GROUP WORK PROJECT # 1 MScFE 642: Deep Learning for Finance

GROUP NUMBER: 7634

FULL LEGAL NAME	LOCATION (COUNTRY)	EMAIL ADDRESS	MARK X FOR ANY NON-CONTRIBUTING MEMBER
Chih wei Ng	Malaysia	ncw388@gmail.com	
Bharat Swami	India	bharatswami1299@gmail.com	
Pole Okuttu Stephen	Uganda	poleoks@gmail.com	

Statement of integrity: By typing the names of all group members in the text boxes below, you confirm that the assignment submitted is original work produced by the group (excluding any non-contributing members identified with an "X" above).	
Team member 1	**Chih wei Ng**
Team member 2	**Bharat Swami**
Team member 3	**Pole Okuttu Stephen**

Use the box below to explain any attempts to reach out to a non-contributing member. Type (N/A) if all members contributed. **Note:** You may be required to provide proof of your outreach to non-contributing members upon request.

Group Number: 7634

This report is done with reference to the Colab file.

Step 1

For this project, the stock price of JPMorgan (JPM) has been selected, with the period being 1 Jan 2017 to 31 Dec 2023.

a)

The plot of the stock price of JPM is shown below.

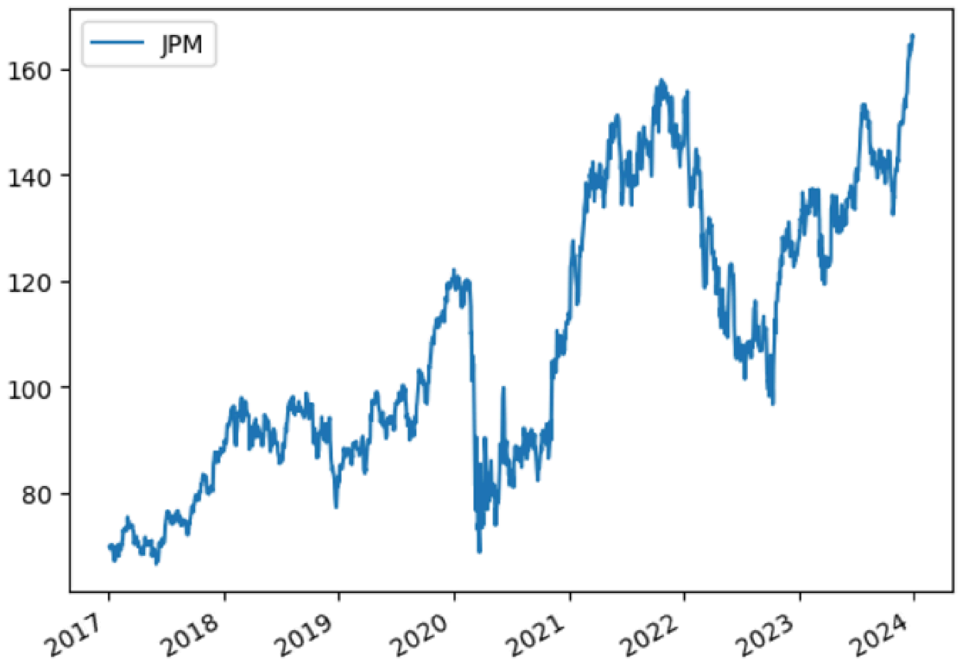


Fig 1: Plot of stock price of JPMorgan.

The summary statistics are below.

Ticker	JPM
count	1760.000000
mean	108.466272
std	25.615222
min	66.640770
25%	88.317184
50%	101.835449
75%	133.327747
max	166.328217

To check for stationarity, the Augmented Dickey-Fuller (ADF) test was carried out, and the results show that it is non-stationary, with the p-value more than 0.05.

```
ADF statistic: -1.049
p-value : 0.735
```

b)

Transformed version of Time Series

Again we are using the 'JPM' ticker time series data for our analysis. In this part, we are going to transform the time series from non-stationary to stationary. Since we have already checked for stationary time series in part 1a, we are going to transform the time series into different other time series and check whether the series transformed into stationary or not. Again, we are going to use the ADF test.

We are going to transform the time series into 4 other different time series with two types of transformations, namely, Log transformation and Differencing. In the first transformation, we are taking the log of all the adjusted close features of the time series and then perform first differencing on them. For the second transformation, we are just first differencing the data. For the third transformation, we are second differencing the data (i.e., we are taking the first differencing of the first difference). Similarly, for third differencing.

Below is the summarized values of these transformed time series-

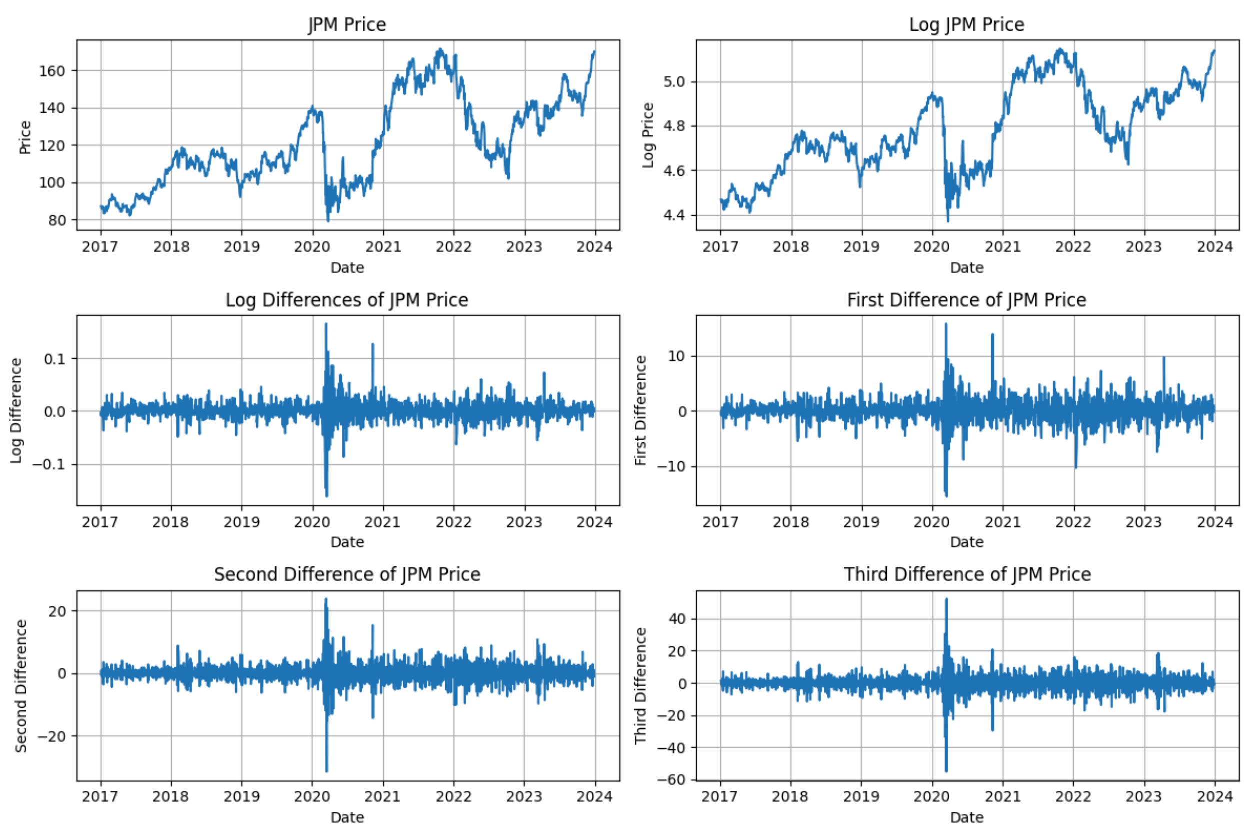


Fig 2 graphs for different transformed data a) original data b) log transformed data

c) log-difference data d) first difference data e) second difference data f) third difference data

	Original Time Series	Log Transformed	Log Difference	First Difference	Second Difference	Third Difference
count	1760.0	1760.0	1759.0	1759.0	1758.0	1757.0
mean	121.5342	4.7819	0.0004	0.0471	0.0001	-0.0004
std	23.2942	0.1912	0.0182	2.0635	3.0131	5.3387
min	79.03	4.3698	-0.1621	-15.55	-31.41	-55.18
25%	103.48	4.6394	-0.0078	-0.9	-1.4375	-2.49
50%	115.535	4.7496	0.0003	0.03	0.01	0.06
75%	139.8625	4.9407	0.0089	1.055	1.3975	2.48
max	171.78	5.1462	0.1656	15.86	23.77	52.36
count	1760.0	1760.0	1759.0	1759.0	1758.0	1757.0
mean	121.5342	4.7819	0.0004	0.0471	0.0001	-0.0004
std	23.2942	0.1912	0.0182	2.0635	3.0131	5.3387

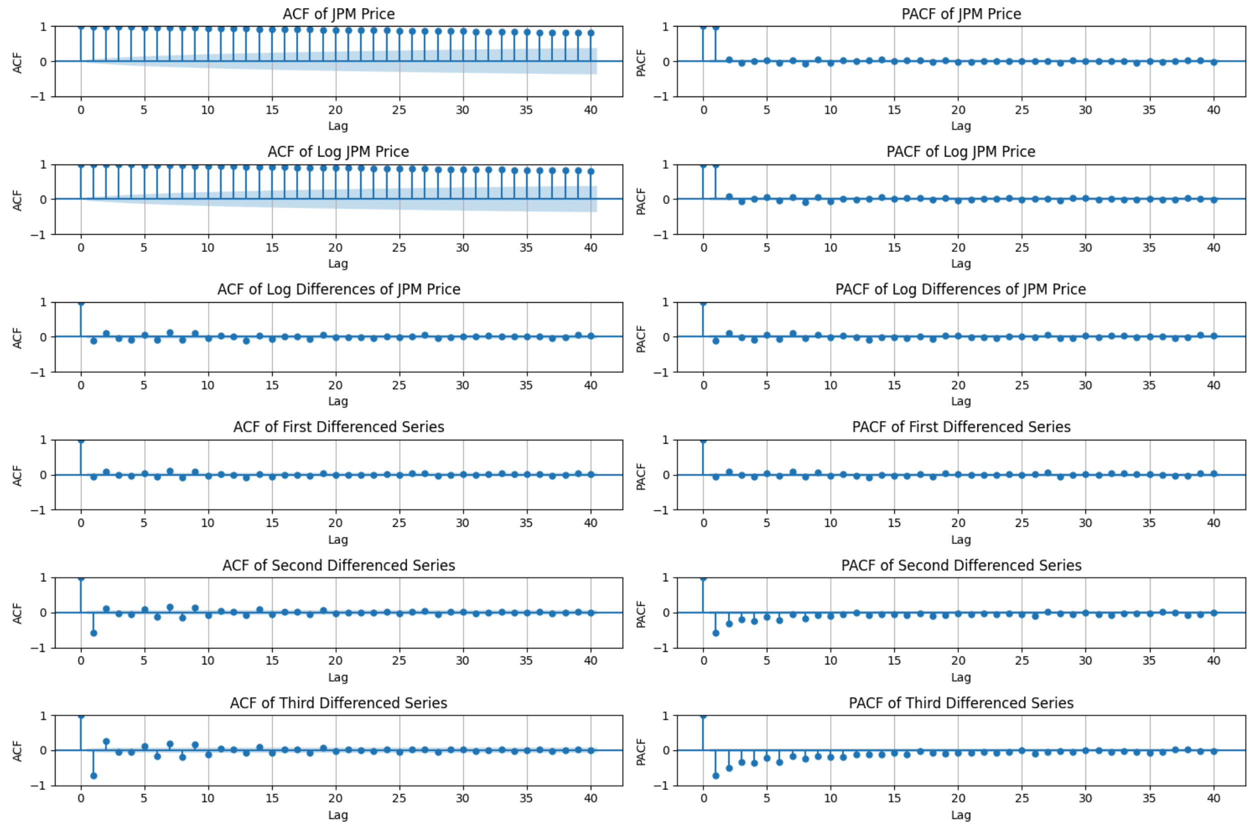
Table 1: Statistic Summary of different time series

Below are the result for ADF test on these series

Time Series	ADF Statistic	p-value	Stationary/Non-Stationary
Original	-1.5683	0.4994	non-stationary
Log transformed	-1.7262	0.4176	non-stationary
Log Difference	-10.3794	2.15823e-18	stationary
First Difference	-12.1927	1.2750e-22	stationary
Second Difference	-15.0814	8.4424e-28	stationary
Third Difference	-18.0877	2.5755e-30	stationary

Table 2: ADF test results for different time series

Below are the ACF and PACF plots for our original and transformed data-



Graph 3: ACF and PACF plots for different time series

From the above statistical summary and graphical visualization we can make a decision on which transformed time series data we are going to choose. Our choice is based on below two points-

1. First we are going to keep in mind that volatility for first, second and third difference are very high as compared to other transformation, and

2. After ADF test we can say that we are rejecting original and Log transformed time series since they are non- stationary,and

3. After ACF and PACF plots we are also rejecting the first, second and third difference time since they have very varying autocorrelation and partial autocorrelation, which in a time series is not a good property for analysis.

Now , we have chosen our time series - Log Difference Time Series for calculation of all part b.

Group Number: 7634

c)

The JPM adjusted closing was fractional differenced to produce a time series that is detrended.

Summary statistics of the differenced series:

Ticker	JPM
Count	1730
mean	11.54
std	3.51
min	-7.52
25%	9.01
50%	11.05
75%	14.14
max	22.63
Skewness	0.07

table 3

Fig 4: Plot comparison of original and fractionally differenced series.

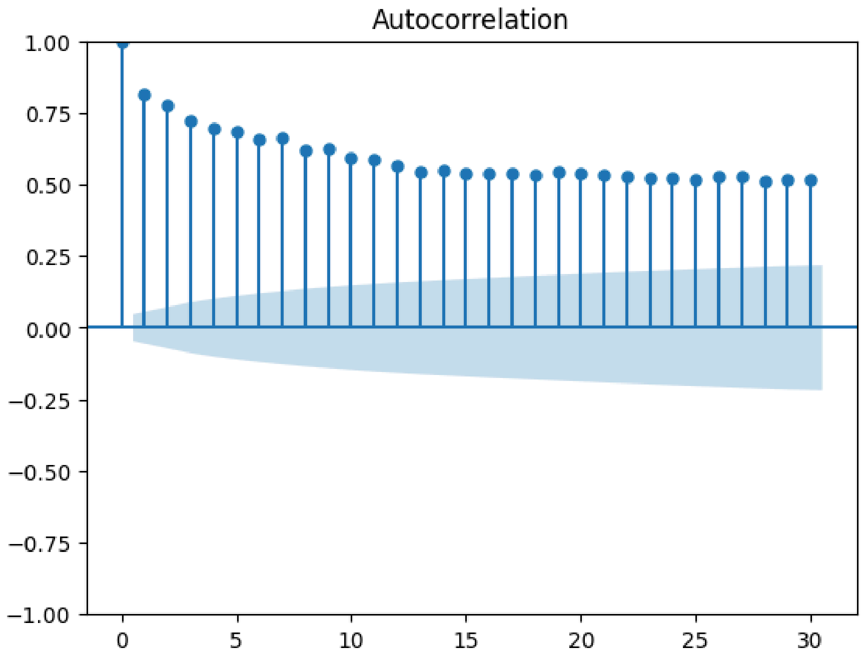


Fig 5. ACF Plot of the differenced series

ADF Statistic: -2.9136380532695325

p-value: 0.043784907475537775

Properties of fractionally differenced time series (JPM)

- Visual Inspection. The line graph comparing the original vs fractionally differenced series shows that the differenced series exhibits a mean-reverting behavior which indicates stationarity.
- ACF plot also shows that as the lag increases, autocorrelation coefficients weaken. This implies correlation weakens indicates stationary tendency of the differenced series.
- ADF test (-2.9136) further confirms stationary with p-value = 0.04.
- The Summary Statistics of the fractionally differenced series shows a mean of 11.55 with sd = 3.51, this indicates moderate variance with a skewness (0.07) that tends towards symmetry in the data distribution.

Group Number: 7634

d) As we can see in the above graphical representation of our original “JPM” stocks data with different transformation we can conclude some basic properties -

- Original data is within the 2000 units.
- Log transformation and log difference transformation have lowest magnitude.
- We can see that the original data is non- stationary by looking at the visuals, it has an upward trend also.
- Fractional differencing tries to smoothen the data but fails to do so because of the strong upward trend of original data.

Group Number: 7634

Step 2

a)

For this step, a classification MLP model was chosen. The returns of the stock price series were used. The inputs to the MLP model were the previous 25 days, 60 days, 90 days, 120 days, and 240 days returns over those respective periods. The output to be predicted is the 60 days forward return, with a "1" for positive return, and "0" for negative return.

For the train-test sample split, the ratio of 80% train - 20% test was used.

As for the structure of the MLP model, 3 hidden layers were chosen, with 25, 15 and 10 nodes respectively. A dropout rate of 20% was used. The activation function for the hidden layers is ReLu, while for the output layer, a sigmoid function was used.

b)

Now on our chosen time series, Log difference transform, we are going to perform a MLP regression model. To perform a MLP on our model we need more features, below are some ways through which we can create some features for our time series (these are common practice in time series). First we are going to add multiple features in our data and after correlation analysis we can drop the features which are highly correlated

1. Moving Average - We can use moving average as one of the features for our calculations. We are going to add a 7-day, 14-day, 30-day, 60-day, 90-day moving average.
2. Exponential Moving Average - Again we are going to add 7-day, 14-day, 30-day, 60-day, 90-day exponential moving average.
3. Rolling standard deviation (volatility) - we are going to add 7-day, 14-day, 30-day, 60-day, 90-day rolling standard deviation to our dataset. 7-day rolling standard deviation is standard deviation for past 7- days.
4. we can also use the different lags, we are using 5-day, 30-day, 60-day and 90-day lag for analysis

Now, below is the heatmap for correlation analysis for these features. This correlation analysis is based on Pearson Coefficient and which only comments on the linear relationship of these features.

Group Number: 7634

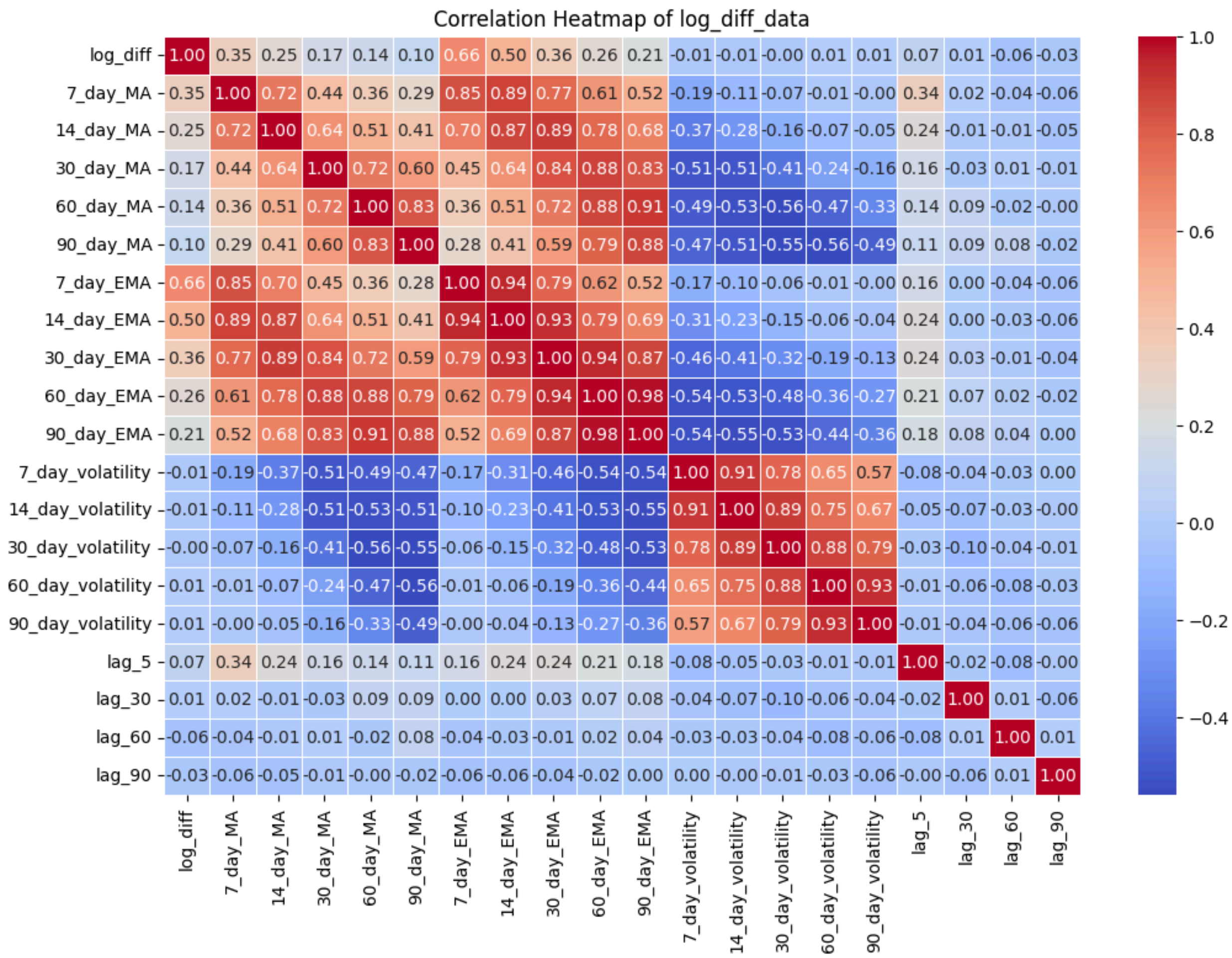


Fig 6 Heapmap of correlation matrix of log-difference time series and it new features

As we can see the Moving averages are highly correlated to each other as well as exponential and volatilities, so we are going to use only 30-day volatility , 7-day EMA and 90-day MA and all the lags

And finally we have 7 features we are going to use here.

First Model:

Now we are going to run a MLP model with 80-20 test-train split. Our first model has the below parameters and structure.

Property Type	Value
Hidden layers	2
Hidden Nodes	Each layer have 100 nodes

Activation Function	Logistic
Regularization hyperparameter	0.001
Learning rate	0.001 (constant)
Solver	Adam
Max iterations	1000

Table 4: Parameter for our base model

After running above model on our dataset we got below result-

Metric	**Value**
Mean Absolute Error (MAE)	0.010829
Mean Absolute Percentage Error (MAPE):	157.278%
R-squared (R^2):	-0.07993
Symmetric Mean Absolute Percentage Error (sMAPE):	157.352%

Table 5: Different metric result of our base model

Above is a very poor result. For example R-squared value is negative, which means a simple mean analysis is better than our MLP model. We need to fine tune our model. To do so, we are going to use Hyperparameter optimization. Below are the different parameters we are going to use to make combinations-

Parameter	**Values**
Hidden layer sizes	[(50,50,50),(100,50,25),(150,100,50,25),(10,),(15,10),(5,3)]
Activation function	['relu', 'tanh', 'logistic']
Optimizer	['adam', 'lbfgs']
Regularization term	[0.0001, 0.001, 0.01, 0.015, 0.02]
Learning rate options	['constant', 'adaptive']

Learning rate Initial Value	[0.001, 0.01, 0.005]

Table 6: Parameters for Hyperparameter Optimization

Above parameters make 1080 candidates to check for.

Best Hyperparameter combination is -

Parameter	**Values**
Hidden layer sizes	(15,10)
Activation function	'logistic'
Optimizer	'adam'
Regularization term	0.015
Learning rate options	'constant'
Learning rate Initial Value	0.01

Table 7: Best Hyper parameter combination

Above combination gives below result-

Metric	**Value**
Mean Absolute Error (MAE)	0.008079
Mean Absolute Percentage Error (MAPE):	209.3866
R-squared (R^2):	0.43300
Symmetric Mean Absolute Percentage Error (sMAPE):	109.0799

Table 8: Different metric result of our fine tubed model

Above results have room for more improvement but it is way better than our first model out R-squared have improved, out sMAPE has improved.

c)

The regression model was used for part C, and the fractionally differenced time series were used as input to train an MLP model to predict the fractionally differenced value of the series.

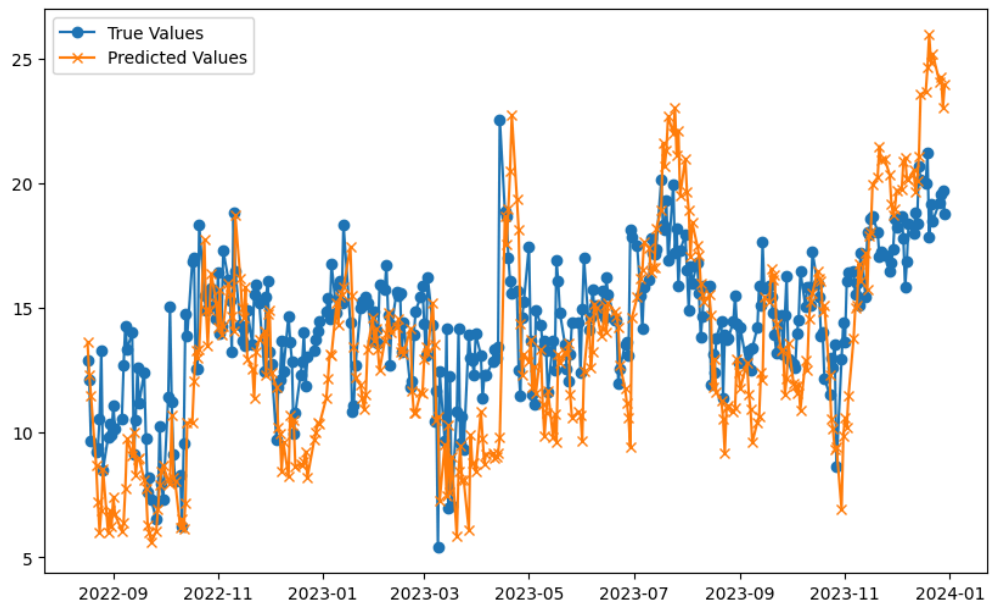


Fig 7. Actual Vs Predicted

Model MSE on test data: 9.944298290542017

Model MAE on test data: 2.528113915757389

The model MSE and MAE indicate a good predictability fractionally differenced series of the JPM, however there is still room for further improvements through hyperparameter tuning, feature engineering, other advanced models, etc.

d)

As a classification model was used in part (a), the metric used to measure the model's performance was the accuracy, with a value of 54.28%.

We cannot compare the models in part b and c directly because both are different types of regression but we can create a general rough idea that how much our model can explain and predict the data.

MLP on log-difference data made a very good predictive model, as we can see in the result table of fine tune model that our MSE is 0.008 and our data range is [-0.16, 0,16], we can say our error is minimal and also out error is less than standard deviation, which is very good for a model like MLP.

Now by value of R-Squared we can see that our model can only explain 43.33% of data by these 7 features which is not impressive but given that there is no other data series and model for comparison, and solely depending on a single time series, it is acceptable and can be used as a basic model for comparison with future enhanced models.

As the classification model is used in part a, it is perhaps quite difficult to compare the results between part a with parts b and c, as they are of a different basis.

Step 3

a)

For this step, the JPM stock price series was converted to image representations using the Gramian Angular Field (GAF) algorithm, and then used as inputs into Convolutional Neural Network (CNN) to make prediction of whether the stock price will go up or down (classification problem) in a forward period.

A window size of 30 days was chosen for the time series before its GAF transformation. For the output labels, a '1' is given if the price went up, and '0' if the price dropped. The prediction is for the 30th day forward.

As for the structure of the CNN model, there are three convolutional layers used, each paired with a Max Pooling layer. The number of output channels of the convolutional layers increases from 16 to 32 to 64, to try to capture more features of the input data, before being connected to a fully connected layer with 1024 hidden representations. The ReLu activation function was used for the layers, except for the output layer, where a sigmoid function was used.

b)

In step 2b we have created a MLP model and with the help of hyperparameter tuning we got the best combination of hyperparameters. Now in this part we are going to use the CNN model and obtain their image representation using Gramian Angular Field (GAF).to see if we can further improve our model. The process will be -

- First, we are preprocessing the data and making sure that out data is within the scale range of [-1,1]
- Then we are going to generate the GAF image for each feature and target value .
- Now we are going to use the GAF image as input value for our CNN model.
- Now we are going to train the CNN model.

Our CNN model have below parameters-

Layer (type)	Output shape	Number of Parameter
Conv2D	(None,7,7,32)	320
MaxPooling2D	(None, 4, 4, 32)	0
Dropout	(None, 4, 4, 32)	0
Conv2D	(None, 4, 4, 64)	18496
MaxPooling2D	(None, 4, 4, 64)	0
Dropout	(None, 4, 4, 64)	0
Flatten	(None, 256)	0
Dense	(None, 128)	32896

Dropout	(None, 128)	0
Dense	(None, 1)	129

Table 9: Model structure of our CNN model

After running above model on our transformed data we got below results-

Metric	Value
Mean Absolute Error (MAE)	0.00889
Mean Squared Error (MSE):	0.000143
R-squared (R^2):	0.2974

Table 10: Metric result for our CNN model

As we can see our CNN model performs worse than MLP model, we can change some parameters and check for further results. But for a better model we need some more features which can explain the behavior of our time series.

c)

The CNN model indicates a good predictive power with Mean squared error of 0.39 and and a mean absolute error of 0.54.

d)

For part a of this step, since it is again a classification problem, the accuracy metric was used, and it has a value of 62.65%, which is an improvement over the MLP model in Step 2a.

CNN model based on GAF image of log -differenced data in part b does not perform well. It perform very bad as compared to our MLP model in part b, The main culprit for this performance is the number of features that we have used for training the model. We have only used the 7 features. We could have used more features if we want but all of them are very highly correlated to each other linearly. And we know the features are correlated then they are good for CNN but not in our case since our data is time series data. CNNs do not make good models for time series data predictions. They can be good for classification prediction of time series but not the value of time series. That’s exactly what we are seeing in the part a in this step. CNN performs better when we are classifying the time series prediction.

Group Number: 7634

Step 4

A comparison of the accuracy results for parts (a) of Steps 2 and 3 is shown below.

	MLP model	CNN model
Accuracy (%)	54.28	62.65

Based on the results in the table above, it looks like the CNN model performed better in terms of accuracy, compared to the MLP model. This is perhaps a bit surprising, given that CNN is pattern location invariant, and is thought to be less effective than the MLP when it comes to analyzing time series models. Perhaps, the fact that CNN is location invariant is useful for this particular time series due to its common patterns and predictability? A further step to check this would be to try out the time series of some other companies to see if they share similar performance.

Another possible reason is that there might have been overfitting during training in the MLP model.

The third possible reason for the difference is the absence of hyperparameter tuning for both the MLP and CNN models. For part a of Steps 2 and 3, the hyperparameters were chosen without any tuning done beforehand.

Now, the CNN in part b doesn't perform as we thought. CNN are complicated and highly dense models which should give high accuracy but for that they need high amount of data with high number of features to make prediction. In our case in part b we are using a low number of features to reduce the computational power and memory which was why they performed poor . If we can increase the features for predictions we can manage to take the error low. Again with only one time series with no other dependent series or data we can't go very high in prediction. In next task we are going to try to increase the dependencies and features for the CNN model.

For parts (c) , of the Step 2 and 3, the CNN still outperforms the MLP in terms of better prediction - , achieving lower MSE (0.39 vs. 9.94) and MAE (0.54 vs. 2.53). The likely cause of this is that the CNN model has strength to perform in GAF transformed time series where there is existence of local dependencies and spatial data. On the other hand, MLPs depend on global dependencies which may fail to learn from complex patterns.

References

1. Zhang, Ashton, et al. Dive into Deep Learning. https://d2l.ai/d2l-en.pdf