**4. Experimentation**

***[Here you write a recipe (steps followed to produce results) for each experiment, then show the results. Against each result, note what the result means. You do not discuss anything at this point. Just write what any other scientist will say the same thing with regard to what you see on the graph or figure.***

***Put results here, I mean a lot of results.***

***You have a set of objectives and sub-objectives in your Chapter 1. Come up with experiments for each of your sub-objectives. Then you can aggregate the results from all these experiments to give the solution to the research problem that was introduced in the dissertation introduction.]***

**4.2 Implementation of LSTM Multivariant price prediction Model**

For the 15 intraday datasets of five digital currencies (USD, Ethereum, bitcoin, Tether coin, and Binance) and their three various frequencies (10mins, 30mins, and 60mins), the LSTM model has been used. Intraday data was collected from March-2022 to Sept-2022. The following essential steps were taken to construct the model and forecast the close price of the cryptocurrencies:

1. Time series data pre-processing to ensure data sanity and perform necessary transformations
2. Scale the data and choose the pertinent features/input variables for modeling.
3. Dividing the data into test and training sets
4. Prepare mini batches that are used to forecast cryptocurrency values using the sliding window method.
5. Design and training of LSTM model
6. Model performance enhancement through hyperparameter tweaking
7. Model validation and prediction
8. Reverse the anticipated target values' scale back to their original value.

**4.2.1 Data pre-processing**

The process of creating a data model must include the pre-processing of the data. The model's performance may suffer if there is an anomaly or data inaccuracy. The first step is to use the *read\_csv()* function from the pandas' package to import the intraday data into a DataFrame and do the following checks:

**Duplicate check:** the purpose of this check is to identify the duplicate records in the dataset and treat duplicate the records. The method *duplicated()* is used which returns boolean values true if duplicate records available in the dataset. There is no duplicate record available in all the datasets. The result of the check is as below for all the datasets:

A picture containing chart

Description automatically generated

*Figure-4.1 Dataset Duplicate Value Result*

**Missing Value check:** if there are null or NA values in the dataset that impact the model performance, they must be treated. To Identify the missing values, the *isnull()* method followed by *sum()* is used that provide a number of missing values. There are missing values for fields like SMAVG\_50d, SMAVG\_100d, and SMAVG\_200dbut there is no impact on modeling performance as they are part of the test dataset. It is discussed during the splitting of the datasets into test and training datasets.

**Table

Description automatically generated**

*Figure-4.2 Dataset Missing values*

**Data Type of Dataset:** The *info()* function, which provides details about the Column name, Non-Null Count, and Data type is used to obtain details about a dataset. Since the Date field's data type is an object, the date is converted using the *to\_datetime(data['Date'])* function before being sorted using the *sort\_values(by="Date", ascending=True)* function. Refer to Figure-4.3 for the *info()* method's output.

**A picture containing text, receipt

Description automatically generated**

*Figure-4.3 Dataset detailed info*

Apart from three checks, data from the datasets selected with help of *from (2022-04-01)* and *to (2022-09-30)* date using code snippet:

#Dataset Selection by date from dataset

data=data[(data.index >= f\_date) & (data.index <= t\_date)]

**4.2.2 Feature Engineering and Scaling**

Identifying and developing relevant input features to feed into the Long Short-Term Memory (LSTM) network is referred to as feature engineering in the context of employing LSTM networks for time series analysis or prediction. These features can be determined from the time series' raw data or from additional outside data sources. Multiple input features that each relate to a different variable in the time series data would be present in a multivariate time series problem.

**Diagram

Description automatically generated**

*Figure 4.4 Feature Engineering for LSTM model*

There are seven numerical (*High, Low, Open, Close, SMAG\_50d, SMAVG\_100d, SMAVG\_200d*) and one date column in the dataset. A simple moving average (SMAVG) is a statistical tool that is frequently used to smooth out short-term volatility in data and highlight longer-term patterns. A SMAVG can be used to smooth out price swings and spot underlying trends in the price of coins in the world of cryptocurrencies. To create a features’ DataFrame, SMAG\_100d is dropped using *drop(['SMAVG\_100d'], axis=1)* functions. Selected features are shown in Figure 4.4,  which will further pass to the prediction model.

Then feature values are scaled to make them all fall within the same range. Additionally, it speeds up network convergence and helps the LSTM network operate better. The features' values are scaled using the Mix-Max scaling approach in the range of 0-1.

Text

Description automatically generated

*Figure 4.5 Feature value scaling using MixMaxScaler*

The *MinMaxScaler* class from the *sklearn.preprocessing* package is used to scale the NumPy array of input features, as illustrated in Figure 4.5.

**4.2.3 Prepare the Input Data for LSTM**

A particular input (3D array) and output (2D array) data structure is needed for LSTM. The first dimension of a three-dimensional data structure is the sequence, the second is the time steps, and the third is the features (refer to Figure 4.6). Before transforming data, Dataset is split into training and train sets, in which 80% of data is assigned to the training set and 20% of data allocate to the test set.

**A picture containing diagram

Description automatically generated**

*Figure-4.6 3D Input Data Structure for LSTM*

The sliding window technique is used to prepare multivariate data, that can be processed by the neural network. Data is sliced into the window of specific sequences with the associate target value. This approach adds multiple data point input datasets (x\_train or x\_test), and the target value follows the window. The Target value is stored in a separate target dataset (y\_train or y\_test).  Refer to Figure 4.7 for the python script of the sliding window.

Table

Description automatically generated

Graphical user interface, text, application

Description automatically generated

*Figure-4.7 Sliding window Technique with python script*

The function *transform\_multivariate\_data()*, which accepts the original dataset, scaled data, and size of the window sequence, is developed to perform the operations mentioned earlier like transformation, splitting, and partition. The function's output is illustrated in Figure 4.8 below.

**A picture containing text

Description automatically generated**

*Figure-4.8 final test and training dataset for LSTM*

**4.2.4 LSTM Model Training**

**Table

Description automatically generated**

*Figure-4.9 LSTM architecture details*

**4.2.4 The model performance Analysis**

Performance is assessed using the Root Mean Square Error (RMSE) and Loss Function (MSE), both of which were defined during model design. After being fitted with the training dataset, the model returns the history variable which contains information about performance metrics and a loss function. The *plot()* function of the *matplotlib.pyplot* library used to depict the error data using two line graphs (epochs v/s loss and epochs v/s RMSE score). For details on both line plots, see Figure 4.10.

Graphical user interface, chart, line chart

Description automatically generated

*Figure-4.10 Error line graphs (Epochs v/s RMSE score and Epochs v/s Loss)*

**4.2.5 Hyperparameter tuning for Model**

The model has two different types of parameters which are model parameters and hyperparameters. The hyperparameters are control implementations of the model and need to be set for better performance of the machine learning model, whereas the model parameters are learned during model training. Tuning the hyperparameters improves the model's precision and generalization.

The GridSearch Method was utilized to adjust the LSTM hyperparameter as part of the experiment. A grid of hyperparameter values is created using the combinations of neurons (16, 32), batch size (32, 64, 128), and dropout (0.1, 0.2), and for each combination, the model is trained to determine which hyperparameter combination is optimal. The potential combinations of the hyperparameters are developed using the *product()* function of the itertools package. The same LSTM design described in section 4.2.4 is employed for the tuning procedure. Two callbacks *Modelcheckpoint* and *EarlyStopping* are passed when creating the model to improve the effectiveness of the grid search parameter tuning. To comprehend the layout of the callbacks and model, consult Figure 4.11.

Graphical user interface, application, Word

Description automatically generated

*Figure-4.11 structure of the fit() functions and callbacks used for the LSTM model*

According to the input parameters (*monitor* and *mode)*, EarlyStopping callbacks track the training loss. It determines if the loss is decreasing or not at the conclusion of each period. If the loss is not decreasing, the training is stopped based on the *patience* parameter. The best model's current state is saved to disc during a ModelCheckpoint callback so that it can subsequently be used to continue training the dataset.

After model training, the hyperparameters and accuracy scores are recorded for each combination in a list. The hyperparameter with the lowest error score is regarded as the best hyperparameters and model trained with the optimized hyperparameters.

**4.2.6 Predict Cryptocurrency price**

Test dataset (20% of the total dataset) is passed to predict the price of five cryptocurrencies as input to *predict()* method of the optimized model. The predicted price is scaled and reverse order, that’s why inverse transformation is performed *inverse\_transform()* of *MinMaxScaler* to get actually values in the correct order.

**Text

Description automatically generated with medium confidence**

*Figure-4.12 functions for Prediction and Unscaled predicted values*

With the help of a line graph, The predicted price, and the original price visualized, it helps to study the outcome of the LSTM model. It will be discussed in the next section for each crypto coin.

**4.3 Results of the LSTM multivariate model**

The below findings are recorded after running the LSTM model for 15 datasets (for 3 intraday frequencies of 5 cryptocurrencies), and they are discussed in this section.

1. Hyperparameters before and after optimization, RMSE score and overall % improvement of training LSTM model
2. Line graph of Loss function mean\_square\_error
3. Line graph of prediction v/s original scaled values for test dataset
4. Line graph of prediction v/s original values with residual bar plot for Overall dataset
   * 1. **Bitcoin**

The results of Bitcoin's before-and-after optimization are displayed in Table 4.x. The 30-minute frequency dataset has the biggest percentage increase in performance, whereas the 10-minute frequency dataset has the lowest percentage improvement.

**Table

Description automatically generated**

*Table 4.X Result of LSTM Hyperparameter Tuning (Bitcoin)*

The RMSE score rises as the number of input samples decreases; for instance, the RMSE score for a 10-min dataset with 20918 training samples is lowest before and after optimization, coming in at 0.0051 and 0.0043, respectively. while the 60-minute dataset with 3446 samples has the greatest RMSE score. Compared to other models in the training set, LSTM predicted closing prices for 10 minutes of the frequency with greater accuracy.

Engineering drawing

Description automatically generated

*Figure 4.X Line graph - Loss Vs Epochs before/after hyperparameter tuning (Bitcoin)*

For all bitcoin datasets, the fitting of the model is increased significantly, and epochs are minimized after hyperparameters tuning, as seen in Figure 4.x. In contrast to other bitcoins datasets, LSTM is a good fit for the 10-mins frequency data because the loss function value is less than 0.001. For that dataset, the optimum hyperparameters are neurons 32, batch size 64, and dropout 0.2. It implies that predication for 10-mins dataset closest to the actual value of close price for training data.

Chart, histogram

Description automatically generated*Figure 4.X Predicated v/s Real Scaled Close Price (Bitcoin)*

Above Figure 4.x illustrated the comparison between the predicated and original values for test dataset using scale close price of the bitcoin. The accurate the predication of price trend is visible in 10-mins bitcoin graph as compared to other line graphs.

Chart

Description automatically generated with low confidence *Figure 4.X* Line graph of the close price with bar plot of the residual *(Bitcoin)*

Figure 4.x shows a line graph for the test and train datasets with a residual bar graph for the test dataset at the bottom of the graph. According to the Table 4.x, the percentage minimum and maximum residual values for the 60-min bitcoin dataset are higher and lower than those for the 10-min dataset. It displays that forecasted values are reasonably near to the real values for the 10-minute dataset, which are also shown in Figure 4.x above.

*Table

Description automatically generated*

*Table 4.x Percentage Min and Max Residual for predicted value (Bitcoin)*

In conclusion, the LSTM performance for the training dataset improved and the lowers the RMSE score across all datasets after hyperparameter tuning, which also reflects in the prediction of the close price of bitcoin. For the 10-min dataset, the closing price prediction is more precise than other bitcoin datasets.

* + 1. **Binance**

The outcomes of the hyperparameter optimization for the Binance coin are shown in Table 4.x. The average improvement performance across all datasets is around 47%.

**Table

Description automatically generated**

*Table 4.X Result of Hyperparameter Tuning (Binance)*

The RMSE value is highest for the 60-min dataset, where it is 0.0263 before optimization and 0.0136 after, whereas it is lowest for the 10-min Binance dataset, respectively 0.0106 and 0.0055 for before and after tuning. Compared to other datasets, the 10-min dataset's prediction error is low with new hyperparameters (neurons-16, brach\_size-32 and dropout-0.2).

Diagram

Description automatically generated

*Figure 4.X Line graph - Loss Vs Epochs before/after hyperparameter tuning (Binance)*

The loss value decreased for all datasets after optimization with epochs except for the 10-min dataset of Binance. For 10-min dataset, 15 epochs were required to achieve the optimized MSE loss values, but its loss value is the lowest among all the datasets. As a result, the LSMT is a good fit 10-min dataset, while underfit for the 60-min dataset as it has higher loss value for training.

Chart, line chart, histogram

Description automatically generated

*Figure 4.X Line graph of Predicated v/s Real Scaled Close Price (Binance)*

The trend comparison of the closing price for the test dataset's original and predicted values is shown in Figure 4. x. The 10-min dataset has a lower predicted line to real price deviation than the 30-min and 60-min datasets. The LSTM can forecast price fluctuations for a 10-min dataset with high accuracy.Chart

Description automatically generated

*Figure 4.X* Line graph of the close price with bar plot of the residual *(Binance)*

The Figure 4.x illustrate line graph of the close price for the overall dataset with bar chart of the residual values. The percentage of residual value range for 10-min data is almost lower as show in Table 4.x. Because of that, the predication line is very close the real price for 10-mins test data as compared to other datasets.

**Table

Description automatically generated**

*Table 4.x Percentage Min and Max Residual for predicted values (Binance)*

Overall, The training dataset's performance is improved when the hyperparameter has been tuned. For the 10-min dataset, the closing price is predicated with high accuracy, that has the lowest RMSE value during the training and test phase. The trend of the line graphs and the residual bar plot further strengthen that claim.

* + 1. **Ethereum**

The LSTM performance result is presented in table 4. x for all datasets of the Ethereum. The efficiency of the model is improved after the optimization for all the datasets. A high improvement is reported for 30-min (61%) dataset followed by the 10-min (26%) and 60-min (19%) datasets.

Table

Description automatically generated

*Table 4.X Result of Hyperparameter Tuning (Ethereum)*

Chart, box and whisker chart

Description automatically generated*Figure 4.X Line graph of Loss Vs Epochs before/after hyperparameter tuning (Ethereum)*

The RMSE score of the LSTM model for the 10-mins training dataset is the lowest before (0.0072) and after (0.0053) hyperparameter tuning than 30-min (0.0221 & 0.0087) and 60-min (0.0124 & 0.0101) dataset. As shown in the line graph of MSE loss function, the value of the loss is decreased for 10-min (approx. 0.001) and 60-min (approx. 0.0015) after the optimization, which means the LSTM model is good for both datasets. The required epochs increase to achieve efficiency for both datasets. The RMSE score is lower for the 10-minute dataset, so the LSTM model performed better than the 60-min dataset during training.

Chart, line chart

Description automatically generated

*Figure 4.X Line graph of Predicated v/s Real Scaled Close Price for Ethereum*

Graphical user interface, chart

Description automatically generated

*Figure 4.X* Line graph of the close price with bar plot of the residual *(Ethereum)*

As show in the Figure 4.x, the line graph of the predicated values for 10-mins is closer the real value and the fluctuation of the price is accurately the predicated by the LSTM model. For 30-min and 60-min dataset, the clear distance is visible between both lines. For 10-mins data, The % Min-Max residual value range is less than other datasets. Among all, The perform of LSTM model is better for 10-min dataset.

Table

Description automatically generated

*Table 4.x Percentage Min and Max Residual for predicted values (Ethereum)*

* + 1. **USD coin**

Compared to other cryptocurrencies, the USD coin's performance improvement is minimal even after the hyperparameter tuning of the LSTM model. According to below Table 4.x, the maximum improvement is recorded for the 60-min (8%) dataset and the lowest for the 10 min dataset (2%).

**Table

Description automatically generated**

*Table 4.X Result of Hyperparameter Tuning (USD)*

The RMSE value is significantly higher for 30-min (0.0162 & 0.0153) and 60-mins (0.0209 & 0.0193) datasets as compared to 10-mins (0.006 & 0.0059) datasets irrespective of the hyperparameters optimization of LSTM model. It shows training performance of LSTM for the 10-min data is more efficient and highly accurate than other datasets.

Chart, engineering drawing, box and whisker chart

Description automatically generated

*Figure 4.X Line graph - Loss Vs Epochs before/after hyperparameter tuning (USD)*

Figure 4. x illustrates the line graph of the Loss values against the epoch of the LSTM model. As shown in the Figure 4.x, there are no significant changes in the Loss values after parameter tuning. It is aligned with the result of the RMSE values displayed in table 4.X. The epochs for 10-min data are reduced from 18 to 8 after optimization of the LSTM, and loss value is below 0.00025 for model training. As a result, the LSTM model is a good fit for 10-min data.

According to Figure 4.x, The line graph of scaled closing price values for the LSTM model's prediction output captures the variation between the predicted and actual close price. The line graph of the 30-minute and 60-minute datasets does not adequately predict the close price's patterns. The LSTM model will occasionally predict and complete trends in the opposite direction for 60 minutes. The 10-min dataset has the highest level of trend prediction accuracy across all datasets.

A picture containing diagram

Description automatically generated

*Figure 4.X Predicated Scaled Close price v/s Original Scaled Close Price (USD)*

According to Figure-4.X, the training and test dataset’s line graph for forecasted and real-time pricing is straight lines with minor changes. The range of prices that USD coins had between March 2022 and September 2022 is the possible cause of it. For the specified interval, the actual closing price of the US coin ranges from 0.9975 to 1.0016. it is preferable to refer to the line graph of scaled values and the percentage of mix-max residual values (refer to Table 4. x) to understand the performance of the LSTM

for USD coin.Graphical user interface, diagram

Description automatically generated

*Figure 4.X Predicated Close Price and Residuals Values (USD)*

*Table

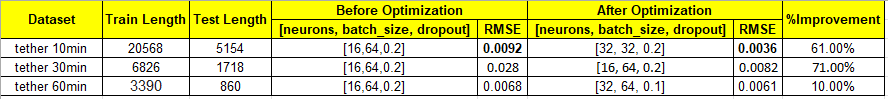
Description automatically generated*

*Table 4.x Percentage Min and Max Residual for predicted values (USD)*

In comparison to the 30-min and 60-min datasets, the percentage of Min and Max residual values is lower for the USD 10-min dataset. It shows that the predicated values based on 10-min data are more in line with the USD's actual close price. In addition, the training dataset's RMSE score, and loss values show that the LSMT model accurately predicts the close price for 10 minutes.

* + 1. **Tether**

Table 4 .x contains the results of the hyperparameter optimization for the tether coin. The LSTM model's improvement percentage is highest for 30-min data (71%) and lowest for 10-min (61%) and 60-min (10%) data after parameter tuning.



*Table 4.X Result of Hyperparameter Tuning (Tether)*

As shown in the table, the RMSE score is maximum for 30-min data before and after that hyperparameter tuning that is 0.028 and 0.0082 respectively, Whereas it is lowest for 60-min data (0.0068) before optimization and 10-min data (0.0036) after optimization. It represents the LSTM model prediction accuracy as higher for the 10-min dataset after optimization as compared to other datasets.

Chart

Description automatically generated

*Figure 4.X Predicated Close Price and Residuals Values (Tether)*

Figure 4. x illustrates the loss value is decreased for the training dataset with epochs for all the datasets of tether coin except the 30-min dataset after optimization of the LSTM model. For 30-min data, there is no change in epochs (10), while the loss values decrease after parameter tuning. The LSTM model is a good fit for 10-min data as the loss value (below 0.0025) is the lowest among all datasets.

A picture containing chart

Description automatically generated

*Figure 4.X Predicated v/s Original Scaled Close Price for Tether*

According to Figure 4.x, The predicated close price of the 10-min dataset can capture the trend of the close price well as compared to the 30-min and 60-min datasets. In line graph, the inverse trend of tether price is captured for 30-min and 60-min datasets for predicated values (refer to green line). The prediction model can capture better trends for 10 min dataset.

Graphical user interface, diagram

Description automatically generated with medium confidence

*Figure 4.X Predicated Close Price and Residuals Values For Tether*

Figure 4. x is difficult to interpret since the graph is a straight line for both real and forecasted values. The close price for March to September 2022 could be the cause. For the collected dataset, the closing price value is observed between 0.9492 and 1.0668. The scaled line graph (Figure 4. x) and the percentage change in the residual values (Table 4. x) are preferable references to evaluate the performance of the LSTM model.

**Table

Description automatically generated**

*Table 4.x Percentage Min and Max Residual for predicted values (Tether)*

The change in the percentage residual values is lower for the 10-mins dataset (-0.25% to .042%) than other datasets, which concludes that the predicted values are comparatively accurate, and LSTM performed better prediction for 10-min data.

In conclusion, the performance of the LSTM model is the best for a 10-mins dataset for all the cryptocurrencies in terms of the RMSE score, Loss function (MSE), residual value, and trend of the predicated close price for training and test datasets as compared to 30-min and 60-mins dataset. In the next phases of the experiment, The close price of the 10-min dataset have been used for the portfolio construction and performance analysis of the portfolios.

**4.4 Portfolio Construction and Performance Analysis**

The Experiment of the four portfolio construction approaches that were mentioned in section 3 and their result are discussed in this section. The close price of each cryptocurrency is input, while the weight of each crypto is the output of the portfolio construction methods. Here, the weight is the percentage of the investment for cryptocurrency in the single portfolio. Section 4.2 provides the foundation for the predicated closing prices of all crypto currencies. After that, how to use the calculated weight for all coins to determine the performance metrics is described for all portfolios. To undertake a comparative analysis of portfolio strategies and their performance over different time periods and the close price trend, portfolios are formed using historical daily data of 2021. Because of up and down trend of the close price as show in Figure 3.x of EDA analysis, the historical data of 2021 have been selected. The historical data of 2021 is a subset of the last 5 years (2018-2022) of cryptocurrency data. It was collected for exploratory data analysis at the time of data collection.

**4.4.1 Data Preparation for Portfolio Construction**

For portfolio creation, Data preparation must require step because the output of section 4.2 is the Intraday close price (10mins frequency of trainset and forecasted values), while daily close price has been required as input here. Secondly, the close price of all cryptocurrencies is in different CSV files for the experiment dataset and historical dataset of 2021. There are two major objective of the data preparation 1) Converting intraday prices into daily prices, and 2) combining the close prices of all crypto assets into a single output file.

**Converting intraday prices into daily prices**

The close prices from the training dataset and forecasted prices have been loaded in the dataframe from the respective CSV files. For each cryptocurrency, The last price of the day has been fetched using the sort (), groupby(), and agg() functions of the dataframe, and the date index reset using *index\_reset() function* from both datasets. Before that, few data pre-processing operations have been performed to remove NA values from predicated dataframe and filter the training dataset. Refer to Figure 4.x to understand the above process for USD coin.

**Text

Description automatically generated Graphical user interface, text, application

Description automatically generated**

*Figure 4.x Derived daily price from the Intraday close Price for USD Coin*

**Combining the close prices of all crypto assets into a single output file**

By executing an inner join on the index column (Date) using the *merge()* function, all coin dataframes are combined into a single dataframe. After that. It is saved to an output csv file using the *to\_csv()* method. Similar to that, all coins' history data from 2021 is combined and saved to an output file. Refer to Figure 4.x for merge operation of final data frame for train and forecasted close price.

*Text

Description automatically generated with medium confidence*

*Figure 4.x Merge and Save operation for close price of predicated price dataset.*

The daily close price of the historical dataset and the experiment dataset for all cryptocurrencies are saved in corresponding CSV files as the final output. Both datasets are used while building a portfolio and analyzing performance.

**Note:** *Going forward, The close price of the training data and forecasted dataset is referred as experiment dataset which is output of the data prepare for portfolio construction and analysis.*

**4.4.2 Mean-Variance Portfolio Method**

The Mean-variance portfolio method is also known as the Modern Portfolio Theory (MPT). As discussed in section 3, The aim of the MTP is to maximize the Sharpe ration and minimize the mean variance of the

portfolio to achieve expected returns for given level of the risk. Following steps are followed to implement the mean variance portfolio methods:

1. Calculate mean daily return and daily variance of return for all cryptocurrencies
2. Find the annual return and variance by annualizing the output of the step 1
3. Generate the random weights for all assets and derive the annual return, sharp ratio, and standard deviation
4. To find optimal portfolio, simulate 500000 portfolios using step 1 to 3 using Monte Carlo Simulation
5. Find the optimal portfolios which have max sharp ratio and min variance
6. Apply step 1 to 5 on Experiment and historical dataset from section 4.4.1

The Sharpe ratio is equal to the expected return of the portfolio divided by the standard deviation (variance) of the portfolio, according to Equation-3.X in Section 3.3. The mean daily return and daily standard deviation of return are determined for all crypto currencies using the *pct\_change(), returns.mean(), and cov()* functions. The mean expected annual return and variance are then obtained by simply annualizing the results to find the sharp ration and mean variance. For the computation of the mean daily return and covariance of daily returns, see Figure 4.x.

A screenshot of a computer

Description automatically generated

*Figure-4.x Daily mean return and Variance Calculation*

The annual return and variance need to be computed for the weights of all cryptocurrencies in a mean-variance portfolio. The Monte Carlo simulation is run with 500000 distinct combinations to determine the portfolio's ideal weights for the highest returns and minimal risk. For each set of portfolios, the annual return, standard deviation, and sharp ratio are computed. Refer to Figure 4. x to construct 5 lacs simulated portfolio using a Monte Carlo simulation.

Text

Description automatically generated

*Figure-4.x Monte Carlo Simulation for Mean-Variance Portfolio*

By using the *idmax()* function on the sharp ratio and the *idmin()* function on the variance columns of the resultant dataframe of the simulated portfolios, the best mean-variance portfolios are created. For a scatter plot of every simulated portfolio, see Figure 4. x. The green start represents the minimum variance portfolio, while the red star represents the maximum sharp ratio portfolio in the scatter plot.

Chart

Description automatically generated

*Figure-4.x Scatter plot of Simulated portfolios*

Using the above experiment, mean variance portfolios are produced for the historical dataset of 2021 as well as the experiment dataset. Below Table 4.x lists the values of weights for the maximum sharp ratio and the minimum variance (volatile) portfolios with annual return, sharp ratio, and standard deviation.

Table

Description automatically generated

*Table-4.x Derived Weights Using Mean-Variance Portfolio Method*

According to Table 4.x, the portfolio returns of the experiment dataset are more likely to be zero or negative (-0.009 and -0.2615) than positive (0.76 and 0.021) for the historical data. As a result, the scatter plot of the simulated portfolios for the experiment dataset is pointing in the opposite direction of that of the historical dataset. It can be seen in Figure 4.x. The downward trend of the major cryptocurrencies from March 2022 to September 2022 is one potential cause of the experiment dataset's portfolios' poor returns.

**4.4.2 Hierarchical** **risk parity Method**

Hierarchical risk method(HRP) is recent advance portfolio construction methods which use Hierachical clusting models in allocation. As part of the experiement, The implementation of Hierarchical risk parityfrompyportfolioopt lirabry is used. According to the *pyportfolioopt.readthedocs.io* documentation, the Hierarchical risk parity Method work as mentioned below:

1. From The all given assets, form distance matrix using correlations among all assets
2. Cluster the assets in tree using Hierarchical clustering with help of distance matrix derived from the step 1
3. Form a means variance portfolio between two assets within clusters
4. Optimally combine the min-portfolios from all nodes by iterating through each level.

To implement the Hierarchical risk parity method, the expected return and clean weight for all the assets are calculated. The expected return is calcualted using the *returns\_from\_prices()* method of *expected\_return* package, while clean weights for HRP portfolio is derived with help of *optimize()* and *clean\_weights()* methods of *HRPOpt* package from the *PyPortfolioOpt* library.

Text

Description automatically generated

*Figure-4.x Output weight of Hierarchical risk method for Experiment Dataset*

A historical closing price dataset and an experiment are both used to build clean weights using the Hierarchical Risk Parity approach. Table 4.x displays the experiment's results. The *plot\_dendrogram()* function of the *PyPortfolioOpt.plotting* package is used to visualize the created clusters of crypto assets. Additionally, the pie chart of clean weights is drawn using the *plot()* function. Figures 4.x and 4.x display the dendrogram plot and pie charts for the two datasets, respectively.

Chart

Description automatically generated

*Figure 4.x Pie chart and dendrogram plot for Experiment data*

Chart, box and whisker chart

Description automatically generated

*Figure 4.x Pie chart and dendrogram plot for Historical data of 2021*

The Ethereum, Binance, and bitcoin created clusters, as shown in Figure 4.x for the experiment dataset, but there is no similarity between USD and tether, so no cluster was created for those assets. For the Historical dataset, two clusters have been established: one for Ethereum and Bitcoin and another for USD and Tether. However, there is no cluster for Binance Coin.

Table

Description automatically generated

*Table-4.x Derived Weights Using Hierarchical Risk parity*

no allocation is made for Ethereum, Binance, and bitcoin, while the USD and tether assets are given respective weights of 0.00636 and 0.99353 for the experiment dataset. Tether dominates the asset allocation for the experiment dataset, which is also shown in the pie chart. Contrarily, for historic closing price data, tether and USD are distributed between 0.53 and 0.46, while the remaining assets are allocated between 0.00010 and.00020. As seen in figure 4.x, it also fit with the group established in the historical dataset's dendrogram and pie chart.

**4.4.3 Kelly’s Criteria**

The kelly’s criteria focus on allocation of risk to few assets and it is considered as too risk. To mitigate that, the risk wagering fraction should be introduced, so Investor can invest the capital to others risk free assets [4]. As part of the experiment, the wagering fraction is considered.

The close price of the assets is loaded into the dataframe from CSV files. The pct\_change() method is used to derive the daily and annual returns of the assets. The cov()  function is applied to the annual return and calculates annual co-variance. The weights of a portfolio are calculated by applying kelly’s criteria with help of the wagering fraction (0.50), and covariance matrix. Refer to Figure 4. x for kelly’s criteria function.

Text

Description automatically generated

Figure 4.x Kelly’s criteria

The output of the Kelly’s criteria method for both datasets are recorded in the below Table 4.x.

Table

Description automatically generated

*Table-4.x Derived Weights Using Kelly’s Criteria*

The weight value of Binance, bitcoin, and Ethereum is zero for both datasets, which indicates those assets are risk assets and no weight allocation is done for the portfolio. The portfolio weights are allocated to tether (0.49 & 0.13) and USD coin (0.010 & 0.37) for both datasets as per wager fraction.

**4.4.4 Equal weight Portfolio Method**

The Equal weight Portfolio Method is also known as naïve asset allocation method. The allocation of asset is done using equation 5 of section 3.3 for equal weight portfolio. There are five cryptocurrencies for portfolio and allocation is done as follow:

wi(weight) = 1/5 = 0.20 for each crypto asset.

For the experiment and historical datasets, The equal weights are assigned and accordingly the performance analysis is be done.

**4.4.5 Portfolio Performance Analysis**

The metrics outlined in section 3.3 constitute the basis of the performance analysis of the portfolio. With the help of the portfolio weights and close price of all crypto coins, metrics like volatility, annual return, Sortino ratio, and sharp ratio can be calculated.

The dataframe is loaded with the dataset's close price, and the df/df.shift(1) function is used to determine the log return. A list contains information on portfolios that are derived in section 4.4. The following metrics are produced for each portfolio as you iterate through the list of their holdings: mean, downside standard deviation, annual volatility, annual sharp ratio, drawdown, annual return, Sortino ratio, and volatility skewness. For the computation of each measure, see Figure 4. x.

Graphical user interface, text, application

Description automatically generated

*Figure-4.x Calculation of portfolio performance measure*

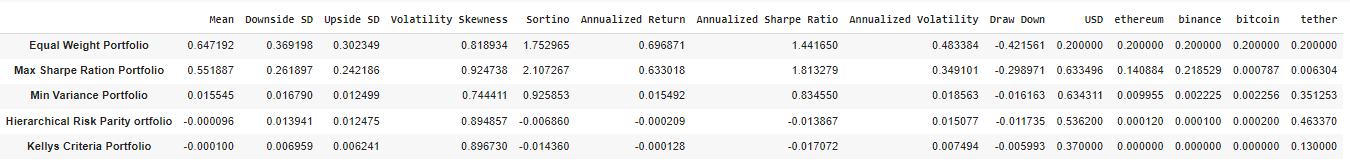
The performance metrics have been calculated for historical and experiment dataset and the results are displayed in the Table 4.x and Table 4.x for experiment and Historical data respectively.

Graphical user interface, text, application

Description automatically generated

*Table-4.x Portfolio measures for performance Analysis (Experiment data)*

All portfolios’ annual return is minus or zero. It indicates portfolios are at loss. The equal weight portfolio is the worst performing portfolio as the annual return (-0.370815), mean (-0.4014), and drawdown (-0.46490) is the lowest as shown in Table 4. x. For The EWP, Annualized volatility (0.3488), downside deviation (0.2900) is the highest, that indicates the risk is very high. All measures indicate that equal distribution of assets for all cryptocurrencies is not a good strategy for volatile assets. The Annual return of Kelly’s criteria and Hierarchical Risk parity portfolios is almost zero, which implies those methods distributed risk better than other methods for experimental data. The annual volatility of HRP and Kelly’s portfolio is 0.0056 and 0.0050, that is lowest. It supports the fact about risk distribution for both methods.



*Table-4.x Portfolio measures for performance Analysis (Historical data)*

Table-4.x illustrates, The annual return is positive for the equal weight (0.6968), max sharp ratio (0.633018), and min variance (0.015492) portfolio, whereas it is almost zero for HRP and Kelly’s portfolio. If we consider the risk distribution of the assets, the equal-weight portfolio has the highest annual volatility (0.4833), Mean (0.6471), and downside standard deviation (0.36198). The numbers are high for the maximum Sharpe ratio portfolio also. The high downside deviation portfolio is not preferable even though good portfolio returning because it indicates the price volatility of the investment [7]. The risk distribution of Kelly’s criteria and HRP portfolio is better than other methods and in line with the finding of the portfolios analysis for the experiment dataset.

**Reference: [54]**

[1] Team, K. (n.d.). *Keras documentation: Callbacks API*. [online] keras.io. Available at: <https://keras.io/api/callbacks/>.

[2] Anon, (2017). Investment Portfolio Optimisation With Python - Python For Finance. [online] Available at: <https://www.pythonforfinance.net/2017/01/21/investment-portfolio-optimisation-with-python/>

‌

[3] pyportfolioopt.readthedocs.io. (n.d.). *Other Optimizers — PyPortfolioOpt 1.5.2 documentation*. [online] Available at: <https://pyportfolioopt.readthedocs.io/en/latest/OtherOptimizers.html#id4> [Accessed 30 Dec. 2022].

‌[4] KellyPortfolio. (n.d.). KellyPortfolio. [online] Available at: <https://thk3421-models.github.io/KellyPortfolio/> [Accessed 30 Dec. 2022].

‌ Brownlee, J. (2017). *How to Diagnose Overfitting and Underfitting of LSTM Models*.

[5] [online] MachineLearningMastery.com. Available at: <https://machinelearningmastery.com/diagnose-overfitting-underfitting-lstm-models/#:~:text=LSTM%20models%20are%20trained%20by> [Accessed 31 Dec. 2022]. In-text citation: (Brownlee, 2017)

[6] Brownlee, J. (2018). *Difference Between a Batch and an Epoch in a Neural Network*. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/> (Brownlee, 2018)

[7] The Balance. (n.d.). *Downside Deviation*. [online] Available at: <https://www.thebalancemoney.com/what-is-downside-deviation-4590379> [Accessed 1 Jan. 2023].‌