K-Means Clustering for Factor Investing

Gavin Sun, Jessie Zhang, Sankalp Narula

University of Waterloo

AFM423

Dr. Tony S. Wirjanto

Apr 20, 2025

ABSTRACT

We utilize the data_ml dataset, enhanced with additional macroeconomic variables (the VIX index and credit spreads), to investigate the effectiveness of the k-means cluster-based investment strategy. Our methodology consists of several key steps: data preprocessing, incorporation of new features, normalization of all variables, and dimensionality reduction using autoencoders. After transforming the feature space, we apply K-means clustering to group stocks and backtest portfolio performance across various cluster sizes. Within each cluster, we construct equal-weight portfolios and compare them against an equal-weight portfolio of all stocks in the data set.

1. INTRODUCTION

K-means clustering is a data exploration technique that uncovers underlying structures by partitioning observations into distinct groups. The algorithm begins by randomly selecting the initial centroid, then assigns each observation to the nearest cluster based on distance. It proceeds by recalculating the centroids for each cluster and reassigning observations iteratively, repeating this process until the centroids stabilize.

In our study, we expand a conventional stock-level dataset by incorporating two critical macroeconomic variables: the VIX (Volatility Index), representing market volatility, and credit spreads, reflecting financial stress. By creating interaction terms between these macroeconomic indicators and stock-specific characteristics, our aim is to capture how stock sensitivities shift under different economic conditions (Wirjanto, 2025b). After applying an autoencoder for dimensionality reduction to distill the most informative features, we employ k-means clustering to group stocks based on their financial behaviours. We then construct equal-weighted portfolios within each cluster.

The performance of these cluster-based portfolios is evaluated through backtesting, with particular emphasis on Sharpe ratios as a measure of risk-adjusted returns.

Specifically, the average return of the portfolio is given by:

$$\bar{r}_p = \mu_p = \mathbb{E}(r_{p)} \approx \frac{1}{T}\sum_{t=1}^{T} r_t^p$$

where r_t^p is the return at time t, μ_p is the mean return, and T is the number of periods. The Sharpe ratio, which measures the risk-adjusted return of the portfolio, is then defined as:

$$SR = \frac{\mu_P}{\sigma_P}$$

2. RESEARCH QUESTIONS

We have two main research questions: In what ways can the k-means clustering method serve as a factor investing approach to improve risk-adjusted returns while taking into account macroeconomic variables like VIX and credit spread? How does the number of clusters influence the portfolio selection and risk-adjusted returns?

These research questions are essential to understanding the core contributions of our study. Including the VIX and credit spread variables enriches the dataset by capturing market volatility and financial stress, allowing the model to uncover more nuanced stock behaviours under different economic conditions. This leads to more meaningful clustering results and improved portfolio performance. Investigating k-means clustering as an approach to factor investing is important because it enables the identification of latent patterns in stock characteristics without relying on traditional, linear factor models. By applying clustering, we systematically group stocks in a way that enhances risk-adjusted returns.

3. VARIABLES AND MEASURES

The dataset used in this project comprises monthly observations of 1,207 publicly listed U.S. stocks for the period from November 30, 1998, to March 31, 2019.

The original dataset includes stock characteristics such as average daily volume (e.g., Advt_12M_Usd), profitability metrics (e.g., Ebit_Ta, Roe), valuation ratios (e.g., Pb, Pe), and various measures of leverage, turnover, and momentum.

For the purposes of this study, we removed the label columns (R1M_Usd, R3M_Usd, R6M_Usd, and R12M_Usd) that represent forward-looking returns and serve as targets in typical supervised learning setups. Our analysis focuses exclusively on the feature variables, which we augmented to create a richer representation of stock behaviour under market stress.

To this end, we introduced a variable transformation strategy: we generated new interaction variables by multiplying each original feature with exogenous macro-financial variables such as the credit spread and the VIX. These exogenous factors were chosen due to their known relationships with market uncertainty and financial conditions. This transformation aims to capture nonlinear effects and conditional sensitivities of stock characteristics to broader economic states.

The final feature matrix consists of the original 93 stock features along with two additional sets of interaction terms: one set formed by multiplying each stock variable with the credit spread and another set formed by multiplying each variable with the VIX index. This expansion increased the number of input features by threefold.

$$\bar{x}_{i,t}^{(k)} = z_t \times x_{i,t}^{(k)}$$

Where $x_{i,t}^{(k)}$ is a time-t value of the kth attribute/characteristic of asset i. z_t be the exogenous indicator, multiplying the two results in the new predictor $\bar{x}_{i,t}^{(k)}$. All numeric variables were then normalized using min-max scaling to ensure uniformity in scale and to improve model performance and interpretability.

After feature engineering and normalization, we employed a deep autoencoder to reduce the high-dimensional feature matrix to a compact representation of 30 latent variables. This dimensionality reduction step was essential to address redundancy, avoid overfitting, and capture the most informative patterns in the data while having exogenous economic data in our features. The final dataset retains the original panel structure (indexed by stock ID and date) and remains free of missing values, though the number of stocks per month may vary due to changes in listing over time.

4. THE APPLICATION OF THE ML APPROACH TO FACTOR INVESTING

This project applies machine learning techniques to uncover latent structure in stock data and use that structure to construct systematic investment portfolios. The central objective is to investigate whether clustering stocks based on high-dimensional financial characteristics enriched with macroeconomic context can produce economically meaningful groupings that translate into superior risk-adjusted returns. We specifically evaluate the performance of k-means clustering applied to a reduced-dimensional embedding of the original feature set, with each resulting cluster interpreted as a standalone investment portfolio.

Using this reduced data, we perform k-means clustering for multiple values of k, aiming to discover natural groupings of stocks with similar latent characteristics. Each cluster is treated as an equal-weight portfolio, stocks within each cluster receive equal allocation, and all clusters are combined with equal weight in the final aggregated portfolio.

To evaluate the practical utility of this approach, we implement a backtesting framework that computes cumulative returns and risk metrics (such as Sharpe ratio, annualized volatility, and maximum drawdown) across different values of k. This allows us to compare the stability and profitability of different cluster structures. Our aim is to determine the number of clusters that yields the most compelling performance profile, balancing diversification with alpha generation.

This application highlights how machine learning, particularly unsupervised learning and dimensionality reduction, can enhance traditional asset selection processes. By capturing complex, nonlinear relationships and economic sensitivities, our model-driven approach provides a scalable and adaptive method for portfolio construction in modern financial markets.

5. EXPERIMENTAL METHODOLOGY

The following section outlines the key steps taken in this paper to implement a clustering-based strategy using K-means clustering, with feature reduction via autoencoders. All relevant R code is provided in the appendix for reproducibility.

Data Preprocessing

We begin with a dataset of 1,207 U.S.-listed stocks covering the period from November 30, 1998, to March 31, 2019. The data is cleaned for missing and outlier values. The dataset is split into a training set (pre-January 1, 2010) and a testing set (post-January 1, 2010).

Additional Features and Normalization

VIX and credit spread are computed and appended to the dataset to increase the size of the dataset. All features are normalized using min-max normalization to ensure the scale of the values for a feature does not affect the final results.

Feature Reduction via Autoencoder

To enhance clustering effectiveness and reduce noise in the dataset, we use an autoencoder to compress the original set of financial features into 30 dimensions. By shrinking the dataset, the autoencoder improves performance by retaining the most important underlying patterns while removing most of the noise. This dimensionality reduction allows K-means clustering to operate on a more compact and informative feature space, resulting in more meaningful stock groupings and ultimately leading to better portfolio construction.

K-Means Clustering, Backtesting, and Portfolio Construction

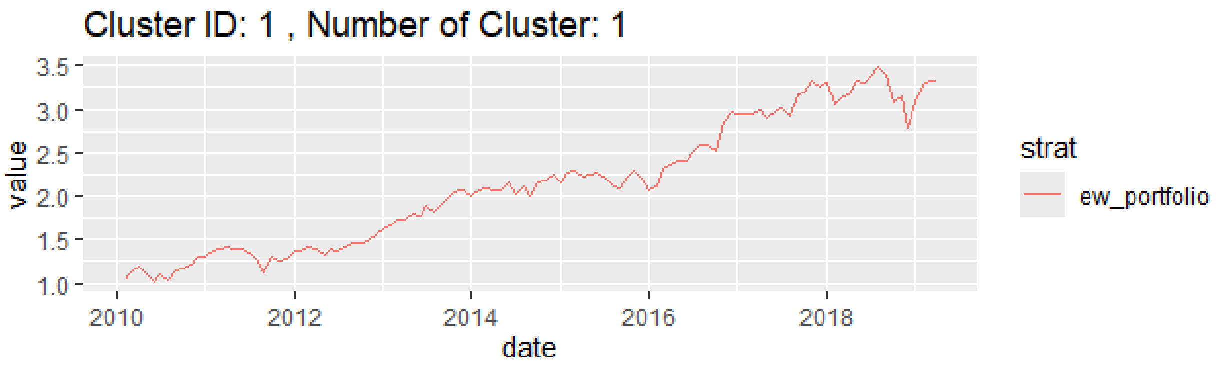
K-means clustering is applied to the compressed data across a range of cluster sizes. For each value of k, stocks are partitioned into k clusters, and equal-weight portfolios are constructed for each cluster. Portfolio returns are computed over the entire period, and backtesting is performed on each cluster individually to evaluate its performance.

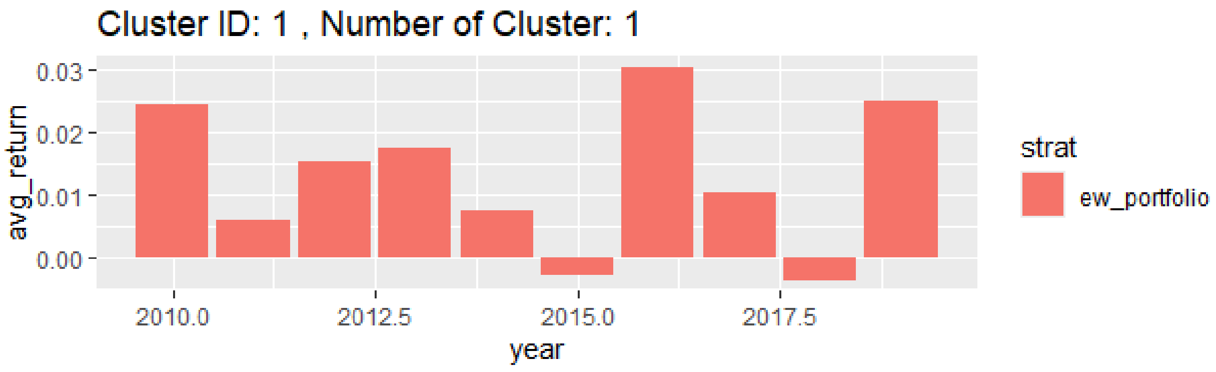
Portfolio Selection and Benchmarking

For each cluster size, we identify the best-performing cluster portfolio based on Sharpe ratio and invest in it as our strategy. The best-performing cluster size is then selected based on the highest Sharpe ratio. This strategy is then benchmarked against an equal-weight portfolio made up of all the stocks in the dataset to assess the effectiveness of the clustering-based strategy in producing higher risk-adjusted returns.

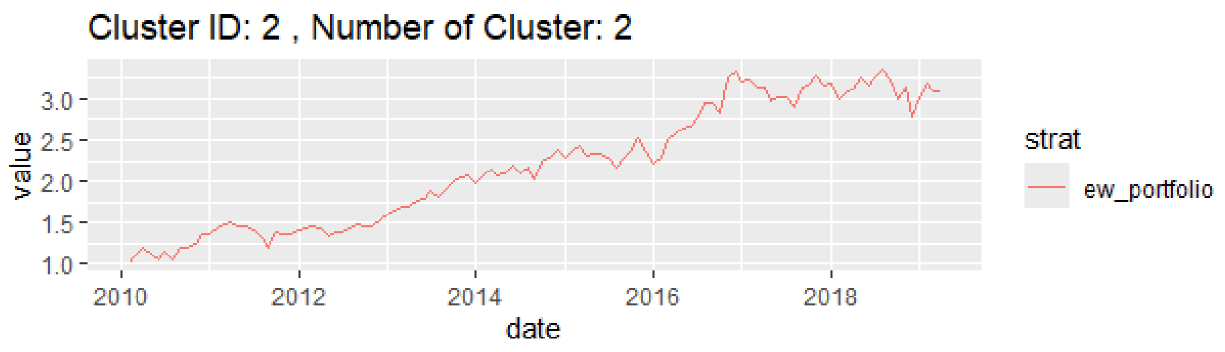
6. RESULTS AND DISCUSSION

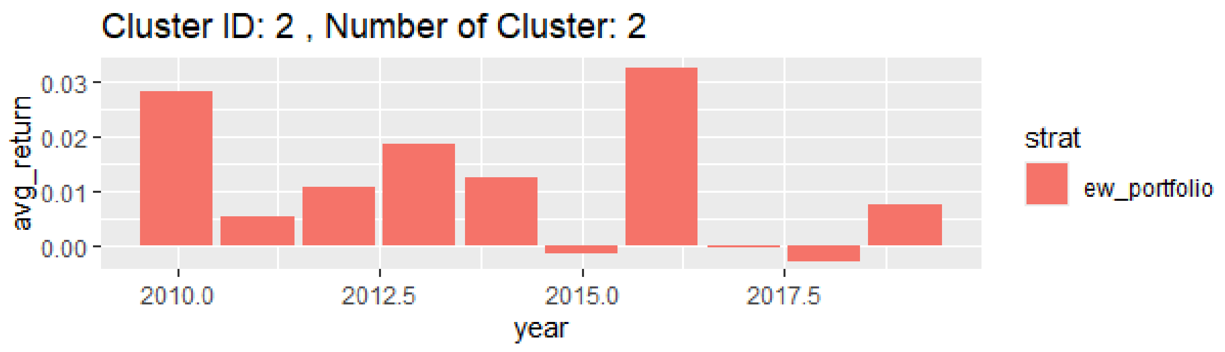
Our objective was to evaluate whether k-means clustering, applied to dimensionally reduced stock data, could be used to construct portfolios with superior risk-adjusted returns. To test this, we visualized and analyzed performance across various cluster sizes, ultimately evaluating whether a cluster-based strategy could outperform the market.

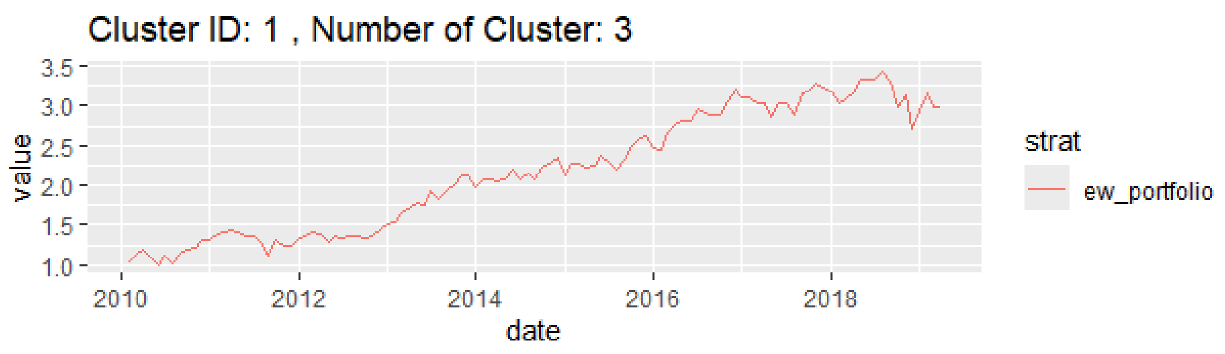


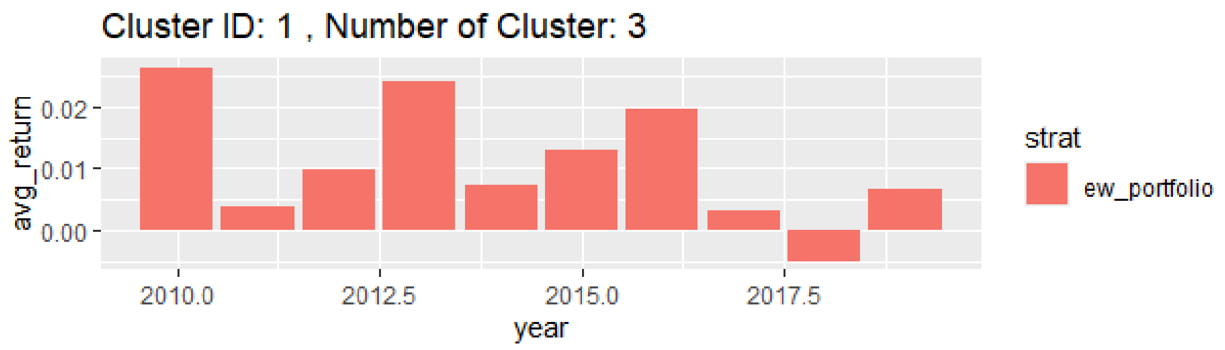


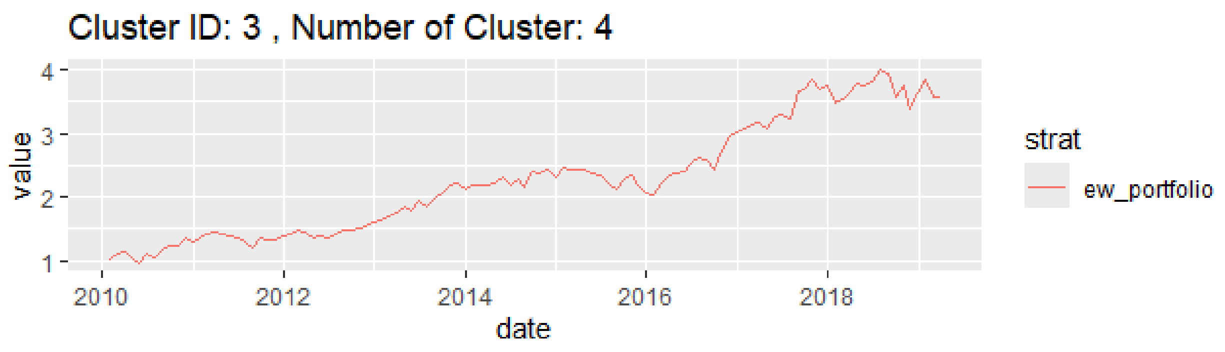
Before looking at the cluster-based results, we look at the following graphs above that show the cumulative return and average annual return of the benchmark portfolio over the test period as a baseline for comparison.

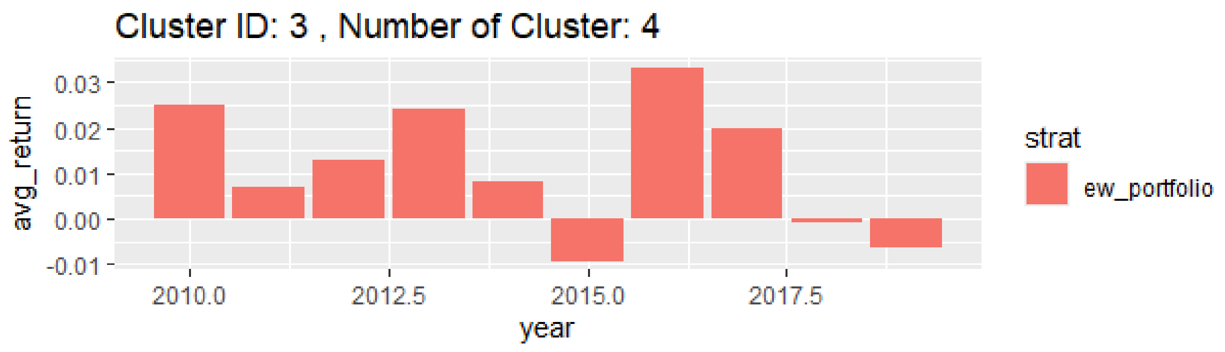

Cluster ID: 2 , Number of Cluster: 2
value
3.0
2.5
2.0
1.5
1.0
2010
2012
2014
2016
2018
date
strat
ew_portfolio

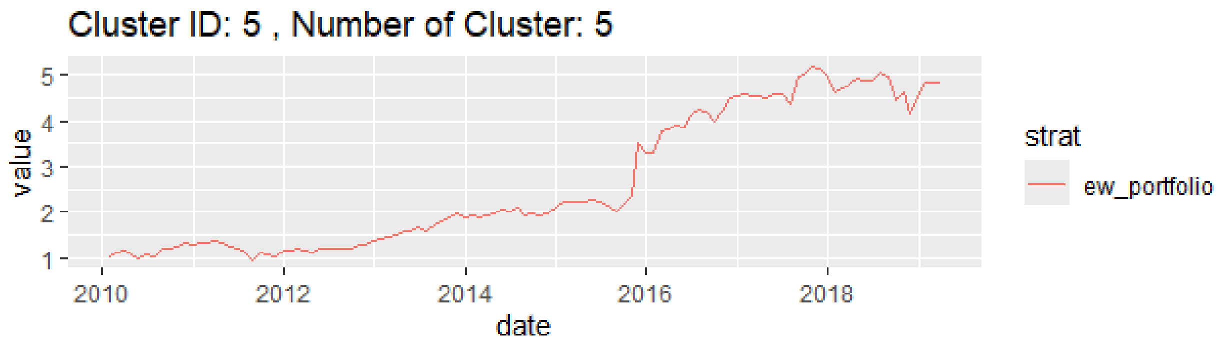

Cluster ID: 2 , Number of Cluster: 2
avg_return
0.03
0.02
0.01
0.00
2010.0
2012.5
2015.0
2017.5
year
strat
ew_portfolio

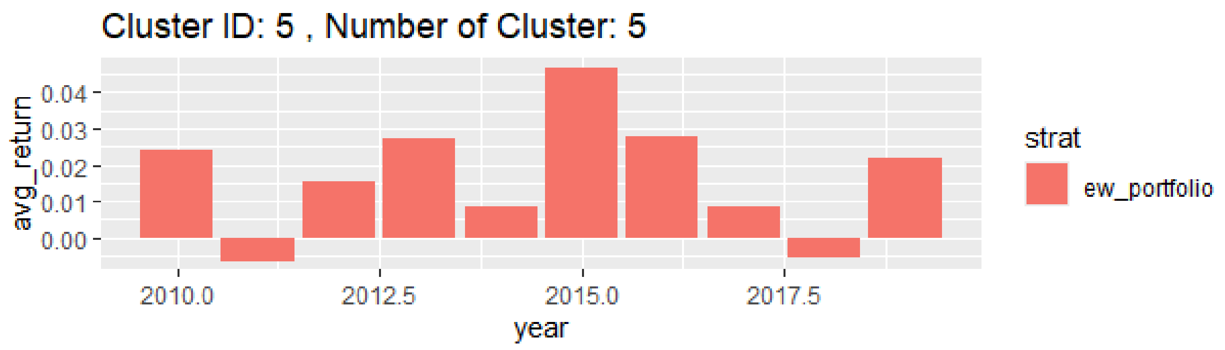

Cluster ID: 1 , Number of Cluster: 3
value
3.5
3.0
2.5
2.0
1.5
1.0
2010
2012
2014
2016
2018
date
strat
ew_portfolio


Cluster ID: 1 , Number of Cluster: 3
avg_return
0.02
0.01
0.00
2010.0
2012.5
2015.0
2017.5
year
strat
ew_portfolio



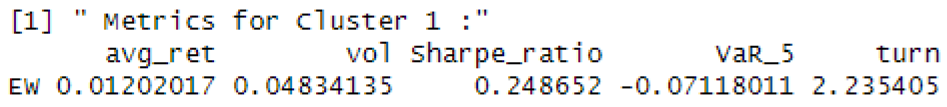






The graphs above display the cumulative returns and average annual returns of the best-performing cluster for each value of k, plotted over the full test period. We observe that certain clusters, particularly at k = 4 & k = 5, deliver higher cumulative returns than the benchmark portfolio. This suggests that k-means is able to find groups that outperform.

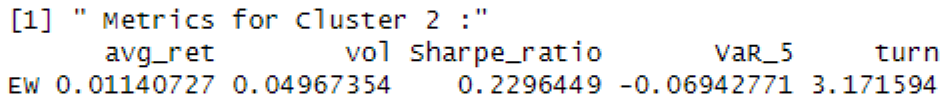
For the benchmark portfolio, the metrics are the following:



```
[1] " Metrics for Cluster 1 :"
      avg_ret        vol Sharpe_ratio       VaR_5     turn
EW 0.01202017 0.04834135     0.248652 -0.07118011 2.235405
```

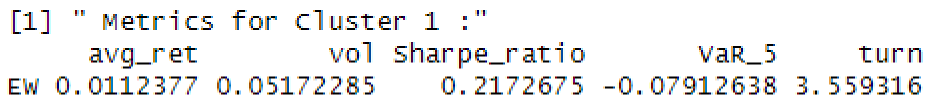
Before analyzing the Sharpe ratios of the cluster-based strategies, we refer to the table above showing the average monthly return, standard deviation, and Sharpe ratio of the benchmark portfolio, which serves as a baseline for evaluating risk-adjusted performance.

For Clustering with 2 clusters, the metrics for the best-performing cluster are the following:



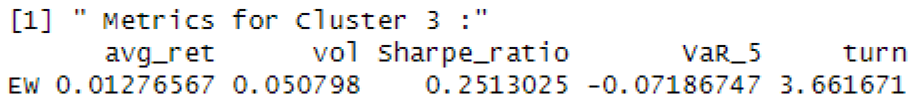
```
[1] " Metrics for Cluster 2 :"
      avg_ret        vol Sharpe_ratio       VaR_5     turn
EW 0.01140727 0.04967354    0.2296449 -0.06942771 3.171594
```

For Clustering with 3 clusters, the metrics for the best-performing cluster are the following:



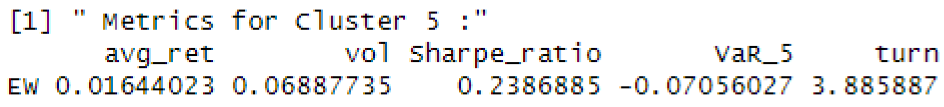
```
[1] " Metrics for Cluster 1 :"
     avg_ret        vol Sharpe_ratio       VaR_5     turn
EW 0.0112377 0.05172285    0.2172675 -0.07912638 3.559316
```

For Clustering with 4 clusters, the metrics for the best-performing cluster are the following:



```
[1] " Metrics for Cluster 3 :"
      avg_ret      vol Sharpe_ratio       VaR_5     turn
EW 0.01276567 0.050798    0.2513025 -0.07186747 3.661671
```

For Clustering with 5 clusters, the metrics for the best-performing cluster are the following:



```
[1] " Metrics for Cluster 5 :"
      avg_ret        vol Sharpe_ratio       VaR_5     turn
EW 0.01644023 0.06887735    0.2386885 -0.07056027 3.885887
```

The tables above show the average monthly returns, standard deviations, and Sharpe ratios of the best-performing cluster for each value of k. These tables directly address our research question regarding whether cluster-based strategies could improve risk-adjusted returns. We observe that only the best-performing cluster for k = 4 has a Sharpe ratio that exceeds the benchmark portfolio, with a Sharpe ratio of 0.251. This suggests that the clustering approach uncovers hidden patterns that contribute to improved risk-adjusted performance, supporting the idea that meaningful stock groupings can emerge from clustering.

In conclusion, our results support the hypothesis that k-means clustering can be effectively applied to portfolio construction. By grouping stocks with similar return characteristics and selecting the top-performing cluster, we demonstrate how machine learning can find different stock groupings and enhance portfolio returns. Our results show how clustering can be used for portfolio construction to produce higher risk-adjusted returns.

7. FUTURE IMPROVEMENTS

While our current strategy focuses on evaluating equal-weight portfolios formed from unsupervised clustering, there are several meaningful extensions that could enhance the effectiveness and sophistication of our approach. One particularly promising direction is the incorporation of dynamic strategies based on the historical performance of clusters. Specifically, for each rebalancing period and given a fixed number of clusters (k), we can compute the rolling 6-month performance of each cluster portfolio. This would transform the strategy from a static, equal-weight allocation to a momentum-driven, cross-cluster approach, potentially enhancing risk-adjusted returns while remaining market-neutral (Wu, Wang, & Wu, 2022).

This extension would also allow us to explore regime-switching behaviour, as different clusters may perform better or worse in varying macroeconomic or volatility environments. By incorporating additional signals, such as changes in credit spread, VIX, or macroeconomic indicators, we could condition our allocation logic and improve timing (Zhang et al., n.d.).

Furthermore, we could consider adaptive clustering, where the optimal number of clusters is not fixed but selected dynamically based on recent in-sample performance or structural changes in the latent feature space. Advanced clustering methods (e.g., Gaussian Mixture Models or spectral clustering) and soft-clustering approaches could also be explored to allow for more nuanced portfolio construction (Vijayalakshmi Pai & Michel, 2009b).

Finally, we plan to evaluate transaction costs and turnover rates, particularly as frequent reallocation across clusters may introduce friction. Incorporating these costs would provide a more realistic assessment of the strategy's viability in live trading scenarios.

8. CONCLUSION

Our study explores the use of k-means clustering as a systematic method for factor investing. The application of clustering on a dimensionally reduced dataset reveals that distinct groupings of stocks can be identified, and these groupings exhibit different performance. In particular, selecting portfolios based on the highest Sharpe ratios within clusters leads to improved risk-adjusted returns compared to traditional equal-weight portfolios. This finding highlights the potential of unsupervised machine learning methods to uncover latent structures in financial data that are not easily captured by traditional factor models.

Lastly, the Sharpe ratio is crucial in our portfolio selection process. By emphasizing portfolios with higher Sharpe ratios, we prioritize strategies that offer superior returns relative to their risk, thus adhering to a more disciplined and quantitatively driven investment approach. We find that the best-performing cluster for $k = 4$ achieves a Sharpe ratio higher than that of the equal-weighted portfolio comprising all stocks in the dataset over the same period. Future research can extend this framework by incorporating dynamic clustering, transaction costs, and adaptive strategies based on changing market regimes, further enhancing the practical applicability of machine learning in portfolio management.

APPENDIX

Code:

```r
if(!require(cowplot)){install.packages("cowplot")}
if(!require(dplyr)){install.packages("dplyr")}
if(!require(keras)){install.packages("keras")}
if(!require(lubridate)){install.packages("lubridate")}
if(!require(quantmod)){install.packages("quantmod")}
if(!require(tensorflow)){install.packages("tensorflow")}
if(!require(tidyverse)){install.packages("tidyverse")}

library(cowplot)
library(dplyr)
library(keras)
library(lubridate)
library(quantmod)
library(tensorflow)
library(tidyverse)

load("data_ml.RData")

# Helper function to normalize numeric columns
min_max_norm <- function(x) {
  if (is.numeric(x)) {
    return((x - min(x, na.rm = TRUE)) / (max(x, na.rm = TRUE) - min(x, na.rm = TRUE)))
  } else {
    return(x)
  }
}

data_ml <- data_ml %>%
  select(-c(R3M_Usd, R6M_Usd, R12M_Usd))
```

```r
add_vix <- function(data, features_short){

  getSymbols.FRED("VIXCLS", env = ".GlobalEnv", return.class = "xts")

  vix <- fortify(VIXCLS)
  colnames(vix) <- c("date", "vix")

  vix <- vix %>% # Take extraction and...
         full_join(data %>% dplyr::select(date), by = "date") %>% # merge data
         mutate(vix = na.locf(vix)) # Replace NA by previous
  vix <- vix[!duplicated(vix),] # Remove duplicates

  # data_cond <- data %>%
  #              dplyr::select(c("stock_id", "date", features_short, "R1M_Usd"))

  data_cond <- data

  names_vix <- paste0(features_short, "_vix")

  feat_vix <- data_cond %>%
                    dplyr::select(all_of(features_short))
  vix <- data %>%
              dplyr::select(date) %>%
              left_join(vix, by = "date")

  feat_vix <- feat_vix *
                    matrix(vix$vix,
                    length(vix$vix),
                    length(features_short))
  colnames(feat_vix) <- names_vix


  data_cond <- bind_cols(data_cond, feat_vix)

  return(data_cond)

}
```

```
add_credit_spread <- function(data, features_short){

  getSymbols.FRED("BAMLC0A0CM",
  env = ".GlobalEnv",
  return.class = "xts")
  head(BAMLC0A0CM)

  cred_spread <- fortify(BAMLC0A0CM)
  colnames(cred_spread) <- c("date", "spread")
  cred_spread <- cred_spread %>%
  full_join(data %>% dplyr::select(date), by = "date") %>%
  mutate(spread = na.locf(spread))
  cred_spread <- cred_spread[!duplicated(cred_spread),]


  # data_cond <- data %>%
  #              dplyr::select(c("stock_id", "date", features_short, "R1M_Usd"))

  data_cond <- data

  names_cred_spread <- paste0(features_short, "_cred_spread")

  feat_cred_spread <- data_cond %>%
                       dplyr::select(all_of(features_short))
  cred_spread <- data %>%
                dplyr::select(date) %>%
                left_join(cred_spread, by = "date")

  feat_cred_spread <- feat_cred_spread *
                        matrix(cred_spread$spread,
                        length(cred_spread$spread),
                        length(features_short))
  colnames(feat_cred_spread) <- names_cred_spread


  data_cond <- bind_cols(data_cond, feat_cred_spread)

  return(data_cond)

}
```

```r
turnover <- function(weights, asset_returns, t_oos){
  turn <- 0
  for(t in 2:length(t_oos)){
    realised_returns <- returns %>% filter(date == t_oos[t]) %>% dplyr::select(-date)
    prior_weights <- weights[t-1,] * (1 + realised_returns) # Before rebalancing
    turn <- turn + apply(abs(weights[t,] - prior_weights/sum(prior_weights)),1,sum)
  }
  return(turn/(length(t_oos)-1))
}

perf_met <- function(portf_returns, weights, asset_returns, t_oos){
  avg_ret <- mean(portf_returns, na.rm = T)                     # Arithmetic mean
  vol <- sd(portf_returns, na.rm = T)                           # Volatility
  Sharpe_ratio <- avg_ret / vol                                 # Sharpe ratio
  VaR_5 <- quantile(portf_returns, 0.05)                        # Value-at-risk
  turn <- 0                                                     # Initialisation of turnover
  for(t in 2:dim(weights)[1]){
    realized_returns <- asset_returns %>% filter(date == t_oos[t]) %>% dplyr::select(-date)
    prior_weights <- weights[t-1,] * (1 + realized_returns)
    turn <- turn + apply(abs(weights[t,] - prior_weights/sum(prior_weights)),1,sum)
  }
  turn <- turn/(length(t_oos)-1)                                # Average over time
  met <- data.frame(avg_ret, vol, Sharpe_ratio, VaR_5, turn)    # Aggregation of all of this
  rownames(met) <- "metrics"
  return(met)
}

perf_met_multi <- function(portf_returns, weights, asset_returns, t_oos, strat_name){
  J <- dim(weights)[2]              # Number of strategies
  met <- c()                        # Initialization of metrics
  for(j in 1:J){                    # One very ugly loop
    temp_met <- perf_met(portf_returns[, j], weights[, j, ], asset_returns, t_oos)
    met <- rbind(met, temp_met)
  }
  row.names(met) <- strat_name      # Stores the name of the strat
  return(met)
}

portf_compo <- function(train_data, test_data){
  N <- test_data$stock_id %>%              # Test data dictates allocation
    factor() %>% nlevels()
  w <- 1/N                                 # EW portfolio
  w$weights <- rep(w,N)
  w$names <- unique(test_data$stock_id)    # Asset names
  return(w)
}
```

```r
features_short <- select(data_ml, -c(stock_id, date, R1M_Usd)) %>% colnames()

data_ml <- add_credit_spread(data_ml, features_short)
data_ml <- add_vix(data_ml, features_short)

features_short <- select(data_ml, -c(stock_id, date, R1M_Usd)) %>% colnames()

# Stock IDs and filtering
stock_ids <- levels(as.factor(data_ml$stock_id))
stock_days <- data_ml %>% group_by(stock_id) %>% summarize(nb = n())
stock_ids_short <- stock_ids[which(stock_days$nb == max(stock_days$nb))]

# Feature selection
print(paste("Number of features:", length(features_short)))
print(features_short)

data_short <- data_ml %>%
  filter(stock_id %in% stock_ids_short) %>%
  select(c("stock_id", "date", all_of(features_short), "R1M_Usd"))

data_short <- data_short %>%
  arrange(stock_id, date)

data_label <- data_short %>%
  select(stock_id, date, R1M_Usd)

# Normalize numeric features only (excluding stock_id and date)
data_short_numeric <- data_short %>% select(all_of(features_short), R1M_Usd)
data_short_normalized <- as.data.frame(lapply(data_short_numeric, min_max_norm))

# Model input dimensions
input_dim <- ncol(data_short_normalized)
print(paste("Input dimension:", input_dim))

# Define the autoencoder
encoding_dim <- 30

input_layer <- layer_input(shape = input_dim)

encoder <- input_layer %>%
  layer_dense(units = 64, activation = "relu") %>%
  layer_dense(units = encoding_dim, activation = "relu")

decoder <- encoder %>%
  layer_dense(units = 64, activation = "relu") %>%
  layer_dense(units = input_dim, activation = "linear")

autoencoder_model <- keras_model(inputs = input_layer, outputs = decoder)

autoencoder_model %>% compile(
  loss = "mean_squared_error",
  optimizer = "adam"
)
```

```r
# Fit model with error handling
tryCatch({
  history <- autoencoder_model %>% fit(
    x = as.matrix(data_short_normalized),
    y = as.matrix(data_short_normalized),
    epochs = 10,
    batch_size = 32,
    verbose = 1
  )

  encoder_model <- keras_model(inputs = input_layer, outputs = encoder)
  reduced_data <- encoder_model %>% predict(as.matrix(data_short_normalized))

  print("Autoencoder successfully trained!")
  print("Sample of reduced data:")
  print(head(reduced_data))

}, error = function(e) {
  print(paste("Error occurred:", e$message))
})

print("Autoencoder model summary:")
summary(autoencoder_model)

print("full data dimensions:")
print(dim(data_short_normalized))
head(data_short_normalized)
print("Sample of full data:")

print("reduced data dimensions:")
print(dim(reduced_data))
print("Sample of reduced data:")
print(head(reduced_data))

# Combine reduced data with stock_id and date from data_label
reduced_data_df <- as.data.frame(reduced_data)
colnames(reduced_data_df) <- paste0("feature_", 1:ncol(reduced_data_df))

final_data <- cbind(data_label, reduced_data_df)

# Print sample of the final combined dataset
print("Sample of final combined dataset:")
print(head(final_data))

# Generate Fibonacci numbers for clustering
max_clusters <- 5  # Define the maximum number of clusters
```

```
results <- list()

for (num_clusters in 1:max_clusters) {
  if (num_clusters > nrow(reduced_data)) break  # Skip if clusters exceed data points

  print(paste("Clustering with", num_clusters, "clusters"))

  # Perform clustering for each date
  final_data <- final_data %>%
    group_by(stock_id) %>%
    arrange(date) %>%
    ungroup()

  # Group by date and perform clustering
  final_data$cluster <- final_data %>%
    group_by(date) %>%
    group_map(~ {
      kmeans_result <- kmeans(.x[-1], centers = num_clusters, nstart = 25)
      tibble(date = .x$date[1], cluster = kmeans_result$cluster)
    }) %>%
    bind_rows()

  final_data$cluster <- as.numeric(unlist(final_data$cluster))

  returns <- final_data %>%                          # Compute returns, in matrix format, in 3 steps:
    filter(stock_id %in% stock_ids_short) %>%    # 1. Filtering the data
    dplyr::select(date, stock_id, R1M_Usd) %>%   # 2. Keep returns along with dates & firm names
    spread(key = stock_id, value = R1M_Usd)      # 3. Put in matrix shape
  sep_oos <- as.Date("2010-01-01")                           # Starting point for backtest
  ticks <- final_data$stock_id %>%                              # List of all asset ids
    as.factor() %>%
    levels()
  N <- length(ticks)                                         # Max number of assets
  t_oos <- returns$date[returns$date > sep_oos] %>%          # Out-of-sample dates
    unique() %>%                                           # Remove duplicates
    as.Date(origin = "1970-01-01")                         # Transform in date format
  Tt <- length(t_oos)                                        # Nb of dates, avoid T = TRUE
  nb_port <- 1                                               # Nb of portfolios/stragegies
  portf_weights_list <- list()
  portf_returns_list <- list()

  m_offset <- 12                                           # Offset in months for buffer period
  train_size <- 5                                          # Size of training set in years
```

```
for (cluster_id in 1:num_clusters){
  cluster_data <- final_data %>% filter(cluster == cluster_id)  # Filter for that cluster_id

  # Recalculate since number of stocks may vary
  ticks <- unique(cluster_data$stock_id)
  N <- length(ticks)

  # Initialize per-cluster storage
  portf_weights <- array(0, dim = c(Tt, nb_port, N))          # Initialize portfolio weights
  portf_returns <- matrix(0, nrow = Tt, ncol = nb_port)       # Initialize portfolio returns

  for(t in 1:(length(t_oos)-1)){                             # Stop before last date: no fwd ret.!
    if(t%%12==0){print(t_oos[t])}                           # Just checking the date status

    train_data <- cluster_data %>% filter(date < t_oos[t] - m_offset * 30,    # Roll window w. buffer
                                       date > t_oos[t] - m_offset * 30 - 365 * train_size)

    test_data <- cluster_data %>% filter(date == t_oos[t])   # Test sample
    realized_returns <- test_data$R1M_Usd

    for(j in 1:nb_port){
      temp_weights <- portf_compo(train_data, test_data)
      ind <- match(temp_weights$names, ticks) %>% na.omit()
      portf_weights[t,j,ind] <- temp_weights$weights
      portf_returns[t,j] <- sum(temp_weights$weights * realized_returns)
    }
  }

  portf_weights_list[[as.character(cluster_id)]] <- portf_weights
  portf_returns_list[[as.character(cluster_id)]] <- portf_returns
}

asset_returns <- data_ml %>%                               # Compute return matrix: start from data
  dplyr::select(date, stock_id, R1M_Usd) %>%              # Keep 3 attributes
  spread(key = stock_id, value = R1M_Usd)                 # Shape in matrix format
asset_returns[is.na(asset_returns)] <- 0                   # Zero returns for missing points
```

```
  metrics_list <- list()

  for (cluster_id in 1:num_clusters) {
    print(paste(" Metrics for Cluster", cluster_id, ":"))

    current_assets <- as.character(unique(final_data$stock_id[final_data$cluster == cluster_id]))

    asset_returns_cluster <- asset_returns %>%
      dplyr::select(date, all_of(current_assets))

    met <- perf_met_multi(
      portf_returns = portf_returns_list[[cluster_id]],
      weights = portf_weights_list[[cluster_id]],
      asset_returns = asset_returns,
      t_oos = t_oos,
      strat_name = c("EW")  # You can customize if you have multiple strategies
    )

    print(met)

    metrics_list[[cluster_id]] <- met

    g1 <- tibble(date = t_oos,
                 ew_portfolio = cumprod(1+portf_returns_list[[cluster_id]][,1])) %>%
      gather(key = strat, value = value, -date) %>%
      ggplot(aes(x = date, y = value, color = strat)) + geom_line() +theme_grey() +
      ggtitle(paste("Cluster ID:", cluster_id, ", Number of Cluster:", num_clusters))

    g2 <- tibble(year = lubridate::year(t_oos),
                 ew_portfolio = portf_returns_list[[cluster_id]][,1]) %>%
      gather(key = strat, value = value, -year) %>%
      group_by(year, strat) %>%
      summarise(avg_return = mean(value)) %>%
      ggplot(aes(x = year, y = avg_return, fill = strat)) +
      geom_col(position = "dodge") + theme_grey() +
      ggtitle(paste("Cluster ID:", cluster_id, ", Number of Cluster:", num_clusters))
    print(plot_grid(g1,g2, nrow = 2))
  }

  results[[num_clusters]] <- metrics_list
}
```

References

Vijayalakshmi Pai, G. A., & Michel, T. (2009b). Evolutionary optimization of constrained k-means clustered assets for diversification in small portfolios. IEEE Transactions on Evolutionary Computation, 13(5), 1030–1053. https://doi.org/10.1109/tevc.2009.2014360

Wirjanto, T. S. (2025). AFM 423/ACTSC 423 Assignment: w2025_423_H2s, R_Code_Chunks_for_Section2.tex, T2.pdf and A2.pdf. University of Waterloo.

Wirjanto, T. S. (2025). AFM 423/ACTSC 423 Slides: Transcripts for RCC2.pdf, R_Code_Chunks_for_Section2.tex, T2.pdf and A2.pdf. University of Waterloo.

Wirjanto, T. S. (2025). AFM 423/ACTSC 423 Assignment: w2025_423_H4s, R_Code_Chunks_for_Section4.tex, T4.pdf and A4.pdf. University of Waterloo.

Wu, D., Wang, X., & Wu, S. (2022). Construction of stock portfolios based on K-means clustering of continuous trend features. Knowledge-Based Systems, 252, 109358. https://doi.org/10.1016/j.knosys.2022.109358

Zhang, W., Djabirov, M., Pan, B., & Dai, Z. (n.d.). A Multi-Factor Adaptive Statistical Arbitrage Model. https://arxiv.org/pdf/1405.2384