GROUP WORK PROJECT # 01 MScFE 632: Machine Learning in Finance

GROUP NUMBER: ANONYMIZED FOR PRIVACY

FULL LEGAL NAME	LOCATION (COUNTRY)	EMAIL ADDRESS	MARK X FOR ANY NON-CONTRIBUTING MEMBER
ANONYMIZED FOR PRIVACY **ETIENNE LANDRY**	ANONYMIZED FOR PRIVACY	ANONYMIZED FOR PRIVACY ANONYMIZED FOR PRIVACY	
WINSTONE ANONYMIZED FOR PRIVACY	ANONYMIZED FOR PRIVACY	ANONYMIZED FOR PRIVACY ANONYMIZED FOR PRIVACY	
Kristof ANONYMIZED FOR PRIVACY	ANONYMIZED FOR PRIVACY	ANONYMIZED FOR PRIVACY ANONYMIZED FOR PRIVACY	

Statement of integrity: By typing the names of all group members in the text boxes below, you confirm that the assignment submitted is original work produced by the group (excluding any non-contributing members identified with an "X" above).

Team member 1	ANONYMIZED FOR PRIVACY **ETIENNE LANDRY**
Team member 2	**WINSTONE** ANONYMIZED FOR PRIVACY
Team member 3	**Kristof** ANONYMIZED FOR PRIVACY

Use the box below to explain any attempts to reach out to a non-contributing member. Type (N/A) if all members contributed.

Note: You may be required to provide proof of your outreach to non-contributing members upon request.

Step 1: working on the basics.

Team member 1: Classification Trees (Basics)

Classification Trees are a powerful, intuitive supervised learning methodology designed to predict categorical outcomes - such as "Buy" or "Sell," "Outperform" or "Underperform." Think of them as a systematic flow chart of yes/no questions that you apply to a new data point to arrive at a final prediction.

The model operates through a process known as recursive partitioning. It starts with the entire dataset and automatically identifies the single most informative question (e.g., "Is the P/E ratio less than 15?") to split the data into two purer subgroups. This process repeats recursively on each new subgroup, building a tree-like structure where each branch represents a decision rule, and each leaf node represents a final prediction.

This approach is fundamentally non-parametric, meaning it makes no prior assumptions about the underlying data distribution, allowing it to capture complex, non-linear relationships and interactions between features that traditional linear models would miss. The primary output is not just a prediction but a clear, visual map of the decision logic, making it one of the most interpretable machine learning techniques available.

In essence, Classification Trees transform complex, multi-dimensional data into a simple, actionable set of rules, providing both high predictive power and unparalleled transparency for strategic decision-making.

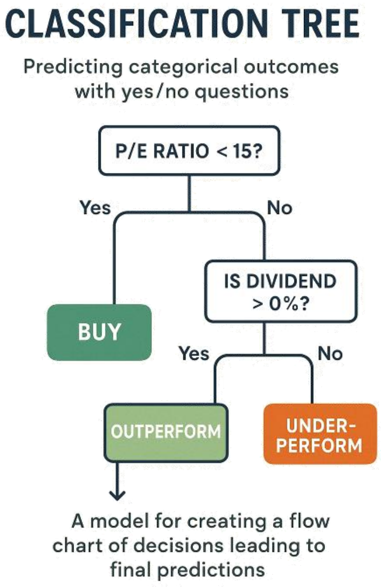

CLASSIFICATION TREE
Predicting categorical outcomes with yes/no questions
P/E RATIO < 15?
Yes
No
BUY
IS DIVIDEND > 0%?
Yes
No
OUTPERFORM
UNDER-PERFORM
A model for creating a flow chart of decisions leading to final predictions

Illustration by author

Keywords: Supervised Learning, Non-Parametric Model, Interpretable AI, Categorical Prediction, Decision Rules, Recursive Partitioning, White-Box Model, Financial Signal Classification

Team Member 2: Lasso Regression

LASSO Regression (Least Absolute Shrinkage and Selection Operator), is a powerful and modern technique within linear regression models. It is a supervised learning method designed for regression tasks where the objective is to predict a continuous outcome variable from a set of input features.

To understand LASSO Regression, it is important to understand the concept of linear regression. Linear regression models the relationship between a dependent variable, such as a stock's return, and one or more independent variables, such as interest rates or inflation. Traditional linear regression methods estimated coefficients to fit a straight line or hyperplane to the data. However, as the number of predictors increases, models can become overly complex and prone to overfitting, which occurs when the model captures noise rather than true underlying patterns.

LASSO addresses this overfitting challenge through regularization, which is a technique used to penalize model complexity. Specifically, it adds an L1 penalty, which is the sum of the absolute values of the coefficients, to the usual least squares cost function. This penalty forces the coefficients of less important features to shrink toward zero and in many cases to exactly zero, effectively removing those variables from the model. This simultaneous action of shrinkage and variable selection makes LASSO especially valuable for high-dimensional datasets where many predictors may be irrelevant or redundant.

The outcome from LASSO is a sparse and interpretable model that retains only the most informative features. In finance and risk modeling fields, LASSO is widely applied to asset pricing, factor selection, and risk analysis where parsimony and robustness are essential.

Mathematically, LASSO modifies the standard linear regression cost function by adding an L1 penalty term. The cost function therefore becomes:

$$J(\theta) = \mathrm{MSE}(\theta) + \alpha \sum_{i=1}^{n} |\theta_i|$$

Where:

- **J(θ)** is the total cost the model tries to minimize.
- **MSE(θ)** is the mean squared error between the predicted and actual values.
- **θ_i** are the model coefficients.
- **α** is a hyperparameter that controls the strength of the penalty.

When α = 0, LASSO behaves like ordinary linear regression. As α increases, the penalty becomes stronger, and more coefficients are shrunk to zero.

Keywords/Tags in Lasso Regression:

A. **Regularization**: A technique used in machine learning to prevent overfitting by penalizing model complexity.
B. **L1 Penalty:** The sum of the absolute values of the regression coefficients, used in LASSO to encourage sparsity.
C. **Feature Selection**: The process of identifying and retaining only the most relevant predictors in a model.
D. **Sparse Model**: A model in which many coefficients are zero, resulting in a simpler and more interpretable structure.
E. **Supervised Learning**: A machine learning paradigm where models are trained on labeled data to predict outcomes.
F. **Linear Model**: A model that assumes a linear relationship between input features and the target variable.
G. **Overfitting:** A modeling error that occurs when a model captures noise in the training data, leading to poor generalization.
H. **High-Dimensional Data**: Datasets with a large number of features, common in financial applications such as factor modeling.
I. **Shrinkage**: The process of reducing the magnitude of regression coefficients to improve model generalization.

Team member c (k-means clustering):

K-Means is an unsupervised machine learning algorithm used for clustering. The goal is to partition a dataset of n observations into k groups (clusters) such that each observation belongs to the cluster with the nearest mean (centroid).

It is:

- Unsupervised → no labels needed, only input features.

- Prototype-based → each cluster is represented by a prototype (the centroid).

- Partitional → directly divides the dataset into non-overlapping clusters, unlike hierarchical clustering.

- Iterative → alternates between assigning points to the nearest centroid and recalculating the centroid until convergence.

Iterative process behind K-Means

1. Initialize: pick k centroids (randomly or with uneducated guess).

2. Assign: each observation is assigned to the nearest centroid using a distance metric (mostly Euclidean, but Manhattan or Chebyshev or other not commonly used metrics are available for specific problems).

3. Update: recalculate each centroid as the mean of the assigned points.

4. Repeat: steps 2–3 until convergence (centroids no longer move significantly, or max iterations reached).

This produces compact, spherical clusters around centroids.

Important distinction from K-Nearest Neighbors is that while KNN use actual data as center of a group, K-Means define the center using a calculated distance

Step 2:

Team member 1: Classification Trees

a). Advantages vs disadvantages

Classification Trees offer several key benefits that make them attractive for financial modeling but, despite their advantages, several limitations must be considered too:

Advantages	Disadvantages
● High Interpretability: The model's decision process is transparent and can be visualized as a simple flowchart, making it easy to explain to stakeholders (e.g., "We buy if the P/E ratio < 20 AND volatility > 0.3"). ● No Assumption of Linearity: They can capture complex, non-linear relationships and interactions between features without requiring transformation. ● Handles Mixed Data Types: Works seamlessly with both numerical and categorical input variables. ● Little Data Preprocessing: Requires no feature scaling or normalization and is	● Prone to Overfitting: Trees can become overly complex, modeling noise in the training data rather than the underlying signal. This necessitates techniques like pruning. ● High Variance: Small changes in the training data can lead to significantly different tree structures, making them less stable than other models. ● Greedy Algorithm: The split at each node is locally optimal but not necessarily globally optimal, which can lead to suboptimal trees. ● Biased Towards Features with More Levels: Features that have more unique

robust to outliers. ● Feature Importance: Naturally ranks features by their predictive power, providing insight into the driving factors of the model.	values (e.g., continuous variables) are favored during the splitting process.

b). Equations

While tree-building is an algorithmic process, it relies on impurity metrics to decide splits. The two most common metrics for classification are:

b-1). Gini Impurity: its mathematical relation can be written as:

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

Where p_i   is the proportion of samples belonging to class i in a node. A Gini impurity of 0 indicates a perfectly pure node.

b-2). Information Gain (based on Entropy):

$$Entropy = -\sum_{i=1}^{C} p_i \log_2(p_i)$$

Information Gain is calculated as the difference between the parent node's entropy and the weighted average entropy of the child nodes after a split. The algorithm chooses the split that maximizes Information Gain (or minimizes Gini Impurity).

C). Features

- **Missing Data:** Can handle missing values through methods like surrogate splits.
- **Outliers:** Generally robust to outliers in feature values.
- **Data Types:** Accommodates numerical and categorical predictors.
- **Interpretability:** Produces a white-box model that is easy to visualize and interpret.

d). Guide

Inputs:

- Feature Matrix (X): A set of predictor variables (e.g., P/E ratio, volatility, moving average).
- Target Vector (y): A categorical outcome variable (e.g., 'Buy', 'Sell').

Outputs:

- A trained tree model capable of predicting the class of new samples.
- The predicted class labels for new data.
- Class probability estimates (if supported).

- A visual representation of the decision rules.

e). Hyperparameters

- Key hyperparameters to tune for optimal performance:
- max_depth: The maximum depth of the tree (prevents overfitting).
- min_samples_split: The minimum number of samples required to split an internal node.
- min_samples_leaf: The minimum number of samples required to be at a leaf node.
- criterion: The function to measure the quality of a split (gini or entropy).
- ccp_alpha: Cost-complexity pruning parameter; increases to prune the tree.

f). Illustration

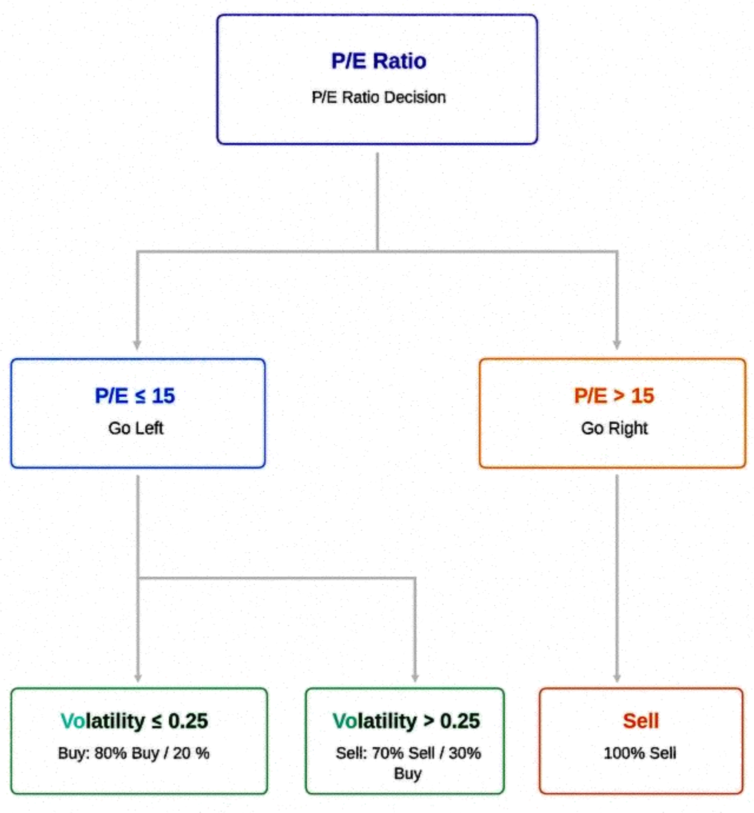


Figure of a classification three algorithm

The above figure displays a simplified classification tree for a trade signal. The root node splits data based on the P/E ratio. Samples with P/E ≤ 15 proceed to the left, where a subsequent split on 30-day volatility further refines the prediction. The leaf nodes show the final classification ('Buy' or 'Sell') and the proportion of samples in each class, demonstrating the model's intuitive decision-making process.

g). Journal Reference

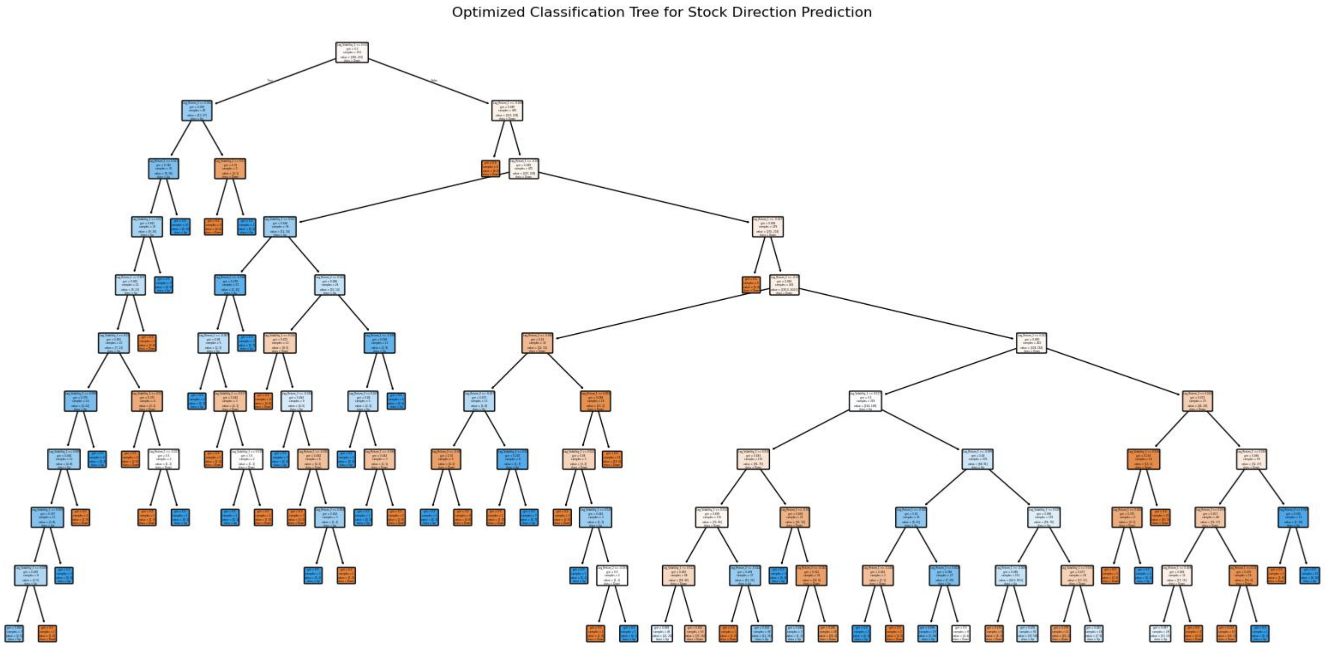
Article: Loh, W. Y., & Shih, Y. S. (1997). Split selection methods for classification trees. Statistica Sinica, 7(4), 815-840.

h). Keywords/tags:

Supervised Learning, Non-Parametric Model, Interpretable AI, Categorical Prediction, Decision Rules, Recursive Partitioning, White-Box Model, Financial Signal Classification, Gini Impurity, Information Gain

g). Computation:

The following computation demonstrates the end-to-end process of building, tuning, and evaluating a Classification Tree model for a binary financial prediction task: forecasting whether a stock's price will go up or down the next day.


Optimized Classification Tree for Stock Direction Prediction

1)   **. Baseline Model Performance (Accuracy: 0.45):**

The initial model, built with default hyperparameters, achieved an accuracy of approximately 45%. This is below the 50% threshold of a random guess. The classification report shows low precision and recall for both classes (0 and 1), indicating that the model is struggling to find meaningful patterns. This is a classic sign of an overfit tree that has learned the noise in the training data rather than a generalizable signal, leading to poor performance on unseen test data.

2)   **. Hyperparameter Tuning:**

We used GridSearchCV to systematically find a better combination of max_depth, min_samples_split, and min_samples_leaf. The process identified {'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 2} as the optimal set. The max_depth being constrained to 10 (as opposed to the unlimited depth of the baseline model) is the key change that helps control overfitting by limiting the complexity of the tree.

3) **. Tuned Model Performance (Accuracy: 0.49):**

The tuned model's accuracy improved to approximately 49.1%. While still close to a random guess, this improvement confirms that applying regularization via hyperparameter tuning is a crucial step. The model is now less complex and more robust. The remaining challenge in achieving high accuracy underscores the inherent difficulty of predicting short-term stock movements, which are often close to random walks influenced by myriad unforeseen factors.

4) **. Model Visualization and Feature Importance:**

The generated tree visualization (which you have) provides the greatest strength of this methodology: transparency. A portfolio manager can literally trace the decision path for any prediction. Furthermore, the feature importance analysis reveals which factors (e.g., Lag_Return_1, Lag_Volatility_1) the model found most predictive. This insight is valuable even if the predictive accuracy is moderate, as it can validate or challenge existing financial hypotheses about what drives price movements.

Recommended Course of Action & Conclusion:
For a real-world application, the following steps are recommended:

- **Feature Engineering:** Incorporate more sophisticated features, such as momentum indicators, relative strength index (RSI), or macro-economic data.
- **Model Ensembling:** Move from a single Classification Tree to an ensemble method like a Random Forest or Gradient Boosting Machine (e.g., XGBoost). These models maintain the tree-based structure but are far more powerful and stable by combining the predictions of many trees, thereby mitigating the high variance of a single tree.
- **Focus on Interpretability:** Use the clear rules generated by the tree to create simple, explainable screening filters for a portfolio, rather than relying on it for high-frequency directional bets.

In conclusion, this exercise demonstrates that while a single Classification Tree may not be a "silver bullet" for highly noisy financial prediction, it serves as an exceptional diagnostic and explanatory tool. Its value lies in its ability to provide a transparent, interpretable starting point for strategy development and feature analysis.

Team Member 2: Lasso Regression

a). Advantages and disadvantages of Lasso Regression

Aspect	**Advantages**	**Disadvantages**
Model Simplicity	Produces sparse models by shrinking some coefficients to	May exclude relevant variables if they are highly correlated with

	exactly zero, enhancing interpretability.	others, leading to underfitting.
Feature Selection	Performs automatic variable selection, which is useful in high-dimensional datasets.	Selection can be unstable when predictors are highly correlated; small changes in data may alter results.
Overfitting Control	Regularization reduces model complexity, helping to prevent overfitting.	The choice of regularization parameter (α) is critical and requires careful tuning.
Computational Efficiency	Efficient to compute using standard optimization techniques and widely available in ML libraries.	May not perform well when the number of predictors is much larger than the number of observations (p > n).
Multicollinearity Handling	Can handle multicollinearity by selecting one variable from a group of correlated predictors.	Does not distinguish between equally informative correlated variables; may arbitrarily select one.
Application in Finance	Useful in asset pricing, factor selection, and risk modeling where parsimony is valued.	Interpretation of coefficients can be challenging when variables are standardized or transformed.

b). Equations and Implementation in Python

LASSO regression minimizes the following cost function:

$$J(\theta) = \mathrm{MSE}(\theta) + \alpha \sum_{i=1}^{n} |\theta_i|$$

Where:

- **MSE(θ)** is the mean squared error between the predicted and the actual values.
- **θi** represents the model coefficients.
- **α≥0** is the regularization parameter that controls the strength of the penalty.

Implementation in Python

We generated a synthetic dataset with 10 features, some of which were made correlated to simulate multicollinearity which is a common issue in financial data.
The target variable was constructed using a sparse set of true coefficients.

LASSO regression was applied across a range of α values to observe how coefficients shrink.
A coefficient path plot was generated to visualize how each feature's weight changes with increasing regularization.
The final model was fitted using **α = 0.1**, and the resulting coefficients were below:

```
LASSO coefficients for alpha=0.1:
     Feature  Coefficient
0  Feature 0     1.142628
1  Feature 1     0.000000
2  Feature 2    -1.822308
3  Feature 3    -0.029643
4  Feature 4     3.305830
5  Feature 5     0.000000
6  Feature 6     0.116296
7  Feature 7    -0.000000
8  Feature 8    -0.000000
9  Feature 9    -0.000000
```

As expected, LASSO shrunk several coefficients to exactly zero, effectively removing those features from the model.

The LASSO coefficient path showed how each feature's coefficient changed as the regularization strength (α) increases:

- Features with weak predictive power are gradually shrunk to zero.
- Strong predictors retain non-zero coefficients even under moderate regularization.

c). Features

LASSO regression is characterized by several notable features that make it particularly useful in financial machine learning:

- **Automatic Feature Selection:** LASSO can shrink some regression coefficients to exactly zero, effectively removing irrelevant variables from the model. This leads to simpler, more interpretable models.
- **Regularization:** By penalizing the absolute values of the coefficients, LASSO helps prevent overfitting, especially in high-dimensional datasets.
- **Sparsity:** The resulting models are sparse, meaning they include only a subset of the original predictors, which is advantageous for interpretability and computational efficiency.
- **Handling Multicollinearity:** LASSO can manage multicollinearity by selecting one variable from a group of highly correlated predictors, though it may not always select the most relevant one.
- **Scalability:** LASSO is computationally efficient and can be applied to large datasets using standard machine learning libraries.

d). Guide: Inputs and Outputs

Inputs

- **Feature Matrix (X):** A two-dimensional array or DataFrame containing the predictor variables.
- **Target Vector (y):** A one-dimensional array or Series containing the response variable.
- **Regularization Parameter (alpha):** A non-negative scalar that controls the strength of the penalty.
- **Optional Parameters:** Maximum number of iterations, tolerance for optimization, etc.

Outputs

- **Estimated Coefficients:** The weights assigned to each predictor, with some potentially set to zero.
- **Selected Features:** The subset of predictors with non-zero coefficients**.**
- **Predicted Values:** The model's predictions for new or existing data.
- **Model Diagnostics:** Metrics such as mean squared error, R-squared, and residuals.

e). Hyperparameters

- **Alpha (α):** The primary hyperparameter controlling the strength of the L1 penalty. Higher values of alpha increase regularization, leading to more coefficients being set to zero.
- **Max_iter:** The maximum number of iterations for the optimization algorithm to converge**.**
- **Tol:** The tolerance for the optimization; determines when to stop iterating.
- **Fit_intercept:** Whether to calculate the intercept for the model.
- **Normalize:** Whether to normalize the feature matrix before regression (deprecated in some libraries in favor of explicit preprocessing).

Hyperparameter tuning, particularly for alpha, is typically performed using cross-validation to identify the value that yields the best predictive performance on unseen data.

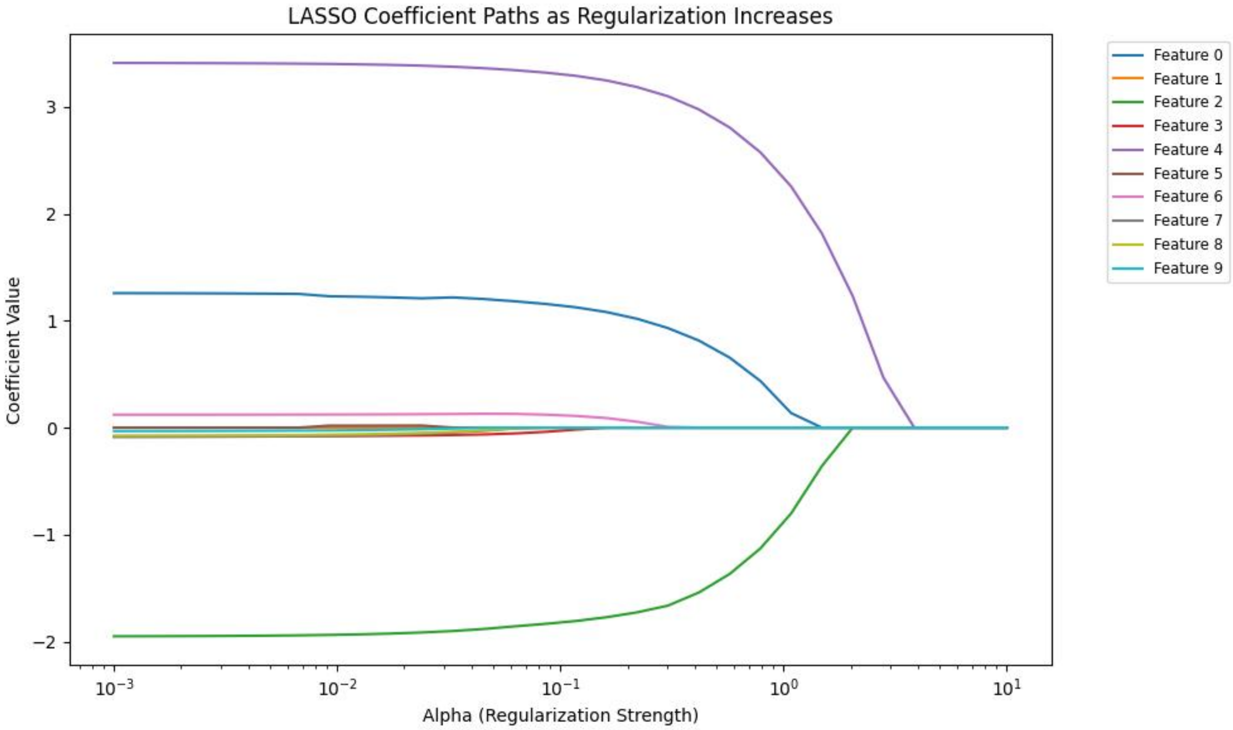
f). Illustration

In our LASSO regression implementation, we demonstrated the effect of the regularization hyperparameter α on model coefficients using a coefficient path plot. The plot displayed how the estimated coefficients for each feature evolve as the value of α increases. As α becomes larger, the penalty for nonzero coefficients increases, causing many coefficients to shrink toward zero. This process results in a sparse model, where only the most relevant features retain nonzero weights.

To systematically address hyperparameter tuning, we evaluated LASSO regression across a range of α values, from very small (minimal regularization) to larger values (strong regularization). For each value of α, we fitted the LASSO model and recorded the resulting coefficients. This approach allowed us to observe which features remained significant as regularization increased and to select an appropriate α that balances model simplicity and predictive accuracy.

In practice, the optimal value of α is typically determined using cross-validation, where the dataset is split into training and validation sets multiple times. The value of α that yields the best predictive performance on unseen data is selected for the final model. This process ensures that the model generalizes well and avoids overfitting.

The following figure (generated in the notebook) illustrates the coefficient shrinkage paths for each feature as a function of α:



LASSO coefficient paths as a function of the regularization parameter α. As α increases, more coefficients are shrunk to zero, resulting in a sparse model.

g.) Journal Reference

Nie, Xiaoyan, and Guohua Deng. "Enterprise Financial Early Warning Based on Lasso Regression Screening Variables." ***Journal of Financial Risk Management*****, vol. 9, no. 4, 2020, pp. 454–461.**

https://doi.org/10.4236/jfrm.2020.94024

Summary of the paper

The paper constructs a financial early warning model for listed companies using LASSO regression for variable selection. By applying LASSO to financial statement data from 2,819 listed enterprises, the authors demonstrate that LASSO effectively reduces multicollinearity and dimensionality, improving the predictive performance of financial risk models. The study highlights the advantages of LASSO in screening relevant variables and enhancing the classification effect of machine learning methods in financial risk management.

h.) Keywords and Tags

As described in step one, the following are the keywords

- Regularization
- L1 Penalty
- Feature Selection
- Sparse Model
- Supervised Learning
- Linear Model
- Overfitting Prevention
- High-Dimensional Data
- Shrinkage
- Variable Selection

Team member c (k-means clustering):

a, Advantages and Disadvantages

Advantages

- Fast and scalable on large sample
- Simple to implement and parallelize.
- Interpretable centroids summarize cluster typical attributes
- Useful for peer grouping, feature bucketing before modeling, and regime detection.
- Works well with PCA/standardization pipelines

Disadvantages

- Assumes roughly spherical, equal-variance clusters under Euclidean distance.
- Sensitive to feature scaling, outliers, and initialization (local minima).
- Choosing k is non-trivial; elbow/silhouette/gap can disagree, clusters can be unstable across samples.
- Empty clusters can occur, categorical or heavy-tailed data may need alternative distance metrics or methods.

b, Equations and Implementation

Objective function to minimize:

$$\min_{\{C_j\}_{j=1}^{k}} J = \sum_{j=1}^{k} \sum_{x_i \in C_j} \left\| x_i - \mu_j \right\|^2$$

Where C_j is a cluster containing every datapoint assigned to cluster j.

x_i is the i-th datapoint

μ_j is the center of cluster j

$\left\| x_i - \mu_j \right\|$ is the Euclidean distance between data i and center of the cluster j

$$\mu_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i$$

$|C_j|$ is the number of datapoints in cluster j

Then we iterate until convergence:

1. Assign x_i to argmin(j) $\left\| x_i - \mu_j \right\|^2$
2. Update μ_j

Model-selection diagnostics:

- Silhouette for point i:

: $s(i) = \dfrac{b(i) - a(i)}{\max\{a(i), b(i)\}}$, where $a(i)$ is mean intra-cluster distance,

$b(i)$ the best (lowest) mean inter-cluster distance.

- Elbow: inspect inertia vs. kkk for diminishing returns (“knee”).
- Stability: resample data, compare clusterings (e.g., ARI/NMI).

c, Features

- Works well on continuous, standardized features (e.g. returns).
- Computation-friendly for recurring re-clustering (daily/weekly refresh).

d, Guide and Hyperparameters

Inputs

- Matrix X: engineered features per asset (e.g., annualized return, volatility, beta, skew, kurtosis).
- Hyperparameters:
    - k: (number of clusters) — chosen via silhouette, elbow, gap statistic, or stability.
    - init: Initialization, selects initial cluster centroids
    - n_init: Number of times the k-means algorithm is run with different centroid seeds
    - Max_iter: Maximum number of iterations
    - Tol: Relative tolerance to declare convergence
    - random_state.

Outputs

Labels: cluster id per asset

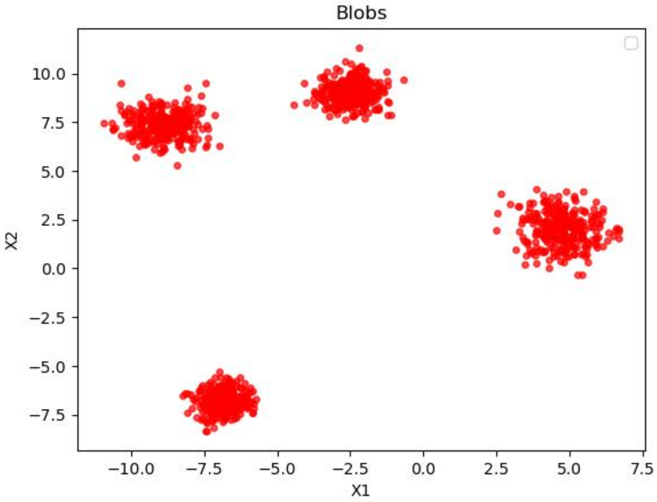
cluster_centers_: coordinate of centroids

Inertia_: sum of squared distance of samples to their closest cluster center

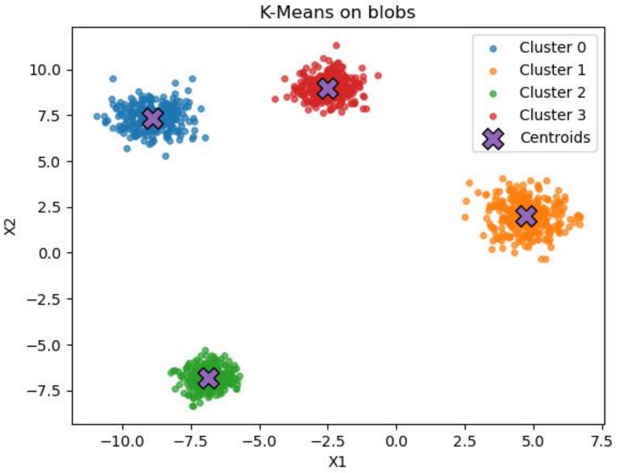
e, Illustration

A working example of K-Means algorithm on simulated data:

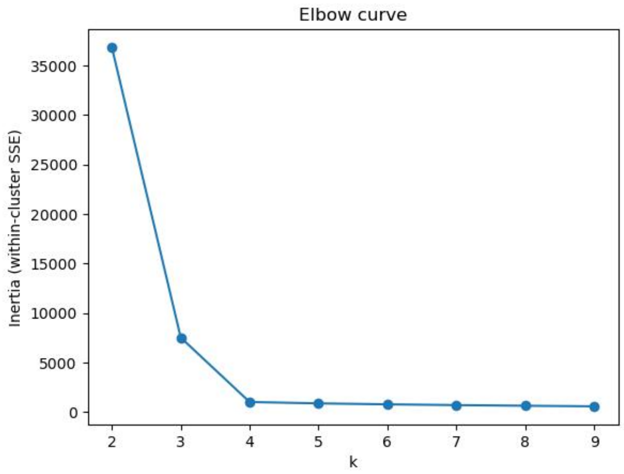
Simulated, non-clustered data:

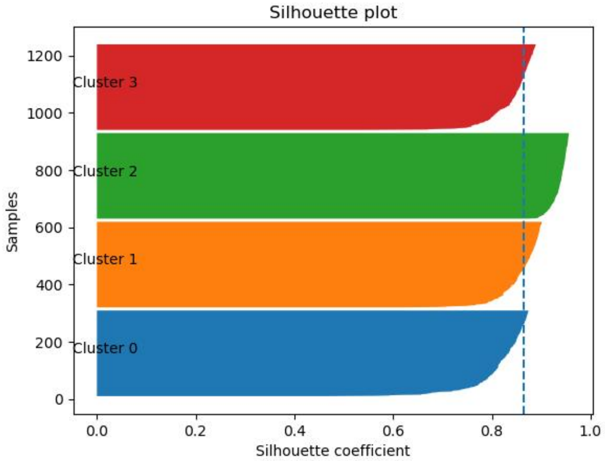


Clustered data using K-Means (details in the provided code):



Evaluation (elbow curve and Silhouette plot):





Based on the evaluation metrics:

- Elbow plot shows that 4 is the number cluster to find
- Silhouette plot shows that the clusters we formed are well established, there is no sign of an element of a cluster would possibly belong to another (negative Silhouette coefficient indicates that)

f, Journal

Aslam, B. (2025). Identifying optimistic stocks with K-means clustering algorithm. *International Review of Economics & Finance, 104,* 104579. https://doi.org/10.1016/j.iref.2025.104579

g, Keywords:

unsupervised learning

clustering

centroid

distance-based

Euclidean distance

Step 3 (As a group): Technical Section

1). Introduction: Why Tune Hyperparameters?

In machine learning, a model has two types of parameters:

- **Model Parameters:** These are internal to the model and learned directly from the training data (e.g., the coefficients in a regression, the split points in a tree).
- **Hyperparameters:** These are external configuration settings that are not learned from the data. They are set prior to the training process and control the very nature of the learning algorithm itself.

Hyperparameter tuning is the process of systematically searching for the optimal combination of these hyperparameters that results in the best-performing model. The primary goal is to find a model that generalizes well to unseen data, effectively balancing the bias-variance trade-off to avoid underfitting (high bias) or overfitting (high variance).

2). Common Tuning Methodologies

Several strategies exist for navigating the hyperparameter space:

- **Grid Search:** This is an exhaustive search method. The data scientist defines a grid of possible values for each hyperparameter, and the algorithm trains and evaluates a model for every single combination in this grid. While computationally expensive, it is thorough and guaranteed to find the best combination within the predefined grid.
- **Random Search:** Instead of searching every combination, Random Search selects random combinations of hyperparameters from a specified distribution for a fixed number of iterations. It is often much more efficient than Grid Search, as it can find a good combination without exploring the entire grid, especially when some hyperparameters have little impact on the result.
- **Bayesian Optimization:** This is a more advanced, probabilistic approach that builds a surrogate model of the objective function (e.g., validation score) to predict which hyperparameter

combinations are most promising. It uses these predictions to intelligently select the next set of hyperparameters to evaluate, typically converging to an optimal set much faster than random or grid search.

The performance of each hyperparameter set is evaluated using a resampling method like k-fold Cross-Validation to ensure the estimate of performance is robust and not dependent on a single train-test split.

3). Examples from Our Methodologies

The importance of tuning is universal across models, as illustrated by the hyperparameters we tuned in our individual analyses:

a). Classification Trees: A tree without constraints will grow until it memorizes the training data, leading to overfitting. Key hyperparameters act as regularization tools to prune the tree and enhance generalization.

Example: max_depth - This hyperparameter restricts the maximum number of levels in the tree. A value that is too low (max_depth=3) may cause underfitting (high bias), while a value that is too high (max_depth=50) will certainly cause overfitting (high variance). Tuning this parameter, as demonstrated in our analysis, is essential for finding the right level of model complexity.

b). LASSO Regression:

The primary hyperparameter in LASSO regression is the regularization parameter, commonly denoted as α (alpha). This parameter controls the strength of the L1 penalty applied to the regression coefficients. A small value of α results in minimal regularization, allowing the model to retain more features, while a large α enforces stronger shrinkage, potentially setting many coefficients to zero. The optimal value of α balances model simplicity and predictive accuracy.

To tune α effectively, we employed a grid search over a logarithmically spaced range of values, from 10^{-3} to 10^{1}. For each candidate value, a LASSO model was trained and evaluated using a validation set. The performance metric used was the mean squared error (MSE) on the validation data. The α value that minimized the validation error was selected for the final model.

This approach was illustrated in our Jupyter notebook, where we plotted the coefficient paths for each feature across varying α values. The visualization clearly demonstrated how increasing regularization leads to progressive shrinkage of coefficients, with many being reduced to zero under moderate to strong regularization.

In practice, hyperparameter tuning for LASSO can also be performed using cross-validation, such as k-fold cross-validation, to ensure robustness across different data splits. Libraries such as scikit-learn provide built-in tools like LassoCV for automated tuning.

In summary, hyperparameter tuning in LASSO regression is essential for achieving a model that is both parsimonious and predictive. The choice of α determines the trade-off between bias and variance, and its careful calibration ensures that the model generalizes well to unseen data.

c). k-Means Clustering:

Hyperparameter tuning for K-means mostly boils down to just using external metrics like silhouette score and elbow plot to find the number of clusters (k).

We simply pick k where the elbow plot bend and that maximizes silhouette score

Initialization can be improved by choosing more sophisticated initialization algorithm like kmeans++ that select initial centroids based on an empirical probability distribution of the points

Number of model runs improve the stability of the results (n_init)

If tolerance (tol) is set to a low value we can avoid premature stopping.

All these hyperparameters are tuned,however the algorithm can still be stuck at local minima.

Step 4 (As a group): Marketing Alpha

In today's complex and data-saturated financial markets, traditional analytical tools are no longer sufficient to secure a sustainable edge. At Team Alpha, we specialize in deploying cutting-edge machine learning (ML) methodologies not as black-box solutions, but as transparent, robust engines for discovery and decision-making. Our integrated approach, leveraging the unique strengths of LASSO Regression, k-Means Clustering, and Classification Trees, provides a multi-faceted lens through which to identify alpha, manage risk, and build stronger, more adaptive portfolios.

1. Cutting Through the Noise: The Power of Strategic Simplification

In an era of information overload, the ability to distinguish signal from noise is paramount. Our use of LASSO Regression provides this critical first filter. While many models drown in hundreds of potential factors, LASSO intelligently distills them down to the few, most impactful drivers of returns. This creates sparse, stable models that are inherently less prone to overfitting historical quirks and more likely to perform consistently in live markets. For your portfolio, this means enhanced robustness and a clearer understanding of the fundamental factors that truly matter, leading to more reliable and interpretable forecasting models.

2. Discovering Hidden Patterns: Adapting to Market Regimes

Financial markets are not static; they evolve through distinct regimes - periods of high volatility, low growth, or speculative bubbles. Traditional models that assume a constant relationship often break down during these shifts. This is where our expertise in k-Means Clustering delivers exceptional value. By analyzing multidimensional market data, we can automatically identify and characterize these hidden regimes. This allows for the development of dynamic, regime-aware strategies. Imagine a portfolio that intuitively adjusts its asset allocation or risk exposure based on the identified market environment—aggressive during trends, defensive during turbulence. k-Means clustering provides the objective, data-driven framework to make this a reality, enabling proactive rather than reactive management.

3. The Logic Behind the Decision: Building Trust with Transparency

Many powerful ML techniques are inscrutable, creating a "trust gap" for portfolio managers and stakeholders. Team Alpha bridges this gap decisively with Classification Trees. This methodology offers a unique combination of predictive power and crystal-clear interpretability. Each decision can be traced through a simple flowchart of logical rules (e.g., "IF P/E is low AND momentum is positive THEN BUY"). This transparency is invaluable. It builds confidence in the strategy, facilitates regulatory compliance, and allows you, the manager, to understand the "why" behind every signal. Furthermore, these models excel at capturing complex, non-linear relationships that linear models miss, uncovering subtle opportunities hidden within the data.

The Team Alpha Synthesis: An Integrated Advantage

The true alpha generated by Team Alpha lies not in any single model, but in our strategic synthesis of these methodologies into a cohesive analytical workflow:

- Filter & Focus (LASSO): We start by using LASSO to screen hundreds of potential factors, identifying the most robust predictors for your specific mandate.
- Contextualize & Segment (k-Means): We then use these key factors to cluster market environments, ensuring our strategies are tailored to the current regime.
- Execute & Explain (Classification Trees): Finally, we build transparent, rule-based models that generate clear, actionable signals within each defined regime, providing both a strategy and its straightforward rationale.

This process transforms raw data into actionable, intelligent strategy. It moves beyond mere prediction to provide a deep, explanatory understanding of market dynamics.

Conclusion: From Data to Decisive Advantage

At Team Alpha, we believe that the most sophisticated technology should yield the most intuitive insights. Our curated suite of machine learning techniques is specifically chosen to provide not just statistical power, but also clarity, stability, and adaptability. We empower you to navigate modern markets with confidence, backed by models that are as explainable as they are effective.

Step 5 (As a group): Learn More

The following resources provide deeper insight into the machine learning methodologies employed by Team Alpha and their powerful applications within the financial industry. These references emphasize the strategic advantages and real-world impact of these techniques.

Journals Articles:

[1] "Artificial Intelligence and Machine Learning in Financial Services." Bank for International Settlements, 7 Nov. 2022, www.bis.org/fsi/publ/insights47.htm.

[2] Hastie, Trevor, et al. "The Elements of Statistical Learning: Data Mining, Inference, and Prediction." 2nd ed., Springer, 2009.

[3] "How Machine Learning Is Shaping the Future of Finance." McKinsey & Company, 15 Feb. 2023, www.mckinsey.com/industries/financial-services/our-insights/how-machine-learning-is-shaping-the-future-of-finance.

[4] Kumar, Mohit, and M. Thenmozhi. "Forecasting Stock Index Movement: A Comparison of Support Vector Machines and Random Forest." Indian Institute of Capital Markets 9th Capital Markets Conference Paper, 2006. SSRN, doi:10.2139/ssrn.876544.

[5] Loh, Wei-Yin, and Yu-Shan Shih. "Split Selection Methods for Classification Trees." Statistica Sinica, vol. 7, no. 4, 1997, pp. 815–840.

[6] López de Prado, Marcos. "The 7 Reasons Most Machine Learning Funds Fail." The Journal of Portfolio Management, vol. 44, no. 6, 2018. SSRN, papers.ssrn.com/sol3/papers.cfm?abstract_id=3086732.

[7] "What is Machine Learning?" MIT Technology Review, MIT, 2021, www.technologyreview.com/2021/11/18/1040759/what-is-machine-learning-we-drew-you-another-flowchart/.

[8] Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd ed., Springer, 2009.

[9]"Machine Learning in Asset Management—Part 1: Portfolio Construction—Overview and Applications." *CFA Institute Research Foundation*, 2020, **https://www.cfainstitute.org/en/research/foundation/2020/machine-learning-in-asset-management-part-1**

[10] Chan-Lau, Jorge A. "Lasso Regressions and Forecasting Models in Applied Stress Testing." *IMF Working Paper* WP/17/108, International Monetary Fund, 2017.
https://www.imf.org/~/media/Files/Publications/WP/2017/wp17108.ashx

[11]Gu, Shihao, Bryan Kelly, and Dacheng Xiu. "Empirical Asset Pricing via Machine Learning." *The Review of Financial Studies*, vol. 33, no. 5, 2020, pp. 2223–2273. Oxford University Press. https://academic.oup.com/rfs/article/33/5/2223/5758276