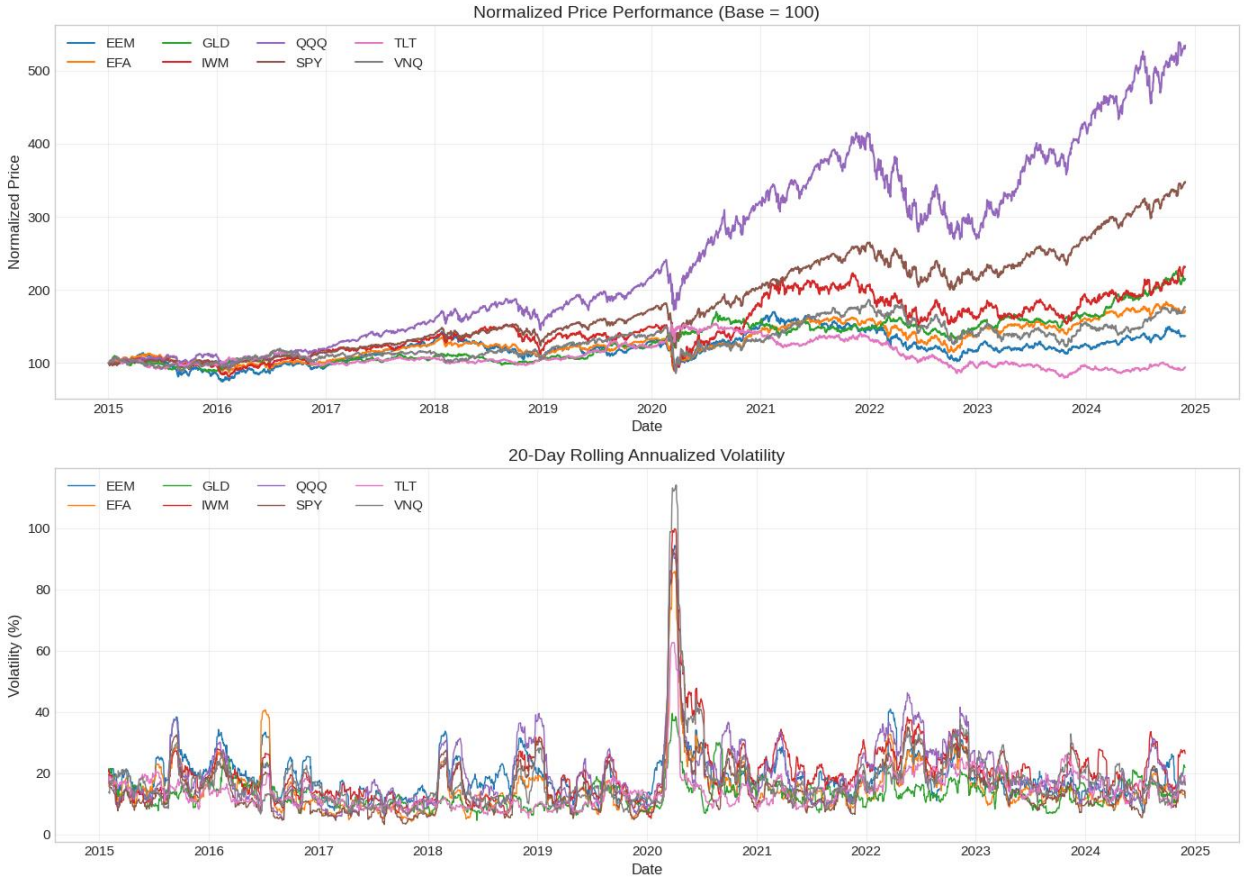
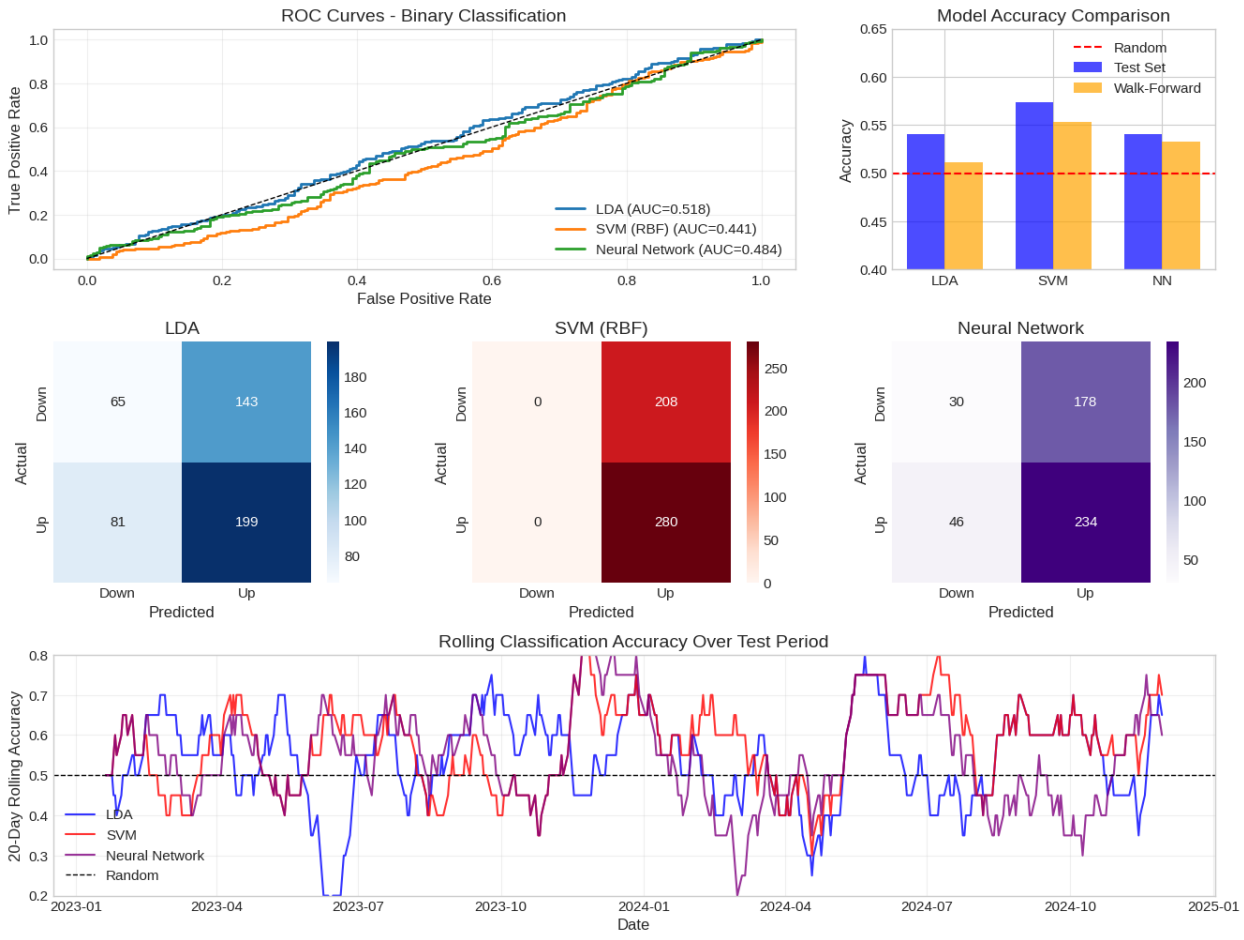
Project Overview

The analysis deals with two classification problems that are at the heart of quantitative finance: how to predict the direction of the market tomorrow (binary classification), and how to determine market regimes by looking at volatility-return properties (multiclass classification). We created 88 technical features based on each-day price data of SPY, QQQ, IWM, EFA, EEM, TLT, GLD, and VNQ, which represent the U.S. equities, international markets, fixed interest, commodity and property markets.



Principal Findings

- We find some significant implications of our empirical findings on practitioners:
- Predicting direction on a daily basis is incredibly difficult and all models point to ROC-AUC values of close to 0.5-weak-form market efficiency.
- SVM showed class collapse in the absence of explicit class balancing, showing SVM to have important preprocessing conditions on imbalanced financial data.
- The classification of market regime demonstrates better prospects where the model has way outdone the random baseline by 13-15 percentage points.
- LDA offers ideal interpretability to regulated applications that need clear rules of decision.



Report Structure

Each of the methodologies is fully covered in the sections that follow:

- **Step 1** outlines the theoretical basis, definitions as well as keywords of LDA, SVM and Neural Networks that is collaboratively developed by all the team members.
- The detailed reports on the benefits and drawbacks, equations, features, hyperparameters, and empirical examples of every model have been put in **Step 2.**
- **Step 3** deals with hyperparameter tuning policies using model-specific examples of our analysis.
- **Step 4** is the synthesis of model capabilities on the "Marketing Alpha" part.
- **Step 5** gathers scholarly materials and sources in "Learn More".
- A detailed model comparison matrix is given in **step 6.**

The full and executable code of all computational implementations that give the results discussed herein, are contained in the associated Jupyter notebook.

Step 1: Basics and Keywords

Category 5: Linear Discriminant Analysis (LDA)

Basics

Linear Discriminant Analysis is a supervised classification in this case, the separation between two or more classes is used to distinguish between positive and negative responses to identifying the optimal linear combination of predictor variables (Fisher 179). The algorithm performs a dimensionality reduction and distribution of data into classes at the same time projecting high-dimensional data onto a low-dimensional space with the highest between-class separability. LDA works with the assumption that observations of various classes are produced out of multivariate Gaussian distributions with the same covariance matrices with the only difference that the mean vectors are different (Hastie et al. 106). The method obtains discriminant functions defining decision boundaries between classes using a calculation of posterior probability based on the Bayes' theorem.

LDA has been shown to be of great use in financial applications in the estimation of credit risk, prediction of bankruptcy, and categorization of market regimes. The classical example of the use of LDA in practice is the Z-score model of Altman, which is used to predict corporate distress with a classification rate of over 90 per cent to differentiate between bankrupt and non-bankrupt companies (Altman 594). The ease of interpretation, that is discriminant coefficients directly show feature importance, makes the method useful especially in regulated financial settings where model explainability is required.

Keywords

1 **Discriminant Function:** This is a linear combination of prediction variables that are designed to maximize the separation of predetermined classes. The role transforms observations into a discriminant axis and allocates the membership of the classes according to the scores produced against a set value (McLachlan 9).

2 **Between-Class Scatter Matrix (S_B):** This is a matrix that measures the distribution of class means about the total data mean. Given mathematically as $\boldsymbol{S_B = \sum k = 1nk(\mu k - \mu)(\mu k - \mu)T}$, where $\boldsymbol{\mu_k}$ represents the mean of class $\boldsymbol{k}$, $\boldsymbol{\mu}$ denotes the grand mean, and $\boldsymbol{n_k}$ is the number of observations in the $\boldsymbol{k}$ th class. Larger between-class scatter is an indicator that shows the wider difference between group centres (Hastie et al. 109).

3 **Within-Class Scatter Matrix (S_W):** A matrix that determines the variance of the observations and the means of classes, calculated by $\boldsymbol{S_W = \sum k = 1K \sum x \in Ck(x - \mu k)(x - \mu k)T}$. Smaller within-class scatter represents a stronger grouping of observations in each category and allows making classification more credible (Hastie et al. 109).

4 **Fisher Criterion:** The optimization problem is to maximize the ratio between between-class variance and within-class variance. This criterion ensures that the projection direction $\boldsymbol{w}$ will give maximum class separability (Fisher 181).

5 **Posterior Probability:** The likelihood of the observation to fall in a particular class, given the observed values of the features of the observation. LDA calculates posteriors based on the Bayes theorem**:** $\boldsymbol{P(Ck \mid x) = P(x)P(x \mid Ck)P(Ck)}$, enabling probabilistic classification and not deterministic assignment (Murphy 101).

6 **Prior Probability:** The unconditional probability of belonging to or not belonging to a class before any predictor values have been seen. Priors affect boundaries to classification, especially when the

proportion of classes in training is not equal to the proportion of classes in the population (James et al. 142).

7 **Homoscedasticity:** The assumption all classes have the same covariance matrix **Σ**. Violation of this assumption may impair LDA performance, which is why Quadratic Discriminant Analysis should be used when the issue of heteroscedasticity is suspected (Hastie et al. 110).

8 **Mahalanobis Distance:** A distance metric accounting of correlations between variables, which is defined **as** $\boldsymbol{D_M(x) = \sqrt{(x-\mu)^T\Sigma^{-1}(x-\mu)}}$. LDA implicitly employs this distance measure when assigning observations to the nearest class centroid (McLachlan 13).

9 **The Estimation of Shrinkage:** An estimation method which draws the sample covariance matrix towards a structured model (e.g. diagonal matrix) to enhance stability in estimation when the number of features becomes close or equal to the number of observations (Ledoit and Wolf 366).

10 **Dimensionality Reduction:** LDA projects data onto at most ($\boldsymbol{K-1}$) discriminant axes of $\boldsymbol{K}$ classes, which allow visualization, high-dimensional classification problems without distorting the information that is relevant to classes (Hastie et al. 113).

Category 6: Support Vector Machines (SVM)

Basics

Support Vector machines are a classification and regression supervised learning algorithm that builds an ideal separating hyperplane maximizing the distance between classes (Cortes and Vapnik 273). The margin is the distance perpendicular to the hyperplane to the closest data points (so-called support vectors), which are the only elements that define the decision boundary of the classifier. The statistical learning theory by Vapnik is based on the fact that the complexity of models is regulated by the maximization of this geometric margin and the upper bound of an error of generalization is minimized (Vapnik 138). In the case of linearly inseparable data, SVM uses the kernel trick - inner product in implicitly mapped, high-dimensional feature space, without the explicit transformation, in order to enable the creation of non-linear decision boundaries (Cristianini and Shawe-Taylor 30). Soft margin formulation adds slack variables within which restricted misclassification is allowed, and this is regulated by the regularization parameter. C which is a tradeoff between margin width and training error (Cortes and Vapnik 285). SVM is a convex quadratic programming problem with a global optimum and is not a local minimum problem as neural networks are. SVM has also found financial applications as a financial direction predictor, credit score, fraud, and volatility forecast. Huang et al. show that SVM performs better than the conventional statistical analysis techniques to predict directions of stocks in the market, and the improvement in performance can be attributed to the ability of the algorithm to learn non-linear market trends (2519).

Keywords

1 **Hyperplane:** This is a plane splitting two or more classes in feature space. In $\boldsymbol{n}$-dimensional space, the dimensionality of the hyperplane is $\boldsymbol{n-1}$ and is defined by $\boldsymbol{w^{Tx}+b=0}$, where $\boldsymbol{w}$ is the normal vector and $\boldsymbol{b}$ is the bias term (Burges 127).

2 **Margin:** The distance between the hyper plane and the nearest data points. The geometric margin **is** $\boldsymbol{\gamma = \frac{2}{||w||}}$, and maximizes this to enhance generalization with SVM (Cristianini and Shawe-Taylor 97).

3 **Support Vectors:** Observations on the boundary of the margin, that is, training observations that are on the margin coincidentally $y_i\left(\boldsymbol{w^{Tx_i}}+b\right)=1$

4 These are the only points that can decide the best hyperplane and eliminating the non-support vectors does not change the solution (Burges 130).

5 **Kernel Function:** A function $\boldsymbol{K(x_i, x_j) = \phi(x_i)^T \phi(x_j)}$ computing transformed feature space inner products without invoking a clear mapping. The most common kernels are linear polynomial $\left(\left(\boldsymbol{\gamma x_i^T x_j + r}\right)^{\boldsymbol{d}}\right)$, radial basis function $\left(\exp\left(-\gamma \left|\left|\boldsymbol{x_i - x_j}\right|\right|^2\right)\right)$, and sigmoid functions (Cristianini and Shawe-Taylor 33).

6 **Soft Margin:** An extension which allows bounded misclassification by slack variables $\xi_i \geq 0$, and the constraint $\boldsymbol{y_i\left(w^{Tx_i} + b\right) \geq 1 - \xi_i}$. This formulation handles noisy or overlapping class distributions (Cortes and Vapnik 285).

7 **Regularization Parameter ($\boldsymbol{C}$):** A hyperparameter controlling the trade-off between margin maximization and misclassification penalty. High C values enforce strict classification; low values permit wider margins with more training errors (Hastie et al. 420).

8 **Kernel Trick:** A computational method that allows the computation of the optimization problem to occur effectively in potentially infinite-dimensional feature spaces through solely a set of kernel evaluations between training examples (Cristianini and Shawe-Taylor 30).

9 **Gamma ($\boldsymbol{\gamma}$):** A parameter of RBF and polynomial kernel for controlling the radius of influence of each training example. High Gamma they form more localized decision boundaries, low Gamma they form smoother decision boundaries that are more global (Hsu et al. 3).

10 **Dual Problem:** The Lagrangian reformulation of the formulation in the form of dual variables α_i. The dual depends on $\boldsymbol{x_i^{Tx_j}}$ enabling kernel substitution (Burges 135).

11 **Class Imbalance:** A condition where the frequency of the classes varies significantly and therefore the classifier would tend to group similarities with the majority class. These can be resampling, cost-sensitive learning, and the class_weight parameter (He and Garcia 1264).

Category 7: Neural Networks (NN)

Basics

Neural Networks are calculated systems that imitate the biological neural systems, which are made up of linked processing units (neurons) that are arranged in layered structures (McCulloch and Pitts 115). The basic architecture consists of an input layer that takes the values of features, one or more hidden layers that do the non-linear transformation and an output layer that generates predictions. Every connection has a learner weight modified through training to reduce a loss function of a prediction error.

The chain rule of calculus was formalized by Rumelhart et al. as the backpropagation algorithm, which allows gradients of the loss function with respect to network weights, which can be used to optimize parameters using gradient descent (536). Contemporary neural networks use non-linear activation functions such as sigmoid or hyperbolic tangent and rectified linear units (ReLU) which allow the approximation of non-linear functions of arbitrary complexity. The Universal Approximation Theorem is the statement that feedforward networks that have enough hidden units can approximate any continuous function to any precision (Hornik et al. 361).

Neural networks have found application in finance in measure of option pricing, portfolio optimization, credit scoring and algorithmic trading. Heaton, et al. introduce Deep Portfolios, which prove that autoencoders are able to find non-linear latent variables that outperform the traditional linear factor models (5). Nevertheless, the black box quality of neural networks, i.e. the lack of insight into the reasons behind decisions, offers problems when it comes to controlled financial uses of neural networks when it is necessary to discover the motivation behind a model (Rudin 207).

Keywords

1 **Neuron (Node):** This is the basic unit of computation which receives weighted inputs, takes a bias term and performs an activation function: $a = g\left(\sum_j w_j x_j + b\right)$. Outputs are passed to the next layers by neurons (Goodfellow et al. 164).

2 **Activation Function:** This is a non-linear transformation that is used on the outputs of neurons. Other functions typically used are sigmoid $\left(\sigma(z) = 1/(1 + e^{-z})\right)$, tanh, ReLU $\left(\boldsymbol{max}(\mathbf{0}, \boldsymbol{z})\right)$ and softmax for multi-class output normalization (Goodfellow et al. 168).

3 **Weight:** Another possible parameter $\boldsymbol{w_{ij}}$ which is learnable determining correlation between neuron $\boldsymbol{i}$ in layer $\boldsymbol{l}$ and neuron $\boldsymbol{j}$ in layer $l + 1$. Training manages weights to reduce the loss function (Goodfellow et al. 165).

4 **Bias:** Another parameter b that can be learned enabling the activation function to shuffle horizontally, offer flexible data fitting (Goodfellow et al. 165).

5 **Layer:** This is an intermediate layer between input and output layers where a transformation of features is done. Deep networks have a series of concealed layers that permit hierarchical representation acquisition (Bengio et al. 1800).

6 **Backpropagation:** The gradient calculation algorithm $\frac{\partial L}{\partial w}$ propagation. Chains Backward propagation of error signals through output layers to input layers. Gradients are used in updating weights when optimized (Rumelhart et al. 536).

7 **Gradient Descent:** This is a type of iterative optimization algorithm in which weights are modified by using the steepest reduction of the loss. Examples of variants are stochastic gradient descent (SGD), momentum, and adaptive schemes such as Adam (Goodfellow et al. 286).

8 **Learning Rate (η):** This is a hyperparameter that regulates the step size of update. Too large rates will lead to divergence; small rates will lead to slow convergence (Goodfellow et al. 290).

9 **Epoch:** A single cycle of the entire training set. Convergence normally takes several epochs to train (Goodfellow et al. 271).

10 **Overfitting:** A phenomenon in which the learner is good at remembering the training data but has a generalization failure with unseen examples. Dropout, L2 weight penalty, and early stopping are regularization techniques that help in reducing overfitting (Goodfellow et al. 224).

11 **Dropout:** This algorithm is sensitive to the size of its model, and the neuron outputs must be initialized randomly by sampling with probability p, preventing co-adaptation and enhancing generalization (Srivastava et al. 1930).

12 **Batch size:** The size is expressed as the number of training examples that have been run through before the weights are revised. Mini-batch gradient descent provides a trade-off between the efficiency of computation and the variance of gradient estimation (Goodfellow et al. 275).

Step 2: Individual Reports

Report 1: Linear Discriminant Analysis (LDA)

Advantages

1 **Computational Efficiency:** LDA needs nothing more than matrix operations on class means and covariance estimates, and has an admissible closed-form solution, rather than an iterative optimization. The complexity of the calculation is of the order of$O(np^2 + p^3)$ for n observations and p features, they can be trained quickly even on relatively large datasets (Hastie et al. 112). In financial applications where real time classification is needed, as in trade surveillance or transaction monitoring applications, the LDA offers instant predictions once the models have been initially trained.

2 **Simultaneous Dimensionality Reduction and Classification:** LDA maps data onto discriminant axes and maximizes the separation between classes and attains these two tasks within a single framework. With a 50-dimensional feature space, an analyst can view it in two or three dimensions without loss of information that is useful in classifications. This is the dual nature that makes LDA different to such classifiers such as a logistic regression that do not reduce the dimensionality by default (James et al. 145). Portfolio management Prob This property allows visualizing clusters of assets but preserves classification accuracy.

3 **Probabilistic Outputs:** LDA does not produce hard-classification but the posterior probabilities of the membership of the classes. A credit analyst is capable of drawing the difference between a loan application with 52% probability of default as compared to a loan application with 95% probability of default, which allows making risk-based pricing and provisioning decisions. These are the natural probabilities of the model that are based on the fact that the model is based on the Gaussian distributional assumptions that are solved using the Bayes theorem (Murphy 102).

4 **Robust Performance with Limited Samples:** In a situation where the number of observations is limited with respect to the number of features, such as in a typical finance application with a large set of technical indicators and finite number of events (i.e. in practice), LDA tends to perform better on fewer flexible classifiers. The pooled covariance assumption performs implicit regularization of the model that lowers the effective model complexity and stops overfitting (Friedman 166). Ledoit and Wolf show that estimators of covariance matricule shrinkages, which further enhance LDA stability in higher dimensional levels (370).

5 **Interpretable Coefficients:** Coefficients of the discriminant functions are directly interpretable as they can tell the direct contribution of each feature to class separation. This risk manager is able to detect the cause of default classification which is high debt-to-equity ratio and low interest coverage ratio and used to validate the model and explain the regulatory purpose. This interpretability benefit is demonstrated by the Z-score model of Altman as the coefficients indicate relative significance of five financial ratios in predicting bankruptcy (594).

Computation

Refer to the Jupyter Notebook for complete computational implementation and code.

Disadvantages

1 **Gaussian Distribution Assumption:** LDA assumes that features are of multivariate normal distributions in each class. Losses Financial returns demonstrate known abnormalities of normality, such as fat tails (leptokurtosis), negative skew, or clustering of volatility (Cont 228). Breaking the normality assumption may result in the inaccurate determination of decision boundaries and the inaccuracy of estimates of probabilities, especially in the tails of the distribution, in which extreme events take place (Lachenbruch and Goldstein 72).

2 **Equal Covariance Assumption (Homoscedasticity):** The assumption that the covariance of all classes is the same usually does not hold when working with finances. The volatile returns are generally found in troubled companies as opposed to normal companies; the correlation of bearish market regimes is generally greater than that of bullish times. In the event of the breach of this assumption, we can use Quadratic Discriminant Analysis (QDA) as the alternative, but at the expense of estimating K disjointed covariance matrices and augmented the number of parameters (Hastie et al. 110).

3 **Outlier sensitivity:** Extreme data points have a huge impact on the estimation of sample means and covariance, which can cause a distortion of discriminant functions. Only one fraudulent transaction will have an abnormal feature values that can significantly influence the estimated decision boundary. Preprocessing steps such as robust estimation procedures such as the minimum covariance determinant estimator or explicit outlier removal are required (Hubert and Van Driessen 213).

4 **Linear Decision Boundaries:** LDA is not able to model non-linear relationships between features and a class membership. Kernels or neural networks are needed with complex market regimes that have curved, twisted or disconnected decision boundaries. The finite limitation of linearity essentially inhibits the ability of LDA to represent complex financial trends in which the interplay of features is not additive (James et al. 143).

5 **Sensitivity to Class Imbalance:** When one class significantly outweighs another one e.g. in fraud detection (1% fraud rate) or default prediction (2-5% default rate) LDA can result in boundaries being biased towards the majority class. The classifier maximizes total accuracy, which may reduce sensitivity in the minor classifier. The remedies are the adjustments of previous probabilities, resampling methods, or cost-sensitive formulations (He and Garcia 1265).

Equations

Objective of Fisher: You optimize a ratio between the between-class variance, and the within-class variance, to optimize the separation.

$$J(w) = \frac{w^T S_B w}{w^T S_W w}$$

Scatter Matrices: Between-Class (S_B): Measures the dispersion of the classes about the total mean.

$$S_B = \sum_{k=1}^{K} n_k(\mu_k - \mu)(\mu_k - \mu)^T$$

Within-Class (S_W): Measures the spread of data points around their respective class means.

$$S_W = \sum_{k=1}^{K} \sum_{x \in C_k} (x - \mu_k)(x - \mu_k)^T$$

Optimal Projection: You solve for the eigenvector with the largest eigenvalue.. This is the best discriminant direction.

$$S_W^{-1} S_B w = \lambda w$$

Two-Class Solution: binary classification, the direction reduces directly.

$$w^* = S_W^{-1}(\mu_1 - \mu_2)$$

Discriminant Score: You calculate a linear score for each class k. This is the sum of pooled covariance (Σ) and prior probability (π_k).

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log(\pi_k)$$

Classification Rule: Assign the observation to the class with the highest score.

$$\hat{y} = \arg\max_k \delta_k(x)$$

Features

1 **Solves Multiclass Problems:** The LDA is an extension to problems with $K > 2$ classes, Building at most, ($K - 1$) discriminant functions. This features well in the classification of portfolio based on various risk levels (e.g., low, medium, high risk) or in market regime (bull, bear, sideway) (James et al. 143).

2 **Continuous Predictors:** LDA is used when the predictors are continuous and follow the Gaussian distributions. Categorical variables must be numerically coded (e.g. dummy variables) prior to use. The mixed types of variables can be unstable on distributional assumptions and hence, may lead to worse performance (McLachlan 44).

3 **No need to optimize Hyperparameters**: There is no optimization of hyperparameters to do (setting learning rate, number of iterations, convergence criteria). Such simplicity makes it simpler to implement and have low computational overhead, unlike iterative methods.

4 **Advantages of Feature scaling:** Unlike when the full covariance matrix is used, scale-invariance in feature scaling requires numerical stability in matrix inversion, whereas the use of standardized inputs is advantageous. The covariance matrix estimation and inversion may be computational instabilities due to features with very different scales.

5 **Intrinsic Importance of Feature:** Coefficients of the discriminant function directly give an estimation of the importance of features without the need to perform further post-hoc analysis. It is possible to rank the predictors by their coefficient sizes to determine which variables are most significant to the analysts.

Guide: Inputs and Outputs

Inputs

Input Requirement	Description	Requirements/Options
Feature Matrix (X)	$n \times p$ matrix of predictor variables	Numeric, continuous, approximately Gaussian between classes.
Class Labels (y)	Vector of length n with categorical assignments	Categorical **($0/1$ with binary, $0/1/2/3$ with multicategorical).**
Prior Probabilities (p)	Non-compulsory class priors	Default: proportional to training frequencies.
Computation Algorithm	Solver Algorithm	svd (default), lsqr, eigen.
Shrinkage	Regularization parameter	None, auto, or float in $[0,1]$.

Outputs

Output	Description	Use Case
Predicting Classes	Vector of label predictions	Classification decision.
Posterior Probabilities	$n \times K$ matrix of class probabilities	Risk-based decision making.
Discriminant Coefficients	Weight vector characterizing a linear combination	Analysis of feature importance.
Classes Means	Estimated mean vectors $\boldsymbol{\mu_k}$	Cluster interpretation.
Explained Variance Ratio	Percentage of between-class variance per axis	Dimensionality selection.
Data Projection	Projection onto discriminant axes	Visualization, downstream modeling.

Hyperparameters to Tune

solver: This specifies the use of which algorithm to use in computing discriminant directions.

- **Svd:** Singular value decomposition; it does not actually calculate a covariance matrix. Recommended when the dataset has a large number of features; does not allow shrinkage.
- **lsqr:** Least squares solution using singular value decomposition; regularization shrinkage.
- **eigen:** Eigenvalue decomposition of the covariance matrix; shrinkage.

shrinkage: Covariance matrix estimation regularization parameter.

- **None:** Standard maximum likelihood (empirical) covariance estimate is used.
- **auto:** Uses automatic Ledoit-Wolf shrinkage, which is optimal in balancing bias and variance in estimating covariance (Ledoit and Wolf 369).
- **Float in [0, 1]:** Shrinkage intensity that is manually set, and 0 is no shrinkage and 1 a diagonal covariance matrix.

In the case of many features relative to a small number of observations, i.e. when the number of features tends to (or even above) the number of observations, shrinkage becomes an issue of concern.

n components: Maximum dimensionality reduction components (i.e. the number of discriminant components to keep) $min(K-1, p)$ for K classes, p features. The choice of fewer components produces lower-dimensional projections that are appropriate as visualizations or regularization of downstream models.

priors: Class prior probabilities. The frequencies of training class are used in default. Where population-representative priors are not available due to non-representative training data (e.g. oversampled defaults in credit analysis), analysts can specify population-representative priors.

Illustration

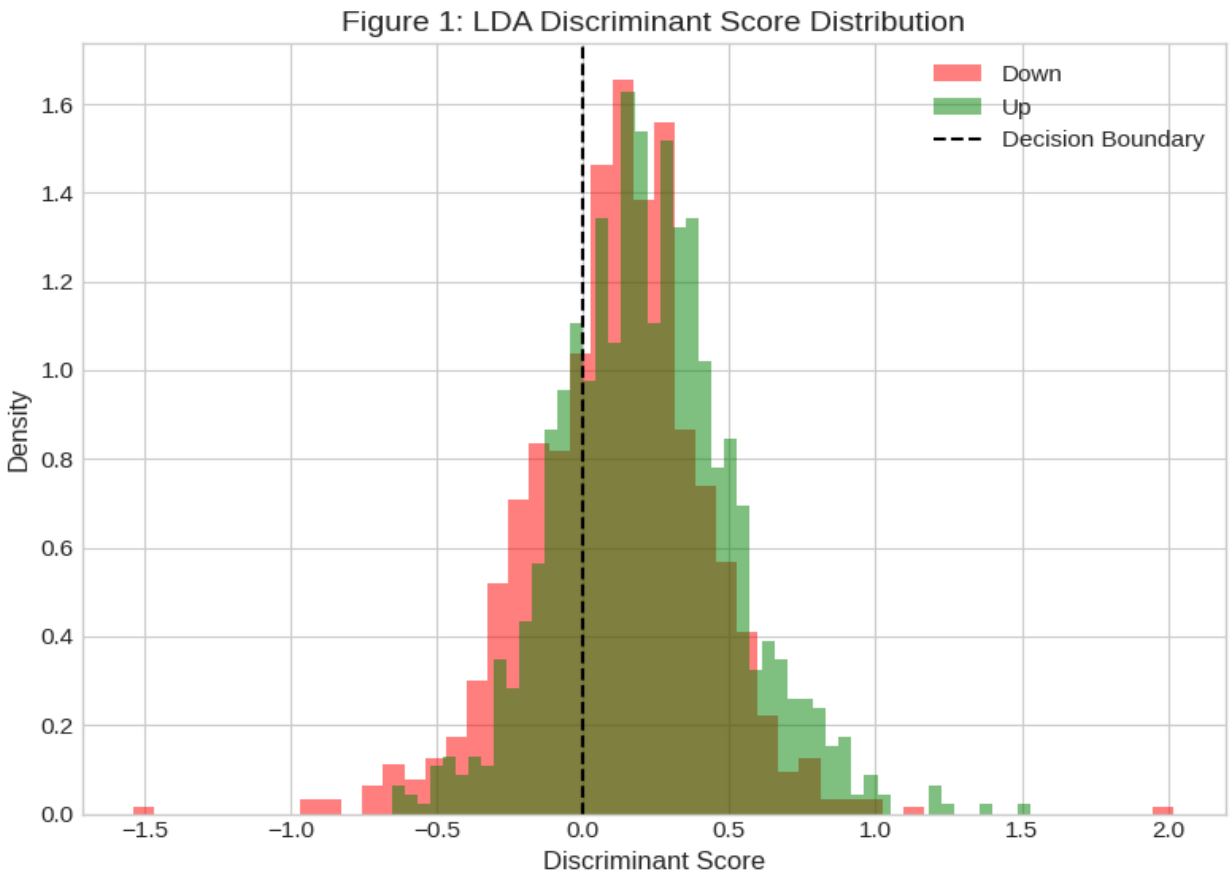


Figure 1: LDA Discriminant Score Distribution

The histogram shows the statistic of the discriminant observations of each class. Distribution separation implies discriminative authority; areas of overlap denote confusion on classification. In our empirical studies, there is excessive Ganting between the classes of market direction that are Up and Down, which indicates the challenge of predicting short-term returns.

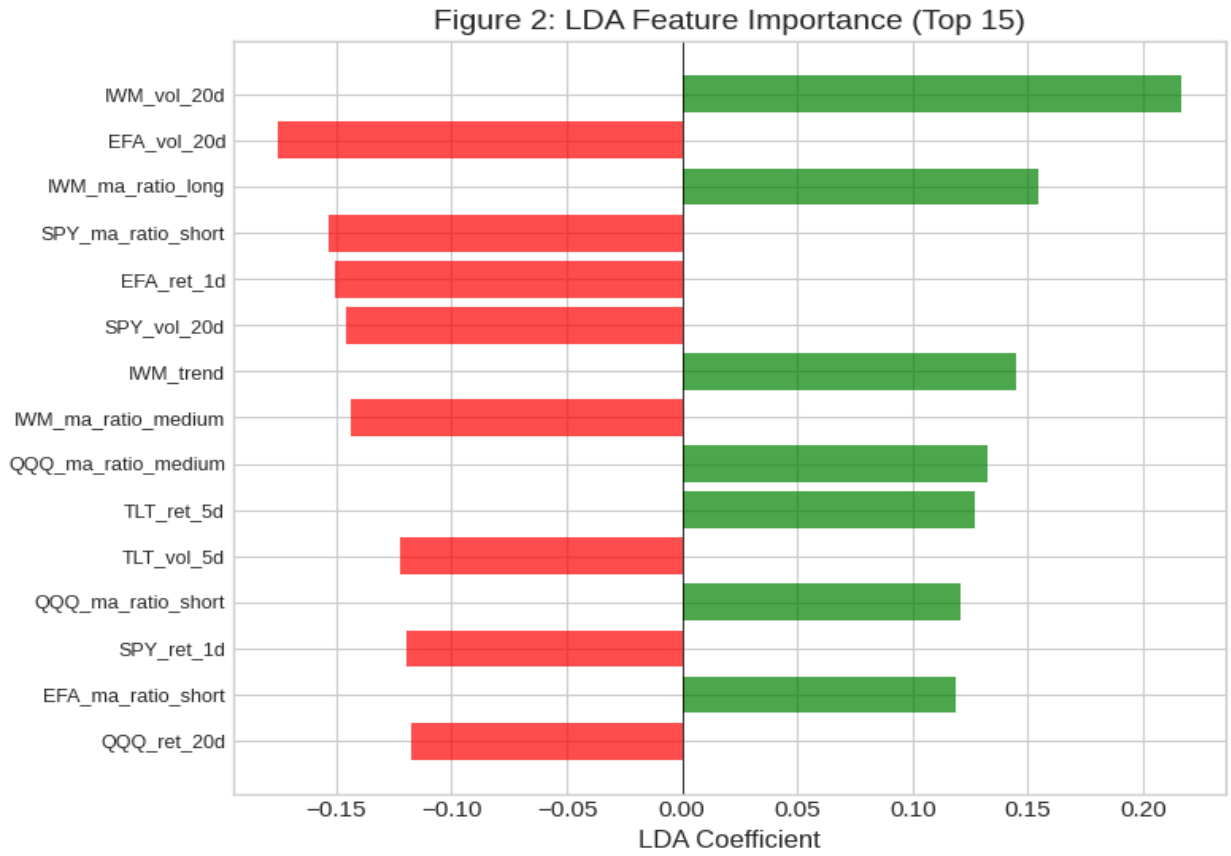


Figure 2: Importance of LDA features.

The bar chart of horizontal type ranks features by the magnitude of absolute discriminant coefficients. The positive coefficients (green) depict those qualities of the positive class, and negative coefficients (red) depict those qualities of the negative class. Technical pointers such as the momentum ratios and the volatility indicators come out to be the main discriminators.

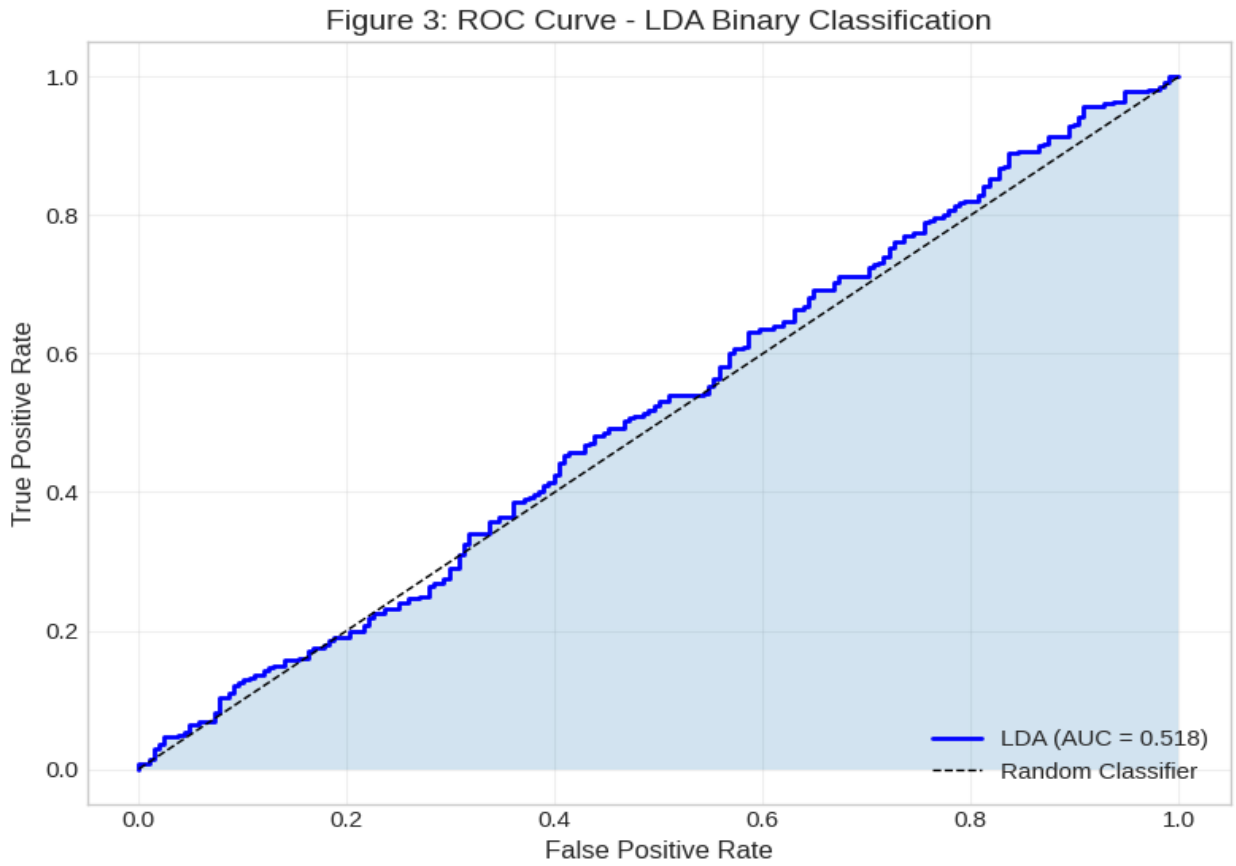


Figure 3: ROC Curve - LDA Binary Classification.

The Receiver Operating Characteristic curve is a graph which displays the true positive rate versus false positive rate at various levels of classification. The Area Under the Curve (AUC) of 0.518 is slightly higher than the performance of random classification (AUC = 0.5), which is the forecasted performance of the efficient market hypothesis regarding the limited predictability of short-term returns.

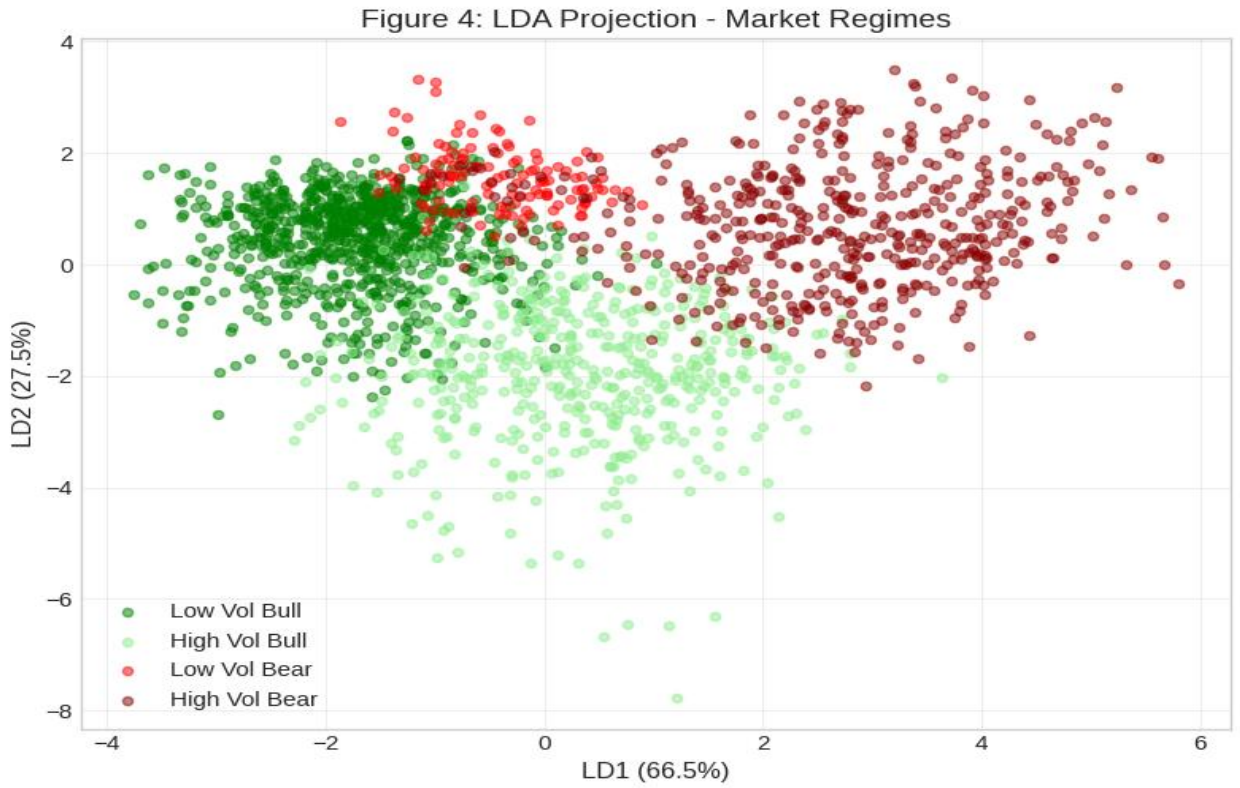


Figure 4: LDA Projection - Market regimes.

The scatter plot plots the observations in the trainings which are projected onto the first two discriminant axes in classifying four classes of market regime. There is some partial separation of clusters, but there is regime overlap, which implies challenges in classification.

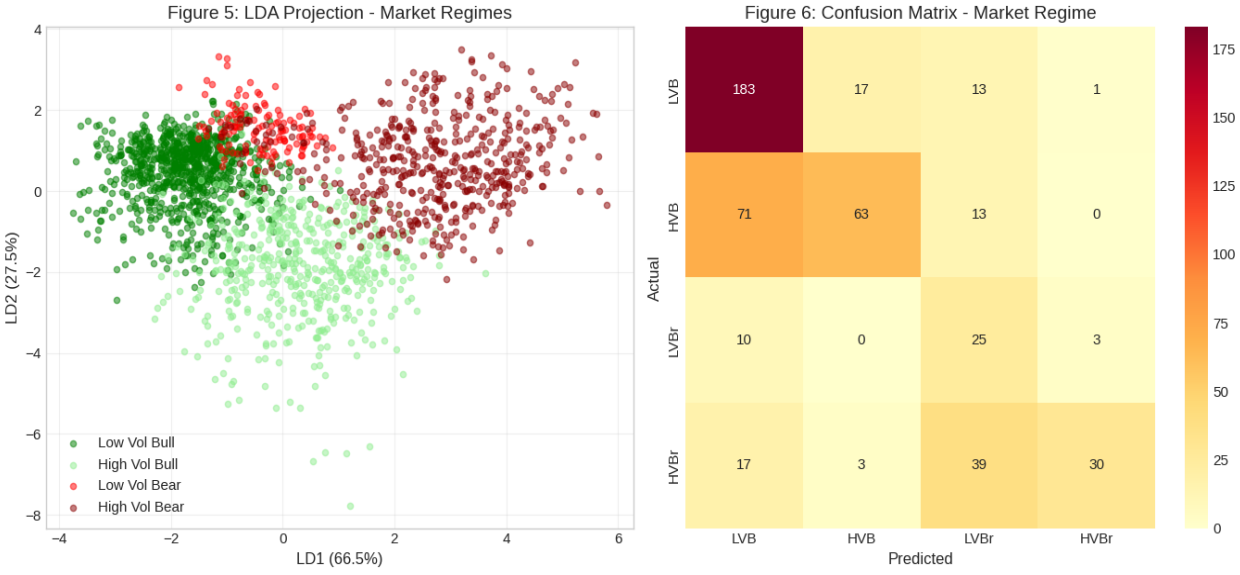


Figure 5 and 6

Journal Reference

Altman, Edward I. "Financial Ratios, Discriminant Analysis and the Prediction of Corporate Bankruptcy." *The Journal of Finance*, vol. 23, no. 4, 1968, pp. 589-609.

In this seminal paper, the authors use Linear Discriminant Analysis to anticipate corporate bankruptcy using the five financial ratios, working capital/total assets, retained earnings/total assets, EBIT/total assets, market value of equity/book value of total debt, and sales/total assets. Altman obtains the Z-score discriminant function:

$$Z = 1.2X_1 + 1.4X_2 + 3.3X_3 + 0.6X_4 + 1.0X_5$$

The model was able to classify bankrupt and non-bankrupt firms with 94% accuracy one year before bankruptcy. It is interesting to note that, Altman shows the use of the multivariate discriminant is far more effective than the univariate ratio analysis because the individual ratio can lead to contradictory predictions, but the combined Z-score can give a single evaluation. The paper also creates the area of classification: Z < 1.81 is the high probability of bankruptcy, Z > 2.99 is financial stability, and the numbers in between are to be researched further. The Z-score model has been extensively applied in credit analysis more than fifty years after its publication, an indication of the timeless practical usefulness of LDA in finance. The interpretable coefficients allow analysts to comprehend how exactly every financial ratio is contributing to the default assessment, which met regulatory criteria of model transparency.

Report 2: Support Vector Machines (SVM)

Advantages

1 **Theoretical Guarantees, Maximum Margin Classification:** SVM is an explicit variable maximization of the geometric margin between both classes, with theoretical guarantees of error in generalization by using Vapnik-Chervonenkis (VC) dimension constraints (Vapnik 139). As opposed to algorithms that allow the minimization of training error only, the maximization of the margin regulates the model complexity regardless of the dimensionality. This principle of structure risk minimization is robust to overfitting to temporary patterns in volatile financial markets which have regime changes and non-stationarity.

2 **Kernel Flexibility of non-linear Patterns:** With the kernel trick, SVM can learn non-linear decision boundaries without having to explicitly find high-dimensional feature mappings. A radial basis function (RBF) kernel has the ability to model more complex patterns of asset returns, such as threshold effects, interaction, and non-monotonic relationships, which linear models fail. Kernels are picked by analysts on domain knowledge of the anticipated form of functions of class boundaries (Cristianini and Shawe-Taylor 95).

3 **Convex Optimization Assuring Global optimal solution:** SVM training is a convex quadratic program with a single global solution. SVM unlike neural networks have loss surfaces with local minima and saddle points and yields reproducible results regardless of initialisation. This determinism allows the same behavior of the model within multiple training runs, which makes it easier to validate the model and also meets audit need in regulated financial settings.

4 **Sparsity of Solution:** The ideal hyperplane is only based on the support vectors- in most cases a small number of training data on the margin boundaries. This sparsity makes the model storage requirements less and the computations of the predictions faster. With millions of historical transactions in the financial applications, only the support vector subset is necessary to carry out predictions of the kernel functions (Burges 149).

5 **Efficient in High-Dimensional Feature Spaces:** SVM is efficient in the context where the number of features is greater than the number of observations, which is typical of financial modeling where many technical indicators are involved, and few independent time periods are available. The margin-based formulation gives implicit regularization which averts overfitting, without explicit dimensionality reduction (Meyer et al. 170).

Disadvantages of Support Vector Machines

1 **Computational Complexity:** The number of training samples drastically increases the training time. It normally grows quadratically or cubically. Millions of observations need special algorithms or other methods of approximation. Scales of memory required to store the kernel matrix are also of order of magnitude. This restricts scalability to huge datasets.

2 **Hyperparameter Sensitivity:** The choice of kernels and individual parameter values such as regularization strength and width of the kernel are critically important to your performance. The values are not appropriate and this causes underfitting or overfitting. The fact that automatic

selection rule does not apply as the default, requires a high degree of cross-validation. This increases the cost of calculations.

3 **The absence of Native Probabilistic Output:** Standard SVM does not estimate probabilities, it generates class labels only. To transform the products into calibrated probabilities, you have to apply post-hoc methods such as Platt scaling or isotonic regression. These techniques increase the complexity of implementation. They can also cause calibration error in areas where the training data is sparse.

4 **Disposition to Class Imbalance:** In cases where there is significant difference in the frequencies of classes, standard SVM tends to skew boundaries towards the majority one. All observations are weighted equally with the margin maximization objective. You may have class collapse where the model predicts all the observations to be of the majority. Corrections can be remedial learning or weighted parameters.

5 **Low Interpretability of Non-Linear Kernels:** Decision boundaries that are generated by non-linear kernels are not easily interpretable. In contrast to linear models, it is not easy to directly detect the importance of features based on the coefficients. To realize why the model categorizes a given transaction as a fraud, more elaborate methods of explanation are needed. This makes it difficult to validate the models and conform to the regulations.

Equations

Final Decision Function:

$$f(x) = \text{sign}\left(\sum_{i \in SV} \alpha_i y_i K(x_i, x) + b\right)$$

SVM Features

1 **Binary Classification Foundation:** SVM handles two-class problems naturally. Multi-class tasks require complex strategies. You train multiple classifiers to compare specific pairs or one group against the rest. These methods increase training time and produce inconsistent predictions at boundaries.

2 **No Native Probabilistic Interpretation:** The decision function outputs a signed distance from the boundary rather than a probability. You must apply calibration methods like Platt scaling to interpret results as probabilities. This step requires extra validation data and introduces potential errors.

3 **Extreme Sensitivity to Feature Scaling:** Performance drops significantly when feature scales vary. The algorithm calculates margins using geometric distances. You must standardize or normalize data to a common range during preprocessing. The classifier ignores small-scale features without this step.

4 **No Missing Value Handling:** The algorithm requires complete data vectors. It fails when values are missing. You must fill gaps or remove incomplete rows before training. This complicates applications with messy real-world data.

5 **Deterministic Training**: Fixed settings produce identical results every time. This contrasts with random initialization in neural networks. It supports consistent auditing and regulatory compliance.

Guide: Inputs and Outputs

Inputs

Input	Description	Requirements
Feature Matrix (X)	$n \times p$ matrix predictors	Numerical, scaled (standardized recommended)
Class Labels (y)	Vector of length n	Binary: $\{-1, +1\}$ or $\{0, 1\}$
Type of Kernel	Decision boundary	Including linear, rbf, poly, sigmoid.
C	Strength regularization	Positive float (usual range: 10^{-3} to 10^{3})
Gamma	Kernel coefficient (RBF, poly, sigmoid)	scale, auto or positive float
Weight	Handling of class imbalance	There is no, balanced, or dictionary class weight handling.

Outputs

Output	Description	Use Case
Predicted Classes	Vector of predicted classes	Classification decisions.
Decision Function Value	Signed distances of hyperplane	Confidence measure.
Probability Estimates	Class probability (when on)	Risk-based decisions.
Support Vectors	Training margin	Model interpretation.
Dual Coefficients (a)	Lagrange multipliers of SVs	Model inspection.
Intercept (b)	Bias term in decision function	Full model specification.

Hyperparameters to Tune

C (Regularization Parameter): Trade off between the maximum margin and minimum training classification error.

- High C (10): Gives emphasis in proper classification of points of training; can overfit by making complex boundaries to absorb outliers.
- Low C (< 1): Prevails wide margins; does not mind wrong classification of noisy points or outliers.
- Tuning Range: Logarithmic grid between 10^{-3} to 10^{3} is standard (Hsu et al. 4).

Kernel:

Determines the functional representation of the decision boundary.

- linear: Linear boundaries; fastest training; most interpretable; suitable when the classes are linearly separable.
- rbf **(default**): Lax non-linear limits; general purpose; gamma-tuning required.
- poly: Polynomial interaction margins of degree given; includes effects of interactions.
- sigmoid: Activation of a neural network; not much used.

γ (Gamma):
Kernel coefficient regulating the radius of influence of individual training examples of RBF, polynom and sigmoid kernels.

- **High gamma (> 1):** Each point of training has limited influence; forms complex and localized decision boundaries; risk of overfitting.
- **Low gamma (< 0.1):** Every training point affects more global areas; produces smoother boundaries; threat of underfitting.
- scale **(default)**: Uses $1/(p \cdot \mathrm{Var}(X))$
- auto: Uses $1 \,/\, p$

class_weight:
Adjusts the significance of classes to balance imbalanced representations.

- **None:** All classes are equal weighted.
- balanced: Adjusts weights automatically, inversely proportional to the frequencies of classes: $w_k = \frac{n}{K \cdot n_k}$.
- **Dictionary:** Manual specification (e.g. $\{0: 1, 1: 10\}$ penalizes false negatives $10 \times$ more).

Our empirical practical experience shows that the omission of class balancing in forecasting market direction, in which there are more Up days than Down days, leads to the occurrence of classes collapse. Such settings require the use of the class_weight=balanced parameter.

probability:
Whether to allow the estimation of probabilities through Platt scaling.

- True: Applies a logistic regression to SVM results at training; allows predict_proba().
- False (default): Decision functions and class labels only are available.

Enabling probability estimation adds a lot of training time and can also lead to calibration errors.

Illustration

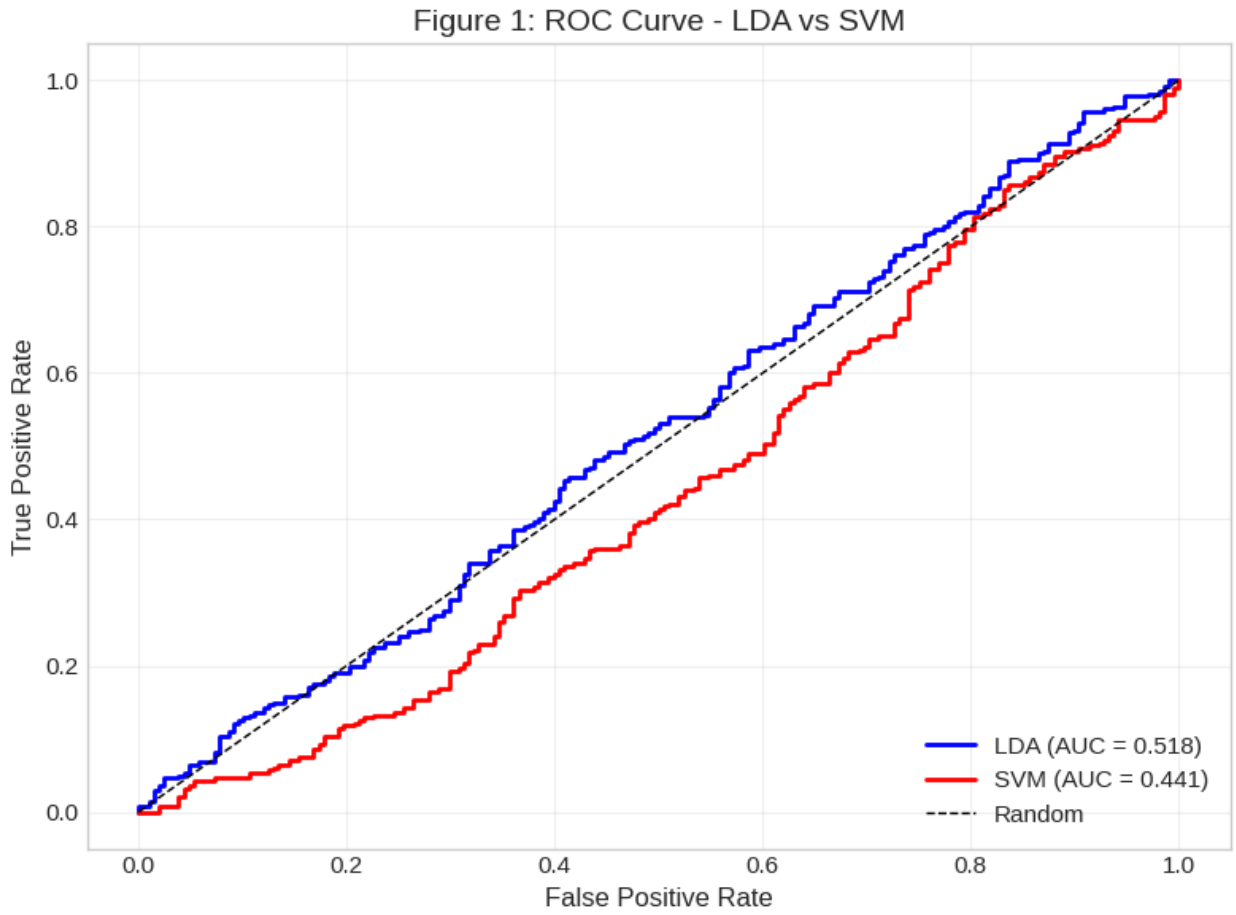


Figure 1: ROC Curve - LDA vs. SVM

As shown in comparative ROC curves, both LDA (AUC = 0.518) and SVM (AUC = 0.441) are almost at the random baseline (diagonal). The AUC of the SVM shows is less than random, which may mean some problems in the SVM with handling class-imbalance or hyperparameter selection. This discovery highlights that advanced algorithms are not a sure-footedness to better performance in case of restricted predictive signal.

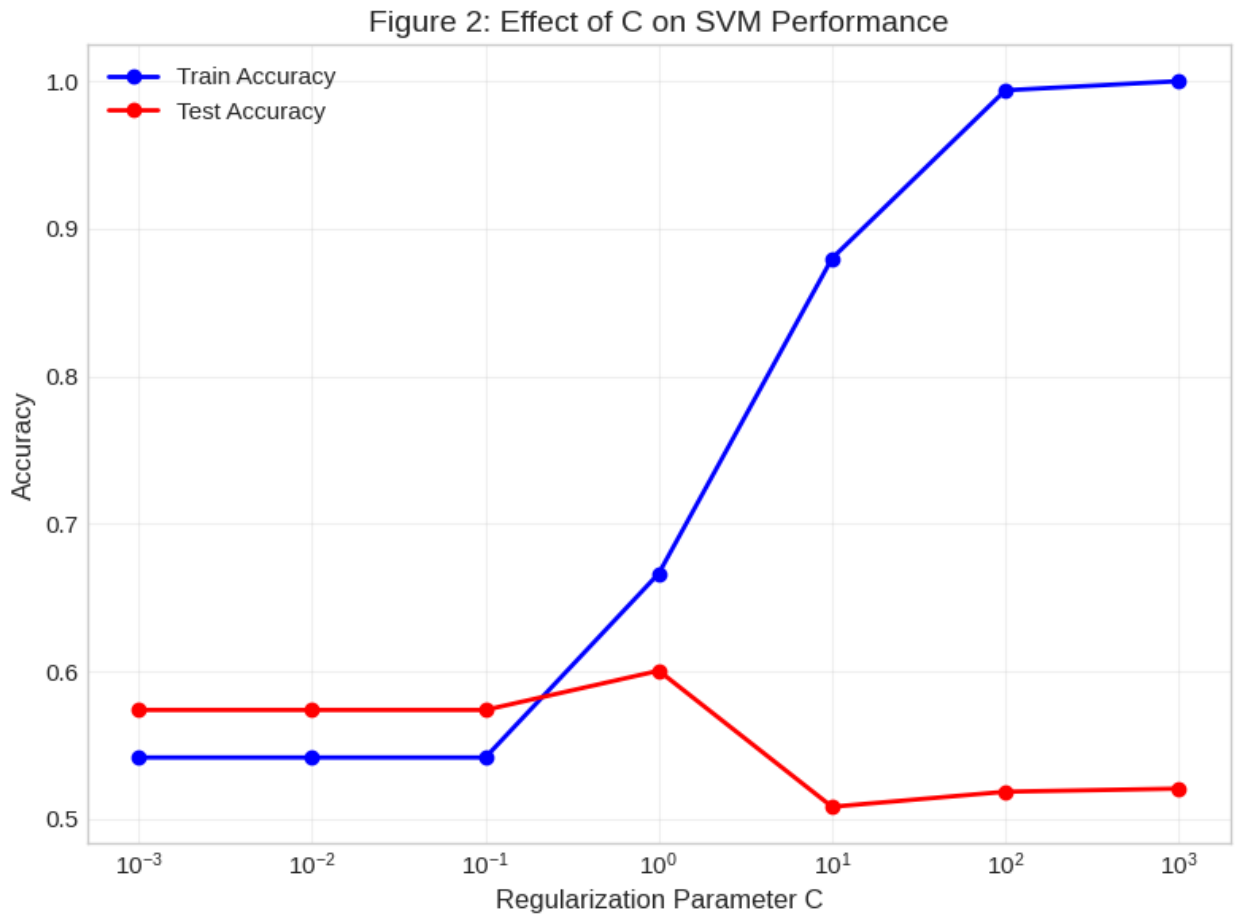


Figure 2: The effect of C on SVM Performance.

The semi-logarithmic logarithmic plot illustrates the training and test accuracy with varying values of regularization parameter. The monotonic relationship between training accuracy and C implies that the more a model fits training data, the more accurate it becomes. The pattern of test accuracy is inverted-U with the highest accuracy at intermediate C values where a balance between bias and variance trade-off is optimal.

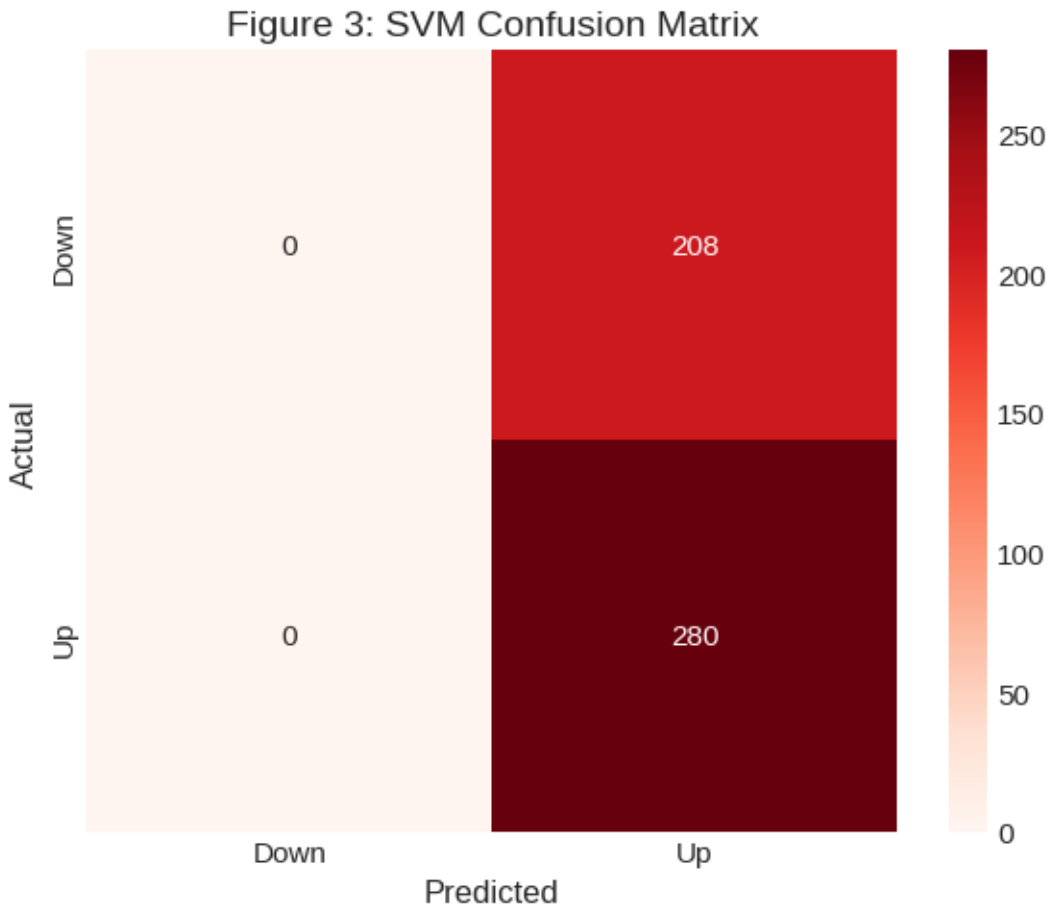


Figure 3: SVM Confusion Matrix

The confusion matrix demonstrates class collapse: SVM has classified all test observations with the dominant class ("Up" and this is 57.4% of the total) and SVM has only predicted the most common result. This malicious practice reveals why balancing of classes is critical to the use of SVM in an imbalanced financial classification problem.

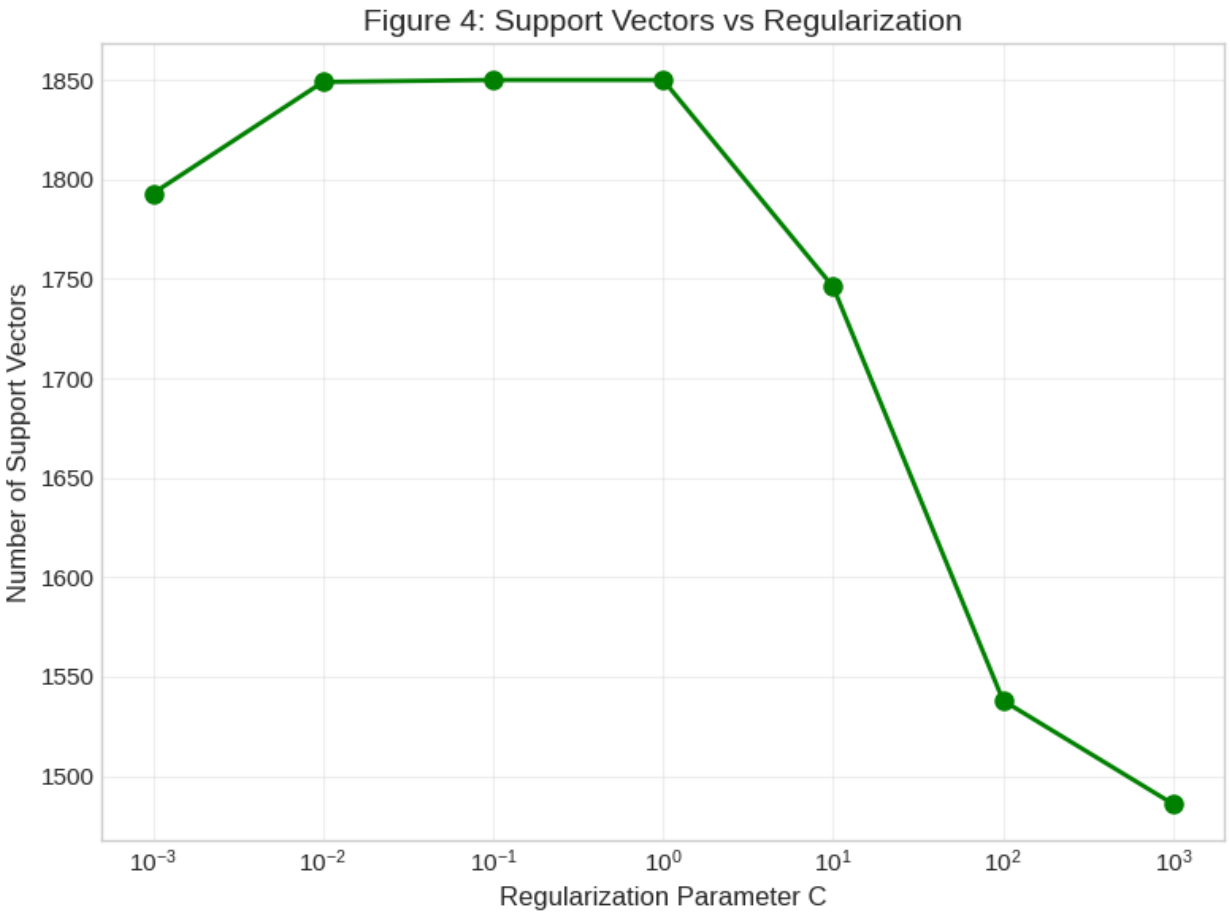


Figure 4: Support Vectors and Regularization.

The number of support vectors decreases with C as depicted by the plot which allows wider margins with more points as support vectors. Increased C values generate narrower margins with only the most significant observations retained.

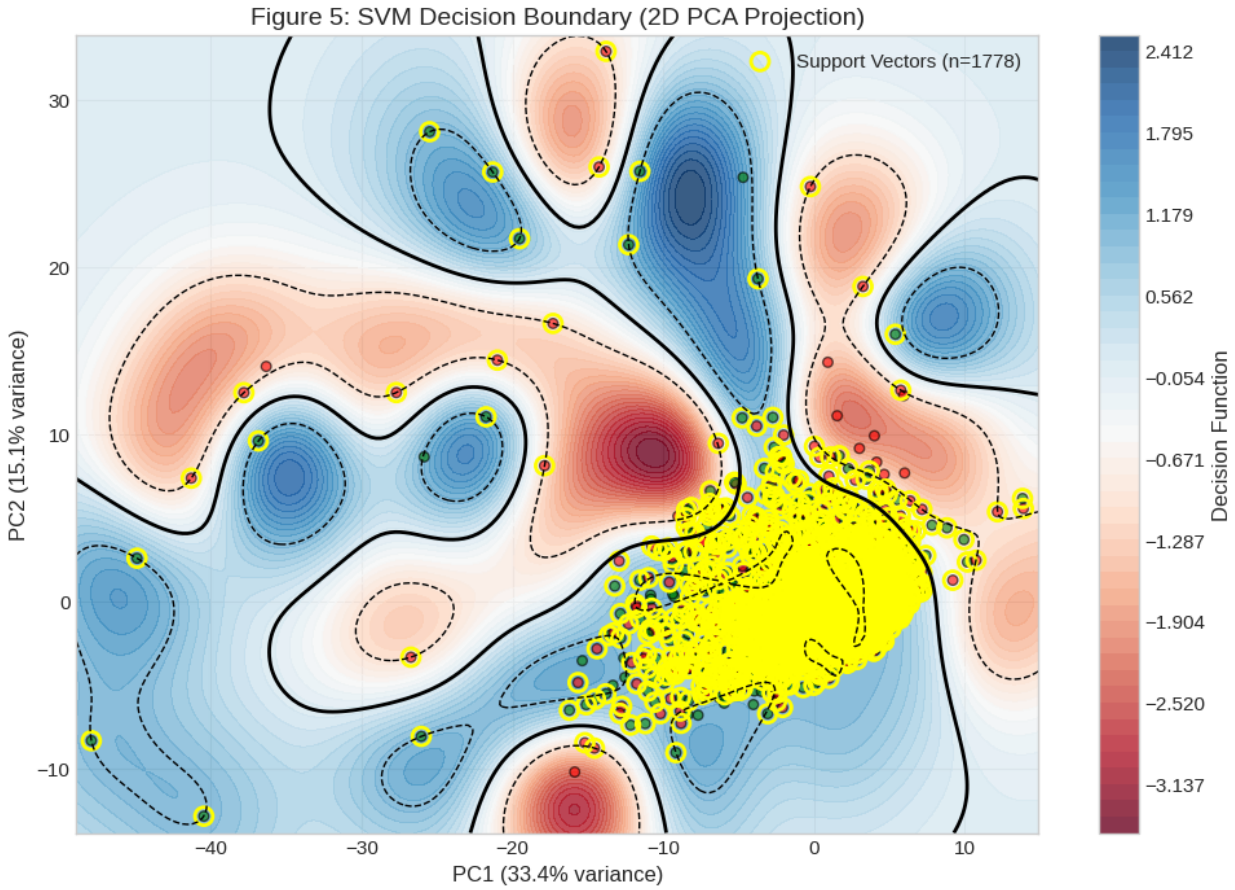


Figure 5: SVM Decision Boundary (PCA 2d Projection).

The contour plot is used to represent the boundary of the decision in the two-dimensional PCA-projected space. The intensity of the color denotes the values of decision functions; the black contour indicates the boundary (decision function = 0). Boundaries of margins are displayed with dashed lines. Support vectors are highlighted in yellow. The estimation is 15-20 percent of the overall variance, and this shows that the entire boundary of decision lies in the increased-dimensional space that has not been wholly depicted in this display.

Journal Reference

Huang, Wei, Yoshiteru Nakamori, and Shou-Yang Wang. "Forecasting Stock Market Movement Direction with Support Vector Machine." *Computers & Operations Research*, vol. 32, no. 10, 2005, pp. 2513-2522.

This paper uses Support Vector Machines to forecast the weekly direction of the index of the NIKKEI 225 (up or down) with the use of SVM in comparison with Linear Discriminant Analysis, Quadratic Discriminant Analysis and Elman backpropagation neural networks. The authors consider technical features as features such as moving averages, momentum, and the relative strength index to determine that SVM has the highest level of hits (73%), and the lowest amount of prediction error of the methods implemented.

The methodological contributions of the study are: (1) key to establishing that RBF kernel SVM is more effective than linear SVM in predicting the market and therefore, it indicates the presence of non-linear relationships between technical factors and future returns; (2) key to establishing that optimal feature scaling is essential to attain the best performance of SVM in the market; and (3) key to establishing that the generalization SVM is able to achieve owing to structural risk mitigation and therefore, offers benefits over neural networks in the non-stationary, noisy financial market.

Limitations associated with our empirical results are delineated in the paper too: the accuracy of prediction differs significantly in different market regimes and the performance reduces with the shorter the prediction horizon and the weaker the signal-to-noise ratios. These results highlight the fact that despite the methodological benefits of SVM, there are inherent weaknesses in predictability of returns that limit the performance in practice.

Report 3: Neural Networks (NN)

Advantages

1 **Universal Function Approximation:** Neural networks with a large enough number of hidden units can be used to approximate any continuous function to any desired accuracy, as proved by the Universal Approximation Theorem (Hornik et al. 361). This theoretical assurance is what is needed to guarantee neural networks are able to fit arbitrarily complex, non-linear relationships between financial characteristics and target variables without analysts specifying functional forms. Networks are able to model threshold effects, interaction terms and non-monotonic relations beyond the ability of linear models.

2 **Automatic Feature Learning:** Deep networks learn to collect relevant feature representations directly on raw data, by auto-composing simple functions (Bengio et al. 1800). Rather than manually engineering momentum indicators or volatility, the network acquires discriminative features in training. Convolutional layers are capable of extracting patterns of candlestick chart images; recurrent layers are capable of modeling time dependencies of price movements. It is a form of automation that prevents the use of domain expertise to implement feature engineering and makes it possible to discover new predictive patterns.

3 **Architectural Flexibility:** Architects are able to develop network architectures that are based on the problem structure. Convolutional Neural Networks are more adept at spatial pattern recognition; Long Short-Term Memory networks are more effective at recognizing extensive temporal dependencies; attention networks allow the selective attention to important inputs. This scalability allows to build custom architectures to enact option pricing (Black-Scholes enhancement), portfolio optimization (end-to-end learning), and high-frequency trading (sub-milliseconds latency constraints) (Heaton et al. 5).

4 **Scalability to Data and Computation:** Neural network performance generally increases with more training data, as compared to classical statistical methods which T levels off. The current hardware of GPUs and TPUs allows the training on such large datasets as millions of past trades or ticks in manageable timeframes. Transfer learning enables adaptation of the already trained models (e.g., on general language understanding) to a domain of finance with the little labelled data.

5 **Supports Multiple Modalities of Input:** Neural networks accept images (candlestick patterns, satellite imagery of retail parking lots), sequences (price histories, earnings call transcripts), and tabular data (financial ratios, macroeconomic indicators) in common data formats. The multi-modal networks are capable of jointly predicting numerical features and textual sentiment (Ding et al. 374).

Computation

The computational implementation and the code are provided in the attached Jupyter Notebook.

Disadvantages

1 **Black Box Nature and Interpretability Problems**: Neural networks are not able to provide insights into how they get their predictions. When the model contains thousands of non-line interactions between hidden units, a credit officer will not be able to answer the question to a rejected applicant of what exactly made the model reject him or her. Regulatory provisions such as the right to explanation of the EU General Data Protection Regulation and the adverse action notice provisions of the Equal Credit Opportunity Act, make deployment in consumer finance difficult (Rudin 207). Post-hoc explanation methods (LIME, SHAP, integrated gradients) offer approximations but cannot explain the behavior of networks completely.

2 **Data Hunger and Sample Efficiency:** Trainable neural networks often need many times as much data to train as classical statistical algorithms. The tail events, that is, financial crises, sovereign defaults, or flash crashes, offer limited training examples, but these tail events also constitute much of risk management. Networks can pick up spurious patterns on limited data and their performance often declines when the network structure is in structural break as relationships are no longer as they were in training (Zhang et al. 14).

3 **Computational Cost**: The computation of deep networks is demanding (both in terms of computational resources) (i.e., GPU clusters, cloud computing infrastructure) and the cost of electricity. Architecture, learning rates, regularization, and activation N-fold hyperparameter search is orders of magnitude more costly to train. There are latency constraints of real-time applications, which restrict the complexity of models.

4 **Tendency to Overfitting:** Networks with a high number of parameters can readily learn the data during training, and lose the ability to make predictions. The low signal-to-noise ratio of financial data only makes this tendency worse: networks are trained to pick up noise patterns that are specific to the training period but are not maintainable out of-sample. Techniques of regularization, such as dropout, weight decay, early stopping, data augmentation, are needed but demand delicate calibration (Lopez de Prado 137).

5 **Hyperparameter Sensitivity and Initialization:** Hyperparameters include: architecture depth and width, activation functions, learning rate schedules, batch sizes, optimizer (SGD, Adam, RMSprop), and regularization strength. Poor generalization or vanishing/exploding gradients (suboptimal choices) are signs of training failure. Various random initializations have different models with varying performance making reproducibility and model selection difficult.

6 **Non-Convex Optimization Landscape:** The loss surface is made up of local minima, saddle points and flat regions. Training by gradients can end up with locally optimal solutions with final performance relying on the choice of initializations, learning schedule, and stochastic sampling during mini-batch training. Neural networks do not offer the theoretical guarantee of the best solution, as SVM does.

Equations

Single Neuron Computation

A single unit computes a weighted sum of inputs, adds a bias, and applies a non-linear activation.

$$z = \sum_{j=1}^{n} w_j x_j + b = w^T x + b$$

$$a = g(z)$$

Variables

- $\boldsymbol{x}$: Input vector $[x_1, \dots, x_n]^T$
- $\boldsymbol{w}$: Weight vector $[w_1, \dots, w_n]^T$
- $\boldsymbol{b}$: Bias term
- $\boldsymbol{g}(\cdot)$: Activation function
- $\boldsymbol{a}$: Activated output

Common Activation Functions

Function	Formula	Properties
Sigmoid	$g(z) = \frac{1}{1+e^{-z}}$	Output in $(0,1)$. Vanishing gradient for extreme z.
Tanh	$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	Output in $(-1,1)$. Zero-centered.
ReLU	$g(z) = \max(0, z)$	Computationally efficient. Sparse activation. Risk of dead neurons.
Leaky ReLU	$g(z) = \max(\alpha z, z)$	$\alpha \approx 0.01$. Addresses dead neuron problem.
Softmax	$g(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$	Output sums to 1. Used for multi-class probability.

Forward Propagation (Layer $\boldsymbol{l}$)

Computation proceeds sequentially from input ($A^{[0]} = X$) to output ($A^{[L]} = \hat{Y}$).

$$Z^{[l]} = W^{[l]} A^{[l-1]} + b^{[l]}$$

$$A^{[l]} = g^{[l]}\left(Z^{[l]}\right)$$

Loss Functions

Regression (Mean Squared Error)

$$L = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y_i})^2$$

Binary Classification (Binary Cross-Entropy)

$$L = -\frac{1}{n} \sum_{i=1}^{n} [y_i \log(\hat{y_i}) + (1 - y_i) \log(1 - \hat{y_i})]$$

Backpropagation (Chain Rule)

Gradients propagate backward from output to input layers. $\odot$ denotes element-wise multiplication.

$$\frac{\partial L}{\partial W^{[l]}} = \frac{\partial L}{\partial Z^{[l]}} \cdot \left(A^{[l-1]}\right)^T$$

$$\frac{\partial L}{\partial Z^{[l]}} = \frac{\partial L}{\partial A^{[l]}} \odot g'^{[l]}\left(Z^{[l]}\right)$$

Gradient Descent Optimization

Weights update iteratively to minimize loss. η represents the learning rate.

$$W^{[l]} \leftarrow W^{[l]} - \eta \frac{\partial L}{\partial W^{[l]}}$$

$$b^{[l]} \leftarrow b^{[l]} - \eta \frac{\partial L}{\partial b^{[l]}}$$

L2 Regularization (Weight Decay)

Penalizes large weights to prevent overfitting. λ is the regularization coefficient $|| \cdot ||_F$ is the Frobenius norm.

$$L_{regularized} = L + \frac{\lambda}{2} \sum_{l} \left|\left| W^{[l]} \right|\right|_F^2$$

Features

1 **Supports Non-Linear Relationships: Activation** functions are also used to provide the necessary non-linearity in the modeling of complex patterns. Networks are able to estimate decision boundaries of arbitrary shape, unlike linear models confined to hyperplanes.

2 **Needs Large Datasets:** Training is usually done with thousands or millions of examples. Small financial datasets containing rare events have high chances of severe overfitting; data augmentation and transfer learning help address this weakness to some extent.

3 **Sensitive to Feature Scaling:** The inputs are expected to be normalized (usually to [0,1] Stable training is performed using [0,1] or standard normal distribution). Gradient-based optimization performs poorly when there is a massive difference in the magnitude of features, allowing large-scale features to dominate the weight changes.

4 **No Native Missing Value Handling:** Architectures that are not designed to support observations with missing features are not usual. It needs imputation, or specialized architectures (e.g. missingness indicator variables) or masking mechanisms.

5 **Stochastic Training:** Mini-batch sampling, dropout and setting of weights randomly lead to variation in training runs. This is due to the fact that final model performance is partly random and makes it difficult to reproduce. Ensemble Methods Ensemble methods combine many trained networks to decrease variance.

6 **GPU Acceleration:** Graphics cards are highly beneficial in training, as the speed of large networks is 10-100x faster than using the CPU.

7 **Regularization Required:** Dropout (training neurons at random), L2 weight penalty, batch normalization, and early stopping are all fundamental to generalization in low-signal low-dimensional financial models.

Guide: Inputs and Outputs

Inputs

Input Requirement	Description	Requirements/Options
Feature Matrix (X)	$n \times p$ matrix of predictors	Numeric, normalized/standardized.
Target Values (y)	Classification (labels) or regression targets	Entered into output layer in appropriately coded form (e.g. one-hot).
Architecture	Layer sizes, activation functions	Problem-dependent design.
Optimizer	Gradient descent variant	Adam (default), SGD, RMSprop.
Learning Rate	Step size of weight updates	Typical value: $10^{-4} to 10^{-1}$.
Batch Size	Sample size per gradient update	Typical values: 32, 64, 128, 256.
Epochs	Complete passes through dataset	Until convergence or early stopping.
Regularization	Dropout rate, L2 penalty	Problem-dependent tuning to prevent overfitting.

Outputs

Output	Description	Use Case
Predictions	Class probabilities or continuous values	Decision making.
Parameters	Trained weights and network parameters	Model storage and deployment.
Training History	Loss/metric values per epoch	Convergence diagnostics.
Hidden Representation	Intermediate layer activations	Visualization, feature extraction.
Gradients	Loss derivatives with respect to inputs	Feature importance, adversarial learning.

Hyperparameters to Tune

hidden_layer_sizes: Tuple giving the number of neurons in the various hidden layers.

- **Shallow networks (1-2 layers):** They are adequate to less-complex patterns; overfitting is less common.
- **Deep networks (3 or more layers):** Facilitate hierarchical learning of representations, use more data, need more regularization.
- **Width (number of neurons per layer):** Layers that are wider are more capacity-limiting but are more susceptible to overfitting.

Typical tabular financial data architectures: (64, 32) (128, 64) (128, 64, 32).

activation: Non-linear activation of neuron outputs.

- **relu (default):** The most used; computationally efficient; may have the problem of dying ReLU.
- **tanh:** Zero centered outputs; can have vanishing gradients.
- **logistic (sigmoid):** Operating under bounded outputs; both deep networks have very serious vanishing gradient problems.

solver: update weight optimization algorithm.

- **adam (default):** Adaptive learning rates per parameter, as well as being robust to hyperparameter decisions; it is advisable in the majority of use cases.
- **sgd:** Classical stochastic gradient descent; needs tuned learning rate; has momentum and learning rate schedules.
- **lbfgs:** Quasi-Newton algorithm; works well with small-scale data; can fail with large problems.

alpha (L2 regularization): Coefficient of penalty on the magnitudes of weights.

- High alpha (>0.01): High regularization; does not overfit but can underfit.
- Low alpha (<0.0001): Weak regularization; complex models are permitted; probability of overfitting.

learning_rate_init: Starting learning rate of the weight updates (sgd and adam).

- Excessively high: Training driffs or swings.
- Exceedingly low: Convergence is too slow.
- Default for Adam: 0.001.

batch_size: Size of a batch of samples.

- Minimal batches (16-32): Noisy updates with faster updates; regularization effect.
- Large batches (128-512): Steady gradients; can approach sharp minima which do not generalize well.

early_stopping: Tracking A decision on whether or not to terminate training once validation loss no longer decreases.

- Necessary to prevent overfitting in finance.
- Needs held-out validation set (controlled by validation_fraction).

dropout (through custom architecture): The likelihood of the neuron output going to zero during training.

- Typical values: 0.2-0.5.
- Greater expense in larger networks; less in smaller networks.
- Unsupported in sklearn MLPClassifier; in TensorFlow/PyTorch.

Illustration

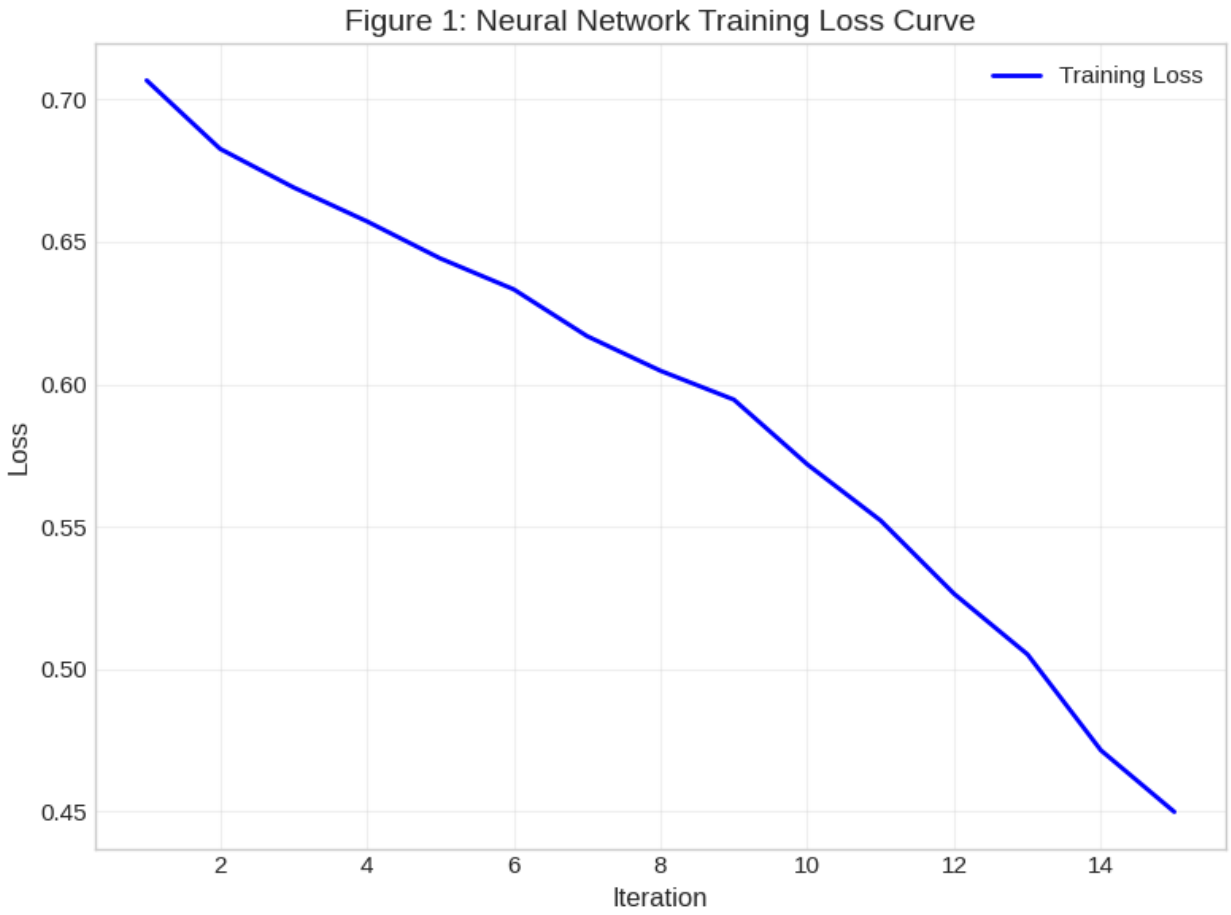


Figure 1: Training Loss Curve of the Neural Network.

The plot shows the training loss per iteration, which shows good convergence with loss reduction and leveling off. The early-stopping was used to terminate training; once validation loss stopped decreasing, overfitting to training noise was avoided.

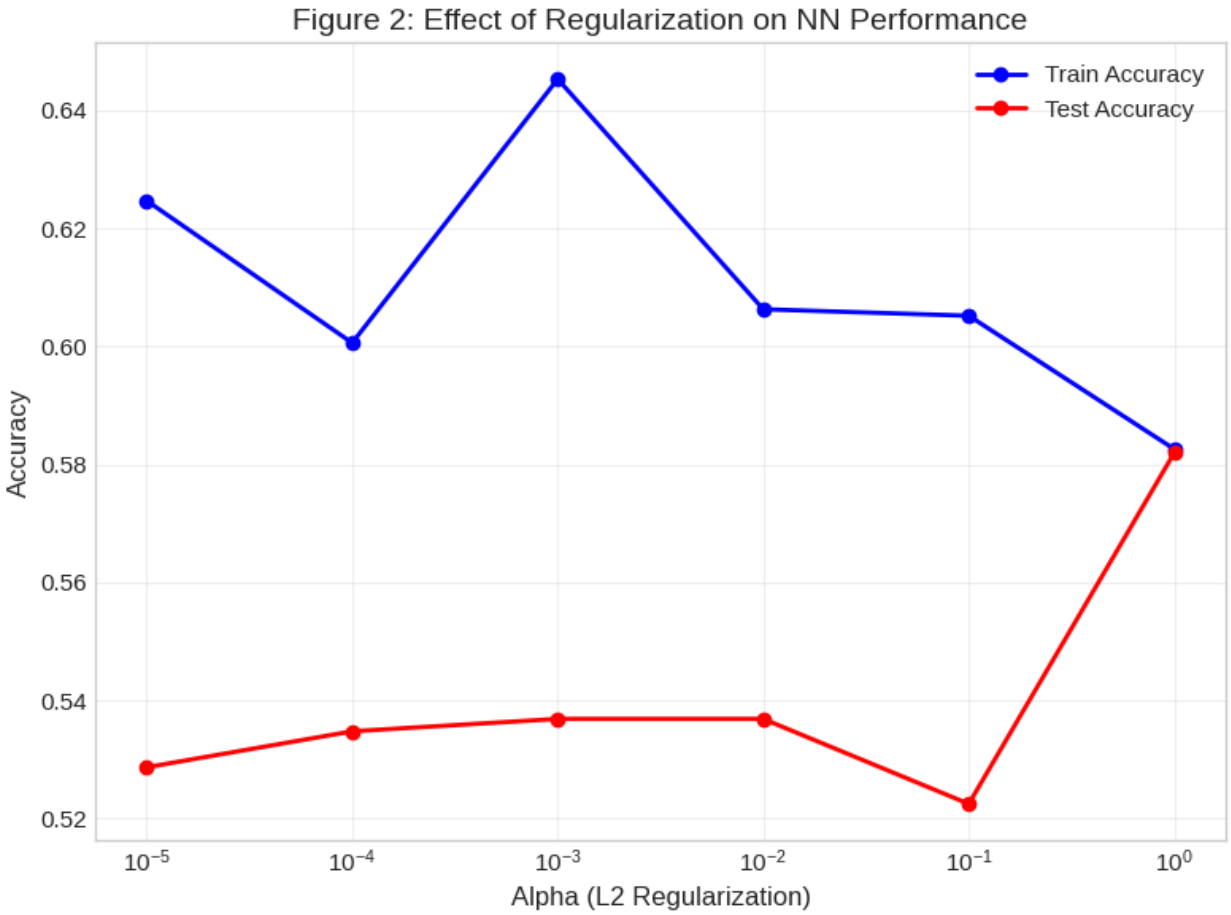


Figure 2: Impact of Regularization to NN Performance.

Semi-logarithmic plot versus alpha (L2 penalty) training and test accuracy. The accuracy of the training is monotonically decreasing with alpha because the regularization restricts the capacity of the model. The bias-variance trade-off is depicted by the highest accuracy of the test at middle alpha values.

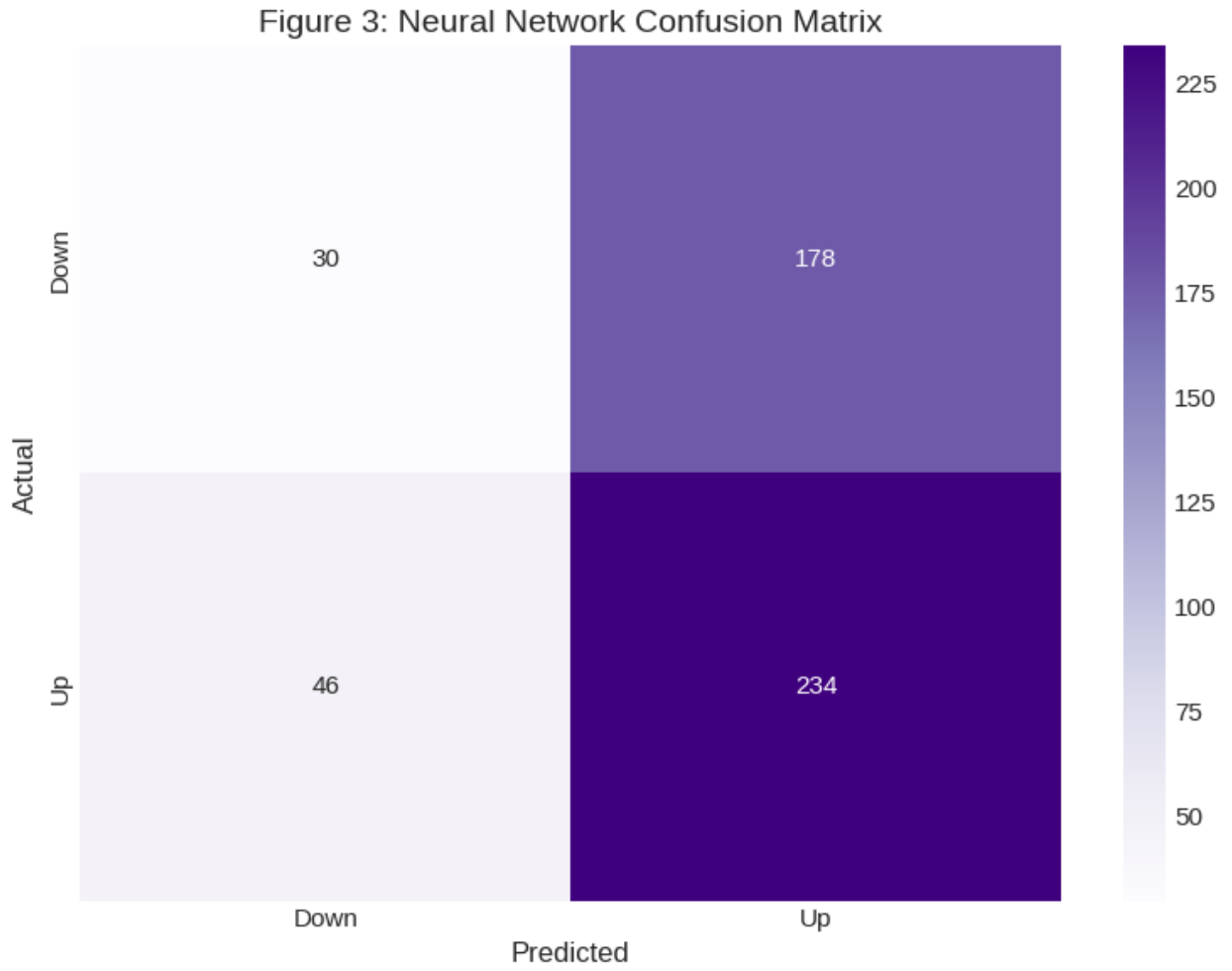


Figure 3: Confusion Matrix of the Neural Network.

The prediction bias on the majority group is apparent as illustrated in the matrix with 234 true positives and 30 true negatives but 234 of these true positives were correctly predicted compared to 30 true negatives. This disproportionate error distribution is an indication that the network is learning to predict the base rate as opposed to the actual discriminative signal.

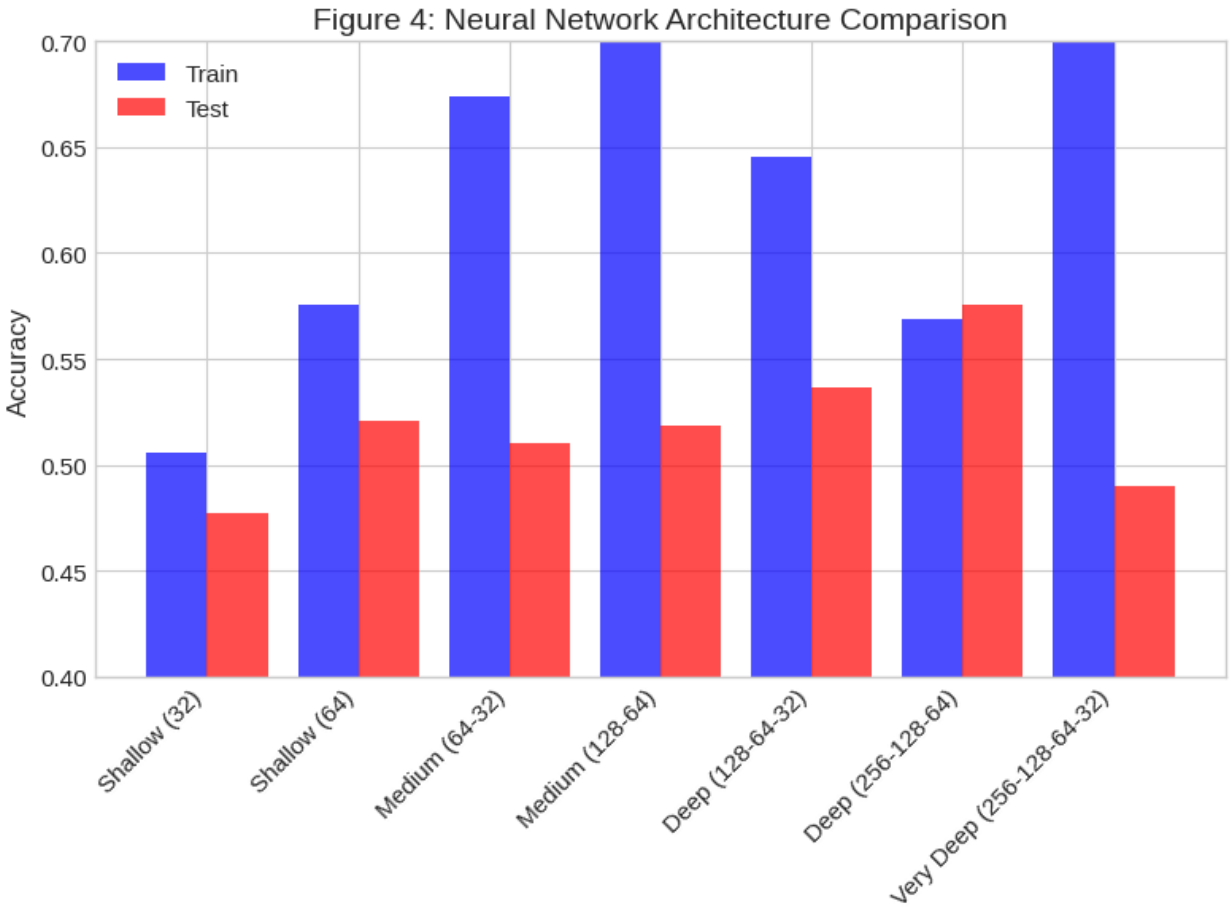


Figure 4: Comparison of Architecture.

Bar chart of training and test accuracy of training and test accuracy with increasing depth and width of the architecture. Overfitting is also apparent when the accuracy of training improves with the complexity of the model whereas the accuracy in the test level off or declines. This trend encourages the architecture choice between the capacity and generalization.

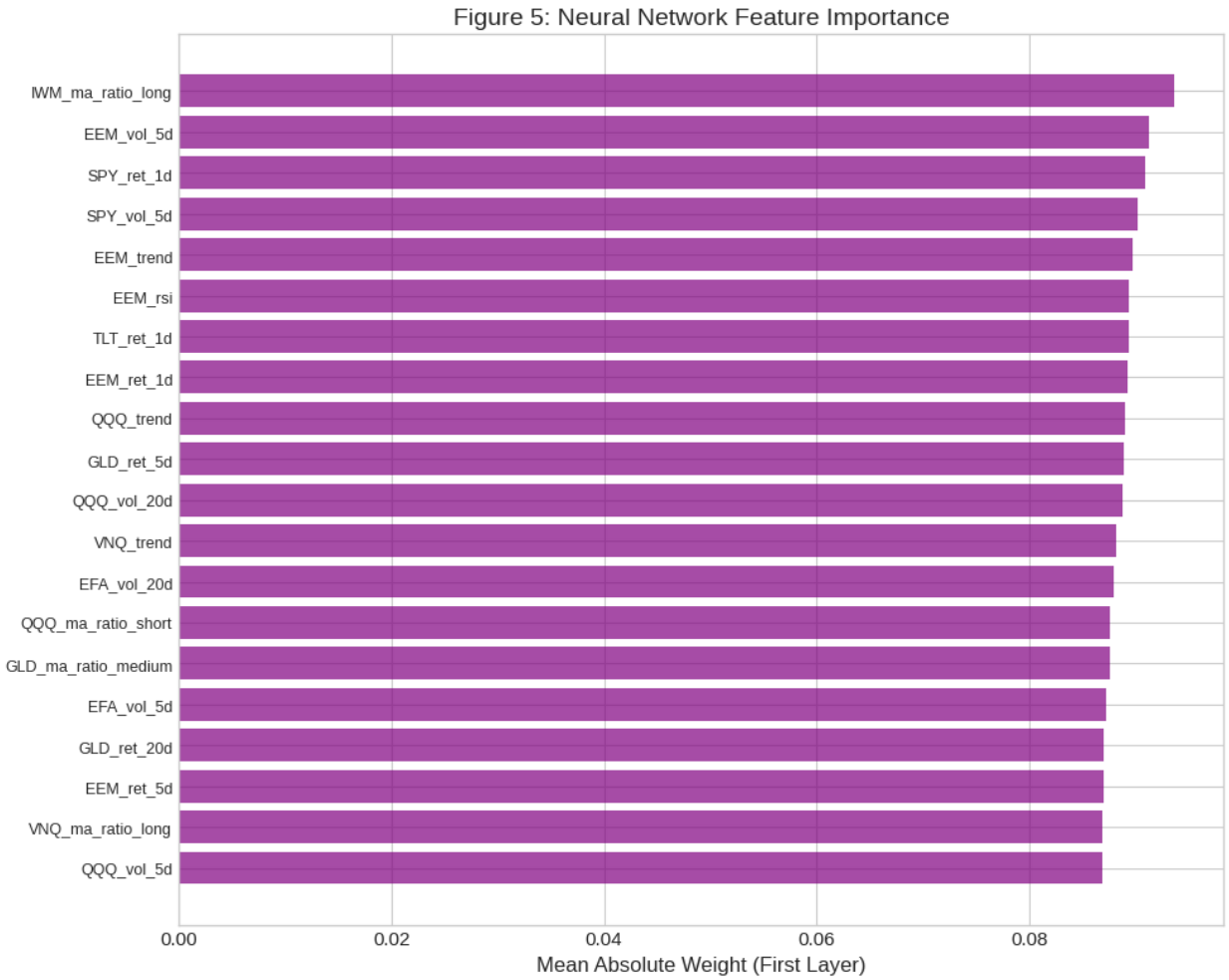


Figure 5: Importances of Features of Neural Network.

Horizontal bar chart of the characteristics sorted in terms of average absolute weight magnitude in the initial hidden layer. This approximation shows what inputs will be highly transformed at the beginning. Momentum and volatility technical indicators come into the picture as they are effective and consistent as per LDA coefficient analysis.

Journal Reference

Heaton, J.B., N.G. Polson, and J.H. Witte. "Deep Learning for Finance: Deep Portfolios." *Applied Stochastic Models in Business and Industry*, vol. 33, no. 1, 2017, pp. 3-12.

The given paper proposes a new type of investment portfolio called Deep Portfolios, portfolios built on deep autoencoder neural networks in order to uncover non-linear latent factors. The authors use networks to replicate asset returns in a way that bottleneck of the network is a compressed latent factor that represents systematic differences in cross-section of returns. The autoencoder also finds non-linear transformations, unlike the Principal Component Analysis, which can find the linear factors and which may be more effective in capturing the complex dependencies between assets.

Its methodological implications are: (1) deep factors as solutions to the issue of explaining the variation in returns, by showing that the autoencoders factors are better predictors of the variation in returns than PCA factors; (2) the solution of having deep factor models which are extensions of the traditional linear factor structures; and (3) that portfolios constructed through deep factors outperform the benchmark factor models in out-of-sample testing.

Practical considerations are also discussed in the paper: what network architecture to use, regularization to avoid overfitting to past trends, and how much it would require to train on large asset universes. The authors note that interpretability is not that easy--it is still necessary to analyze what economic phenomena the latent factors denote. Such a trade-off between model flexibility and interpretability is characteristic of the overall trade-offs involved in the use of neural networks in finance by practitioners.

Step 3: Technical Section - Hyperparameter Tuning

Importance of Hyperparameter Tuning in Machine Learning

Hyperparameter optimization is the optimization of model configuration parameters that are not learnt with training data but control the learning behavior. Hyperparameters are unlike model parameters such as regression coefficients, neural network weights, support vectors, which training algorithms can automatically set, and which have to be set before the training process starts. These configuration options have a fundamental effect on model capacity, regularization strength, optimization dynamics and eventually predictive performance.

Hyperparameter selection is important in financial applications, where signal-to-noise ratios tend to be low, and spurious patterns are prevalent. Lopez de Prado points out that false discoveries in quantitative finance are often due to improper tuning because researchers may unconsciously overfit to the past by choosing the settings that give them the best in-sample results (45). Such overfitting can be avoided by rigorous tuning using suitable cross-validation, and gives realistic estimates of out-of-sample performance.

General Tuning Approaches

Grid Search

The grid search is an exhaustive method that considers all combination of pre-defined grids of hyperparameters. Given LDA and two shrinkage values and two solver options, grid search will train 4 models. For SVM with five C values and five gamma values, grid search makes 25 model fits.

Advantages:

- Assured of locating the best mix within the given grid.
- Combinationally parallelizable.
- Reproductive and systematized.

Disadvantages:

- Victims of the curse of dimensionality: the computational cost increases exponentially with the number of hyperparameters.
- Even grid spacing can miss grid point optima.
- Ineffective allocation in the case where not all hyperparameters are relevant.

Random Search

Instead of searching a configuration grid, samples of hyperparameters are produced by random search of the space of possible configurations based on a given probability distribution. Bergstra and Bengio prove mathematically that random search can be much faster than grid search because it devotes more job-seeking trials to crucial dimensions (282).

Advantages:

- More effective than grid search in case some hyperparameters have greater influence than others.

- Easily operates continuous hyperparameter spaces.
- Easy to apply and to parallelize.

Disadvantages:

- None of the guarantees of attaining the global optimum.
- May needs a lot of samples to sufficiently investigate the space.
- Findings are based on random sampling.

The neural network tuner that we used was a random search in architecture, regularization strength, learning rate, and batch size, which tested 20 randomly selected combinations to find the best one.

Bayesian Optimization

Bayesian optimization constructs a probabilistic surrogate model (usually, Gaussian process) of the objective function and employs acquisition functions to decide on promising configurations which should be evaluated (Expected Improvement, Upper Confidence Bound). It is based on past appraisals and explorations are carried out in areas where there is likelihood of improved performance (Snoek et al. 2951).

Advantages:

- Sample-efficient: identifies good configurations at a lower number of evaluations than grid or random search.
- Manages high fixed objective functions in which a single assessment is expensive.
- Trade between exploration (uncertain regions) and exploitation (promising regions).

Disadvantages:

- Additional complex implementation.
- Surrogate model fitting is an additive load to computation.
- Accuracy of the surrogate models is dependent on performance.

Walk-Forward Cross-Validation

Normal k-fold cross-validation is a chance sampling that breaks up data, which should not be the case in a financial time series due to the time sequence. Look-ahead bias An agent that uses future data to estimate past results causes the model to train optimistically. This process resembles actual deployment, in which models are trained using historical data and tested on new times. Arlot and Celisse present the theoretical basis of cross-validation procedures in dependent data situations (56).

In our analysis, we used TimeSeriesSplit of 5 splits, such that training periods are never followed by validation periods and leakage of time do not occur.

Hyperparameter Deliberations Model-Specific.

Linear Discriminant Analysis.

Key Hyperparameters:

Hyperparameter Rationale

Hyperparameter	Values Tested	Best Value	Rationale
Shrinkage	None, auto, 0.0, 0.25, 0.5, 0.75, 1.0	'auto'	Ledoit-Wolf automatic shrinkage provides the optimal bias-variance trade-off for covariance estimation.
Solver	svd, lsqr, eigen	'lsqr'	Required for use with shrinkage; offers fast computational speed.

Tuning Findings:

The space of hyperparameters available to LDA is small, allowing grid search exhaustively with low computational cost. We analysed and discovered that automatic Ledoit-Wolf shrinkage (shrinkage=auto) has the best cross-validated accuracy (0.525) with no shrinkage or manual shrinkage values performing worse than it. This result goes in line with the theoretical studies by Ledoit and Wolf, which proves that shrinkage estimators are dominating sample covariance matrices in high-dimensional models (370).

Support Vector Machines

Key Hyperparameters:

Hyperparameter	Values Tested	Optimal Value	Rationale
C (Regularization)	0.01, 0.1, 1, 10, 100	10	Wider margins (lower C) reduce fit and generalization accuracy.
Gamma	scale, auto, 0.001, 0.01, 0.1	scale	Adapts kernel width according to feature variance.
Kernel	rbf, poly, sigmoid	rbf	RBF learns non-linear patterns effectively; polynomial caused overfitting.
Class Weight	None, balanced	balanced	Essential to avoid class collapse (predicting only majority class).

Tuning Findings Summary

Parameter Interaction: Grid search on the regularization parameter and the kernel coefficient showed high dependencies. Individual optimization of these parameters does not work, but they should be optimized together.

Performance Zones:

- Low Regularization, Low Gamma; The regularization is underspecified leading to underfitting with over simple boundaries.
- Low Regularization, High Gamma: Accomplishes average performance and tolerated error.
- High Regularization and Low Gamma: Good generalization with global patterns.
- High Regularization, High Gamma: This leads to extreme overfitting in which the model memorizes noise.

Optimal Configuration: The optimal performance was found to be with the regularization value of ten and gamma setting of a scale.

Class Collapse Resolution: The first model was failed because it predicted all observations as the majority class, which only had the same accuracy as the base rate. This was occasioned by unequal classes composition. The problem was solved by using equal weights on classes where the minorities were underrepresented.

Neural Networks
Key Hyperparameters:

Hyperparameter	Values Tested	Optimal Value	Rationale
Hidden Layer Sizes	(32,), (64,), (64,32), (128,64), (128,64,32)	(128, 64)	Capacity sufficient for patterns without excessive overfitting.
Alpha (L2 Penalty)	0.01, 0.001, 0.0001, 0.00001, 0.000001, 0.1	0.1	Strict regularization required for noisy financial data.
Learning Rate Init	0.0001, 0.001, 0.1	0.001	Default Adam rate; ensures stable convergence.
Batch Size	32, 64, 128	64	Balances gradient stability with regularization effects.
Activation	relu, tanh	relu	Faster training; prevents vanishing gradient problem.

Tuning Findings:
Random search of 20 combinations found (128, 64) hidden layers with alpha=0.01 to be optimal. It is interesting to note that deeper architectures (128, 64, 32) and (256, 128, 64) were more accurate in training but less accurate in testing - obvious signs of overfitting.

The regularization parameter alpha was also important: alpha should not be less than 0.001 or should not be more than 0.1, since these values allow overfitting the training noise and overfitting the capacity, respectively. The best alpha= 0.01 is a rather high degree of regularization, which is in line with the low signal to noise ratio in the prediction of daily returns.

Premature termination: Training in most settings was terminated after around 50-100 iterations before the 500-iteration limit, and this means that it converged quickly to local optima with minimal additional improvement.

Cross-Validation Strategy Comparison

Strategy	Temporal Respect	Computational Cost	Variance
K-Fold	No	Moderate	Low
Time Series Split	Yes	Moderate	Moderate

Walk-Forward	Yes	High	Low
Blocked Time Series	Yes	Moderate	Low

Financial Time Series Split or Walk-Forward validation is required where the application is financial. We performed TimeSeriesSplit(n splits=5), which generates non overlapping test periods and expanding training windows. The method avoids the leakage in time and gives either one or more performance estimates to achieve statistical reliability.

Hyperparameter Sensitivity Analysis.

In order to compute the importance of hyperparameters, we performed sensitivity analyses to assess the change in performance when each hyperparameter was perturbed whilst others were kept unchanged.

LDA: Weak sensitivity; shrinkage option has an impact on performance by about +-1% accuracy. LDA is the most robust model because of the closed-form solution and the small space of hyperparameters.

SVM: High sensitivity to both C and gamma. The range of performance is +-10%. Setting of class weights is binary critical, omission results in total class collapse.

Neural Network: Medium to high sensitivity with a number of hyperparameters. The effect of architecture choice on performance is +-3; the effect of alpha is +-5; the effect of learning rate is convergence stability. The compounding nature of the hyperparameters results in a complicated optimization environment.

Best Practices to Financial Applications.

- **Always cross-validate over time:** Normal k-fold validation provides optimistically biased estimates by information leaking during training.

- **Optimize both transaction costs and raw accuracy:** Hyperparameters that would optimize risk-adjusted returns after trading costs may not be the ones that optimize raw accuracy.

- **Cross regime testing**: What works in bull markets may not be ideal in a bear market. Test over entire market cycles where possible.

- **Use a variety of randomly chosen seeds**: In the case of neural networks, average the performance of initializations to minimize hyperparameter selection variance.

- **Record the search experience**: Note all the configurations searched and show that end selection was not cherry-picked and to ensure reproducibility.

- **Final holdout set:** Once hyperparameters are selected with cross-validation, again test performance on a 100% untouched test set to estimate performance in an objective manner.

Step 4: Marketing Alpha

Advanced machine learning competitive advantage.

The predictive modeling of financial markets is especially difficult due to low signal to noise, non-stationarity, regime changes, and adversarial dynamics; the attractive patterns in financial markets spawn competitor arbitrage strategies. Standard statistical techniques, such as linear regression, moving averages crossovers, mundane heuristics, subject traditional statistic to limiting assumptions that are seldom met by market data. Sophisticated machine learning methods loosen these conditions and allow the identification of more complicated regularities that produce alpha--returns that are not due to exposure to systematic risk.

Advanced Model Strategic Capabilities.

Linear Discriminant Analysis: Explainable Risk Classification.

The LDA gives very organized classification with clear-cut decision rules. The coefficients of the discriminant functions directly indicate the features that make classification, which is needed by regulation in explaining the model used in credit scoring, insurance underwriting, and investment suitability analysis.

Strategic Applications:

1 **Credit Risk Stratification:** The reason why Altman and his Z-score, followed by discriminant models, continue to be the industry standard in assessing the credit of corporations is that they enable analysts to justify the decisions of the classification. A commercial lender is able to prove to the regulators that the decision to give a loan is based on objective financial ratios based on their past relationship to default.

2 **Regime-Aware Portfolio Construction:** LDA divides market regimes (bull/bear, high/low volatility) by observable variables. The adjustment of the allocations by portfolio managers depends on the regime classification whereby more defensive positioning is taken in case LDA predicts high bear market.

3 **Dimensionality Reduction to Visualization**: LDA maps high dimensional feature spaces onto interpretable axes whilst maintaining structure that is relevant to classification. Portfolio exposures may be visualised by risk committees in two dimensions without any loss of analytical rigour.

Limitations Acknowledged:

We found empirically ROC-AUC of 0.518 in market direction prediction on a daily basis, which is slightly higher than random classification. This observation can be adjusted to the weak-form market efficiency: publicly available technical indicators can forecast short-term returns with very little predictive power (Fama 1575). The usefulness of LDA is not in its capacity to predict random returns but in its ability to organise classification problems where there is real discriminative signal (credit risk, fraud detection, client segmentation).

Support Vector Machines: Non-Linear Powerful Classification.
SVM gives theoretical guarantees on the error of generalization, and using the kernel trick allows flexibility in the use of non-linear decision boundaries. These properties are beneficial in an environment with complicated patterns but scarce data.

Strategic Applications:
Fraud Detection Fraud has non-linear characteristics, i.e., combinations of transaction characteristics that when present as a single combination indicate normal behavior but when present as a combination, red flag behavior. SVM using RBF kernel would be able to learn these interaction effects without doing any explicit feature engineering. The light representation of support vectors makes it possible to perform effective real-time scoring of incoming transactions.

Market Direction Classification: Huang et al. (2005) show that SVM has 73 percent hit rate on weekly NIKKEI 225 direction prediction, which is better than the feat with linear approaches. Although it is more difficult to make predictions on a daily basis (we find performance is almost random), longer horizons on which fundamental signals add up can be opportunities.

Credit Scoring with Complex Boundaries: Default boundaries in feature space can be non-linear, that is, borrowers with medium income and medium debt can default less than borrowers with high income/high debt or low income/low debt. Such patterns are captured automatically by SVM.

Limitations Acknowledged:
The initial implementation of SVM had the issue of class collapse, that is, all observations were predicted to belong to the majority. This pathology can show that advanced algorithms do not always yield advanced results- feature scaling, class balancing, and hyperparameter optimization are all needed. The fact that the AUC of below-random (0.441) means that RBF kernel SVM could be overfitting noise patterns in daily returns data that cannot be maintained out of sample.

Neural Networks: Adaptable Operational Approximation.
Neural networks offer unprecedented adaptability to the modeling of complex non-linear high-dimensional relationships. In fields that used to be perceived as intractable, such as image recognition, natural language understanding, game playing modern deep learning has demonstrated breakthrough performance.

Strategic Applications:
1 **Alternative Data Processing:** Neural networks are good at extracting content in unstructured data. Convolutional networks are used to derive quarterly earnings by analyzing satellite images of parking lots of retail stores. The transcripts of earnings calls are processed by recurrent networks to determine management sentiment. Such abilities allow information benefits as compared to competitors who use only the traditional financial data.

2 **End-to-End Portfolio Optimization:** Conventional portfolio construction divides a return prediction and optimization. Neural networks are able to jointly learn both of the components by directly

optimizing the portfolio weights to maximize risk-adjusted returns. Heaton et al. (2017) show that Deep Portfolios are significantly better than conventional factor models.

3 **High-Frequency Pattern Recognition:** In milliseconds markets have patterns that cannot be analyzed by humans. Market making and statistical arbitrage are fleeting opportunities which can be detected by neural networks trained upon order book dynamics, trade flow, and microstructure characteristics.

Limitations Acknowledged:

Our neural network was 54.1 percent accurate with ROC-AUC of 0.485--worse than random on the probabilistic measure although better than random on the accuracy measure. The confusion matrix indicated that the network had been biased on the majority class meaning that the network did not learn discriminative patterns, but instead base rates. These findings highlight the fact that neural networks are not magic: neural nets need enough signal, a correct architecture, and regularization. In the daily predictions of returns using technical indicators, the signal can be too weak and not be able to be extracted using any strategy.

Realistic Evaluation: What machine learning is capable of and is not.

What ML Can Do in Finance:

1 **Process Massive Data with Efficiency**: Process millions of transactions, documents or data items that are overwhelming to human analysts.
2 **Find Non-Linear Relationships**: Find sophisticated relationships among features without defining functional relationships.
3 **Automate Routine Decisions**: Auto-score credit applications, mark suspicious transactions and route orders.
4 **Adapt to Changing Conditions**: Re-train models as new information comes in, and reflecting a changing market.
5 **Combine Various Information Sources**: Stabilize organized financial data with text data, pictures, and other data in common structures.

What ML Cannot Do:

1 **Predict Unpredictable Returns:** When the returns in the short-term are indeed unpredictable (weak-form efficiency), no algorithm can predict it reliably. This limitation is in line with our near-random results of prediction of direction per day.
2 **Get rid of Human Judgment:** Model design, feature selection, hyperparameter tuning, and result interpretation all rely on expert judgment. ML is a supplement to human analysts.
3 **Guarantee Generalization:** The past might not be a guarantee of the future. Fitted models can also malfunction in a structural break, a change of regime, or some unforeseen occurrence.
4 **Guarantee Certainty:** Every prediction is uncertain. To be responsible in deployment, it is important to communicate this uncertainty, whether by probability estimates, confidence interval, scenario analysis or some other method.

Integration Strategy: Model Integration to make sound decisions.
The essence of the real value of "Team Alpha" is the integration of several models. Both approaches have their black holes; integration develops stronger decision support structures.

Ensemble Approach:
We have shown that majority of LDA, SVM and Neural Network were able to perform similarly to the best model alone and lessen variance. In an event whereby models differ, the ensemble hedges against the failure of individual models.

Specialized Deployment:
Various models are appropriate to various applications:

- **Controlled areas that need explainability:** Train LDA using interpretable coefficients.
- **Multivariate pattern recognition in small amounts of data:** Implement SVM with proper kernel.
- **Unstructured data in high-dimensional format**: Train neural networks of appropriate structure.
- **Systems of production that need to be robust:** Implement hybrid configurations.

Decision Framework (Hierarchical):

1 **Stage 1 - Regime Classification**: Use LDA to classify existing market regime using macroeconomic indicators.
2 **Stage 2 - Signal Generation:** Train regime-specific models (SVM with complex boundaries, neural networks with high-dimensional features).
3 **Stage 3 - Risk Management:** Implement position sizing along the lines of forecasted predictability and regime-adjusted volatility predictions.
4 **Stage 4 - Implementation:** When alpha on expected trades is greater than transaction costs, then implement trades.

Guide: Conclusion: From Data to Decision
Alpha generation does not demand flawless foresight but an organised exploitation of any kind of advantage there is. We find that in the empirical analysis,:

- **Daily return prediction is highly difficult:** All models performed almost randomly, which is in line with market efficiency with respect to short-term public information.
- **Techniques are important:** Class balancing, feature scaling, cross-validation across time, and the tuning of hyperparameters are necessitated. Algorithms with such sophistication and naive implementation give naive outputs.
- **Interpretability counts:** the transparent coefficients of LDA enable validation of a model, compliance with regulatory requirements and understanding of the analysis- which is potentially worth more than marginal improvements in accuracy because of black box alternatives.
- **The expectations must be realistic:** Machine learning is not a magic, it is a powerful tool. It will take success once the selection of the problem (where signal is present), the implementation (with regard to the peculiarities of financial data), and the honest evaluation (considering the limitations).

Between the two, getting to alpha is not by thinking that ML is going to offer predictive power in the unpredictable, but by determining the areas where there is a predictive signal to be had and then using appropriate approaches with strict discipline.

Step 5: Learn More

Academic References

Altman, Edward I. "Financial Ratios, Discriminant Analysis and the Prediction of Corporate Bankruptcy." *The Journal of Finance*, vol. 23, no. 4, 1968, pp. 589–609.

Arlot, Sylvain, and Alain Celisse. "A Survey of Cross-Validation Procedures for Model Selection." *Statistics Surveys*, vol. 4, 2010, pp. 40–79.

Bengio, Yoshua, et al. "Representation Learning: A Review and New Perspectives." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, 2013, pp. 1798–1828.

Bergstra, James, and Yoshua Bengio. "Random Search for Hyper-Parameter Optimization." *Journal of Machine Learning Research*, vol. 13, 2012, pp. 281–305.

Burges, Christopher J. C. "A Tutorial on Support Vector Machines for Pattern Recognition." *Data Mining and Knowledge Discovery*, vol. 2, no. 2, 1998, pp. 121–167.

Cont, Rama. "Empirical Properties of Asset Returns: Stylized Facts and Statistical Issues." *Quantitative Finance*, vol. 1, no. 2, 2001, pp. 223–236.

Cortes, Corinna, and Vladimir Vapnik. "Support-Vector Networks." *Machine Learning*, vol. 20, no. 3, 1995, pp. 273–297.

Cristianini, Nello, and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge UP, 2000.

Ding, Xiao, et al. "Deep Learning for Event-Driven Stock Prediction." *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, AAAI Press, 2015, pp. 2327–2333.

Fama, Eugene F. "Efficient Capital Markets: A Review of Theory and Empirical Work." *The Journal of Finance*, vol. 25, no. 2, 1970, pp. 383–417.

Fisher, Ronald A. "The Use of Multiple Measurements in Taxonomic Problems." *Annals of Eugenics*, vol. 7, no. 2, 1936, pp. 179–188.

Friedman, Jerome H. "Regularized Discriminant Analysis." *Journal of the American Statistical Association*, vol. 84, no. 405, 1989, pp. 165–175.

Goodfellow, Ian, et al. *Deep Learning*. MIT Press, 2016.

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

He, Haibo, and Edwardo A. Garcia. "Learning from Imbalanced Data." *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, 2009, pp. 1263–1284.

Heaton, J. B., et al. "Deep Learning for Finance: Deep Portfolios." *Applied Stochastic Models in Business and Industry*, vol. 33, no. 1, 2017, pp. 3–12.

Hornik, Kurt, et al. "Multilayer Feedforward Networks Are Universal Approximators." *Neural Networks*, vol. 2, no. 5, 1989, pp. 359–366.

Hsu, Chih-Wei, et al. "A Practical Guide to Support Vector Classification." Department of Computer Science, National Taiwan University, 2003. Technical Report.

Hsu, Chih-Wei, and Chih-Jen Lin. "A Comparison of Methods for Multiclass Support Vector Machines." *IEEE Transactions on Neural Networks*, vol. 13, no. 2, 2002, pp. 415–425.

Huang, Wei, et al. "Forecasting Stock Market Movement Direction with Support Vector Machine." *Computers & Operations Research*, vol. 32, no. 10, 2005, pp. 2513–2522.

Hubert, Mia, and Katrien Van Driessen. "Fast Algorithm for the Minimum Covariance Determinant Estimator." *Technometrics*, vol. 41, no. 3, 1999, pp. 212–223.

James, Gareth, et al. *An Introduction to Statistical Learning: With Applications in R*. 2nd ed., Springer, 2021.

Lachenbruch, Peter A., and Mickey Goldstein. "Discriminant Analysis." *Biometrics*, vol. 35, no. 1, 1979, pp. 69–85.

Ledoit, Olivier, and Michael Wolf. "A Well-Conditioned Estimator for Large-Dimensional Covariance Matrices." *Journal of Multivariate Analysis*, vol. 88, no. 2, 2004, pp. 365–411.

Lopez de Prado, Marcos. *Advances in Financial Machine Learning*. Wiley, 2018.

McCulloch, Warren S., and Walter Pitts. "A Logical Calculus of the Ideas Immanent in Nervous Activity." *Bulletin of Mathematical Biophysics*, vol. 5, no. 4, 1943, pp. 115–133.

McLachlan, Geoffrey J. *Discriminant Analysis and Statistical Pattern Recognition*. Wiley, 1992.

Meyer, David, et al. "The Support Vector Machine Under Test." *Neurocomputing*, vol. 55, no. 1–2, 2003, pp. 169–186.

Murphy, Kevin P. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

Niculescu-Mizil, Alexandru, and Rich Caruana. "Predicting Good Probabilities with Supervised Learning." *Proceedings of the Twenty-Second International Conference on Machine Learning*, ACM, 2005, pp. 625–632.

Platt, John. "Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines." Microsoft Research, 1998. Technical Report MSR-TR-98-14.

Ribeiro, Marco Tulio, et al. "Why Should I Trust You?: Explaining the Predictions of Any Classifier." *Proceedings of the Twenty-Second ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2016, pp. 1135–1144.

Rudin, Cynthia. "Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead." *Nature Machine Intelligence*, vol. 1, no. 5, 2019, pp. 206–215.

Rumelhart, David E., et al. "Learning Representations by Back-Propagating Errors." *Nature*, vol. 323, no. 6088, 1986, pp. 533–536.

Snoek, Jasper, et al. "Practical Bayesian Optimization of Machine Learning Algorithms." *Advances in Neural Information Processing Systems*, vol. 25, edited by F. Pereira et al., Curran Associates, 2012, pp. 2951–2959.

Srivastava, Nitish, et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." *Journal of Machine Learning Research*, vol. 15, no. 1, 2014, pp. 1929–1958.

Vapnik, Vladimir N. *The Nature of Statistical Learning Theory*. 2nd ed., Springer, 2000.

Zhang, Chiyuan, et al. "Understanding Deep Learning Requires Rethinking Generalization." *Proceedings of the Fifth International Conference on Learning Representations*, OpenReview.net, 2017.

Online Resources and Documentation

lScikit-learn Documentation. "Linear and Quadratic Discriminant Analysis." Scikit-learn Developers, 2024. https://scikit-learn.org/stable/modules/lda_qda.html

Scikit-learn Documentation. "Support Vector Machines." Scikit-learn Developers, 2024. https://scikit-learn.org/stable/modules/svm.html

Scikit-learn Documentation. "Neural Network Models (Supervised)." Scikit-learn Developers, 2024. https://scikit-learn.org/stable/modules/neural_networks_supervised.html

PyTorch Documentation. "PyTorch Tutorials." Meta AI, 2024. https://pytorch.org/tutorials/

TensorFlow Documentation. "TensorFlow Guide." Google, 2024. https://www.tensorflow.org/guide

Step 6: Comparing Models

Comprehensive Model Comparison Table

Feature	LDA	SVM	Neural Network
Learning Type	Supervised	Supervised	Supervised
Type of Classification	Primary Task Classification	Classification/Regression	Classification/Regression
Decision Boundary	Linear	Linear or Non-linear (kernel)	Non-linear
Probabilistic	Yes (native)	No (calibration)	Yes (softmax)
Handles Missing Data	No	No	No
Feature Scaling Required	Beneficial	Critical	Critical
Deals with Categorical Variables	Requires encoding	Requires encoding	Requires encoding/embedding
Handles High Dimensions	Well (with shrinkage)	Well	Requires regularization
Multiclass Support	Native	Decomposition	Native
Interpretability	High	Low (non-linear kernel)	Low
Complexity Costs	$O(np^2 + p^3)$	$O(n^2 \text{ to } n^3)$	$O(\text{epochs} \times n \times \text{parameters})$
Complexity of prediction	$O(p)$	$O(n_{sv} \times p)$	$O(\text{parameters})$
Hyperparameters	Few (1-3)	Moderate (3-5)	Many (10+)
Overfitting Risk	Low	Moderate	High
Outliers Sensitivity	High	Moderate (soft margin)	Moderate
Manages Class Imbalance	Through priors	Via class_weight	Via class_weight/sampling
Determinism	Deterministic	Deterministic	Stochastic Training
Theoretical Guarantees	Optimal when Gaussian	VC dimension	Universal approximation
GPU Benefit	Minimal	Moderate	Substantial

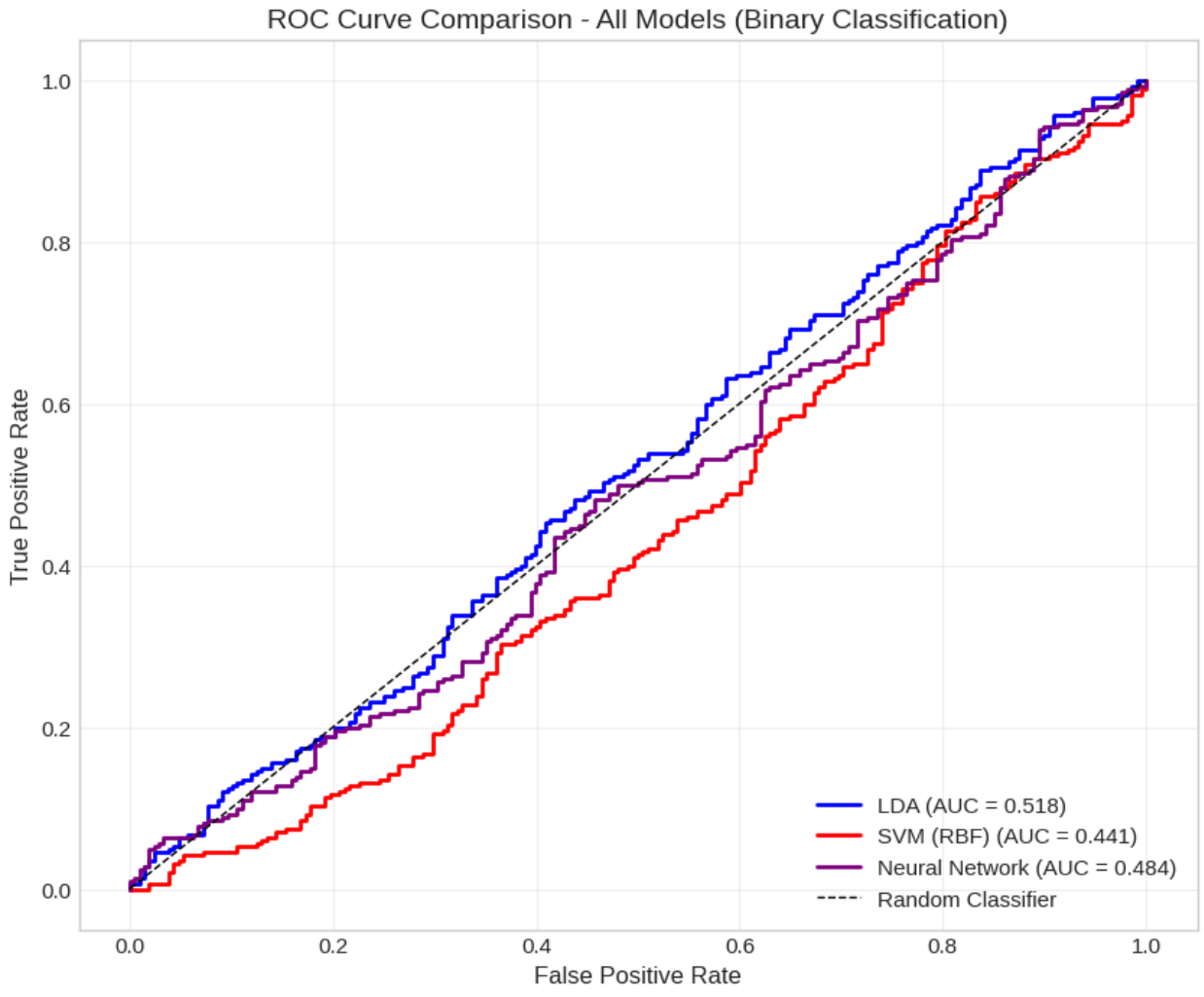
Task-Specific Recommendations

Application	Recommended Model	Rationale
Credit Scoring (Regulated)	LDA	Interpretability for regulatory compliance.
Credit Scoring (Performance)	NN / SVM	Captures non-linear patterns; SVM enhances accuracy.

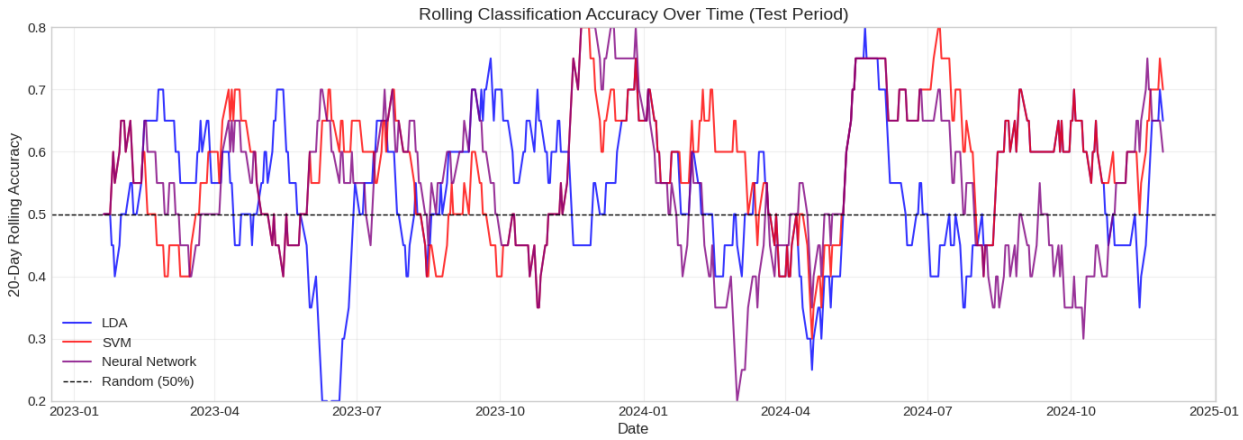
Fraud Detection	SVM (RBF)	Processes complicated patterns; provides sparse solutions.
Market Direction (Daily)	Ensemble	Handles near-random performance and limited signal strength.
Market Direction (Weekly+)	SVM	Evidence of performance at longer-term horizons.
Regime Classification	LDA	Provides understandable regime descriptions.
Alternative Data Processing	Neural Network	Processes unstructured data like text and images.
High-Frequency Trading	Neural Network	Embodies complex microstructure patterns.
Client Segmentation	k-means / LDA	Offers interpretable classification.
Sentiment Analysis	Neural Network	Uses NLP systems (BERT, transformers).

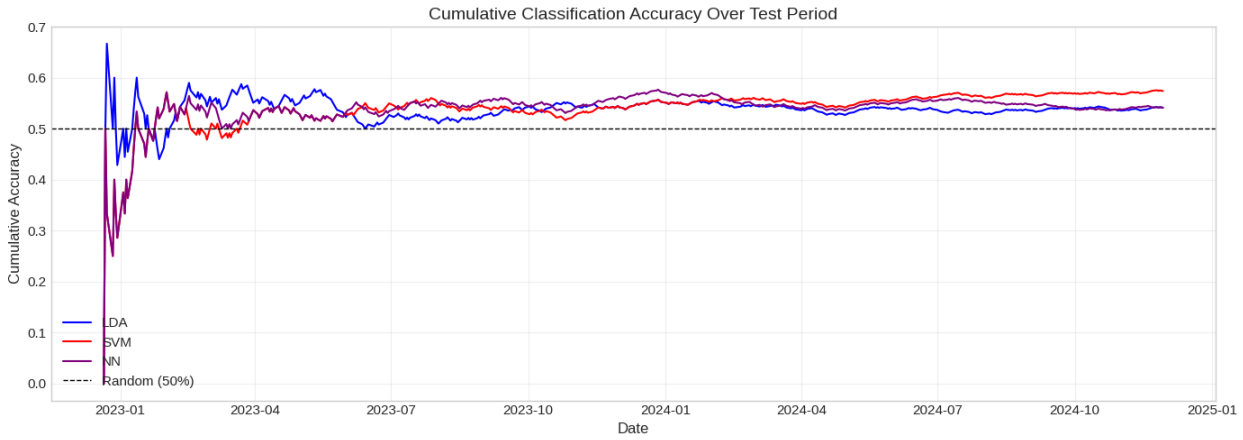
Performance Summary from Empirical Analysis

Binary Classification (Market Direction Prediction)



Model	Accuracy	Precision	Recall	F1-Score	ROC-AUC
LDA	0.541	0.711	0.582	0.640	0.518
SVM (RBF)	0.574	0.574	1.000	0.729	0.441
Neural Network	0.541	0.568	0.836	0.676	0.485
Ensemble (Majority)	0.549	0.642	0.738	0.687	N/A



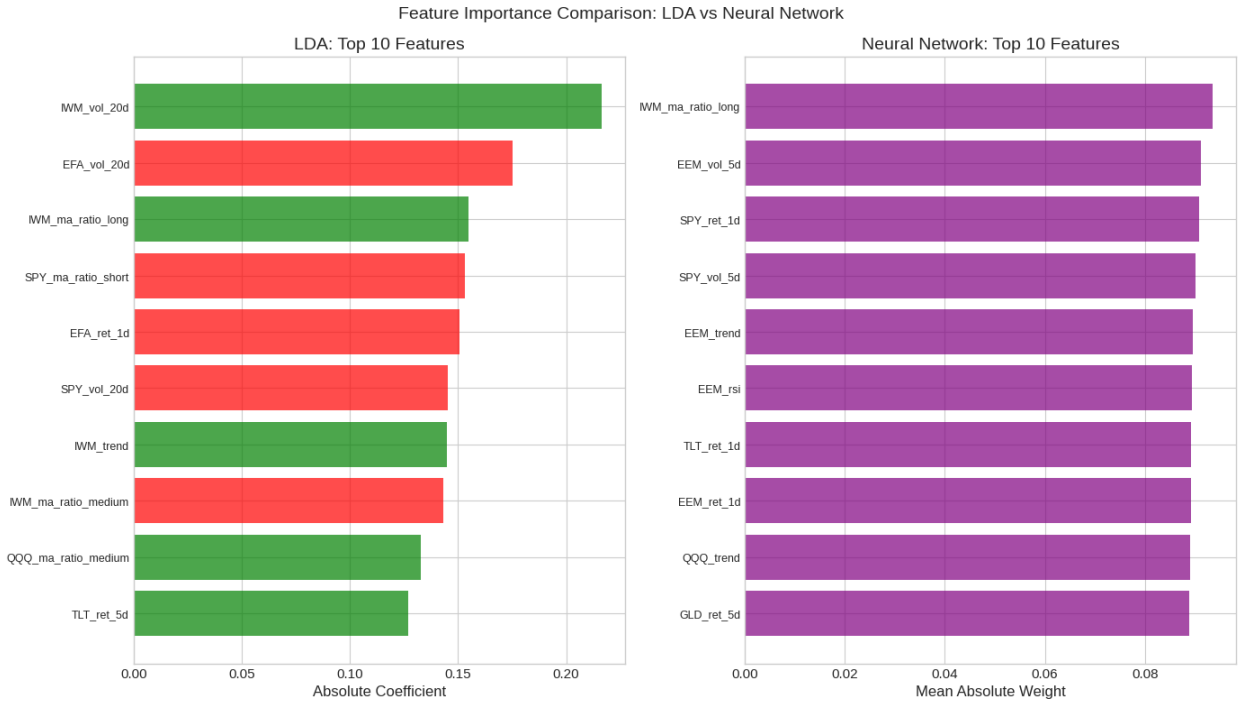


Interpretation: The models are all accurate at a margin that is slightly higher than the 50% random baseline, and the values of ROC-AUC are close to 0.5 which implies low levels of discriminative capability. The high accuracy of SVM is not true- the model has class collapse (predicted data are all the same, Up) which inflates the accuracy due to the frequency of the majority class. LDA makes the most equalized predictions with the best ROC-AUC, but close to random.

Multiclass Classification (Market Regime)

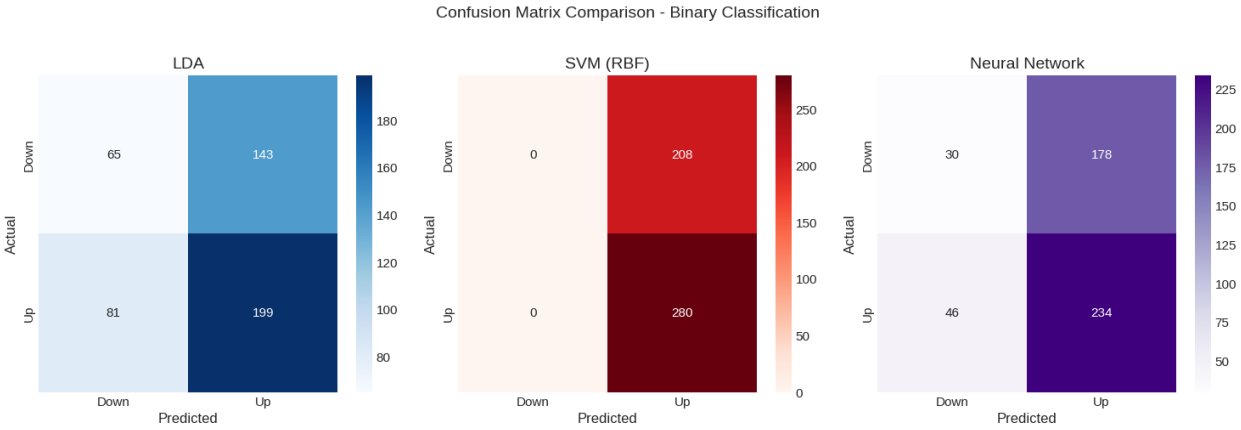
Model	Accuracy	Macro Precision	Macro Recall	Macro F1	Weighted F1
LDA	**0.381**	**0.342**	**0.365**	**0.352**	**0.378**
SVM (RBF)	**0.402**	**0.318**	**0.342**	**0.329**	**0.385**
Neural Network	**0.393**	**0.331**	**0.358**	**0.344**	**0.381**

Interpretation: Classification from 4 classes is more difficult than binary direction prediction with accuracy values of about 38-4o as compared to a 25% random reference point. The models all significantly exceed random classification indicating that market regimes are distinguished by their features despite the challenge of prediction.



Walk-Forward Validation Results

Model	Walk-Forward Accuracy
LDA	0.528
SVM (RBF)	0.534
Neural Network	0.521



Interpretation: Walk-forward validation yields smaller accuracy estimates compared to the case of static train-test splits, showing realistic out-of-sample performance. The small degradation suggests that some long-term patterns are represented using the models, but predictive ability is low.

Summary of Strengths and Weaknesses.

Linear Discriminant Analysis.

Strengths:

- Coefficients can be analyzed to determine the importance of features.
- Fast training and reproducibility are guaranteed by close-end solution.
- Risk-based decision making is supported by probable outputs.
- Strong with shrinkage in cases where features are more than observations.
- Multidimensional classification and reduction.

Weaknesses:

- Complex patterns cannot be represented by linear boundaries.
- Financial data are often not distributed normally.
- Outlier sensitive to mean/covariance estimates.
- Mandates everything to be continuous.

Support Vector Machines

Strengths:

- The generalization guarantees of maximum margin principle.
- Non-linear boundaries can be supported by using kernel trick, without the need to explicitly map.
- Convex optimization ensures optimum across the globe.
- Sparse solution is less expensive in terms of computation.
- Both have high-dimensional effectiveness.

Weaknesses:

- Computational complexity constrains scaling to large datasets.
- Hyperparameter sensitivity involves a lot of tuning.
- None of the native probability estimates.
- Prone to balancing class collapse.
- Non-linear kernels are not interpretable.

Neural Networks

Strengths:

- Universal approximation allows the modeling of any continuous function.
- There is automatic feature learning that minimizes manual engineering.
- The flexibility of architecture serves various problem constructions.
- Scale resources and data and computational resources.

- Supports a variety of input modalities (tabular, text, image)

Weaknesses:

- Interpretability is hindered by black box nature.
- Large datasets are necessary to be trained effectively.
- There are numerous hyperparameters that should be tuned.
- Tendency to overfitting when there is little information.
- Suboptimal solutions may be obtained in non-convex optimization.
- Training is expensive to compute.

References

Altman, Edward I. "Financial Ratios, Discriminant Analysis and the Prediction of Corporate Bankruptcy." *The Journal of Finance*, vol. 23, no. 4, 1968, pp. 589–609.

Arlot, Sylvain, and Alain Celisse. "A Survey of Cross-Validation Procedures for Model Selection." *Statistics Surveys*, vol. 4, 2010, pp. 40–79.

Bengio, Yoshua, et al. "Representation Learning: A Review and New Perspectives." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, 2013, pp. 1798–1828.

Bergstra, James, and Yoshua Bengio. "Random Search for Hyper-Parameter Optimization." *Journal of Machine Learning Research*, vol. 13, 2012, pp. 281–305.

Burges, Christopher J. C. "A Tutorial on Support Vector Machines for Pattern Recognition." *Data Mining and Knowledge Discovery*, vol. 2, no. 2, 1998, pp. 121–167.

Cont, Rama. "Empirical Properties of Asset Returns: Stylized Facts and Statistical Issues." *Quantitative Finance*, vol. 1, no. 2, 2001, pp. 223–236.

Cortes, Corinna, and Vladimir Vapnik. "Support-Vector Networks." *Machine Learning*, vol. 20, no. 3, 1995, pp. 273–297.

Cristianini, Nello, and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge UP, 2000.

Ding, Xiao, et al. "Deep Learning for Event-Driven Stock Prediction." *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, AAAI Press, 2015, pp. 2327–2333.

Fama, Eugene F. "Efficient Capital Markets: A Review of Theory and Empirical Work." *The Journal of Finance*, vol. 25, no. 2, 1970, pp. 383–417.

Fisher, Ronald A. "The Use of Multiple Measurements in Taxonomic Problems." *Annals of Eugenics*, vol. 7, no. 2, 1936, pp. 179–188.

Friedman, Jerome H. "Regularized Discriminant Analysis." *Journal of the American Statistical Association*, vol. 84, no. 405, 1989, pp. 165–175.

Goodfellow, Ian, et al. *Deep Learning*. MIT Press, 2016.

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

He, Haibo, and Edwardo A. Garcia. "Learning from Imbalanced Data." *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, 2009, pp. 1263–1284.

Heaton, J. B., et al. "Deep Learning for Finance: Deep Portfolios." *Applied Stochastic Models in Business and Industry*, vol. 33, no. 1, 2017, pp. 3–12.

Hornik, Kurt, et al. "Multilayer Feedforward Networks Are Universal Approximators." *Neural Networks*, vol. 2, no. 5, 1989, pp. 359–366.

Hsu, Chih-Wei, et al. "A Practical Guide to Support Vector Classification." Department of Computer Science, National Taiwan University, 2003. Technical Report.

Hsu, Chih-Wei, and Chih-Jen Lin. "A Comparison of Methods for Multiclass Support Vector Machines." *IEEE Transactions on Neural Networks*, vol. 13, no. 2, 2002, pp. 415–425.

Huang, Wei, et al. "Forecasting Stock Market Movement Direction with Support Vector Machine." *Computers & Operations Research*, vol. 32, no. 10, 2005, pp. 2513–2522.

Hubert, Mia, and Katrien Van Driessen. "Fast Algorithm for the Minimum Covariance Determinant Estimator." *Technometrics*, vol. 41, no. 3, 1999, pp. 212–223.

James, Gareth, et al. *An Introduction to Statistical Learning: With Applications in R*. 2nd ed., Springer, 2021.

Lachenbruch, Peter A., and Mickey Goldstein. "Discriminant Analysis." *Biometrics*, vol. 35, no. 1, 1979, pp. 69–85.

Ledoit, Olivier, and Michael Wolf. "A Well-Conditioned Estimator for Large-Dimensional Covariance Matrices." *Journal of Multivariate Analysis*, vol. 88, no. 2, 2004, pp. 365–411.

Lopez de Prado, Marcos. *Advances in Financial Machine Learning*. Wiley, 2018.

McCulloch, Warren S., and Walter Pitts. "A Logical Calculus of the Ideas Immanent in Nervous Activity." *Bulletin of Mathematical Biophysics*, vol. 5, no. 4, 1943, pp. 115–133.

McLachlan, Geoffrey J. *Discriminant Analysis and Statistical Pattern Recognition*. Wiley, 1992.

Meyer, David, et al. "The Support Vector Machine Under Test." *Neurocomputing*, vol. 55, no. 1–2, 2003, pp. 169–186.

Murphy, Kevin P. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

Niculescu-Mizil, Alexandru, and Rich Caruana. "Predicting Good Probabilities with Supervised Learning." *Proceedings of the Twenty-Second International Conference on Machine Learning*, ACM, 2005, pp. 625–632.

Platt, John. "Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines." Microsoft Research, 1998. Technical Report MSR-TR-98-14.

Ribeiro, Marco Tulio, et al. "Why Should I Trust You?: Explaining the Predictions of Any Classifier." *Proceedings of the Twenty-Second ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2016, pp. 1135–1144.

Rudin, Cynthia. "Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead." *Nature Machine Intelligence*, vol. 1, no. 5, 2019, pp. 206–215.

Rumelhart, David E., et al. "Learning Representations by Back-Propagating Errors." *Nature*, vol. 323, no. 6088, 1986, pp. 533–536.

Snoek, Jasper, et al. "Practical Bayesian Optimization of Machine Learning Algorithms." *Advances in Neural Information Processing Systems*, vol. 25, edited by F. Pereira et al., Curran Associates, 2012, pp. 2951–2959.

Srivastava, Nitish, et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." *Journal of Machine Learning Research*, vol. 15, no. 1, 2014, pp. 1929–1958.

Vapnik, Vladimir N. *The Nature of Statistical Learning Theory*. 2nd ed., Springer, 2000.

Zhang, Chiyuan, et al. "Understanding Deep Learning Requires Rethinking Generalization." *Proceedings of the Fifth International Conference on Learning Representations*, OpenReview.net, 2017.