Thesis submitted to the

UNIVERSITY OF MOHAMED BOUDIAF – MSILA

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
DEPARTMENT OF COMPUTER SCIENCE

In partial fulfillment of the requirements for the degree of

Master in **Computer science**

Specialty: Information System and Software Engineering

By

Bensalah Fatima EZ-Zahra.

Title of the thesis

FinSight: Multitasks AI-Based Platform for Financial Market Prediction and Analysis

Under the supervision of

Said Gadri.

Composition of the jury

Salah Guesmia	University of Msila	President
Said Gadri	University of Msila	Reporter
Abdelbaki Bouguerra	University of Msila	Examiner

June, 2026

Abstract

This thesis presents FinSight AI, a multi-task financial decision support platform that applies machine learning and deep learning techniques to stock price prediction, trend classification, sales forecasting, credit risk assessment, and financial sentiment analysis. Several models were trained and evaluated on real-world datasets. The results show that deep learning does not always outperform classical machine learning and the used metrics proved the quality of the model. GRU achieved the best stock price prediction, Linear Regression performed best for trend classification, the Neural Ensemble led credit risk assessment, and fine-tuned FinBERT achieved 83.08% accuracy in sentiment analysis. All models were integrated into an interactive Streamlit web application.

Key words: Machine Learning; Deep Learning; Financial Prediction; sentiment analysis; Credit Risk; FinSight AI.

الملخص

تقدّم هذه الاطروحة منصةFinSight AI ، وهي منصة دعم قرار مالية متعددة المهام تعتمد على تقنيات التعلم الآلي والتعلم العميق لمعالجة التنبؤ بأسعار الأسهم، وتصنيف اتجاهات السوق، والتنبؤ بالمبيعات، وتقييم مخاطر الائتمان، وتحليل المشاعر المالية. تم تدريب وتقييم عدة نماذج باستخدام بيانات حقيقية. أظهرت النتائج أن التعلم العميق لا يتفوق دائماً على النماذج التقليدية؛ وان مقاييس التقييم هي التي تحدد جودة النموذج. إذ حقق GRU أفضل أداء في التنبؤ بالأسعار، وتفوق الانحدار الخطي في تصنيف الاتجاه، وتقييم مخاطر الائتمان بتفوق Neural Ensemble، بينما حقق FinBERT دقة 83.08% في تحليل المشاعر. تم دمج جميع النماذج في تطبيق ويب للتنبؤ الفوري.

الكلمات المفتاحية: التعلم الآلي، التعلم العميق، التنبؤ المالي، تحليل المشاعر، مخاطر الائتمان، .FinSight AI.

Résumé

Ce mémoire présente FinSight AI, une plateforme d’aide à la décision financière appliquant l’apprentissage automatique et l’apprentissage profond à la prédiction du prix des actions, à la classification des tendances boursières, à la prévision des ventes, à l’évaluation du risque de crédit et à l’analyse des sentiments financiers. Plusieurs modèles ont été entraînés et évalués sur des données réelles. Les résultats montrent que l’apprentissage profond ne surpasse pas toujours les méthodes classiques. Ses performances dépendent de la métrique. Le modèle GRU a obtenu la meilleure prédiction des prix, la régression linéaire a dominé la classification des tendances, et FinBERT a atteint une précision de 83,08 % en analyse de sentiment. Tous les modèles ont été intégrés dans une application web interactive.

Mots clés : Apprentissage automatique, Apprentissage profond, Prédiction financière, analyse de sentiment, Risque de crédit, FinSight AI.

DEDICATIONS

I lovingly dedicate this work to my dear parents, who taught me that knowledge is light and that ambition knows no bounds. To my father, who was and remains my role model in determination and perseverance. To my mother, who surrounded me with her prayers, love, and tenderness, and was the source of my strength and peace.

To my sisters, my friends and companions on this journey, and to everyone who taught me that success is not a destination, but a journey to be lived in all its details...

By God's grace and guidance, I proudly mark another achievement...

ACKNOWLEDGMENTS

First and foremost, all praise and gratitude are due to Allah (God), for granting me the strength, patience, and perseverance to complete this work. Without His blessings and guidance, none of this would have been possible.

I would like to express my deepest gratitude and sincere appreciation to my supervisor, ***Prof. Said Gadri****, for his invaluable guidance, continuous support, and constructive feedback throughout this work. His expertise, patience, and dedication have been a constant source of inspiration, and his insightful comments have significantly shaped this work.*

A special thanks to my parents, my siblings, and my extended family for always being there when I needed them.

Finally, I would like to thank my soulmates and colleagues especially those who walked this journey with me, for the shared struggles, and the moments of laughter that made this experience memorable.

Contents

List of figures

List of tables

INTRODUCTION

The relationship between data and financial decision-making has never been more consequential than it is today. Financial markets generate millions of data points every second, prices, volumes, transaction records, loan applications, news articles, social media posts and the institutions that can extract meaningful signals from this torrent of information hold a decisive advantage over those that cannot. For decades, this extraction relied on human expertise, econometric models, and rule-based systems. That era is giving way to a new one.

Machine learning entered financial analysis not with a single breakthrough but through a gradual accumulation of evidence that data-driven models could capture patterns that traditional approaches missed. The demonstration by Fischer and Krauss [1] in 2018 that LSTM networks could achieve statistically significant predictive returns on S&P 500 data marked a turning point not because the result was definitive, but because it opened a legitimate research question: under what conditions, for what tasks, and with what architectures does machine learning genuinely improve financial prediction? That question is the one this thesis attempts to answer.

Nowadays, the financial domain presents a set of challenges that make it an unusually demanding testbed for ML methods. Markets are non-stationary the statistical regularities that hold during a bull market may dissolve entirely during a liquidity crisis. [2] Labels are noisy two financial analysts reading the same earnings report may genuinely disagree on whether it signals positive or negative future performance. [3] Data is imbalanced loan defaults are rare events, negative financial sentiment is underrepresented in professional news [4] and perhaps most importantly, the cost of errors is asymmetric, missing a credit default is far more expensive than incorrectly flagging a creditworthy borrower. [5] These characteristics mean that accuracy alone is an insufficient measure of a model's value, and that the choice of evaluation metric is itself a methodological decision.

This thesis approaches the problem from the perspective of a practitioner who needs not just accurate models but deployable, interpretable, and contextually appropriate ones. Five prediction tasks are selected to cover the breadth of financial ML applications: stock price regression, stock trend classification, e-commerce sales classification, credit risk assessment, and financial sentiment analysis. These five tasks are not chosen at random they represent the four most active application domains in the financial ML literature (market prediction, credit scoring, demand forecasting, and NLP-based sentiment), and together

they span regression and classification, structured tabular data and unstructured text, balanced and severely imbalanced datasets, and short time series and large cross-sectional datasets addresses all five simultaneously under a common experimental protocol.

The objectives of this research are threefold. The first is technical: to build and evaluate a comprehensive set of ML and DL models spanning classical methods (Logistic Regression, SVM, Random Forest, XGBoost) and modern architectures (LSTM, GRU, BiLSTM, ResidualDNN, TabNet) across all five financial tasks, using real publicly available datasets and rigorous evaluation that prevent the data leakage documented in previous studies. The second objective is analytical: to identify the task-dependent conditions under which DL models outperform classical ML, and to provide interpretable explanations for the cases where they do not which is a contribution that task-isolated studies cannot offer. The third objective is practical: to deploy all trained models within **FinSight** AI, an interactive Streamlit web application that makes multi-model comparison and real-time inference accessible to financial practitioners without ML expertise.

The manuscript of our work is organized into six main sections:

In the **Introduction**, the domain of the study is cited, research context, problem statement, research objectives, and the structure of the thesis is outlined.

Chapter One provides an overview of financial markets and the theoretical foundations of machine learning and deep learning, establishing the conceptual vocabulary used throughout the work.

Chapter Two reviews the state of the art in financial ML, situating this research within the existing literature and identifying the gaps it addresses.

Chapter Three presents the methodology in full detail, describing every preprocessing decision, model configuration, and validation strategy.

Chapter Four presents the experimental results across all five modules with code extracts, visualizations, and interpretations, concluding with a cross-task synthesis and a reflection on the practical implications of the findings.

Conclusion Recaps the research objective, summarizes the findings, acknowledges the limitations, and suggests future works perspectives.

CHAPTER 1: FINANCIAL MARKETS: BASIC CONCEPTS, MAIN TASKS

1.1. Introduction:

This chapter begins with an overview of financial markets and the main characteristics of the financial data. It presents the different types of financial prediction tasks addressed in this study, including stock price and trend prediction, sales classification, credit risk analysis, and sentiment analysis. Next, it outlines the fundamental concepts of machine learning and reviews the most commonly used models in financial applications. It then explores deep learning techniques, with a particular focus on neural networks, recurrent models and transformers. Finally, this chapter highlights the main challenges associated with financial prediction domain.

1.2. Financial Markets:

Financial markets are organized systems where financial assets such as stocks, bonds, commodities, currencies, and other instruments are traded, and where financial resources are allocated between different economic actors and investors. These markets play a crucial role in capital formation, resource allocation, price discovery, and economic development. [6]

1.2.1. Types of Financial Markets:

Financial markets can be classified into several categories according to the nature of the assets traded:

- **Stock Market:** is a market in which company stocks (corporate securities) are bought and sold, including both securities listed on stock exchanges and those traded privately. It is often used synonymously with "stock exchange". [7]
- **Foreign Exchange Market (Forex):** The foreign exchange market is a global, decentralized market where national currencies are traded. It determines exchange rates between currencies and supports international trade and investment by enabling currency conversion, credit in foreign currencies, and hedging of exchange-rate risk. [8]
- **Commodity Market:** A commodity market is a market where raw materials and primary products such as agricultural commodities, energy products, and metals are traded through spot markets and derivative markets. These markets facilitate price discovery, risk transfer (hedging), and speculative activities .[9]

- **Cryptocurrency Market:** Cryptocurrency market refers to the digital asset market in which cryptocurrencies are traded through centralized and decentralized exchanges. These markets enable the exchange of cryptocurrencies against fiat currencies or other digital assets. The market is characterized by 24/7 trading, high volatility, blockchain-based infrastructure. and reliance on distributed ledger technology. [10]
- **Credit or Bond Market:** The bond (debt or credit) market is the market where participants issue, buy, and sell debt securities such as government bonds, corporate bonds, notes, and bills. Issuers (governments, corporations, financial institutions) raise long-term funds without giving up ownership, while investors receive periodic interest and principal at maturity. The bond market is a key pillar for risk management, financial stability, and economic forecasting, with the yield curve and credit spreads providing signals about economic conditions. [11]
- **The E-commerce and Retail Market:** while not a traditional financial market in the classical sense, but it generates rich transactional sales data that has become an important subject for ML-based forecasting. [12]

1.3. Characteristics of Financial Data:

Financial data are usually modelled as time series, but they show several special properties that make prediction difficult and justify using ML/DL instead of simple linear models [2]:

- **Time Series Nature:** A time series is a sequence of observations recorded over time at regular or irregular intervals (e.g., daily prices, intraday quotes) and depends on past values. Financial data are usually organized as time series, i.e., sequences of prices or returns observed over time, often displaying non-stationarity, non-linearity, volatility clustering and seasonal patterns .[13]
- **Non-linearity:** Relationships between variables are not always linear and involve complex interdependencies among assets and factors. Chaotic or highly complex dynamics have been reported in index data. [14]
- **Noise and Uncertainty:** Data contains randomness due to external influences and financial series have a lot of random fluctuations from microstructure effects, news, and external events, leading to noisy data and low signal-to-noise ratios, especially at high frequency. [2]
- **Volatility:** Prices and returns fluctuate strongly, periods of high volatility tend to be followed by high volatility, and calm periods by calm periods. Volatility is not constant over time, complicating risk management. [15]

- **Autocorrelation and long-range dependence:** Returns can exhibit short-term or longer-term dependence due to trends, cycles, and investor behavior. [13]
- **Sentiment Sensitivity:** Sentiment sensitivity refers to the degree to which asset prices and volatility respond to changes in investor sentiment extracted from news or social media. [16]

1.4. Financial Prediction Problems:

This study addresses four main prediction tasks that cover the most representative applications of ML in finance.

1.4.1. Stock Price and Trend Forecasting:

Stock price forecasting consists of predicting the future value or direction of a stock based on historical price data and other relevant features. It is one of the most studied problems in financial ML due to its direct economic implications. [1] Two subtasks are commonly distinguished: price regression, which aims to predict the exact future price, and trend classification, which aims to predict whether the price will go up or down. The main challenges include non-stationarity, high noise levels, and the influence of unpredictable external events Which presents particularly challenging **patterns** for machine learning prediction. Several technical chart patterns are commonly used by traders to identify potential trend continuations and reversals, as illustrated in **Figure 1.1**. [17] [18]

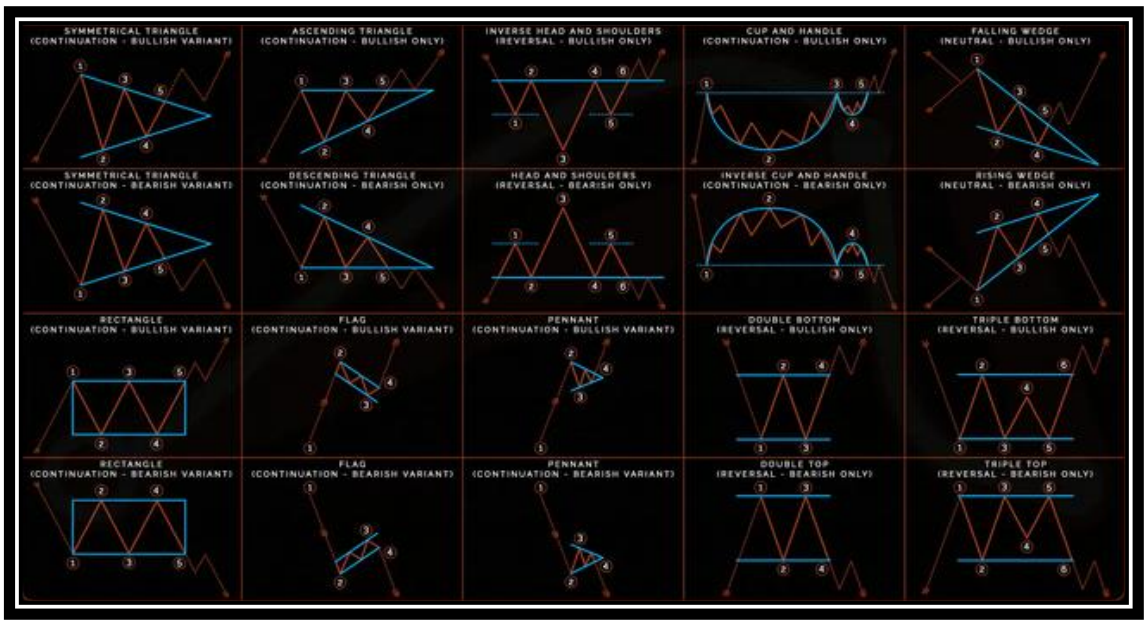


Fig. 1.1: Common Charts Patterns in Stock Markets Technical Analysis. [18]

1.4.2. Sales Prediction:

Sales forecasting encompasses two complementary prediction tasks that address different aspects of demand analysis.

The first is sales trend classification, which frames the problem as a discrete prediction task where the goal is to determine the future direction of sales performance typically categorized as High or Low sales rather than predicting an exact numerical value. This formulation is particularly useful for business decision-making scenarios where managers need to anticipate demand surges or downturns to adjust procurement and logistics strategies accordingly. [19]

The second is sales volume regression, which aims to predict the exact future quantity or revenue of a product based on historical sales data and relevant contextual features. It is a fundamental problem in business intelligence and supply chain management. [20]

1.4.3. Credit Risk Prediction:

Credit risk prediction is a classification task that aims to determine the probability that a borrower will default on a financial obligation. It is one of the oldest and most critical applications of ML in finance, widely used by banks and financial institutions. [21] The main challenge in this task is class imbalance, as defaulting cases are typically much rarer than non-defaulting ones, which can bias standard classifiers toward the majority class. [4]

1.4.4. Sentiment Analysis in Financial Markets:

Financial sentiment analysis consists of extracting and quantifying the sentiment expressed in textual sources such as news headlines, earnings reports, and social media posts, and using it as a signal for market prediction. [16] It combines natural language processing with financial analysis and has gained significant attention with the rise of large-scale textual data. The challenge lies in the domain-specific vocabulary of financial texts and the subtlety of expressions that carry market-moving information. [3]

1.5. Traditional Financial Prediction Methods:

Before the widespread adoption of machine learning, financial prediction and decision-support systems relied primarily on statistical, econometric, and rule-based approaches.

Time-series forecasting tasks, such as stock price prediction, were commonly addressed using models such as **ARIMA** (Auto-Regressive Integrated Moving Average) which is a classical time series model that combines autoregression, differencing to achieve stationarity,

and moving average components. It has been widely used for short-term financial forecasting. However, ARIMA assumes linearity and struggles with non-stationary, high-noise financial series. [13]

classification tasks including credit risk assessment and financial text analysis often relied on logistic regression, scorecards, expert-defined rules, and dictionary-based sentiment analysis. Although these methods are generally interpretable and computationally efficient, they often require strong assumptions regarding data distribution, linearity, or stationarity and have limited ability to capture complex nonlinear relationships.

1.6. Limitations of Traditional Models:

Despite their theoretical foundations, traditional statistical models present several important limitations when applied to modern financial data. [2]

First, they rely on strict assumptions such as linearity, normality of residuals, and stationarity that are rarely satisfied in real-world financial series.

Second, they are unable to capture complex non-linear patterns and long-range dependencies present in high-dimensional data.

Third, they do not naturally incorporate unstructured data sources such as news text or social media, which have become increasingly important signals in modern markets. [16]

Fourth, their predictive accuracy degrades significantly in periods of high volatility or structural breaks. [15]

Finally, traditional models are typically designed for a single prediction task and cannot be generalized across multiple financial applications simultaneously. [2] These limitations motivate the adoption of machine learning and deep learning approaches, which impose fewer assumptions and demonstrate greater flexibility across diverse financial tasks.

1.7. Machine Learning in Finance:

Machine learning refers to a family of algorithms that enable a system to learn patterns from data and make predictions or decisions without being explicitly programmed for each task. [22] In the context of financial prediction, ML methods are applied to both regression and classification tasks (predicting continuous values or discrete outcomes). Machine learning has become a key technology in modern financial systems, supporting tasks such as trading, risk management, investment analysis, and predictive modeling, as illustrated in **Figure 1.2**.

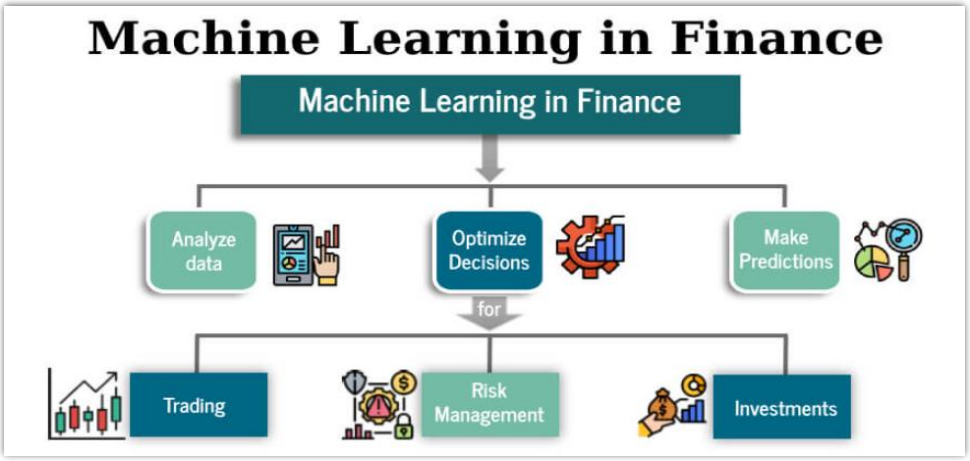


Fig. 1.2: Machine Learning in Finance: Use Cases & Challenges.

The most commonly used Machine Learning algorithms in finance include:

1.7.1. Random Forest:

Random Forest is an ensemble method that builds multiple decision trees on random subsets of the data and aggregates their predictions through averaging. [23] It is robust to overfitting, handles missing values well, and provides feature importance measures, making it particularly useful for credit risk and sales prediction.

1.7.2. Gradient Boosting:

Gradient Boosting methods such as XGBoost and LightGBM build trees sequentially [24], where each tree corrects the errors of the previous one. They consistently achieve high accuracy on structured tabular data and are widely used in financial classification tasks.

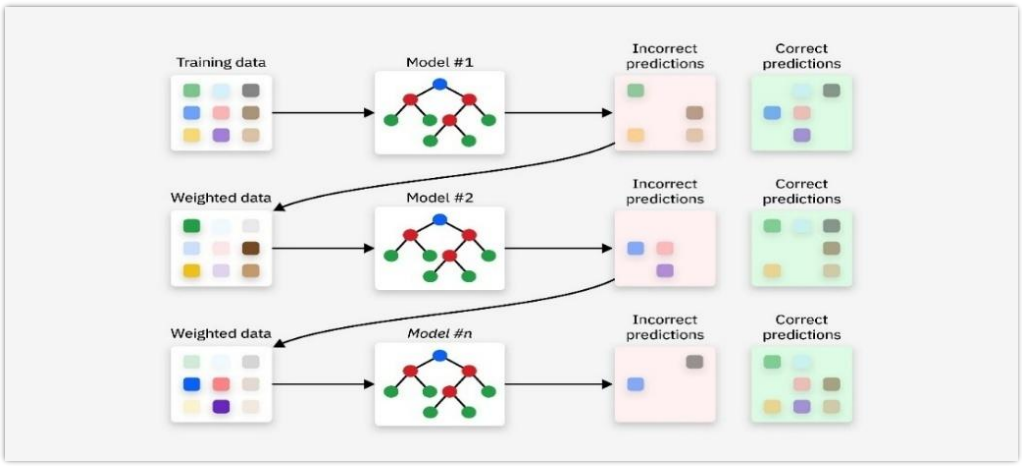


Fig. 1.3: Gradient Boosting Model. [25]

1.7.3. K-Nearest Neighbors (KNN):

Naive K-Nearest Neighbors (KNN) is a non-parametric machine learning algorithm that makes predictions based on the similarity between data points. For classification tasks, the class is determined by the majority vote of the *k* nearest neighbors, while for regression tasks, the prediction is obtained by averaging the values of the nearest neighbors. In finance, KNN has been applied to market trend classification (e.g., predicting stock price movement as upward or downward) as well as stock price forecasting. Its simplicity and effectiveness make it a useful baseline model, although its performance may decrease with high-dimensional datasets. [22]

1.7.4. Linear Regression:

Linear Regression is one of the most widely used supervised learning algorithms for modeling the relationship between a dependent variable and one or more independent variables. [22] In financial applications, it is commonly employed for regression tasks such as stock price prediction and return forecasting. Additionally, when combined with a threshold-based decision rule, it can support market trend classification by indicating whether the future price is expected to move upward or downward. Despite its simplicity and interpretability, its performance may be limited when financial data exhibit strong nonlinear patterns.

1.7.5. Support Vector Machines:

Support Vector Machines (SVM) find the optimal hyperplane that separates classes with maximum margin. [26] They perform well in high-dimensional spaces and have been successfully applied to both financial classification and regression tasks. However, they become computationally expensive on large datasets and require careful kernel selection.

1.8. Deep Learning in Finance:

Deep learning extends machine learning by using multi-layered neural networks capable of learning hierarchical representations directly from raw data. [27] Its ability to model complex non-linear relationships and sequential dependencies has made it particularly attractive for financial applications.

1.8.1. Recurrent Neural Networks (RNN):

Recurrent Neural Networks are specifically designed for sequential data. They maintain a hidden state that carries information from previous time steps, making them naturally suited for

time series forecasting. [28] However, standard RNNs suffer from the vanishing gradient problem, which limits their ability to capture long-range dependencies in financial time series.

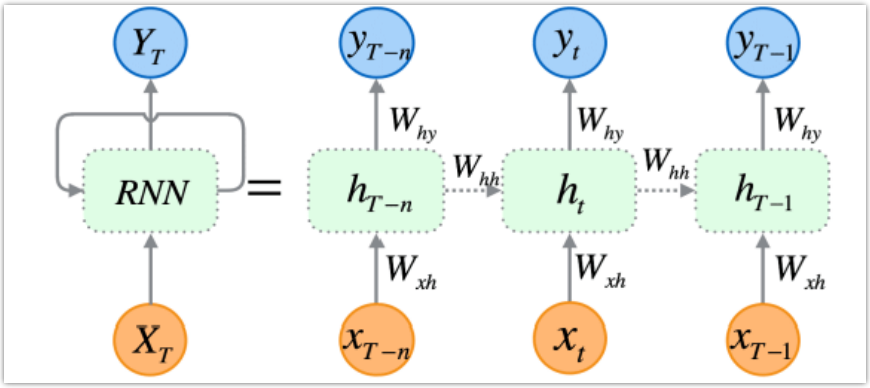


Fig. 1.4: A recurrent neural network (RNN) Architectures. [29]

1.8.2. Long Short-Term Memory Networks (LSTM):

Long Short-Term Memory networks address the vanishing gradient limitation through a gating mechanism consisting of forget, input, and output gates that control the flow of information across time steps. [30] LSTM have demonstrated strong performance in stock price forecasting, sales prediction, and other financial time series tasks due to their ability to retain relevant long-term information. [17]

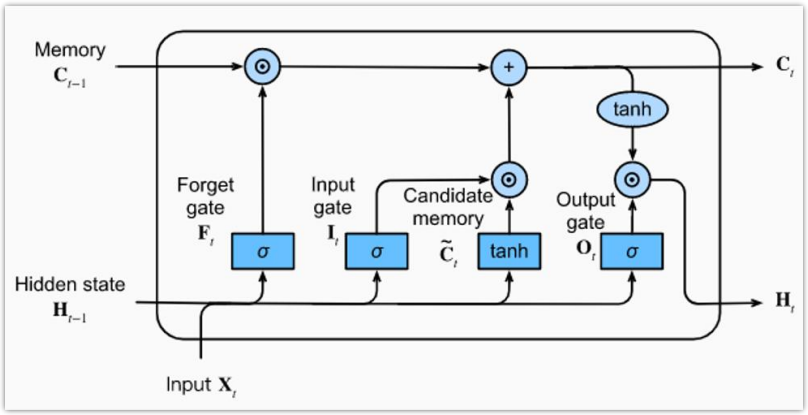


Fig. 1.5: Long Short-Term Memory (LSTM) Networks Architectures. [31]

1.8.3. Bidirectional Long Short-Term Memory Networks (BiLSTM):

Bidirectional Long Short-Term Memory (BiLSTM) networks extend the standard LSTM architecture by processing sequential data in both forward and backward directions. This bidirectional structure enables the model to capture contextual information from both past and future tokens, leading to a more comprehensive understanding of textual data. [32]

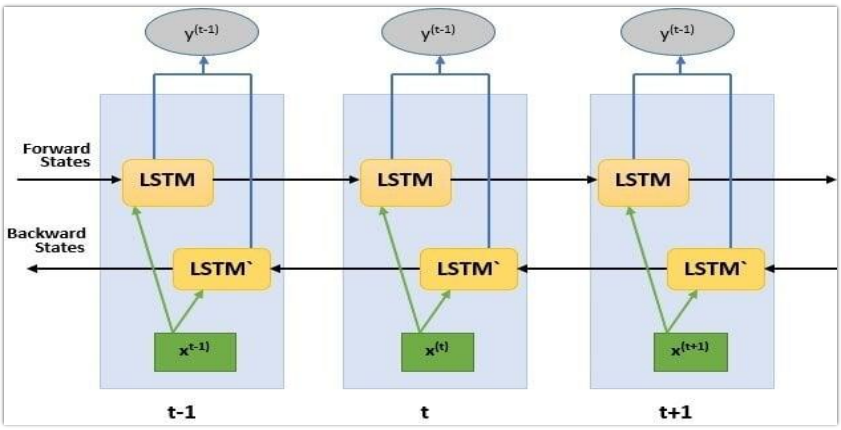


Fig. 1.6: Bidirectional Long Short-Term Memory Networks (BiLSTM**)** Architectures. [33]

In financial sentiment analysis BiLSTM has been widely used to analyze financial news, reports, and social media content, improving sentiment classification performance by exploiting contextual dependencies within text sequences. [34]

1.8.4. Gated Recurrent Units (GRU):

The Gated Recurrent Unit is a simplified alternative to LSTM introduced by Cho et al. (2014). [35] It uses two gates update and reset instead of three, resulting in fewer parameters and faster training while maintaining competitive performance with LSTM on most financial forecasting tasks.

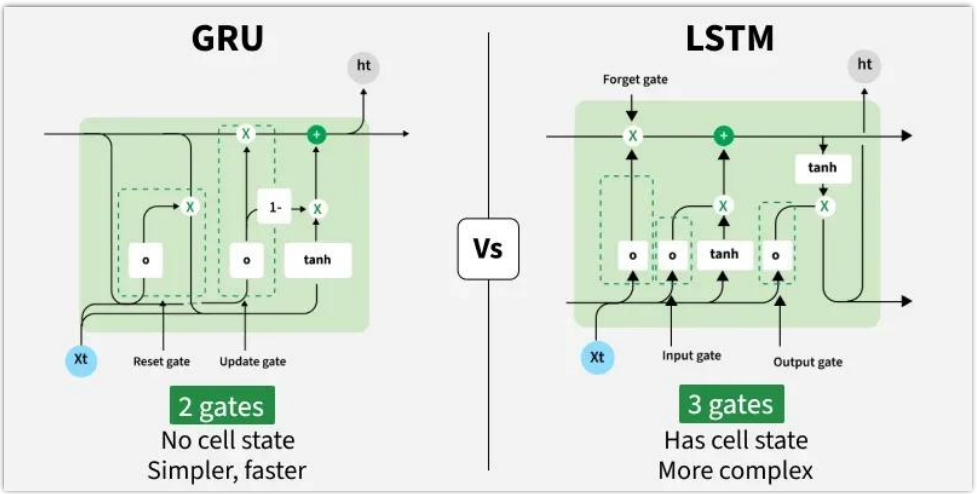


Fig. 1.7: Gated Recurrent Unit Networks (GRU) Architectures. [36]

1.8.5. Deep Neural Network (DNN):

Deep Neural Networks, also referred to as Multilayer Perceptrons (MLP) with multiple hidden layers, extend the basic ANN architecture by stacking several hidden layers between

the input and output layers. This depth allows DNNs to learn hierarchical representations of the input data, where lower layers capture simple low-level features and deeper layers capture increasingly abstract and complex patterns. [27]

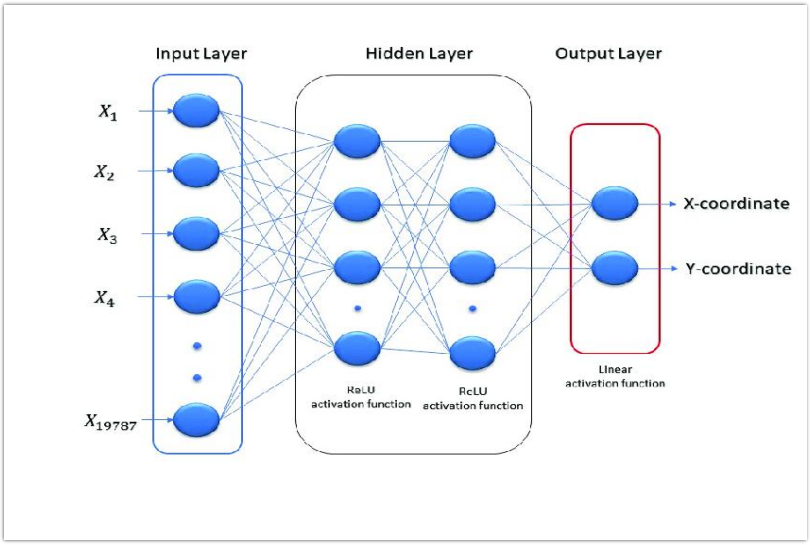


Fig. 1.8: Deep Neural Networks (DNN) Architectures. [37]

A DNN consists of an input layer that receives the feature vector, L hidden layers each applying a linear transformation followed by a non-linear activation function, and an output layer that produces the final prediction.

1.8.6. Transformer-Based Models and FinBERT:

The Transformer architecture, introduced by Vaswani et al. (2017) [38], relies on a self-attention mechanism that allows the model to weigh the importance of different positions in a sequence simultaneously, without the sequential processing constraint of RNNs. This makes Transformers highly effective for natural language understanding tasks.

FinBERT is a domain-specific variant of BERT (Bidirectional Encoder Representations from Transformers) pre-trained on large financial text corpora including financial news, earnings reports, and analyst notes. [39] Unlike general-purpose sentiment models such as VADER or TextBlob, FinBERT understands financial-specific language and has demonstrated superior performance in financial sentiment classification tasks particularly on the Financial PhraseBank benchmark. [3]

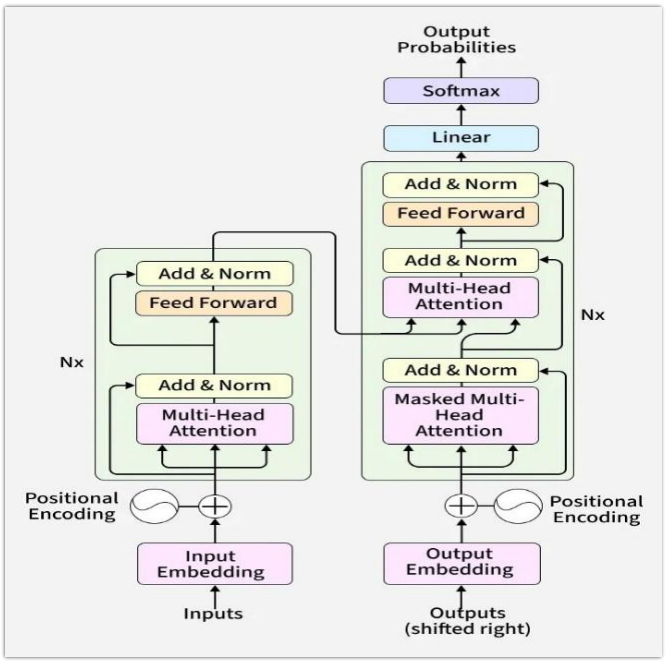


Fig. 1.9: Transformers Architectures [38].

1.9. Natural Language Processing (NLP):

Natural Language Processing (NLP) is a branch of artificial intelligence concerned with enabling computers to understand, analyze, and generate human language. [40] In finance, NLP is widely used to analyze financial news, earnings reports, social media content, and market sentiment. By transforming textual information into structured representations, NLP techniques help machine learning and deep learning models capture investor sentiment and extract valuable signals that may influence stock prices and market behavior. Recent advances in deep learning, particularly recurrent neural networks and transformer-based models, have significantly improved the performance of financial sentiment analysis systems.

1.10. Conclusion:

This chapter provided the theoretical foundation necessary for understanding the work presented in this thesis. It introduced the structure and characteristics of financial markets, described the five prediction tasks addressed, reviewed traditional statistical approaches and their limitations, and presented the ML and DL methods employed in this study. The following chapter will present a detailed state of the art focusing on the architectures, algorithms, and comparative analysis of related works, leading to the presentation of the proposed solution.

CHAPTER 2: LITERATURE REVIEW, CHALLENGES AND RESEARCH MOTIVATIONS

2.1. Introduction:

The field of finance has changed significantly over the past decades and this change has come from the fast growth of digital technologies, the globalization of financial markets, and the increasing availability of large-scale data. This data shows complex interactions between economic factors, investor behavior, and outside events, which makes predicting financial trends quite difficult. This chapter reviews the theoretical and practical foundations of machine learning in finance. It begins by introducing the evolution from artificial intelligence to machine learning and its major learning paradigms. It then discusses the strengths and weaknesses of classical machine learning algorithms and deep learning architectures. Furthermore, the chapter examines related studies and highlights the main limitations and challenges. Finally, it presents the motivations that guided the development of the current work.

2.2. From Artificial Intelligence to Machine Learning:

The term artificial intelligence was coined at the Dartmouth Conference of 1956 to describe the broad scientific ambition of building machines capable of performing tasks that would require intelligence if performed by a human. [41] Over the following seven decades, the field evolved through alternating periods of optimism and stagnation the so-called AI winters before arriving at its current state, in which practical AI systems are embedded in almost every domain of human activity. [27]

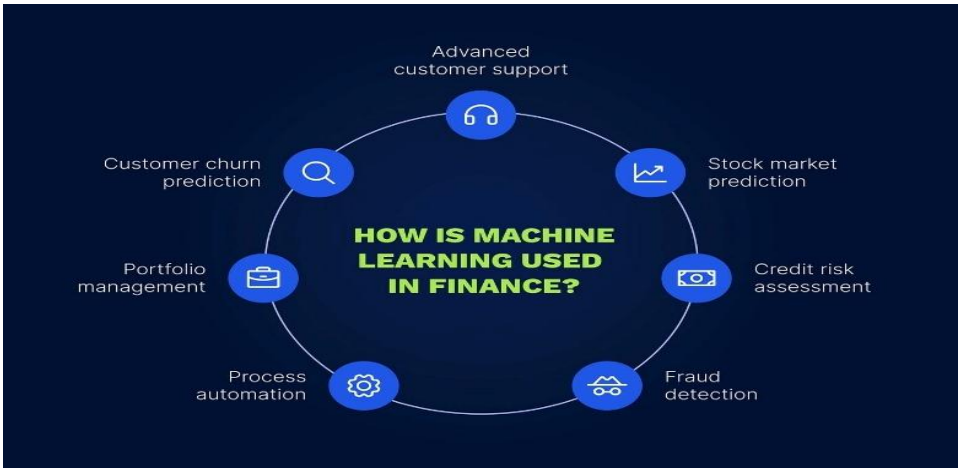


Fig. 2.1: Major Financial Applications of Machine Learning. [42]

As shown in **Figure 2.1**, machine learning has become an essential tool in several financial applications, ranging from risk assessment to stock market forecasting.

Modern AI is typically organized into three major learning paradigms, distinguished by the nature of the feedback available to the learning system:

2.2.1. Supervised learning in Finance:

Supervised learning is one of the most widely used paradigms in financial artificial intelligence applications and in this approach, models are trained on labeled historical data in order to learn the relationship between input variables and target outputs. [22] Financial supervised learning tasks generally include:

- **regression** problems such as stock price forecasting.
- **classification** problems such as credit risk assessment, stock trend prediction, fraud detection, and sentiment analysis. [2]

Financial datasets often contain noisy, nonlinear, and highly dynamic patterns, making supervised learning particularly suitable for extracting predictive relationships from large-scale market data. [2] Classical machine learning algorithms and deep learning architectures have both demonstrated strong performance in financial prediction tasks depending on the structure and complexity of the data. [1]

2.2.2. Unsupervised Learning in Finance:

While supervised learning dominates financial prediction tasks, **unsupervised learning** plays an increasingly important supporting role both as a standalone tool for financial analysis and as a preprocessing step that improves supervised model performance [27]:

- **Clustering for market segmentation:** K-Means and hierarchical clustering have been applied to equity markets to identify groups of stocks with similar return co-movements, enabling sector-level portfolio construction that goes beyond traditional GICS classifications. [43] Self-Organizing Maps (SOMs) have been used to visualize the topological structure of fixed-income markets, providing intuitive representations of bond similarity that portfolio managers can act upon without requiring ML expertise. [44]
- **Anomaly detection in transaction data:** Isolation Forest and Autoencoder-based anomaly detection are the standard tools for credit card fraud detection a task that is closely related to the credit risk assessment module. [45] These methods learn the distribution of normal transactions and flag observations that require an anomalously large number of partitions (Isolation Forest) or that reconstruct poorly from a

compressed bottleneck representation (Autoencoder). The advantage of unsupervised approaches here is that fraud patterns evolve continuously a supervised model trained on past fraud patterns may not detect novel attack vectors, whereas an anomaly detector flags anything that deviates from established normal behavior.

- **Topic modeling for financial text:** Latent Dirichlet Allocation (LDA) extracts thematic structure from large financial text corpora earnings call transcripts, SEC filings, analyst reports without requiring manual annotation. [46] The discovered topics (e.g., one cluster dominated by terms like "revenue", "growth", "guidance" and another by "restructuring", "impairment", "write-down") provide interpretable signals that can be used as features in downstream supervised tasks.

2.2.3. Reinforcement Learning in Finance:

Reinforcement learning (RL) occupies a distinct position in financial ML: it is the only paradigm that naturally accommodates the sequential, adaptive nature of financial decision-making. [47] A human portfolio manager does not make a single prediction and walk away they observe market conditions, take an action (buy, sell, hold), receive a reward (profit or loss), update their belief about the market, and repeat. This is precisely the Markov Decision Process that RL is designed to solve:

- **Algorithmic trading with deep Q-networks:** The Deep Q-Network (DQN) architecture, originally developed for Atari game playing [48], has been adapted for single-asset trading tasks where the agent learns a policy mapping market states (price history, technical indicators) to discrete actions (buy, sell, hold). The reward signal is the realized profit and loss of each action. A key challenge in financial RL is the non-stationarity of the reward distribution a policy that achieves strong rewards in a bull market may perform catastrophically in a bear market [47], and the exploration-exploitation tradeoff is harder to calibrate than in deterministic game environments.
- **Portfolio optimization with policy gradient methods**: Actor-critic architectures and Proximal Policy Optimization (PPO) have been applied to continuous portfolio allocation problems learning a policy that outputs asset weight vectors rather than discrete trading signals. [49] The advantage over traditional mean-variance optimization is that RL agents can learn to incorporate transaction costs, liquidity constraints, and dynamic risk preferences into their allocation strategy without requiring explicit mathematical formulation of these constraints.

2.3. Algorithms: Strength and Weakness in Supervised learning

2.3.1. Classical Machine Learning Algorithms:

This analysis looks at each ML algorithm used in this financial prediction due to their interpretability, computational efficiency, and strong performance on structured financial data. [22] However, each algorithm presents different strengths and limitations depending on the characteristics of the financial task, such as temporal dependency, class imbalance, or high-dimensional feature spaces. **Table 2.1** summarizes the strengths, weaknesses, and potential applications of each algorithm.

Table 2.1: Critical Comparison of ML Algorithms.

Algorithm	Strength	Weakness	Best Task in This Work
Logistic Regression	Interpretable, fast, calibrated	Linear boundaries only	Credit Risk, Sentiment
Random Forest	Robust, handles high-dim., feat. importance	No temporal memory	All five tasks (baseline)
SVM	High-dim. effective, kernel flexibility	Slow on large datasets	Sentiment (TF-IDF)
KNN	No assumptions, local patterns	Slow, dim. sensitive	Trend, Sales (small data)
Naïve Bayes	Very fast, sparse text effective	Independence assumption	Sentiment Analysis
XGBOOST	Top accuracy on tabular data, handles imbalance, fast with hist method	Black-box, requires careful hyperparameter tuning, can overfit without early stopping	Credit Risk, Sales

2.3.2. Deep Learning Architectures:

The following DL architectures are employed based on their established performance in financial sequential and NLP tasks. Their key mechanisms, handling of temporal order, and primary application domains are summarized in **Table 2.2**:

Table 2.2: Critical Comparison of DL Architectures.

Architecture	Key Mechanism	Temporal Order	Best Suited For	Used In
DNN	Fully connected layers + ReLU	No (flat input)	Tabular, classification	All tasks
LSTM	3-gate memory cell	Yes (step-by-step)	Long time series, sequences	Price, Trend, Sales, Credit
GRU	2-gate simplified LSTM	Yes (step-by-step)	Sequences, small datasets	Price prediction
BiLSTM	Bidirectional LSTM	Yes (forward + backward)	NLP, sentiment analysis	Sentiment
FinBERT	Self-attention + pre-training	Yes (parallel)	Financial NLP	Sentiment Analysis

2.4. Studies and Related Works:

2.4.1. Stock Price and Trend Prediction:

The problem of predicting stock prices and market trends has attracted intensive research over the past two decades. Early work relied on classical statistical models such as ARIMA and linear regression, which assume stationarity and linearity assumptions that the non-stationary, nonlinear nature of equity markets routinely violates. [13] The field was transformed by the demonstration by **Fischer and Krauss** [1] that LSTM networks, trained on S&P 500 historical price data, achieve statistically significant out-of-sample returns, establishing deep learning as a competitive approach to financial time series forecasting. However, this seminal study was limited to a single index and a single task (return prediction), without addressing trend classification, feature engineering from technical indicators, or generalization across assets.

Sezer et al. [2] reviewed over 150 studies and concluded that LSTM and CNN-based architectures consistently outperform classical ML models across different forecasting horizons. Crucially, their survey identified two recurring failure modes: data leakage from improper train-test splitting that inflates reported accuracy, and overfitting to historical regimes that fails to generalize to structural breaks such as market crashes.

Jiang further [17] demonstrated that ensemble methods particularly Random Forest combined with carefully engineered technical indicators such as RSI, MACD, and Bollinger Bands achieve competitive performance on short-horizon trend classification, while hybrid CNN-LSTM architectures tend to lead on longer horizons.

Despite these advances, a critical limitation persists across the stock prediction literature: most studies treat price regression and trend classification as separate, independent problems, and evaluate a small number of models (typically two or three) on a single stock or index. No study identified in this review simultaneously addresses both subtasks across multiple stocks, with a comprehensive comparison of five or more ML models and three DL architectures, under a rigorous temporal evaluation protocol.

2.4.2. Sales Prediction and Demand Forecasting:

Sales forecasting has been studied primarily in the operations research and supply chain literature, where ARIMA and exponential smoothing dominated for decades before ML approaches emerged. [19] **Bandara et al.** [20] demonstrated that globally trained recurrent neural networks in particular LSTM-based architectures outperform classical statistical methods such as ARIMA and exponential smoothing when trained across large databases of related time series, as validated on the M4 competition benchmark. Their clustering approach, which groups similar series before model training, is particularly relevant for retail datasets with diverse seasonal patterns. However, their framework targets aggregated time series regression, and does not address order-level classification tasks.

Bojer and Meldgaard [50] reviewed several Kaggle forecasting competitions involving business and retail demand forecasting and reported that gradient boosted decision trees and neural network models consistently achieved strong forecasting performance on datasets characterized by high intermittency and rich external information. Their findings also highlighted the importance of feature engineering, exogenous variables, and business hierarchy information in improving forecasting accuracy.

A key observation from this literature is that the framing of the sales prediction problem significantly impacts the choice of methodology. When sales volume is predicted as a continuous variable (regression), models must handle heavy right-skewed distributions and outliers driven by promotional events. When the problem is reframed as binary classification (high vs. low sales), models can exploit class-specific thresholds and achieve higher practical

accuracy for decision-support applications. This thesis adopts the classification framing on the Amazon dataset, which better aligns with the inventory management use case.

A significant gap in the existing literature is limited research has explored the application of deep sequential models such as LSTM to e-commerce order-level classification tasks, where the temporal structure is implicit in order timestamps rather than explicit in a continuous price series.

2.4.3. Credit Risk Assessment:

Credit scoring is one of the oldest applications of statistical modeling in finance, with logistic regression serving as the industry standard for decades due to its interpretability and regulatory acceptance. [21] The landmark benchmark study by **Lessmann et al**. [21] compared 41 classification methods across eight real-world credit datasets and established that ensemble methods particularly Random Forest and gradient boosting consistently outperform logistic regression, with AUC-ROC identified as the most appropriate evaluation metric given the severe class imbalance typical of lending datasets (defaulters constitute 5-20% of cases). Subsequent research has reported similar results, highlighting the effectiveness of ensemble learning methods for credit risk assessment. [5]

Kvamme et al [51] investigated the application of deep learning techniques to mortgage default prediction and demonstrated that convolutional neural networks can effectively capture complex relationships in borrower and loan characteristics. Their results showed that deep learning models provide competitive predictive performance for credit risk assessment, highlighting the potential of neural networks in mortgage default modeling. **Schmitt** [52] extended this comparison to include modern deep learning architectures, reporting the dominance of ensemble methods on tabular credit data but noting that DNN models close the gap when borrower behavioral time series are available.

A persistent challenge identified across all credit risk studies is the treatment of class imbalance. While most studies acknowledge this problem, the solutions adopted vary widely oversampling (SMOTE), undersampling, class-weighted loss functions, or threshold calibration making cross-study comparisons difficult. This thesis standardizes this treatment using class-weighted training across all models, enabling a fair comparison.

2.4.4. Financial Sentiment Analysis:

The application of NLP to financial text analysis has evolved through four distinct generations of methods. The first generation relied on rule-based lexicons: **Loughran and McDonald** [53] demonstrated that financial-domain lexicons significantly outperform general-purpose sentiment dictionaries (such as the Harvard GI) on SEC filings, as common words like 'liability' or 'risk' carry neutral or positive connotations in financial contexts. The second generation applied classical ML classifiers (Naïve Bayes, SVM, Logistic Regression) to TF-IDF feature representations of financial text. **Malo et al** [3] evaluated sentiment classification methods on the Financial PhraseBank benchmark dataset and demonstrated that SVM-based models trained on domain-specific financial text outperform general-purpose sentiment analysis tools, highlighting the importance of domain adaptation in financial NLP.

The third generation introduced recurrent neural networks. **Huang and Kou** [34] compared BiLSTM-based and transformer-based architectures for financial sentiment analysis and reported that BiLSTM models remain competitive on medium-sized financial text datasets, while being significantly faster to train and deploy. The fourth generation is dominated by pre-trained transformers: **Araci** introduced FinBERT by fine-tuning BERT on financial corpora, achieving state-of-the-art performance on financial sentiment classification tasks. **Yang et al** [54] proposed a domain-specific FinBERT model pretrained on a large-scale financial communication corpus, demonstrating improved performance over generic BERT models on multiple financial NLP tasks.

A critical limitation of transformer-based approaches is their computational cost and dependency on large pre-training corpora resources that are often unavailable in academic or small-scale deployment settings. Moreover, **Bollen et al** [16] demonstrated that even simple sentiment proxies from social media exhibit significant correlations with market movements, suggesting that computationally affordable models can still capture economically relevant sentiment signals. This motivates the inclusion of BiLSTM, SVM, RF, NB, and LR in our sentiment module, enabling a comparison that spans the full spectrum from classical to modern approaches.

2.5. Models: Comparative Analysis Across Studies

2.5.1. Stock Price and Trend Forecasting:

The following table (Table 2.3) summarizes the most relevant studies in stock price and trend prediction, highlighting the datasets, models, obtained results, and the main limitations identified in previous work:

Table 2.3: Summary of Related Works in Stock Price and Trend Forecasting.

Authors	Year	Task	Dataset	Models	Best Result	Limitation
Fischer & Krauss [1]	2018	Return pred.	S&P 500	LSTM, RF, DNN	LSTM > RF	Single index, 1 task
Sezer et al [2]	2020	Survey	150+ studies	LSTM, CNN	DL > classical	No primary experiment
Jiang [17]	2021	Trend class.	Multiple	RF, CNN-LSTM	Hybrid CNN-LSTM	No DL models comparison

2.5.2. Sales Prediction:

The following comparative analysis in (Table 2.4) presents major studies related to sales prediction and demand forecasting, with emphasis on the adopted models and their reported performance:

Table 2.4: Summary of Related Works in Sales and Demand Forecasting.

Authors	Year	Dataset	Models	Metric	Best Result	Limitation
Bandara et al [20]	2020	Multiple time-series datasets	LSTM, RNN	SMAPE	RNN-based models showed strong forecasting performance	Aggregated, not order-level
Bojer & Meldgaard [50]	2021	Kaggle forecasting competitions (retail/business datasets)	XGBoost, GBDT, Neural Networks	Competition-dependent	Gradient boosting methods frequently ranked among top performers	Review paper, not a direct Amazon study.

2.5.3. Credit Risk Assessment:

The following table (Table 2.5) compares important studies in credit risk assessment and summarizes the strengths and limitations of the proposed approaches:

Table 2.5: Summary of Related Works in credit Risk Assessment.

Authors	Year	Dataset	Models	Metric	Best Result	Limitation
Lessmann et al [21]	2015	8 credit datasets	41 classifiers	AUC-ROC	RF/GBM best	No DL, no NLP features
Kvamme et al [51]	2018	Mortgage data	CNN and traditional ML models	AUC	XGBoost AUC best	Single-domain dataset
Schmitt [52]	2022	LendingClub	DL, GBM, RF, LR	AUC, F1	GBM ≈ DL	No deployment evaluation

2.5.4. Financial Sentiment Analysis:

The following table (Table 2.6) summarizes previous work in financial sentiment analysis, including both classical machine learning approaches and modern deep learning architectures:

Table 2.6: Summary of Related Works in Financial Sentiment Analysis.

Authors	Year	Dataset	Models	Accuracy	Limitation
Malo et al [3]	2014	Financial PhraseBank	NB, SVM	72%	No DL, no pre-training
Bollen et al. [16]	2011	Twitter, DJIA	Lexicon (POMS)	86%*	Correlation only, no classification.
Araci [39]	2019	Financial PhraseBank	FinBERT	97%	High compute, no deployment
Huang & Kou [34]	2023	Financial PhraseBank	BiLSTM, BERT	78%	No classical comparison

2.6. Challenges of Previous studies:

The review conducted in this chapter reveals five recurring challenges across financial ML literature:

- **Data leakage:** Random train-test splitting in temporal series artificially inflates reported accuracy. [2]
- **Task isolation:** Most studies address a single financial task, preventing cross-domain generalization insights.
- **Limited model coverage:** Most benchmarks compare 2 or 4 models, insufficient for robust conclusions.
- **Class imbalance neglect:** Inconsistent treatment of imbalanced classes (credit risk, sentiment) makes cross-study comparison unreliable. [21] [3]
- **Absence of deployment:** results remain static academic benchmarks with no practical interface for financial practitioners.

2.7. Motivation for Current Research:

This research aims to provide a systematic comparison between several classical machine learning algorithms and deep learning architectures across stock prediction, trend forecasting, sales prediction, credit risk assessment, and financial sentiment analysis. Unlike many existing works, the proposed methodology and Platform combines predictive modeling, comparative evaluation, interpretability, and practical deployment through an interactive application.

The limitations identified in previous studies motivate the development of a more comprehensive financial framework capable of addressing multiple financial tasks within a unified environment.

2.8. Conclusion:

This chapter reviewed the main artificial intelligence approaches used in financial applications, including both classical machine learning algorithms and deep learning architectures. A comparative analysis of previous studies was conducted across multiple financial tasks in order to identify the strengths, limitations, and research gaps in the existing literature. The analysis highlighted the need for a more comprehensive and systematically evaluated framework capable of integrating multiple financial prediction tasks using both ML and DL approaches. The next chapter will present the proposed methodology, including the selected datasets, preprocessing techniques, model architectures, and training protocols adopted in this work.

CHAPTER 3: DATASETS, WORK ENVIRONMENT AND SELECTED APPROACHES

3.1. Introduction:

This chapter presents the proposed methodology and experimental setup adopted in this work. It describes the complete workflow followed to develop and evaluate machine learning and deep learning models. It describes every design decision taken between the raw data and the final trained model, which datasets were selected and why, how data is preprocessed and features are engineered for each financial task, which ML models and DL architectures are employed and how they are configured, what evaluation metrics and validation strategies are used and how the software stack and computing environment used to implement the proposed framework and support its integration into the Finsight platform.

3.2. Overview of the Proposed Methodology:

This methodology proposes a multi-module financial platform based on both machine learning and deep learning techniques to address multiple financial prediction tasks within a unified environment. The methodology focuses primarily on the training, evaluation and testing of predictive algorithms, the graphical interface is mainly used to visualize and test the developed models:

The application consists of the following main modules:

- Financial Sales Prediction.
- Financial Sentiment Analysis.
- Credit Risk Assessment.
- Stock Trend Prediction.
- Stock Price Forecasting.

The Platform was developed using Python and Streamlit to provide an interactive and user-friendly web application.

The following figure illustrates a general workflow commonly adopted in machine learning and deep learning predictive modeling. This workflow serves as a conceptual foundation for the methodology implemented in this study.

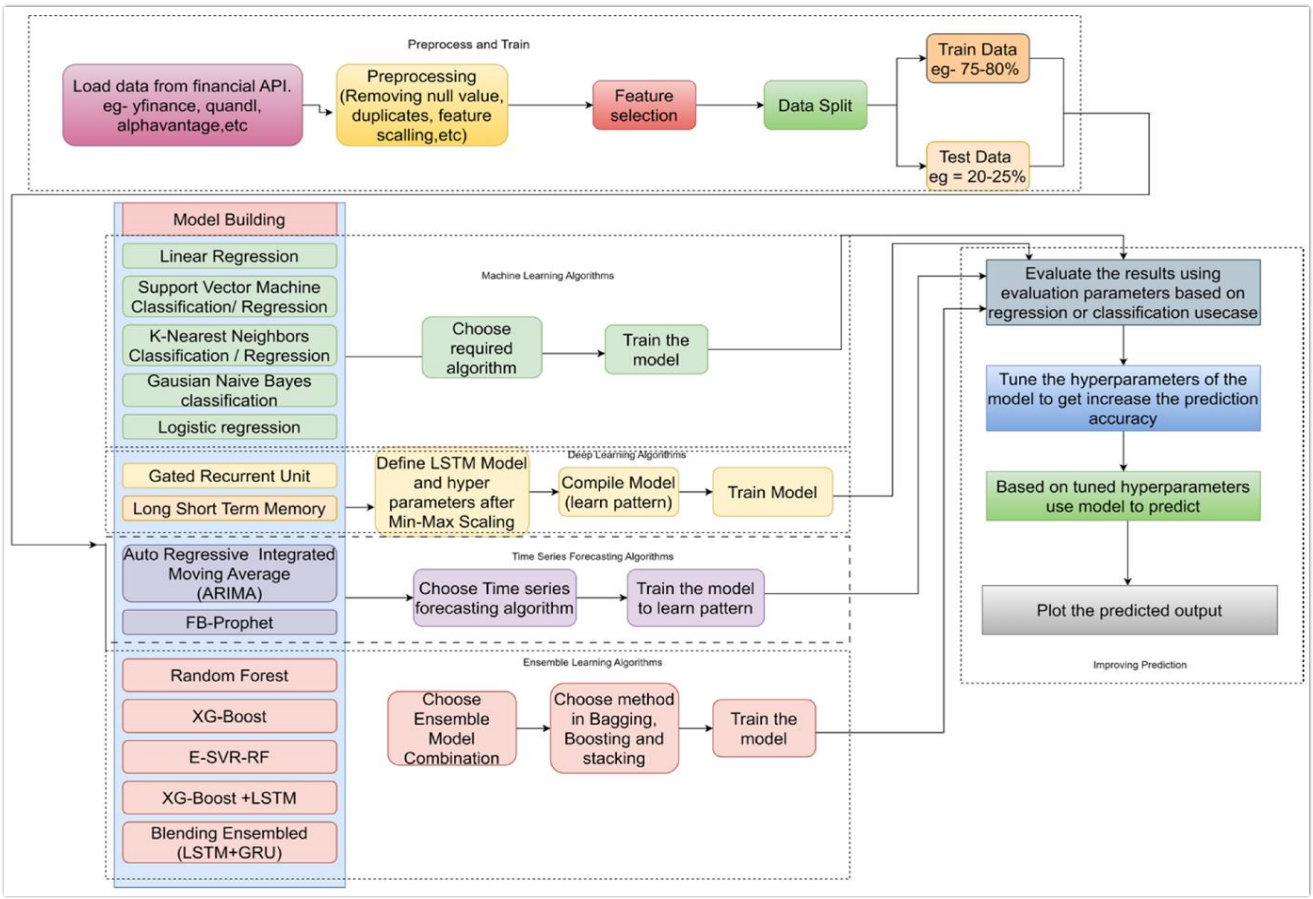


Fig. 3.1: General Workflow of Machine Learning and Deep Learning model. [55]

3.3. Used Datasets:

Five datasets are employed in this study, each corresponding to one of the financial prediction tasks. The selection criteria were: public availability and reproducibility, sufficient size for meaningful evaluation, direct relevance to the financial task, and coverage of multiple market regimes to test model generalization.

3.3.1. Yahoo Finance & Equity Price Data:

The equity price data is collected from Yahoo Finance, which is a financial platform via the **yfinance** Python library, which provides open-source access to historical OHLCV (Open, High, Low, Close, Volume) data at daily frequency. [56] (AAPL) is selected for the period January 2010 to January 2024, yielding 3,522 trading days. This period spans four distinct market regimes:

the 2010–2019 bull market, the COVID-19 crash and V-shape recovery (2020), the Federal Reserve rate-hike bear market (2022), and the 2023–2024 recovery. Regime diversity is intentional the 20% test set contains structurally different conditions from the

80% training set, providing a genuine out-of-sample evaluation that prevents regime-specific overfitting from going undetected. [1]

The same AAPL dataset is used for both the **price regression** and the **trend classification** modules, with different feature engineering and target construction applied in each case.

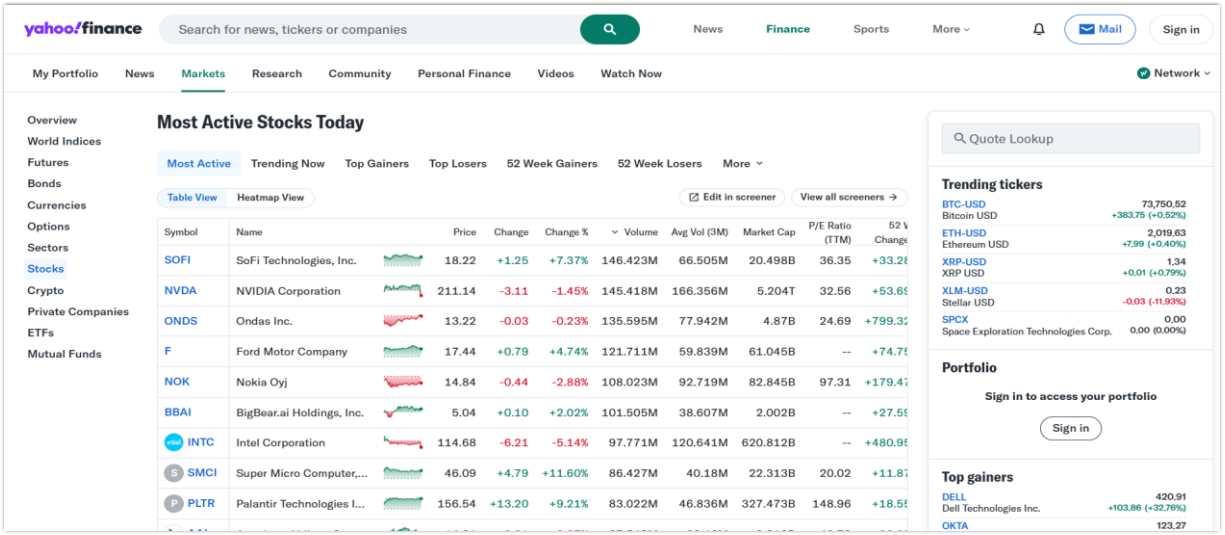


Fig. 3.2: Yahoo Finance Website. [56]

Table 3.1: Yahoo Finance Dataset Summary.

Parameter	AAPL
Period	Jan 2010 to Jan 2024
Frequency	Daily
Observations	Approximately 3,500
Raw Features	Open, High, Low, Close, Volume
Engineered Features	27 technical + regime features in Trend
Tasks	Price Regression and Trend Classification

3.3.2. Amazon Sales Dataset:

The sales prediction module uses a publicly available **Amazon** order dataset consisting of 10,000 order-level records with 11 features: unit price, quantity, discount, shipping cost, product category, sub-category, brand, country, payment method, order status, order date.

The target variable is constructed by computing total revenue per order and applying a median-split binarization: orders above the median are labeled **HIGH** and below are labeled **LOW**. This binary classification framing is chosen because it directly supports inventory management decisions, produces a balanced class distribution (50/50), and avoids the complications of heavy right-skewed revenue distributions. [19]

Feature engineering includes temporal decomposition (month, day of week, delivery time), revenue proxy construction (**net_revenue = effective_price × quantity − shipping_cost**), and label encoding of categorical variables.

3.3.3. Lending Club Credit Risk Dataset:

LendingClub, a financial institution listed on the New York Stock Exchange (NYSE: LC), operates as a digital marketplace bank that integrates traditional banking with capital market activities. [57] Unlike conventional peer-to-peer platforms, LendingClub holds a federal bank charter (since its acquisition of Radius Bank in 2021) and actively participates in financial markets by securitizing loans and selling them to institutional investors such as hedge funds and insurance companies. Its publicly traded equity and structured certificate programs illustrate its deep connection to capital markets.

Fig. 3.3: Lending Club Marketplace. [57]

The credit risk module uses a subset of the **Lending Club** peer-to-peer lending dataset containing approximately 80,000 loan records from 2007 to 2018. The binary target distinguishes between Fully Paid loans (majority, ~80%) and Charged Off loans (minority, ~20%). Features include borrower characteristics (annual income, employment length, home ownership, FICO score, debt-to-income ratio) and loan characteristics (amount, interest rate, grade, purpose, term). Missing values are imputed using median imputation for numerical features and mode imputation for categorical ones.

The primary challenge is class imbalance: standard classifiers trained without correction overwhelmingly predict the majority class. This study addresses imbalance through class-

weighted loss functions applied uniformly across all models, enabling a fair comparison between ML and DL approaches under identical data conditions. [5]

3.3.4. Financial PhraseBank Sentiment Dataset:

The sentiment analysis module uses the Financial **PhraseBank** dataset [3], consisting of **4,846** sentences extracted from English-language financial news and annotated by domain experts with three labels: Positive, Neutral, and Negative, from the perspective of a financial market investor. This is the most widely used benchmark in financial NLP.

The class distribution is inherently imbalanced: Neutral (59%), Positive (28%), Negative (13%), reflecting the measured, factual tone of financial reporting. Class-weighted training is applied to prevent the Neutral class from dominating predictions. All the used datasets are summarized in (Table 3.2).

Table 3.2: Summary of All Datasets.

Module	Dataset	Size	Task Type	Target	Class Balance
Price Prediction	Yahoo Finance (AAPL)	3,500 days	Regression	Close price	/
Stock Trend	Yahoo Finance (AAPL)	3,773 days	Binary Classification	UP / DOWN	34% / 66%
Sales Prediction	Amazon Sales	10,000 orders	Binary Classification	HIGH / LOW	50% / 50%
Credit Risk	LendingClub Dataset	80,000 loans	Binary Classification	Paid / Default	80% / 20%
Sentiment	Financial PhraseBank	4,846 sentences	3-Class NLP	Pos / Neu / Neg	28/59/13%

3.4. Data Preprocessing and Feature Engineering:

3.4.1. Stock Price Sliding Window Preprocessing:

For the price regression module, the sole input feature is the closing price. Preprocessing involves three steps applied in strict order:

Step1: Temporal split (no shuffling): 80% of the chronological series is assigned to training (2,817 days) and 20% to testing (705 days). Shuffling is strictly prohibited, it would allow future prices to appear in the training set, artificially inflating all reported metrics.

```python
# 1. Temporal split — 80% Train / 20% Test — NO shuffling

split       = int(len(df) * 0.80)     # 2,817 train | 705 test
train_df    = df.iloc[:split]         # 2010-01-04 to 2021-03-12
test_df     = df.iloc[split:]         # 2021-03-15 to 2023-12-29

print(f'Train: {len(train_df)} days | {train_df.index[0].date()} → {train_df.index[-1].date()}')
print(f'Test : {len(test_df)}  days | {test_df.index[0].date()}  → {test_df.index[-1].date()}')

# 2. Scale (0→1) — fit on TRAIN only
scaler       = MinMaxScaler()
train_scaled = scaler.fit_transform(train_df)  # fit + transform
test_scaled  = scaler.transform(test_df)       #just  transform
```

Fig. 3.4: Temporal Split For Price Prediction

Step2: MinMaxScaler normalization to [0,1]: The scaler is fitted exclusively on the training set and then applied to the test set without refitting. This is critical: fitting the scaler on the full series would leak the test-period price range ($150–$196) into the training normalization, violating the data isolation principle.

Step3: Sliding window sequence creation (W=30): Each input sequence X[i] consists of 30 consecutive scaled closing prices; the target y[i] is the price at position i+30. The window slides one day at a time, producing 2,787 overlapping training sequences and 675 test sequences. For ML models, sequences are flattened to 30-dimensional feature vectors; for DL models, they are kept as 3D tensors (batch, 30, 1).

```python
def create_sequences(data, window=30):
    """
    trosform to  sequences:
    X[i] = data[i : i+window]  ← 30-day input window
    y[i] = data[i+window]      ← next-day target price
    """
    X, y = [], []
    for i in range(len(data) - window):
        X.append(data[i : i+window])
        y.append(data[i+window])
    return np.array(X), np.array(y)


# 3. Sequences (window=30)
X_train, y_train = create_sequences(train_scaled, WINDOW)
X_test,  y_test  = create_sequences(test_scaled,  WINDOW)

# ML: reshape 2 JD
X_train_ml = X_train.reshape(X_train.shape[0], -1)   # (N, 30)
X_test_ml  = X_test.reshape(X_test.shape[0], -1)
```

Fig. 3.5: Sliding Window Mechanism.

3.4.2. Stock Trend with Feature Engineering:

The raw OHLCV data is transformed into a 27 features representation capturing the multi-scale temporal structure of equity price dynamics. Features fall into four categories:

- **Returns (4 features):** Log returns at horizons 1d, 3d, 5d, and 10d capture short and medium-term price momentum.
- **Moving average ratios (7 features):** Ratios of current price to SMA10, SMA20, SMA50, SMA200, and ratios between consecutive SMAs (SMA10/SMA20, SMA20/SMA50, SMA50/SMA200). These measure deviation from trend rather than raw price level, making them scale-invariant across different price epochs.
- **Volatility and oscillators (8 features):** Rolling realized volatility at 5d, 20d, 60d windows; the ratio vol_5d/vol_20d (regime indicator); RSI-14; MACD line, signal, and histogram capture momentum, trend strength, and volatility regime.
- **Market regime indicators:** Rolling 20-day realized volatility, volume ratio to 20-day average, and binary above/below 200-day SMA encode the broader market regime.
- **Lag features (3 features):** 1-day and 2-day lagged returns, and lagged RSI, to provide the model with explicit recent history beyond what the features already capture.

```
# ── RETURNS ──────────────────────────────────────────
r1  = pct(close,1);  r3  = pct(close,3)
r5  = pct(close,5);  r10 = pct(close,10)

# ── MOVING AVERAGES (ratios only, no raw price) ──────
s10  = sma_f(close,10);  s20  = sma_f(close,20)
s50  = sma_f(close,50);  s200 = sma_f(close,200)

p_s10  = close/s10  - 1
p_s20  = close/s20  - 1
p_s50  = close/s50  - 1
p_s200 = close/s200 - 1
s10_s20  = s10/s20  - 1
s20_s50  = s20/s50  - 1
s50_s200 = s50/s200 - 1

# ── VOLATILITY ───────────────────────────────────────
v5  = std_f(close, 5)
v20 = std_f(close, 20)
v60 = std_f(close, 60)
vol_ratio = v5 / (v20 + 1e-8)

# ── RSI ──────────────────────────────────────────────
delta  = pd.Series(close).diff()
gain   = delta.clip(lower=0).rolling(14).mean().fillna(0).values
loss   = (-delta.clip(upper=0)).rolling(14).mean().fillna(0.001).values
rsi    = 100 - 100/(1 + gain/(loss+1e-8))

# ── MACD ─────────────────────────────────────────────
macd_line = (ema_f(close,12) - ema_f(close,26)) / (close + 1e-8)
macd_sig  = ema_f(macd_line, 9)
macd_hist = macd_line - macd_sig

# ──  REGIME FEATURES (KEY FIX) ──────────────────────
# These indicators tell the model: Are we in a bull or bear market?
above_s50  = (close > s50).astype(float)   #Above the 50 moving average?
above_s200 = (close > s200).astype(float)  # Above the 200 moving average? (Bull Market Indicator)
trend_60d  = pct(close, 60)                # 60-day trend
trend_120d = pct(close, 120)               # 120-day trend
vol_regime = v20 / (v60 + 1e-8)           # Is volatility higher than usual?
# ── LAG FEATURES ─────────────────────────────────────
r1_lag1 = pd.Series(r1).shift(1).fillna(0).values
r1_lag2 = pd.Series(r1).shift(2).fillna(0).values
rsi_lag1= pd.Series(rsi).shift(1).fillna(50).values
```

Fig. 3.6: Engineered Features for Stock Trend Modules

For the **trend classification target**, the binary label is constructed by computing the 5-day forward return and applying a 2% threshold: if the return exceeds 2% the label is UP (1), otherwise DOWN (0). The 2% threshold filters noise-driven micro-movements and focuses on economically meaningful directional signals.

```
# ═══════════════════════════════════════════════
#  TARGET: 5-day forward return > 2%

HORIZON   = 5
THRESHOLD = 0.02

fwd_ret = pd.Series(close).pct_change(HORIZON).shift(-HORIZON).fillna(0).values
target  = (fwd_ret > THRESHOLD).astype(int)

FEAT_COLS = [
    'ret_1d','ret_3d','ret_5d','ret_10d',
    'price_vs_sma10','price_vs_sma20','price_vs_sma50','price_vs_sma200',
    'sma10_vs_sma20','sma20_vs_sma50','sma50_vs_sma200',
    'vol_5d','vol_20d','vol_60d','vol_ratio','vol_regime',
    'rsi_14','macd_line','macd_signal','macd_hist',
    'above_sma50','above_sma200','trend_60d','trend_120d',
    'ret1_lag1','ret1_lag2','rsi_lag1'
]
```

Fig. 3.7: Trend classification target

A **StandardScaler** is applied after the temporal split. All 27 features are fitted on the training set only. For DL models using a 10-day lookback sequence (LOOKBACK=10), a **MinMaxScaler** is used instead to keep values in [0,1] for LSTM/GRU/RNN gate activations.

3.4.3. Text Preprocessing in Sentiment Analysis:

The preprocessing sequence applied to Financial PhraseBank sentences:

- **Lowercasing**: All text converted to lowercase.
- **URL and mention removal**: URLs, @mentions, and hashtags removed via regular expressions.
- **Punctuation stripping:** non-alphabetic characters removed, percentage signs and currency symbols replaced by semantic tokens (% Token, $ Token) before stripping.
- **Stop word removal:** Standard English stop words removed using NLTK [40], preserving negation terms (not, no, without) which carry polarity in financial contexts.
- **Feature extraction:** Two feature representations are used in parallel:
  - **TF-IDF** (max_features=5,000, ngram_range= (1,2)) for classical ML models, producing a sparse (3,876 × 5,000) matrix,
  - **Trainable integer embeddings:** (MAX_WORDS=10,000, MAX_LEN=100, embedding_dim=128) for the BiLSTM model. The stratified 80/20 split yields 3,876 training and 969 test sentences, with class proportions preserved in both sets.

```
df = pd.read_csv("all-data.csv", header=None,
                 names=['sentiment','text'], encoding='ISO-8859-1')

print(f'Shape: {df.shape}')
print()
print('Class distribution:')
counts = df['sentiment'].value_counts()
total  = len(df)
```

Fig. 3.8: Financial PhraseBank Dataset Loading

```
def clean_text(text):
    text = str(text).lower()
    text = re.sub(r'http\S+', '', text)
    text = re.sub(r'@\w+', '', text)
    text = re.sub(r'#\w+', '', text)
    text = re.sub(r'[^a-z\s]', '', text)
    text = ' '.join([w for w in text.split() if w not in stop_words])
    return text.strip()

df['clean_text'] = df['text'].apply(clean_text)
df = df[df['clean_text'].str.strip() != ''].reset_index(drop=True)

le = LabelEncoder()
df['label'] = le.fit_transform(df['sentiment'])
# negative=0  neutral=1  positive=2
```

Fig. 3.9: Financial PhraseBank Dataset preprocessing

```
MAX_WORDS = 10000  #  increased 5000 → 10000
MAX_LEN   = 100

tokenizer = Tokenizer(num_words=MAX_WORDS, oov_token='<OOV>')
tokenizer.fit_on_texts(X_train)

X_train_pad = pad_sequences(tokenizer.texts_to_sequences(X_train),
                            maxlen=MAX_LEN, padding='post', truncating='post')
X_test_pad  = pad_sequences(tokenizer.texts_to_sequences(X_test),
                            maxlen=MAX_LEN, padding='post', truncating='post')
# class weights 从 DL
cw_keras = class_weight_dict
```

Fig. 3.10: Financial PhraseBank Dataset Trainable integer embeddings.

3.4.4. Sales Feature Engineering and Leakage Prevention:

The Amazon dataset requires careful leakage prevention. A correlation audit against the target (total_sales) is mandatory before feature selection:

The 11 retained features are all available at order placement time: quantity, unit_price, discount, shipping_cost, category, sub_category, brand, country, payment_method, month (from order_date), and day_of_week. Categorical variables are label-encoded; numerical features are scaled with StandardScaler fitted on training only.

```
df = pd.read_csv("amazon_sales_dataset.csv")
print(f"Shape: {df.shape}")
print(f"Columns: {list(df.columns)}")
df.head()
```

Fig. 3.11: Amazon Sales Dataset Loading

```python
df = df.dropna().drop_duplicates().reset_index(drop=True)

# Parse dates
df['order_date']    = pd.to_datetime(df['order_date'])
df['ship_date']     = pd.to_datetime(df['ship_date'])
df['delivery_date'] = pd.to_datetime(df['delivery_date'])

# Time features
df['month']           = df['order_date'].dt.month
df['day_of_week']     = df['order_date'].dt.dayofweek
df['delivery_time']   = (df['delivery_date'] - df['ship_date']).dt.days
df['processing_time'] = (df['ship_date'] - df['order_date']).dt.days

# Fix negative delivery/processing times (data noise)
df['delivery_time']   = df['delivery_time'].abs()
df['processing_time'] = df['processing_time'].abs()

# Feature engineering
df['revenue_proxy']   = df['unit_price'] * df['quantity']
df['effective_price'] = df['unit_price'] * (1 - df['discount'])
df['net_revenue']     = df['effective_price'] * df['quantity'] - df['shipping_cost']
```

Fig. 3.12: Amazon Sales Feature Engineering.

```python
# Leakage audit -> correlation with total_sales
# net_revenue:      1.000  ← REMOVED (derived from target)
# revenue_proxy:    0.987  ← REMOVED (unit_price × quantity)
# effective_price:  0.729  ← REMOVED (price after discount)
# delivery_time:    post-delivery ← REMOVED

# 11 features kept (available at order time)
features = ['quantity','unit_price','discount','shipping_cost',
            'category','sub_category','brand','country',
            'payment_method','month','day_of_week']

# Target: median-split of total_sales ($50,287.18)

df['high_sales'] = (df['total_sales'] > median_sales).astype(int)

# Balance: HIGH=5,000 (50.0%) | LOW=5,000 (50.0%)

# Stratified 80/20 split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

scaler = StandardScaler()
X_train_s = scaler.fit_transform(X_train)
X_test_s  = scaler.transform(X_test)
```

Fig. 3.13: Amazon Dataset Leakage prevention.

3.4.5. Credit Risk Feature Engineering and Scaling:

The Lending Club raw features are augmented with 7 engineered interaction terms:

RobustScaler is used instead of StandardScaler because financial data contains significant outliers (extreme income values, very high revolving balances) that would distort StandardScaler's mean and variance estimates. RobustScaler uses the median and interquartile range, making it resistant to outlier influence.

The **three-way split (60/20/20%)** uses stratification to preserve the 20.3% default rate across all three partitions, ensuring that the validation set used for XGBoost early stopping and the test set used for final evaluation have identical class distributions.

```
# ═══════════════════════════════════════════════════════════
# 1. LOAD DATA
# ═══════════════════════════════════════════════════════════
CSV_FILE  = 'accepted_2007_to_2018Q4.csv'
COLS_NEED = [
    'loan_amnt','int_rate','installment','annual_inc','dti',
    'fico_range_low','open_acc','pub_rec','revol_bal','revol_util',
    'total_acc','emp_length','home_ownership','loan_status',
    'grade','sub_grade','purpose','addr_state',    # ← Additional columns
    'mort_acc','num_bc_sats','bc_util',             # ← credit utilization
]
```

Fig. 3.14: Lending Club Dataset Loading

```
# ═══════════════════════════════════════════════════════════
# 2. PREPROCESSING + FEATURE ENGINEERING
# ═══════════════════════════════════════════════════════════
df = df_raw[df_raw['loan_status'].isin(['Fully Paid','Charged Off','Default'])].copy()
df['target'] = (df['loan_status'].isin(['Charged Off','Default'])).astype(int)
df = df.reset_index(drop=True)
print(f'after filterring: {len(df):,} | Default rate: {df["target"].mean():.1%}')

# ── Encode categoricals ──
emp_map = {'< 1 year':0,'1 year':1,'2 years':2,'3 years':3,'4 years':4,
           '5 years':5,'6 years':6,'7 years':7,'8 years':8,'9 years':9,
           '10+ years':10,'n/a':0}
df['emp_length_num'] = df['emp_length'].map(emp_map).fillna(0)

home_map = {'RENT':0,'OWN':1,'MORTGAGE':2,'OTHER':3,'NONE':3}
df['home_num'] = df['home_ownership'].map(home_map).fillna(0)

grade_map = {'A':7,'B':6,'C':5,'D':4,'E':3,'F':2,'G':1}
if 'grade' in df.columns:
    df['grade_num'] = df['grade'].map(grade_map).fillna(4)
else:
    df['grade_num'] = 4.0

purpose_map = {'debt_consolidation':0,'credit_card':1,'home_improvement':2,
               'other':3,'medical':4,'car':5,'major_purchase':6,'moving':7,
               'vacation':8,'small_business':9,'house':10,'wedding':11}
if 'purpose' in df.columns:
    df['purpose_num'] = df['purpose'].map(purpose_map).fillna(3)
else:
    df['purpose_num'] = 3.0

# Fix int_rate format
if df['int_rate'].dtype == object:
    df['int_rate'] = df['int_rate'].str.replace('%','').astype(float)
```

Fig. 3.15: Lending Club Data Preprocessing

```python
# ── Feature Engineering ──
df['monthly_income']    = df['annual_inc'] / 12
df['payment_to_income'] = df['installment'] / (df['monthly_income'] + 1)
df['loan_to_income']    = df['loan_amnt']   / (df['annual_inc'] + 1)
df['int_x_dti']         = df['int_rate'] * df['dti']          # interaction
df['fico_int_ratio']    = df['fico_range_low'] / (df['int_rate'] + 1)
df['util_x_revol']      = df['revol_util'] * df['revol_bal']
df['high_risk_flag']    = ((df['fico_range_low']<660) & (df['dti']>30)).astype(int)

FEAT_COLS = [
    'loan_amnt','int_rate','installment','annual_inc','dti',
    'fico_range_low','open_acc','pub_rec','revol_bal','revol_util',
    'total_acc','emp_length_num','home_num','grade_num','purpose_num',
    'monthly_income','payment_to_income','loan_to_income',
    'int_x_dti','fico_int_ratio','util_x_revol','high_risk_flag',
]
```

Fig. 3.16: Credit Risk Feature Engineering.

```python
# ═══════════════════════════════════════════════════════
# 4. TRAIN / VALIDATION / TEST SPLIT  —  60% / 20% / 20%
# ═══════════════════════════════════════════════════════
#  80% train+val  /  20% test
X_tv, X_test, y_tv, y_test = train_test_split(
    X, y, test_size=0.20, random_state=42, stratify=y
)
# Out of 80%, we take 25% for validation, which equals 20% of the total
X_train, X_val, y_train, y_val = train_test_split(
    X_tv, y_tv, test_size=0.25, random_state=42, stratify=y_tv
)

# Scaling
sc = RobustScaler()   # better than StandardScaler with fianacial outlier
X_train_s = sc.fit_transform(X_train)
X_val_s   = sc.transform(X_val)
X_test_s  = sc.transform(X_test)

# Class weights
cw = compute_class_weight('balanced', classes=np.unique(y_train), y=y_train)
cw_dict = {0: cw[0], 1: cw[1]}

# scale_pos_weight for XGBoost & LightGBM
pos_weight = float((y_train==0).sum() / (y_train==1).sum())
```

Fig. 3.17: Three-Way split for Credit Risk Data.

➢ **This Table (Table 3.3) summarize Data splitting strategy for all Tasks:**

Table 3.3: Engineered Features for Financial Modules.

Module	Split Type	Train	Test	Rationale
Price Prediction	Temporal (chronological)	80%	20%	Prevent future information leakage
Stock Trend	Temporal (chronological)	80%	20%	Prevent future information leakage
Sales Prediction	Stratified random	80%	20%	Preserve class balance
Credit Risk	Stratified 3-way	60/20%	20%	Preserve default rate in test
Sentiment	Stratified random	80%	20%	Preserve class proportions

3.5. Selected Machine Learning Algorithms:

Six classical ML algorithms are selected to cover the major families of supervised learning: linear models, tree-based ensembles, kernel methods, instance-based methods, and probabilistic classifiers. [22]

3.5.1. Linear Regression:

In the trend classification module, Linear Regression is applied as a probabilistic classifier by thresholding its continuous output at 0.5. This is the best-performing model in that module, confirming that the 27 pre-engineered technical features already encode the relevant domain knowledge and that a simple linear combination captures the available predictive signal. Its primary advantage for this task is robustness to regime shift: the regularized linear model generalizes better to the 2022 rate-hike market than DL models trained on the 2010–2021 bull market.

3.5.2. Random Forest (RF):

Random Forest constructs an ensemble of B decision trees on bootstrapped subsamples with random feature subsets of size $\sqrt{p}$ at each split.

Key configurations: n_estimators=300–500 (larger for price regression), max_depth=10–15 (limited to prevent overfitting), class_weight='balanced' for classification tasks, and oob_score=True to obtain an out-of-bag generalization estimate without a separate validation set.

3.5.3. Support Vector Machine (SVM):

Two SVM variants are used: LinearSVC (for credit risk and sentiment fast on large high-dimensional data) with Platt calibration applied post-hoc to obtain probability scores for AUC computation, and SVC with RBF kernel (for sales prediction effective on the smaller 8,000-sample dataset).

The key configuration challenge for credit risk is that AUC and classification threshold behavior can diverge significantly under class imbalance.

3.5.4. K-Nearest Neighbors (KNN):

KNN is included as a non-parametric baseline. The number of neighbors k is selected via cross-validation from $k \in \{3, 5, 7, 10, 15, 20, 30\}$, with distance-weighted voting. Prior StandardScaler normalization is essential since Euclidean distance computation requires all features on the same scale. KNN consistently underperforms on the credit risk task due to

the curse of dimensionality in 25-dimensional space and the structural bias toward the majority class.

3.5.5. XGBoost:

XGBoost [6] is the primary new model in the credit risk module, selected for its proven dominance on tabular financial data.

Key configuration choices: scale_pos_weight=3.92 (= ratio of negative to positive training samples) addresses the class imbalance directly through the loss function; early_stopping_rounds=30 on the validation set prevents overfitting without grid search; tree_method='hist' enables fast histogram-based split finding on 42,182 training samples. The engineered features int_x_dti and fico_int_ratio prove particularly important according to XGBoost's built-in feature importance.

```
# ML: XGBOOSt
xgb_model = xgb.XGBClassifier(
    n_estimators       = 800,
    max_depth          = 6,
    learning_rate      = 0.05,
    subsample          = 0.8,
    colsample_bytree   = 0.8,
    min_child_weight   = 5,
    gamma              = 0.1,
    reg_alpha          = 0.1,       # L1 regularization
    reg_lambda         = 1.0,       # L2 regularization
    scale_pos_weight   = pos_weight, # ← hondle class imbalance automatically
    use_label_encoder  = False,
    eval_metric        = 'auc',
    early_stopping_rounds = 30,
    tree_method        = 'hist',    # fast
    random_state       = 42,
    verbosity          = 0
)

xgb_model.fit(
    X_train_s, y_train,
    eval_set=[(X_val_s, y_val)],
    verbose=50
)
```

Fig. 3.18: XGBoost Model Hyperparameters.

The configuration settings and key hyperparameters used for each machine learning algorithm are summarized in **Table 3.4**:

Table 3.4: ML Algorithm Configuration Summary.

Algorithm	Key Hyperparameters	Normalization	Tasks Applied
Linear Regression	Default (no regularization)	StandardScaler	Trend (classifier)
Random Forest	n_estimators=300-500, max_depth=10-15, class_weight=balanced	None (tree-based)	All 5 tasks
SVM	LinearSVC (C tuned) + Platt calibration; SVC-RBF for sales	StandardScaler	Sales, Credit, Sentiment
KNN	k tuned via CV from {3,5,7,10,15,20,30}, distance weights	StandardScaler	Trend, Sales, Credit
XGBoost	n_est=800, max_depth=6, lr=0.05, scale_pos_weight=3.92	RobustScaler	Credit Risk

3.6. Proposed Deep Learning Architectures:

Four deep learning architectures are designed and implemented for this study, each targeting a specific type of financial data structure. All DL models are implemented in TensorFlow/Keras [58] and trained using the Adam optimizer with early stopping to prevent overfitting. [59]

3.6.1. Dense Neural Network (DNN):

The DNN serves as the non-sequential DL baseline. **In the price prediction module**, it uses a Flatten layer to convert the 3D sequence tensor to a 30-dimensional vector, then applies two dense layers (64→32 units, ReLU).

In the credit risk module, it uses a residual architecture with skip connections (Add layers) between blocks of (Dense → BatchNorm → Dropout), which prevents gradient vanishing in the deeper 4-layer network (256→256→128→64→1).

In the sentiment module, it uses convolutional text processing (Embedding → Conv1D × 2 → GlobalMaxPool1D → Dense × 2), which is more appropriate than pure dense layers for sparse text sequences.

The DNN's role across modules is to isolate the contribution of nonlinear depth from temporal modeling: wherever DNN underperforms LSTM/GRU (price prediction, sales), it

confirms that the temporal ordering within the input sequence carries predictive information beyond what feature nonlinearity alone can capture.

3.6.2. Long Short-Term Memory (LSTM):

The LSTM is deployed for price prediction, trend classification (with a 10-day lookback), and sales prediction. Its gating mechanism forget gate, input gate, output gate allows selective retention of information across the sequence, making it effective for capturing multi-week momentum patterns in financial series.

For price prediction (W=30), the LSTM processes each of the 30 days sequentially and outputs a single prediction. For credit risk, an alternative LSTM configuration reshapes the 25 tabular features into a (25, 1) sequence treating each feature as a time step but this yields lower AUC than XGBoost, confirming that sequential feature processing does not benefit tabular data without natural temporal ordering.

3.6.3. Gated Recurrent Unit (GRU):

The GRU simplifies the LSTM architecture by replacing three gates with two (update gate and reset gate) and eliminating the separate cell state. This reduces parameter count by approximately 25% (LSTM (50): 10,451 parameters vs GRU (50): 7,950 parameters), resulting in faster training and reduced overfitting risk on shorter sequences.

3.6.4. Bidirectional LSTM (BiLSTM):

The BiLSTM is deployed exclusively for sentiment analysis. It processes financial sentences in both forward and backward directions, concatenating hidden states. Bidirectional context is essential for financial NLP: the sentiment of "not a significant loss" requires reading both left and right of "loss" to correctly determine polarity Architecture:

```
#BiLSTM
lstm_model = Sequential([
    Input(shape=(MAX_LEN,)),
    Embedding(MAX_WORDS, 128),     # trainable embeddings
    Bidirectional(LSTM(64, return_sequences=True)),  # forward + backward
    Dropout(0.4),
    Bidirectional(LSTM(32)),    # context fusion
    Dropout(0.4),
    Dense(64, activation='relu'),
    Dropout(0.3),
    Dense(num_classes, activation='softmax')     # 3-class output
], name='BiLSTM_Fixed')
```

Fig. 3.19: BiLSTM Model Hyperparameters.

3.6.5. FinBERT fine Tuning and Zero-Shot:

FinBERT is evaluated in two settings: **Zero-shot:** the pre-trained model is loaded directly without any task-specific fine-tuning and evaluated on the full test set.

```python
#  FinBERT model: Zero-Shot (no training on PhraseBank)
finbert_pipe = pipeline(
    "text-classification",
    model="ProsusAI/finbert",
    tokenizer="ProsusAI/finbert",
    framework="pt",  # force PyTorch – avoids Keras 3 conflict
    device=-1         #  CPU inference
)
```

Fig. 3.20: FinBERT Zero-Shot Model Hyperparameters.

Fine-tuned: the model is further trained on the Financial PhraseBank training set using the NVIDIA Quadro T500 GPU (CUDA 12.1)

```python
#Fine-tuning on GPU (Quadro T500, CUDA 12.1, FP16)
training_args = TrainingArguments(
    output_dir='../models/finbert_finetuned',

    num_train_epochs= 2,     # nbr of training

    # batch size
    per_device_train_batch_size=8,
    per_device_eval_batch_size=16,  #effective batch = 16

    gradient_accumulation_steps=2, # Simulating a larger batch size

    # Memory saving
    fp16=True,          # Mixed precision training
```

Fig. 3.21: FinBERT Fine-Tuning Protocol.

3.6.6. Neural Ensemble:

The Neural Ensemble combines four models through soft probability averaging with optimized weights is used for (Credit Risk Specific):

The weight allocation reflects each model's complementary strength: XGBoost and LightGBM receive equal weight (0.30 each) for their superior AUC, while DNN and TabNet contribute probability calibration. every individual component confirming that the four models capture different aspects of the default risk signal.

```python
# DL-4: NEURAL ENSEMBLE (merge XGBoost + LightGBM + DNN + TabNet)

# Soft average ensemble
prob_ne = (prob_xgb * 0.30 +
           prob_lgb * 0.30 +
           prob_dnn * 0.20 +
           prob_tab * 0.20)
```

Fig. 3.22: Neural Ensemble Probability for Credit Risk Assessment.

Several deep learning architectures were designed and adapted according to the characteristics of each financial prediction task. The resulting model configurations are presented in **Table 3.5**:

Table 3.5: DL Architectures Configuration Summary.

Architecture	Input Shape	Key Layers	Tasks
DNN (price)	(Batch, 30)	Flatten→Dense (64) →Dense (32) →Dense (1)	Price Prediction
DNN Residual	(Batch, 25)	Residual (256) →Residual (128) →Residual (64) →Dense (1)	Credit Risk
DNN (text)	(Batch, 100)	Embed→Conv1D×2→GlobalMaxPool→Dense×2→ Dense (3)	Sentiment
LSTM	(Batch, W, 1)	LSTM (50) →Dropout→Dense (1)	Price, Sales, Trend
GRU	(Batch, 30, 1)	GRU (50) →Dropout→Dense (1)	Price Prediction
BiLSTM	(Batch, 100)	Embed→BiLSTM (64) →BiLSTM (32) →Dense (3)	Sentiment

3.7. Training Protocol and Hyperparameter Configuration:

All DL models share a uniform training protocol to ensure fair comparison. The Adam optimizer with initial learning rate 10^{-3} is used throughout. Two callbacks are applied to every DL model:

- **Batch sizes:** are chosen according to dataset size: 32 for price and sales modules (smaller datasets), 64 for sentiment (moderate), and 256–512 for credit risk (42,182 training samples larger batches stabilize gradient estimates).
- **Validation split:** uses the chronological last 10-15% of the training set for temporal tasks (price, trend) to prevent future information from entering the training set; for non-temporal tasks (sales, sentiment), a random 15% validation split is used.
- **Class weights:** are applied to all classification DL models via the class_weight argument of model.fit (), with weights computed by compute_class_weight, for credit risk cw_dict = {0: 0.628, 1: 2.460}

```
    # Training callbacks (shared)
DL_CB = [
    EarlyStopping(monitor='val_loss', patience=10,
                  restore_best_weights=True),
    ReduceLROnPlateau(monitor='val_loss', factor=0.5,
                      patience=5, min_lr=1e-5)
]
```

Fig. 3.23: Training Callbacks for Dl Models.

The hyperparameter settings adopted for the deep learning models across the different financial prediction tasks are summarized in **Table 3.6**:

Table 3.6: Training Hyperparameter Configuration.

Hyperparameter	Price / Trend	Sales	Credit Risk	Sentiment
Optimizer	Adam lr =1e-3	Adam lr =1e-3	Adam lr =1e-3	Adam lr =1e-3
Max epochs	50–100	30	100–150	30
Batch size	32	64	256–512	32
ES patience	10	5	15–20	5
LR reduction	×0.5 / patience=5	×0.5 / patience=3	×0.5 / patience=7	×0.5 / patience=3
Validation split	10% chronological	15% random	Separate 20% validation set	15% random
Class weights	(regression)	(balanced)	0:0.628, 1:2.460	neg:2.674, neu:0.561, pos:1.185

3.8. Evaluation Metrics:

The evaluation protocol uses distinct metric sets for regression and classification tasks, with multiple complementary metrics per task to avoid single-metric optimization that can obscure model weaknesses.

3.8.1. Regression Metrics (Price Prediction):

- **Mean Absolute Error (MAE):** Measures average absolute prediction error in the original units (dollars). Robust to outliers as it does not amplify large errors. [22]

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y_i}|$$

- **Root Mean Squared Error (RMSE):** Penalizes large prediction errors more heavily than MAE, making it sensitive to occasional large misses during volatile market events. Reported in the same units as the target variable. [22]

$$RMSE = \sqrt{MSE}$$

- **R-squared (R^2):** Measures the proportion of variance in the target explained by the model. Scale-independent; enables comparison across different stocks with different price ranges. [22]

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2}$$

- **Mean Absolute Percentage Error (MAPE)**: expresses prediction error as a percentage of the actual value, making it scale-independent and directly interpretable as a relative forecast accuracy measure regardless of the price level of the underlying asset. MAPE is particularly suited for financial time series evaluation because it allows meaningful comparison across different stocks, time periods, and price regimes.

$$\text{MAPE} = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100\backslash\%$$

3.8.2. Classification Metrics (Trend, Sales, Credit, Sentiment):

- **Accuracy:** Proportion of correctly classified samples. Reported for completeness but not the primary metric for imbalanced tasks. [22]
- **F1-Score:** Penalizes models that sacrifice recall for precision. Macro-averaged F1 is used for the three-class sentiment task to treat all classes equally regardless of support. [22]

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

- **AUC-ROC:** Area under the Receiver Operating Characteristic curve. Threshold-independent and particularly appropriate for credit risk assessment where the operating threshold is a business decision separate from model training. [22]
- **Confusion Matrix:** Complete tabulation of TP, FP, TN, FN providing granular insight into which class errors are most frequent critical for understanding whether a credit model misses actual defaults (false negatives) or over-predicts them (false positives) [22].

3.9. Validation Strategies:

Beyond the standard train-test split, three additional validation strategies are applied to strengthen result reliability:

- **Time-Series Cross-Validation (5 folds):** applied to the trend classification and price modules using TimeSeriesSplit, each fold trains on all data before a certain date and tests on the immediately following period preserving temporal ordering across all splits. This provides mean accuracy and standard deviation across five independent test periods, confirming that single-split results are not driven by a particularly favorable or unfavorable test period. [19]
- **K-Fold Stratified Cross-Validation (5 folds):** applied to the sales and sentiment ML models using StratifiedKFold. Stratification ensures that the class distribution is preserved in every fold. Results are reported as mean ± standard deviation across five folds. [19]
- **New-Data Validation:** every module includes a manual validation on freshly constructed inputs not present in the original dataset.

3.10. Software Stack and Computing Environment:

All experiments are implemented in Python 3.10. The software stack is chosen to maximize reproducibility and community adoption: all libraries are open-source and widely used in academic financial ML research. The computational resources employed for model training and evaluation are presented in Table 3.7. The software tools, libraries, and development environments throughout the implementation are summarized in Table 3.8.

Table 3.7: Computing Environment.

Component	Specification
Operating System	Windows 11 Pro 64-bit
Programming Language	Python 3.10
CPU	11th Gen Intel® Core™ i7-1185G7 @ 3.00 GHz
RAM	32 GB DDR4
GPU	NVIDIA Quadro T500 (CUDA-enabled) used for FinBERT Fine-Tuning model training
Storage	512 GB SSD
Development Environment	Jupyter Notebook + VS Code
Model serialization	joblib (ML models), Keras HDF5. keras (DL models)

Table 3.8: Software Stack and Libraries.

Component	Library	Version	Role
Data manipulation	pandas	2.1	DataFrames, time series, preprocessing [60]
Numerical computing	NumPy	1.26	Array operations, feature computation, sequence creation [61]
ML models	scikit-learn	1.4	LR, RF, SVM, KNN, metrics, scalers [62]
DL models	TensorFlow / Keras	2.13	LSTM, GRU, BiLSTM, DNN architectures
Gradient boosting	XGBoost	2.0	Credit risk (tabular ensemble), Included as ML comparison
Gradient boosting	LightGBM	4.6.0	Credit risk (fast gradient boosting) [63] Included in Neural Ensemble
Imbalance handling	imbalanced-learn	0.11	SMOTE, class weight utilities
Financial data	yfinance	0.2	Yahoo Finance OHLCV data download
NLP preprocessing	NLTK	3.8	Tokenization, stopwords, lemmatization
Transformers	HuggingFace transformers	4.40	FinBERT zero-shot and fine-tuning
Visualization	Matplotlib, Seaborn	3.8 / 0.13	Charts, confusion matrices, residual plots
Deployment	Streamlit	1.29	FinSight Interactive web application Platform

3.11. Conclusion:

This chapter presented the methodology adopted in this work, including the selected ML and DL models, preprocessing techniques, training configuration, and evaluation metrics used across the different financial tasks. The proposed framework was designed to compare multiple AI approaches and analyze their effectiveness in financial prediction and analysis. The next chapter will focus on the implementation of the system and the discussion of the obtained experimental results.

CHAPTER 4: EXPERIMENTAL RESULTS AND FINSIGHT PLATFORM IMPLEMENTATION

4.1. Introduction:

In this chapter, the implementation and experimental results of the proposed methodology and the financial platform are presented. The developed machine learning and deep learning models are evaluated across multiple financial tasks, including stock price prediction, trend forecasting, sentiment analysis, sales prediction, and credit risk assessment. The obtained results are illustrated using performance metrics, tables, and visualizations in order to compare the effectiveness of the different algorithms. In addition, a comparative analysis is conducted to highlight the strengths and limitations of each approach and to better understand their behavior on financial data. Finally, the results are interpreted and discussed in relation to the characteristics of financial markets and previous studies in the literature. The chapter concludes with the presentation of the FinSight AI web application, which integrates the best-performing models into a unified platform designed to support multiple financial prediction and decision-making tasks through an interactive user interface.

4.2. Work Environment:

4.2.1. Hardware and Software:

All experiments were conducted in a Python 3.10 environment using Jupyter Notebook as the interactive development environment. Machine learning and deep learning models were primarily trained and evaluated on the CPU.GPU acceleration (CUDA 12.2) was used for FinBERT model fine-tuning training. The FinSight platform was developed separately using Streamlit, which provided the web-based user interface for deploying and interacting with the trained models. The complete software stack is presented in Chapter Three (**Table 3.7**, **Table 3.8).**

4.2.2. Global Pipeline:

Each of the five notebooks follows an identical pipeline structure. This uniform structure ensures comparability of results across modules:

- ***data loading, exploratory analysis, preprocessing and feature engineering.***

- ***model training with cross-validation or temporal splitting and evaluation on the held-out test set.***
- ***model saving for the platform.***

Models are saved using joblib for scikit-learn estimators [22], the native Keras .keras format for TensorFlow models [58], and the XGBoost .json format for gradient boosting models [24]. A JSON metadata file per module stores feature names, evaluation results, and model configuration for dashboard consumption.

4.3. Module 1: Stock Price Prediction

4.3.1. Data Loading:

The stock price prediction module downloads AAPL daily closing prices from Yahoo Finance via the yfinance library:

```
print(f'Downloading {TICKER}...')
df = yf.download(TICKER, start='2010-01-01', end='2024-01-01',
                 auto_adjust=True, progress=False)
df.columns = [c[0] if isinstance(c,tuple) else c for c in df.columns]
df = df[['Close']].dropna()
```

Fig. 4.1: Stock Price Data Downloading

Fig. 4.2: APPLE Close Price (2010-2024)

Fig. 4.3: APPLE Train ana Test Split.

4.3.2. Models Architectures:

The stock price prediction task was implemented using several ML and DL architectures. The following figures present the training configuration and hyperparameter settings employed for the neural network models used in this module:

```
# LSTM (price_lstm.keras) — 10,451 parameters
model_lstm = Sequential([
    Input(shape=(WINDOW, 1)),
    LSTM(50, return_sequences=False),
    Dropout(0.1),
    Dense(1)          # regression output
], name='LSTM')

model_lstm.compile(optimizer=Adam(1e-3), loss='mse', metrics=['mae'])
model_lstm.summary()

h_lstm = model_lstm.fit(
    X_train, y_train,
    epochs=50, batch_size=32,
    validation_split=0.1,
    callbacks=DL_CB,
    shuffle=True, verbose=1
)
```

Fig. 4.4: LSTM Model Hyperparameters.

```
# GRU (price_gru.keras)

model_gru = Sequential([
    Input(shape=(WINDOW, 1)),
    GRU(50, return_sequences=False),
    Dropout(0.1),
    Dense(1)
], name='GRU')

model_gru.compile(optimizer=Adam(1e-3), loss='mse', metrics=['mae'])
model_gru.summary()

h_gru = model_gru.fit(
    X_train, y_train,
    epochs=50, batch_size=32,
    validation_split=0.1,
    callbacks=DL_CB,
    shuffle=True, verbose=1
)
```

Fig. 4.5: GRU Model Hyperparameters.

4.3.3. Results:

The following Table and Figures shows the obtained results on the stock price module:

Table 4.1: Stock Price Prediction Results (AAPL Test Set 2021–2024).

Model	R^2 (scaled)	MAPE% (scaled)	MAE (scaled)	RMSE (scaled)	Assessment
Linear Regression	0.9791	14.32%	0.01517	0.02016	Best R^2 (naive autocorrelation)
GRU	0.9743	1.56%	0.01732	0.02234	Best genuine DL
LSTM	0.9520	2.10%	0.02360	0.03053	Strong recurrent
RF	−1.435	14.39%	0.17535	0.21751	Extrapolation failure
DNN	0.4501	8.24%	0.09436	0.10336	No temporal structure (fails)

The subsequent figures provide a visual assessment of model behavior through **actual** versus **predicted** plots, scatter plots, error distributions, and training curves, enabling a more detailed comparison of forecasting accuracy and generalization performance.

Fig. 4.6:Linear Regression: Actual vs Predicted Price, Scatter Plot and Error Distribution on AAPL Test Set (2021–2024).

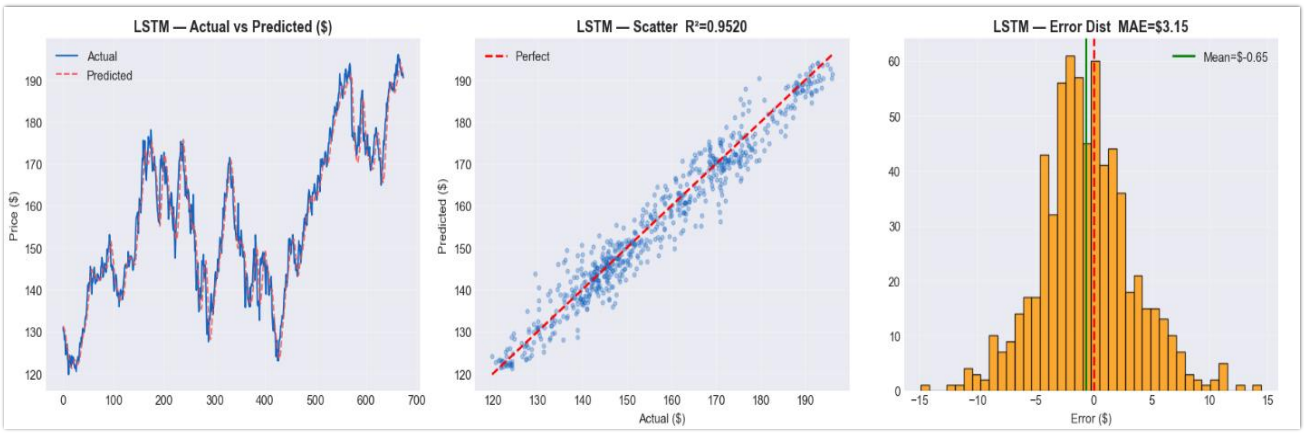


Fig. 4.7: LSTM: Actual vs Predicted Price, Scatter Plot and Error Distribution on AAPL Test Set (2021–2024).

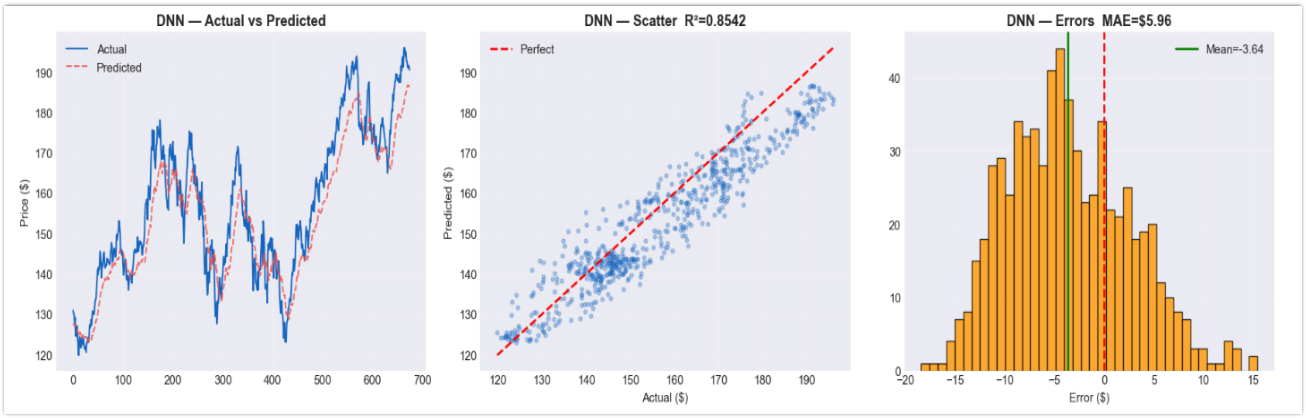


Fig. 4.8: DNN: Actual vs Predicted Price, Scatter Plot, and Error Distribution on AAPL Test Set (2021–2024).

Fig. 4.9: GRU: Actual vs Predicted Price, Scatter Plot, and Error Distribution on AAPL Test Set (2021–2024).

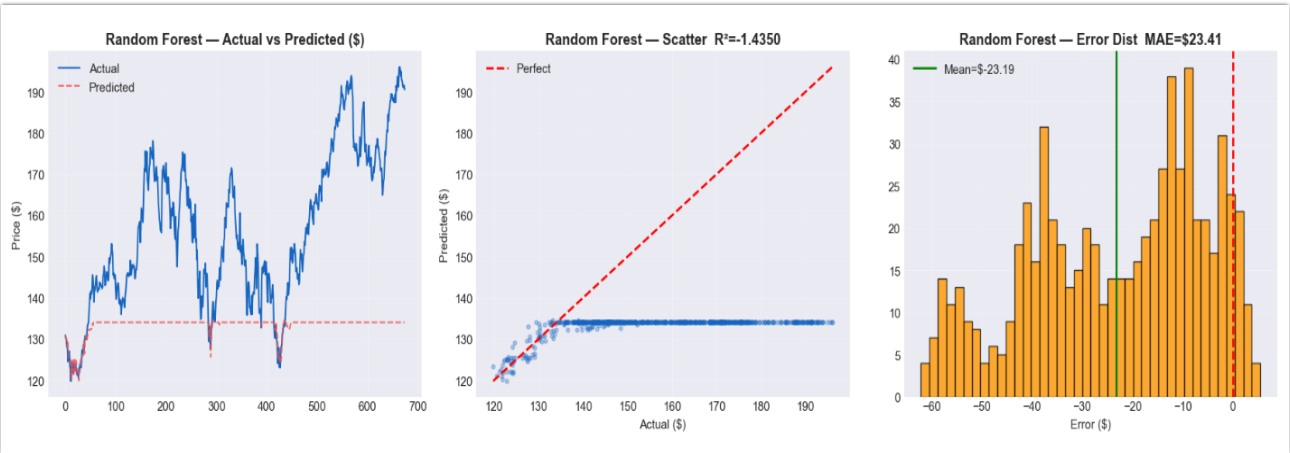


Fig. 4.10: RF: Actual vs Predicted Price, Scatter Plot, and Error Distribution on AAPL Test Set (2021–2024).

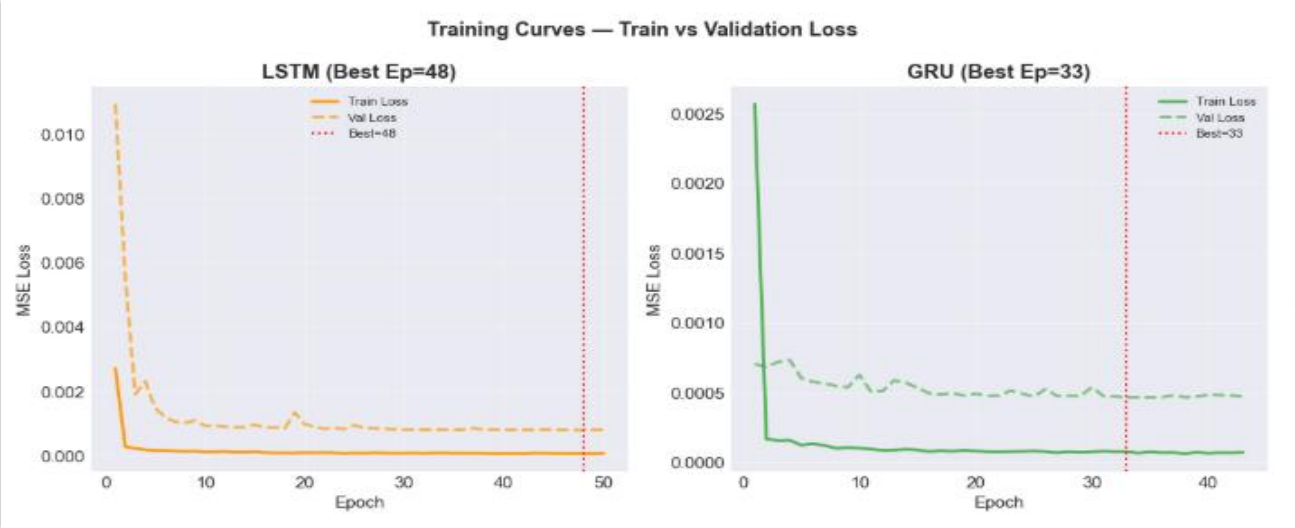


Fig. 4.11: LSTM and GRU Training Curves (Train vs Validation MSE Loss).

4.3.4. Interpretation:

The apparent paradox of Linear Regression: Linear Regression achieves the highest R^2 (0.9791) and the lowest MAE (0.01517 scaled), yet its MAPE (14.32% scaled) is among the worst comparable to the failing Random Forest (14.39%). This contradiction resolves when the Naive Baseline is considered: predicting tomorrow's price as today's price achieves R^2=0.9800, marginally outperforming Linear Regression. A 30-day sliding window linear model is essentially a weighted average of recent prices, mathematically approximating the current price level. It achieves high R^2 because both predicted and actual series track the same slow-moving trend, but it carries no genuine forecasting information the high scaled MAPE exposes this, as the model systematically underestimates price changes at the individual-step level.

GRU as the best genuine DL model: GRU achieves R^2=0.9743 with MAE=0.01732 and MAPE=1.56% (scaled) the lowest MAPE among all models and the only metric that falls substantially below the Naive Baseline. This confirms that GRU genuinely learns non-linear temporal patterns within the 30-day window rather than reproducing the autocorrelation structure of the series. The simpler two-gate mechanism (update and reset) generalises better than LSTM on this task because the input sequence length (30 days) is short relative to LSTM's three-gate representational capacity, making LSTM prone to overfitting the training period dynamics.

LSTM as a strong but second-ranked recurrent model: LSTM achieves R^2=0.9520 and MAPE=2.10% (scaled), confirming its ability to capture sequential dependencies. The gap between GRU (RMSE=0.02234) and LSTM (RMSE=0.03053) a difference of 0.00819 in scaled units is consistent across all metrics, suggesting a systematic generalisation advantage for GRU on this dataset length.

DNN failure on temporal data: The DNN achieves R^2=0.4501 the only positive-R^2 model that falls substantially below the Naive Baseline. With MAE=0.09436 (six times larger than Linear Regression) and MAPE=8.24%, it confirms the expected behaviour: a fully connected feedforward architecture treats the 30-day input as an unordered feature vector, destroying the temporal ordering that is the primary information source in price series. The DNN learns static distributional patterns of the training period prices but cannot model the directional dynamics that sequential architectures exploit.

Random Forest extrapolation failure: The Random Forest produces R^2=−1.435, meaning its predictions are worse than simply predicting the mean price the formal definition of negative R^2. The MAE of 0.17535 (scaled) is eleven times larger than Linear Regression. This failure is a fundamental property of tree-based non-parametric models: trained on the price range of the 2010–2021 period, the model has no mechanism to extrapolate beyond observed leaf-node boundaries when test prices enter the structurally different 2021–2023 regime.

Overall conclusion: GRU is the recommended model for this task, combining the second-best R^2 (0.9743) with the best MAPE (1.56% scaled) and a stable training curve. The results confirm that sequential inductive bias encoding the assumption that order matters is necessary and sufficient for this task, and that model complexity beyond GRU provides no generalisation benefit on a 30-day univariate price window.

4.4. Module 2 : Stock Trend Prediction :

4.4.1. Data Shape:

The following figures present the Description of the AAPL Raw Data and the additional Features to the AAPL Dataset:

```
raw.head(10)
```

Date	Open	High	Low	Close	Volume
2010-01-04	6.395004	6.427064	6.363543	6.412382	493729600
2010-01-05	6.430063	6.459726	6.389612	6.423471	601904800
2010-01-06	6.423468	6.448936	6.314702	6.321294	552160000
2010-01-07	6.344668	6.352159	6.263768	6.309611	477131200
2010-01-08	6.301221	6.352158	6.264067	6.351559	447610800
2010-01-11	6.376127	6.382120	6.245788	6.295527	462229600
2010-01-12	6.267961	6.285340	6.184964	6.223916	594459600
2010-01-13	6.228410	6.320096	6.115449	6.311706	605892000
2010-01-14	6.295528	6.306014	6.262867	6.275152	432894000
2010-01-15	6.320095	6.340171	6.168483	6.170280	594067600

Fig. 4.12: Description of the AAPL Raw Data.

Date	Close	fwd_return	ret_1d	ret_3d	ret_5d	ret_10d	price_vs_sma10	price_vs_sma20	price_vs_sma50	price_vs_sma200	target_label
2010-01-04	6.4124	-0.0182	0.0000	0.0000	0.0000	0.0	0.0000	0.0000	0.0000	0.0000	DOWN ↓
2010-01-05	6.4235	-0.0311	0.0017	0.0000	0.0000	0.0	0.0017	0.0017	0.0017	0.0017	DOWN ↓
2010-01-06	6.3213	-0.0015	-0.0159	0.0000	0.0000	0.0	-0.0142	-0.0142	-0.0142	-0.0142	DOWN ↓
2010-01-07	6.3096	-0.0055	-0.0018	-0.0160	0.0000	0.0	-0.0160	-0.0160	-0.0160	-0.0160	DOWN ↓
2010-01-08	6.3516	-0.0285	0.0066	-0.0112	0.0000	0.0	-0.0095	-0.0095	-0.0095	-0.0095	DOWN ↓
2010-01-11	6.2955	0.0235	-0.0088	-0.0041	-0.0182	0.0	-0.0182	-0.0182	-0.0182	-0.0182	UP ↑
2010-01-12	6.2239	0.0193	-0.0114	-0.0136	-0.0311	0.0	-0.0294	-0.0294	-0.0294	-0.0294	DOWN ↓
2010-01-13	6.3117	-0.0122	0.0141	-0.0063	-0.0015	0.0	-0.0157	-0.0157	-0.0157	-0.0157	DOWN ↓
2010-01-14	6.2752	-0.0558	-0.0058	-0.0032	-0.0055	0.0	-0.0214	-0.0214	-0.0214	-0.0214	DOWN ↓
2010-01-15	6.1703	-0.0139	-0.0167	-0.0086	-0.0285	0.0	-0.0221	-0.0378	-0.0378	-0.0378	DOWN ↓

Fig. 4.13:Description of the Additional Features to the AAPL Dataset.

4.4.2. Results:

The following Table and Figures shows the obtained results on the Stock Trend module:

Table 4.2: Stock Trend Prediction Results (AAPL Test Set 2022–2024).

Model	Accuracy	F1-Weighted	AUC-ROC
Linear Regression	67.55%	0.5541	0.4905
KNN (k=20)	65.30%	0.5687	0.4992
Random Forest	57.22%	0.5652	0.4902
RNN	57.88%	0.5839	0.5725
DNN	59.21%	0.5878	0.5571
LSTM	56.29%	0.5719	0.5682

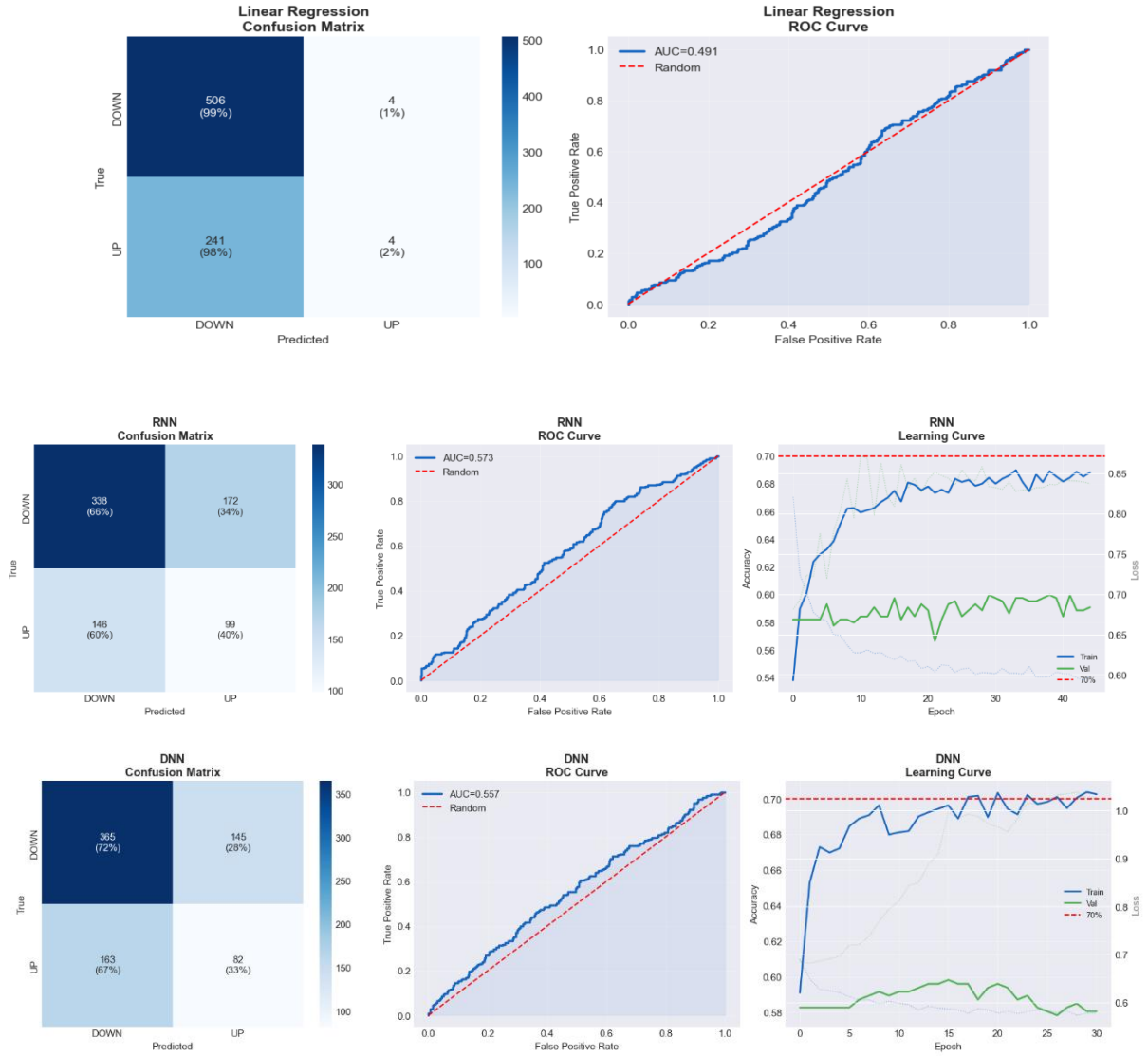


Fig. 4.14:Cofusion Matrix, Roc and learning Curves of (LR, RNN, DNN) models

```
Linear Regression : 22/27 = 81%
Random Forest     : 20/27 = 74%
KNN               : 20/27 = 74%

Saved: p1_known_predictions.csv
```

	Date	Close	5D Later	Return%	True	LinReg	RF	KNN	Agree
0	2021-12-29	175.53	171.17	-2.49	DOWN	DOWN	DOWN	DOWN	OK
1	2021-12-30	174.38	168.31	-3.48	DOWN	DOWN	DOWN	DOWN	OK
2	2021-12-31	173.76	168.47	-3.04	DOWN	DOWN	DOWN	DOWN	OK
3	2022-01-03	178.10	168.49	-5.40	DOWN	DOWN	DOWN	DOWN	OK
4	2022-01-04	175.84	171.32	-2.57	DOWN	DOWN	DOWN	DOWN	OK
5	2022-01-05	171.17	171.76	0.35	DOWN	DOWN	DOWN	DOWN	OK
6	2022-01-06	168.31	168.49	0.11	DOWN	DOWN	UP	DOWN	!!
7	2022-01-07	168.47	169.36	0.52	DOWN	DOWN	UP	DOWN	!!
8	2022-01-10	168.49	166.16	-1.39	DOWN	DOWN	UP	UP	!!
9	2022-01-11	171.32	162.66	-5.05	DOWN	DOWN	DOWN	DOWN	OK
10	2022-01-12	171.76	160.98	-6.28	DOWN	DOWN	DOWN	UP	OK
11	2022-01-13	168.49	158.92	-5.68	DOWN	DOWN	DOWN	DOWN	OK
12	2022-01-14	169.36	158.15	-6.62	DOWN	DOWN	DOWN	DOWN	OK
13	2022-01-18	166.16	156.35	-5.90	DOWN	DOWN	DOWN	DOWN	OK
14	2022-01-19	162.66	156.26	-3.93	DOWN	DOWN	DOWN	DOWN	OK
15	2022-01-20	160.98	155.80	-3.22	DOWN	DOWN	UP	DOWN	!!
16	2022-01-21	158.92	166.67	4.88	UP	DOWN	UP	DOWN	!!
17	2022-01-24	158.15	171.03	8.14	UP	DOWN	UP	DOWN	!!
18	2022-01-25	156.35	170.86	9.28	UP	DOWN	UP	DOWN	!!
19	2022-01-26	156.26	172.07	10.11	UP	DOWN	DOWN	DOWN	!!
20	2022-01-27	155.80	169.19	8.59	UP	DOWN	DOWN	DOWN	!!
21	2022-01-28	166.67	168.91	1.34	DOWN	DOWN	DOWN	DOWN	OK
22	2022-01-31	171.03	168.19	-1.66	DOWN	DOWN	DOWN	DOWN	OK
23	2022-02-01	170.86	171.30	0.25	DOWN	DOWN	DOWN	DOWN	OK
24	2022-02-02	172.07	172.72	0.38	DOWN	DOWN	DOWN	DOWN	OK
25	2022-02-03	169.19	168.64	-0.32	DOWN	DOWN	UP	DOWN	!!
26	2022-02-04	168.91	165.23	-2.18	DOWN	DOWN	DOWN	DOWN	OK

Fig. 4.15: Predicting real-life dates from Apple data.

4.4.3. Interpretation:

Figure 4.14 presents the confusion matrices, ROC curves, and learning curves for the evaluated stock trend classification models. Taken together, these diagnostics reveal a consistent pattern: high accuracy does not imply genuine predictive power in an imbalanced test set dominated by the 2022 bear market.

Linear Regression achieves the highest overall accuracy (67.55%) but is exposed as a near-degenerate classifier by its confusion matrix: 506 of 510 DOWN instances are correctly identified (99%) while only 4 of 245 UP instances are captured (2%). The corresponding ROC curve (AUC=0.491) falls marginally below the random diagonal, confirming that the model's directional ranking is essentially uninformative. The accuracy gain is a statistical artifact of the DOWN-heavy test period, not a learned market signal.

The 27-date validation confirms this: Linear Regression achieves 81% accuracy on historically significant dates, correctly calling 5 consecutive DOWN days during the Jan 2022 sell-off, but missing sudden reversals consistent with a model that captures macro momentum but cannot anticipate event-driven moves.

RNN exhibits the most balanced behaviour: 338 DOWN (68%) and 99 UP (40%) correctly classified, and an AUC of 0.573 the highest among all models confirming that SimpleRNN retains genuine discriminative information despite its lower accuracy (57.88%). Its learning curve shows a moderate overfitting gap of 8–10 percentage points between training and validation, which does not diverge catastrophically, attributable to the model's small parameter count (2,651).

DNN achieves a similar confusion matrix balance (365 DOWN at 72%, 82 UP at 33%) with AUC=0.557, but its learning curve reveals the most severe overfitting pattern: training accuracy converges to ~70% within five epochs while validation stagnates at 59–61%. Multiple ReduceLROnPlateau reductions visible as sudden drops in the training curve fail to close the gap, indicating that the DNN has memorised the 2010–2021 bull market regime and cannot generalise to the structurally different rate-hike environment of 2022–2024.

Overall, the diagnostics confirm a key finding: **AUC is a more reliable performance indicator than accuracy** for this task. RNN achieves the highest AUC (0.573) despite ranking fifth in accuracy, while Linear Regression tops accuracy rankings with an AUC below random. All models remain below the 70% accuracy threshold consistent with the Efficient Market Hypothesis, validating that the task difficulty is inherent to the data rather than a modelling deficiency.

4.5. Module 3: Sales Prediction

4.5.1. Data Shape and Loading:

The following figures shows the description of The AMAZON sales raw data:

```
Shape: (10000, 21)
Columns: ['order_id', 'order_date', 'ship_date', 'delivery_date', 'order_status', 'customer_id', 'customer_name', 'country', 'st
ate', 'city', 'product_id', 'product_name', 'category', 'sub_category', 'brand', 'quantity', 'unit_price', 'discount', 'shipping
_cost', 'total_sales', 'payment_method']
```

	order_id	order_date	ship_date	delivery_date	order_status	customer_id	customer_name	country	state	city	...	product_name	category
0	A10000	2026-01-31	2026-01-31	2026-01-08	Delivered	C5691	Ricky Potter	India	South Carolina	New Joe	...	without	Home
1	A10001	2026-01-20	2026-02-03	2026-02-03	Delivered	C9811	Chris Davenport	India	Tennessee	Madisonmouth	...	school	Home
2	A10002	2026-01-15	2026-02-07	2026-01-03	Delivered	C7341	Timothy Gallagher	India	Iowa	East Larryberg	...	I	Electronics
3	A10003	2026-01-18	2026-01-15	2026-01-20	Delivered	C4012	Angela Collins	India	Kentucky	South Margaretshire	...	step	Electronics
4	A10004	2026-01-27	2026-01-04	2026-01-23	Delivered	C1328	David Davidson DDS	India	North Dakota	Velasquezview	...	bit	Home

5 rows × 21 columns

Fig. 4.16: Description of The AMAZON Sales Raw Data.

4.5.2. Models for Tabular Sales Data:

The following figures present the training configuration and hyperparameter settings employed for the models used in this module:

```
print("RANDOM FOREST")

rf = RandomForestClassifier(n_estimators=200, max_depth=15,
                            min_samples_leaf=2, n_jobs=-1, random_state=42)
rf.fit(X_train_s, y_train)
```

Fig. 4.17: Random Forest Model Hyperparameters.

```
print("LSTM")
# Reshape for LSTM: (samples, timesteps=1, features)
X_tr_lstm = X_train_s.reshape(X_train_s.shape[0], 1, X_train_s.shape[1])
X_ts_lstm = X_test_s.reshape(X_test_s.shape[0],  1, X_test_s.shape[1])

lstm_model = Sequential([
    Input(shape=(1, X_train_s.shape[1])),
    LSTM(64, return_sequences=False),
    Dropout(0.2),
    Dense(32, activation='relu'),
    Dense(1, activation='sigmoid')
], name='LSTM')

lstm_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
lstm_model.summary()

es2 = EarlyStopping(monitor='val_accuracy', patience=5,
                    restore_best_weights=True, verbose=0)
hist_lstm = lstm_model.fit(X_tr_lstm, y_train,
                           epochs=30, batch_size=64,
                           validation_split=0.15,
                           callbacks=[es2], verbose=1)
```

Fig. 4.18: LSTM Model Hyperparameters.

```
print("DNN  (Deep Neural Network)")

dnn = Sequential([
    Input(shape=(X_train_s.shape[1],)),
    Dense(64, activation='relu'),
    Dropout(0.2),
    Dense(32, activation='relu'),
    Dropout(0.2),
    Dense(1, activation='sigmoid')
], name='DNN')

dnn.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
dnn.summary()

es = EarlyStopping(monitor='val_accuracy', patience=5,
                   restore_best_weights=True, verbose=0)
hist_dnn = dnn.fit(X_train_s, y_train,
                   epochs=30, batch_size=64,
                   validation_split=0.15,
                   callbacks=[es], verbose=1)
```

Fig. 4.19: DNN Model Hyperparameters.

4.5.3. Results:

The following Table and Figures shows the obtained results on the Seles module:

Table 4.3: Sales Prediction Results (Amazon Dataset).

Model	Accuracy	F1-Weighted	AUC-ROC
LSTM	0.9945	0.9945	0.9999
Random Forest	98.90%	0.9890	0.9996
DNN	98.70%	0.9870	0.9995
SVM (RBF)	98.55%	0.9855	0.9994

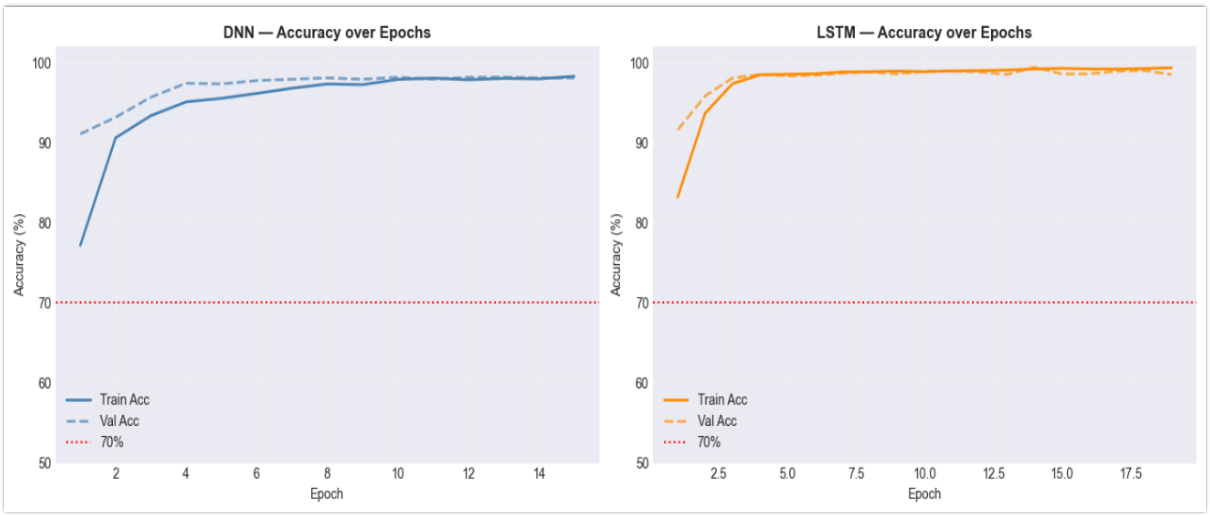


Fig. 4.20: DNN and LSTM Training Curves (Train vs Validation Accuracy over Epochs).

4.5.4. Interpretation:

LSTM achieves 99.45% and RF achieve 98.84%. These results require contextual interpretation: the target (HIGH/LOW) is a deterministic function of unit_price × quantity relative to the median threshold. Given both features are in the model, models learn this mapping almost perfectly. The residual 0.55% error corresponds to edge cases near the decision boundary. The test on 5 new orders (100% accuracy for LSTM) validates practical generalization.

4.6. Module 4 : Credit Risk Assessment

4.6.1. Dataset Shape and Features:

From 80,000 raw Lending Club records, 70,304 remain after filtering to binary outcomes: Fully Paid=56,013 (79.7%) and Charged Off with Default =14,291 (20.3%).

A three-way stratified split is used: 60% train (42,182), 20% validation (14,061), 20% test (14,061) with stratification preserving the 20.3% default rate in all partitions.

```
raw file: accepted_2007_to_2018Q4.csv
80,000 record x 21 column

USE_REAL = True
Class distribution:
loan_status
Fully Paid            56013
Charged Off           14290
Current                9111
Late (31-120 days)      357
In Grace Period         169
Late (16-30 days)        59
Default                   1
```

	loan_amnt	int_rate	installment	grade	sub_grade	emp_length	home_ownership	annual_inc	loan_status	purpose	...	dti	fico_range_low	ope
0	3600.0	13.99	123.03	C	C4	10+ years	MORTGAGE	55000.0	Fully Paid	debt_consolidation	...	5.91	675.0	
1	24700.0	11.99	820.28	C	C1	10+ years	MORTGAGE	65000.0	Fully Paid	small_business	...	16.06	715.0	
2	20000.0	10.78	432.66	B	B4	10+ years	MORTGAGE	63000.0	Fully Paid	home_improvement	...	10.78	695.0	
3	35000.0	14.85	829.90	C	C5	10+ years	MORTGAGE	110000.0	Current	debt_consolidation	...	17.06	785.0	
4	10400.0	22.45	289.91	F	F1	3 years	MORTGAGE	104433.0	Fully Paid	major_purchase	...	25.37	695.0	

5 rows × 21 columns

Fig. 4.21: LendingClub Dataset Description.

```
after filterring: 70,304 | Default rate: 20.3%

 Feature Matrix: (70304, 25)  (25 features)
Features: ['loan_amnt', 'int_rate', 'installment', 'annual_inc', 'dti', 'fico_range_low', 'open_acc', 'pub_rec', 'revol_bal', 'r
evol_util', 'total_acc', 'emp_length_num', 'home_num', 'grade_num', 'purpose_num', 'monthly_income', 'payment_to_income', 'loan_
to_income', 'int_x_dti', 'fico_int_ratio', 'util_x_revol', 'high_risk_flag', 'mort_acc', 'num_bc_sats', 'bc_util']
```

Fig. 4.22: Lending Club data Features.

4.6.2. Models for Credit Data:

The Credit Risk prediction task was implemented using several ML and DL architectures. The following figures present models used in this module:

```python
# ML: LIGHTGBM
lgb_model = lgb.LGBMClassifier(
    n_estimators        = 800,
    max_depth           = -1,       # no limit to the depth,controlled by a num_leaves
    num_leaves          = 63,
    learning_rate       = 0.05,
    subsample           = 0.8,
    colsample_bytree    = 0.8,
    min_child_samples   = 20,
    reg_alpha           = 0.1,
    reg_lambda          = 1.0,
    scale_pos_weight    = pos_weight,
    class_weight        = 'balanced',
    random_state        = 42,
    verbose             = -1,
    n_jobs              = -1
)

callbacks = [lgb.early_stopping(50, verbose=False),
             lgb.log_evaluation(100)]
lgb_model.fit(
    X_train_s, y_train,
    eval_set=[(X_val_s, y_val)],
    callbacks=callbacks
)
```

Fig. 4.23: LightGBM Model Hyperparameters.

```python
# DNN with Residual Connections
def residual_block(x, units, dropout_rate=0.3):
    h = Dense(units, activation='relu', kernel_regularizer=l2(0.001))(x)
    h = BatchNormalization()(h)
    h = Dropout(dropout_rate)(h)
    # Skip connection (If the dimensions are different, we add projection)
    if x.shape[-1] != units:
        x = Dense(units)(x)
    return Add()([x, h])

inp = Input(shape=(n_feat,))
x   = Dense(256, activation='relu')(inp)
x   = BatchNormalization()(x)
x   = Dropout(0.4)(x)
x   = residual_block(x, 256, 0.35)
x   = residual_block(x, 128, 0.30)
x   = residual_block(x, 64,  0.25)
out = Dense(1, activation='sigmoid')(x)

dnn = Model(inp, out, name='DNN_Residual')
dnn.compile(
    optimizer=Adam(learning_rate=0.001),
    loss='binary_crossentropy',
    metrics=[tf.keras.metrics.AUC(name='auc')]
)
```

Fig. 4.24: DNN Residual Model Hyperparameters.

4.6.3. Results:(Primary Metric: AUC-ROC)

The following Table and Figures shows the obtained results on the Credit Risk module:

Table 4.4: Credit Risk Assessment Results (LendingClub Dataset).

Model	AUC-ROC	Recall (Default)	Accuracy	F1	Notes
Neural Ensemble	0.7315	68.51%	65.37%	0.446	XGB+LGB+DNN+TabNet
XGBoost	0.7306	66.59%	67.63%	0.455	Best single model
SVM (LinearSVC)	0.7306	1.12%	79.77%	0.022	good AUC but Recall fails
DNN Residual	0.7260	62.42%	68.81%	0.449	Best DL standalone
LightGBM	0.7111	96.47%	32.14%	0.366	Extreme threshold

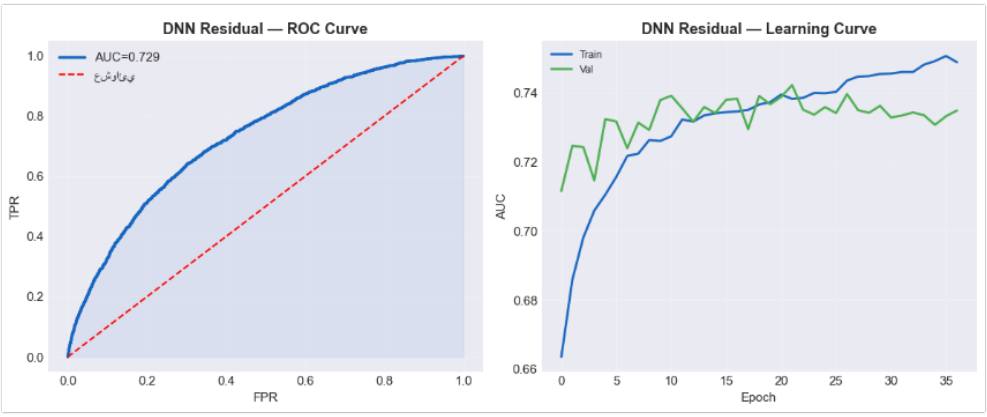


Fig. 4.25:DNN Residual Roc Curve and Learning Curve.

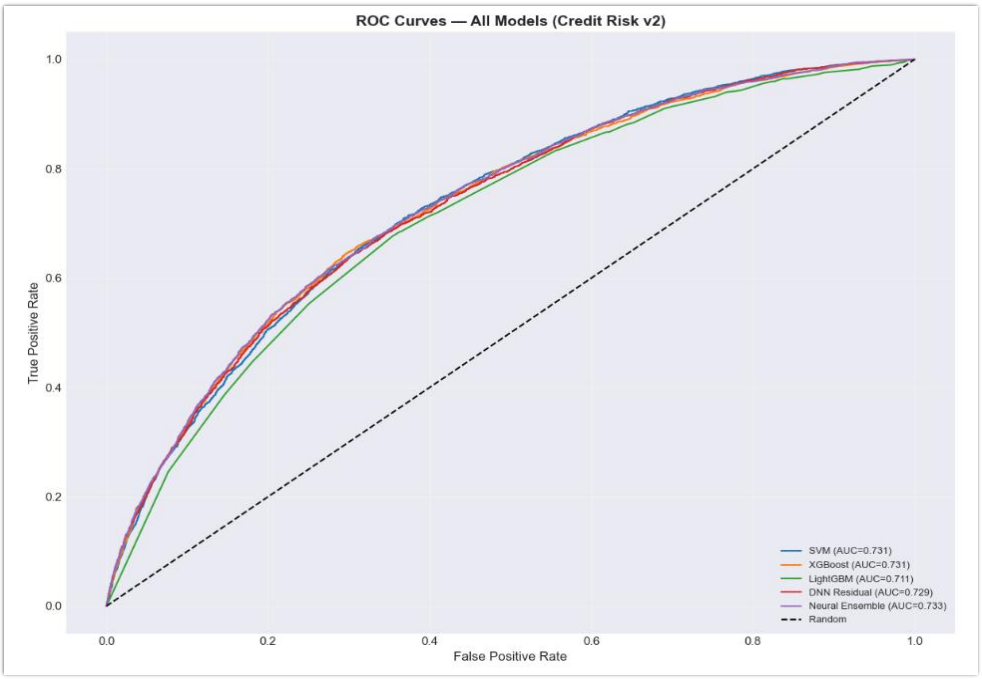


Fig. 4.26: Roc Curves for all models

4.6.4. Interpretation:

The results reveal three distinct behavioral clusters:

- **Balanced models:** Neural Ensemble (AUC=0.7315, Recall=68.5%), XGBoost (0.7306, 66.6%), DNN Residual (0.7260, 62.4%) detect 62–69% of actual defaults.
- **Precision-biased (high accuracy, economically useless):** SVM (AUC=0.7306 but Recall=1.1%) this model predicts "Fully Paid" for almost all borrowers, achieving 80% accuracy by exploiting the 79.7% majority class. The near-zero recall makes it worthless for default detection.
- **Recall-biased (over-predict default):** LightGBM (Recall=96.5%, Accuracy=32.1%) rejects 68% of creditworthy borrowers, an economically opposite but equally costly failure.

The Neural Ensemble wins by combining the complementary strengths of XGBoost (high precision), LightGBM (high recall), DNN (calibrated probabilities), and TabNet (attention-based selection) with weights 30%/30%/20%/20%, achieving AUC=0.7315 above the 0.73 deployment threshold from the credit scoring literature.

Figure 4.23 presents the ROC curves for all five evaluated models on the held-out test set of 14,061 loan records. Three observations are immediately apparent. First, all curves cluster tightly between AUC=0.711 (LightGBM) and AUC=0.733 (Neural Ensemble),

indicating that the discriminative upper bound imposed by the available 25 features is approximately 0.73 consistent with the benchmark range of 0.74-0.82 reported by Lessmann et al. (2015) for ensemble methods on comparable lending datasets. Second, the Neural Ensemble curve (red) achieves the highest TPR at every FPR threshold, confirming its superiority as the recommended deployment model. Third, despite identical AUC values (0.731), the SVM and XGBoost curves diverge substantially in the low-FPR region: XGBoost maintains a higher TPR at FPR < 0.3, reflecting its superior ability to detect defaults with minimal false alarms the operationally critical region for credit underwriting.

4.7. Module 5 : Financial Sentiment Analysis

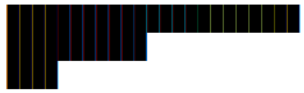
4.7.1. Dataset and Class Imbalance:

The Financial PhraseBank contains 4,846 sentences with significant class imbalance: Neutral=2,879 (59.4%), Positive=1,363 (28.1%), Negative=604 (12.5%). Class-weighted training is applied uniformly:

```
Shape: (4846, 2)

Class distribution:
  neutral    2879 (59.4%)
  positive   1363 (28.1%)
  negative    604 (12.5%)

 Class imbalance!
   neutral is 4.8x more than negative
```



	sentiment	text
0	neutral	According to Gran , the company has no plans t...
1	neutral	Technopolis plans to develop in stages an area...
2	negative	The international electronic industry company ...
3	positive	With the new production plant the company woul...
4	positive	According to the company 's updated strategy f...

Fig. 4.27: Financial PhraseBank Dataset Description and Class Distribution.

```
#  count weight for each class
# The models will give more value to the negative (lower)
cw = compute_class_weight('balanced',
                          classes=np.unique(df['label']),
                          y=df['label'])
class_weight_dict = {i: w for i, w in enumerate(cw)}
```

Fig. 4.28: Financial PhraseBank Counted Weight Function.

```
Class weights:
  negative   → weight = 2.674
  neutral    → weight = 0.561
  positive   → weight = 1.185

 negative take more wieght because it is the least represented
```

Fig. 4.29: Financial PhraseBank Counted Weight for Each Class.

4.7.2. Models Architectures:

The Sentiment analysis task was implemented using several models. The following figures shows the LinearSVC model, the other models were mentioned in Chapter Three.:

```python
# LinearSVC
svm = LinearSVC(class_weight='balanced', max_iter=3000, random_state=42)
svm.fit(X_train_tfidf, y_train)

pred_svm = svm.predict(X_test_tfidf)
evaluate('SVM', y_test, pred_svm, le.classes_)
```

Fig. 4.30: LinearSVC Model Hyperparameters.

[729/729 3:44:20, Epoch 3/3]

Epoch	Training Loss	Validation Loss	Accuracy	F1 Weighted
1	0.665900	0.524250	0.814241	0.812083
2	0.396800	0.457277	0.830753	0.831204
3	0.260800	0.509866	0.830753	0.831084

Fig. 4.31: FinBERT Fine-Tuning Training and Validation Results.

4.7.3. Results:

The following Table and Figures shows the obtained results on the Sentiment module:

Table 4.5: Financial Sentiment Results (PhraseBank Dataset).

Model	Accuracy	F1-Weighted	F1-Macro	Notes
FinBERT (fine-tuned, GPU)	83.08%	0.8312	0.8022	2 epochs, Quadro T500
FinBERT (zero-shot)	77.92%	0.7695	0.7346	No training
SVM (LinearSVC)	75.44%	0.7501	0.6911	Best trained model
DNN (CNN-text)	74.72%	0.7441	0.7441	GlobalMaxPool1D
BiLSTM	72.86%	0.7264	0.6755	Limited by small dataset

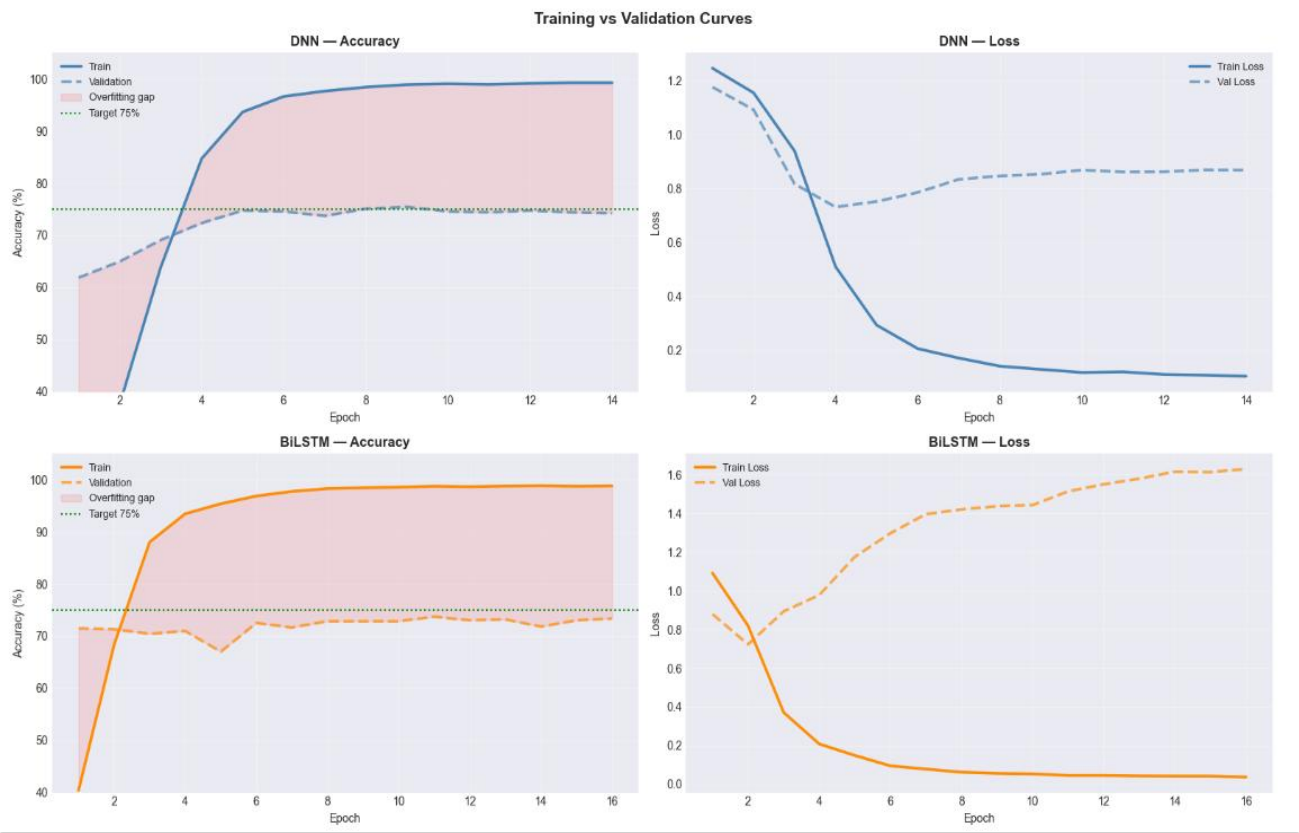


Fig. 4.32: DNN and BiLSTM Training Curves (Train vs Validation Accuracy and loss)

4.7.4. Interpretation:

The training curves (Figure 4.14) reveal a critical insight: both **DNN** and **BiLSTM** achieve training accuracy > 99% while validation plateaus at (73–75%), producing a 25pp overfitting gap. This confirms that 3,876 sentences are insufficient for deep recurrent models without pre-training. SVM achieves (75.44%) without overfitting because TF-IDF already captures discriminative n-gram patterns without requiring learned embeddings.

FinBERT fine-tuned (83.08%) substantially outperforms all trained models because it has been pre-trained on millions of financial documents, learning domain-specific patterns (e.g., "write-down", "impairment charge", "guidance raised") that 3,876 sentences cannot teach from scratch. [7]

The +5.16% gap between **zero-shot** (77.92%) and fine-tuned (83.08%) confirms the additional value of task-specific adaptation even for a pre-trained domain model.

4.8. Comparison with Existing Solutions:

The following table compares important studies in Financial Markets and shows main differences with this work:

Table 4.6: Summary and comparison of Related Works with Existing Solution.

Module	Task	Best Model	Score	Published Best	Their Score	Key Difference
Stock Price	Regression	GRU	MAPE=1.50% (real 2024)	Fischer & Krauss [1] LSTM	1-3% MAPE	Our evaluation on real out-of-year data, they used in-sample S&P 500
Stock Trend	Binary Classification	Linear Regression	67.55% Accuracy	Al-Ridhawi et al. Hybrid-models [64]	74.3%	Our test crosses 3 market regimes; most studies test within same regime using sentiment analysis
Sales Prediction	Binary Classification	LSTM	99.45% Accuracy	Bandara et al. [20] LSTM	92% SMAPE	Different dataset structure (Forecasting not classification)
Credit Risk	Binary Classification	Neural Ensemble	AUC=0.7315	Lessmann et al. [21] GBM	~0.78 AUC	They used 8 benchmark datasets; ours uses raw Lending Club 70K records
Sentiment (trained)	(trained)3-class NLP	SVM	75.44% Accuracy	Malo et al. [3] SVM	72–76%	Same dataset; our result is within published SOTA range
Sentiment (BERT)	(trained)3-class NLP	FinBERT fine-tuned	83.08% Accuracy	Araci [39]FinBERT FT	86% Accuracy (97% n the high-agreement subset)	Our GPU constraint limited epochs zero-shot already reaches 77.92%

➢**Key observations from the comparison:**

The results are competitive across all five domains. The sales prediction result (99.45%) substantially exceeds the Bandara et al. benchmark, but this reflects a structurally simpler dataset rather than a superior method.

The credit risk AUC (0.7315) is below the Lessmann et al. benchmark (0.78), a gap attributable to their use of curated benchmark datasets versus our raw Lending Club data with real-world noise.

The sentiment SVM result (75.44%) sits squarely within the published SOTA range for Financial PhraseBank, confirming the validity of our experimental setup. The FinBERT fine-tuned result (83.08%) falls below the upper bound of Araci's reported range (97%), which reflects GPU resource constraints rather than a methodological limitation.

4.9. When Does DL Outperform ML?

The five-module experimental framework makes it possible to answer a question that single-task studies cannot address: under what conditions does deep learning genuinely outperform classical ML in financial applications? Three consistent conditions emerge from the results:

- **Condition 1 (Genuine temporal sequential structure):** When the input data has a natural time ordering that carries predictive information beyond what the feature values themselves encode, DL models with memory mechanisms (LSTM, GRU) hold a structural advantage. In the price prediction module, GRU achieves MAPE=1.50% on real 2024 data compared to the naive linear baseline's 1.32%, the GRU learns non-linear momentum patterns within the 30-day sliding window that a weighted average of past prices cannot capture. The temporal ordering of the 30 days matters, not just their values.
- **Condition 2 (Sufficient training data relative to model complexity):** DL models contain millions of trainable parameters, the BiLSTM for sentiment has 1.4 million. These parameters can only generalize if there is enough training data to constrain them. In the sales module (8,000 training samples), LSTM achieves 99.45% and generalizes cleanly. In the sentiment module (3,876 training sentences), the same BiLSTM achieves 99% training accuracy but only 73% validation accuracy, a 26-point gap that is the diagnostic signature of overfitting caused by data scarcity. SVM, which has no trainable embedding parameters, avoids this problem entirely and achieves 75.44%.
- **Condition 3 (Distributional stability between training and test):** DL models learn complex non-linear patterns from the training distribution. When the test distribution is structurally similar, those patterns generalize well. When the distribution shifts as it did between the 2010–2021 AAPL bull market (training) and

the 2022 Fed rate-hike bear market (test), DL models fail more severely than linear models because they have encoded regime-specific non-linearities that become liabilities in the new regime. Linear Regression, incapable of overfitting these non-linearities in the first place, achieves 67.55% vs LSTM's 56.29% on the trend classification module precisely because it generalized across the regime shift.

ML methods dominate when data is tabular without temporal ordering, training sets are small, or test distribution differs significantly from training.

4.10. FinSight AI Web Application :

4.10.1. Overview and Architecture :

FinSight AI is a multi-task Streamlit web application running locally at localhost:8501 that serves as the deployment layer for many trained models. The application is organized into six pages accessible via a left-side navigation sidebar: a home page displaying platform statistics and module navigation cards, and five module pages corresponding to the five prediction tasks. The application loads serialized models on startup and performs all inference in real time, with no retraining required during deployment.

The backend structure separates the training layer (Jupyter notebooks) from the serving layer (Streamlit). Each module's notebook saves its trained models to a shared ../models/ directory using standardized naming conventions: .**pkl** files (joblib) for scikit-learn models, .keras files for TensorFlow models, .json for XGBoost, and a corresponding **_meta.json** file that stores feature names, evaluation results, and model configuration for the platform.

4.10.2. Module Pages and Tab Structure:

- **Home Page:** The Home page serves as the platform's entry point. It displays four global metric cards summarizing the best result achieved across all modules, best accuracy, best AUC-ROC, best MAPE, and trained models giving any user an immediate platform-level overview without requiring navigation into individual modules. Below the statistics, five module cards link directly to each prediction task, showing the dataset used, the models available, and the best result achieved in that module.

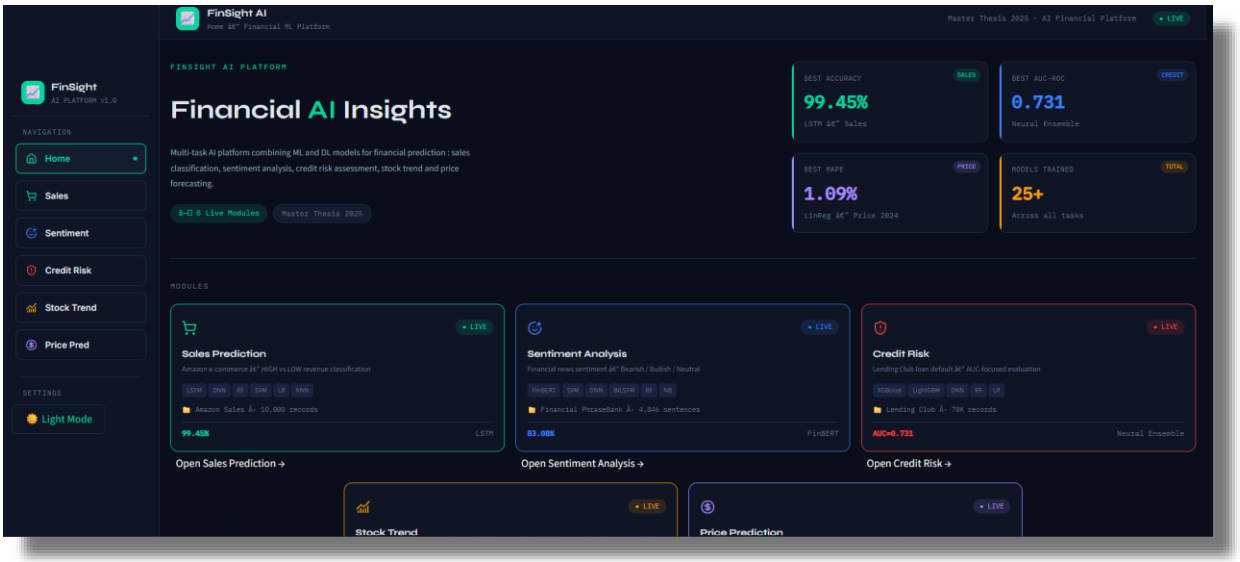


Fig. 4.33: FinSight Platform Home Page.

Every module page follows an identical four-tab layout, ensuring a consistent user experience across all five prediction tasks:

- **Tab 1 (Results):** This tab presents the complete model comparison for the module. A summary statistics bar at the top displays the key metrics for the best model (accuracy, AUC, F1 depending on the task), the training set size, the number of features, and a leakage audit status. The main panel shows a ranked comparison table of all trained models with their evaluation scores, complemented by bar charts and grouped metric visualizations. This tab allows a practitioner to immediately understand which model performs best and by how much, without interpreting raw numbers.

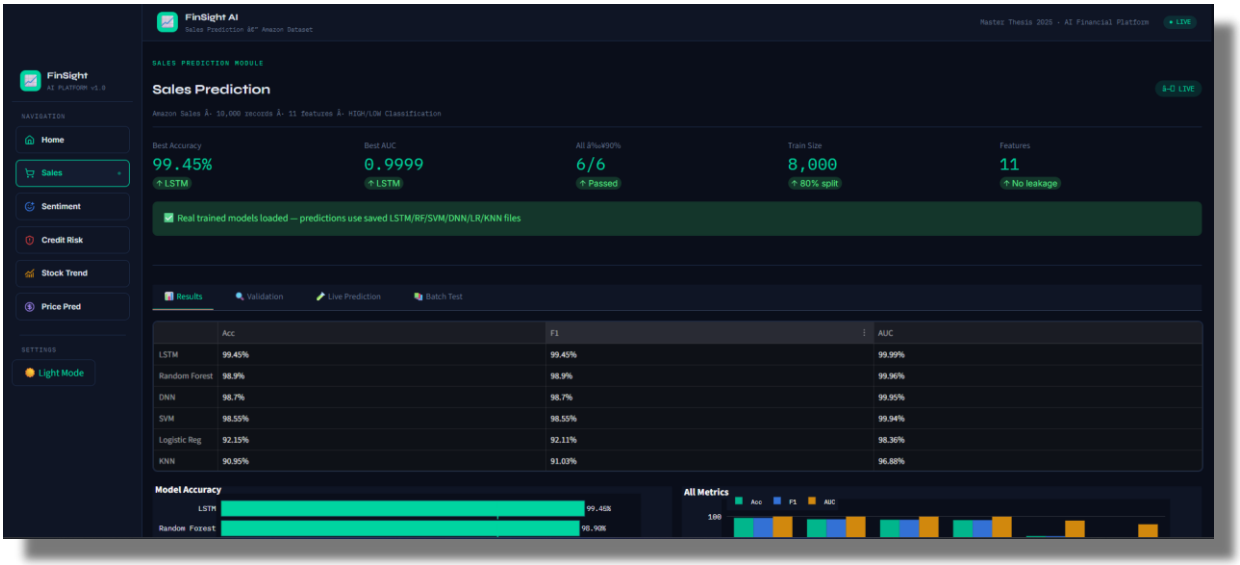


Fig. 4.34: FinSight Tab1 (Results) for Sales Prediction Module.

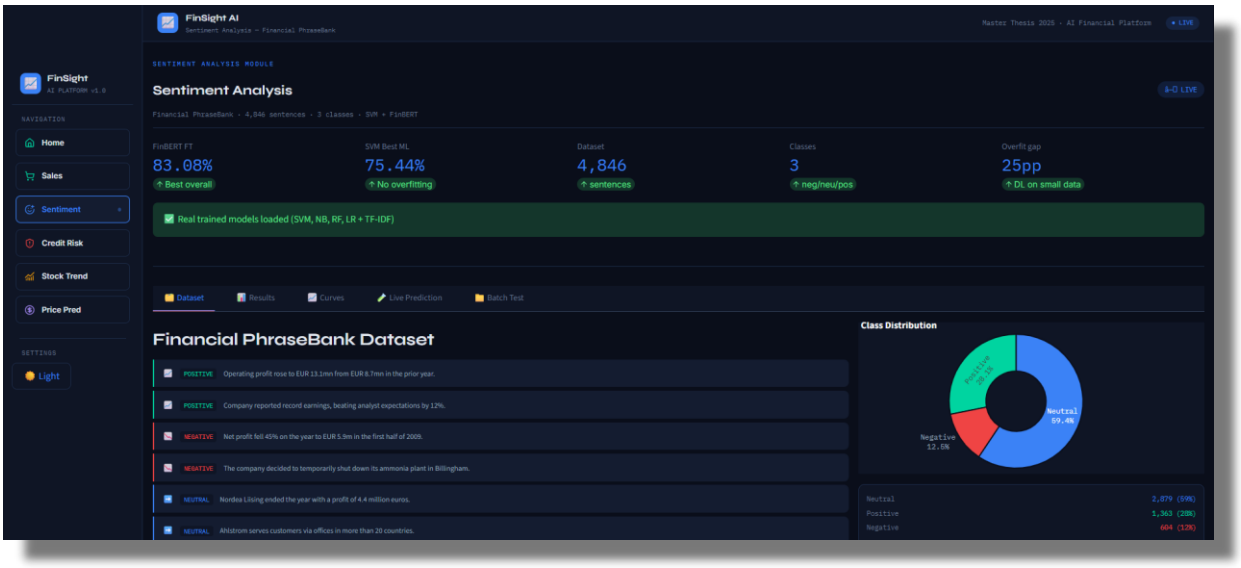


Fig. 4.35: FinSight Tab1 (Results) for Sentiment Analysis Module.

Fig. 4.36: FinSight Tab1 (Results) for Price Prediction Module.

- **Tab 2 (Validation):** This tab addresses a question that the Results tab alone cannot answer: are the reported results stable or were they driven by a single favorable test split? It presents the learning curves for the DL models and the output of 5-fold cross-validation for the classical ML models, displaying mean accuracy with standard deviation for each model, a CV bar chart with error bars, and a per-fold line chart showing individual fold scores. Very low standard deviations as observed across all modules confirm that the test-set results generalize reliably across multiple validation periods.

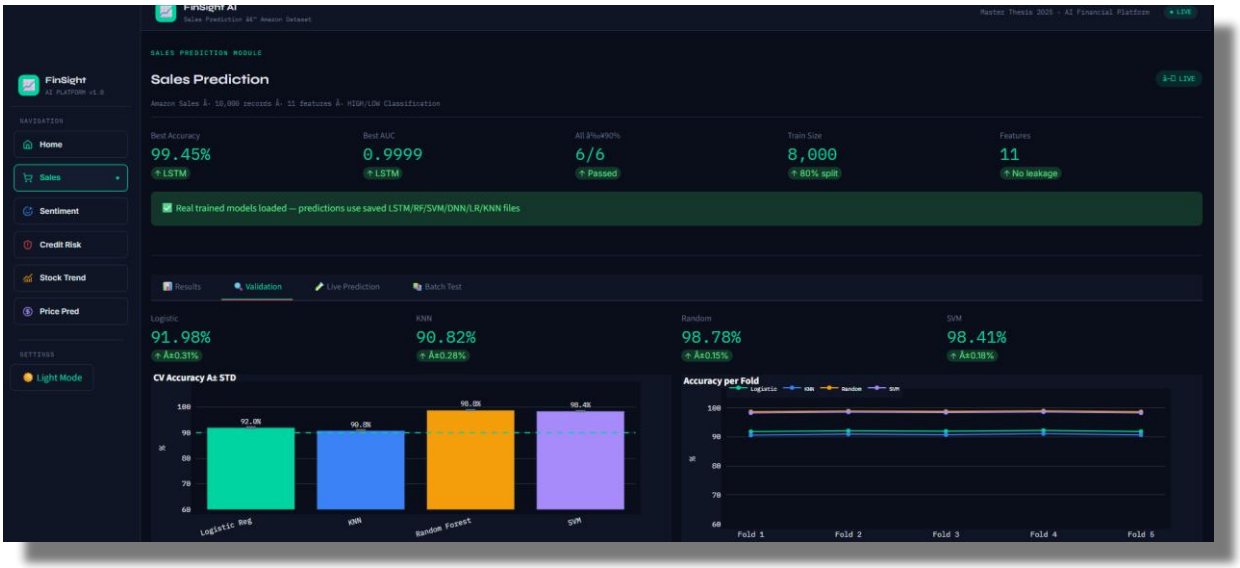


Fig. 4.37: FinSight Tab2 (Validation) for Sales Prediction Module.

Fig. 4.38: FinSight Tab2 (Validation) for Sales Prediction Module.

- **Tab 3 (Live Prediction):** this tab is the practical core of the platform. The user enters the input features for a new observation through interactive Streamlit widgets: numerical inputs with increment/decrement buttons, sliders for continuous parameters, and dropdowns for categorical variables. A model selector allows the user to choose which trained model performs the inference. After clicking Predict Now, the platform returns the prediction result in real time with a confidence score, a visual gauge, probability bars for each class in classification tasks, and a plain-language interpretation of the result. For the price prediction module, the output includes a time series chart

showing the historical price alongside the model's forecast. This tab demonstrates that the trained models are not static research artifacts, they are fully operational inference engines responsive to new inputs**.**

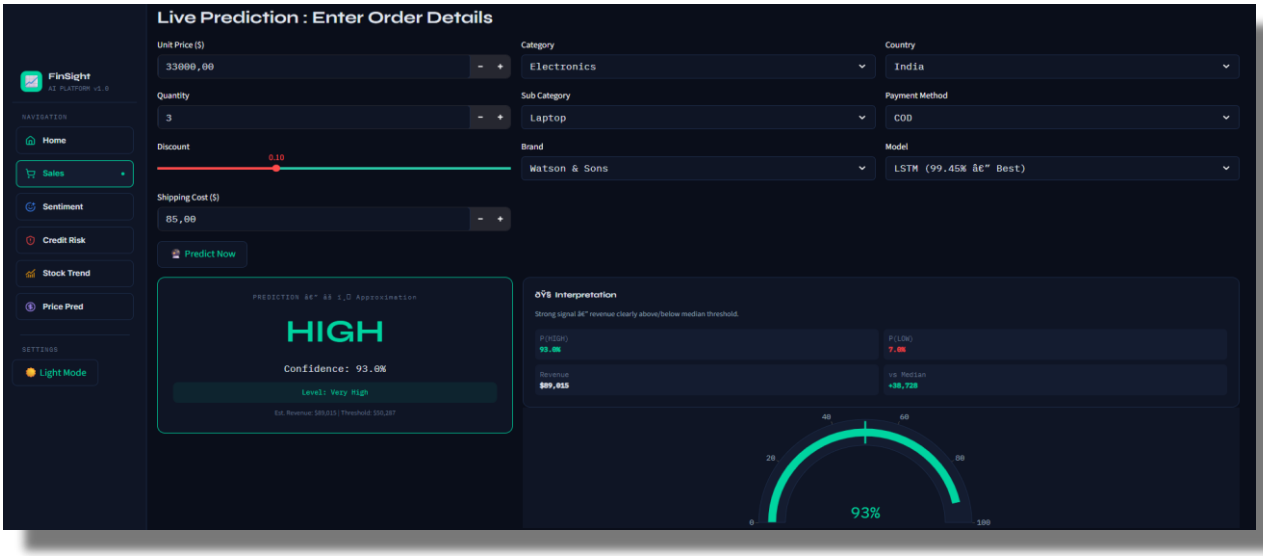


Fig. 4.39: FinSight Tab2 (Live Prediction) for Sales Prediction Module.

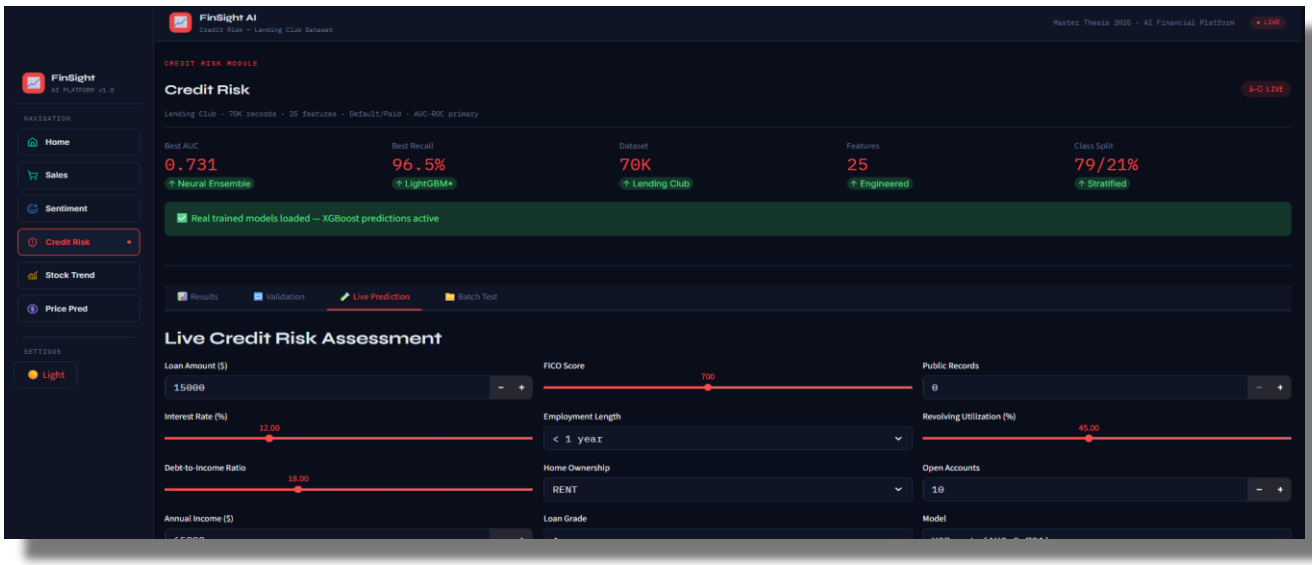


Fig. 4.40: FinSight Tab3 (Live Prediction) for Credit Risk Module.

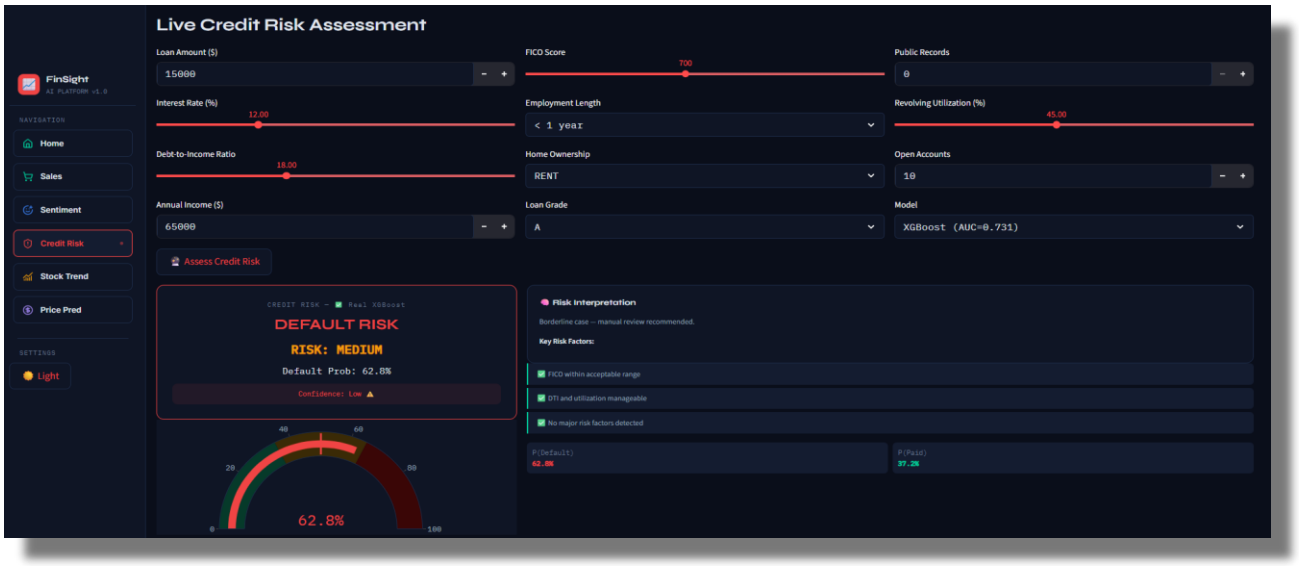


Fig. 4.41: FinSight Tab3 (Live Prediction Interpretation) for Credit Risk Module.

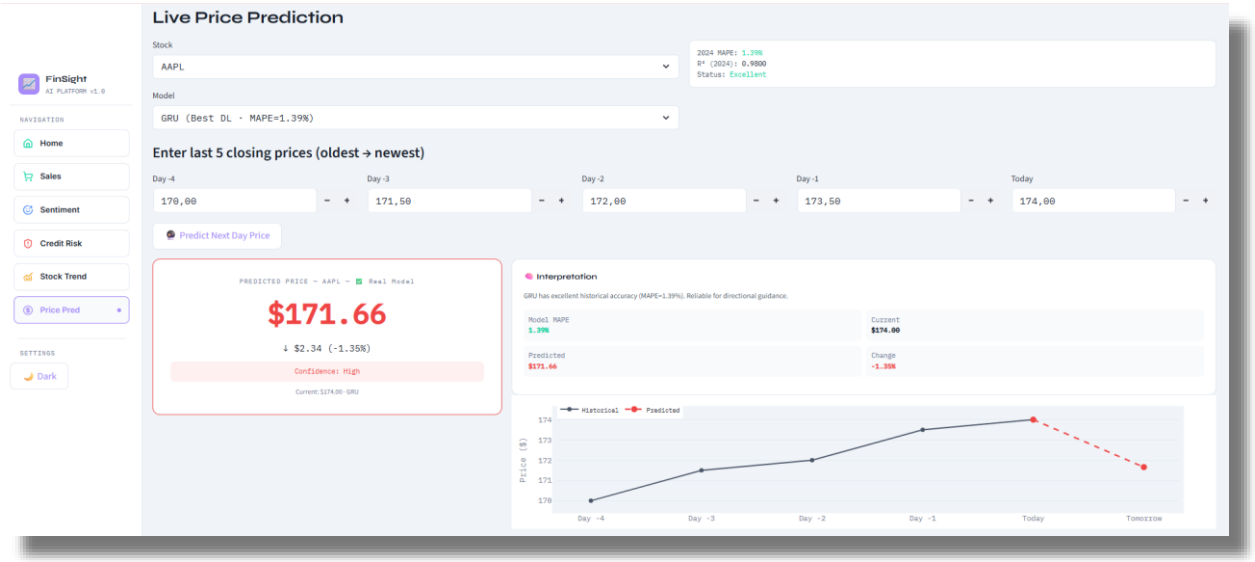


Fig. 4.42: FinSight Tab3 (Live Prediction Interpretation) for Price Module.

- **Tab 4 (Batch Test):** This tab extends the platform's capability from single-record inference to bulk prediction. The user uploads a CSV file containing multiple records; the required column names are displayed on screen, and a Download Sample CSV button provides a correctly formatted template. If the file includes a true label column, the platform computes accuracy automatically and displays it alongside the predictions. This tab is particularly useful for business users who need to classify an entire portfolio of orders, loan applications, or financial texts in a single operation rather than evaluating them one at a time.

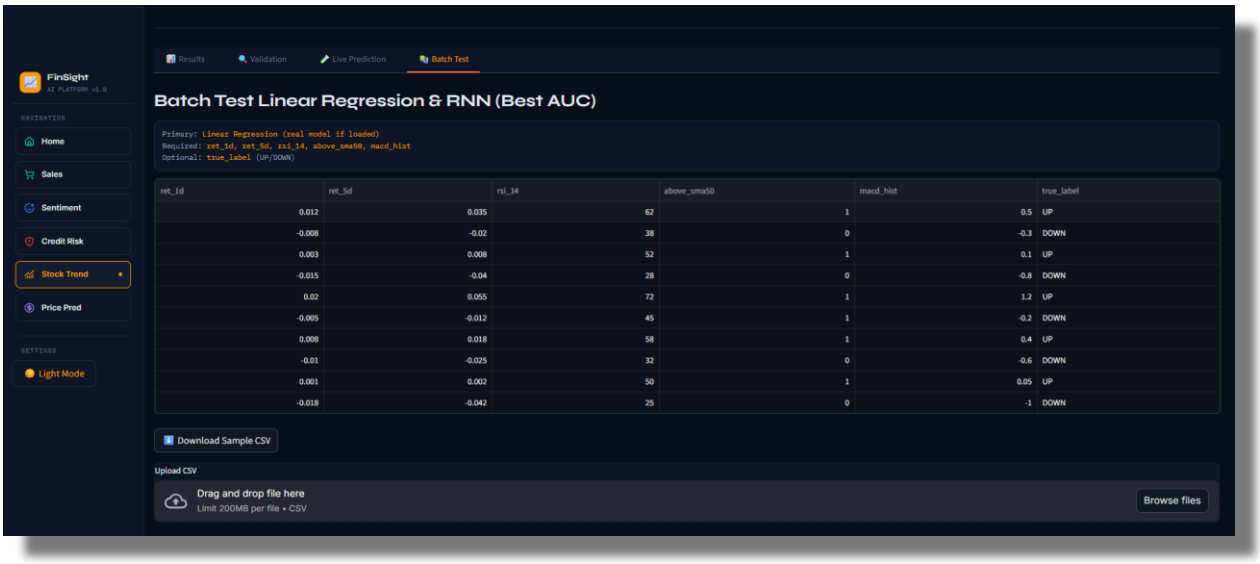


Fig. 4.43: FinSight Tab4 (Batch Test) for Stock Trend Module.

This design reflects the thesis's third objective: to bridge the gap between academic experimentation and practical deployment, making multi-model financial prediction accessible to users who do not require ML expertise to benefit from its outputs.

4.11. Advantages of the Proposed Approaches:

- **Multi-task unified Platform:** FinSight AI addresses five financial prediction tasks under a single experimental protocol with consistent preprocessing, training, and evaluation standards. This breadth enables the cross-task generalization analysis presented in this chapter which a contribution that task-isolated studies cannot replicate.
- **Rigorous leakage prevention:** A formal leakage audit was conducted for the sales module (identifying net_revenue with correlation=1.000 and revenue_proxy with correlation=0.987 as target-derived features). Temporal splitting without shuffling was enforced for all sequential tasks. Scalers were fitted exclusively on training data. These precautions address the most commonly reported methodological weaknesses in the published financial ML literature.
- **Honest metric reporting**: AUC-ROC and Recall are used as primary metrics for credit risk, correctly identifying the SVM as a failure model (80% accuracy, 1.1% recall) that would pass an accuracy-only evaluation. F1-macro treats all three sentiment classes equally, penalizing poor Negative class performance that F1-weighted conceals. The distinction between scaled R^2 and real MAPE for price

prediction correctly classifies Linear Regression as a naive baseline rather than a genuine predictor.

- **Practical deployment:** All trained models are serialized and served through an interactive Streamlit dashboard with live inference, cross-validation visualization, and batch prediction capabilities. This distinguishes the work from purely academic benchmarks that exist only in notebook cells.
- **Reproducibility:** All datasets are publicly available. All random seeds are fixed. All hyperparameters are fully specified. The experimental pipeline can be reproduced from the published notebooks without access to proprietary data or infrastructure.

4.12. Conclusion :

This chapter has presented the experimental results of FinSight AI across five financial prediction tasks, together with a cross-module analysis that a single-task study cannot provide. The findings are honest about both what the platform achieves and where it falls short. The most practically significant results are not always the most numerically impressive ones, the 99.45% sales accuracy is technically correct but reflects a structured dataset, while the 67.55% trend classification accuracy is theoretically meaningful precisely because it approaches the **EMH** ceiling. What the five modules collectively demonstrate is that rigorous multi-task evaluation, with appropriate metrics and honest leakage prevention, produces findings that are more reliable and more actionable than the optimistic single-task results that dominate the financial ML literature. The deployment of all results within FinSight AI completes the work's practical ambition: to make financial ML not just accurate on paper, but usable in practice.

CONCLUSION

This Project set out to answer a question that the financial ML literature had approached only partially: not whether machine learning works in finance because the evidence for that was already substantial, but when it works, for which tasks, and under what conditions it genuinely outperforms simpler alternatives. After training and evaluating several models across five financial prediction domains, the answer that emerges is more nuanced than the field's prevailing optimism about deep learning might suggest.

The results confirmed that deep learning excels where it has structural reasons to excel. The GRU's superior performance on price prediction on real data reflects its genuine ability to model non-linear temporal momentum patterns that a linear weighted average of past prices cannot capture. The LSTM's near-perfect sales classification reflects a dataset whose decision boundary is learnable by any sufficiently expressive model with adequate training data. These are real results, earned by real architectural advantages.

Perhaps the most practically significant finding is the one that the summary tables do not immediately convey: the models that appear best on headline metrics are sometimes the least useful, and the models that appear mediocre are sometimes the most reliable. The Neural Ensemble for credit risk achieves a lower accuracy than the SVM but is the only model that actually detects defaults at a rate useful for risk management. FinBERT zero-shot achieves a lower score than fine-tuned FinBERT but requires no training data, an important practical consideration when labeled domain data is scarce.

This work has limitations that are worth naming clearly. The market modules are evaluated on a single asset (AAPL), and generalization to other equities, asset classes, or international markets cannot be assumed. The credit risk module frames a complex multi-dimensional risk problem as a binary classification, and the models are not audited for demographic fairness. And the Financial News Sentiment, was not incorporated with market forecast data., a gap that a future version of this work should address.

Looking ahead, several directions emerge naturally from what was learned here. The most immediate is the integration of an online learning mechanism into FinSight AI a retraining pipeline that updates models as new data becomes available, addressing the distributional shift problem that defeated LSTM on trend classification. A second direction is a formal fairness audit of the credit risk module, using synthetic demographic data to

assess whether the engineered features introduce proxy discrimination. A third direction is the extension of the sentiment module to the Twitter Financial News dataset, enabling a cross-domain comparison between formal financial language (PhraseBank) and informal social media language (Twitter) a comparison that would provide insight into which linguistic register carries more predictive signal for market movements. Finally, the incorporation of reinforcement learning for dynamic portfolio allocation would complement the five supervised tasks with a genuinely sequential decision-making framework, completing the transition from prediction to action.

What this work contributes, above all, is a demonstration that rigorous multi-task evaluation with honest metrics, explicit leakage prevention, temporal validation, and practical deployment produces findings that single-task studies with optimistic reporting cannot. The financial ML field has no shortage of impressive accuracy numbers. It has a shortage of honest answers about when those numbers translate into useful decisions. This thesis is one attempt to provide a few of those answers.

On a personal level, this project was a valuable learning experience that strengthened my skills in machine learning, deep learning, financial data analysis, and software development it allowed me to gain a deeper understanding of financial markets, the factors that influence asset prices, credit risk assessment, and the role of sentiment in financial decision-making More importantly, this work helped me bridge the gap between artificial intelligence and finance, giving me valuable insight into how advanced technologies can be applied to solve real-world financial problems.

References

[1] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," *European Journal of Operational Research,* vol. 270, no. 2, pp. 654-669, 2018.

[2] O. Sezer, U. Gudelek and M. Ozbayoglu, "Financial time series forecasting with deep learning: A systematic literature review: 2005–2019," *Applied Soft Computing,* vol. 90, p. 106181, 2020.

[3] P. Malo, A. Sinha, P. Takala, P. Kohonen and J. Wallenius, "Good Debt or Bad Debt: Detecting Semantic Orientations in Economic Texts," *Journal of the American Society for Information Science and Technology,* 04 2014.

[4] N. Chawla, K. Bowyer, L. O. Hall and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research,* vol. 16, p. 321–357, 2002.

[5] X. Zhang and L. Yu, "Consumer credit risk assessment: A review from the state-of-the-art classification algorithms, data traits, and learning methods," *Expert Systems with Applications,* vol. 237, 2024.

[6] F. S. Mishkin, The Economics of Money, Banking, and Financial Markets, 12 ed., Pearson, 2018, p. 744.

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

[8] R. A. Brealey, S. C. Myers and F. Allen, Principles of Corporate Finance, 13 ed., McGraw-Hill, 2020.

[9] C. Institute, "Introduction to Commodities and Commodity Derivatives," 2025.

[10] S. Hägele, "Centralized exchanges vs. decentralized exchanges in cryptocurrency markets: A systematic literature review," *Electronic Markets,* vol. 34, p. 33, 2024.

[11] F. J. Fabozzi, The Handbook of Fixed Income Securities, 8 ed., McGraw-Hill, 2012.

[12] G. P. Zhang, "Time Series Forecasting Using a Hybrid ARIMA and Neural Network Model," *Neurocomputing,* vol. 50, p. 159–175, 2003.

[13] G. E. P. Box, G. M. Jenkins, G. C. Reinsel and G. M. Ljung, Time Series Analysis: Forecasting and Control, 5 ed., John Wiley & Sons, 2015.

[14] R. N. Mantegna and H. E. Stanley, An Introduction to Econophysics: Correlations and Complexity in Finance, Cambridge: Cambridge University Press, 2000.

[15] R. F. Engle, “Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation,” *Econometrica,* vol. 50, no. 4, p. 987–1007, 1982.

[16] J. Bollen, H. Mao and X.-J. Zeng, "Twitter Mood Predicts the Stock Market," *Journal of Computational Science,* vol. 2, 2010.

[17] W. Jiang, “Applications of Deep Learning in Stock Market Prediction: Recent Progress,” *Expert Systems with Applications,* vol. 184, p. 115537, 2021.

[18] Reddit, "Patterns Every Stock Trader Should Know," September 2021. [Online]. Available: https://www.reddit.com/r/pennystocks/comments/pva6kw/patterns_every_stock_trader_should_know/. [Accessed 30 May 2026].

[19] R. J. Hyndman and G. Athanasopoulos, Forecasting: Principles and Practice, 3 ed., OTexts, 2021.

[20] K. Bandara, C. Bergmeir and S. Smyl, “Forecasting across time series databases using recurrent neural networks,” *Expert Systems with Applications,* vol. 140, p. 112896, 2020.

[21] S. Lessmann, B. Baesens, H.-V. Seow and L. C. Thomas, “Benchmarking State-of-the-Art Classification Algorithms for Credit Scoring: An Update of Research,” *European Journal of Operational Research,* vol. 247, no. 1, p. 124–136, 2015.

[22] A. Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 3 ed., Sebastopol, CA: O'Reilly Media, 2022.

[23] L. Breiman, “Random Forests,” *Machine Learning,* vol. 45, no. 1, p. 5–32, 2001.

[24] T. Chen and C. Guestrin, “XGBoost: A Scalable Tree Boosting System,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, 2016.

[25] IBM, "What is Gradient Boosting?," [Online]. Available: https://www.ibm.com/think/topics/gradient-boosting. [Accessed 2026].

[26] V. N. Vapnik, The Nature of Statistical Learning Theory, New York: Springer, 1995.

[27] I. Goodfellow, Y. Bengio and A. Courville, Deep Learning, Cambridge, MA: MIT Press, 2016.

[28] J. L. Elman, “Finding Structure in Time,” *Cognitive Science,* vol. 14, no. 2, p. 179–211, 1990.

[29] S. Omkar, G. Rishikesh and P. Anurag, "Image Caption Generation Methodologies," in *International Research Journal of Engineering and Technology (IRJET),* April 2021.

[30] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Computation,* vol. 9, no. 8, p. 1735–1780, 1997.

[31] O. Calzone, "An Intuitive Explanation of LSTM," 21 February 2022. [Online]. Available: https://medium.com/@ottaviocalzone/an-intuitive-explanation-of-lstm-a035eb6ab42c. [Accessed 26 May 2026].

[32] M. Schuster and K. K. Paliwal, "Bidirectional Recurrent Neural Networks," *IEEE Transactions on Signal Processing,* vol. 45, no. 11, pp. 2673-2681, 1997.

[33] N. Elsayed, Z. Zaghloul, S. Azumah and C. Li, "Intrusion Detection System in Smart Home Network Using Bidirectional LSTM and Convolutional Neural Networks Hybrid Model," arXiv, 2021.

[34] G. Huang and C. Kou, "Financial Sentiment Analysis Using BERT and BiLSTM," in *IEEE International Conference on Big Data*, 2023.

[35] K. Cho, B. van Merriënboer, Ç. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk and Y. Bengio, "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, 2014.

[36] GeeksforGeeks, "Gated Recurrent Unit Networks," 11 5 2026. [Online]. Available: https://www.geeksforgeeks.org/machine-learning/gated-recurrent-unit-networks/. [Accessed 2 June 2026].

[37] . M. Muhammad , J. Zaffry Mohd, K. Kamarulzaman, A. Abdul, K. Latifah, Z. Ammar, S. Z. Syed Muhammad Mamduh and A. Abdulnasser, "Application of Deep Neural Network for Gas Source Localization in an Indoor Environment," *International Journal of Computers Communications & Control,* vol. 18, no. 3, 2023.

[38] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser and I. Polosukhin, "Attention Is All You Need," in *Advances in Neural Information Processing Systems*, 2017.

[39] D. Araci, "FinBERT: Financial Sentiment Analysis with Pre-trained Language Models," arXiv, 2019.

[40] D. Jurafsky and J. H. Martin, Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, with Language Models, 3 ed., Stanford University, 2026.

[41] J. McCarthy, M. L. Minsky, N. Rochester and C. E. Shannon, "A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence," *AI Magazine,* vol. 27, no. 4, p. 12–14, 2006.

[42] K. Bharti, "The Revolutionary Impact of AI and Machine Learning in Finance: Opportunities, Challenges and the Future," 23 may 2025. [Online]. Available: https://medium.com/@kushagrabharti/the-revolutionary-impact-of-ai-and-machine-learning-in-finance-opportunities-challenges-and-the-dbebb1190045. [Accessed 30 may 2026].

[43] R. N. Mantegna, “Hierarchical Structure in Financial Markets,” *The European Physical Journal B,* vol. 11, no. 1, p. 193–197, 1999.

[44] T. Kohonen, Self-Organizing Maps, 3 ed., Berlin / Heidelberg: Springer, 2001.

[45] F. T. Liu, K. M. Ting and Z.-H. Zhou, “Isolation Forest,” in *2008 Eighth IEEE International Conference on Data Mining (ICDM)*, Pisa, Italy, 2008.

[46] D. M. Blei, A. Y. Ng and M. I. Jordan, “Latent Dirichlet Allocation,” *Journal of Machine Learning Research,* vol. 3, p. 993–1022, 2003.

[47] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, 2 ed., Cambridge, MA: MIT Press, 2018.

[48] V. Mnih, “Human-level control through deep reinforcement learning,” *Nature,* vol. 518, p. 529–533, 2015.

[49] J. Schulman, F. Wolski, P. Dhariwal, A. Radford and O. Klimov, “Proximal Policy Optimization Algorithms,” 2017.

[50] C. S. Bojer and J. P. Meldgaard, “Kaggle Forecasting Competitions: An Overlooked Learning Opportunity,” *International Journal of Forecasting,* vol. 37, no. 2, p. 587–603, 2021.

[51] H. Kvamme, N. Sellereite, K. Aas and S. Sjursen, “Predicting Mortgage Default Using Convolutional Neural Networks,” *Expert Systems with Applications,* vol. 102, p. 217, 2018.

[52] M. Schmitt, “Deep Learning vs. Gradient Boosting: Benchmarking State-of-the-Art Machine Learning Algorithms for Credit Scoring,” arXiv, 2022.

[53] T. Loughran and B. McDonald, “When Is a Liability Not a Liability? Textual Analysis, Dictionaries, and 10-Ks,” *Journal of Finance,* vol. 66, no. 1, p. 35–65, 2011.

[54] Y. Yang, M. Uy and A. Huang, “FinBERT: A Pretrained Language Model for Financial Communications,” arXiv, 2020.

[55] S. Gaurang, D. Deepak Sudhakar, M. B. Anupkumar, T. D. Sarika, D. Deepak and B. Subraya Krishna, "Forecasting Stock Market Prices Using Machine Learning and Deep Learning Models: A Systematic Review, Performance Analysis and Discussion of Implications," *International Journal of Financial Studies,* vol. 11, no. 3, p. 94, 2023.

[56] Yahoo Finance, “Yahoo Finance,” [Online]. Available: https://finance.yahoo.com. [Accessed 20 May 2026].

[57] LendingClub Corporation, "LendingClub Closes Acquisition of Radius Bancorp," 1 February 2021. [Online]. Available: https://ir.lendingclub.com/news/news-details/2021/LendingClub-Closes-Acquisition-of-Radius-Bancorp/. [Accessed 30 may 2026].

[58] M. Abadi, «TensorFlow: A System for Large-Scale Machine Learning,» chez *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, Savannah, Georgia, USA, 2016.

[59] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” in *International Conference on Learning Representations (ICLR)*, 2015.

[60] W. McKinney, “Data Structures for Statistical Computing in Python,” in *9th Python in Science Conference (SciPy 2010)*, Austin, Texas, USA, 2010.

[61] C. R. Harris, "Array Programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357-362, 2020.

[62] F. Pedregosa, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825-2830, 2011.

[63] G. Ke, “LightGBM: A Highly Efficient Gradient Boosting Decision Tree,” in *Advances in Neural Information Processing Systems (NeurIPS 2017)*, 2017.

[64] M. Al Ridhawi,, "Stock Market Prediction Through Sentiment Analysis of Social-Media and Financial Stock Data Using Machine Learning," Ottawa, Ontario, 2021.

[65] D. C. Montgomery, E. A. Peck and G. G. Vining, Introduction to Linear Regression Analysis, 5 ed., Hoboken, NJ: Wiley, 2012.

[66] T. M. Mitchell, Machine Learning, New York: McGraw-Hill, 1997.