Advanced Financial Technology

Edward Bickerton
Department of Computer Science
University of Bristol
rw19842@bristol.ac.uk

I. Principal Component Analysis

A. Dataset

The dataset [1] I have chosen for this task consists of daily opening prices of $1,000$ of the top cryptocurrencies (price data originally sourced from [2]). The data spans from the 17th of September 2014 to the 16th of May 2023. Hence, this dataset is very large with $1,000$ features (excluding date) and $3,164$ instances. Fig. 1 shows this data for the two largest cryptocurrencies by market capitalisation: Bitcoin [3] and Ethereum [4].

From fig. 1 we see that both cryptocurrencies are highly volatile, and both appear to be highly correlated, indeed the Pearson product-moment correlation coefficient is 0.923. We also note that price data for Ethereum before roughly 2018 is missing, this is in part due to Ethereum being in a relatively early stage of development up until the Byzantium fork in late 2017 [5]. Likewise, many of the other cryptocurrencies in the dataset do not have price data going back to 2014, resulting in $2,053,223$ missing values (65% of the data).

Thus, I filter the data to contain only price data from 2019 onwards, then cryptocurrencies with more than 10% missing values are removed. Finally, any remaining missing values are filled using backfill. The resulting dataset contains 295 features and $1,597$ instances.

B. Method

Frequently used by data analysts, Principal Component Analysis (PCA) is a method of reducing the dimensionality of complex datasets; that is given a dataset with many features, output another dataset with fewer features. In this way PCA can be used to extract key structures from the original dataset. The concept of redundancy is the key concept behind dimensionality reduction, for example if two features of our dataset are highly correlated then the data can be expressed concisely by removing one of these features. No information is lost since the remaining feature can be used to calculate the removed feature via best-fit line between the two.

If we think of our dataset as a matrix, $M \in \mathbb{R}^{n\times d}$ (n & d are the number of instances and attributes respectively) then PCA can be described simply in terms of linear algebra. PCA is the process of finding new basis vectors such that when M is transformed with respect to these basis vectors, its underlying structure becomes apparent. Note: framing the problem in this way necessarily assumes linearity.

Video presentation available at: http://tiny.cc/68b7vz

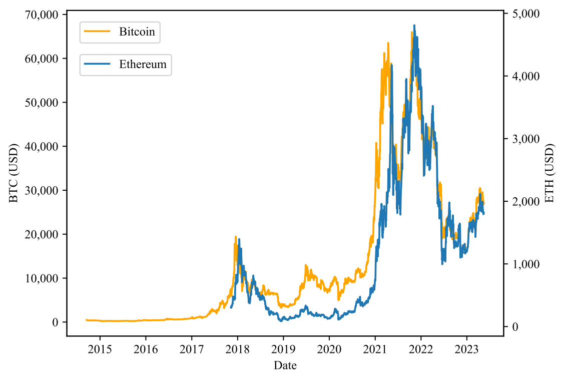

Fig. 1. Price data for Bitcoin and Ethereum in US dollars.

Let $Y \in \mathbb{R}^{n\times d}$ be the resulting matrix of the transformation of M. What does it mean for Y to make the underlying structure of M apparent? Consider the covariance matrix of Y, $C_Y = \{c_{ij}\} \in \mathbb{R}^{d\times d}$ where c_{ij} is the covariance between columns i and j of Y (hence the diagonal terms are the variances of their corresponding column of Y). For the case where the mean of each column is zero,

$$C_Y = \frac{1}{n}Y^TY. \tag{1}$$

PCA assumes that the signal to noise ratio (SNR, the ratio of variances $\sigma^2_{signal}/\sigma^2_{noise}$) is greater than one and thus that the dynamic of interest lies along the line of greatest variance. Thus, the target matrix Y should have a diagonal covariance matrix C_Y, i.e., distinct columns of Y have zero covariance (and hence have zero correlation).

To make this concrete, let $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_d \in \mathbb{R}^d$ be such a set of basis vectors and $P = (\boldsymbol{p}_1 \ \ldots \ \boldsymbol{p}_d) \in \mathbb{R}^{d\times d}$, the matrix consisting of these vectors as columns, then

$$MP = Y \in \mathbb{R}^{n\times d}. \tag{2}$$

The vectors $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_d$ are known as the principal components of M. Assuming the columns of M have zero means then substituting eq. 2 into eq. 1 gives the following:

$$\begin{aligned} C_Y = \frac{1}{n}Y^TY &= \frac{1}{n}\left(MP\right)^T\left(MP\right) \\ &= \frac{1}{n}P^TM^TMP \\ &= P^T\left(\frac{1}{n}M^TM\right)P \\ &= P^TC_MP \end{aligned} \tag{3}$$

where C_M is the covariance matrix of M. Hence the goal of PCA is to find a matrix P such that $P^T C_M P$ is a diagonal matrix.

This problem becomes soluble if we assume that the principal components are orthogonal. By theorem 4 in appendix A of [6] since C_M is symmetric (the covariance of column i and j of M is equal to the covariance of column j and i of M) it can be diagonalised by a matrix of its orthonormal eigenvectors. Hence, if $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_d \in \mathbb{R}^d$ are the orthonormal eigenvectors of C_M and $E = (\boldsymbol{e}_1 \; \ldots \; \boldsymbol{e}_d) \in \mathbb{R}^{d\times d}$ then $\exists$ a diagonal matrix $D \in \mathbb{R}^{d\times d}$ such that $C_M = EDE^T$. Substituting this into eq. 3 gives $C_Y = P^T \left(EDE^T\right) P$. Thus, we can take the principal components to be the orthonormal eigenvectors of C_M i.e., letting $P = E$ (by theorem 1 in appendix A of [6]: $E^T = E^{-1}$). This method of finding the principal components is known as Eigen decomposition [7].

In practice this means that given a dataset with d columns and n rows, if we first subtract the mean from each column and compute the corresponding covariance matrix. We then find the principal components by calculating the eigenvalues of this matrix. When constructing P we order the principal components with respect to descending eigenvalues, since these correspond to the variance of the data in the direction of the principal component. Using P we then compute the matrix Y. The latter columns of Y explain less of the total variance and as such are less significant. Thus, to reduce the dimension of Y you simply remove the latter columns corresponding to insignificant principal components.

C. *Results & Analysis*

Before applying PCA to the cryptocurrency dataset I apply a standard scaler. This removes the mean of each feature as required for PCA and scales each column to have unit variance, ensuring features with large values don't dominate the results.

Fig. 2 demonstrates that PCA succeeded in identifying the strong correlation between the prices of Bitcoin and Ethereum.

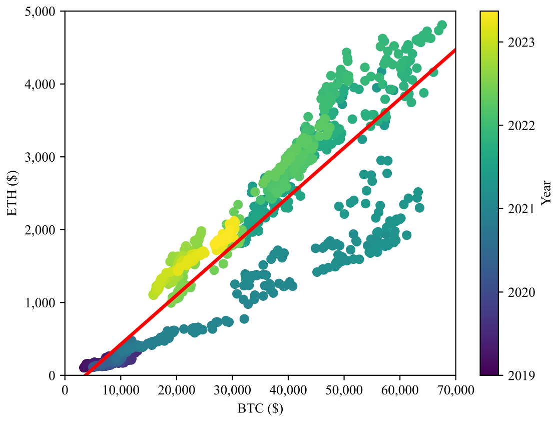

Fig. 2. Scatter plot of ETH against BTC. The red line shows the direction of the first principal component in this 2-dimensional subspace.

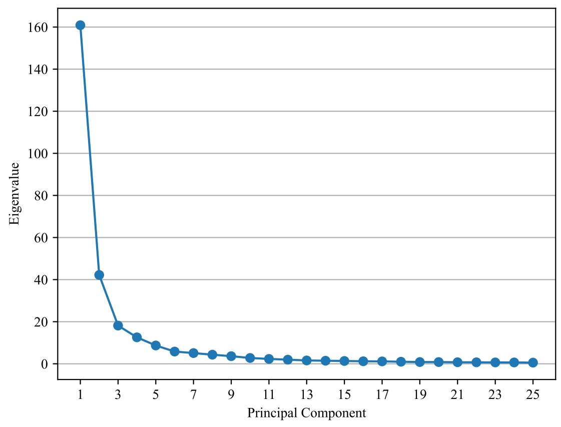

Fig. 3. Scree plot of the cryptocurrency dataset for the first 25 of 295 principal components obtained from Eigen decomposition PCA.

Fig. 3 shows the scree plot for the cryptocurrency dataset, the eigenvalue of the corresponding principal component, equivalent to the variance of the data in the direction of the principal component. We can see that there is significant variation for the first 10 and very little variation for the latter components; indeed, Kaiser's rule, which suggests discarding components whose eigenvalue is less than one (i.e., it contains less information than a single feature of the original dataset), results in a dataset with just 17 features.

In fact, upon observing fig. 4, one could argue that Kaiser's rule has selected too many components and that 5 would suffice. From fig. 4 we see that just 5 components explain over 80% of the price data of the 295 different cryptocurrencies.

D. *Clustering using PCA features*

From inspecting fig. 2 it appears that the points form at least two clusters. To investigate further and demonstrate the utility of PCA, I use a Gaussian mixture model to group the data points, using the 5 features generated by PCA in the previous section.

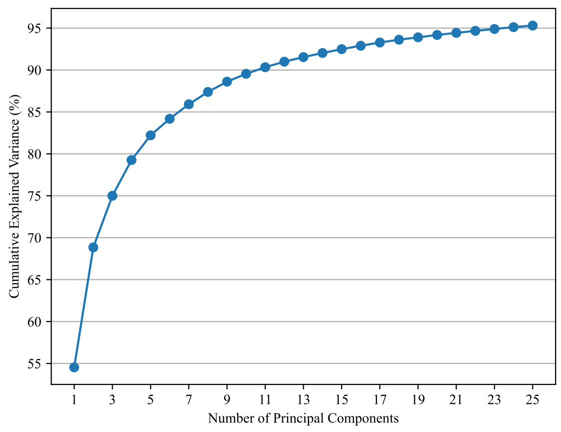

Fig. 4. Cumulative explained variance of the principal components of the cryptocurrency dataset. Explained variance is the eigenvalue as a ratio of the sum of eigenvalues (i.e., total variance).

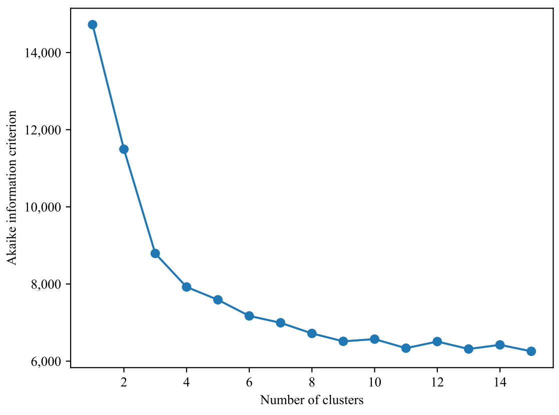

Fig. 5. Elbow plot for Gaussian mixture models.

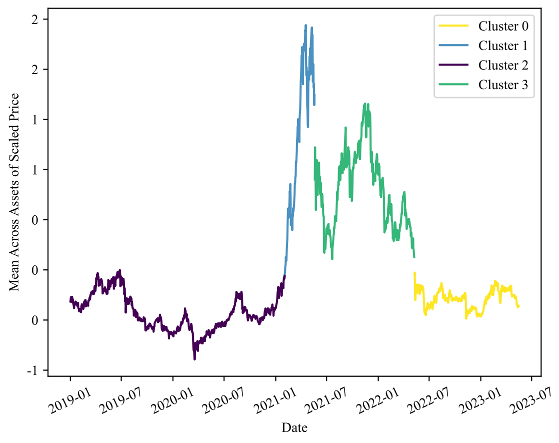

Fig. 7. Average scaled price over all 295 cryptocurrencies. The colours represent the categories given by the Gaussian mixture model applied to the 5 PCA features.

The Gaussian mixture model (GMM) requires the tuning of a key hyper-parameter: the number of clusters to group the data into. Thus, I create a train-test split using 70% of the data as training data and train GMMs with 1 to 15 clusters. Using the Akaike information criterion on the test data to evaluate the performance of each GMM, which can be seen in fig. 5.

From fig. 5 I choose the number of clusters to be four, before training a final GMM of the full dataset, the results of which can be seen in fig. 6 & 7.

From fig. 6 we see that the GMM has identified categories which were not visible upon initial inspection of fig. 2, and fig. 7 shows that the GMM has identified time periods for which the cryptocurrency market behaves very differently. This could be invaluable for predicting price movements, for example for cluster 1 you can expect high volatility with a strong upward trend.

The benefits of applying PCA before the GMM is twofold; it removes noise from the data, and it dramatically reduces the computation expense since the GMM is clustering points in a 5-dimensional space as opposed to 295 dimensions.

II. Dow Jones Industrial Average

Beyond dimensionality reduction, PCA can also be used to generate portfolios by mapping the eigenvectors to portfolio weights.

A. Data Preprocessing

For this task I use price data of the 30 stocks in the Dow Jones Industrial Average (DJIA) from the year 2000 to 2019. Fig. 8 shows the correlation matrix of this dataset, we can see that a significant number of the stocks are highly correlated, many of which are positively correlated.

Before applying PCA I must handle the missing values in the dataset of which there are $6,504$ (4.5%). Upon inspection it appears that some stocks have many missing values, likely because they were added to the DJIA after the start date of the

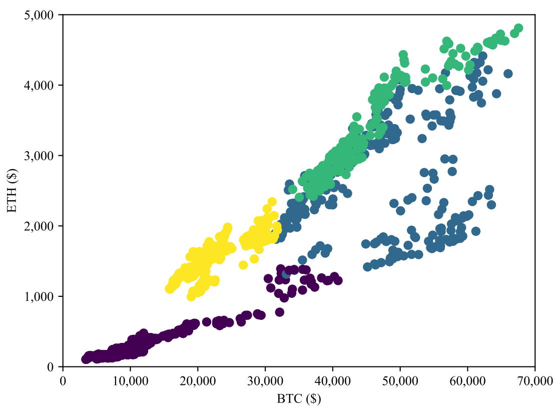

Fig. 6. Scatter plot of ETH against BTC. The colours represent the categories given by the Gaussian mixture model applied to the 5 PCA features.

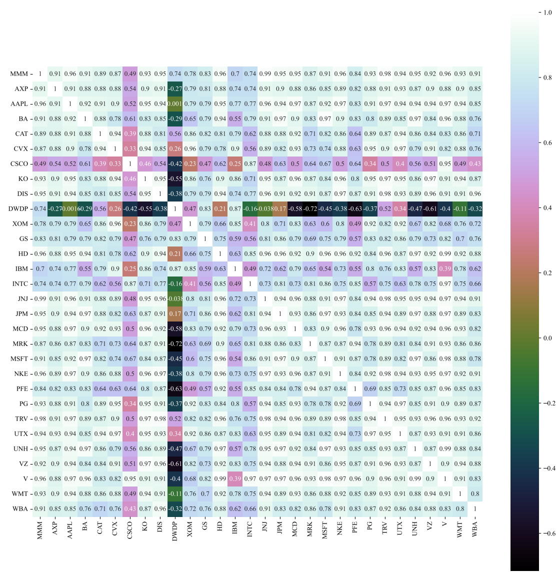

Fig. 8. Correlation matrix of the Dow Jones Industrial Average for the years from 2000 to 2019.

data. Thus, I remove columns which are missing more than 10% of their data, removing stocks with ticker "DWDP" and "V". Remaining missing values are filled in using forward fill.

Since, for the purpose of allocating capital efficiently to a portfolio we are interested in the returns of a stock and not merely their prices, I convert the price data to daily returns as a percentage. I also remove any outliers beyond three standard deviations.

Finally, as done for the cryptocurrency data I use the standard scaler to remove the mean and scale each column to unit standard deviation.

B. Method

Before performing PCA, the data is split into train and test sets to perform the analysis regarding the best portfolio. Using the data from the year 2000 to mid 2015 (80% of the data) for training and the remaining is held-out to perform backtesting on the portfolios.

From fig. 9 we see that the most important component explains over 35% of the variance. The level of correlation between the stocks is reflected by the sharp drop in the explained variance of the principal components.

Now to construct the portfolios, for this I convert the principal components to portfolio weights. Since the principal components are eigenvectors, I refer to these portfolios as Eigen portfolios. The eigenvectors cannot directly be used as although they are orthonormal, they do not sum to one. Thus, I follow traditional normalisation while considering the magnitudes of the variables, i.e., let $\boldsymbol{p} = \{p_i\} \in \mathbb{R}^{28}$ be a principal component we wish to convert to an Eigen portfolio. Then the weights are given by:

$$\omega_i = \frac{|p_i|}{\sum_{n=1}^{28} |p_n|}. \tag{4}$$

Three of the Eigen portfolios can be seen in fig. 10.

C. Evaluation

Intuitively, each Eigen portfolio represents some kind of independent risk factor. Eigen portfolio 1 (corresponding to the 1st principal component) typically corresponds to a systematic risk factor as this portfolio explains the largest ratio of the variance of the DJIA. From fig. 10 we see that the weights of this portfolio are almost even across all stocks. Eigen portfolios 3 and 6 on the other hand correspond to a risk factor associated with a specific industry sector. For example, we see that portfolio 3 assigns more than 10% to each of Exxon Mobil Corporation (XOM) and Chevron Corporation (CVX). These companies are the largest and second largest direct descendants of John D. Rockefeller's Standard Oil.

To evaluate the Eigen portfolios, I calculate their respective Sharpe ratios which can be seen in fig. 11. The Sharpe ratio is a widely used financial metric that measures the risk-adjusted return of an investment or portfolio by considering both the investment's returns and its volatility. A higher Sharpe ratio indicates a better risk-adjusted return, as it means the portfolio generated more return per unit of risk.

However, Sharpe ratio has limitations; it assumes that returns are normally distributed and only considers volatility as a measure of risk. Therefore, to better evaluate the performances of the portfolios it is worth backtesting the portfolios on the test data. The results of back testing can be seen in fig. 12.

As expected, Eigen portfolio 1 performs almost exactly as the equally weighted index. Portfolio 6 marginally outperformed the weighted index, unsurprisingly as this portfolio achieved the highest Sharpe ratio on the training data.

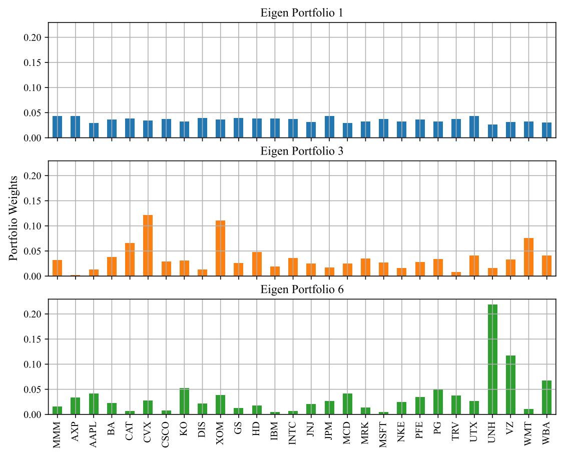

Fig. 10. Three Eigen portfolios composed of stocks in the DJIA.

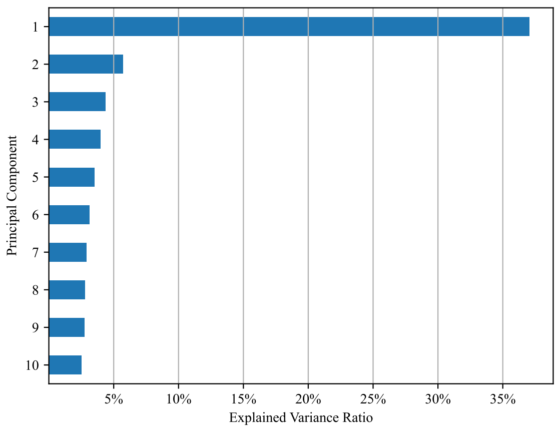

Fig. 9. Explained variance of the top ten principal components of the DJIA.

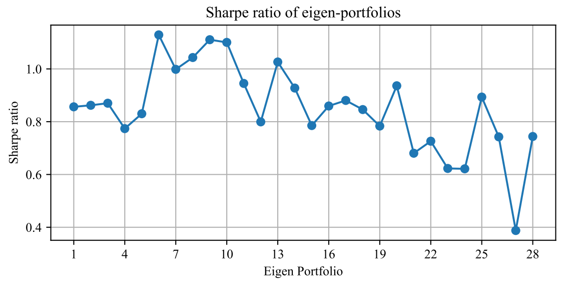

Fig. 11. Sharpe ratios of each of the Eigen portfolios based on training data.

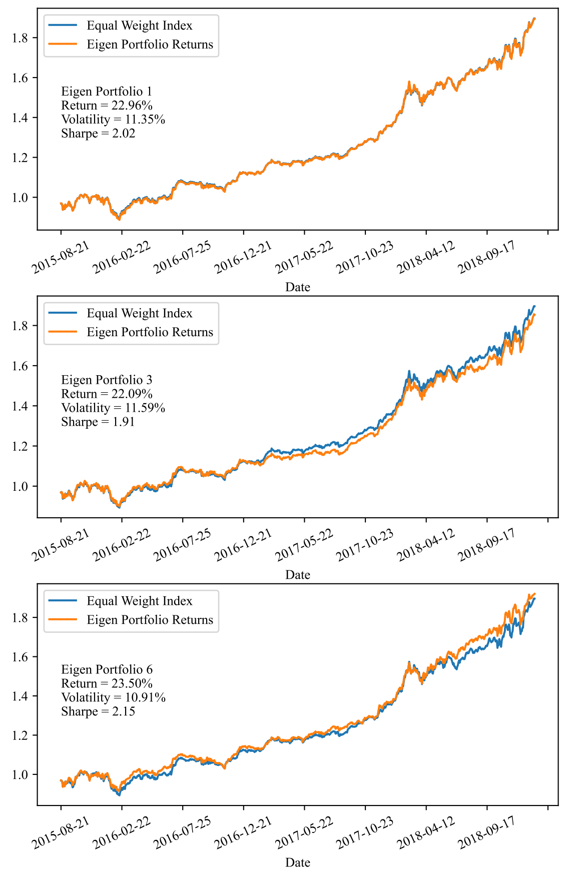

Fig. 12. Returns of the equal weighted index vs. Eigen portfolios.

III. Hierarchical Risk Parity

Hierarchical risk parity (HRP), introduced in [8], is an approach to portfolio management which seeks to allocate risk rather than capital. Portfolio weights are assigned such that all assets contribute the same amount of risk.

A. Data Preprocessing

For this task I will be using a dataset consisting of 30 stocks in the FTSE 100 from 2010 to 2019, three of which can be seen in fig. 13. We can see that all three display an upwards trend, while this is less pronounced for Burberry Group and Diageo, while AstraZeneca appears to be significantly more volatile.

I handle missing values as in section II-A, resulting in stock with ticker "CDI" being dropped and $1,302$ missing values being filled using forward fill. Finally, I calculate the annual returns and create a train-test split with 20% of the data held out for testing.

B. Hierarchical Tree Clustering

This stage of HRP groups similar stocks into clusters based on the correlation matrix. Let $X \in \mathbb{R}^{t \times n}$ represent our dataset where t is number of instances and n is the number of features (in this case stocks) and $C_X = \{\rho_{ij}\} \in \mathbb{R}^{n \times n}$ is the corresponding correlation matrix. From C_X we calculate the correlation-distance matrix D and from that calculate another distance matrix $\bar{D}$:

$$D = \{d_{ij}\} = \sqrt{0.5 \times (1 - \rho_{ij})} \in \mathbb{R}^{n \times n}, \tag{5}$$

$$\bar{D} = \{\bar{d}_{ij}\} = \sqrt{\sum_{k=1}^{n} (d_{ki} - d_{kj})^2} \in \mathbb{R}^{n \times n}. \tag{6}$$

The elements of D, d_{ij} can be interpreted as the distance between two assets i and j, while $\bar{d}_{ij}$ indicates the closeness in similarity of these assets with the rest of the portfolio.

Assets are then clustered in a recursive manner via agglomerative clustering in which each point is initially considered an individual cluster and at each step of the recursion the closest pair of clusters is merged until only one cluster remains. Where closeness is defined using $\bar{D}$, and distance between clusters is defined via a linkage criterion such as single linkage in which the distance between two clusters is defined as the distance between their closest points.

Fig. 14 shows the resulting dendrogram of this clustering; the longer the branches are, the more dissimilar two clusters are. For example, we see that "BDEV" and "BKG" are represented as being close to each other which is expected as both are real estate companies.

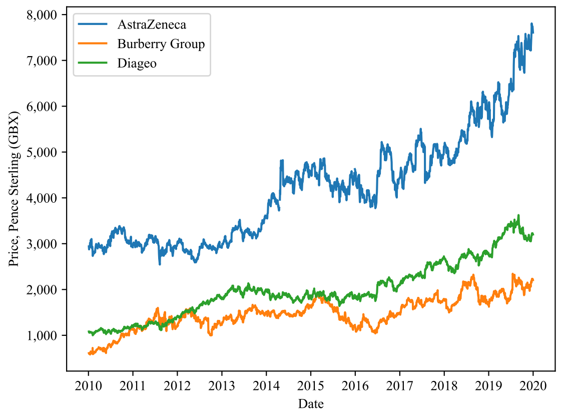

Fig. 13. Price data from three stocks in the FTSE 100.

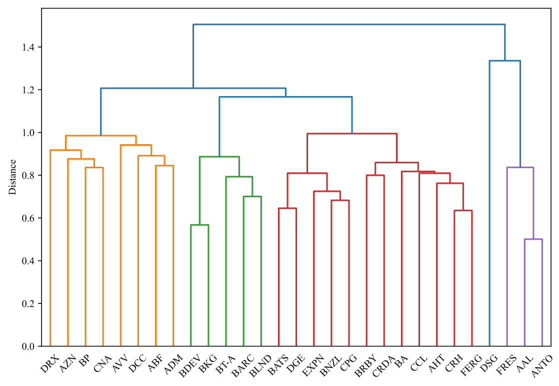

Fig. 14. Dendrogram of our 29 FTSE 100 stocks resulting from Hierarchical Tree Clustering.

C. Quasi-diagonalisation

Also known as matrix seriation, this stage rearranges the covariance matrix such that similar stocks (based on the hierarchical clusters of the previous step) are placed together. The transformed covariance matrix, $\mathcal{V}_X \in \mathbb{R}^{n\times n}$, is known as quasi-diagonal as (unlike in PCA) the off-diagonal elements aren't zero. It allows us to distribute weights optimally following an inverse-variance allocation.

D. Recursive Bisection

The final stage of HRP consists of distributing the allocation through recursive bisection based on the cluster covariance matrix, V_c which is a subset of the rows and columns of $\mathcal{V}_X$ based on the assets in cluster c.

We initialise the weights of all assets to one: $W_i = 1,\ \forall i = 1,\ldots,n$. Since the result of III-B was a binary tree, we iterate through the tree starting at the root cluster which contains all sub clusters and stocks. Let V_L and V_R be the covariance matrices of the corresponding left and right sub-clusters respectively. For each sub-cluster we calculate the expected portfolio variance, v_L and v_R according to eq. 7;

$$v_\Gamma = w_\Gamma^T V_\Gamma w_\Gamma, \tag{7}$$

where, w_Γ is a weight distribution vector,

$$w_\Gamma = \frac{\text{diag}[V_\Gamma]^{-1}}{\text{tr}\left(\text{diag}[V_\Gamma]^{-1}\right)}. \tag{8}$$

The weights of the assets in the left and right sub-clusters are then updated as follows:

$$\begin{aligned} W_l &\leftarrow \frac{v_R}{v_L + v_R} W_l, && \forall l \in \text{left sub-cluster}, \\ W_r &\leftarrow \frac{v_L}{v_L + v_R} W_r, && \forall r \in \text{right sub-cluster}. \end{aligned}$$

These steps are repeated until we reach the leaf nodes.

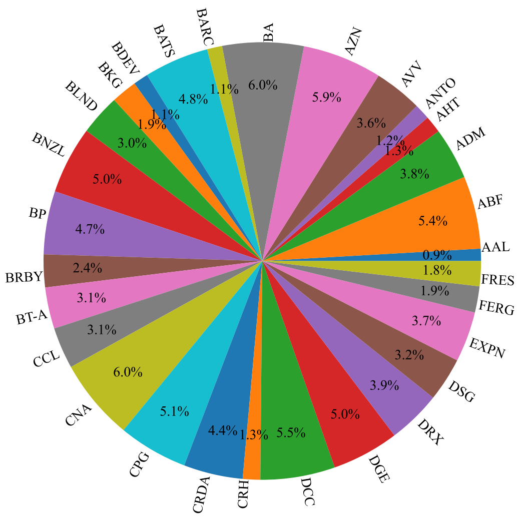

Fig. 15. Portfolio weights of our 29 FTSE 100 stocks derived from HRP.

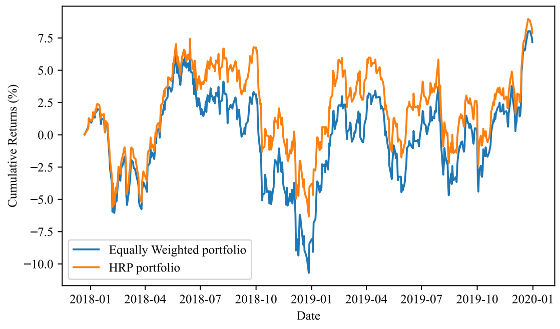

Fig. 16. Performance of HRP portfolio against an equally weighted portfolio on unseen test data.

E. Evaluation

The performance of the HRP portfolio on the held-out test data can be seen in fig. 16. We see that the portfolio derived from HRP consistently outperforms the portfolio constructed simply by taking an equal weighting of each of the 29 stocks. The Sharpe ratios, 0.325 and 0.288 of the HRP and equally weighted portfolios respectively further supports the advantage of HRP.

IV. Challenges of Machine Learning in Portfolio Management

Model interpretability poses a key challenge of using machine learning in portfolio management. The models are often considered to be "black boxes" because their decision-making process is not easily understood, especially for individuals unfamiliar with the algorithms involved. This issue is particularly crucial in portfolio management as portfolio managers may have a fiduciary duty to clients. It is critical that the manager can explain their investment decisions to their clients or even regulatory bodies. Interpretability is also vital for investors to have confidence in the decision-making process. Additionally, when a model is highly interpretable it is easier to identify and correct biases or errors, thus improving the performance of the resulting portfolio.

To mitigate the issue of interpretability, model-agnostic interpretability techniques have been developed. For example, Local Interpretable Model-agnostic Explanations (LIME) introduced in [9], focusses on providing insights into how a particular prediction was made in classification problems. The key idea being to approximate the decision boundary by perturbing the input and observing the corresponding predictions of the perturbed instances. Similarly, SHapley Additive exPlanations (SHAP) introduced in [10] assigns each feature an importance value for a particular prediction.

Rule-based models, such as decision trees offer greater transparency and interpretability since they provide explicit rules for (in this case) allocating weights to assets. Another benefit of these models is that regulatory constraints or expert knowledge can be explicitly baked into the decision-making process.

Finally Explainable AI is an emerging field seeking to develop new machine learning techniques which are inherently interpretable, a brief history of which can be found in [11].

Another challenge is the issue of data quality and availability. Machine learning models, including those utilised in portfolio management, heavily rely on training data to make accurate decisions reliably. However, financial data can be noisy, inconsistent, and as we saw in previous sections, often contains missing values.

It is therefore essential that data preprocessing steps are carried out before training any models. Most machine learning methods cannot handle missing values and so missing values must be either removed or filled in via, for example, mean or regression imputation. Similarly, removing outliers can improve the performance of models. These values can be detected by considering their z-score, the number of standard deviations by which the value is above or below the population mean.

Often, to collect the quantity of financial data required, data scientists necessarily source data from multiple sources. To form a unified dataset, multiple data formats and any inconsistencies must be dealt with. Techniques such as data normalisation and standardisation can be applied to ensure consistency and compatibility.

The utilisation of data in machine learning also raises ethical considerations, such as privacy and unwanted societal biases. Datasheets for Datasets proposed in [12] aims to mitigate these and other issues by improving communication between dataset creators and dataset consumers.

V. Personalised Portfolio Selection

To reduce labour costs associated with lengthy meetings with clients sifting through various investment proposals, the work of a financial advisor can be streamlined or replaced entirely by exploiting automated recommender systems. Such systems aim to address the information explosion that came with the internet, commonly used for recommending music or tv shows, the techniques involved are often domain agnostic thus can be adapted for use in portfolio selection. Ensuring the needs of investors are matched with appropriate financial products.

In this section I give a critical review of the following recent paper, Multi-Task Feature Learning for Knowledge Graph Enhanced Recommendation [13], in which a state-of-the-art recommender system (RS) is presented.

Motivated by the shortcomings of collaborative filtering (CF), a popular technique which utilises historical user-item interactions (such as clicking, watching, or purchasing) to make recommendations, Multi-task feature learning approach for Knowledge graph enhanced Recommendation (MKR) seeks to augment (CF) via the use of external information in the form of a knowledge graph (KG). The issues associated with collaborative filtering are well known, namely the sparsity of user-item interactions and the cold start problem, and the concept of supplementing CF with side information such as social networks, attributes, and multimedia is not novel. In fact, the paper outlines several existing recommender systems which utilise KGs. However, each one is shown to have a downside, for example Personalized Entity Recommendation [14] and Factorisation Machine with Group lasso [15] which are said to have limited utility to generic scenarios due to their reliance on manually designed meta-paths/meta-graphs. Whereas MKR is a generic, end-to-end deep recommendation framework.

Knowledge graphs tend to be preprocessed via knowledge graph embedding (KGE) methods due to their high dimensionality and heterogeneity, in which entities and relations are embedded into a low-dimensional vector space. Rather than treat each task individually, training the KGE and then the RS, the paper recognises that the two tasks are not mutually independent and design a *cross&compress unit* in MKR. The cross&compress unit controls knowledge transfer between the two tasks and hence the representations of items and entities complement each other.

The paper validates its claimed contributions through both a theoretical and empirical approach. For the theoretical analysis, the cross&compress units are shown to be able to model the order of item-entity feature interaction up to exponential degree. Demonstrating their superiority as compared to competing methods. Although, this theoretical approach does not guarantee performance, hence the need for experiments to be designed to test the claims empirically.

Using datasets: MovieLens-1M, Book-Crossing, Last.FM, and Bing-News, MKR is evaluated against four real world recommendation scenarios: movie, book, music, and news. Each dataset is split as 60% training data, 20% for validation (tuning the hyper-parameters) and 20% for testing. The paper uses both AUC and accuracy as performance metrics, reporting the averages of three experiments. The results are impressive, with MKR outperforming each of the baseline models. However, it's worth noting that the hyper-parameter settings of baselines were set as reported in their original papers. Hence the competing models may have achieved better performance if their hyper-parameters were tuned.

The MKR model has many learned parameters, so the interpretability of the model is very poor. I would be interested in applying the mitigating techniques for interpretability discussed in IV such as LIME [9] to MKR and in this way, gain insight into how MKR could be improved.

Interestingly, the cross-knowledge transfer of items was shown to benefit both KGE and RS tasks in MKR. Thus, a further investigation into MKR, focussed on the performance of the KGE task with comparisons to other contemporary KGE methods could be fruitful.

Finally, the current architecture of MKR leaves lots of room for modifications. For example, the extraction of user u's latent feature vector $\boldsymbol{u}_L$ could be performed with more elaborate neural networks (such as CNNs) than the simple, fully connected MLP and different activation functions could be tested. Similarly, variations of predicting and score functions, f_{RS} and f_{KG} could be experimented with.

References

[1] E. Bickerton. Crypto-currency-price-data. [Online]. Available: https://github.com/rw19842/Cryptocurrency-Price-Data/blob/main/cryptocurrency_daily_open.csv

[2] U. Buttar. Cryptocurrency historical prices [updated daily]. [Online]. Available: https://www.kaggle.com/datasets/usamabuttar/cryptocurrency-historical-prices-updated-daily?resource=download

[3] Bitcoin is an innovative payment network and a new kind of money. [Online]. Available: https://bitcoin.org

[4] Ethereum is the community-run technology powering the cryptocurrency ether (eth) and thousands of decentralized applications. [Online]. Available: https://ethereum.org

[5] Byzantium ethereum fork. [Online]. Available: https://ethereum.org/en/history/#byzantium

[6] J. Shlens, "A tutorial on principal component analysis," 2014.

[7] H. Tatsat, S. Puri, B. Lookabaugh, and a. O. M. C. Safari, *Machine Learning and Data Science Blueprints for Finance*, 1st ed. O'Reilly Media, Inc., 2020.

[8] M. L. de Prado, "Building diversified portfolios that outperform out-of-sample."

[9] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?": Explaining the predictions of any classifier," 2016.

[10] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.

[11] F. Xu, H. Uszkoreit, Y. Du, W. Fan, D. Zhao, and J. Zhu, "Explainable ai: A brief survey on history, research areas, approaches and challenges," in *Natural Language Processing and Chinese Computing*, J. Tang, M.-Y. Kan, D. Zhao, S. Li, and H. Zan, Eds. Cham: Springer International Publishing, 2019, pp. 563–574.

[12] T. Gebru, J. Morgenstern, B. Vecchione, J. W. Vaughan, H. Wallach, H. D. I. au2, and K. Crawford, "Datasheets for datasets," 2021.

[13] H. Wang, F. Zhang, M. Zhao, W. Li, X. Xie, and M. Guo, "Multi-task feature learning for knowledge graph enhanced recommendation," 2019.

[14] X. Yu, X. Ren, Y. Sun, Q. Gu, B. Sturt, U. Khandelwal, B. Norick, and J. Han, "Personalized entity recommendation: A heterogeneous information network approach," in *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, ser. WSDM '14. New York, NY, USA: Association for Computing Machinery, 2014, pp. 283–292.

[15] H. Zhao, Q. Yao, J. Li, Y. Song, and D. L. Lee, "Meta-graph based recommendation fusion over heterogeneous information networks," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '17. New York, NY, USA: Association for Computing Machinery, 2017, pp. 635–644.