A Plug-and-Play Machine Learning Framework for Diverse Applications in Finance [*]

Chen Rui, Chen Xingnuo, Liang Shihao, Liu Zhengze, Zhong Ruohan
The University of Hong Kong
COMP 7409 Group 7

Abstract

In the financial sector, machine learning (ML) algorithms play a pivotal role in numerous applications, including risk management, asset management, market analysis, and the development of trading strategies. Despite the prevalence of these algorithms, current ML frameworks, such as sklearn, though rich in algorithmic variety and offering user-friendly training and prediction pipelines, often fall short in providing comprehensive data preprocessing methods and modules. This limitation, which necessitates specialized domain knowledge, poses challenges in effectively adapting these frameworks to practical financial contexts. To address this gap, we introduce a novel framework designed to seamlessly integrate into various financial applications. It features an intuitive pipeline complemented by a plug-and-play data module dedicated to data engineering tasks. Our framework's versatility is demonstrated through extensive experiments across a spectrum of financial applications, confirming its capability to robustly support diverse algorithms tailored to specific financial tasks. Furthermore, our framework encompasses comprehensive evaluation metrics and advanced visualization tools. These features facilitate in-depth analysis of results, thereby enhancing the understanding and interpretability of outcomes in financial applications. This comprehensive approach positions our framework as a significant advancement in the application of machine learning in finance and trading. Code are available at GitHub repository: https://github.com/pooruss/ML-Framework-for-Diverse-Applications-in-Trading-and-Finance

***K*eywords** Machine learning · Finance applications · Machine learning framework

1 Introduction

1.1 Background and Motivation

In recent years, with the rapid development of artificial intelligence in information processing applications, AI methods have been applied across various domains such as business, engineering, management, science, and finance.[1] Particularly in the financial sector, the advancement of intelligence and modernization has led to the widespread application of machine learning as a powerful tool, showcasing immense potential. Machine learning in the financial domain involves the utilization of algorithms and statistical models to analyze extensive financial data, generating insights without explicit instructions. [2] Through the training of these algorithms and models, a deep understanding of the patterns and trends in the financial market is obtained, creating valuable insights that assist financial institutions in making more accurate predictions and decisions. Machine learning has become an indispensable key tool in the realms of finance and trade.

Despite the numerous benefits that machine learning brings to the financial sector, there are also significant drawbacks that must be considered. As the market evolves and becomes increasingly complex, the demand for robust machine-learning solutions tailored to the subtle nuances of the financial industry becomes crucial. Existing machine learning tools often fall short of meeting the specific needs of the financial sector, lacking functionalities and solutions

[*] *Authorship: The authors wish to emphasize that the order of authorship is purely alphabetical and does not reflect the extent of their contributions. Each author has contributed equally to this work, playing an integral role in its development and execution.*

tailored to this domain, resulting in a gap in addressing the unique challenges posed by financial data.

Furthermore, another drawback of machine learning in the financial domain is the lack of model interpretability. [3] Machine learning algorithms typically exhibit complex and challenging-to-understand characteristics, with many available solutions being either code-intensive or highly encapsulated. This complexity poses challenges for both machine learning beginners who may not be proficient in programming and financial decision-makers in understanding the reasoning behind specific decisions.

Recognizing these limitations, our project aims to provide a tailored and highly interpretable set of machine learning algorithms and resources for professionals in the financial industry and machine learning beginners. By eliminating the need for complex coding and simplifying the entire process from data to algorithms, we intend to foster the integration and development of the financial discipline with the field of machine learning. This initiative aims to make it easier for financial practitioners to harness the advantages of machine learning while increasing the interpretability of algorithms, making it easier for decision-makers to understand and trust the results generated by models. This effort aims to meet the growing demands of the financial industry and propel further development in financial intelligence.

1.2 Project Achievements

In the present project, our commitment lies in developing a custom-designed Python machine-learning framework for the financial domain, aiming to provide a plug-and-play solution for industry professionals and novice machine-learning practitioners. The core contributions of this framework are articulated in the following aspects:

1. **Data Classification and Customization**: We systematically categorize financial data into five major classes, including risk management and financial fraud, to better cater to diverse industry needs and provide a clearer direction for the selection and application of machine learning algorithms.
2. **Efficient Implementation of Machine Learning Algorithms**: We have implemented ten efficient machine learning algorithms, including PCA and Decision Trees, etc., showcasing both efficiency and a high degree of interpretability. These algorithms have proven effective in addressing various demands within the financial domain.
3. **Optimization of Data Characteristics**: Recognizing the unique characteristics of financial data in different scenarios, we have optimized each class of financial data. It involves preprocessing, model adjustments, and so on to ensure optimal performance of machine learning algorithms across different problem contexts.
4. **Diversified Solutions**: Tailored to each class of financial problems, we offer a variety of operational machine learning algorithms, enabling users to choose the most suitable method based on specific requirements.

2 Framework

2.1 Dataset

Risk Management

- **Credit Scoring Analysis**: A sophisticated machine learning approach is utilized to evaluate borrower credit risks, crucial for loan approval decisions. This includes analyzing historical borrowing records, repayment behaviors, and financial transactions.
- **Market Risk Estimation**: Advanced regression techniques forecast financial market volatilities, aiding investors in decision-making. Models integrate market volatility indices, economic indicators, and global financial trends.
- **Credit Risk Assessment**: Classification and regression techniques gauge potential debt defaults, essential for risk management in financial institutions. This involves scrutinizing creditworthiness, considering macroeconomic conditions and financial history.

Investment Management

- **Strategic Portfolio Construction**: Algorithms design optimized investment portfolios to maximize returns or minimize risks. These maintain a balance between risk and reward, ensuring diversification and market responsiveness.

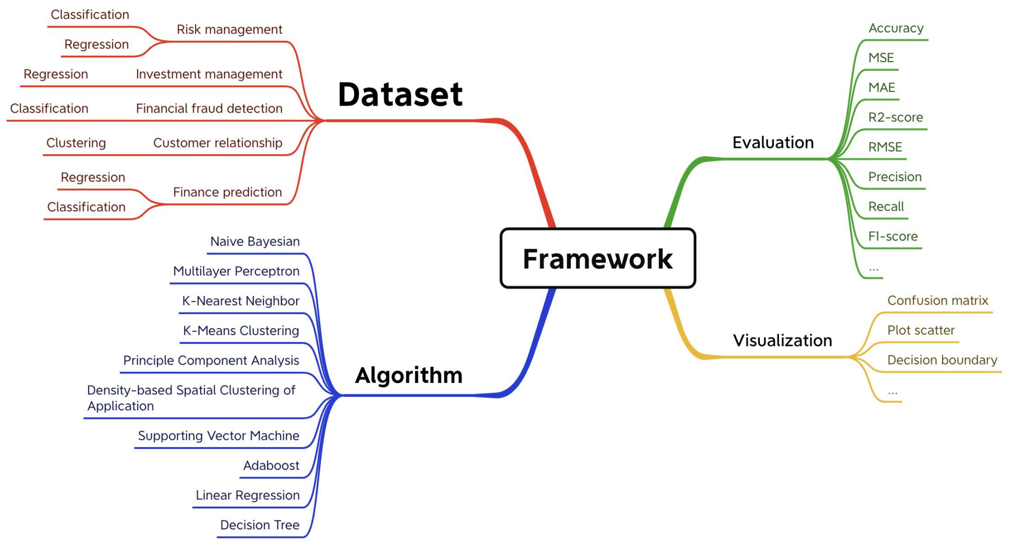

Figure 1: Framework overview.

- **Market Trend Prediction**: Machine learning models predict stock, commodity, and asset price trends. The analysis includes market data, news sentiment, and historical pricing patterns.

Financial Fraud Detection

- **Credit Card Fraud Identification**: Models detect fraudulent activities in credit card transactions, recognizing anomalies against historical patterns.
- **Detection of Anomalous Transactions**: Systems monitor unusual financial activities, identifying potential fraud or money laundering through aberrant transaction patterns.

Customer Relationship

- **Customer Segmentation**: Machine learning categorizes customers to understand needs and provide personalized services, analyzing buying patterns, preferences, and demographics.
- **Predicting Customer Churn**: Algorithms predict customer attrition, focusing on factors leading to turnover and strategies for customer retention.

Finance Prediction

- **Analyzing Financial Statements**: Techniques dissect and forecast a company's financial health and performance, analyzing balance sheets, income statements, and cash flow.
- **Projection of Profit Margins**: Profit forecasts are based on historical data, market trends, and economic indicators, analyzing company metrics and industry trends.

2.2 Algorithm

In this section, we introduce the supported algorithms in our framework.

2.2.1 Adaboost (Adaptive Boosting)

Algorithm Principles Adaptive Boosting (AdaBoost) is an ensemble method that combines multiple weak learners, typically decision trees, to create a strong classifier. The method focuses on correctly classifying the instances where previous learners failed. Specifically,

- *Weak Learners*: Simple classifiers which are only slightly better than random guessing.
- *Weighted Training Data*: Each instance in the training set is assigned a weight that indicates its importance in classification.

- *Iterative Improvement*: Each successive learner focuses more on the instances that were misclassified by previous learners.

Algorithm Workflow

1. **Initialize Weights**: Initialize the weights of each instance in the training dataset equally.
2. **Train Weak Learners**: Sequentially train weak learners on the weighted training data.
3. **Adjust Weights**: Increase the weights of misclassified instances to focus subsequent learners more on these instances.
4. **Combine Learners**: Aggregate the weak learners to form a strong classifier.
5. **Compute Final Model**: Determine the final output based on the weighted sum of the weak learners' predictions.
6. **Model Evaluation**: Assess the performance of the final model and adjust parameters if necessary.

2.2.2 DBSCAN

Algorithm Principles Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a density-based clustering algorithm that identifies core points, boundary points, and noise points by defining the density within a neighborhood. It groups closely packed data points together. Specifically,

- Core Points: A data point is considered a core point if it has at least *min_samples* data points within an *eps* distance.
- Boundary Points: A data point is a boundary point if it is within an *eps* distance of a core point but does not have enough neighbors to qualify as a core point itself.
- Noise Points: Data points that are neither core points nor boundary points are labeled as noise.

Algorithm Workflow

1. **Initialize Parameters**: In the initialization phase, set two key parameters of the DBSCAN algorithm, namely the neighborhood radius (*eps*) and the minimum number of samples.
2. **Compute Distance Matrix**: For each sample in the dataset, calculate its distance to all other samples.
3. **Find Core Points**: Calculate distances between points and identify core points based on the specified minimum samples parameter.
4. **Build Clusters**: Starting from core points, label core points and use density reachability to assign samples to different clusters.
5. **Label Noise Points**: Mark all samples that do not belong to any cluster as noise points.
6. **Output Cluster Results**: Return the cluster assignment labels for each sample.

2.2.3 ID3 Decision Tree

Algorithm Principles Decision Tree is a common class of machine learning methods and a very common classification method. It is a supervised learning method. Decision trees are based on a tree structure to make decisions. Generally, a decision tree contains a root node, several internal nodes, and several leaf nodes. The purpose of decision tree learning is to generate a decision tree with strong generalization ability, that is, the ability to process unknown examples. Its basic process follows a simple and intuitive "divide and conquer" strategy.

Each internal node represents a judgment on an attribute:

- Each branch represents the output of a judgment result.
- Each leaf node represents a classification result.
- The root node contains the complete set of samples

The core of the ID3 algorithm is to select the features to be divided based on information gain, and then recursively construct a decision tree.

Algorithm Workflow

1. **Select features**: Starting from the root node, calculate the information gain of all possible features, and select the feature with the largest information gain as the node dividing feature.
2. **Create child nodes**: Create a corresponding number of child nodes according to the different values of the selected feature, where the number of child nodes depends on the number of values of the feature.
3. **Build a decision tree**: Repeat the recursive selection of features and create sub-nodes on each node to build a decision tree.
4. **Obtain the decision tree**: Repeat the above steps until features cannot be selected or the categories are the same. When features cannot be selected, stop dividing, mark the current node as a leaf node, and set the leaf node to the most common category in the data set. Obtain a decision tree.

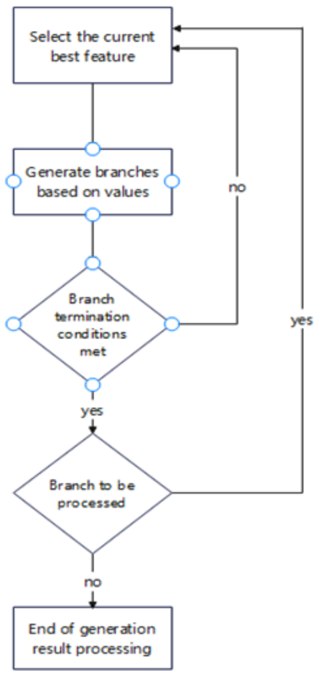
Select the current best feature
Generate branches based on values
no
Branch termination conditions met
yes
yes
Branch to be processed
no
End of generation result processing

Figure 2: The Workflow of ID3 Decision Tree

2.2.4 K-Means

Algorithm Principles K-means clustering is a method of vector quantization, that aims to partition given data objects into k clusters in which each object belongs to the cluster with the nearest centroid of the cluster. The K-Means is an unsupervised algorithm, resulting in a partitioning of the data objects into k clusters based on their similarity.

Algorithm Workflow

1. **Parameter selection**: selection of the number of clusters.
2. **Initialization**: select k points as the initial centroids. Initial centroids are often chosen randomly.
3. **Optimization**: repeatedly, form k clusters by assigning all objects to the closest centroid and recomputing the centroid of each cluster, until the centroids don't change.

2.2.5 K-Nearest Neighbor

Algorithm Principles The k-nearest neighbors' algorithm is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions. It is also a "lazy learning" model, which means it only conducts computation when faced with a classification or prediction. Before that, it stores all its training data, relying on much memory. The initial ideas were from Evelyn Fix and Joseph Hodges[1] and the concept was expanded by Thomas

Cover[2]. In our experiment, we only use the k-nearest neighbor algorithm to do classification. Overall, the goal of the k-nearest neighbor algorithm is to identify the nearest neighbors of a given query point. It is crucial for this algorithm to define its “k” value and choose a decision metric. One of those commonly used decision metrics is Manhattan distance, which is also used in our experiment.

Algorithm Workflow

1. **Parameter selection**: the best choice of k depends on the data. Generally, smaller k values make classification results more distinct but more susceptible to outliers at the same time.
2. **Distance measure**: given an unknown record, compute distances to other records.
3. **Identification**: identify k nearest neighbors to determine the class label of the unknown record (e.g., by taking a majority vote)

2.2.6 Linear Regression

Algorithm Principles Regression is used to predict the relationship between input and output variables, especially when the value of the input variable changes, and the value of the output variable also changes. A regression model is a function that represents the mapping from input variables to output variables. The purpose of regression is to predict the target value of an array type.

Linear regression is based on known data sets and uses the gradient descent method to train parameter $w(y = wx + b)$ of the linear regression model. Through training, the data in the data will meet expectations, thereby using the linear regression model to predict unknown categories of data.

Algorithm Workflow

1. **Parameter initialization**: Set the learning rate (*lr*) and the number of iterations (*n_iters*) as the hyperparameters of the algorithm, and initialize the weights (weights) and biases (bias).
2. **Fit**: Use gradient descent to iteratively adjust weights and biases to minimize the mean square error loss function.
3. **Predict**: After training is completed, use the obtained weights and biases to predict new input data.

2.2.7 Multi-layer Perceptron

Algorithm Principles Multilayer perceptron (MLP) is a deep learning model based on feedforward neural networks consisting of multiple neuron layers, where each neuron layer is fully connected to the previous layer. A multilayer perceptual machine generally consists of three layers: an input layer, a hidden layer and an output layer. Where the hidden layer consists of weights (generally represented as a matrix) and biases. When a nonlinear hidden layer is introduced, it is theoretically possible to fit arbitrary functions as long as there are enough hidden nodes and the more hidden layers there are, the easier it is to fit more complex functions as well.

In addition to information transfer and processing through weights, activation functions are also involved in the process. The activation function can act as a nonlinear mapping, which can limit the output amplitude of the neuron to a certain range. Among them, the most commonly used activation function is the Sigmoid function, which can map the number of $(-\infty, +\infty)$ to the range of $(0, 1)$.

Algorithm Workflow

1. **Initialization**: randomly initialize model weights W and bias b
2. **Forward propagation**: pass the input value X into the model for forward propagation to get the probability of the result.
3. **Calculate the loss function**: use the loss function to calculate the difference between the predicted value and the actual value to get the loss value J (commonly used logistic regression loss functions include cross entropy function, etc., and we chose the mean square error function as the loss function in our implementation)
4. **Calculate the gradient**: Calculate the loss gradient according to the model weights.
5. **Update using gradient descent**: use gradient descent to do backpropagation to update the model weights. The updated weights will reduce the probability of outputting an incorrect result, thus increasing the probability of outputting the correct category.

6. **Iteration**: Repeat the above steps until the classification results meet the requirements.

2.2.8 Naive Bayesian

Algorithm Principles The naive Bayes algorithm is a classification method based on the Bayes theorem and the feature conditional independence assumption. For a given training dataset, the joint probability distribution of inputs and outputs is first learned based on the feature conditional independence assumption. Then based on this model, the Bayes theorem is used to find the output with the highest posterior probability y for a given input X. Unlike other classifiers, the naive Bayes is a classification algorithm based on probability theory.

In this case, the Bayesian formula is shown below. The Bayes formula describes the relationship between prior probability, posterior probability and likelihood probability. For a given sample, the problem of estimating the posterior probability $P(c|X)$ translates into how to estimate the prior probability $P(c)$ and the likelihood probability $P(X|c)$ based on the training data.

$$P(c|X) = \frac{P(X|c)P(c)}{P(X)}$$

To compute the prior and likelihood probabilities, $P(c)$ can be estimated by the frequency of occurrence of each type of sample when the training set contains sufficient independent and identically distributed samples, according to the law of large numbers. In contrast, the likelihood probability $P(X|c)$ is computed based on the conditional independence assumption and the method of great likelihood estimation.

Algorithm Workflow

1. **Calculate the prior probability**: Based on the training data, count the occurrence probability $P(c_i)$ of each category c_i, i.e., prior probability
2. **Calculate the likelihood probability**: Calculate the conditional probability $P(X_j|C_i)$ of each feature attribute for each category, i.e., the likelihood probability.
3. **Process the test set data**: for the given data X to be categorized, calculate $P(X|c_i) * P(c_i)$ for each category separately.
4. **Getting the classification result**: the category with the largest probability is taken as the final classification result.

It is worth noting that the above process is a description of discrete features. For continuous numerical features such as financial data, it is generally assumed that the attribute values in the same category obey a normal distribution, and the mean and variance of the distribution of attribute values under the current category and the attribute in which they are located are first calculated, and then the probability density of the current attribute value in this distribution is calculated as follows. In order to adapt to our target domain, which is the financial domain, what we have implemented in the code is the likelihood probability calculation for continuous features.

$$P(X_i|c) = \frac{1}{\sqrt{2\pi}\sigma_{c,i}} exp(-\frac{(x_i - \mu_{c,i})^2}{2\sigma_{c,i}^2})$$

2.2.9 Principle Component Analysis

Algorithm Principles Principal Component Analysis (PCA) is a commonly used dimensionality reduction technique in data analysis and machine learning. PCA captures the linear combinations of the original features that exhibit the most significant variation in the data. It transforms high-dimensional data into a lower-dimensional form, representing the data in terms of its principal components.

Algorithm Workflow

1. **Standardization and Data Centering**: Data is standardized before input, and centering is performed by calculating the mean of the data to eliminate translation in the data.
2. **Covariance Matrix Computation**: After standardization, the covariance matrix is computed to understand the relationships between all feature pairs in the dataset.
3. **Eigenvalue Decomposition**: The next step involves the eigenvalue decomposition of the covariance matrix, resulting in a set of eigenvalues and their corresponding eigenvectors. Eigenvalues represent the variance explained by each principal component, while eigenvectors indicate the directions in the original feature space that maximize variance.

4. **Selection of Principal Components**: Sorting the eigenvalues in descending order, the eigenvector corresponding to the largest eigenvalue is the first principal component, the eigenvector corresponding to the second largest eigenvalue is the second principal component, and so on. The desired number of principal components (k) is retained.
5. **Projection**: Finally, the original data is projected onto the lower-dimensional space defined by the selected principal components.

2.2.10 SVM (Support Vector Machine)

Algorithm Principles Support Vector Machine (SVM) is a supervised learning algorithm used for classification and regression analysis. It creates a hyperplane or a set of hyperplanes in a high-dimensional space to separate different classes. Specifically,

- *Margin*: The main goal of SVM is to maximize the margin between the data points of different classes.
- *Support Vectors*: Data points closest to the hyperplane which influence the position and orientation of the hyperplane.
- *Kernel Trick*: SVM can use a kernel function to transform the data into a higher dimensional space to make it easier to separate the data linearly.

Algorithm Workflow

1. **Choose Kernel and Parameters**: Select an appropriate kernel (e.g., linear, polynomial, RBF) and parameters for the SVM model.
2. **Transform Data**: If a non-linear kernel is used, transform the data into a higher-dimensional space.
3. **Construct Hyperplane**: Identify the optimal hyperplane that maximizes the margin between different classes.
4. **Identify Support Vectors**: Determine the support vectors that are critical in defining the hyperplane.
5. **Model Training**: Train the SVM model using the training dataset.
6. **Classification/Regression**: Use the trained model for classification or regression on new data.

2.3 Evaluation

Accuracy Accuracy measures the proportion of correct predictions made by the model out of all predictions. It is a fundamental metric for classification problems, providing a general idea of the model's performance.

Precision Precision evaluates the proportion of true positive predictions in the positive class. It is crucial in scenarios where the cost of false positives is high, emphasizing the quality of the positive predictions.

Recall Recall, or sensitivity, measures the proportion of actual positives correctly identified. It is particularly important in cases where missing a positive instance (false negative) has serious implications.

F1-score The F1-score is the harmonic mean of precision and recall. This metric is useful when seeking a balance between precision and recall, especially in uneven class distributions.

MSE Mean Squared Error (MSE) measures the average of the squares of the errors—that is, the average squared difference between the estimated values and the actual value. It is widely used in regression analysis.

MAE Mean Absolute Error (MAE) is a measure of errors between paired observations expressing the same phenomenon. It calculates the average absolute difference between the predicted values and observed values.

RMSE Root Mean Squared Error (RMSE) is the square root of the mean of the squared differences between predicted and actual values. It provides a measure of how far the predictions deviate from the actual values.

R2-score The R2-score, or coefficient of determination, measures the proportion of the variance in the dependent variable that is predictable from the independent variables. It indicates the goodness of fit of the model.

2.4 Visualization

Confusion Matrix A confusion matrix is a visualization tool typically used in supervised learning. It is a table used to describe the performance of a classification model on a set of test data for which the true values are known.

Plot Scatter Scatter plots are used to observe relationships between variables. In the context of machine learning, scatter plots can help visualize the distribution of data points and the relationship between variables, aiding in identifying patterns or trends.

Decision Boundary A decision boundary is a surface that separates data points belonging to different class labels. It is used in classification problems to help visualize the model's classification logic, showing where the algorithm divides or 'decides' between classes.

2.5 Pipeline

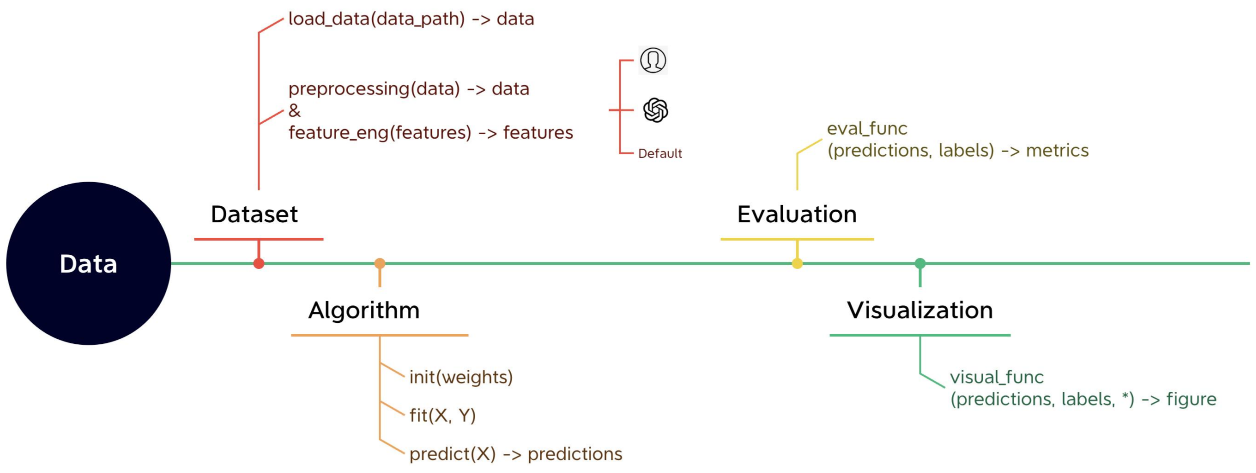
load_data(data_path) -> data
preprocessing(data) -> data
&
feature_eng(features) -> features
Default
eval_func
(predictions, labels) -> metrics
Data
Dataset
Evaluation
Algorithm
Visualization
init(weights)
fit(X, Y)
predict(X) -> predictions
visual_func
(predictions, labels, *) -> figure

Figure 3: Pipeline overview.

Our pipeline-based framework is composed of four principal components: data loading and preprocessing, algorithm initialization and training, evaluation, and visualization. Each component is described as follows:

Data Loading and Preprocessing. At the beginning of the pipeline, data is loaded and preprocessed:

- `load_data(data_path)`: This function takes the path to the dataset and returns the loaded data.
- `preprocessing(data)`: It processes the raw data to prepare it for feature extraction or model input.
- `feature_eng(features)`: Performs feature engineering to enhance the algorithm's performance.

Algorithm. The algorithm's life cycle is as follows:

- `init(weights)`: Initializes the model's weights, setting the stage for training.
- `fit(X, Y)`: Trains the model on the input features `X` with corresponding labels `Y`.
- `predict(X)`: Generates predictions for the input features `X`.

Evaluation. The model's performance is assessed using the following function:

- `eval_func(predictions, labels)`: Evaluates the predictions against the true labels to compute performance metrics.

Visualization. Finally, results are visualized for better interpretation:

- `visual_func(predictions, labels, '-')`: Generates figures to visualize the comparison between predictions and actual labels, facilitating interpretation of the model's performance.

3 Experiments

Introducing 5 common financial topics by focusing on relative data sets.

3.1 Datasets

3.1.1 Financial fraud detection

Credit card fraud detection

The dataset contains transactions made by credit cards in September 2013 by European cardholders, presenting transactions occurred in two days, where 492 frauds appeared out of 284,807 transactions. As the result of Principal Component Analysis (PCA), all variables are numerical, except two features, 'Time' and 'Amount'. This measure is to protect user identities and sensitive features.

Source: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

3.1.2 Customer relationship management

Credit Card Customer Churn Prediction
The dataset collects 10,000 data on whether credit card users are churning or not in three countries, France, Germany and Spain, and the statistics include 10 indicators such as credit score, age, and tenure. This dataset can be used to determine whether a user will be churned based on basic information about the user, so that user groups can be segmented and future behaviours can be predicted.

It is worth mentioning that for discrete features such as country and gender, one-hot coding technique is used to convert the discrete classification labels into binary vectors before training and testing when pre-processing the data. For example, the three categories of countries within countries could be coded as $[1, 0, 0]$, $[0, 1, 0]$ and $[1, 0, 0]$.

Key Features:

- ***CreditScore*** : The user's credit score, whose value is an integer between 350 and 850.
- ***EstimatedSalary***: Estimated income of users
- ***age***: Contains the age information of the customer.

Source: https://www.kaggle.com/datasets/rjmanoj/credit-card-customer-churn-prediction/

3.1.3 Financial forecast

Profit Prediction
This is a Data set for the Stock Price of Netflix, containing open, high, low, closed and volume data. This Data set starts from 2002 to 2022. It was collected from Yahoo Finance.

Key Features:

- ***Open*** : It is the price at which the financial security opens in the market when trading begins.
- ***Close***: Closing price generally refers to the last price at which a stock trades during a regular trading session.
- ***High***: The high is the highest price at which a stock is traded during a period.
- ***Low***: The low is the lowest price at which a stock is traded during a period.
- ***Volume***: Volume measures the number of shares traded in a stock or contracts traded in futures or options.

Source: https://www.kaggle.com/datasets/meetnagadia/netflix-stock-price-data-set-20022022/

3.1.4 Risk management

Credit Risk Dataset
This data set contains information about customers, covering nearly two thousand uniform statistics on income, age, previous loan amounts, and historical defaults. The data set can be used for risk assessment, credit scoring and other financial-related analysis to help financial institutions better understand customers' credit profile and repayment ability.

Key Features:

- ***clientid*** **(Customer ID)**: Contains the unique identifier of the customer.
- ***income***: Contains the income distribution statistics of customers, usually expressed in monetary units (such as US dollars).
- **age**: Contains the age information of the customer, especially the age of the defaulting customer.
- ***loan*** **(loan amount)**: Contains the customer's current loan amount, which may indicate an outstanding loan.
- ***default*** **(default record)**: The customer's historical default record, is used to record statistical data on whether the customer has failed to repay the loan on time or has experienced other defaults.

Source: https://www.kaggle.com/datasets/urstrulyvikas/lending-club-loan-data-analysis

3.1.5 Investment and Asset Management

Boston House Prices
This dataset describes various attributes of Boston suburbs or towns, sourced from the Boston Standard Metropolitan Statistical Area (SMSA) in 1970. It encompasses factors such as crime rate, property-tax rates, and demographic indicators like the lower status percentage of the population. The dataset is applicable in addressing questions related to real estate and housing market analysis, the prediction of market trends, urban planning, and socioeconomic research within the financial domain.

Key Features:

- ***ZN***: proportion of residential land zoned for lots over 25,000 sq. ft.
- ***INDUS***: proportion of non-retail business acres per town
- ***CHAS***: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
- ***NOX***: nitric oxides concentration (parts per 10 million)
- ***RM***: average number of rooms per dwelling
- ***AGE***: proportion of owner-occupied units built before 1940
- ***DIS***: weighted distances to five Boston employment centers
- ***RAD***: index of accessibility to radial highways
- ***TAX***: full-value property-tax rate per $10,000
- ***PTRATIO***: pupil-teacher ratio by town
- ***MEDV***: Median value of owner-occupied homes in $1000s
- ***B***: 1000(Bk0.63)2 where Bk is the proportion of blacks by town
- ***LSTAT***: % lower status of the population

Source: https://www.kaggle.com/datasets/vikrishnan/boston-house-prices

3.2 Evaluation

3.2.1 Cluster Distance

1. Average Intra-cluster Distance
   - The Average Intra-cluster Distance[4] measures the average distance between data points within the same cluster. A smaller Average Intra-cluster Distance typically indicates better clustering quality, as data points within the same cluster are closer together, indicating higher compactness.
   - This metric can be used to assess the tightness of data points within a cluster, helping identify whether there is overlap between clusters or if data points within a cluster are scattered.
2. Inter-cluster Distance
   - The Inter-cluster Distance[4] measures the average distance between data points of different clusters. A larger Inter-cluster Distance usually indicates better clustering quality, as there is a higher separation between different clusters.
   - This metric can help identify whether there is a clear separation between clusters and whether there is enough distance between different clusters to maintain their independence.

We integrate both metrics to evaluate the clustering quality. A good clustering should have a small Average Intra-cluster Distance (indicating tightness within clusters) and a large Inter-cluster Distance (indicating high separation between clusters).

Applicable algorithms: DBSCAN

3.2.2 Variance Ratio

The variance ratio[5] evaluates the extent to which each principal component (feature) explains the variance in the data. Each value of the variance ratio represents the proportion of the variance of the corresponding principal component in the total variance. A higher variance ratio indicates that the respective principal component explains more data variance, implying a significant contribution to the information in the data.

Principal component analysis aims to achieve dimensionality reduction by selecting a subset of principal components while retaining as much information from the original data as possible. Therefore, higher variance ratios generally indicate better performance.

Applicable algorithms: PCA

3.2.3 Silhouette Coefficient

The Silhouette coefficient[6] is used to assess clustering performance. After applying the algorithm, the data is used for clustering, and the Silhouette coefficient is calculated to demonstrate its effectiveness. A higher Silhouette coefficient indicates better performance, suggesting higher separation between principal components. The computation is given by:

$$Silhouette \quad Coefficient = \frac{b-a}{max(a,b)}$$

where:
a: the average distance from a data point to other points in the same cluster.
b: the average distance from a data point to the nearest cluster that the point is not a part of.

The Silhouette coefficient ranges from -1 to 1. Values close to 1 indicate that data points within the assigned cluster are more densely packed than those in different clusters, signifying better clustering. Values close to 0 indicate similar density within and between clusters, suggesting overlapping clusters or suboptimal clustering. Values close to -1 indicate that data points within the assigned cluster are more dispersed than those in other clusters, possibly indicating misclassification.

Applicable algorithms: PCA

3.2.4 Mean square error

Mean square error[7] refers to the expected value of the square of the difference between the estimated value of the parameter and the true value of the parameter, recorded as MSE. The calculation formula is as follows:

$$\text{MSE}\left(y, y'\right) = \frac{1}{m}\sum_{i=1}^{m}\left(y_i - y_i'\right)^2$$

Where,
m represents the number of samples.
y_i means the real value.
y' means the predicted value. The range $[0, +\infty)$ is equal to 0 when the predicted value fully matches the actual value, which is a perfect model. The larger the error, the greater the value. In short, the smaller the value, the more accurate the machine learning network model is, and on the contrary, the worse it is.

Applicable algorithms: Linear Regression, MLP

3.2.5 Root mean square error

RMSE is the average of the square root of the error between the predicted value and the true value.

Root mean square error (RMSE) [8], also known as standard error, is the average root of the deviation between the observed value and the true value and the average of the observed numbers. Root mean square error is the deviation between a measured observation and a true value. Standard errors are usually sensitive to specific or specific errors in group measurements, so standard errors can well reflect the accuracy of measurements. The standard error can be used as a criterion for evaluating the accuracy of the measurement process. The calculation formula is as follows:

$$\mathrm{RMSE}\left(y, y'\right)=\sqrt{\frac{1}{m} \sum_{i=1}^{m}\left(y_{i}-y_{i}'\right)^{2}}$$

Where,
m represents the number of samples.
y_i means the real value.
y' means the predicted value.
Its calculation method is to square, average, and then square. The root mean square error is used to measure the deviation between the observed value and the true value. Similarly to MSE, the smaller the difference between our predicted value and the actual value, the higher the accuracy of the model; On the contrary, it is lower.

Applicable algorithms: Linear Regression

3.2.6 Mean absolute error

Mean Absolute Error (MAE) [9] refers to the average absolute deviation of absolute deviation of each measurement value. The average absolute error can avoid the problem of mutual cancellation of errors, and can accurately reflect the size of actual prediction errors. The calculation formula is as follows:

$$\mathrm{MAE}\left(\mathrm{y}, \mathrm{y}'\right)=\frac{1}{\mathrm{m}} \sum_{\mathrm{i}=1}^{\mathrm{m}}\left|\mathrm{y}_{\mathrm{i}}'-\mathrm{y}_{\mathrm{i}}\right|$$

Where,
m represents the number of samples.
y_i means the real value.
y' means the predicted value.
The range $[0,+\infty)$ is similar to MSE and RMSE. The smaller the difference between the predicted value and the actual value, the better the model; on the contrary, the worse the model.

Applicable algorithms: Linear Regression

3.2.7 R-Square

The judgment coefficient[10] is also used to explain the difference score of the regression model. Its useful value range is $[0,1]$, if R2 is less than zero, it indicates that the model's performance is not good, and the closer it is to 1, the better the independent variable can explain the variance change of the dependent variable. The higher the value, the better. The higher the value, the worse the effect. R2 is the best indicator for measuring linear regression. The calculation formula is as follows:

$$R^{2}=1-\frac{\sum_{i=1}^{m}\left(y_{i}'-y_{i}\right)^{2}}{\sum_{i=1}^{m}\left(\bar{y}-y_{i}\right)^{2}}$$

Where,
m represents the number of samples.
y_i means the real value.
y' means the predicted value.
$\bar{y}$ means the average of the real value. $R2 \leq 1$, the larger R2, the better. When your prediction model does not make any errors: $R2=1$ When our model equals the baseline model: $R2=0$ If $R2<0$, the learned model is not as good as the benchmark model.

Applicable algorithms: Linear Regression

3.2.8 Accuracy, Precision, Recall, F1-score and Confusion Matrix

Different from the evaluation function of the predicted value, there is also a set of evaluation functions for the predicted category. This evaluation function needs to start with the confusion matrix [11]:

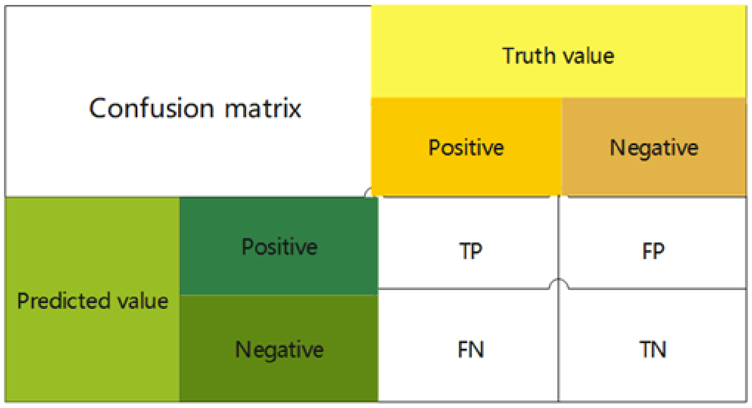

Figure 4: The Confusion Matrix

- TP (True Positive): The predicted result is positive, but it is also positive.
- FP (False Positive): The prediction result is positive, but the actual result is negative.
- FN (False Negative): The predicted result is negative, but it is actually positive.
- TN (True Negative): The predicted result is negative, but it is also negative.

1. Precision: How many of the predicted positive results are positive? The value range is $[0, 1]$. The larger the value, the better the prediction effect. The calculation formula is as follows:

$$P = \frac{TP}{TP + FP}$$

2. Recall: The prediction result is the proportion of positive samples with correct prediction to all positive samples. The value range is $[0, 1]$. The larger the value, the better the prediction effect. The calculation formula is as follows:

$$P = \frac{TP}{TP + FN}$$

3. Accuracy: The proportion of correctly predicted samples to the total sample. The value range is $[0, 1]$. The larger the value, the better the prediction effect. The calculation formula is as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

4. F1-Score: Comprehensively consider precision and recall, The calculation formula is as follows:

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

Applicable algorithms: Naive Bayes, MLP, ID3

3.3 Results

3.3.1 Financial fraud detection

The credit card fraud dataset was processed through the K-Nearest Neighbor (KNN) algorithm and the experiment achieved good results. The KNN algorithm applied to classify the dataset achieved an accuracy of **78.9%**. With precision values of **74.2%** for identifying non-fraudulent transactions and **86.0%** for detecting fraudulent transactions, the algorithm demonstrates good performance in accurately predicting both classes. The recall values of **88.9%** for non-fraudulent transactions and **68.9%** for fraudulent transactions indicate that the algorithm effectively captures a high proportion of both classes. Overall, the KNN algorithm shows promising performance in accurately classifying credit card fraud, with balanced precision and recall values.

3.3.2 Customer relationship management

For Credit Card Customer Churn Prediction in User Relationship Management (see Chapter 4.1.2), we use Naive Bayes and MLP algorithms to predict whether a particular customer is at risk of churn.

1. **Naive Bayes**
   The dataset was processed through the Naive Bayes algorithm, which was chosen because it runs extremely fast when dealing with datasets consisting of a large amount of data, as it does not require iterative training. The classification effectiveness of the algorithm was evaluated by calculating the precision, accuracy, recall and F1 score of the classification. Its data metrics and confusion matrix are shown below. The data on the left and right sides of the positive slash symbol split are categorical evaluation metrics for unchurned and churned users, respectively.

Table 1: Naive Bayes Classification Result

Accurancy	**Precision**	**Recall**	**F1 Score**
0.812	0.855/0.527	0.923/0.354	0.888/0.423

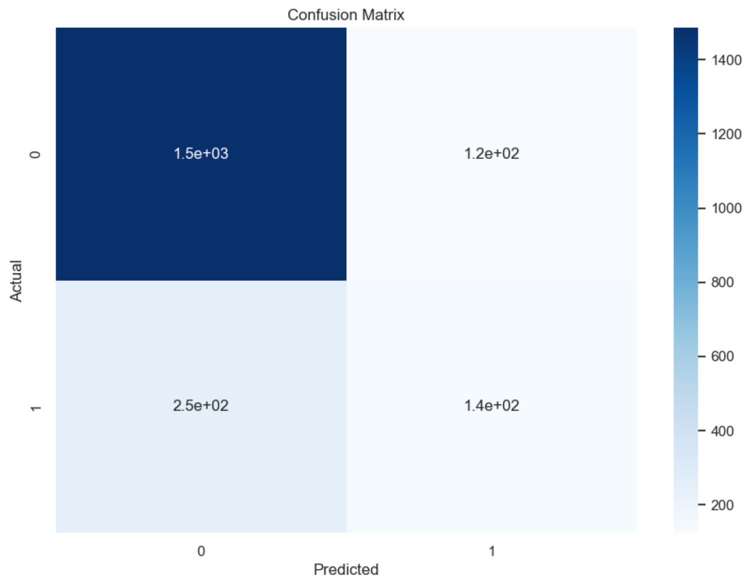

Figure 5: Confusion matrices for classification by Naive Bayes

It can be seen that Naive Bayes maintains a high rate of correct classification. However, it can also be seen that the algorithm has a tendency to predict samples as unchurned users, which may be related to the imbalance of sample categories in the dataset and the inherent limitation of the Naive Bayes algorithm based on the assumption of feature independence.

2. **MLP**
   In processing this dataset using the MLP algorithm, a multilayer perceptron consisting of 200 hidden layer neurons was constructed. After dividing 80% of the data into a training set and training the rest of the data as a test set, the final loss function (mean square error loss function) reached a very small order of magnitude. However, the classification correctness is only 0.805 and the model classifies all samples as unchurn. This could be due to reasons such as overfitting during the training process or an imbalance of sample classes in the dataset itself. In addition, due to the high time complexity of the algorithm, the training time is long for data-heavy datasets, and the time complexity can be reduced by optimizing a large number of iterative operations into matrix operations in subsequent improvements.

3.3.3 Financial forecast

For the Netflix stock data, given history volumes, we aim to predict if a stock's open price is higher or lower than the close price. We formulate this task as a classification task: The label is 0 if the close price is higher than the open price, and is 1 if the open price is higher than the close price. As shown in 6, predictions were 56% accurate using SVM. However, all predictions are labeled 1, might be overfitted or the SVM is not applicable due to high task complexity.

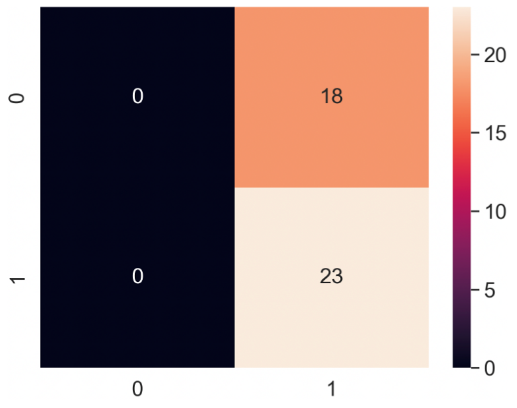

Figure 6: The Visualization Figure of Clustering by Using PCA

3.3.4 Risk management

For the Credit Risk Dataset in the risk management scenario (mentioned in Chapter 2.1), we use PCA and DBSCAN algorithms to understand the customer's credit status and repayment ability.

1. **PCA**
   By applying principal component analysis (PCA), we first normalized the data to ensure that different features have the same scale. We selected the first two principal components to construct a 3D space to visualize the clustering results, with each point representing a data point and the colors indicating different clusters. As can be seen from the figure, there is a clear degree of separation between different clusters.

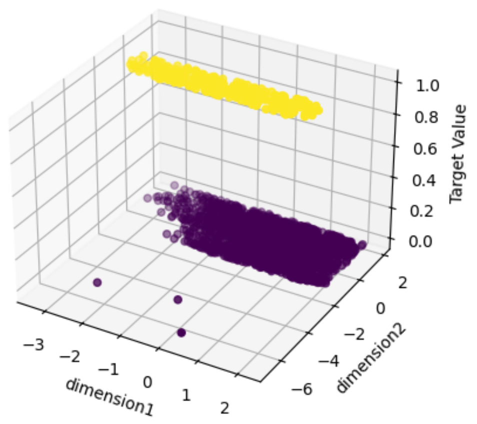

Figure 7: The Visualization Figure of Clustering by Using PCA

At the same time, we use the evaluation indicators of the evaluation module to obtain some key results of PCA:

Silhouette Coefficient: 0.13658947373996197

Variance Ratio for Top 2 Components: [0.4808843 0.33345008]

The Silhouette Coefficient is 0.1366, which means that the data after principal component analysis (PCA) has a certain clustering structure to some extent.

Variance Ratio shows the proportion of variance in the original data retained by the first two principal components. The first principal component retains approximately 48.09% of the variance of the original data, while the second principal component retains approximately 33.35% of the variance. This means that the first two principal components returned by PCA explain about 81.4% of the total variance. By retaining fewer principal components, we can still retain most of the information, thereby reducing the dimensionality of the data and providing a better understanding of the original data. Pretty good dimensionality compression.

2. **DBSCAN**
We also cluster the results after using the DBSCAN algorithm and display the results through a 3D image. In this figure, the points of different colors represent different clusters generated by the DBSCAN algorithm. It can be observed that DBSCAN is better able to capture the density distribution of the data and form clusters of different shapes. This has shown good results for the Credit Risk Management scenario problem.

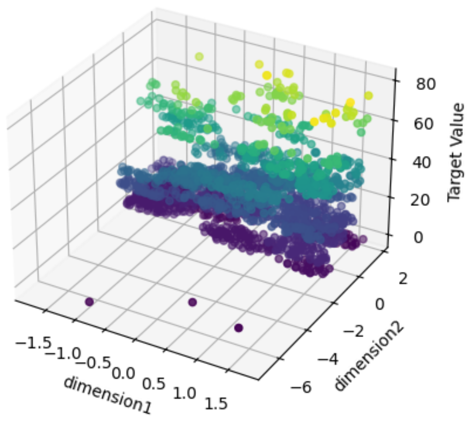

Figure 8: The Visualization Figure of Clustering by Using DBSCAN

At the same time, we use the evaluation indicators of the evaluation module to obtain some key results of the application of DBSCAN in the Credit Risk Dataset:

Average Intra-cluster distance: 0.4546550052052663

Inter-cluster distance: 2.5862008110099484

The average intra-cluster distance is 0.4547, indicating that the samples within each cluster are relatively close, indicating that DBSCAN successfully clusters similar data points together, which is an indicator of a good clustering effect.

The distance between clusters is 2.5862. The distance between clusters is large, which means that there is a large distance between different clusters. They are relatively separated, indicating that different clusters are spatially separated from each other. Ensure that there is enough space between clusters to avoid Over-merging.

The data points within each cluster are close, and the separation between different clusters is high. These results show that DBSCAN performs better clustering effect in the Credit Risk data set.

3.3.5 Investment and Asset Management

By analyzing the results of the linear regression model on the Boston housing price prediction data set, we found that These four evaluation functions indicate that the predicted results are very close to real data, and the model performs well.

Table 2: Linear Regression Classification Result

Mse	**MAE**	**RMSE**	**R2**
54.39	4.97	7.37	0.18

4 Conclusion

In this section, we will summarize the work content of the entire project and discuss the value and significance of the project. In addition, we will provide an outlook on the problem of machine learning algorithm packages in the financial domain we study. Finally, the division of labor among the members of our group is listed.

4.1 Discussion

Machine learning is a data-driven artificial intelligence technology that allows machines to autonomously learn and discover patterns from data and is used for tasks such as classification, clustering, regression, and prediction.[12] Most issues in the financial field are driven by data, and data affects the operations and decisions of many financial practitioners. Therefore, introducing machine learning algorithms that can process massive amounts of data in the financial field will undoubtedly help financial practitioners improve efficiency.

This project is aimed at the problem that the existing machine learning framework has high learning costs for non-professionals and has too many and complex functions. It has designed a set of lightweight machine-learning algorithm packages that can cope with most finance-related tasks. At the same time, it is also aimed at beginners of machine learning. In view of the problem that common machine learning algorithm packages have a high degree of encapsulation and are difficult to learn by reading source code, the designed algorithm package has a low degree of encapsulation and can provide beginners with the reference to implement classic machine learning algorithms.

In total, this project has implemented ten classic machine learning algorithms, which can be used for issues such as risk management, investment and asset management, financial fraud detection, customer relationship management, and financial forecasting in the financial field. The specific algorithms include commonly used regression and classification algorithms, including the naive Bayes algorithm based on statistics, multi-layer perceptron, support vector machine and linear regression that use gradient descent to update parameters, and the unsupervised algorithm K-means clustering algorithm and DBSCAN, integrated learning algorithm Adaboost, feature dimensionality reduction algorithm PCA, as well as decision tree algorithm and KNN algorithm. Then, we tested our algorithms in dozens of datasets, evaluated their performance through accuracy, MSE and other indicators, and used visualization methods such as confusion matrices to display the results of the model. The results show that the algorithm package we designed can better complete various financial machine-learning tasks, proving its effectiveness. This shows that the algorithm package we designed can well assist financial practitioners in some scenarios where machine learning algorithms can help. Finally, we encapsulate our entire algorithm package in a framework, which allows non-professional operators to use the algorithm package through terminal interaction.

There are also some shortcomings to the overall project. First of all, the final accuracy, precision and other indicators of some algorithms on a small number of test data sets are not ideal. This may be related to the imbalance of positive and negative samples in the selected data set. In addition, the training speed of some algorithms is slow, which may be related to the high time complexity of the algorithm during implementation.

4.2 Future Work

As mentioned above, the project still has some shortcomings that can be improved. In addition, there are many directions that can be improved for our goals. We summarize possible future work directions as follows:

- **Optimize the existing algorithm**: reduce the time complexity of the algorithm through matrix operations and other methods, speed up the training time and improve the user experience;
- **Add support for more commonly used machine learning algorithms**: Reinforcement learning algorithms and deep learning algorithms have been widely used in financial product pricing and other fields. Support for

machine learning algorithms such as convolutional neural networks and Q-learning algorithms can be added to suit more financial scenarios;

- **Design a more interactive interface through front-end technology**: The current terminal interaction method is still not friendly enough for non-professional users. Using front-end technology to deploy projects on web pages, the graphical interface can allow more users to get started quickly;
- **Add support for data preprocessing**: Many public data sets have various degrees of irregularities and require data preprocessing in advance and cannot be used directly. This is usually difficult for the non-professional users we target. Therefore, data preprocessing can be encapsulated in an interactive interface to further reduce the user's burden.

4.3 Division of Labour

Table 3: Division of Labour

Team Members	**Specific Tasks**
	Algorithm implementation
Chen Rui	Data preprocessing and Evaluation
	Dataset collection
	Algorithm implementation
Chen Xingnuo	Data preprocessing and Evaluation
	Dataset collection
	Algorithm implementation
Liang Shihao	Data preprocessing and Evaluation
	Dataset collection
	Algorithm implementation
Liu Zhengze	Data preprocessing and Evaluation
	Report writing
	Algorithm implementation
Zhong Ruohan	Framework implementation
	Data preprocessing and Evaluation

References

[1] et al. K.Yogesh. Artificial intelligence (ai): Multidisciplinary perspectives on emerging challenges, opportunities, and agenda for research, practice and policy. *International Journal of Information Management*, 57:101994, 2021.

[2] Antoniya Shivarova. Matthew f. dixon, igor halperin, and paul bilokon: Machine learning in finance from theory to practice. *Financial Markets and Portfolio Management*, 35:555 – 557, 2021.

[3] Cynthia Rudin, Chaofan Chen, Zhi Chen, Haiyang Huang, Lesia Semenova, and Chudi Zhong. Interpretable machine learning: Fundamental principles and 10 grand challenges. *ArXiv*, abs/2103.11251, 2021.

[4] Rahul Malik and Raj Kumar. Validity measure of cluster based on the intra-cluster and inter-cluster distance. 2012.

[5] Elmer Hubert Johnson. Elementary applied statistics: For students in behavioral science. by linton c. freeman. new york: John wiley & sons, 1965. 298 pp. tables and figures. $6.95. *Social Forces*, 44:455–456, 1966.

[6] Peter J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.

[7] *Mean Squared Error*, pages 337–339. Springer New York, New York, NY, 2008.

[8] Rob J Hyndman and Anne B. Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22:679–688, 2006.

[9] Cort J. Willmott and Kenji Matsuura. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate Research*, 30:79–82, 2005.

[10] Brian C. Mustain. Correlation and causation. *Wilmott*, 1937.

[11] Stephen V. Stehman. Selecting and interpreting measures of thematic classification accuracy. *Remote Sensing of Environment*, 62:77–89, 1997.

[12] Yang-Hui He, Kyu-Hwan Lee, and Thomas Oliver. Machine-learning number fields. *arXiv preprint arXiv:2011.08958*, 2020.