A DATAFRAME

res = sm.tsa.adfuller(df['close_MSFT'])
print('p-value:{}'.format(res[1]))

p-value:0.89275611704664
```

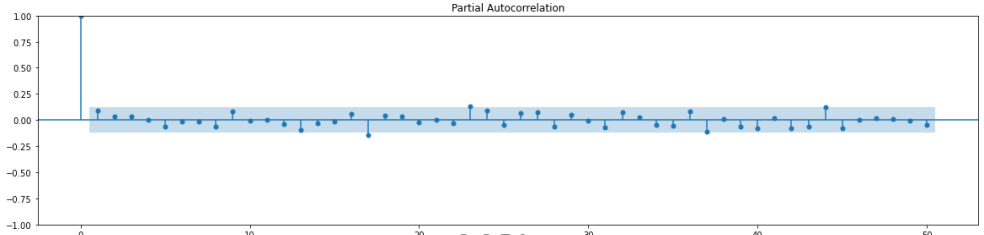
```
In [7]:
res = sm.tsa.adfuller(df['close_MSFT'].diff().dropna(),regression='c')
print('p-value:{}'.format(res[1]))

p-value:5.796597124653734e-25
```

MSFT

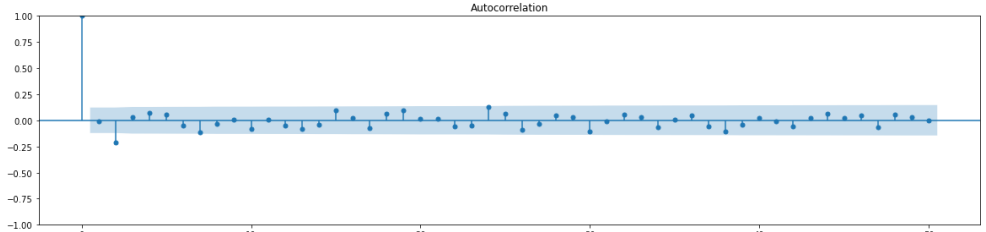
Determining AR and MA Components

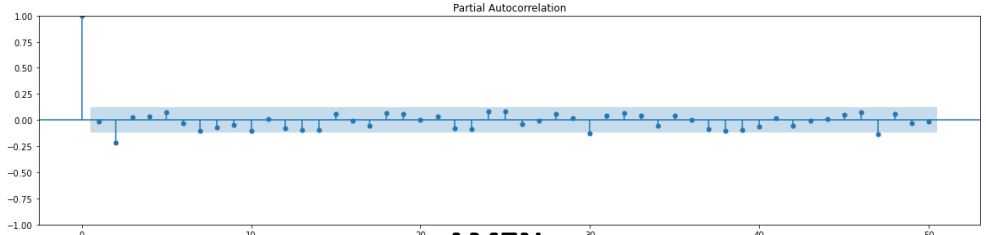
- ACF and PACF concludes no correlation of AR **(0)** and MA **(0)** (this shows how complex stock prices are to predict).



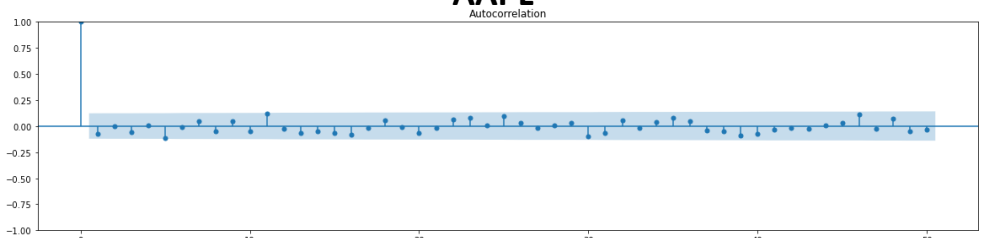


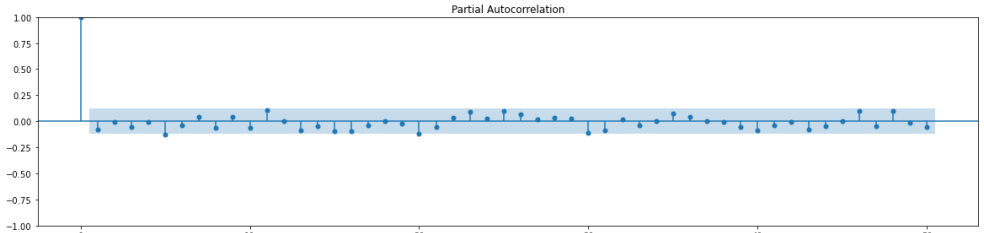
AAPL



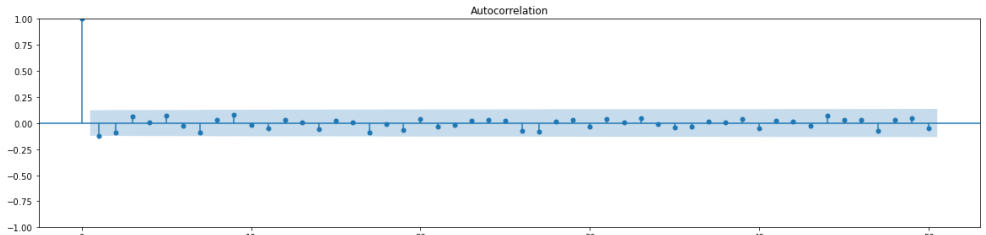


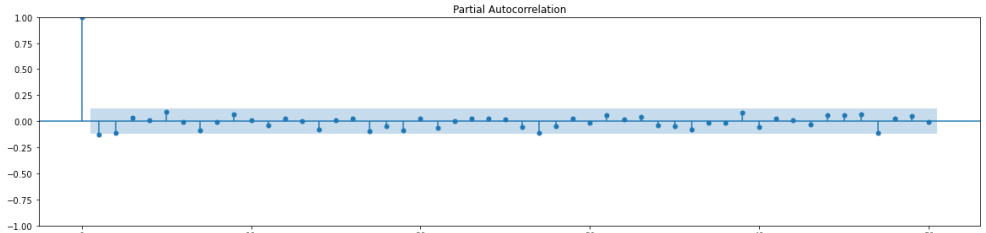
AMZN





GOOGL





MSFT

DATA MODELLING

Defining Parameters

The parameter used for each stocks are **(0,1,0) (1,1,1,5)** with assumption :

- Seasonal autoregressive of **P = 1**, the value is influenced by 5 periods ago which indicates the total business days in a week.
- Seasonal differencing of **D = 1**, the model has differencing it once at lag of 5 periods.
- Seasonal moving average of **Q = 1**, the value of the series is influenced by the noise 5 periods ago.
- Seasonal Periodicity of **s = 5**, indicating seasonality repeats every 5 time periods. Which is actually 5 business days of the week.
- Autoregressive of **p = 0**.
- Integration of **d = 1.**
- Moving average of **q = 0.**

Results Example

SARIMAX Results

Dep. Variable:	close_MSFT	No. Observations:	260
Model:	SARIMAX(0, 1, 0)x(1, 1, [1], 5)	Log Likelihood	-689.053
Date:	Wed, 12 Jun 2024	AIC	1390.106
Time:	11:27:27	BIC	1411.330
Sample:	06-12-2023	HQIC	1398.645
	- 06-07-2024		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
close_AAPL	0.5255	0.103	5.079	0.000	0.323	0.728
close_AMZN	0.8649	0.101	8.522	0.000	0.666	1.064
close_GOOGL	0.2448	0.080	3.061	0.002	0.088	0.402
ar.S.L5	0.1206	0.085	1.423	0.155	-0.046	0.287
ma.S.L5	-0.9994	2.738	-0.365	0.715	-6.365	4.366
sigma2	12.3702	33.442	0.370	0.711	-53.174	77.914

Ljung-Box (L1) (Q):	5.06	Jarque-Bera (JB):	299.70
Prob(Q):	0.02	Prob(JB):	0.00
Heteroskedasticity (H):	0.97	Skew:	0.58
Prob(H) (two-sided):	0.89	Kurtosis:	8.20

- While having **good correlation** between close prices of AAPL, AMZN, and GOOGL to MSFT, the model needs more refinement due to having **non significant seasonal terms**.
- Refining these seasonal parameters is needed find the **optimal model**.

KEY LEARNING

Conclusion

- Model output a **random walk** to represent the series gradually increase or decrease over time with several **drift** in nature.
- With said **unpredictability** and **stationarity**, one important thing to learn is that price changes are “**random**” and cannot be fully predicted using past price data.

Area of Improvement

- Using hyper parameter **optimization** (Grid Search, Random Search, Bayesian Optimization, Genetic Algorithm, Particle Swarm Optimization) by taking intuitive of the range of parameters.
- Using other model methods to **compare** (ARCH/GARCH/LSTM/Prophet).
- Optimize the cloud **scheduling** method to ensure that the ECS instances operate only when necessary, reducing unnecessary resource usage.
- Setting up **monitoring** and **logging** to track performances of ECS.
- **Auto-Scaling** to the ECS to adjust resources based on workload.

THANK YOU