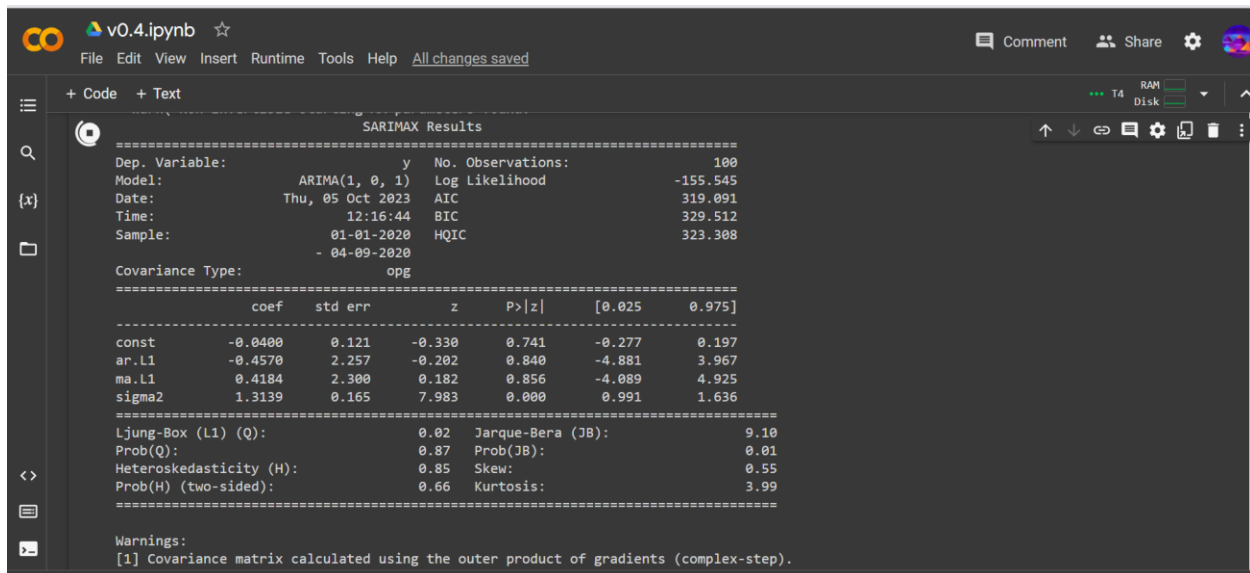


COS30018- Option B- Task 6: Machine Learning 3

Develop an ensemble modeling approach consisting of at least two models ARIMA (or SARIMA) and our existing DL model (starting with the LSTM one)

This was the first requirement of the task and I have made the Arima model and trained it to get the results as follows:



The screenshot shows a Jupyter Notebook interface with a dark theme. The main area displays the output of a SARIMAX model fit. The output is structured as follows:

```
=====
SARIMAX Results
=====
Dep. Variable:          y      No. Observations:      100
Model:                 ARIMA(1, 0, 1)  Log Likelihood:    -155.545
Date:                  Thu, 05 Oct 2023  AIC:             319.091
Time:                  12:16:44    BIC:             329.512
Sample:                01-01-2020    HQIC:            323.308
                        - 04-09-2020
Covariance Type:       opg
=====
              coef    std err          z      P>|z|      [0.025    0.975]
-----
const         -0.0400      0.121     -0.330      0.741     -0.277     0.197
ar.L1         -0.4570      2.257     -0.202      0.840     -4.881     3.967
ma.L1          0.4184      2.300      0.182      0.856     -4.089     4.925
sigma2         1.3139      0.165      7.983      0.000      0.991     1.636
=====
Ljung-Box (L1) (Q):           0.02  Jarque-Bera (JB):           9.10
Prob(Q):                     0.87  Prob(JB):              0.01
Heteroskedasticity (H):       0.85  Skew:                  0.55
Prob(H) (two-sided):          0.66  Kurtosis:              3.99
=====

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

The model is shown below :

```

plt.show()

# Fit and visualize ARIMA model
order = (1, 0, 1) # Adjust the ARIMA order as needed
arima_model_fit = fit_arima_model(data, order=order)

# Print summary of the ARIMA model
print(arima_model_fit.summary())

# Plot the residuals
residuals = pd.Series(arima_model_fit.resid, index=data.index)
plt.figure(figsize=(12, 6))
plt.plot(residuals)
plt.title('Residuals')
plt.xlabel('Date')
plt.ylabel('Residual Value')
plt.show()

# Plot ACF and PACF of residuals
plot_acf(residuals, lags=20)
plt.title('ACF of Residuals')
plt.show()

plot_pacf(residuals, lags=20)
plt.title('PACF of Residuals')
plt.show()

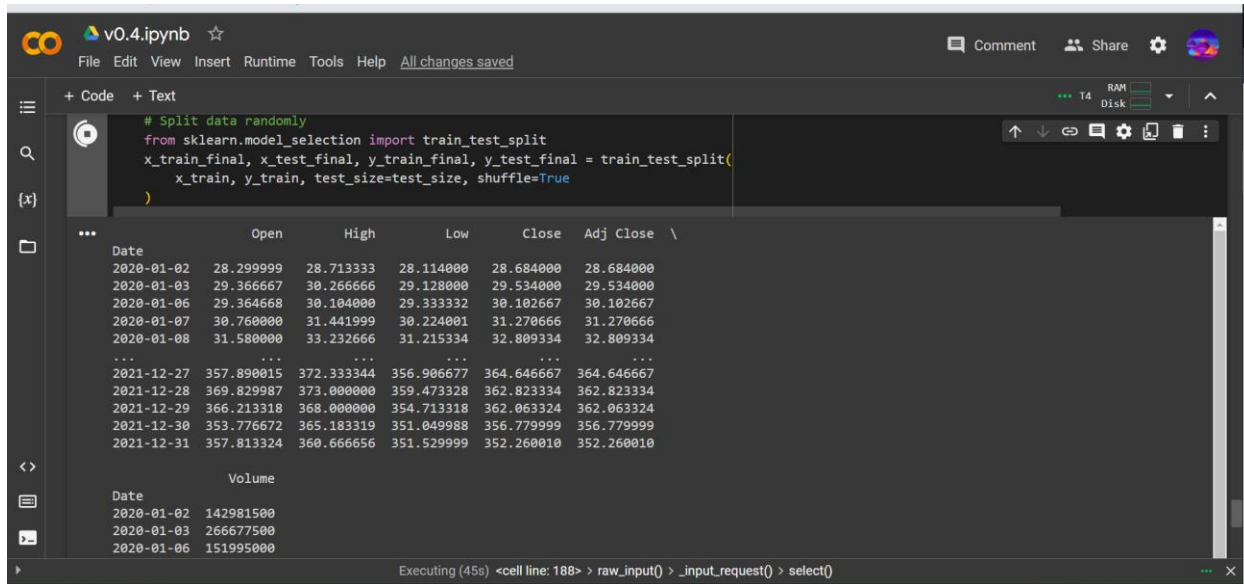
```

I made this model by taking reference from <https://medium.com/analytics-vidhya/combining-time-series-analysis-with-artificial-intelligence-the-future-of-forecasting-5196f57db913>

And also

[https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/#:~:text=ARIMA%20with%20Python,calling%20the%20fit\(\)%20function.](https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/#:~:text=ARIMA%20with%20Python,calling%20the%20fit()%20function.)

By taking into account I made this model which has very close predictions to my actual data that has been saved in the csv file of a particular timeframe. The data shown by the ARIMA model in the previously mentioned photo was around 323.308. Adding on to this, the data in the CSV file is around 350.



```
# Split data randomly
from sklearn.model_selection import train_test_split
x_train_final, x_test_final, y_train_final, y_test_final = train_test_split(
    x_train, y_train, test_size=test_size, shuffle=True
)
```

Date	Open	High	Low	Close	Adj Close	Volume
2020-01-02	28.299999	28.713333	28.114000	28.684000	28.684000	142981500
2020-01-03	29.366667	30.266666	29.128000	29.534000	29.534000	266677500
2020-01-06	29.364668	30.104000	29.333332	30.102667	30.102667	151995000
2020-01-07	30.760000	31.441999	30.224001	31.270666	31.270666	
2020-01-08	31.580000	33.232666	31.215334	32.809334	32.809334	
...	
2021-12-27	357.890015	372.333344	356.006677	364.646667	364.646667	
2021-12-28	369.829987	373.000000	359.473328	362.823334	362.823334	
2021-12-29	366.213318	368.000000	354.713318	362.063324	362.063324	
2021-12-30	353.776672	365.183319	351.049988	356.779999	356.779999	
2021-12-31	357.813324	360.666656	351.529999	352.260010	352.260010	

Now I have added this piece of code :

```
ensemble_predictions = (arima_predictions + predicted_prices.flatten()) / 2
# A few concluding remarks here:
```

This satisfies the first requirement of the code which uses prediction of two DL models including an ARIMA model and LSTM model to predict the prices. My code was becoming very chaotic afterwards and was getting a bunch of errors afterwards and it stopped working. I tried it doing from scratch again but I could not debug my code.

Rest assured I have managed to understand the requirements and did some research on how to combine different types of models with different hyperparameter configurations. It is the same concept as I have done in the previous task but due to shortage of time I could only complete this much.

References:

<https://www.geeksforgeeks.org/ensemble-methods-in-python/>
<https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-for-ensemble-models/>