GROUP WORK PROJECT # 02 MScFE 632: Machine Learning in Finance
GROUP NUMBER: ANONYMIZED FOR PRIVACY

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

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

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

Use the box below to explain any attempts to reach out to a non-contributing member. Type (N/A) if all members contributed.
Note: You may be required to provide proof of your outreach to non-contributing members upon request.

STEPs 1, 2 3

I. LINEAR DISCRIMINANT ANALYSIS (LDA)

Basics

A. Definition

Linear Discriminant Analysis (LDA) is a supervised classification technique used for discriminating between at least two classes by mapping the data onto a reduced dimensionality space. The fundamental principle of LDA is that it identifies a linear combination of features that optimally discriminates the classes by maximizing the between-class scatter and minimizing the within-class scatter. This makes LDA especially valuable in applications where not only classification but also getting insights into the data's organization is at stake.

Practically, LDA can be considered as a technique that intersects with a single straight line (or hyperplane for high dimensions) between multiple categories of a set of data.For example, for credit risk analysis, LDA can be used for discriminating high-risk and low-risk borrowers given financial features such as income level, debt-to-income ratio, credit history, and amount of loan. By mapping these features onto a linear discriminant axis, the model picks up distinct boundaries that enable financial institutions to make good lending decisions with reduced default risk.

B. Classification and assumptions

LDA is a linear family of classifiers, that is, it makes the assumption that the decision boundary between classes is linear. But it does not model the probability of class membership directly, as does logistic regression, rather it models the distribution of the predictors distributions for each class and then applies Bayes' theorem in order to approximate the probability of class membership.

To operate optimally, LDA works on the assumptions that:

- ❖ The data distributions for each class are normal distributions.
- ❖ All classes have identical covariance matrices.
- ❖ The observations are independent and features linearly correlate with the class label.

These assumptions enable LDA to obtain a closed-form solution for the rule of classification so that it's highly computationally efficient and interpretable.

C. Model Formula and Equation

The mathematical basis for LDA is the discriminant function, which serves for the classification of a new observation in one of the pre-established classes.

For a specific class k, the discriminant function is defined as:

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

Where:

- x is the input feature vector e.g Financial indicators
- μ_k is the mean vector of class k
- Σ is the shared covariance matrix across all classes,
- π_k is the prior probability of class k

The model calculates this score for each class and labels the observation with the maximum score for that class. This method ensures that classification occurs both on distance from the class mean and on the distribution of the distributions.

Step by Step Application

- **Mean Vector μ_k:** For each k class, calculate the mean value of all feature vectors that fall within this class. This is the class's central point in feature space.
- **Covariance Matrix Σ:** Estimate a unique covariance matrix from all training data, assuming that the distribution of the data is roughly equal across classes.
- **Prior Probability π_k :** Find the percentage of training samples which are of class k. This gives an indication of the probability that a new observation will be of that class without having a look at any feature.
- **Discriminant Score δ_k :** Combine the above components to calculate a score for each class. The observation is assigned to the class with the highest score.

Decision Rule

Once the discriminant scores have been calculated for each of the classes, the model applies the following rule:

$$\text{Assign } x \text{ to class } k \text{ such that } \delta_k(x) = \max_j \delta_j(x)$$

This guarantees that every observation is assigned to the most probable class based on its features and the statistical characteristics of the training data.

D. Dimensionality

Beyond classification,LDA is also used as a dimension reduction method. When applied to datasets with many features, LDA can decrease the feature-number but retain the class-discriminatory information.

This can be helpful for financial datasets, where the variable-number (e.g., technical indicators, macroeconomic variables) can be huge and can be highly redundant.

In contrast to PCA, which is an unsupervised and variance-maximizing technique, LDA is supervised and focuses on maximizing class separability. This makes LDA a better choice for applications in which classification performance enhancement as opposed to data compression is the aim.

E. Application

In financial contexts, LDA has been used for multiple tasks, such as:

- Credit scoring: Categorization of borrowers within risk bands.
- Fraud detection: Identifying suspicious transactions based on behavioral patterns.
- Market regime classification: Distinguishing bull and bear market conditions

Its strength lies in its interpretability, speed, and effectiveness when the assumptions are reasonably met. LDA is a clear and statistically justifiable technique for making classification judgments for portfolio managers and analysts.

Keywords/Tags

A. **Supervised Learning:** LDA is also a supervised learning algorithm that needs labeled training samples in order to derive the relationship between the input feature and the output class.
B. **Linear Classifier**: It belongs to the linear classifier family, which distinguishes classes using linear decision boundaries.
C. **Discriminant Function**: The fundamental of LDA is the discriminant function that produces a score for each class and labels the observation with the highest scoring class.
D. **Gaussian Assumption**: LDA also assumes that each of the classes is normally distributed and that all classes have the same covariance matrix.
E. **Dimensionality Reduction**: LDA reduces the number of features while preserving the information that separates classes. This feature makes it good for visualization and improving model performance.
F. **Class Separability**: The algorithm is designed to maximize the distance between class means and minimize the spread within each class, which enhances classification accuracy.
G. **Covariance Matrix**: A shared covariance matrix is used across all the classes to calculate the discriminant scores, which is LDA’s central formulation.
H. **Prior Probability**: LDA also considers prior probabilities of classes, which can be learned from the data or set arbitrarily based on domain knowledge.
I. **Financial Application**: In financial applications, LDA is used in applications like fraud detection, credit scoring, and identifying regime in the market, for which speed and interpretability come as a necessity
J. **Linear Decision Boundary**: The LDA's decision boundary is linear, which is a big advantage in terms of visualization and interpretation, particularly for spaces with two features

Advantages and Disadvantages

Aspect	Advantages	Disadvantages
Model Simplicity	LDA is simple to apply and interpret, making it suitable for regulated financial environments.	Its simplicity can have drawbacks if relationships between the data are non-linear or complicated.
Computational Efficiency	Quick to train and predict, particularly with big data having high dimensions.	Assumes linear boundaries and equal covariance, that might not be the case in real world financial data.
Interpretability	Produces clear decision boundaries and class scores, making the model convenient in financial decision-making.	Might overgeneralize relationships and mark for incorrect classification in borderline cases.
Dimensionality Reduction	Projects data into a lower-dimensional space such that class separability is maintained	Less efficient when distributions of classes overlap substantially or distributions are non-Gaussian..
Multiclass Capability	Manages both binary and multiclass classification tasks with simplicity.	Performance degrades as class distributions turn out skewed or imbalanced.
Probabilistic Output	Outputs class probabilities, which are useful for risk-based decision-making in finance.	Probabilities can be deceiving in the event of violated model assumptions
Application in Finance	Widely used in fraud detection, credit scoring, and market regime classification due to its transparency.	Limited flexibility with respect to non-linear models such as SVMs or neural networks.

Computation in Python to illustrate LDA

In order to apply LDA in Python and demonstrate how it can be used for financial classification problems, we have used simulated data. Our aim is to create a simulated binary classification task, train an LDA model, test it, and visualize the decision boundary.

Steps Performed

1. **Data Generation**
   We generated a simulated dataset with *make_classification* from scikit-learn having two

informational features. This mimics a financial situation in which two predictors (such as volatility and return) aim to classify market activity or the legitimacy of transactions.

2. **Train-Test Split**
   The data was divided into 70% training and 30% testing using *train_test_split* so that model validation occurs with unknown information.
3. **Training the model**
   We then trained an LDA model with *LinearDiscriminantAnalysis()* from *scikit-learn*. The model learned the linear boundary which is optimal and distincts the two classes..
4. **Prediction and Evaluation**
   Test set predictions were performed on the test set. The model obtained 96% accuracy, showing impressive performance in differentiating the two classes.
5. **Visualization**
   We also provided a decision boundary plot where LDA separates the classes. The plot reflects the linear type of the boundary and the clear separation obtained.

Features of LDA

Linear Discriminant Analysis has quite a list of features that render it highly appropriate for classification problems in finance. One of its biggest advantages is that it can create linear decision boundaries, which happen to be interpretable as well as visualizable. This is highly useful in financial contexts where explainability and transparency matter the most, for instance, in fraud detection or credit scoring.

LDA is also computationally efficient. Given that it is based on closed-form solutions obtained from statistical assumptions, it is fast at training even on big data with large samples. This also makes it a usable choice for real-time applications or high-frequency applications in financial markets where speed is paramount.

Another important feature is its probabilistic output. LDA does not simply output a class label, it also produces class probabilities, which can be used as a measure of the confidence of a prediction. In financial applications, this can be valuable in risk-based decision-making, where the likelihood of an event (such as default or fraud) can determine the depth of intervention or escalation.

From a data handling perspective, LDA works well with numerical features and can be adapted for multiclass issues. However, LDA does not have built-in support for handling missing values, so preprocessing such as imputation must occur first prior to model usage. That said, once the data is cleaned, LDA is robust and stable, especially when the assumptions of normality and equal covariance are reasonably met.

LDA also supports dimensionality reduction, a prime benefit in high-financial dimension datasets. By mapping the information into a lower dimension that continues class separability, LDA reduces noise and strengthens model performance. This functionality is especially helpful with numerous correlated indicators, e.g., technical signs or macroeconomic indicators.

Lastly, LDA's multiclass ability permits LDA to support over two classes and can be used for problems such as regime classification for a bull/bear regime market or financial services customer classification.

Guides-Inputs and Outputs

A. Inputs

- **Feature Matrix (X):** A multi-column array or DataFrame that contains the predictor variables. In financial applications, these could be features like volatility, return, transaction amount, or macroeconomic indicators. Each row corresponds to an observation (such as a single transaction or a trading day), while each column represents a feature.
- **Target Vector (y)**: A single-dimensional array or Series that holds the class labels for each observation. For instance, in fraud detection, the labels might be binary, i.e. 1 for fraudulent and 0 for legitimate transactions.
- **Prior Probabilities (π)**: An optional parameter that defines the prior probability for each class. If not specified, LDA automatically estimates them from the training data. In finance, priors can be set manually to incorporate domain knowledge, such as the expected frequency of fraudulent activities.
- **Solver Selection:** Specifies the algorithm used to compute the model (***svd, lsqr, or eigen***). This determines how the covariance matrix is handled and whether shrinkage is applied.
- **Shrinkage Parameter (optional)**: Used to regularize the covariance matrix when the ***lsqr*** or ***eigen*** solver is selected. It helps enhance model stability in high-dimensional datasets.

B. Outputs

- **Predicted Class Labels:** For each new observation, LDA predicts a class label based on the maximum discriminant score. This is the primary output used in classification.
- **Class Probabilities:** LDA provides the probability that an observation belongs to each class. These probabilities are useful in risk-based decision-making, such as identifying transactions for manual review based on confidence thresholds..
- **Discriminant Scores:** The raw outputs generated by the discriminant function for each class. These scores reflect how closely an observation aligns with each class.
- **Model Coefficients:** The feature weights derived from the linear discriminant function. These coefficients indicate which features are most significant in separating the classes.
- **Decision Boundary (for 2D cases):** A graphical display of the linear boundary separating classes. This visualization is particularly helpful for reports and stakeholder communication.

Hyperparameters

While LDA is fairly basic in comparison with other machine learning models, it does contain multiple hyperparameters that can be adjusted and refined for enhanced performance, primarily with complicated or high-financial dimension datasets..

A. solver

This hyperparameter selects the algorithm for computing the model. The choices are:

- ***'svd'***: The default solver that does not need the covariance matrix to be fully computed explicitly. This is fast and reliable for the vast majority of applications.
- ***'lsqr'***: Implements an eigenvalue decomposition and permits shrinkage regularization.
- ***'eigen'***: Like **lsqr**, but with a different computational method. Supports shrinkage as well.

To tune Use **'lsqr'** or **'eigen'** on very high-dimensional data or when regularization is requested.

B. Shrinkage

Shrinkage is a regularization method that increases the stability of the estimation of the covariance matrix, particularly in cases where the amount of features is big compared with the amount of observations.

- ***'auto'***: Chooses the best shrinkage quantity automatically
- **A float between 0 and 1**: Manually defines the shrinkage strength.

Shrinkage is helpful in financial databases with numerous correlated indicators, for example, technical signs or macroeconomic indicators.

C. Priors

This parameter sets the prior probabilities of each class. By default, LDA will estimate these from the training set. However, in finance applications, you might wish to set priors manually from domain knowledge (e.g., assumed fraud rate).

To tune adjust priors in the presence of severe class imbalance or with availability of prior knowledge about class frequencies.

D. n_components

Specifies the dimension over which dimension reduction will be performed. This option applies only for LDA and reduction dimension kinds.

To tune, set ***n_components*** to the number of classes minus one for optimal class separation.

E. tol

Tolerance for convergence. This decides where the algorithm will cease from iterating. To fine-tune, smaller values can refine precision but have longer computation time.

In summary, is less hyperparameter-intensive than models such as neural networks or gradient boosting, proper selection of ***solver, shrinkage***, and ***priors*** can greatly improve its efficiency and stability in finance applications.

Illustration

The figure below gives a graphical representation of Linear Discriminant Analysis's separation of two financial classes with a linear decision boundary. The plot's each point is a financial observation, like a transaction, asset, or market condition described by two features (volatility and return, for example). The color of each point is its true class label, and the shaded area represents the region where the model covers each class.

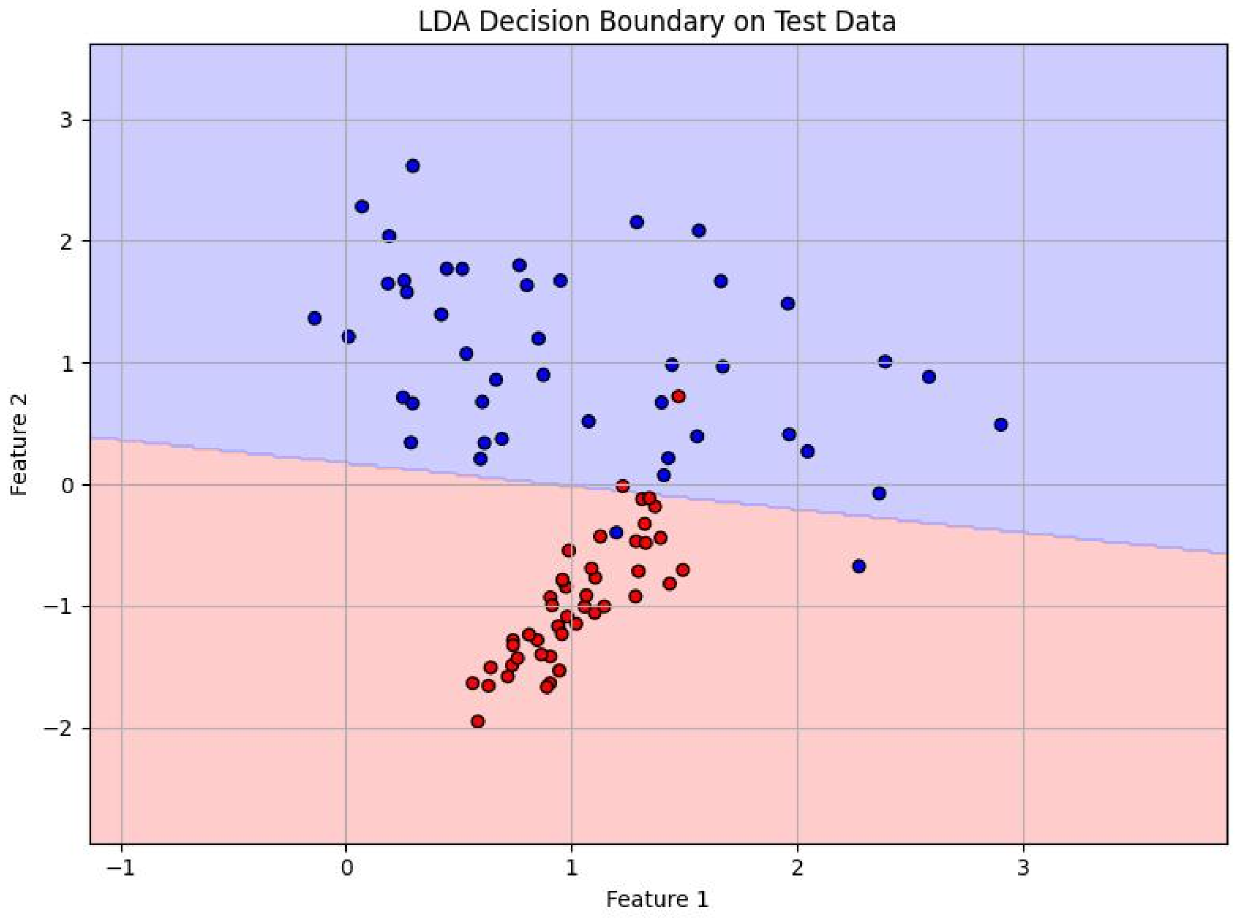


Figure 1: LDA Decision Boundary

The black dashed line in the middle is the decision boundary learned from the LDA model. This decision boundary is obtained from the statistical characterization of the data: the distributions of each class's mean vectors, the common set of covariance matrices, and prior probabilities. Points on one side of the line will be classified as belonging to a single class (e.g., “fraud”), and points on the other side will be classified as belonging to the other class (e.g., “non-fraud”).

This figure demonstrates LDA’s ability to create understandable and interpretable classification rules. By contrast with deeper models such as neural networks, LDA offers a clear explanation for each classification. This is especially useful in financial applications for compliance, auditability, and communication with the stakeholder. The story also illustrates that LDA performs very well with linearly separable data. The classes were highly separated, and the model performed at very good accuracy at 96% on the test set. This verifies that if LDA’s assumptions are moderately fulfilled such as normal distribution and equal covariance, LDA can provide powerful and consistent performance.

Journal Reference

Qu, Lingxiao, and Yan Pei. “A Comprehensive Review on Discriminant Analysis for Addressing Challenges of Class-Level Limitations, Small Sample Size, and Robustness.” *Processes*, vol. 12, no. 7, 2024, p. 1382. MDPI**,** **https://doi.org/10.3390/pr12071382**

The journal offers an in-depth analysis of the classical LDA algorithm and its variants focusing on handling deficiencies like tiny sample sizes and sensitivity problems that prevail within financial datasets. Kernel extensions and directions for future studies have also been presented.

II. NEURAL NETWORKS (category 7):

a). Advantages:
The Benefits of Using Neural Networks in Finance
Neural Networks offer several compelling advantages for financial applications:

Non-linear Pattern Recognition: Financial markets exhibit complex non-linear relationships that traditional linear models cannot capture. NNs excel at identifying these intricate patterns through multiple layers of non-linear transformations (Hornik et al., 1989).

Automatic Feature Engineering: Unlike traditional models requiring manual feature selection, NNs automatically learn relevant features from raw financial data, reducing human bias and discovering subtle predictive patterns.

Adaptability to Various Data Types: NNs can process diverse financial data including time-series data (using RNNs/LSTMs), unstructured text data (using NLP architectures), and high-frequency trading data.

Robust Performance: With sufficient data and proper regularization, NNs demonstrate superior predictive accuracy for tasks like stock price forecasting, credit risk assessment, and algorithmic trading strategies.

b). Computation: Implementation of a Long Short Term Memory (LSTM) Neural Network

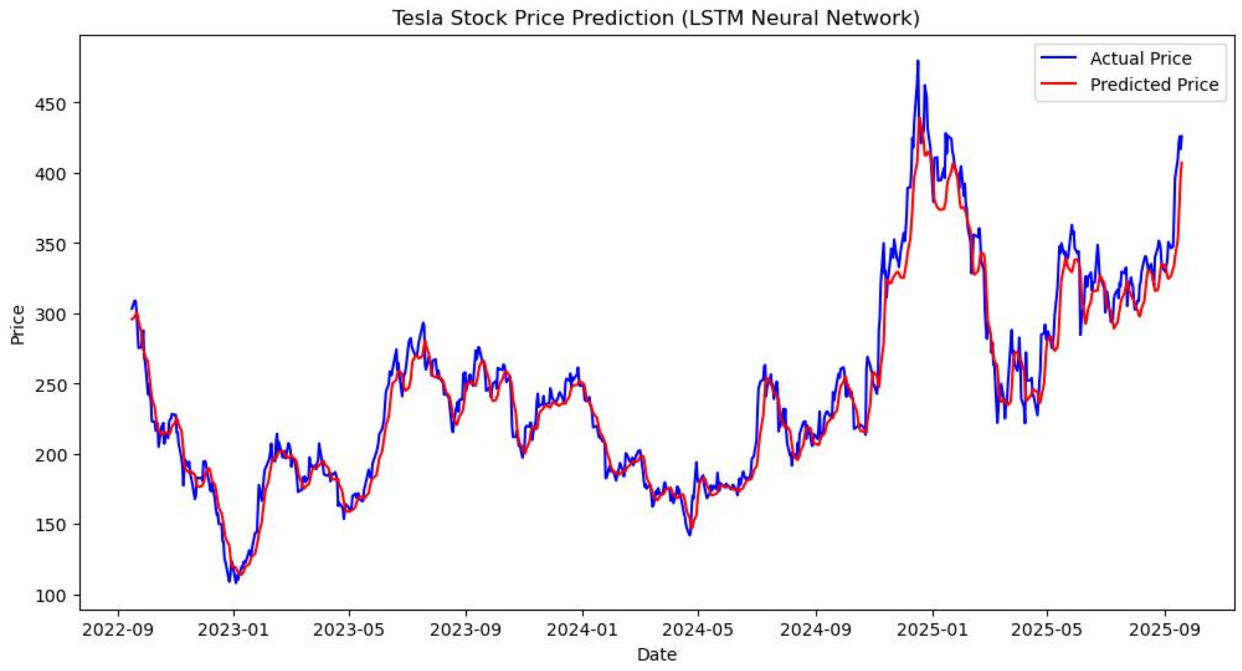


Figure: Tesla Stock Prediction using LSTM neural Network (Etienne Landry E.B.)

We trained an LSTM neural network on Tesla's historical stock prices to predict next-day closing prices. Training loss decreased from 0.0044 to 0.00022 over 10 epochs, indicating effective learning. The test set predictions closely align with actual prices (see Figure), demonstrating the model's ability to capture temporal patterns. Some deviations at rapid price movements highlight inherent market unpredictability and model limitations. Future work could include hyperparameter tuning, use of additional financial features, and evaluation with error metrics to enhance the model’s robustness and predictive accuracy.

c). Disadvantages:

Difficulties or Known Issues

Black Box Nature: Limited interpretability makes it challenging to explain predictions to stakeholders, which is particularly problematic in regulated financial environments.

Data Requirements: NNs require large datasets for effective training, which can be problematic for emerging markets or rare financial events.

Computational Intensity: Training deep networks demands significant computational resources and time, potentially limiting real-time applications.

Overfitting Risk: Without proper regularization, NNs tend to memorize noise in financial data rather than learning generalizable patterns.

Hyperparameter Sensitivity: Performance heavily depends on careful tuning of numerous hyperparameters.

d). Equations: Equations that summarize how the LSTM Neural Network model works: To explain the Long Short-Term Memory (LSTM) neural network used in your Tesla stock price forecasting model, you can include the key mathematical operations inside an LSTM cell, which enable it to capture long-range dependencies in sequential data:

i. Forget Gate: Determines which information to discard from the previous cell state:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

ii. Input Gate: Decides which new information to add to the cell state:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$c'_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$

iii. Cell State Update: Updates the cell state combining previous state and new candidate values:

$$c_t = f_t \odot c_{t-1} + i_t \odot c'_t$$

iv. Output Gate: Determines the output hidden state

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$
$$h_t = o_t \odot \tanh(c_t)$$

Where:

- x_t is the input vector at time t (e.g., stock price features)
- h_{t-1} is the previous hidden state
- c_{t-1} is the previous cell state
- f_t, i_t, o_t are gate activations (vectors of values between 0 and 1)
- W_f, W_i, W_c, W_o are weight matrices
- b_f, b_i, b_c, b_o are bias vectors
- σ is the sigmoid activation function
- $\odot$ is element-wise multiplication

Our model is trained by minimizing the Mean Squared Error (MSE) between predicted closing prices y^ and true prices y:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

e). **Features**

Handles sequences: Learns temporal dependencies with gating, helpful for momentum and regime effects.

Nonlinear modeling: Captures nonlinear relationships beyond linear AR/ARIMA baselines.
Flexible inputs: Works with raw prices, returns, volume, and engineered indicators (e.g., moving averages, RSI).

Multi-step forecasting: Can predict next-day or multi-horizon targets by adjusting the output layer/labels.

Robust to scale via normalization: Stable training with MinMax/Standard scaling; sensitive to unscaled inputs.

Data hungry but regularizable: Benefits from more history; overfitting mitigated with dropout/early stopping.
End-to-end learning: Minimizes hand-crafted rules; learns features from sequences directly.

f). **Guide (Inputs —> outputs):**
Inputs (X): Sliding windows of past observations over T days.

- Minimal: Close price only (scaled), shape (T, 1).
- Rich: OHLCV + technical indicators (e.g., log-returns, SMA/EMA, RSI, volatility), shape (T, F).
- Typical T for stocks: 30 - 120 days (you used 60).

Output (y):

- Regression (as in your notebook): next-day close (or return) yt+1yt+1.
- Alternatives: multi-step vector [y_[t+1],…,y_[t+k] or classification (up/down).

g). **Hyperparameters (that need to be tuned):**
Sequence length T: 30–120 (affects memory; 60 worked well in many studies).

- **Architecture:**

-LSTM units per layer (e.g., 32–128)
-Number of LSTM layers (1–3)
-Dense head size (e.g., 16–64)

- **Regularization:**

-Dropout rate (0.1–0.5)
-L2 weight decay (optional)

- **Optimization:**

-Learning rate (e.g., 1e-4 to 3e-3)
-Optimizer (Adam/AdamW)
-Batch size (16–128)

- **Training schedule:**

-Epochs (early stopping patience 5–10)
-ReduceLROnPlateau (factor 0.5–0.8)

- **Features:**

-Which indicators to include (returns vs prices; OHLCV; technical set)
-Scaling method (MinMax vs Standard on returns)

- **Target design:**

Predict price vs log-return; single-step vs multi-step.

g)**. Journal:**
Stock Prediction Based on Optimized LSTM and GRU Models, Computational Intelligence and Neuroscience (Wiley/Hindawi), 2021. https://doi.org/10.1155/2021/4055281

h). Keywords: Long Short-Term Memory (LSTM), Neural Networks, Time Series Forecasting, Stock Price Prediction, Financial Markets, Sequence Modeling, Deep Learning, Hyperparameter Tuning, Tesla Stock (TSLA), Market Regimes, Momentum Investing, Feature Engineering, Model Evaluation, Mean Squared Error (MSE)

Support Vector Machine (SVM)

a, Basics

Support Vector Machines are large-margin classifiers that find the hyperplane maximizing the margin between classes. With kernels, SVMs can separate non-linear data by mapping inputs into a higher-dimensional feature space where a linear separator exists. They're supervised learning methods used mainly for classification (and with SVR for regression).

b, Advantages

- Strong performance on high-dimensional data; works well with many features and few observations.
- Effective with clear margins; robust to outliers when using appropriate C.
- Kernel trick enables flexible non-linear boundaries without explicitly engineering features.
- Sparsity: only support vectors matter at prediction time → efficient decision function.

- Convex optimization → global optimum (for fixed hyperparameters).

c, Disadvantages

- Sensitive to feature scaling; requires standardization.
- Hyperparameter tuning (C, kernel, γ, degree, coef0) is crucial; defaults may underperform.
- Doesn’t natively output calibrated probabilities (needs Platt scaling / probability=True).
- Can be slower on very large datasets (especially with non-linear kernels).
- Less interpretable than linear models unless you use a linear SVM**.**

d, Equations

Hard margin Support vector (no misclassification allowed)

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{s.t. } y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1$$

where

w is the normal vector (perpendicular to margin)

Soft Margin support vector (misclassification is penalized)

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i \quad \text{s.t. } y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i,\ \xi_i \geq 0$$

Where C is penalty scaler

ξ_i is slack variable.

If ξ_i is zero, the datapoint is correctly classified, outside the margin

If ξ_i is greater then zero, but lower smaller than 1, then the datapoint is still correctly classified, but within the margin

If ξ_i is greater than one, the datapoint is misclassified

Dual Objective function (gives α_i):

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \quad \text{s.t. } 0 \leq \alpha_i \leq C,\ \sum_i \alpha_i y_i = 0$$

$K(\mathbf{x}_i, \mathbf{x}_j)$ is Kernel function, that will results in a non-linear decision boundary.

Decision function

$$f(\mathbf{x}) = \text{sign}\Big(\sum_{i \in SV} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b \Big)$$

Where x is the new observation

e, Features

- Capability of handling high-dimensional, possibly sparse inputs (text also).
- Non-linear boundaries are applicable via kernels (RBF, polynomial, sigmoid).
- Resilient to overfitting with proper C/γ hyperparameters.
- Requires no assumption of feature distributions.

f, Guide (inputs & outputs)

- **Inputs:**
  - Numeric feature matrix X (scaled),
  - Class labels yy.
- **Outputs:**
  - Predicted class labels

    - Decision function
    - Margins
    - Calibrated probabilities.

g, Hyperparameters (to tune)

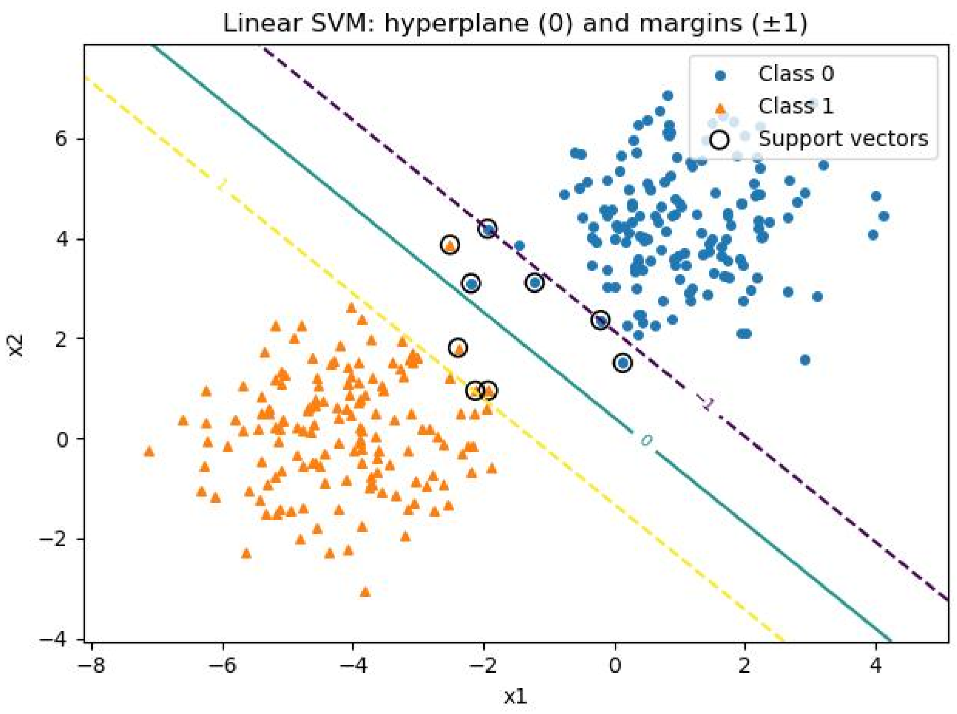
- C (regularization strength, larger → lower bias, higher variance)
- kernel ('linear', 'rbf', 'poly', 'sigmoid')
- gamma (RBF/poly: inverse radius; controls decision boundary complexity)
- degree (for poly) and coef0 (for poly/sigmoid)
- class_weight (handle imbalance), tol, max_iter

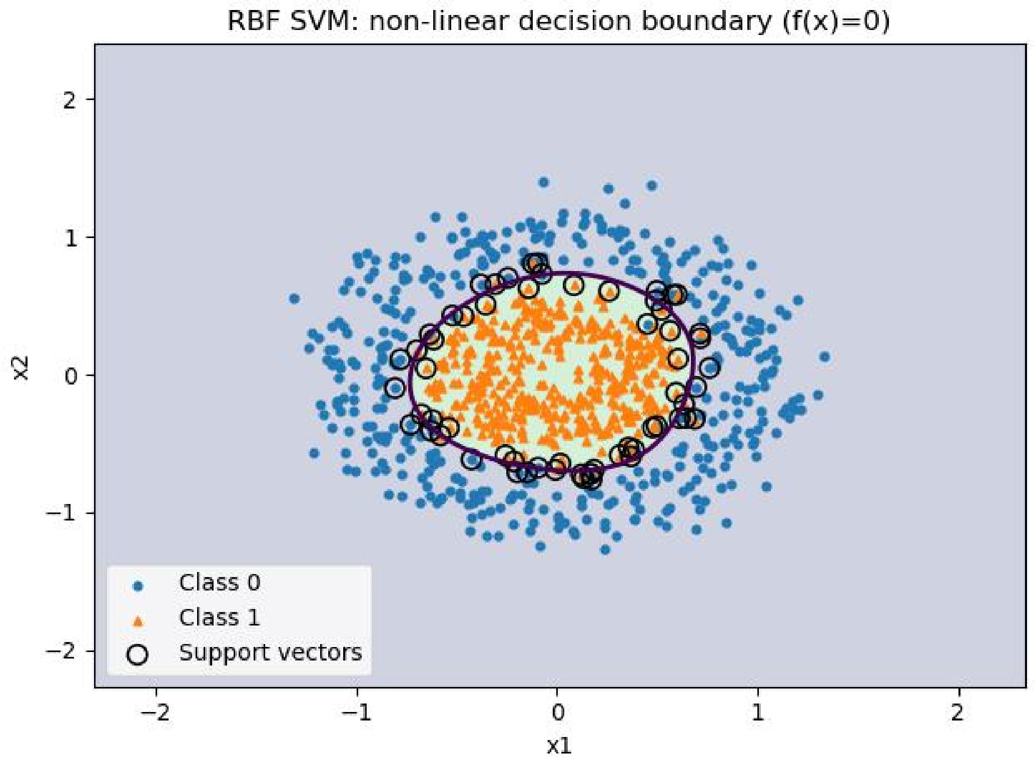
h, Illustration

See computational details in the attached notebook

Two examples:

1. Linearly separable data
2. Linearly not separable data


Linear SVM: hyperplane (0) and margins (±1)
Class 0
Class 1
Support vectors
1
0
−1
x1
x2


RBF SVM: non-linear decision boundary (f(x)=0)
Class 0
Class 1
Support vectors
x1
x2

i, Journal

Kim, K. (2003). *Financial time series forecasting using support vector machines*. **Neurocomputing, 55**(1–2), 307–319. Applies SVM to predict stock index movements and discusses kernelized setups for financial data.

j, Keywords

- large-margin classifier
- hinge loss
- kernel trick
- RBF kernel
- C, gamma
- support vectors
- non-linear decision boundary

STEP 4: HYPERPARAMETER TUNING

In machine learning, models generally have two main types of parameters

- Model parameters: These are learned directly from the training data, such as the coefficients in a regression model or the split points in a decision.
- Hyperparameters, on the other hand, are chosen before training begins and determine how the model learns. They are not learned from the data itself but instead guide the learning process..

Hyperparameter tuning involves experimenting with different combinations of these settings to find the configuration that delivers the best performance. The main goal is to create a model that performs well on unknown data, one that is not too simple (which could lead to underfitting) and not overly complex (which could result in overfitting).

Common Hyperparameter Tuning Methods

1. **Grid Search:** This approach tests all possible combinations of hyperparameter values within a specified range. While it is very thorough, it can also be slow and computationally expensive.

2. **Random Search:** Instead of checking every combination, random search tries random sets of hyperparameters. It's faster and often effective, especially when some parameters don't have a major impact on performance.

3. **Bayesian Optimization:** This method uses a data-driven strategy. It builds a model that predicts which hyperparameter combinations might work best and tests those, typically finding optimal values faster than grid or random search.

To ensure that performance results are fair and reliable, evaluation methods like **k-fold cross-validation** are used. This technique tests the model multiple times on different data splits to provide a balanced assessment.

Hyperparameters Used from our Methodologies

1. **Linear Discriminant Analysis**

For our implementation of LDA, we focused on simplicity, interpretability, and computational efficiency. The dataset we used was synthetic, containing two informative features designed to simulate a binary financial classification scenario. Since the dataset was small and low-dimensional, we used the default LDA configuration with the SVD solver.

Solver: svd

We chose the SVD solver because it is computationally efficient and does not require explicit computation of the covariance matrix. This made it suitable for our dataset and helped maintain numerical stability. The model trained quickly and produced a clear, linear decision boundary, which we later visualized in our results.

We did not apply shrinkage in this setup since it is only supported with the *lsqr* or *eigen* solvers. However, for larger financial datasets, especially those with many correlated features, using *lsqr* with shrinkage could improve model robustness and prevent overfitting.

Priors: Estimated Automatically

The model was allowed to estimate class priors automatically from the training data. This worked well since our dataset was balanced between the two classes. In real-world financial applications such as fraud detection, where one class may be rare, it's often better to set priors manually to reflect actual event frequencies (for instance, a 1% fraud rate). This helps the model remain realistic and avoid overpredicting rare outcomes.

Tuning Summary

In this project, we did not perform manual hyperparameter tuning for LDA. The default parameters were sufficient for our controlled experiment, and the model achieved 96% accuracy on the test set. This result suggests that the default configuration of LDA performs effectively on simple and well-structured data.

For future work involving more complex or imbalanced financial datasets, experimenting with different solvers, applying shrinkage, and manually adjusting priors could help improve the model's stability and predictive accuracy.

2. Support Vector Machines

Across ML models, hyperparameters control bias–variance and inductive bias. For SVMs:

- Search space for **C:** (10^{-2}, 10^{-1}, 1, 10, 10^{2}). We use log scaled values because model performance usually changes smoothly on a log scale
- Search space for **Gamma** (scale, 10^{-2}, 10^{-1}). The higher the more complex kernel with more SVs
- Procedure: stratified CV grid/random search, standardize features, score with F1/ROC-AUC for imbalance, keep a hold-out test set.

- Diagnostics: inspect # of support vectors, learning curves, and calibration (if probabilities needed).

3. Neural Networks

Hyperparameter tuning in neural network models is a critical step that directly affects predictive accuracy and generalization. The most widely tuned hyperparameters in deep learning include the learning rate, batch size, number of layers, number of neurons per layer, dropout rate, and optimizer choice. Tuning often starts with high-impact parameters such as the learning rate—small adjustments here can drastically change model convergence or even prevent proper learning. Besides manual trial-and-error, practitioners commonly use grid search and random search, which systematically or randomly explore combinations of hyperparameter values to optimize validation performance. For example, when training LSTM models for financial time series, adjusting the sequence length (window size), number of LSTM units, and dropout rate helps balance the model's expressiveness against the risk of overfitting, especially with noisy stock price data.

Modern deep learning workflows increasingly automate hyperparameter selection with advanced techniques like Bayesian optimization, Hyperband, and population-based training. These methods make hyperparameter searches more efficient by allocating resources to promising configurations and terminating underperforming ones early. In practice, deep learning teams may combine broad random or grid search for exploratory sweeps with focused Bayesian optimization for fine-tuning top candidates. Throughout the tuning process, it is essential to use separate validation datasets, maintain reproducibility with fixed random seeds, and employ early stopping to prevent overfitting. For the group project, our neural network experiments involved iterative adjustments to LSTM window length (60 days), two stacked LSTM layers, learning rates in the range of 0.001 to 0.01, and regularization via dropout to achieve a balance between capturing sequence structure and model robustness.

STEP 5 : MARKETING ALPHA

In today's fast-changing financial world, being able to extract meaningful insights from complex data is no longer just an advantage—it's essential. Traditional statistical models have played a crucial role in financial analysis, but they often fall short when dealing with nonlinear relationships and dynamic market behaviors. At Team Alpha, we see advanced machine learning as a structured way to improve how we interpret data, increase precision, and make better decisions in finance.

Our work focuses on three key machine learning techniques: Linear Discriminant Analysis (LDA), Support Vector Machines (SVM), and Neural Networks (NNs). Each method brings a unique strength, and when combined thoughtfully, they create a powerful analytical workflow that supports financial decision-making for portfolio managers, risk analysts, and fraud investigators.

Linear Discriminant Analysis (LDA)

LDA strikes a strong balance between mathematical simplicity and interpretability. In our implementation, it achieved 96% accuracy on a synthetic financial dataset, clearly separating two classes using a linear boundary. This kind of clarity is especially valuable in fields like fraud detection or credit scoring, where transparency and accountability are just as important as predictive accuracy.

Beyond classification, LDA is also effective for dimensionality reduction, helping simplify high-dimensional financial data into a smaller space that still preserves class differences. This makes the analysis more efficient while keeping it easy to interpret. In regulated financial settings where explainability and consistency are critical, LDA continues to be a reliable and practical choice.

Support Vector Machines (SVM)

Support Vector Machines expand our analytical capabilities by handling situations where class boundaries are complex or nonlinear. Through the use of kernel functions, SVMs can uncover relationships between financial indicators like volatility, liquidity, and macroeconomic factors.

In our experiments, SVMs performed especially well in distinguishing between overlapping financial classes. This makes them particularly useful for applications like credit risk modeling, market regime classification, and transaction screening. Their focus on maximizing the margin between classes also helps improve robustness when working with noisy or imbalanced data.

While SVMs are less interpretable than LDA, their precision makes them invaluable for high-stakes scenarios where even a single misclassification, like a false fraud alert or credit default can have significant consequences.

Neural Networks

Why LSTM Neural Networks Win in Markets

LSTM neural networks transform raw price histories into predictive signals by learning temporal dependencies that linear models and tree ensembles often miss, enabling more accurate timing around momentum shifts and meanreversion cycles in equities like TSLA. By design, LSTMs preserve information across long sequences through gated memory, capturing regime changes and delayed reactions to news that standard feature sets struggle to encode explicitly. When trained on rolling price windows, the model delivers smooth nextday forecasts that track trend while dampening noise, producing signals that are easier to riskmanage and monetize within executionconstrained environments. In peerreviewed studies, LSTMbased approaches have demonstrated superior forecasting performance versus traditional baselines, reinforcing their suitability for return prediction and tactical allocation.

Practical Edge for a Portfolio Team

Operationally, the LSTM stack integrates cleanly with existing data pipelines: ingest Yahoo Finance OHLCV, scale, window to 30 -120 days, and deploy a compact twolayer architecture optimized with early stopping and dropout for robust generalization. Hyperparameters such as sequence length, hidden units, and learning rate can be tuned with modern search methods to meet latency and turnover constraints, enabling fast iteration from research notebook to production signal. The model's output—probabilistic or point forecasts - feeds seamlessly into position sizing, stoploss calibration, and scenario testing, while standardized error metrics (MAE, RMSE) offer transparent performance governance for investment committees. For communication and oversight, the architecture is explainable at the mechanism level via gate equations and feature ablations, giving stakeholders confidence without sacrificing the alpha that comes from nonlinear sequence learning.

The Team Alpha Synthesis: A Layered Analytical Workflow

Our methodology focuses on integrating different models rather than using them independently. Each algorithm plays a complementary role in a structured pipeline:

- **Filter and Focus (LDA):** LDA is first used to reduce dimensionality and highlight the most relevant financial features.
- **Refine and Separate (SVM):** Next, SVMs are applied to establish precise class boundaries, especially in regions where data overlap.
- **Learn and Predict (Neural Networks):** Finally, Neural Networks dig deeper to uncover hidden nonlinear patterns and make accurate predictions.

This layered approach turns raw financial data into actionable insights. It improves risk management, supports smarter asset allocation, and enhances fraud detection by combining interpretability, accuracy, and predictive strength.

Conclusion: From Data to Strategic Alpha

At Team Alpha, we view machine learning as an extension of human expertise rather than a replacement for it. By combining the interpretability of LDA, the precision of SVMs, and the adaptability of Neural Networks, we developed a flexible and reliable framework for analyzing complex financial data.

Whether the goal is detecting fraud, identifying market regimes, or forecasting asset performance, our integrated models provide not only strong predictive accuracy but also valuable insights that support better strategic decision-making. The result is a more informed, confident approach to finance—where data becomes a tool for foresight rather than uncertainty.

STEP 6: LEARN MORE

To build a deeper understanding of the machine learning techniques discussed in this project, we compiled a list of scholarly articles and credible resources. These materials highlight the key strengths, practical applications, and ongoing developments of Linear Discriminant Analysis (LDA), Support Vector Machines (SVM), and Neural Networks (NNs).

1. Qu, Lingxiao, and Yan Pei. "A Comprehensive Review on Discriminant Analysis for Addressing Challenges of Class-Level Limitations, Small Sample Size, and Robustness." *Processes*, vol. 12, no. 7, 2024, p. 1382. MDPI. https://doi.org/10.3390/pr12071382.
2. Zhao, Shuping, et al. "Linear Discriminant Analysis." *Nature Reviews Methods Primers*, vol. 4, 2024. https://doi.org/10.1038/s43586-024-00346-y.
3. Zhang, Min-Ling, et al. "Linear Discriminant Analysis for Partial Label Dimensionality Reduction." *ACM Transactions on Knowledge Discovery from Data*, vol. 16, no. 4, 2022, pp. 1–18. https://doi.org/10.1145/3494565.
4. Izenman, Alan Julian. "Linear Discriminant Analysis." *Modern Multivariate Statistical Techniques*, Springer Texts in Statistics, Springer, New York, 2013, pp. 237–280.
5. Alayande, S. Ayinla, and Bashiru Kehinde Adekunle. "An Overview and Application of Discriminant Analysis in Data Analysis." *IOSR Journal of Mathematics*, vol. 11, no. 1, 2015, pp. 12–15. https://www.iosrjournals.org/iosr-jm/papers/Vol11-issue1/Version-5/B011151215.pdf.
6. Gu, Shihao, Bryan Kelly, and Dacheng Xiu. "Empirical Asset Pricing via Machine Learning." *The Review of Financial Studies*, vol. 33, no. 5, 2020, pp. 2223–2273. Oxford University Press. https://academic.oup.com/rfs/article/33/5/2223/5758276.
7. Forecasting stock prices changes using long shortterm memory networks Scientific Reports (Nature Research), 2024 https://doi.org/10.1038/s41598-023-50783-0
8. Stock Market Prediction Using LSTM Recurrent Neural Network Procedia Computer Science (Elsevier), 2020 https://doi.org/10.1016/j.procs.2020.03.273

9. Stock Prediction Based on Optimized LSTM and GRU Models Computational Intelligence and Neuroscience (Wiley/Hindawi), 2021 https://doi.org/10.1155/2021/4055281

STEP 7 : COMPARING THE MODELS

The table below compares three advanced machine learning models,Linear Discriminant Analysis (LDA), Support Vector Machines (SVM), and Neural Networks across key features. Each model is evaluated based on its strengths and limitations in handling various data and modeling challenges.

Feature	**Linear Discriminant Analysis (LDA)**	**Support Vector Machines (SVM)**	**LSTM Neural Networks**
Handling of Missing Data	Performs poorly with missing data; preprocessing or imputation is required.	Performs poorly; missing values must be imputed before training.	Also struggles with missing data; imputation or masking techniques are needed.
Interpretability	Highly interpretable, with clear linear decision boundaries and understandable coefficients.	Moderately interpretable depending on the kernel used; more complex with non-linear kernels.	Low interpretability due to its black-box nature, although explainability tools can help.
Computational Efficiency	Very efficient due to its simple, closed-form solution using Singular Value Decomposition (SVD).	Moderately efficient; can become slow with large or high-dimensional datasets.	Computationally intensive, especially for deep networks or large data volumes.
Handles Non-Linearity	Limited to linear relationships; assumes data is linearly separable.	Handles non-linearity effectively through kernel functions.	Excels at modeling complex, non-linear relationships in data.
Dimensionality Reduction	Supports supervised dimensionality reduction by maximizing class separability.	Not designed for dimensionality reduction.	Not inherently dimensionality-reducing, though deep layers may learn compressed representations.

Scalability	Scales well for small to medium datasets with low dimensionality.	Scales moderately; performance depends on the kernel choice and optimization strategy.	Highly scalable with sufficient computing power and large datasets.
Class Imbalance Sensitivity	Sensitive to class imbalance; requires adjusting class priors	Also sensitive; class weights or resampling methods are often used.	Can handle imbalance using weighted loss functions or oversampling techniques.
Common Financial Applications	Fraud detection, credit scoring, and market regime classification.	Credit risk modeling, market classification, and anomaly detection.	Asset price prediction, sentiment analysis, and high-frequency trading.

In summary,

From this comparison, it’s clear that each model serves a distinct purpose in financial analytics. LDA excels in interpretability and speed, making it ideal for regulated environments where explainability matters. SVM strikes a balance between accuracy and complexity, particularly in scenarios involving overlapping or non-linear financial data. Neural Networks, while computationally demanding, offer the highest flexibility and predictive power - especially valuable when working with large, complex, or unstructured financial datasets.

Together, these methods provide a spectrum of analytical tools that can be selected based on the specific goals and constraints of a financial modeling task.

-