NFT Market Price Prediction and Trading

Abstract

Non-Fungible Token (NFT) has quickly gained attention from 2021 with record sales in the market. While there are increased interests in NFT, research on the topic is still very limited. As a result, we wanted to investigate how we can predict price of an NFT product and create a usable system that can help investors to build portfolio of NFT stock. The backbone is a machine learning model trained to create price prediction, and currently our neural networks can achieve 95% accuracy. Based on our model, we generated a whole year trading data for each collections, and we implemented a quantitative trading platform that returns best portfolio allocations for NFT investors.

1 Introduction and Problem Statement

Non-Fungible Token (NFT) is a type of cryptocurrency that shows an ownership on an asset. NFT is stored on a blockchain, and it is not fungible, meaning it is unique and therefore cannot be exchanged [**?**]. In recent years NFT has attracted great attention from investors and researchers. According to Coingecko, NFT has accounted for 1.3% of the entire cryptocurrency market in only 5 months [1] .

While there are increased interest in NFT, research on the topic is still very limited, and most of them are focused on the technical aspect, such as protocols, copyright regulations, and standards. For our contribution we focused on the predictability of NFT sales by training a machine learning model to predict NFT price. Furthermore, we implemented an application that can recommend NFT investment combination based on an input US dollar amount, an idea that has not been done before.

[1]Data source from Coingecko (May, 2021) https://www.coingecko.com/en/nft

2 Related Work

Literature review has been done to show related works by other researchers in this field. For example, Nadini [2] and his team have identified that items exchanged on the NFT market are organized in collections, sets of NFTs that, in most cases, share some common features, where most collections can be categorized in six categories: art, collectible, games, Metaverse, utility, and others. So in the Exploratory Data Analysis part, we made several data aggregations based on NFT collection categories to find internal patterns. Beyond the above categories mentioned in Nadini's work, we expand them into more detailed categories. They also mention that the market was fully dominated by the Art category, and in particular by the CryptoKitties collection. Other categories, such as Games, and Metaverse, also have influence. Similar finding is shown in the following part of our work.

Uygun [3] trained convolutional neural networks (CNN) to conduct predictions about future sales of NFT products. They use visual feature extraction approaches to predict Creator, trader, marketplace, and domain based information. This approach works pretty well on visual NFTs like avatar. However NFT products include not only figures, but also Games, or Virtual Properties. So we are going to find a more generalized way to build the prediction model. Based on the fact that all transactions are time series and the advantage of Long Short-Term Memory (LSTM) network to predict time series, we decide to design an NFT trading strategy based on LSTM model from scratch.

3 Approach

We performed model selection and used a machine learning model to predict the NFT price based on historical data on NFT transcation. Beyond the above review for the NFT market, we specifically focus our research on the price prediction for the NFT sales. We plan to do

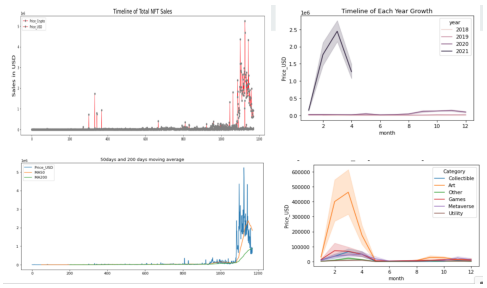

Figure 1: Timeline of Total NFT Sales

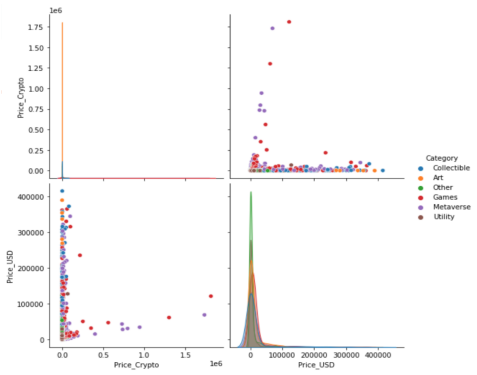

Figure 2: Pair plot of Total NFT Sales

model selection and use a machine learning model to predict the NFT price based on historical data on NFT transcation. There are three major problems to be considered when predicting for NFT products: model and features, potential biases and the relationship with cryptocurrencies. Furthermore, we also developed a quantitative trading strategy to help investor build the best portfolio of NFT Stock.

3.1 Exploratory Data Analysis

Some key findings of our data set includes: Sharp increase at the beginning of 2021, and Art category has the dominant market size. Many outliers detected in the boxplot, Price in USD dollars has less variance than Price in Crypto. There's no strong multi-variate correlation between each category. We also identify top 5 buyers, sellers, collections, distribution and box plot. See Appendix I for more details.

3.2 Preprocessing of the train and test data

According to the Pairplot in Fig 2, Price in crypto has larger variance than Price in USD. We choose Price in USD as the target for preprocessing. We group by the sum of each day's USD price to get the Daily Price of the market. We also group by each day's collection data for quantitative trading strategy. To remove the outliers, we dropped the The data points which fall below Q1 – 1.5 IQR or above Q3 + 1.5 IQR We split train and validation dataset into 90/10 and cross-validation with k-fold size=10 to fit in the LSTM Neural Network.

3.3 LSTM Neural Network

The price prediction of our project is based on LSTM Model. Long Short-Term Memory (LSTM) has been so designed that the vanishing gradient problem is almost completely removed, while the training model is left unaltered. Long time lags in certain problems are bridged using LSTMs where they also handle noise, distributed representations, and continuous values. With LSTMs, there is no need to keep a finite number of states from beforehand as required in the hidden Markov model (HMM). LSTMs provide us with a large range of parameters such as learning rates, and input and output biases. Hence, no need for fine adjustments. The complexity to update each weight is reduced to O(1) with LSTMs, similar to that of Back Propagation Through Time (BPTT), which is an advantage.

The Architecture of LSTM consists of four layers that interact with one another in a way to produce the output of that cell along with the cell state. These two things are then passed onto the next hidden layer. Unlike RNNs which have got the only single neural net layer of tanh, LSTMs comprises of three logistic sigmoid gates and one tanh layer. Gates have been introduced in order to limit the information that is passed through the cell. They determine which part of the information will be needed by the next cell and which part is to be discarded. The output is usually in the range of 0-1 where '0' means 'reject all' and '1' means 'include all'.

3.4 Training and Hyper parameter Tuning of LSTM

To normalize our data, we scale the training data set by using Scikit-Learn's MinMaxScaler with numbers between zero and one. To processing the time series data in the form of a 3D array to the LSTM model, we create data in 60 timesteps before using numpy to convert it into an array. Finally, we convert the data into a 3D array with X_train samples, 60 timestamps, and one feature at each step. To split the test data, we modify the test set (notice similarities to the edits we made to the training set): merge the training set and the test set on the 0 axis, set 60 as the time step again, use MinMaxScaler, and reshape data. Then, inverse_transform puts the stock prices in a normal readable format.

For hyper parameters tuning, we used a n_layers x LSTM+ Dropout + Dense architecture for hyper param-

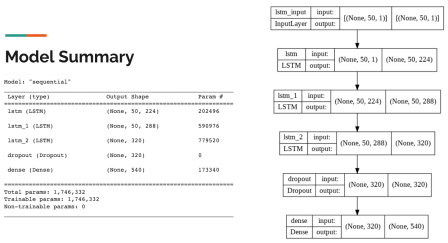

Figure 3: Model Summary

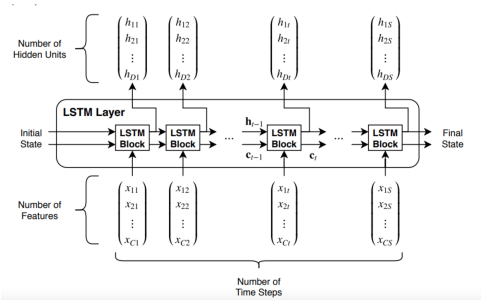

Figure 4: Architecture of LSTM Neural Network

eter tuning. For the first layer, when we reframed our time series to a supervised learning structure, we have created a 3D array. The first dimension is the number of pairs input-output and it is inferred by the layer itself. The second dimension is the number of backward steps, for example, we used the data of the 10 previous days (240 steps) and the third dimension is the number of labels used, in our case 2 (Price in USD, Price in Crypto). The second and third dimensions have to be set explicitly on the first LSTM layer. Furthermore, set return_sequences to True, False is the default value. When we set return_sequences to True, it makes the output usable to another LSTM. If the next layer is not an LSTM, just leave return_sequences by his default value.

For neurons selection, We used the Tuner to select values between 32 and 512 with a step of 32. For Layers selection, we applied trial range() that will take a new value. For example, the first iteration 'n_layer' may take the value 1, which means the loop will have range(1), so we will add 1 LSTM layer, or could take a value of 4 and add 4 layers. Furthermore, to avoid overfitting the neural network, we add a dropout layer. The dropout layer hides neurons randomly, and the number of neurons hidden is set by the dropout rate. If the dropout rate is 0.5, the dropout layer will hide half of the neurons every iteration. Finally, to ensure positive values we chose to use a relu or a sigmoid activation function. We used Keras Tuner's hp.choice() which it allows us to set a list of possible choices from where values can be taken. The best architecture of our LSTM is shown below.

3.5 Quantitative Trading

Each collection's transaction data includes a short period of time. Therefore, we applied our trained LSTM model to generated a whole year price data for over 1000 collections in the market in order to develop our trading strategy.

3.5.1 Mean Variance trading strategy

The first strategy we applied is Mean variance. Mean-variance analysis is one part of modern portfolio theory, which assumes that investors will make rational decisions about investments if they have complete information. One assumption is that investors seek low risk and high reward. There are two main components of mean-variance analysis: variance and expected return. Variance is a number that represents how varied or spread out the numbers are in a set. For example, variance may tell how spread out the returns of a specific security are on a daily or weekly basis. The expected return is a probability expressing the estimated return of the investment in the security. If two different securities have the same expected return, but one has lower variance, the one with lower variance is the better pick. Similarly, if two different securities have approximately the same variance, the one with the higher return is the better pick.In modern portfolio theory, an investor would choose different securities to invest in with different levels of variance and expected return. The goal of this strategy is to differentiate investments, which reduces the risk of catastrophic loss in the event of rapidly changing market conditions. To implement mean variance strategy, we used Efficient Frontier — a set of optimal portfolios providing the highest expected portfolio return for a given level of risk — or framing it differently — providing the minimum level of risk for the expected portfolio return. The Mathematical possible formulations of our portfolio is the following:

$$\min w^T \Sigma w$$
$$s.t. \quad w^T \mathbf{1} = 1$$
$$w \geq 0$$

where w is the vector of weights, is a vector of asset returns, is the covariance matrix, _p is the target expected portfolio return. Two of the constraints are:

non-negative weights 0 short-selling is not allowed

weights must sum up to 1 (no leverage is allowed)

To solve the problem and obtain the Efficient Frontier, we could define a range of possible expected portfolio returns and then for each value find the weights that min-

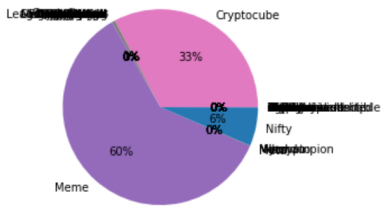

Figure 5: Portfolio Allocation of Mean Variance

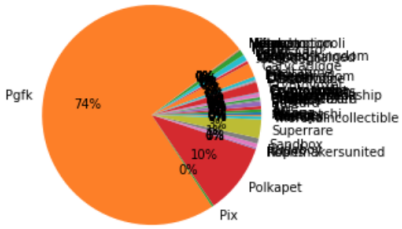

Figure 6: Portfolio Allocation of Hierarchical Risk Parity

imize the variance. We start by creating a simple benchmark strategy — the 1/n portfolio. The idea is that on the first day of the test, we allocate 1/n of our total capital to each of the considered n assets. To keep it simple, we do not do any rebalancing. What often happens in practice is that the portfolio is rebalanced every X days to bring the allocation back to 1/n. We can imagine that we hold a portfolio of two collections X and Y and at the beginning of the investment horizon the allocation is 50–50. Over a month, the price of X increased sharply while the price of Y decreased. As a result, asset X constitutes 65 of our portfolio's worth, while Y has only 35. We might want to rebalance back to 50–50 by selling some of X and buying more Y. Below is a example NFT portfolio allocation of our Mean Variance Strategy.

3.5.2 Hierarchical Risk Parity Strategy

The second strategy we applied is Hierarchical Risk Parity. It is is a method to perform asset allocation without the need to invert a covariance matrix. It essentially computes a hierarchical tree (using a standard hierarchical clustering algorithm) from the correlation matrix, and then diversifies across the different clusters. Our algorithm can be broken down into 3 major steps: Hierarchical Tree Clustering, Matrix Seriation, Recursive Bisection.

For Hierarchical Tree Clustering, we calculate the tree clusters based on the TxN matrix of stock returns where T represents the time series of the data and N represents the number of collections in our portfolio. We combines the items into a cluster rather than breaking down a cluster into individual items it does an agglomerative clustering. Firstly, Given a TxN matrix of collection prices, calculate the correlation of each collections's returns with the other collections which gives us an NxN matrix of these correlations, ρ. Secondly, The correlation matrix is converted to a correlation-distance matrix D, where

$$D(i,j) = \sqrt{0.5 * (1 - \rho(i,j))}$$

Then, we calculate another distance matrix $\overline{D}$, where

$$\bar{D}(i,j) = \sqrt{\sum_{k=1}^{N} (D(k,i) - D(k,j))^2}$$

which is formed by taking the Euclidean distance between all the columns in a pair-wise manner. In order to form clusters of collections using these distances in a recursive manner. We denote the set of clusters as U. The first cluster (i^*, j^*) is calculated as

$$U[1] = \text{argmin}_{(i,j)} \bar{D}(i,j)$$

We now update the distance matrix D by calculating the distances of other items from the newly formed cluster. This step is called linkage clustering and there are different ways of doing this. Hierarchical Risk Parity uses single linkage clustering which means the distances between two clusters is defined by a single element pair – those two elements which are closest to each other.We remove the columns and rows corresponding to the new cluster – in this case we remove rows and columns for stocks a and b.

$$\bar{D}(c,U[1]) = \min(\bar{D}(c,a), \bar{D}(c,b)) \pm \min(21,30) = 21$$
$$\bar{D}(d,U[1]) = \min(\bar{D}(d,a), \bar{D}(d,b)) = \min(31,34) = 31$$
$$\bar{D}(e,U[1]) = \min(\bar{D}(e,a), \bar{D}(e,b)) = \min(23,21) = 21$$

In this way, we go on recursively combining collections into clusters and updating the distance matrix until we are left with one giant cluster of stocks as shown in the following image where we finally combine d with ((a, b), c, e).

For **Matrix Seriation**, We use the order of hierarchical clusters from the previous step, we rearrange the rows and columns of the covariance matrix of collections so that similar investments are placed together and dissimilar investments are placed far apart. This rearranges the original covariance matrix of stocks so that larger covariances are placed along the diagonal and smaller ones around this diagonal and since the off-diagonal elements are not completely zero.

Finally, for **Recursive Bisection**, we split the weights along the dendrogram using naive risk parity (weights

based on the inverse of asset's risk) from the top of the tree to the leafs. Firstly, We initialise the weights of the collections

$$W_i = 1, \forall i = 1, \ldots, N$$

At the end of the tree-clustering step, we were left with one giant cluster with subsequent clusters nested within each other. We now break each cluster into two sub-clusters by starting with the topmost cluster and traversing in a top-down manner. This is where Hierarchical Risk Parity makes use of Step-2 to quasi-diagonalize the covariance matrix and uses this new matrix to recurse into the clusters. Hierarchical tree clustering forms a binary tree where each cluster has a left and right child cluster V_1 and V_2. For each of these sub-clusters, we calculate its variance, where

$$V_{adj} = w^T V w$$
$$w = \frac{\text{diag}[V]^{-1}}{\text{trace}\big(\text{diag}[V]^{-1}\big)}$$

A weighting factor is calculated based on the new covariance matrix:

$$\alpha_1 = 1 - \frac{V_1}{V_1 + V_2}; \alpha_2 = 1 - \alpha_1$$

formula for the weighting factor. It comes from the classical portfolio optimisation theory where we have the following optimisation objective,

$$\min \frac{1}{2} w^T \sigma w$$
$$s.t. e^T w = 1; e = 1^T$$

where w are the portfolio weights and σ is the covariance of the portfolio. For a minimum variance portfolio, the solution to the above equation comes out to be,

$$w = \frac{\sigma^{-1} e}{e^T \sigma^{-1} e}$$

And if σ is diagonal,

$$w = \frac{\sigma_{n,n}^{-1}}{\sum_{i=1} N \sigma_{i,i}^{-1}}$$

Since, we only have N = 2 (2 subclusters) this equation becomes,

$$w = \frac{1/\sigma_1}{1/\sigma_1 + 1/\sigma_2} = 1 - \frac{\sigma_1}{\sigma_1 + \sigma_2}$$

Finally, The weights of stocks in the left and right subclusters are then updated respectively:

$$W_1 = \alpha_1 * W_1$$
$$W_2 = \alpha_2 * W_2$$

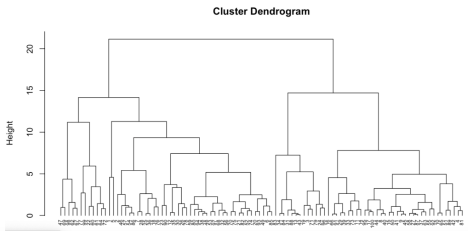

Figure 7: Example of Hierarchical clustering

We recursively execute steps 2-5 on V_1 and V_2 until all the weights are assigned to the stocks. Above is a example NFT portfolio allocation of our Hierarchical Risk Parity Strategy.

3.5.3 Mean Conditional Value at Risk

The final Strategy we used is Mean Conditional Value at Risk. Mean Conditional Value at Risk. Conditional Value at Risk (CVaR) is a statistical technique used to measure the level of financial risk within a firm or an investment portfolio over a specific time frame. CVaR is the expected loss if that worst-case threshold is ever crossed. In order to calculate the Mean Conditional Value at Risk of a portfolio, we takes follow the steps: Firstly, Calculate periodic returns of the collections in the portfolio. Secondly, create a covariance matrix based on the returns and calculate the portfolio mean and standard deviation(weighted based on investment levels of each collection in portfolio). Thirdly, Calculate the inverse of the normal cumulative distribution (PPF) with a specified confidence interval, standard deviation, and mean. Finally, Estimate the value at risk (VaR) for the portfolio by subtracting the initial investment from the calculation.

3.6 Application Realization

Using our trained model, we created a quantitative trading platform where investors to use. When given an input on the amount of US dollar, the application will give out a recommended NFT buying combination for most profit. The application it self is built using *streamlit*, which is an open-sourced library that can build applications for machine learning. In the input page, the application prompts the user to input a certain amount of US dollar. Once the user run the application, it generates four output pages, showing the recommended investment combination, cumulative return graph, daily price change, and the risk plot. See Appendix for more information

4 Evaluation

The evaluation for this project contains two parts: Neural network evaluation and quantitative strategy evaluation.

4.1 LSTM Evaluation

We take six metrics to measure our model performance: Mean Square Error, Root Mean Square Error, Mean Absolute Error, R-Squared ,Bias, and Variance. The result plot is shown below:

[LSTM Evaluation

MAE	**MSE**	**RMSE**
0.006898	0.000275	0.016570
R Squared	**bias**	**Variance**
0.8579	0.841	0.013

For Mean Squared Error, we take the distances from the actual prices of test set to the predicted prices of test set and squaring them. Low Mean Squared Error indicating a general good forecast of our model. Furthermore, the low score of root mean squared error also indicates a good fit for our model. We also calculated the absolute error by getting the value of the difference between the forecasted test set and the actual values. Our low MAE tells us a small error we can expect from the forecast on average. To calculate R squared, we take sum of squares due to regression measures how well the regression model represents the data used for modeling, divided by the total sum of squares measures the variation in the actual data. Our r-squared of 0. 85 reveals that 0.85 of the variability observed in the target variable is explained by the LSTM model. Finally, our variance and bias score extracted by the sklearn indicates that we achieve a balance between the Bias error and the Variance error, the LSTM model neither learns from the noise nor makes sweeping assumptions on the data. The accuracy plot of our model is shown in AppendixI:

4.2 Quantitative Strategy Evaluation

We takes several metrics to evaluation of strategies, the cumulative Returns, Riskplot, and the Sharp Ratio. The Sharpe ratio is a measure of excess portfolio return over the risk-free rate relative to its standard deviation

$$S(x) = \frac{(r_x - R_f)}{Std\,\mathrm{Dev}\,(r_x)}$$

vspace*250px where: $x =$ The investment $r_x =$ The average rate of return of x $R_f =$ The best available rate of return of a risk-free collection $\mathrm{Std\,Dev}\,(r_x) =$ The standard deviation of r_x

Thus the ratio represents the reward per unit of variability. Sharpe Ratios less than 1.0 are considered to be deficient, whereas everything above 1.0 is perceived to be adequate. Investors review a ratio of at least 2.0 to be very good, while a value higher than 3.0 is appraised to be outstanding. Our goal is to compose a portfolio which maximizes the Sharpe Ratio and compare its performance to other frequently applied strategies. When we investing 1000 dollars, the cumulative return of is 840, with sharp ratio of 0.75, indicating the returns of all collections were lower than the risk-free rate. For the Hierarchical Risk Parity Strategy, we got 1227 USD in cumulative return with sharp ratio of 1.79, indicating a good performance that the portfolio is offering excess returns relative to its volatility. For the Mean Conditional Value at Risk, we got 947 USD return with a sharp ratio of 2.51, indicating a very good performance of our strategy

5 Conclusion and Future Work

We explored the total price change into 12 months from year 2018 - 2021. Although some months in 2021 lacks the price data in the dataset we are using, we could tell that from this price trend in year 2018- 2021, increase of total NFT transactions started from 2021 February. And the plots on different categories shown us that Arts are the most popular NFT products. Price of some other categories like Games and Collectable also increased a bit compared to the Art products. We speculated that crazy trading in the Art field also stimulated capitals into other related NFT markets. For price prediction, we trained and tuned the architecture to find the optimized layers and neurons. Furthermore, we also set drop out layers to avoid over-fitting. the LSTM neural network implemented is capable of predicting prices for NFT, achieving 95% accuracy with good balance in bias-variance trade-off. We use the trained neural network to do data augmentation on one-year price data of each collections. Based on the prediction, we implemented three quantitaitve strategies with optimized cumulative returns and sharp ratio. For future work, researchers can consider extracting more comprehensive data set that contains other variables that might effect the price trend, such as ESG, market volumn, etc. And some other deep learning models can be considered or ensemble: Recurrent Neural Network, DNN(Deep Neural Netowrk), etc. For the quantitative trading platform, researchers might consider incorporating the strategies to real NFT trading software to make high frequency trade.

References

[1] DAS, D., BOSE, P., RUARO, N., KRUEGEL, C., AND VIGNA, G. Understanding security issues in the NFT ecosystem. CoRR abs/2111.08893 (2021).

[2] NADINI, M., A. L. D. G. F. E. A. Mapping the nft revolution: market trends, trade networks, and visual features. Sci Rep (2021).

[3] UYGUN, Y., ORUC, M. F., AND SEFER, E. Nft sales price pre- diction using visual feature extraction.

[4] López de Prado, Marcos and López de Prado, Marcos, Building Diversified Portfolios that Outperform Out-of-Sample (May 23, 2016). Journal of Portfolio Management, 2016; https://doi.org/10.3905/jpm.2016.42.4.059.

[5] N. Gorgolis, I. Hatzilygeroudis, Z. Istenes and L. –. G. Gyenne, "Hyperparameter Optimization of LSTM Network Models through Genetic Algorithm," 2019 10th International Conference on Information, Intelligence, Systems and Applications (IISA), 2019, pp. 1-4, doi: 10.1109/IISA.2019.8900675.

[6] Brownlee, Jason. "Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras." Machine Learning Mastery, 27 Aug. 2020, https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/.

[7] Dyer, John Elder, Kevin Mackinnon, Ronald. (2014). A Survey and Discussion of Competing Mean-Variance Statistics in Portfolio Analysis. Journal of Financial Education. 40. 22-55.

[8] Harezlak, Armando Teixeira-Pinto amp; Jaroslaw. "Machine Learning for Biostatistics." 3 Hierarchical Clustering, 26 July 2021, https://bookdown.org/tpinto_home/Unsupervised-learning/hierarchical-clustering.html.

[9] Cota, Nicolas Fermin, et al. "Hierarchical Risk Parity Implementation in RCPP and OpenMP." Rcpp, https://gallery.rcpp.org/articles/hierarchical-risk-parity/.

[10] Harezlak, Armando Teixeira-Pinto amp; Jaroslaw. "Machine Learning for Biostatistics." 3 Hierarchical Clustering, 26 July 2021, https://bookdown.org/tpinto_home/Unsupervised-learning/hierarchical-clustering.html.

[11] Lioudis, Nick. "Understanding the Sharpe Ratio." Investopedia, Investopedia, 3 Jan. 2022, https://www.investopedia.com/articles/07/sharpe_ratio.asp.

Appendix I Exploratory Data An and Quantitative Strategy Evaluation

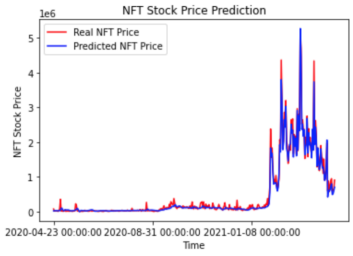

Figure 8: Predicted and Actual price in Test Set

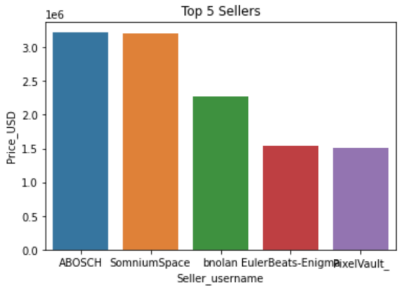

Figure 9: Top 5 Sellers

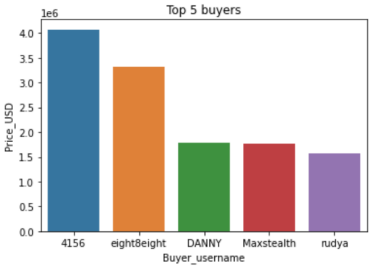

Figure 10: Top 5 Buyers

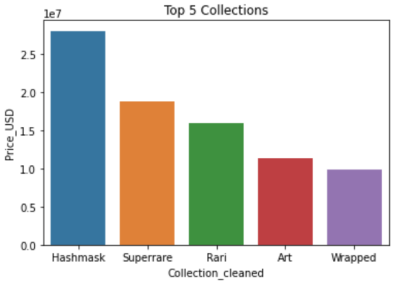

Figure 11: Top 5 Collections

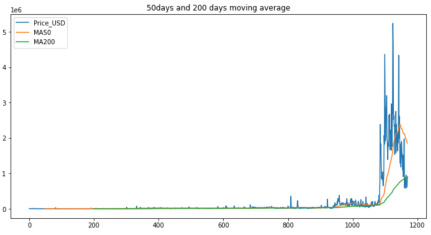

Figure 12: 50 and 200 days moving average

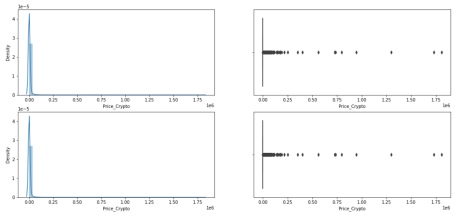

Figure 13: Distribution and Boxplot of Price

Appendix II Web Application Display

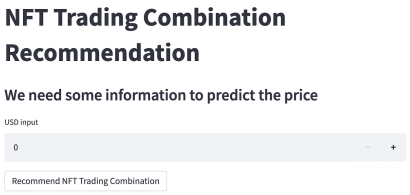

Figure 14: Application Input Page

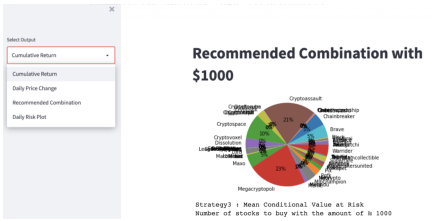

Figure 15: Recommended Combination when input $1000

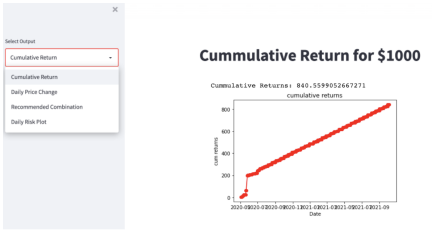

Figure 16: Cumulative Return Page with Recommended Investment

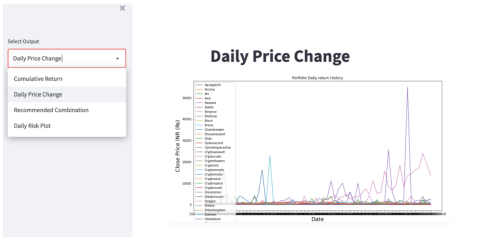

Figure 17: Daily Price Graph

Figure 18: Risk Plot for the Suggested Investment