FULL LEGAL NAME	LOCATION (COUNTRY)	EMAIL ADDRESS	MARK X FOR ANY NON-CONTRIBUTING MEMBER
CHIEDOZIE AUGUSTINE IKE-OFFIAH	NORTH CYPRUS (TURKEY)	chiexaustine2005@gmail.com	
ARIFIN LIAUW	INDONESIA	james.arifin@gmail.com	
SHIVANSH KUMAR	INDIA	shivansh.business23@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	**CHIEDOZIE AUGUSTINE IKE-OFFIAH**
Team member 2	**ARIFIN LIAUW**
Team member 3	**SHIVANSH KUMAR**

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.
N/A

Step 1

In this GWP we will build a Deep Learning model that tries to predict short-term market trends across different asset classes.

- We commenced our process by importing all necessary Python libraries as the first step, followed by retrieving all data of 5 different ETF's SPY, TLT, SHY, GLD, DBO from January 1st, 2018 to December 30th, 2022. We clean for missing data and conduct the necessary formatting process.
- Once we have cleaned data, we conduct Exploratory Data Analysis and Figure 1 shows the summary of the analysis.

Summary Statistics:

Ticker	SPY	TLT	SHY	GLD	DBO
count	1258.000000	1258.000000	1258.000000	1258.000000	1258.000000
mean	344.308625	134.630079	84.553426	150.921431	11.588172
std	66.613497	18.284340	1.595424	22.501465	3.341802
min	222.949997	92.400002	80.589996	111.099998	5.200000
25%	283.965012	119.635002	83.269997	125.865000	9.530000
50%	326.759995	136.504997	84.620003	159.434998	10.915000
75%	402.577507	148.054996	86.250000	169.345005	13.252500
max	477.709991	171.570007	86.800003	193.889999	21.080000

Figure 1: Exploratory Data Analysis

- From there, we visualize the ETF's price in one plot as shown in Figure 2.

Figure 2: Visualization of 5 ETFs' Closing Price

- Based on the closing price data, we convert the data to calculate the return for each portfolio to standardize in the evaluation process to make an apple-to-apple comparison. Figure 3 shows each portfolio histogram distribution. From the histogram distribution plot on the daily returns for each portfolio, we could see that the distribution follows na ormal distribution with a slight right (SPY and GLD) or left skewness (TLT). The biggest return range is on SPY.

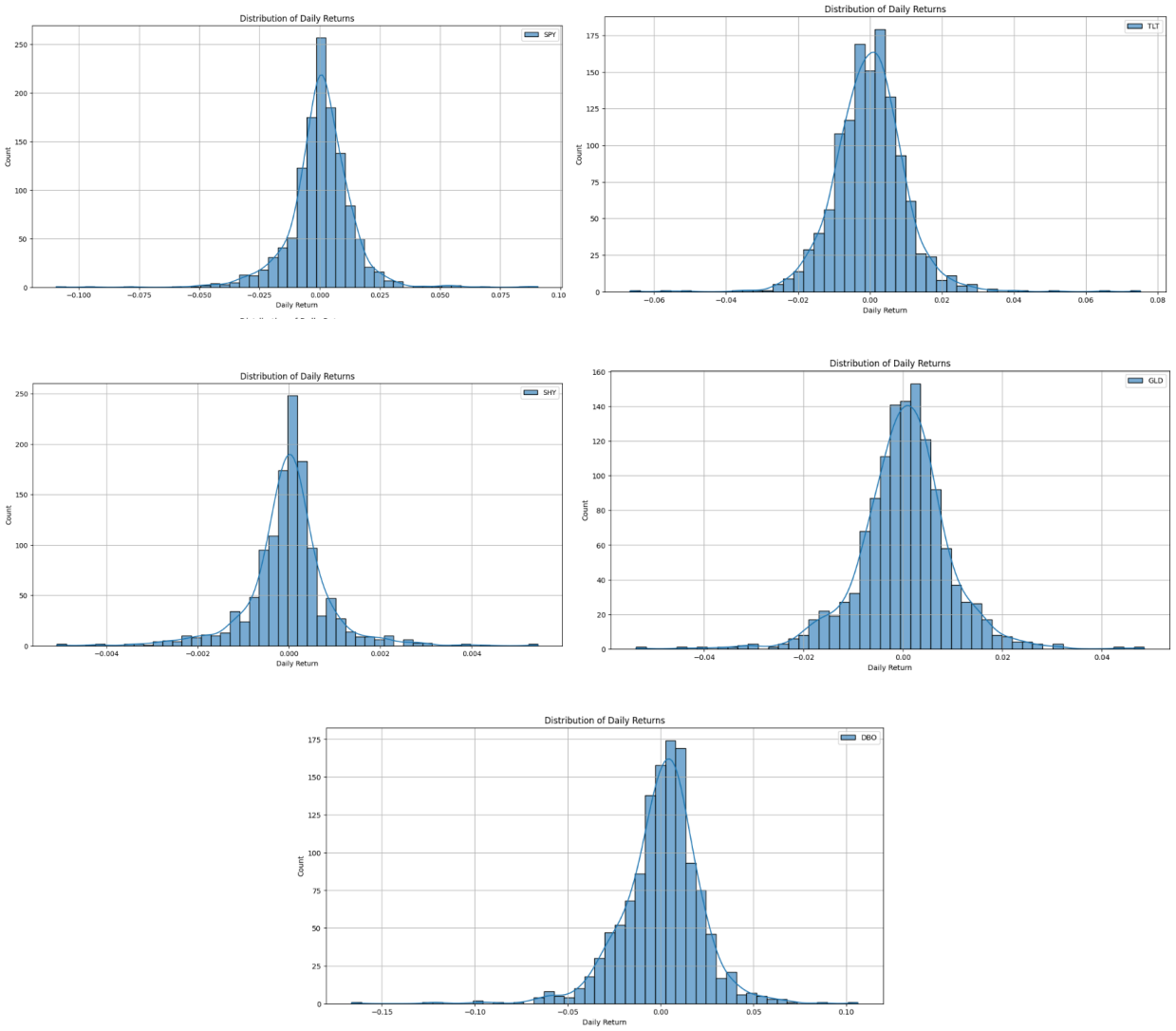

Figure 3: Histogram Distribution fo 5 ETFs

- Based on the correlation matrix, Figure 4, we could see that SPY had a negative correlation with Long Term Bond TLT (-0.25) and Short Term Bond SHY (-0.08) with weak positive correlation with Gold GLD and have some co-movement with Oil DBO (+0.36). From this correlation matrix, we can say that among the 5 ETFs, there is a potential for portfolio combination for Portfolio

Management and Risk Management while still capturing the upside. Long Term Bond and Short Term Bond can act as cushion when the equity market faces a crash, while gold is still the best safe haven.

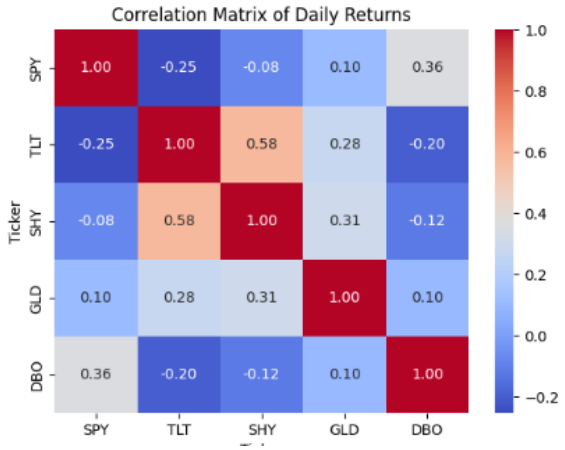

Figure 4: Correlation Matrix

- Upon checking the stationary of the calculated return, using the Augmented Fuller-Dickey (ADF), we conclude that the return is non-stationary

Asset	ADF Statistic	p-value	Stationarity Conclusion
SPY	-1.2506	0.6515	Non-stationary
TLT	-0.4058	0.9091	Non-stationary
SHY	-0.0643	0.9529	Non-stationary
GLD	-1.1950	0.6757	Non-stationary
DBO	-1.0006	0.7531	Non-stationary

Figure 5. ADF on Return and Conclusion

- We use log return to remove non-stationarity, with Figure 6 representing the visualization of Stationary.

Figure 6. Stationary Log Return

- Figure 7 shows the statistical summary for the log return. In terms of volatility, DBO is the most volatile asset, while the lowest volatile asset is SHY, which is quite natural considering its purpose. Looking at a mean daily return, SPY, GLD, and DBO give a positive return, showing that such asset is a growth asset while TLT and SHY have a negative return, indicating this asset serves its purpose for stability and hedging in situations where interest rate hike. DBO and SPY have the highest range indicating these two assets are high risk assets.

Statistic	SPY	TLT	SHY	GLD	DBO
Count	1257.000000	1257.000000	1257.000000	1257.000000	1257.000000
Mean	0.000283	-0.000175	-0.000025	0.000238	0.000301
Std	0.013687	0.010159	0.000918	0.009057	0.021876
Min	-0.115887	-0.069010	-0.005101	-0.055190	-0.182065
25%	-0.005266	-0.005884	-0.000354	-0.004336	-0.009655
50%	0.000727	0.000000	0.000000	0.000518	0.002411
75%	0.007088	0.005495	0.000351	0.005080	0.012613
Max	0.086731	0.072503	0.005438	0.047390	0.100955

Figure 7. Summary Statistic for Log Return of 5 ETFs

- Now we will do ACF plots of our return to check for stationarity, and below you guys can see the ACF plots suggest that the data is stationary. Figure 8 represents the plots. The sequence of the plots are SPY, TLT, SHY, GLD and DBO, respectively from left to right and top to bottom.

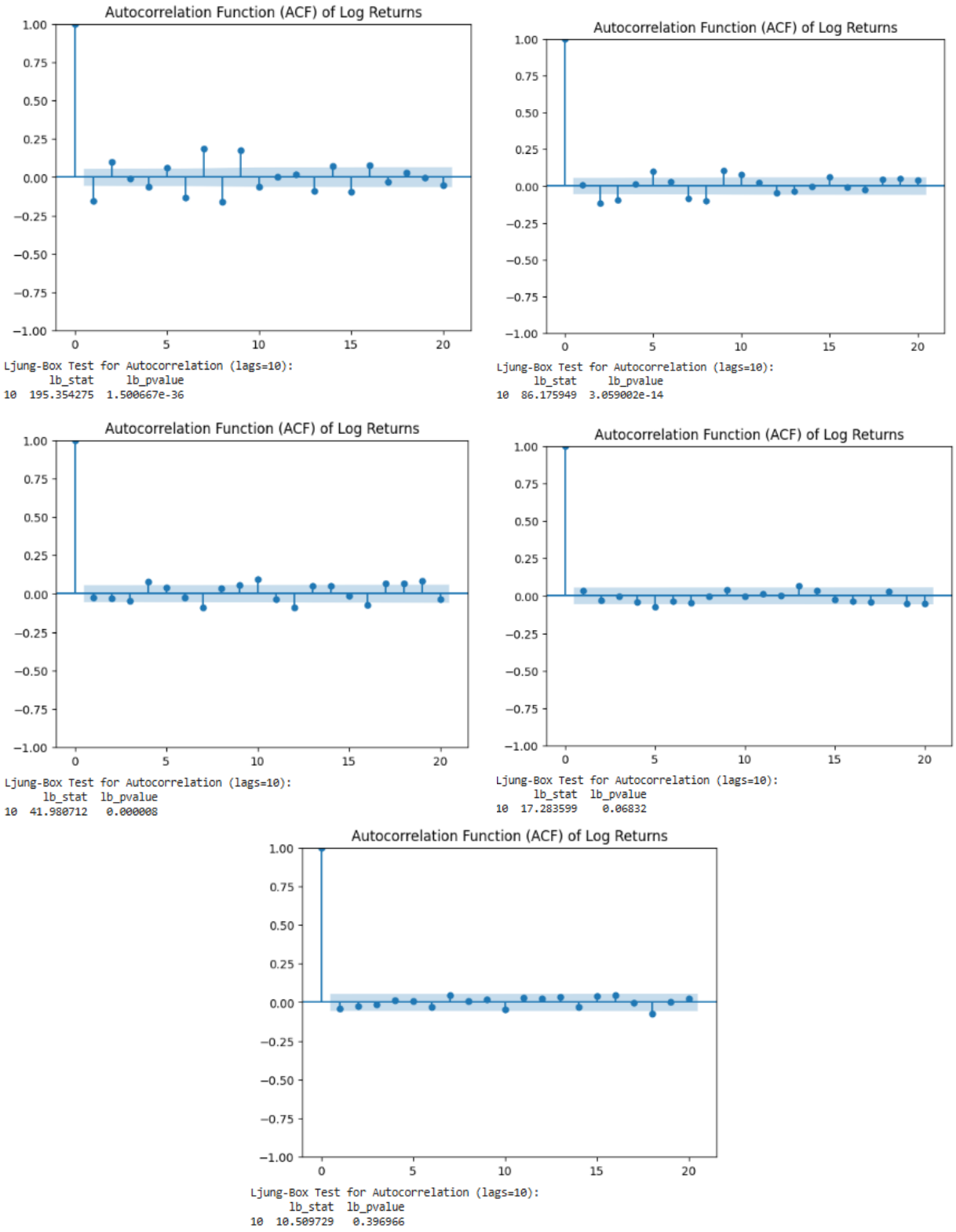

Figure 8. ACF Plot of 5 ETFs

- Then, we return to ADF on log return, and the result shows that the data is stationary. Figure 9 shows the summary of the results.

Asset	ADF Statistic	p-value	Stationarity Conclusion
SPY	-10.8025	0.0000	Stationary
TLT	-10.6434	0.0000	Stationary
SHY	-6.6507	0.0000	Stationary
GLD	-15.4176	0.0000	Stationary
DBO	-36.9405	0.0000	Stationary

Figure 9: ADF on Log Return and Conclusion

Step 2

---

a. Now In this step, we will build DL models using RNN. Here's how we are going to do it:

- We will first build a DL model that uses LSTM network architecture for each of the 5 different asset classes to predict the 25-day ahead return of each ETF.
- After that, we will train each of the 5 models and perform an in-sample predictive performance test. We will also test each model out-of-sample and compare the results of both across the assets.
- Then, we will develop a trading strategy that uses the out-of-sample prediction of all the different models and backtest it to check our predictive performance.

Now, as we are going to use the same data structure and LSTM architecture, we are going to define our model workflow in one go here; it will be applicable for all 5 respective ETF models:

- First, we will prepare the data for each model, as we have to build individual models for each ETF; in this, we will copy only the required ETF that is needed for that model.

- Then we will prepare the data for input by doing some feature engineering in which we will calculate 1-day, 10-day, and 50-day cumulative returns of respective ETFs to predict the 25-day ahead return; you can see this in Figure 10.

Ticker	Date	SPY	Ret_10	Ret_50	Ret25
49	2018-03-15	-0.001090	0.000269	0.000229	-0.000310
50	2018-03-16	-0.002913	0.000188	0.000137	-0.000282
51	2018-03-19	-0.013623	-0.000063	-0.000042	-0.000282
52	2018-03-20	0.001699	-0.000071	-0.000091	-0.000274
53	2018-03-21	-0.001921	-0.000087	-0.000129	-0.000154

Figure 10: Prepared Dataset to Use as Input for SPY ETF Prediction

- After that, we will define our data into a train, test, and validation set; we will have a window, i.e., a sequence length of 30, as we are using LSTM architecture.
- In the next step, we will scale the different variables under the min-max scalar to ease our computation, and then finally we will split the data into a train and test split. You can see the shape of the split in figure 11.

```
Shape of training data (709, 30, 3) (709, 1)
Shape of test data (414, 30, 3) (444, 1)
```

Figure 11: Share of Train and Test Data Split for SPY ETF

- Now we will finally discuss our LSTM architecture; we will define how the architecture is working : (Note: This architecture will be the same for all the LSTM models for each ETF.)
- Our model architecture consists of a sequential LSTM network, which we have designed for regression. It starts with stacking three LSTM layers, each containing 50 units, and utilizing the tanh activation function, with L2 regularization (L2=0.01) which we have applied to the kernel weights to prevent overfitting. The first two LSTM layers return sequences, allowing the propagation of sequential information through the network, while the third LSTM layer does not

return sequences, outputting only the final hidden state. Each LSTM layer is followed by a dropout layer with a dropout rate of 0.2, ensuring robustness and generalization. After the LSTM layers, a dense layer with 20 units and ReLU activation is added, followed by another dropout layer. The network ends with an output layer comprising a single neuron with a linear activation function, suitable for regression predictions. The model is compiled with the Adam optimizer. using a learning rate of 1e-4, and the loss function is a mean absolute error (MAE). Early stopping is also included to monitor the validation loss, with a patience of 5 epochs and the ability to restore the best-performing weights. At last, we can see the model summary with all the layers and parameters below in figure 12.

Layer (type)	Output Shape	Param #
lstm_25 (LSTM)	(None, 30, 50)	10,800
dropout_32 (Dropout)	(None, 30, 50)	0
lstm_26 (LSTM)	(None, 30, 50)	20,200
dropout_33 (Dropout)	(None, 30, 50)	0
lstm_27 (LSTM)	(None, 50)	20,200
dropout_34 (Dropout)	(None, 50)	0
dense_16 (Dense)	(None, 20)	1,020
dropout_35 (Dropout)	(None, 20)	0
dense_17 (Dense)	(None, 1)	21

Total params: 52,241 (204.07 KB)
Trainable params: 52,241 (204.07 KB)
Non-trainable params: 0 (0.00 B)

Figure 12: Model Summary

- After that, we will define the final features (learning rate, loss function, early stopping criteria, batch size, etc.) and train the model. After training, we need to do the job of testing and to find out how our model is performing, so we will train the model, and to check the performance of our model, we will do an out-of-sample prediction using the standard out-of-sample R-squared measured and check the results.

Now we will briefly discuss the training and testing performance of the model we built.

b. From model fit, we can summarize the training loss and validation loss for each asset, SPY, TLT, SHY, GLD, and DBO. Figure 13 shows the training loss and validation loss for each asset at the beginning (Epoch 1) and at the end (Epoch 100). Based on this figure, we could see all the training losses improve significantly from the beginning towards the end of training, similar to the validation loss. However, if we scrutinize, there is a slight difference in behavior between training loss and validation loss among the assets. SPY and SHY are the best performers, as the model shows a strong convergence and generalization. The validation loss tracks training loss. closely suggests high reliability. Meanwhile, for assets of TLT, GLD, and DBO, although there is a significant improvement for training loss, in terms of validation loss, they exhibit a sign of overfitting or plateauing validation loss, especially for the last 10 epochs. For GLD, the final validation loss is still higher than the training loss, indicating modern overfitting.

We could improve the model for TLT, GLD, and DBO by enhancing it with stronger regularization or loosening the earlier stop criteria.

Asset	Loss	Epoch 1	Epoch 100
SPY	Training	1.8448	0.4036
	Validation	1.7684	0.3565
TLT	Training	1.9281	0.4869
	Validation	2.095	0.5938
SHY	Training	2.1508	0.5524
	Validation	2.2134	0.4736
GLD	Training	1.9446	0.5223
	Validation	2.0915	0.5767
DBO	Training	1.9192	0.5108
	Validation	1.9693	0.5741

Figure 13: Summary of Training Loss and Validation Loss for Asset

Based on the result of out-of-sample R-squared, Figure 14 shows the summary for each asset as follows:

Asset	R^2_{OOS}
SPY	-0.095
TLT	-0.127
SHY	0.068
GLD	-0.051
DBO	-0.179

Figure 14: Summary of Out-of-Sample R-squared.

Based on the summary above, we can conclude that the LSTM model prediction based on the out-of-sample R-square is worse than the prediction based on mean return as the baseline, indicating the failure of the model to learn the temporal patterns. The LSTM model has the worst performance when applied to oil (DBO) assets. In general, we could say that Pred LSTM shows limited variation, especially of SPY, TLT, GLD, and DBO considering the actual is pretty variable or fluctuates. The reason the LSTM model can still perform positively on SHY is because actual SHY assets fluctuate less. In this regard, we consider the LSTM model is either underfitting or has poor learning to capture the underlying pattern in the fluctuation of return in SPY, TLT, GLD and DBO. The worst performance of the model on DBO assets indicates that the model fails to capture the direction and magnitude of the price movement, as the predicted LSTM result (negative return with small fluctuation) contradicts the actual return (positive returns, large variability).

We could improve the model performance by the following:

- Model refinement by reassessing hyperparameters such as learning rate, window size or number of neurons
- Tuning and increasing training data
- Increase data richness or adjust feature engineering to improve pattern recognition
- Use other Deep Learning architecture or regularization methods to reduce model underfitting

d. Trading Strategy and Backtesting Model:

Consequently, we are going to back-test our model prediction on the test data in our data set. The period of the backtest strategy is the period explained by our test data. The trading strategy we implement is as follows:

I. Every 25 days, we rebalance our strategy by trading long on the best two ETFs, which are predicted to do best, and trading short on the best two ETFs, which are predicted to do worse. It is important to remember that we use the results of the test predictions (out-of-sample predictions) to carry this out.

Ii. We incorporate all five asset classes in our portfolio to make this prediction as described in (I) above. By implementing this trading strategy, with our predicted results, we get the plot represented in Figure 15. In the figure, we can see that our strategy does not outperform the buy-and-hold strategy. It even performs worse compared to an equal-weighted portfolio (represented by the orange line). In essence, the strategy's asset allocation and rebalancing strategy failed to capture the market trends effectively. This may be due to a host of reasons. One of these major reasons could be the effectiveness of the prediction of our LSTM model. Let us have an insight into the predictive power of our test results.

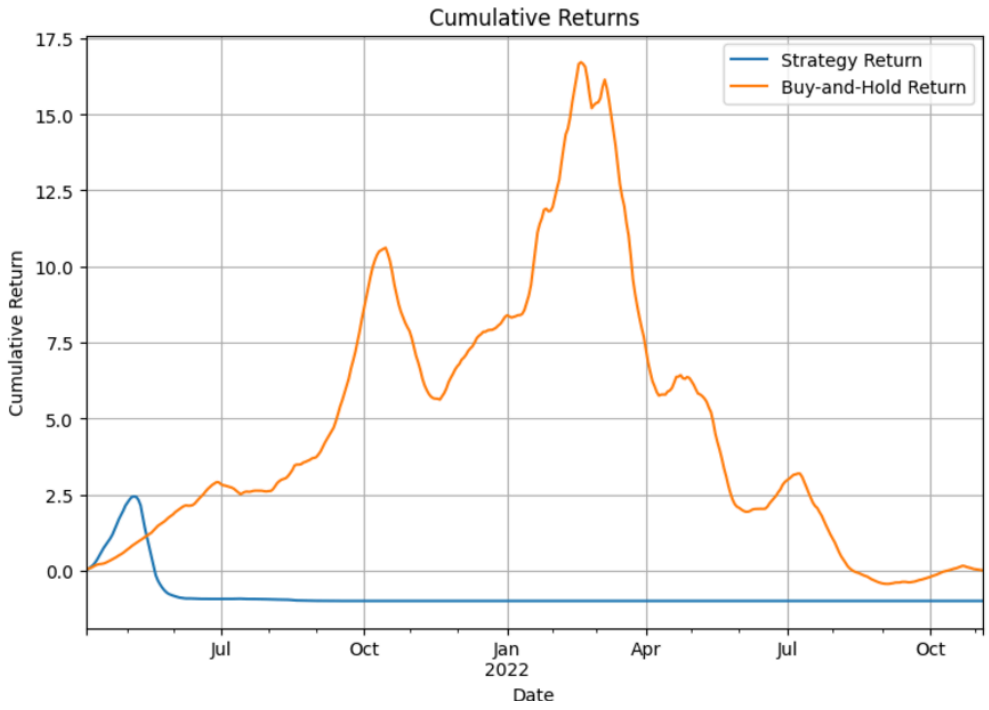

Figure 15: Cumulative returns of the buy-and-hold strategy and the long-and-short strategy for single input, single output predictions.

A quick look at the scatter plots for our predictions presented in Figure 15b, which is the scatter plot between the actual results and the predicted results, shows a scatter plot that is not uniform. Another reason for the underperforming strategy return may be due to the frequency of rebalancing (Bernoussi and Rockinger). The frequency of rebalancing used in this work is 25 days. This could lead to comparatively high transaction costs, which could render our trading strategy ineffective. Another

possible reason for the ineffectiveness of our method may be due to the high volatility of the market for the assets in the portfolio. If we take a careful look at Figure 2, we immediately notice that the ETF 'SPY' is very volatile when compared to other assets. This could have led the trading strategy we implemented to focus more on the movement of SPY instead of also considering all the other assets. This could consequently introduce unwanted noise into our model. To improve our model, we can do a number of things; for one, we could investigate further to understand the underlying market conditions that may have affected our strategy method and act accordingly. We may proceed to use other machine learning techniques to improve prediction, or better still, tweak more of the parameters in our existing machine learning method (LSTM) to improve predictions.

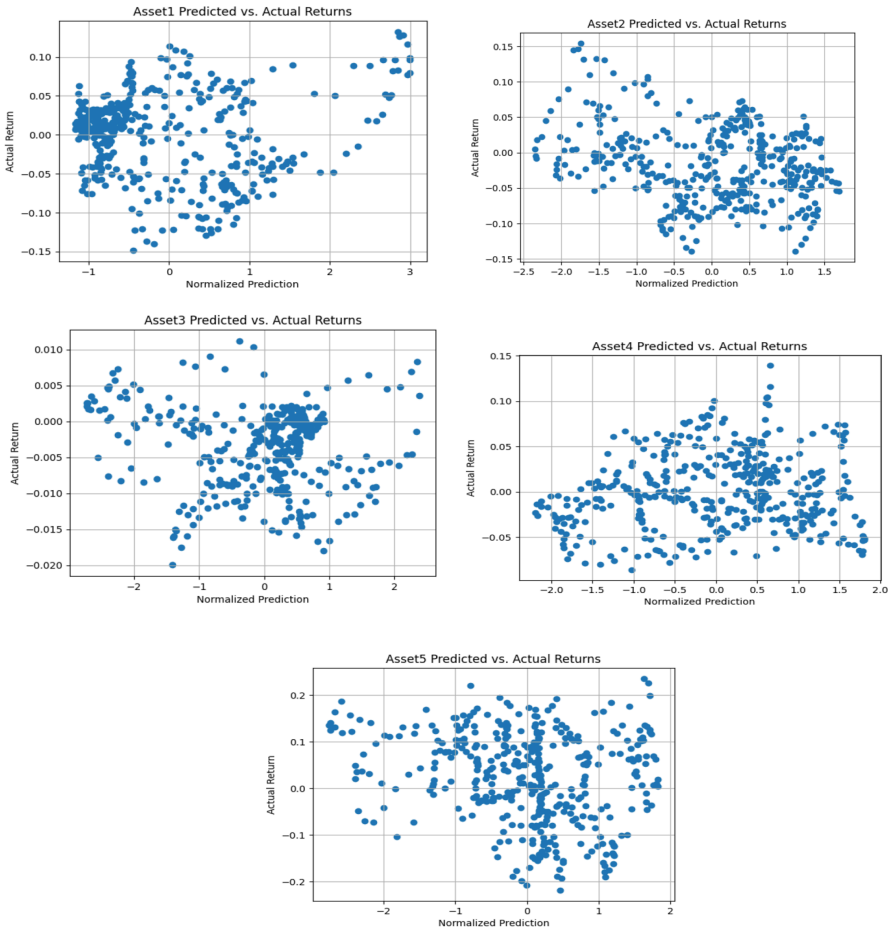

Figure 15b: Scatter plots for the predicted asset returns and the actual asset returns for the five exchange-traded funds.

Step 3

---

a. Now, In this step we will also build a DL models but this time of multi-output, here how we are going to do :

- We will first build a DL model with multi-output that uses LSTM network architecture that blends together all of the information from the 5 models of Step 2, to predict the 25-day ahead return of each ETF.
- After that we will Train the single model with 5 outputs and do an in-sample predictive performance and also Test each of the models out of sample and compare the results of both across the assets.
- Then we will use a trading strategy that uses the out-of-sample prediction of all the different models and the backtest it checks our predictive performance.

We have to build a single model for this step but the model will be multi-output model, here we are going to use Multi-output Bidirectional LSTM architecture, here how the workflow goes :

- First we will prepare the data, we will concatenate all the ETF data together in a single dataset.
- Then we will prepare the data for input by doing the same feature engineering in which we will calculate 1-day, 10-days, and 50-day cumulative returns of all the ETF at one go to predict the 25-day ahead return, you can see the complete dataset below in Figure 16.

Ticker	Date	Ret_10_SPY	Ret_50_SPY	Ret25_SPY	Date	Ret_10_TLT	Ret_50_TLT	Ret25_TLT	Date	Ret_10_SHY	Ret_50_SHY	Ret25_SHY	Date	Ret_10_GLD	Ret_50_GLD
49	2018-03-15	0.000269	0.000229	-0.000310	2018-03-15	0.000077	-0.000427	-0.000149	2018-03-15	-0.000002	-0.000044	-0.000023	2018-03-15	0.000014	-0.000020
50	2018-03-16	0.000188	0.000137	-0.000282	2018-03-16	0.000123	-0.000511	-0.000111	2018-03-16	-0.000002	-0.000049	-0.000019	2018-03-16	-0.000063	-0.000018
51	2018-03-19	-0.000063	-0.000042	-0.000282	2018-03-19	0.000118	-0.000541	-0.000125	2018-03-19	-0.000002	-0.000044	-0.000017	2018-03-19	-0.000025	-0.000047
52	2018-03-20	-0.000071	-0.000091	-0.000274	2018-03-20	0.000068	-0.000553	-0.000152	2018-03-20	-0.000004	-0.000048	-0.000014	2018-03-20	-0.000177	-0.000082
53	2018-03-21	-0.000087	-0.000129	-0.000154	2018-03-21	0.000090	-0.000535	-0.000093	2018-03-21	0.000004	-0.000041	-0.000019	2018-03-21	0.000060	0.000093
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Figure 16: Combined prepared Data for using All ETF returns

- After that we will define our data into feature, train, test and validation set, also we will also have a window i.e sequence length of 30 as we are using LSTM architecture.

- In the next step we will scale the different variables under the min max scalar to ease our computation then finally we will split the data in train and test split, you can see the shapes in Figure 17 .

```
Training data shapes -> X_train: (798, 30, 10) , y_train: (798, 5)
Test data shapes -> X_test: (325, 30, 10) , y_test: (325, 5)
```

Figure 17 : Shape of Training and Testing Split

- Now finally we will define our LSTM model architecture. This time we are going to do things a little bit differently; we are going to use a bidirectional ("Keras documentation: Bidirectional layer") (Khaled A. Althelaya, Department of Information and Computer Science, Engineering King Fahd University of Petroleum and Minerals, Dhahran, Kingdom of Saudi Arabia) LSTM model for our prediction. Here's how our model works:
    - This time also our model architecture consists of an LSTM model designed for a regression task, predicting continuous values for 5 different ETFs. It begins with four bidirectional LSTM layers, each containing 50 units and utilizing the same tanh activation function, with L2 regularization (L2=0.01) applied to the kernel weights to prevent overfitting. The first three bidirectional LSTM layers return sequences, allowing the propagation of sequential information through the network, while the fourth Bidirectional LSTM layer does not return sequences, outputting only the final hidden state. Each Bidirectional LSTM layer is followed by a dropout layer with a dropout rate of 0.2, ensuring robustness and generalization. After the LSTM layers, the model includes two dense layers with 20 and 10 units, respectively, both using ReLU activation, followed by dropout layers. The network finishes with an output layer comprising 5 neurons with a linear activation function suitable for regression predictions. The model is compiled with the Adam optimizer, using a learning rate of 1e-4, and the loss function is mean squared error (MSE). Early stopping is employed to monitor the validation loss with a patience of 5 epochs and the ability to restore the best-performing weights. At last we can see the model summary with all the layers and params below in Figure 18.

Model: "sequential_12"

Layer (type)	Output Shape	Param #
bidirectional_10 (Bidirectional)	(None, 30, 100)	24,400
bidirectional_11 (Bidirectional)	(None, 30, 100)	60,400
dropout_52 (Dropout)	(None, 30, 100)	0
bidirectional_12 (Bidirectional)	(None, 30, 100)	60,400
dropout_53 (Dropout)	(None, 30, 100)	0
bidirectional_13 (Bidirectional)	(None, 30, 100)	60,400
dropout_54 (Dropout)	(None, 30, 100)	0
bidirectional_14 (Bidirectional)	(None, 100)	60,400
dropout_55 (Dropout)	(None, 100)	0
dense_26 (Dense)	(None, 20)	2,020
dropout_56 (Dropout)	(None, 20)	0
dense_27 (Dense)	(None, 10)	210
dropout_57 (Dropout)	(None, 10)	0
dense_28 (Dense)	(None, 5)	55

Total params: 804,857 (3.07 MB)
Trainable params: 268,285 (1.02 MB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 536,572 (2.05 MB)

Figure 18 : Bidirectional Multi-ouput LSTM Architecture Model

- After that like Step 2 we will again define final features (learning rate, loss function, early stopping criteria, batch size, etc) and train the model, and after training we need to do the job of testing and to find how our model is performing so, we will train the model and to check the performance of our model we will do an out of sample prediction using the standard out-of-sample R-squared measured and check the results.

b. From the result of the multi-output training and validation process, we got an initial training loss of 11.0779 and validation loss of 10.9634, while at the final epoch, the final training loss became 1.1351 and validation loss became 1.2814.

The multi-output model effectively learns and generalizes,, achieving significant loss reductions across training and validation. While slightly less precise than the best-performing single-output models like SPY and SHY, it provides an efficient, unified approach for multi-asset prediction with robust performance.

Based on the result of the out-of-sample R-squared of Bidirectional multi-output model, Figure 19 shown the summary for each asset as follows:

Asset	R^2_{OOS}
SPY	-0.218
TLT	-0.359
SHY	-1.018
GLD	0.047
DBO	-1.329

Figure 19. Summary of Out-of-sample R-squared.

Based on the summary above, we can conclude that the LSTM model with multi-output prediction based on the out-of-sample R-square is worse than the prediction based on mean return as the baseline, indicating failure of the model to learn the temporal patterns. The model works slightly better on Gold, GLD, as shown by positive out-of-sample R-square. The worst performance is on Oil, DBO, indicating the multi-output model is extremely underfitting or has a huge misalignment with actual return. Therefore, we can conclude that the multi-output model is not improving the model and the model still faces underfitting showing by its poor performance to make predictions.

We could improve the model performance by following:

- Model refinement by revisit the architecture and hyperparameters such as number of layers, unit, learning rate
- Increase data richness or adjust feature engineering to improve pattern recognition such as fundamental information, macro or micro information
- Use other models that fit for multi-output.

d. **Trading and Backtest Strategy**

Just like we did in 2d, we carry out a backtesting strategy using the predictions we obtain from implementing a multi-output LSTM model on all five ETF assets. Using the predictions, we obtain the figure as obtained in Figure 20.

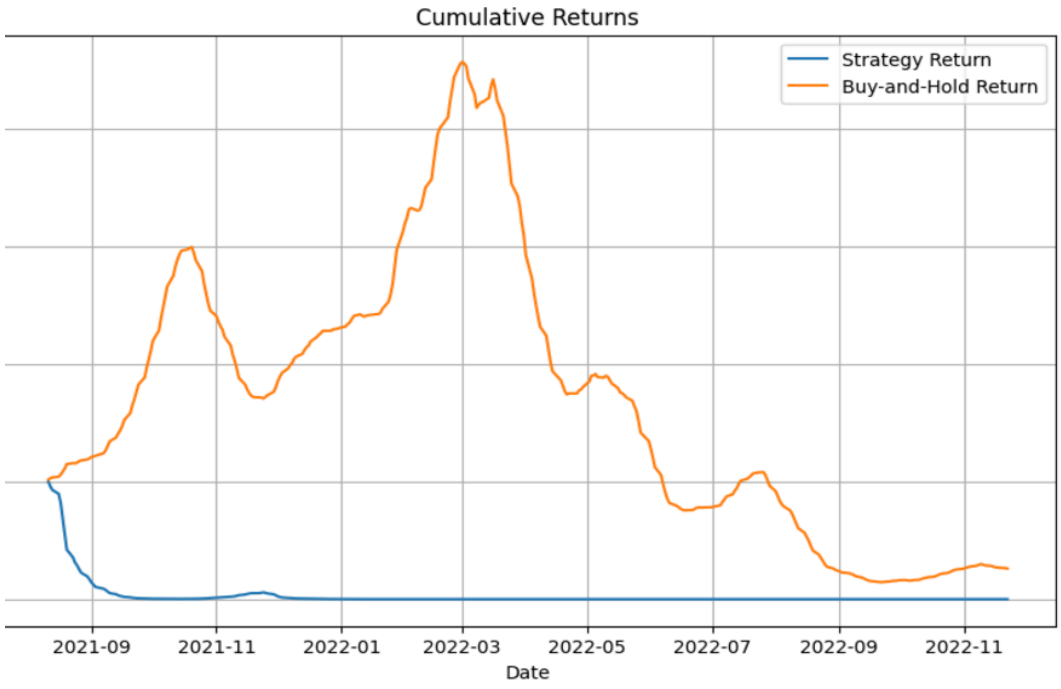

Figure 20: Cumulative returns for the buy-and-hold strategy and the long and short trading strategy for the multi-output model.

Just like we saw for the single input, single output predictions, we also notice that the multi-output model performs worse when it comes to predictions. We notice that after the first initial phase, the strategic return goes almost below zero and does not even attempt to replicate the buy-and-hold strategy. The same reasons apply to 2d, as to why the model could not properly replicate the buy and hold strategy. The single input, the single output, provides predictions for each asset independently, which allows it to uniquely adapt to the behavior of each asset. Contrast this with the multi-output model, which focuses on all the assets at once. In the case when there are relatively volatile assets, as we have observed with SPY, the multi-input, multi-output model might also incorrectly assign weights to the most volatile asset. One of the reasons the multi-input, multi-output model could not perform better could be due to a lack of cross-asset dependency for the model to leverage. Even though these strategies for both cases are not good trading strategies, we can opt for the single input, single output case as it performs marginally better than the other case.

Step 4

As we have seen in previous steps, we have worked with 2 types of model architecture, i.e single-output architecture and Multi-output architecture. We will now discuss the results obtained through both of the models we will compare the results and try to understand their workings, but before that let's discuss the predictability Implications of both architectures, In single-output architecture, our model is optimized individually for each target, and it is assumed that this focused learning may lead to better performance for specific ETFs because the model is tailored to capture unique patterns in each series. On the other hand, Bidirectional Multi-output models ("tf.keras.layers.Bidirectional") simultaneously learn all targets, capturing dependencies and shared patterns among the ETFs. This can be advantageous if the ETFs are interrelated. Single output models are easier to train and interpret, with fewer constraints in capturing patterns specific to each output. On the other hand Muti output models are hard to train, as the shared architecture might dilute the model's ability to specialize in specific patterns for each ETF, though a single model with multiple outputs is more scalable for large datasets, as it avoids the overhead of training separate models.

From our simulations, we notice that the single-input strategy performs better than the multi-output strategy because it isolates prediction errors for each asset, which could provide us with more reliable results. The multi-output strategy underperforms significantly due to possible correlated errors between assets and the impact of volatile assets like the ETF SPY. Both strategies, however, fail to outperform the buy-and-hold strategy, which is used as a benchmark. This could be directly related to errors in prediction accuracy and frequent rebalancing. Improving prediction models and reconsidering rebalancing frequency could indeed potentially enhance trading performance.

Overall, single-output models may perform better if the ETFs are highly independent or if their unique behaviors are critical to the prediction task. If the ETFs exhibit significant correlations or are part of a broader portfolio strategy, a multi-output model is more suitable as it leverages relationships and provides consistency across predictions.

References

1. "Keras documentation: Bidirectional layer." *Keras*, https://keras.io/api/layers/recurrent_layers/bidirectional/. Accessed 3 December 2024.
2. Khaled A. Althelaya Department of Information and Computer Science, Engineering King Fahd University of Petroleum and Minerals, Dhahran, Kingdom of Saudi Arabia. "Evaluation of bidirectional LSTM for short-and long-term stock market prediction." *IEEE*, 2018, https://ieeexplore.ieee.org/abstract/document/8355458/citations?tabFilter=papers#citations.
3. "tf.keras.layers.Bidirectional." *TensorFlow*, https://www.tensorflow.org/api_docs/python/tf/keras/layers/Bidirectional#call-arguments_1. Accessed 3 December 2024.
4. Bernoussi, Rim El, and Michael Rockinger. "Rebalancing With Transaction Costs: Theory, Simulations, and Actual Data." *Financial Markets and Portfolio Management*, vol. 37, no. 2, Sept. 2022, pp. 121–60. https://doi.org/10.1007/s11408-022-00419-6.