**A Hybrid Network Analysis and Machine Learning Model for Enhanced**

**Financial Distress Prediction**

**Abstract:**

Financial distress prediction is crucial to financial planning, particularly amid emerging uncertainties. This study introduces a novel methodology for predicting financial distress by amalgamating network analysis and machine learning techniques. The approach involves establishing two company networks based on their similarity and correlation in crucial financial indicators. The first network reflects similarity across five features, while the second captures correlation in the most critical feature. Subsequently, seven network-centric features are extracted and integrated into the dataset as new variables. Community detection algorithms are also applied to cluster companies, with the resulting labels added as categorical variables. This process yields a modified dataset comprising both initial and network-based variables. Five classification algorithms are employed to forecast financial distress across three scenarios. Initially, models are trained using only the initial features. In subsequent scenarios, network-centric features from similarity and correlation networks are incorporated, enhancing the predictive accuracy of machine learning models. Notably, features from the similarity network play a pivotal role in this improvement. The proposed model showcases superior predictive capabilities and offers a holistic understanding of the dynamic interactions among financial entities. The results underscore the efficacy of network-based strategies in refining financial distress prediction models, providing valuable insights for decision-makers.

**Introduction**

Financial distress prediction is a pivotal realm within corporate finance and economics, presenting an ongoing challenge for investors, businesses, and the broader financial ecosystem. It’s a major challenge and frequently a sign of an imminent crisis. Predicting financial crises signals imminent challenges and holds profound implications for preserving shareholders’ capital. As the pulse of economic stability, predicting financial distress has spurred extensive research, evolving from early multivariate statistical models like the Altman Z-score model [1] to the contemporary frontier of machine learning (ML) algorithms. Altman’s model was based on logistic regression and was widely used and developed by other researchers [2], [3], [4]. Applying machine learning algorithms for forecasting financial distress has gained significant prominence in corporate finance [5]. This trend has witnessed substantial exploration, revealing promising outcomes in predicting financial distress. Numerous studies have emphasized the efficacy of diverse machine-learning techniques across various industries and countries. Notable methods include support vector machines (SVMs), artificial neural networks (ANN), ensemble classifiers, deep learning models, and hybrid machine learning technologies. Beyond numerical factors, integrating textual and management considerations has emerged as a valuable avenue [6]. When equipped with advanced techniques like network analysis and community detection, such models offer a nuanced understanding of the intricate dynamics within financial ecosystems. Through the exploration of these methodologies, researchers aim to enhance the accuracy of predictions and contribute to the development of effective risk management strategies, fostering resilience and stability in the face of economic uncertainties. Analyzing the companies’ networks helps us make better and more accurate predictions of the companies’ financial health. This research introduces a network-based approach for forecasting financial distress using machine learning algorithms. We construct a comprehensive network of businesses by evaluating the similarity and correlation of vital financial variables among companies. Subsequently, a thorough analysis of this network allows us to extract novel information and insights, which are then integrated into our prediction model. Establishing this institutional network enables the extraction of valuable information, contributing to refining our prediction models. Furthermore, this method is particularly beneficial for businesses with restricted available data, as it accurately determines their position within the network, enhancing our ability to predict financial distress.

In the contemporary financial landscape, predicting financial distress is crucial for minimizing risks and ensuring the sustainability of businesses and institutions. Financial distress refers to the situation where an organization is unable to meet its financial obligations, leading to bankruptcy or other severe consequences. Traditional methods of predicting financial distress primarily rely on financial ratios and statistical models, but these methods often fail to capture the complexity and non-linearity of financial data. A **Hybrid Network Analysis and Machine Learning Model** offers a promising solution to this problem by combining the power of traditional financial analysis with advanced machine learning techniques. This hybrid model integrates both quantitative financial indicators and qualitative market sentiments to enhance the accuracy of financial distress predictions. Machine learning algorithms can effectively handle large datasets and complex patterns, while network analysis provides insights into the interrelationships and dependencies between different financial variables, improving predictive capabilities.

### Existing System

The existing systems for predicting financial distress mainly rely on traditional statistical models and financial ratio analysis. These methods, such as **Altman Z-score**, **Logistic Regression**, and **Probit models**, have been widely used for years to predict bankruptcy or financial failure. They use historical financial data, like profitability, liquidity, and leverage ratios, to assess the health of a company. However, these models have certain limitations, including their inability to capture complex relationships between variables, non-linear trends, and dynamic market factors. Furthermore, they primarily focus on quantitative financial data, disregarding external qualitative factors like market sentiment, economic conditions, or social media sentiments, which can significantly affect a company's financial health. Despite improvements in computational techniques, traditional models often lack flexibility, making them less effective in an increasingly data-driven and interconnected financial world.

### Disadvantages of the Existing System

1. **Limited Data Utilization**: Traditional models often rely solely on quantitative financial data (e.g., ratios and balance sheets) without considering qualitative factors such as market sentiment, news, or external events. This restricts the model's ability to capture all relevant signals of potential financial distress.
2. **Inability to Model Complex Relationships**: Traditional statistical methods are linear and fail to capture the complex, non-linear relationships between financial variables. This limitation reduces the ability to accurately model the intricacies of modern financial markets.
3. **Lack of Real-Time Adaptability**: Existing systems do not always adapt to changing market conditions or real-time data inputs. Financial markets are dynamic, and static models often fail to predict distress in real-time or in reaction to sudden shifts in economic or social environments.

### Proposed System

The **Hybrid Network Analysis and Machine Learning Model for Enhanced Financial Distress Prediction** aims to address the shortcomings of existing methods by combining traditional financial analysis with modern machine learning techniques. This system uses a **Hybrid Network** that integrates **financial indicators**, **macroeconomic data**, and **market sentiments** to capture both the direct and indirect relationships influencing a company's financial health. The network analysis helps in identifying dependencies and interactions between various financial entities, such as assets, liabilities, and market conditions, which traditional models cannot account for. Meanwhile, machine learning algorithms such as **Logistic regression**, **Decision tree Classifier** , k nearest Neighbor Classifier and Passive Aggressive Classifier are employed to process large, high-dimensional datasets, identifying patterns and making predictions with higher accuracy.

By combining the strengths of both approaches, the proposed system can handle both structured data (such as balance sheets) and unstructured data (such as news articles and social media posts), improving the model’s robustness and predictive power. The hybrid nature of the system ensures that it adapts more effectively to new data and changing conditions, offering real-time predictions of financial distress.

### Advantages of the Proposed System

1. **Improved Prediction Accuracy**: The combination of network analysis and machine learning allows the system to capture both linear and non-linear relationships, as well as interdependencies between financial variables. This leads to more accurate and reliable predictions compared to traditional methods.
2. **Incorporation of Diverse Data Sources**: Unlike traditional systems that rely solely on quantitative data, the hybrid model can incorporate a wide range of data, including financial ratios, market sentiment, and external economic factors, making it a more comprehensive tool for predicting financial distress.
3. **Real-Time and Dynamic Adaptability**: The proposed system can adapt to changing market conditions by processing real-time data and continuously updating predictions based on the latest available information. This dynamic adaptability helps in identifying potential financial distress before it becomes critical.

**Literature Review**

Financial distress prediction is a critical area of research in finance and economics, aiming to identify early warning signs of financial instability in companies and containing different dimensions. The scientific literature frequently associates the concept of financial distress with bankruptcy, insolvency, the likelihood of default, and patterns of failure. Financial distress is commonly defined as the state of a company facing challenges in meeting its financial obligations [3]. In parallel, it is essential to distinguish between insolvency, bankruptcy, and distressed assets. Insolvency represents a company’s inability to meet its financial obligations, indicating a severe financial crisis. On the other hand, bankruptcy is legal, often following insolvency, where a company undergoes a legal process to resolve its outstanding debts. Distressed assets encompass assets of a company facing financial distress, signalling a potential need for intervention.

In recent years, many researchers have employed various methods to predict various aspects of a company’s future financial status, including bankruptcy, insolvency, and distress. For this purpose, researchers have presented their models and used different indicators for prediction.

The early identification of a firm’s potential failure was facilitated by introducing financial ratios and key indicators. The literature has put forth an extensive array of ratios for this purpose [7].

In this area, the Altman Z-score is a widely used model of financial distress, measuring financial distress inversely [8]. The significance of financial ratio analysis, encompassing profitability ratios, debt ratios, liquidity ratios, and cash flow ratios, has also been underscored in predicting financial distress [9]. Moreover, the examination of corporate social responsibility’s influence on financial distress has been a subject of study, where indicators such as the Altman Z-score and ZM-Score play pivotal roles in assessing the financial health of entities encountering distress [10].

In financial distress prediction, diverse indicators have emerged as a focal point for researchers seeking insights into various facets of a company’s future financial standing, encompassing bankruptcy, insolvency, and distress. Traditional models, notably featuring financial ratios and statistical approaches, have served as enduring pillars in this domain. Noteworthy financial ratios such as return on capital employed, cash flows to total liability, asset turnover ratio, fixed assets to total assets, debt to equity ratio, and firm size, as highlighted by [11] have proven instrumental in predicting financial distress. Additionally, [12] delves into the predictive capabilities of traditional distress prediction models, particularly in identifying early-stage financial distress for firms. Leveraging financial ratios such as leverage, liquidity, profitability, and activity ratios [13]. Moreover, the impact of leverage on corporate financial distress has been studied, revealing a positive effect on financial distress measured by the Z-score [14]. A diversification strategy has also been investigated, with evidence suggesting its potential to reduce the level of financial distress [15]. The literature also emphasizes the importance of ownership structure, with studies indicating its effects on the likelihood of financial distress [8].

In terms of bankruptcy prediction models, various approaches have been explored, including the use of cash flow ratios, logistic regression, and the Altman, Ohlson, Springate, and Zmejewski models [16], [17]. Furthermore, the predictive ability of chosen bankruptcy models has been studied to assess credit risk and predict the financial situation to indicate the probable bankruptcy of a company [18]. The bias of unhealthy small and medium-sized enterprises (SMEs) in bankruptcy prediction models has also been addressed, emphasizing the importance of using a financial health indicator to construct the estimation sample for accurate predictions [19].

In conclusion, exploring various variables and indicators in the literature has provided valuable insights into the multifaceted landscape of financial distress prediction. Researchers have extensively examined factors influencing predictive models, from traditional financial ratios to diverse indicators encompassing ownership structure and corporate social responsibility. This foundational understanding sets the stage for the subsequent discussion on diverse models employed in the field, offering a comprehensive overview of the evolving strategies for financial distress prediction.

In addition to the indices used, the models employed in this field also play a significant role in predicting financial distress. Machine learning approaches have gained traction in financial distress prediction in recent years. The application of machine learning has further promoted studies on financial distress prediction and improved financial distress prediction accuracy [20]. In recent years, machine learning algorithms have become extensively utilized in predicting company financial difficulties [21]. Previous studies have applied the support vector machine prediction model in financial distress [22]. In addition, the rapid advancement of computers and software has given rise to other techniques such as data mining, machine learning, deep learning, and artificial intelligence [23]. Furthermore, machine learning models have become a trend in quantitative finance, leading to a surge in interest in big data applications in the financial markets [24]. Reference [25] emphasized the focus on statistical and machine learning models for predicting financial distress. Machine learning methods, including artificial neural networks, support vector machines, and random forests, have demonstrated their effectiveness in forecasting financial trouble [26] machine learning methods, including random forest, support vector machines, and neural networks, effectively predicted financial distress and established early warning mechanisms for companies. Furthermore, [23] highlighted the emergence of machine learning techniques, including data mining, deep learning, and artificial intelligence, in financial distress prediction.

Moreover, comparing traditional and machine learning models has been a subject of interest. Several studies have highlighted the superiority of machine learning-based models over conventional methods in predicting corporate financial distress [11]. These machine-learning techniques include support vector machines, deep learning models, hybrid machine-learning technologies, genetic algorithms, and neural network models [20], [21], [22], [27], [28]. Reference [29] compared traditional methods such as logistic regression with machine learning models like random forest and neural networks to identify the model with the highest predictive accuracy of financial distress. This comparison sheds light on the effectiveness of machine learning approaches in financial distress prediction. Furthermore, [21] proposed combining financial management theory with machine learning algorithms to develop effective methods for predicting financial distress. Additionally, [24] and [30] we have enhanced the financial distress warnings evaluation system by incorporating ecological efficiency, indicating the potential for integrating diverse factors into distress prediction models.

The application of these models has been observed in various sectors, such as banking, real estate, and manufacturing, indicating the versatility of machine learning in financial distress prediction [16], [31], [32].

Various types of financial data have been commonly employed in machine learning models to predict financial distress. These models have been widely used due to their capability to model complicated features of financial data [33]. The types of financial data commonly employed in machine learning models for predicting financial distress include financial ratios, discriminant analysis, and linear discriminant analysis [1]. Additionally, deep learning-based models have been designed to predict a company’s financial distress [25]. Furthermore, the authors of a study used a decision tree and classified regression tree of machine learning models to predict the financial fraud of listed companies in the United States [34]. Moreover, machine learning models have been used to predict the behavior of various aspects of financial markets given some input features [35]. These models have also been applied in developing the financial time series forecasting task and found to outperform other statistical models [36].

In summary, exploring machine learning models in financial distress prediction reveals a dynamic landscape of innovation and adaptability. As we transition to the discussion on the financial data utilized with these models, it becomes evident that the intricate interplay between advanced algorithms and diverse datasets, coupled with the critical aspect of feature selection, is crucial for achieving accurate predictions. Feature selection, drawing considerable attention from researchers, is essential in enhancing the predictive capabilities of machine learning models and contributes to the robustness of financial distress prediction strategies.

Feature selection involves identifying the most relevant variables, while feature engineering involves creating new features from the existing ones to enhance the model’s predictive capability. Accurate financial distress prediction relies on the careful selection of suitable features. Machine learning methods, including support vector machines, artificial neural networks, and deep learning models, have been extensively employed to predict financial crises. These models require careful feature selection and engineering to achieve optimal performance. Emphasized the widespread use of machine learning algorithms in corporate financial distress prediction [32].

For instance, [37] specific financial ratios, such as return on capital employed, cash flows to total liability, asset turnover ratio, fixed assets to total assets, debt to equity ratio, and firm size, significantly contribute to distress prediction. In addition, [9] emphasized using diverse business analytics techniques, including statistical and artificial intelligence methodologies, to enhance the precision of financial crisis prediction models.

Hybrid machine-learning techniques have also been explored for financial distress prediction, aiming to establish effective prediction models focused on developing an effective financial distress prediction model using hybrid machine-learning techniques [22]. In addition, [22] introduced a novel framework for a financial early warning system that enhances its effectiveness by integrating the unconstrained distributed lag model with commonly employed financial distress prediction models, such as the logistic model and SVM.

Furthermore, the role of macroeconomic determinants in corporate financial distress has been investigated, highlighting the importance of systematic variable selection approaches to develop alternative models of financial distress [23].

Moreover, the incorporation of machine learning technologies, such as big data analysis, network analysis, and sentiment analysis, has been recognized as a favorable method for conducting systemic risk analysis in the financial industry [38] This suggests that machine learning aids in predicting individual company distress and contributes to assessing and measuring systemic risk in financial markets. Additionally, the use of specific financial ratios and indicators, such as return on equity (ROE), debt-to-equity ratio (DER), and current ratio (CR), in machine learning models has been found to significantly impact the prediction of financial distress conditions [39]. This emphasizes the importance of feature selection and the incorporation of relevant financial variables in developing accurate distress prediction models. Moreover, the literature indicates that machine learning models have demonstrated high prediction accuracy, ranging from 78% to 93%, in forecasting financial distress, surpassing the performance of traditional discriminant models [27]. This highlights the potential of machine learning models in providing robust and reliable predictions for financial distress, thereby assisting stakeholders in making informed decisions.

In recent years, the convergence of finance, network analysis, and machine learning has spurred the development of inventive methodologies for predicting and comprehending financial distress. Extensive applications of network analysis have emerged as a powerful tool to unravel the interconnectedness of financial challenges within diverse organizations or industries. This approach involves deploying statistical connection indicators, including correlation, granger causality, and tail dependency. These metrics delineate and scrutinize the intricate network of transmission effects among financial institutions, providing valuable insights into the dynamics of financial distress [40]. Market-based measures of interconnectedness have been used to investigate the relationship between financial stability and interconnectedness, with extensive literature reviews discussing various financial network models [41]. Additionally, qualitative approaches through literature reviews have been used to systematically analyze the network formed by financial distress literature in specific sectors [42]. Furthermore, network analysis has been utilized to predict bank distress and to provide support for measures of interconnectedness in early-warning models, aiming to capture the interconnectedness among financial entities that could trigger the formation of contagion channels [43], [44]. Assessing interconnectedness in financial institutions has been identified as an early warning indicator for distress in financial [45]. Several methodologies have been suggested in scholarly works to gauge the level of interconnection across financial institutions and systems [46].

Moreover, network models have played a crucial role in managing systemic risk by capturing the interconnectedness among financial entities, which could lead to amplifying shocks to the financial system [44]. The interconnectedness measures in financial networks are based on the topology of links between banks, insurers, and financial services companies [47]. Additionally, network analysis has been used to model financial distress propagation on customer-supplier networks, allowing the investigation of possible scenarios for the functioning of financial distress propagation and the assessment of economic health within the network [48].

In the context of specific sectors, such as the textile and garment industry, network analysis has been employed to predict financial distress and its impact, aiming to determine the effect of profitability, liquidity, and solvency on financial distress in these sectors [42], [49]. Furthermore, network analysis has been applied to predict financial distress in various sectors, including infrastructure, utilities, transportation, and retail trade, using financial ratios and macroeconomic variables as independent variables [45], [50], [51], [52].

Applying network analysis, researchers have been able to discern systemically essential banks and possible contagion paths, shedding light on the interconnectedness of financial institutions and the potential for contagion [53]. Furthermore, the study of contagion and systemic risk in financial networks has evolved from seminal works to a review of subsequent literature, indicating the continuous development and relevance of network analysis in understanding financial distress [54].

In tandem with the foundational studies discussed in the financial distress prediction, recent contributions have significantly shaped the landscape of credit prediction and financial network analysis. In this area [55] It introduced an innovative approach that explicitly considers feature interactions, offering a nuanced understanding of complex relationships within financial datasets. By incorporating contrastive learning techniques, the study sheds light on the subtle dependencies among features, contributing to more accurate credit predictions. This research significantly contributes to the evolving landscape of credit prediction methodologies by acknowledging and leveraging the power of feature interactions in financial modelling.

In the domain of shareholding networks, a notable exploration has been undertaken by [56]. This research employed a sophisticated Voting Game Approach to uncover and analyze control associations within shareholding networks. By employing game-theoretic principles, the study provided insights into the dynamics of control exerted by shareholders in corporate structures.

The use of complex network theory to model the structure of the financial system and analyze risk contagion, particularly in banking systems, has been widely adopted by [57], [58]. This approach has allowed for the systematic analysis of how network structure and bank characteristics affect solvency distress contagion risk in interbank networks [59]. As the economic and financial system becomes more complex, the use of complex networks to study systemic risk and risk contagion has become increasingly important [60].

The examination and prediction of situations in which the stability of actual financial networks is susceptible to contagion and the conditions under which it becomes widespread have been emphasized as crucial [48]. Additionally, the use of multilayer network analysis has been employed to model risk contagion in venture capital markets, providing insights into how risk can spread through connections between market participants and harm total market robustness [61].

Studies have also focused on the spatial network contagion of environmental risks among countries, demonstrating the applicability of network analysis beyond traditional financial contexts [62]. Moreover, the warning of financial distress and bankruptcy has been investigated using financial data reported in financial statements, indicating the diverse applications of network analysis in understanding financial distress [30].

Statistical methods and machine learning have been extensively employed in prior research endeavors centered on bankruptcy and financial distress prediction. However, a notable gap exists, as no study has concentrated on creating a network structure among companies based on similarities in their financial data. It subsequently utilized this network analysis for predictive purposes regarding their future statuses. Establishing a network among companies based on limited financial indicators enables the examination of intricate relationships between them. Leveraging both network analysis and machine learning techniques, a heightened capacity emerges to forecast the probability of financial distress. This integrative methodology not only refines the accuracy of forecasting companies’ future statuses but also allows for predicting potential bankruptcies, even when comprehensive financial information is limited. It facilitates precise predictions regarding a company’s risk of financial distress and bankruptcy without necessitating an abundance of detailed financial data.

Therefore, the present study aims to delve more deeply into this phenomenon. The subsequent section will expound on the intricacies of forming networks among companies. Following this, we will undertake a detailed analysis of these networks. By harnessing the insights derived from this network analysis, our research will then progress to predicting the likelihood of financial distress for individual companies.

**Hardware Requirements :**

➢ Processor - Pentium –IV

➢ RAM - 4 GB (min)

➢ Hard Disk - 20 GB

➢ Key Board - Standard Windows Keyboard

➢ Mouse - Two or Three Button Mouse

➢ Monitor - SVGA

**Software Requirements :**

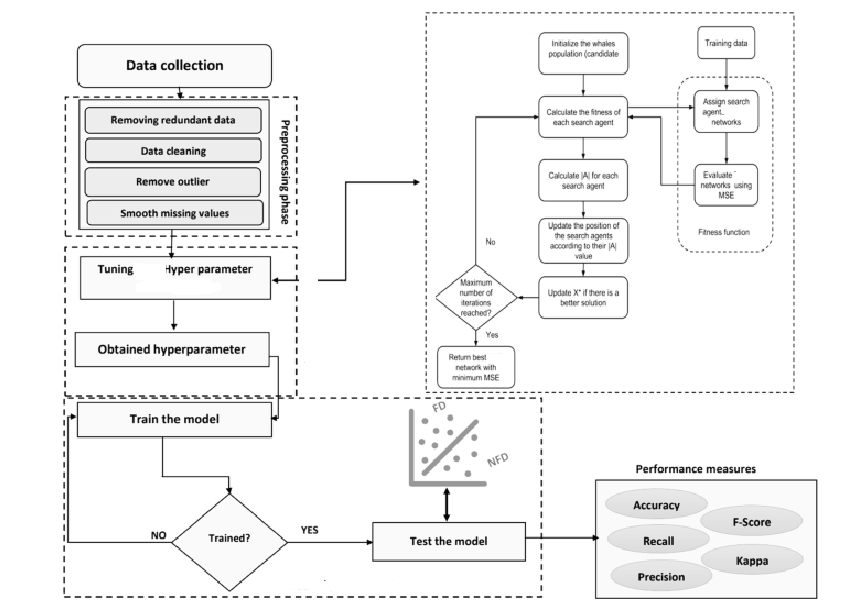
Operating system : Windows 7 Ultimate.

Coding Language : Python.

Front-End : Python.

Back-End : HTML, Css

**System Architecture**



**Modules Description:**

**A. Data Preprocessing and Feature Selection**

In the initial stage, meticulous handling of the dataset is essential. The data preprocessing phase involves a comprehensive analysis to enhance problem understanding. This includes addressing missing and duplicate data, evaluating target variable balance, and refining the dataset using machine learning techniques and feature correlation analyses. Identifying highly significant features is a pivotal outcome, forming the foundation for subsequent steps. We calculate feature correlation with the target variable to rank features based on their efficacy in prediction. After distilling the most influential features, the next step involves constructing two networks. One is based on the similarity of companies in these selected features, and the other is based on the correlation of companies in the most crucial feature. This strategic move aims to capture and emphasize the interrelationships between companies based on their commonalities, setting the stage for a more nuanced understanding of the underlying dynamics. Utilizing the top 5 features streamlines the analysis and ensures a focused exploration of critical indicators that contribute significantly to the model’s predictive capacity.

**B. Network Construction**

Understanding the financial market as an intricate network enables us to scrutinize publicly listed companies’ structure and developmental patterns.

Two different methods have been employed to construct a network among companies. In the first approach, the network is formed by assessing the similarity between companies. This involves gauging the similarity among companies in specific indicators mentioned in the preceding subsection, linking companies with comparable characteristics.

The second method employs the correlation between companies in specific indicators for network construction. This approach generates a correlation matrix among companies, and the distances between network points are subsequently determined based on this matrix. This method has previously been utilized to construct a financial network [64]. In the following subsection, we will elaborate on the methods employed for network formation. These strategies aim to comprehensively understand the relationships between companies, offering insights into their structural and correlational dynamics.

**1) Construct the Network Based on Similarity**

To construct a network based on the similarity, we utilized the distances between companies in 5 feature values described in the previous section. We first normalized the dataset to avoid the undue influence of any single feature, considering their distinct ranges. This normalization ensures that each of the selected five features contributes equally to network formation, fostering a balanced and unbiased representation.

We obtained five matrices by calculating the distances between companies for each of the chosen features, each presenting the inter-company distances within a specific feature. Subsequently, a consolidated matrix emerges through the computation of the average distances within these matrices. This matrix (D ), representing collective distances, is the foundation for network formation. In this matrix, di,j is average distance between company i and j. In the subsequent stages, this matrix will be instrumental in delineating the relationships and connectivity patterns between companies based on their similarities in the selected features. This method ensures a fair consideration of all chosen features and fosters a holistic understanding of the nuanced interplay between companies in the network.

Then, we use the K-Nearest Neighbors (KNN) algorithm to construct the network. Using the distance matrix (D ), each node will be connected to its K nearest neighbors. To obtain the best number of neighbors, we will try the algorithm with different K numbers in our dataset and choose the best number. This method to construct the network was earlier used to create a network between time series [33].

**2) Construct the Network Based on Correlation**

In the second approach, we constructed financial market networks using a methodology grounded in correlation matrices. This involved calculating the correlation coefficient between pairs of companies based on a chosen financial indicator.

Upon computing the correlation coefficients, a matrix of dimensions N×N is obtained, where N represents the number of companies in consideration. Subsequently, the distance between any two companies can be determined using . This method allows for a quantitative representation of company relationships, emphasizing their interconnectivity in the financial market.

di,j=2(1−ρi,j)−−−−−−−−√(1)

this Equation ρi,j is the correlation coefficient between companies i and j.

The symmetric N×N distance matrix (D ) illustrates the distances among N firms. Utilizing this distance matrix, a weighted matrix (W ) can be formed to depict the intricate topology of the network. company i and j are linked within the network, and the weight of the link (wi,j ) can be calculated by , signifying the strength of the connection between the two companies.

wi,j=exp(−di,j)(2)

**C. Network Analysis**

Following the formation of two networks using the methods discussed in the previous subsection, the next step involves analyzing each of these networks. This analysis encompasses extracting seven key network features and clustering companies within the network through community detection.

In the following subsection, we explain these steps.

**1) Feature Extraction**

This research employed network-based variables as predictors for forecasting financial distress. The network-based features extracted from these two networks are as follows.

* Degree Centrality
* Betweenness Centrality
* Closeness Centrality
* Clustering Coefficient
* Page Rank Centrality
* Average Neighbor Degree
* Clustering Coefficient Weighted

To better understand these features, we briefly define them in this section.

a: Degree Centrality

Degree centrality, a widely utilized centrality metric, quantifies the number of direct connections associated with a specific node. It can be understood as representing the immediate risk of a node in capturing entities flowing through the network, like information. In examinations of weighted networks, the concept of centrality degree is commonly broadened to encompass the sum of weights [65].

b: Betweenness Centrality

Interactions between two non-directly connected nodes depend on other nodes within the set, especially those along the paths connecting the two nonadjacent nodes. These intermediary nodes, often called ‘in-between nodes,’ may influence interactions between the two nonadjacent nodes. The concept of betweenness is rooted in the idea of network paths. Reference [66] defines a network path as a sequence of nodes traversed by following connections from one to another throughout the network. A geodesic path represents the shortest route through the network from one node to another. The betweenness of a node is computed as the fraction of the shortest paths between pairs of nodes that traverse through this specific node [65].

c: Closeness Centrality

It measures a node’s closeness to all other nodes in a network, considering the shortest path distance. Nodes with elevated proximity centrality demonstrate the capability to connect with other nodes in the network quickly [65].

d: Clustering Coefficient

The metric known as the clustering coefficient of a node within a network quantifies the likelihood that its neighboring nodes are connected. This measurement serves as an indicator of the tendency of nodes to form clusters or communities [67].

e: Page Rank Centrality

Page rank serves as an algorithm utilized by Google to assess the ranking of web pages within search results. In the context of network analysis, Page rank centrality quantifies the importance of a node by considering the importance of nodes that have links directed towards it [68].

f: Average Neighbor Degree

This evaluates the mean degree of a node’s neighbors within the network, offering insights into the local structure surrounding that particular node [68].

g: Clustering Coefficient Weighted

This represents an expansion of the clustering coefficient, factoring in the strength or intensity of connections among nodes. It gauges the inclination of nodes to create clusters, incorporating the weights of their connections [61].

After computing these 7 features for each network, we added them as new features to the original dataset. As a result of this process, we have a new dataset including financial indicators and network-centric features as predictor variables.

**2) Community Detection**

During the community detection phase, we employ a label propagation algorithm to categorize companies based on common characteristics or patterns within the network. These identified clusters then act as supplementary labels or categories for the diverse attributes present in the dataset. The assignment of these categories can yield valuable insights, potentially enhancing the predictive accuracy of the models. In this step we used a community detection algorithm for two networks (networks based on similarity and correlation). The number of clusters and nodes in each cluster are different in each network. In the next section, we present each network cluster characteristic.

This strategic categorization contributes to a more nuanced understanding of the dataset’s structure and fosters the potential for refining and optimizing predictive models based on shared characteristics among companies. After this step, we add cluster number as a definite feature to the dataset.

**D. Machine Learning Models**

In this step, we have a new dataset including financial indicators and new features obtained from the network.

Now, we can use different classification algorithms, such as Support Vector Machines and Decision Trees, to predict financial distress. In this phase, to compare different approaches to network construction, we use features from the similarity-based network and the correlation-based network separately and analyze the obtained results. The following subsection introduces machine learning models and evaluation matrices used in this phase.

**1) Classification Models**

Utilizing classification algorithms allows us to harness the existing data for both model training and prediction. In this section of the study, we have incorporated five widely recognized and extensively used machine learning models. The subsequent paragraphs provide a succinct overview of each model.

a: Logistic Regression

Logistic regression is a foundational model for binary classification tasks. It assesses the relationship between the dependent binary variable and the independent variables, making it well-suited for predicting financial distress where the outcome is binary either distressed or not [69].

b: K-Nearest NEIGHBORS (KNN)

KNN, a non-parametric algorithm, classifies or predicts an instance based on its proximity to the K nearest neighbors within the feature space. The algorithm’s output is influenced by the most predominant class among its K nearest neighbors in the context of classification, or it computes the average of their values in the case of regression [69].

c: Support Vector Machine (SVM)

Support vector machines are potent models for classification and regression tasks. SVMs excel in delineating decision boundaries in high-dimensional spaces, making them advantageous in scenarios where the data may not be linearly separable [70].

d: Decision Tree

Decision trees are versatile models adept at handling complex relationships within data. These hierarchical structures recursively split the dataset based on features, providing interpretable results and identifying significant predictors for financial distress [70].

**2) Evaluation**

Diverse metrics such as accuracy is utilized to assess the models and compare the efficiency of different approaches. The performance of the implemented models is computed for each scenario. This section involves evaluating and comparing the results derived from the model. A detailed explanation of the outcomes obtained from the model on the subsequent dataset will be provided. We will introduce the metrics mentioned above to enhance comprehension of the model evaluation stage.

a: Accuracy

This statistic measures the overall efficiency of our model, calculated as the proportion of accurately predicted instances (true positives and true negatives) to the total number of data points using  In this context, TP denotes true positives, and TN denotes true negatives. Essentially, accuracy represents the proportion of accurate predictions [71].

Accuracy=TP+TNTotalnumberofdata(3)

**FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

* ECONOMICAL FEASIBILITY
* TECHNICAL FEASIBILITY
* SOCIAL FEASIBILITY

**ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

**TECHNICAL FEASIBILITY**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

**SYSTEM DESIGN**

**UML DIAGRAMS:**

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

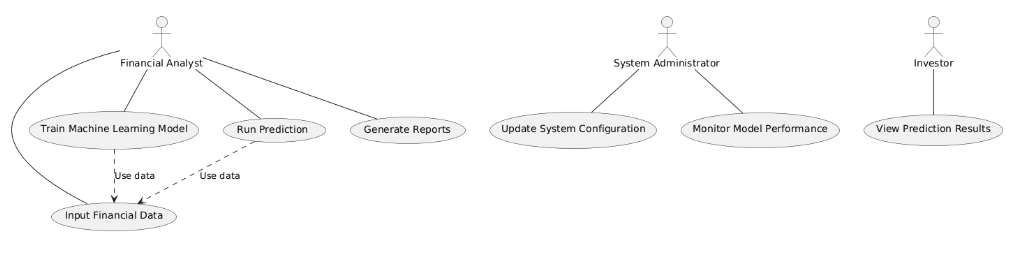
**GOALS:**

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Integrate best practices.

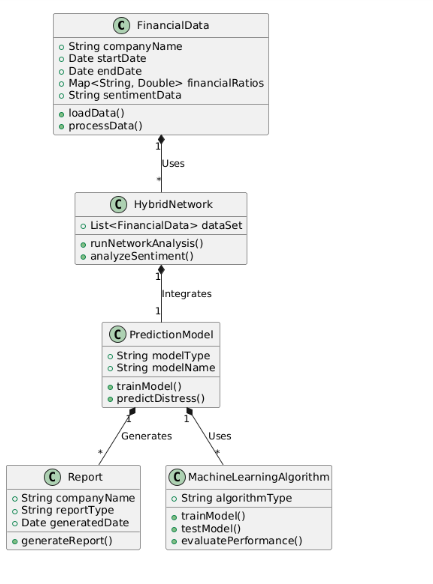
**USE CASE DIAGRAM:**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.



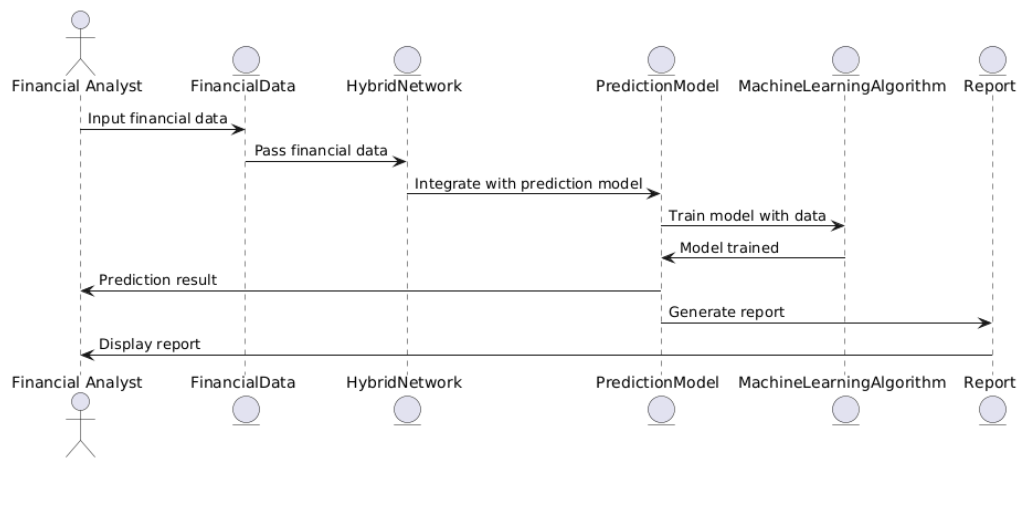
**CLASS DIAGRAM:**

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



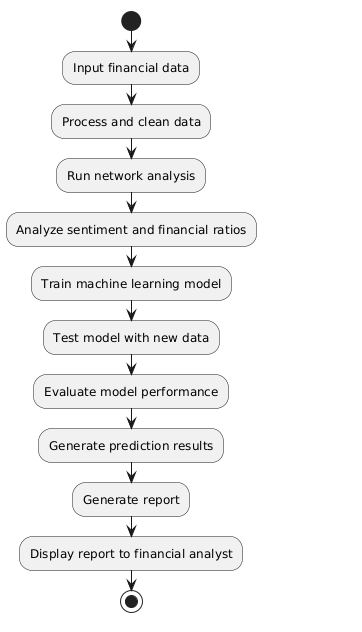
**SEQUENCE DIAGRAM:**

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



**ACTIVITY DIAGRAM:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



**SOFTWARE ENVIRONMENT**

**Python** is a high-level, interpreted scripting language developed in the late 1980s by Guido van Rossum at the National Research Institute for Mathematics and Computer Science in the Netherlands. The initial version was published at the alt. Sources [newsgroup](https://en.wikipedia.org/wiki/Usenet) in 1991, and version 1.0 was released in 1994.

Python 2.0 was released in 2000, and the 2.x versions were the prevalent releases until December 2008. At that time, the development team made the decision to release version 3.0, which contained a few relatively small but significant changes that were not backward compatible with the 2.x versions. Python 2 and 3 are very similar, and some features of Python 3 have been back ported to Python 2. But in general, they remain not quite compatible.

Both Python 2 and 3 have continued to be maintained and developed, with periodic release updates for both. As of this writing, the most recent versions available are 2.7.15 and 3.6.5. However, an official [End of Life date of January 1, 2020](https://pythonclock.org/) has been established for Python 2, after which time it will no longer be maintained. If you are a newcomer to Python, it is recommended that you focus on Python 3, as this tutorial will do.

Python is still maintained by a core development team at the Institute, and Guido is still in charge, having been given the title of BDFL (Benevolent Dictator For Life) by the Python community. The name Python, by the way, derives not from the snake, but from the British comedy troupe [Monty Python’s Flying Circus](https://en.wikipedia.org/wiki/Monty_Python%27s_Flying_Circus), of which Guido was, and presumably still is, a fan. It is common to find references to Monty Python sketches and movies scattered throughout the Python documentation.

**WHY CHOOSE PYTHON**

If you’re going to write programs, there are literally dozens of commonly used languages to choose from. Why choose Python? Here are some of the features that make Python an appealing choice.

**Python is Popular**

Python has been growing in popularity over the last few years. The 2018 [Stack Overflow Developer Survey](https://insights.stackoverflow.com/survey/2018) ranked Python as the 7th most popular and the number one most wanted technology of the year. [World-class software development countries around the globe use Python every single day.](https://realpython.com/world-class-companies-using-python/)

According to [research by Dice](https://insights.dice.com/2016/02/01/whats-hot-and-not-in-tech-skills/) Python is also one of the hottest skills to have and the most popular programming language in the world based on the [Popularity of Programming Language Index](https://pypl.github.io/PYPL.html).

Due to the popularity and widespread use of Python as a programming language, Python developers are sought after and paid well. If you’d like to dig deeper into [Python salary statistics and job opportunities, you can do so here](https://dbader.org/blog/why-learn-python).

**Python is interpreted**

Many languages are compiled, meaning the source code you create needs to be translated into machine code, the language of your computer’s processor, before it can be run. Programs written in an interpreted language are passed straight to an interpreter that runs them directly.

This makes for a quicker development cycle because you just type in your code and run it, without the intermediate compilation step.

One potential downside to interpreted languages is execution speed. Programs that are compiled into the native language of the computer processor tend to run more quickly than interpreted programs. For some applications that are particularly computationally intensive, like graphics processing or intense number crunching, this can be limiting.

In practice, however, for most programs, the difference in execution speed is measured in milliseconds, or seconds at most, and not appreciably noticeable to a human user. The expediency of coding in an interpreted language is typically worth it for most applications.

### Python is Free

The Python interpreter is developed under an OSI-approved open-source license, making it free to install, use, and distribute, even for commercial purposes.

A version of the interpreter is available for virtually any platform there is, including all flavors of Unix, Windows, macOS, smart phones and tablets, and probably anything else you ever heard of. A version even exists for the half dozen people remaining who use OS/2.

### Python is Portable

Because Python code is interpreted and not compiled into native machine instructions, code written for one platform will work on any other platform that has the Python interpreter installed. (This is true of any interpreted language, not just Python.)

### Python is Simple

As programming languages go, Python is relatively uncluttered, and the developers have deliberately kept it that way.

A rough estimate of the complexity of a language can be gleaned from the number of keywords or reserved words in the language. These are words that are reserved for special meaning by the compiler or interpreter because they designate specific built-in functionality of the language.

Python 3 has 33 keywords, and Python 2 has 31. By contrast, C++ has 62, Java has 53, and Visual Basic has more than 120, though these latter examples probably vary somewhat by implementation or dialect.

Python code has a simple and clean structure that is easy to learn and easy to read. In fact, as you will see, the language definition enforces code structure that is easy to read.

But It’s Not That Simple For all its syntactical simplicity, Python supports most constructs that would be expected in a very high-level language, including complex dynamic data types, structured and functional programming, and [object-oriented programming](https://realpython.com/python3-object-oriented-programming/).

Additionally, a very extensive library of classes and functions is available that provides capability well beyond what is built into the language, such as database manipulation or GUI programming.

Python accomplishes what many programming languages don’t: the language itself is simply designed, but it is very versatile in terms of what you can accomplish with it.

## Conclusion

This section gave an overview of the **Python** programming language, including:

* A brief history of the development of Python
* Some reasons why you might select Python as your language of choice

Python is a great option, whether you are a beginning programmer looking to learn the basics, an experienced programmer designing a large application, or anywhere in between. The basics of Python are easily grasped, and yet its capabilities are vast. Proceed to the next section to learn how to acquire and install Python on your computer.

**Python** is an [open source](https://simple.wikipedia.org/wiki/Open_source) [programming language](https://simple.wikipedia.org/wiki/Programming_language) that was made to be easy-to-read and powerful. A [Dutch](https://simple.wikipedia.org/wiki/Netherlands) programmer named [Guido van Rossum](https://simple.wikipedia.org/wiki/Guido_van_Rossum) made Python in 1991. He named it after the television show [Monty Python's Flying Circus](https://simple.wikipedia.org/wiki/Monty_Python%27s_Flying_Circus). Many Python examples and tutorials include jokes from the show.

Python is an interpreted language. Interpreted languages do not need to be [compiled](https://simple.wikipedia.org/wiki/Compiled_language) to run. A program called an [interpreter](https://simple.wikipedia.org/wiki/Interpreter_(computing)) runs Python code on almost any kind of computer. This means that a programmer can change the code and quickly see the results. This also means Python is slower than a compiled language like [C](https://simple.wikipedia.org/wiki/C_(programming_language)), because it is not running [machine code](https://simple.wikipedia.org/wiki/Machine_code) directly.

Python is a good programming language for beginners. It is a high-level language, which means a programmer can focus on what to do instead of how to do it. Writing programs in Python takes less time than in some other languages.

Python drew inspiration from other programming languages like C, [C++](https://simple.wikipedia.org/wiki/C%2B%2B), [Java](https://simple.wikipedia.org/wiki/Java_(programming_language)), [Perl](https://simple.wikipedia.org/wiki/Perl), and [Lisp](https://simple.wikipedia.org/wiki/LISP).

Python has a very easy-to-read syntax. Some of Python's syntax comes from C, because that is the language that Python was written in. But Python uses whitespace to delimit code: spaces or tabs are used to organize code into groups. This is different from C. In C, there is a [semicolon](https://simple.wikipedia.org/wiki/Semicolon) at the end of each line and curly braces ({}) are used to group code. Using whitespace to delimit code makes Python a very easy-to-read language.

**Python use [change / change source]**

Python is used by hundreds of thousands of programmers and is used in many

places. Sometimes only Python code is used for a program, but most of the time it is used to do simple jobs while another programming language is used to do more complicated tasks.

Its [standard library](https://simple.wikipedia.org/w/index.php?title=Standard_library&action=edit&redlink=1) is made up of many [functions](https://simple.wikipedia.org/wiki/Computable_function) that come with Python when it is installed. On the [Internet](https://simple.wikipedia.org/wiki/Internet) there are many other [libraries](https://simple.wikipedia.org/w/index.php?title=Library_(computing)&action=edit&redlink=1) available that make it possible for the Python language to do more things. These libraries make it a powerful language; it can do many different things.

Some things that Python is often used for are:

* Web development
* Scientific programming
* Desktop [GUIs](https://simple.wikipedia.org/wiki/GUI)
* Network programming
* [Game](https://simple.wikipedia.org/wiki/Video_game) programming

**Domain Specification**

**MACHINE LEARNING**

Machine Learning is a system that can learn from example through self-improvement and without being explicitly coded by programmer. The breakthrough comes with the idea that a machine can singularly learn from the data (i.e., example) to produce accurate results.

Machine learning combines data with statistical tools to predict an output. This output is then used by corporate to makes actionable insights. Machine learning is closely related to data mining and Bayesian predictive modeling. The machine receives data as input, use an algorithm to formulate answers.

A typical machine learning tasks are to provide a recommendation. For those who have a Netflix account, all recommendations of movies or series are based on the user's historical data. Tech companies are using unsupervised learning to improve the user experience with personalizing recommendation.

Machine learning is also used for a variety of task like fraud detection, predictive maintenance, portfolio optimization, automatize task and so on.

**Machine Learning vs. Traditional Programming**

Traditional programming differs significantly from machine learning. In traditional programming, a programmer code all the rules in consultation with an expert in the industry for which software is being developed. Each rule is based on a logical foundation; the machine will execute an output following the logical statement. When the system grows complex, more rules need to be written. It can quickly become unsustainable to maintain.

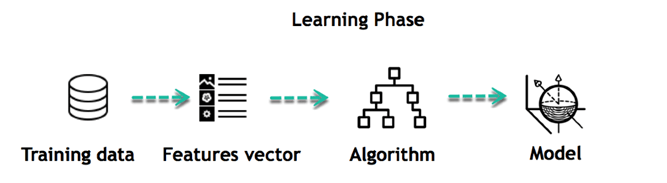
**Machine Learning**https://lh7-us.googleusercontent.com/ImRrh_rNf9QFETqztAjZ5tjlE7s2nia_UWLUWv2rFZUsppK08LO3Lgona7ISwBnjfrjw3tBumOxQyKCr8-3JLY7EkMzoohVjmz-qD1xqKYr-6_562ISDoAbzShkY6Kft-aYsdCFWlnEfmV91C0vBlw

How does Machine learning work?

Machine learning is the brain where all the learning takes place. The way the machine learns is similar to the human being. Humans learn from experience. The more we know, the more easily we can predict. By analogy, when we face an unknown situation, the likelihood of success is lower than the known situation. Machines are trained the same. To make an accurate prediction, the machine sees an example. When we give the machine a similar example, it can figure out the outcome. However, like a human, if its feed a previously unseen example, the machine has difficulties to predict.

The core objective of machine learning is the **learning** and**inference**. First of all, the machine learns through the discovery of patterns. This discovery is made thanks to the **data**. One crucial part of the data scientist is to choose carefully which data to provide to the machine. The list of attributes used to solve a problem is called a **feature vector.** You can think of a feature vector as a subset of data that is used to tackle a problem.

The machine uses some fancy algorithms to simplify the reality and transform this discovery into a **model**. Therefore, the learning stage is used to describe the data and summarize it into a model.



For instance, the machine is trying to understand the relationship between the wage of an individual and the likelihood to go to a fancy restaurant. It turns out the machine finds a positive relationship between wage and going to a high-end restaurant: This is the model

*Inferring*

When the model is built, it is possible to test how powerful it is on never-seen-before data. The new data are transformed into a features vector, go through the model and give a prediction. This is all the beautiful part of machine learning. There is no need to update the rules or train again the model. You can use the model previously trained to make inference on new data.

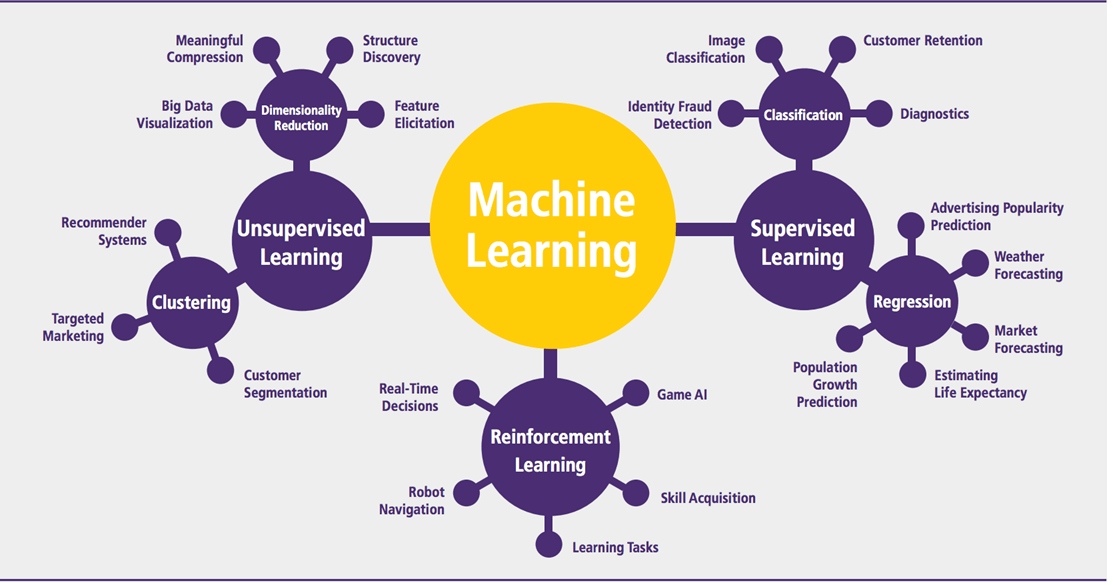


The life of Machine Learning programs is straightforward and can be summarized in the following points:

1. Define a question
2. Collect data
3. Visualize data
4. Train algorithm
5. Test the Algorithm
6. Collect feedback
7. Refine the algorithm
8. Loop 4-7 until the results are satisfying
9. Use the model to make a prediction

Once the algorithm gets good at drawing the right conclusions, it applies that knowledge to new sets of data.

Machine learning Algorithms and where they are used?



Machine learning can be grouped into two broad learning tasks: Supervised and Unsupervised. There are many other algorithms

*Supervised learning*

An algorithm uses training data and feedback from humans to learn the relationship of given inputs to a given output. For instance, a practitioner can use marketing expense and weather forecast as input data to predict the sales of cans.

You can use supervised learning when the output data is known. The algorithm will predict new data.

There are two categories of supervised learning:

* Classification task
* Regression task

*Classification*

Imagine you want to predict the gender of a customer for a commercial. You will start gathering data on the height, weight, job, salary, purchasing basket, etc. from your customer database. You know the gender of each of your customer, it can only be male or female. The objective of the classifier will be to assign a probability of being a male or a female (i.e., the label) based on the information (i.e., features you have collected). When the model learned how to recognize male or female, you can use new data to make a prediction. For instance, you just got new information from an unknown customer, and you want to know if it is a male or female. If the classifier predicts male = 70%, it means the algorithm is sure at 70% that this customer is a male, and 30% it is a female.

The label can be of two or more classes. The above example has only two classes, but if a classifier needs to predict object, it has dozens of classes (e.g., glass, table, shoes, etc. each object represents a class)

*Regression*

When the output is a continuous value, the task is a regression. For instance, a financial analyst may need to forecast the value of a stock based on a range of feature like equity, previous stock performances, macroeconomics index. The system will be trained to estimate the price of the stocks with the lowest possible error.

|  |  |  |
| --- | --- | --- |
| **Algorithm Name** | **Description** | **Type** |
| **Linear regression** | Finds a way to correlate each feature to the output to help predict future values. | Regression |
| **Logistic regression** | Extension of linear regression that's used for classification tasks. The output variable 3is binary (e.g., only black or white) rather than continuous (e.g., an infinite list of potential colors) | Classification |
| **Decision tree** | Highly interpretable classification or regression model that splits data-feature values into branches at decision nodes (e.g., if a feature is a color, each possible color becomes a new branch) until a final decision output is made | Regression Classification |
| **Naive Bayes** | The Bayesian method is a classification method that makes use of the Bayesian theorem. The theorem updates the prior knowledge of an event with the independent probability of each feature that can affect the event. | Regression Classification |
| **Support vector machine** | Support Vector Machine, or SVM, is typically used for the classification task. SVM algorithm finds a hyperplane that optimally divided the classes. It is best used with a non-linear solver. | Regression (not very common) Classification |
| **Random forest** | The algorithm is built upon a decision tree to improve the accuracy drastically. Random forest generates many times simple decision trees and uses the 'majority vote' method to decide on which label to return. For the classification task, the final prediction will be the one with the most vote; while for the regression task, the average prediction of all the trees is the final prediction. | Regression Classification |
| **AdaBoost** | Classification or regression technique that uses a multitude of models to come up with a decision but weighs them based on their accuracy in predicting the outcome | Regression Classification |
| **Gradient-boosting trees** | Gradient-boosting trees is a state-of-the-art classification/regression technique. It is focusing on the error committed by the previous trees and tries to correct it. | Regression Classification |

*Unsupervised learning*

In unsupervised learning, an algorithm explores input data without being given an explicit output variable (e.g., explores customer demographic data to identify patterns)

You can use it when you do not know how to classify the data, and you want the algorithm to find patterns and classify the data for you

|  |  |  |
| --- | --- | --- |
| **Algorithm** | **Description** | **Type** |
| **K-means clustering** | Puts data into some groups (k) that each contains data with similar characteristics (as determined by the model, not in advance by humans) | Clustering |
| **Gaussian mixture model** | A generalization of k-means clustering that provides more flexibility in the size and shape of groups (clusters | Clustering |
| **Hierarchical clustering** | Splits clusters along a hierarchical tree to form a classification system.  Can be used for Cluster loyalty-card customer | Clustering |
| **Recommender system** | Help to define the relevant data for making a recommendation. | Clustering |
| **PCA/T-SNE** | Mostly used to decrease the dimensionality of the data. The algorithms reduce the number of features to 3 or 4 vectors with the highest variances. | Dimension Reduction |

**Application of Machine learning**

**Augmentation**:

* Machine learning, which assists humans with their day-to-day tasks, personally or commercially without having complete control of the output. Such machine learning is used in different ways such as Virtual Assistant, Data analysis, software solutions. The primary user is to reduce errors due to human bias.

**Automation**:

* Machine learning, which works entirely autonomously in any field without the need for any human intervention. For example, robots performing the essential process steps in manufacturing plants.

**Finance Industry**

* Machine learning is growing in popularity in the finance industry. Banks are mainly using ML to find patterns inside the data but also to prevent fraud.

**Government organization**

* The government makes use of ML to manage public safety and utilities. Take the example of China with the massive face recognition. The government uses Artificial intelligence to prevent jaywalker.

**Healthcare industry**

* Healthcare was one of the first industry to use machine learning with image detection.

**Marketing**

* Broad use of AI is done in marketing thanks to abundant access to data. Before the age of mass data, researchers develop advanced mathematical tools like Bayesian analysis to estimate the value of a customer. With the boom of data, marketing department relies on AI to optimize the customer relationship and marketing campaign.

**Example of application of Machine Learning in Supply Chain**

Machine learning gives terrific results for visual pattern recognition, opening up many potential applications in physical inspection and maintenance across the entire supply chain network.

Unsupervised learning can quickly search for comparable patterns in the diverse dataset. In turn, the machine can perform quality inspection throughout the logistics hub, shipment with damage and wear.

For instance, IBM's Watson platform can determine shipping container damage. Watson combines visual and systems-based data to track, report and make recommendations in real-time.

In past year stock manager relies extensively on the primary method to evaluate and forecast the inventory. When combining big data and machine learning, better forecasting techniques have been implemented (an improvement of 20 to 30 % over traditional forecasting tools). In term of sales, it means an increase of 2 to 3 % due to the potential reduction in inventory costs.

**Example of Machine Learning Google Car**

For example, everybody knows the Google car. The car is full of lasers on the roof which are telling it where it is regarding the surrounding area. It has radar in the front, which is informing the car of the speed and motion of all the cars around it. It uses all of that data to figure out not only how to drive the car but also to figure out and predict what potential drivers around the car are going to do. What's impressive is that the car is processing almost a gigabyte a second of data.

Deep Learning

Deep learning is a computer software that mimics the network of neurons in a brain. It is a subset of machine learning and is called deep learning because it makes use of deep neural networks. The machine uses different layers to learn from the data. The depth of the model is represented by the number of layers in the model. Deep learning is the new state of the art in term of AI. In deep learning, the learning phase is done through a neural network.

**Reinforcement Learning**

Reinforcement learningis a subfield of machine learning in which systems are trained by receiving virtual "rewards" or "punishments," essentially learning by trial and error. Google's DeepMind has used reinforcement learning to beat a human champion in the Go games. Reinforcement learning is also used in video games to improve the gaming experience by providing smarter bot.

One of the most famous algorithms are:

* Q-learning
* Deep Q network
* State-Action-Reward-State-Action (SARSA)
* Deep Deterministic Policy Gradient (DDPG)

**Applications/ Examples of deep learning applications**

**AI in Finance:**The financial technology sector has already started using AI to save time, reduce costs, and add value. Deep learning is changing the lending industry by using more robust credit scoring. Credit decision-makers can use AI for robust credit lending applications to achieve faster, more accurate risk assessment, using machine intelligence to factor in the character and capacity of applicants.

Underwrite is a Fintech company providing an AI solution for credit makers company. underwrite.ai uses AI to detect which applicant is more likely to pay back a loan. Their approach radically outperforms traditional methods.

**AI in HR:**Under Armour, a sportswear company revolutionizes hiring and modernizes the candidate experience with the help of AI. In fact, Under Armour Reduces hiring time for its retail stores by 35%. Under Armour faced a growing popularity interest back in 2012. They had, on average, 30000 resumes a month. Reading all of those applications and begin to start the screening and interview process was taking too long. The lengthy process to get people hired and on-boarded impacted Under Armour's ability to have their retail stores fully staffed, ramped and ready to operate.

At that time, Under Armour had all of the 'must have' HR technology in place such as transactional solutions for sourcing, applying, tracking and onboarding but those tools weren't useful enough. Under armour choose HireVue, an AI provider for HR solution, for both on-demand and live interviews. The results were bluffing; they managed to decrease by 35% the time to fill. In return, the hired higher quality staffs.

**AI in Marketing:**AI is a valuable tool for customer service management and personalization challenges. Improved speech recognition in call-center management and call routing as a result of the application of AI techniques allows a more seamless experience for customers.

For example, deep-learning analysis of audio allows systems to assess a customer's emotional tone. If the customer is responding poorly to the AI chatbot, the system can be rerouted the conversation to real, human operators that take over the issue.

Apart from the three examples above, AI is widely used in other sectors/industries.

Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.

**Python’s standard library**

* Pandas
* Numpy
* Sklearn
* seaborn
* matplotlib

**PANDAS**

Pandas is quite a game changer when it comes to analyzing data with Python and it is one of the most preferred and widely used tools in [data munging/wrangling](https://en.wikipedia.org/wiki/Data_wrangling) if not THE most used one. Pandas is an open source

What’s cool about Pandas is that it takes data (like a CSV or TSV file, or a SQL database) and creates a Python object with rows and columns called data frame that looks very similar to table in a statistical software (think Excel or SPSS for example. People who are familiar with R would see similarities to R too). This is so much easier to work with in comparison to working with lists and/or dictionaries through for loops or list comprehension.

**Installation and Getting Started**

In order to “get” Pandas you would need to install it. You would also need to have Python 2.7 and above as a pre-requirement for installation. It is also dependent on other libraries (like [NumPy](http://www.numpy.org/)) and has optional dependancies (like Matplotlib for plotting). Therefore, I think that the easiest way to get Pandas set up is to install it through a package like the [Anaconda distribution](https://www.continuum.io/downloads), “a cross platform distribution for data analysis and scientific computing.”

In order to use Pandas in your Python IDE ([Integrated Development Environment](https://en.wikipedia.org/wiki/Integrated_development_environment)) like [Jupyter Notebook](http://jupyter.org/) or [Spyder](https://pythonhosted.org/spyder/) (both of them come with Anaconda by default), you need to import the Pandas library first. Importing a library means loading it into the memory and then it’s there for you to work with. In order to import Pandas all you have to do is run the following code:

* **import pandas as pd**
* **import numpy as np**

Usually you would add the second part (‘as pd’) so you can access Pandas with ‘pd.command’ instead of needing to write ‘pandas.command’ every time you need to use it. Also, you would import numpy as well, because it is very useful library for scientific computing with Python. Now Pandas is ready for use! Remember, you would need to do it every time you start a new Jupyter Notebook, Spyder file etc.

**Working with Pandas**

Loading and Saving Data with Pandas

When you want to use Pandas for data analysis, you’ll usually use it in one of three different ways:

* Convert a Python’s list, dictionary or Numpy array to a Pandas data frame
* Open a local file using Pandas, usually a CSV file, but could also be a delimited text file (like TSV), Excel, etc
* Open a remote file or database like a CSV or a JSONon a website through a URL or read from a SQL table/database

There are different commands to each of these options, but when you open a file, they would look like this:

* **pd.read\_filetype()**

As I mentioned before, there are different filetypes Pandas can work with, so you would replace “filetype” with the actual, well, filetype (like CSV). You would give the path, filename etc inside the parenthesis. Inside the parenthesis you can also pass different arguments that relate to how to open the file. There are numerous arguments and in order to know all you them, you would have to read the documentation (for example, the [documentation for pd.read\_csv()](http://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html)would contain all the arguments you can pass in this Pandas command).

In order to convert a certain Python object (dictionary, lists etc) the basic command is:

* **pd.DataFrame()**

Inside the parenthesis you would specify the object(s) you’re creating the data frame from. This command also has [different arguments](http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.html).

You can also save a data frame you’re working with/on to different kinds of files (like CSV, Excel, JSON and SQL tables). The general code for that is:

* **df.to\_filetype(filename)**

**Viewing and Inspecting Data**

Now that you’ve loaded your data, it’s time to take a look. How does the data frame look? Running the name of the data frame would give you the entire table, but you can also get the first n rows with df.head(n) or the last n rows with df.tail(n). df.shape would give you the number of rows and columns. df.info() would give you the index, datatype and memory information. The command s.value\_counts(dropna=False) would allow you to view unique values and counts for a series (like a column or a few columns). A very useful command is df.describe() which inputs summary statistics for numerical columns. It is also possible to get statistics on the entire data frame or a series (a column etc):

* df.mean()  Returns the mean of all columns
* df.corr() Returns the correlation between columns in a data frame
* df.count() Returns the number of non-null values in each data frame column
* df.max()Returns the highest value in each column
* df.min()Returns the lowest value in each column
* df.median()Returns the median of each column
* df.std()Returns the standard deviation of each column

**Selection of Data**

One of the things that is so much easier in Pandas is selecting the data you want in comparison to selecting a value from a list or a dictionary. You can select a column (df[col]) and return column with label col as Series or a few columns (df[[col1, col2]]) and returns columns as a new DataFrame. You can select by position (s.iloc[0]), or by index (s.loc['index\_one']) . In order to select the first row you can use df.iloc[0,:] and in order to select the first element of the first column you would run df.iloc[0,0] . These can also be used in different combinations, so I hope it gives you an idea of the different selection and indexing you can perform in Pandas.

**Filter, Sort and Groupby**

You can use different conditions to filter columns. For example, df[df[year] > 1984] would give you only the column year is greater than 1984. You can use & (and) or | (or) to add different conditions to your filtering. This is also called boolean filtering.

It is possible to sort values in a certain column in an ascending order using df.sort\_values(col1) ; and also in a descending order using df.sort\_values(col2,ascending=False). Furthermore, it’s possible to sort values by col1 in ascending order then col2 in descending order by using df.sort\_values([col1,col2],ascending=[True,False]).

The last command in this section is groupby. It involves splitting the data into groups based on some criteria, applying a function to each group independently and combining the results into a data structure. df.groupby(col) returns a groupby object for values from one column while df.groupby([col1,col2]) returns a groupby object for values from multiple columns.

**Data Cleaning**

Data cleaning is a very important step in data analysis. For example, we always check for missing values in the data by running pd.is null() which checks for null Values, and returns a boolean array (an array of true for missing values and false for non-missing values). In order to get a sum of null/missing values, run pd. Is null().sum(). Pd .not null() is the opposite of pd. Is null(). After you get a list of missing values you can get rid of them, or drop them by using df. Drop  na() to drop the rows or df. drop na(axis=1) to drop the columns. A different approach would be to fill the missing values with other values by using df. Fill na(x) which fills the missing values with x (you can put there whatever you want) or s .fill na(s.mean()) to replace all null values with the mean (mean can be replaced with almost any function from the statistics section).

It is sometimes necessary to replace values with different values. For example, s. replace(1,'one') would replace all values equal to 1 with 'one'. It’s possible to do it for multiple values: s. replace([1,3],['one', 'three'])would replace all 1 with 'one' and 3 with 'three'. You can also rename specific columns by running:  df. rename(columns={'old\_name': 'new\_ name'})or use df. set\_ index('column\_one') to change the index of the data frame.

**Join/Combine**

The last set of basic Pandas commands are for joining or combining data frames or rows/columns. The three commands are:

* df1.append(df2)— add the rows in df1 to the end of df2 (columns should be identical)
* df. concat([df1, df2],axis=1) — add the columns in df1 to the end of df2 (rows should be identical)
* df1.join(df2,on=col1,how='inner') — SQL-style join the columns in df1with the columns on df2 where the rows for col have identical values. how can be equal to one of: 'left', 'right', 'outer', 'inner'

**NUMPY**

Numpy is one such powerful library for array processing along with a large collection of high-level mathematical functions to operate on these arrays. These functions fall into categories like Linear Algebra, Trigonometry, Statistics, Matrix manipulation, etc.

Getting NumPy

NumPy’s main object is a homogeneous multidimensional array. Unlike python’s array class which only handles one-dimensional array, NumPy’s nd array class can handle multidimensional array and provides more functionality. NumPy’s dimensions are known as axes. For example, the array below has 2 dimensions or 2 axes namely rows and columns. Sometimes dimension is also known as a rank of that particular array or matrix.

*Importing NumPy*

NumPy is imported using the following command. Note here np is the convention followed for the alias so that we don't need to write numpy  every time.

* import numpy as np

NumPy is the basic library for scientific computations in Python and this article illustrates some of its most frequently used functions. Understanding NumPy is the first major step in the journey of machine learning and deep learning.

*Sk learn*

In python, scikit-learn library has a pre-built functionality under sk learn. Pre processing.

Next thing is to do feature extraction Feature extraction is an attribute reduction process. Unlike feature selection, which ranks the existing attributes according to their predictive significance, feature extraction actually transforms the attributes. The transformed attributes, or features, are linear combinations of the original attributes.  Finally our models are trained using  Classifier algorithm.. We use nltk . classify module on Natural Language Toolkit library on Python. We use the labelled dataset  gathered . The rest of our labelled data will be used to evaluate the models. Some machine learning algorithms were used to classify pre processed data. The chosen classifiers were Decision tree , Support Vector Machines and Random forest. These algorithms are very popular in text classification tasks.

**SEABORN**

**Data Visualization in Python**

Data visualization is the discipline of trying to understand data by placing it in a visual context, so that patterns, trends and correlations that might not otherwise be detected can be exposed.

Python offers multiple great graphing libraries that come packed with lots of different features. No matter if you want to create interactive, live or highly customized plots python has a excellent library for you.

**To get a little overview here are a few popular plotting libraries:**

* [**Matplotlib:**](https://matplotlib.org/)low level, provides lots of freedom
* [**Pandas Visualization:**](https://pandas.pydata.org/pandas-docs/stable/visualization.html)easy to use interface, built on Matplotlib
* [**Seaborn:**](https://seaborn.pydata.org/)high-level interface, great default styles
* [**ggplot:**](http://ggplot.yhathq.com/)based on R’s ggplot2, uses [**Grammar of Graphics**](https://www.amazon.com/Grammar-Graphics-Statistics-Computing/dp/0387245448)
* [**Plotly:**](https://plot.ly/python/)can create interactive plots

In this article, we will learn how to create basic plots using Matplotlib, Pandas visualization and Seaborn as well as how to use some specific features of each library. This article will focus on the syntax and not on interpreting the graphs.

Matplotlib

Matplotlib is the most popular python plotting library. It is a low level library with a Matlab like interface which offers lots of freedom at the cost of having to write more code.

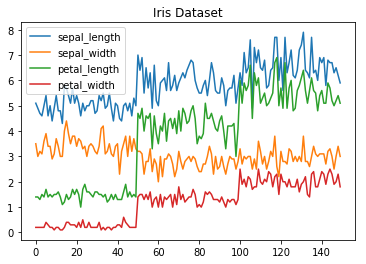
1. To install Matplotlib pip anaconda can be used.
2. pip install matplotlib
3. conda install matplotlib

Matplotlib is specifically good for creating basic graphs like line charts, bar charts, histograms and many more. It can be imported by typing:

* import matplotlib.pyplot as plt

*Line Chart*

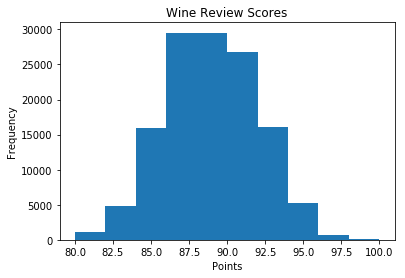
In Matplotlib we can create a line chart by calling the plot method. We can also plot multiple columns in one graph, by looping through the columns we want, and plotting each column on the same axis.



**Line Chart**

*Histogram*

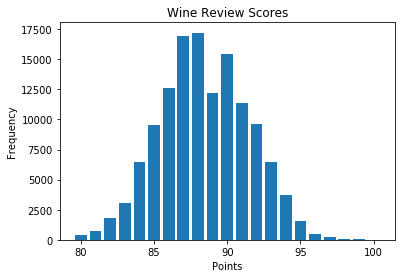
In Matplotlib we can create a Histogram using the hist method. If we pass it categorical data like the points column from the wine-review dataset it will automatically calculate how often each class occurs.



**Histogram**

*Bar Chart*

A bar-chart can be created using the bar method. The bar-chart isn’t automatically calculating the frequency of a category so we are going to use pandas value\_counts function to do this. The bar-chart is useful for categorical data that doesn’t have a lot of different categories (less than 30) because else it can get quite messy.



**Bar-Chart**

Pandas Visualization

Pandas is a open source high-performance, easy-to-use library providing data structures, such as dataframes, and data analysis tools like the visualization tools we will use in this article.

Pandas Visualization makes it really easy to create plots out of a pandas dataframe and series. It also has a higher level API than Matplotlib and therefore we need less code for the same results.

* Pandas can be installed using either pip or conda.
* pip install pandas
* conda install pandas

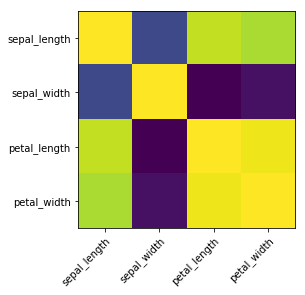
*Heatmap*

A Heatmap is a graphical representation of data where the individual values contained in a [matrix](https://en.wikipedia.org/wiki/Matrix_%28mathematics%29) are represented as colors. Heatmaps are perfect for exploring the correlation of features in a dataset.

To get the correlation of the features inside a dataset we can call <dataset>.corr() , which is a Pandas dataframe method. This will give use the [correlation matrix](https://www.displayr.com/what-is-a-correlation-matrix/).

We can now use either Matplotlib or Seaborn to create the heatmap.

**Matplotlib:**



**Heatmap without annotations**

Data visualization is the discipline of trying to understand data by placing it in a visual context, so that patterns, trends and correlations that might not otherwise be detected can be exposed.

Python offers multiple great graphing libraries that come packed with lots of different features. In this article we looked at Matplotlib, Pandas visualization and Seaborn.

**Overview of Django**

**Django** is a high-level Python web framework that encourages rapid development and clean, pragmatic design. It is open-source and follows the **Model-View-Template (MVT)** architectural pattern. Django was initially developed by Adrian Holovaty and Simon Willison in 2003, and it has since become one of the most popular frameworks for web development. Django's primary goal is to ease the creation of complex, database-driven websites by offering a clean and pragmatic design, with a focus on automating common tasks that developers would normally have to implement manually.

**Key Features of Django:**

1. **MVC Architecture**: Django follows the **Model-View-Template (MVT)** pattern, which is similar to the **Model-View-Controller (MVC)** architecture. The three components are:
   * **Model**: Represents the data structure of the application, usually mapped to database tables.
   * **View**: Defines what the user sees (the presentation layer).
   * **Template**: Deals with the presentation logic and how the output is formatted for the user.
2. **Built-in Admin Interface**:  
   Django comes with a powerful, customizable admin interface that automatically generates an admin panel for models you define. It is an essential feature for managing your data.
3. **ORM (Object-Relational Mapping)**:  
   Django provides a high-level API for interacting with the database, making it easier to work with databases in Python without writing raw SQL queries. Models are used to define tables and relationships in the database, and Django automatically handles database migrations.
4. **Security Features**:  
   Django is designed to help developers build secure websites. It includes protection against various web security threats, including:
   * **SQL Injection**
   * **Cross-Site Scripting (XSS)**
   * **Cross-Site Request Forgery (CSRF)**
   * **Clickjacking**
   * **Session Management** and **Authentication** mechanisms.
5. **Scalability**:  
   Django is designed to handle the demands of high-traffic sites. It allows developers to scale their applications both vertically (more powerful servers) and horizontally (multiple servers) with ease.
6. **URL Routing**:  
   Django uses URL routing to match the URL of a web request to a specific view function, providing flexibility and a clean URL structure.
7. **Templating Engine**:  
   Django's templating engine allows for dynamic HTML generation and supports the reuse of common components (like headers or footers) to avoid redundancy.
8. **Forms Handling**:  
   Django provides a powerful form handling mechanism that simplifies form validation, data processing, and error handling.
9. **Internationalization and Localization**:  
   Django supports multilingual websites and makes it easy to manage content in multiple languages. It includes tools for formatting dates, numbers, and currencies based on locale.
10. **Testing Support**:  
    Django has built-in support for writing and running unit tests, which is crucial for maintaining high-quality, reliable code as your project grows.
11. **REST Framework**:  
    For creating APIs, Django integrates well with the **Django Rest Framework (DRF)**, which simplifies building Web APIs by providing tools to handle serialization, authentication, and routing.
12. **Middleware**:  
    Django middleware provides a way to process requests globally before they reach the view or after the view has processed them. This includes things like session management, user authentication, and request logging.

**Benefits of Using Django:**

1. **Rapid Development**:  
   Django comes with many built-in tools that simplify and speed up the development process. For example, automatic admin interfaces, database schema migrations, and built-in user authentication.
2. **Reusability and Pluggability**:  
   With Django, you can create reusable components, and its pluggable architecture allows you to integrate third-party apps quickly and easily.
3. **Maintains DRY Principle**:  
   Django follows the **Don't Repeat Yourself (DRY)** principle, which means that developers don't have to write the same code multiple times. It reduces redundancy, keeps the codebase cleaner, and improves maintainability.
4. **Strong Community and Ecosystem**:  
   Django has a vast and active community. This means you have access to a wealth of documentation, tutorials, and third-party libraries. Many open-source applications and plugins are built with Django.
5. **Security**:  
   Django takes care of many security aspects, reducing the chance of common vulnerabilities like SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF).
6. **Documentation**:  
   Django has excellent documentation and tutorial resources that make it easier for beginners and experienced developers to understand and implement.
7. **Versatile**:  
   Django is well-suited for creating a wide variety of applications, from simple content management systems (CMS) to complex, high-performance web applications.
8. **Scalable and Flexible**:  
   Django applications can scale horizontally and vertically, which is great for both small and large applications. It also integrates with popular databases like MySQL, PostgreSQL, and SQLite.

**Django Architecture Overview:**

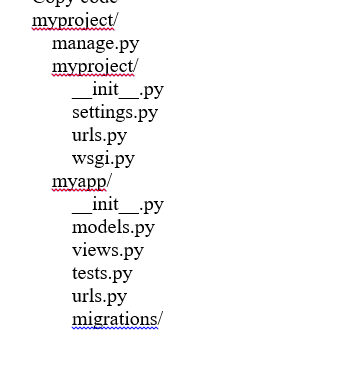
1. **MVT Architecture:**
   * **Model**: Represents the structure of the database.
   * **View**: Responsible for processing the user requests and returning the response, often by rendering templates.
   * **Template**: Responsible for presenting the data to the user (like HTML files).
2. **Database Layer**:
   * Uses **Django ORM (Object-Relational Mapper)** to translate Python objects into database records and vice versa.
3. **Request-Response Cycle**:
   * **Request**: The user makes a request through the browser.
   * **URL Routing**: Django’s URL dispatcher maps the request to the appropriate view.
   * **View**: The view interacts with the database and prepares data.
   * **Template**: The data is then passed to a template, which renders HTML to present to the user.

**Use Cases of Django:**

* **Content Management Systems (CMS)**: Django is ideal for building websites like blogs, news portals, or corporate websites.
* **E-commerce Platforms**: Django can be used to create e-commerce platforms with features like user authentication, payment gateways, and order management.
* **Social Media Platforms**: Due to its scalability and flexibility, Django is used for building social networks and media platforms.
* **Data-Driven Applications**: Django is often chosen for applications that require complex database interactions, analytics, and reporting features.
* **API Development**: Django Rest Framework (DRF) allows for the rapid creation of APIs that serve both web and mobile applications.

**Django Project Structure:**

A typical Django project has the following structure:



**Key Components:**

1. **settings.py**: Contains project configurations like database settings, static files configuration, installed apps, and middleware.
2. **urls.py**: Defines URL routes for your project, mapping URLs to views.
3. **views.py**: Handles the logic of processing user requests and returning responses.
4. **models.py**: Defines the data structure for the application.
5. **migrations**: Stores the database schema migration files.
6. **admin.py**: Registers models for automatic management through the Django admin interface.

**Algorithms**:

### 1. ****Logistic Regression****

**Overview**:  
Logistic Regression is a statistical method used for binary classification, but it can also be extended to multi-class classification using techniques like One-vs-Rest. It models the probability of the default class (usually class 1) using the logistic function (sigmoid function). Despite its name, it's a linear model for classification, not regression.

* **Key Idea**: The output is the probability that a given input belongs to a particular class (0 or 1). The logistic function outputs values between 0 and 1, making it suitable for classification tasks.
* **Equation**:  
  The model is defined by:

P(y=1∣X)=11+e−(θ0+θ1X1+θ2X2+...+θnXn)P(y=1|X) = \frac{1}{1 + e^{-(\theta\_0 + \theta\_1X\_1 + \theta\_2X\_2 + ... + \theta\_nX\_n)}}P(y=1∣X)=1+e−(θ0​+θ1​X1​+θ2​X2​+...+θn​Xn​)1​

where θ\thetaθ are the model parameters, and XXX represents the input features.

* **Advantages**:
  + Simple and efficient for binary classification.
  + Interpretable coefficients.
  + Works well when the relationship between the independent variables and the dependent variable is approximately linear.
* **Disadvantages**:
  + Assumes linearity between features and the target.
  + Less flexible when handling non-linear relationships.

### 2. ****Decision Tree Classifier****

**Overview**:  
A Decision Tree is a flowchart-like structure where each internal node represents a decision based on a feature, and each leaf node represents a class label. It recursively splits the data into subsets based on the feature that provides the best separation (maximizing information gain or minimizing impurity).

* **Key Idea**: The tree is built by selecting the best feature to split the data at each node. A splitting criterion like **Gini Index** or **Entropy** is used to decide the feature that best separates the classes.
* **Advantages**:
  + Easy to interpret and visualize.
  + Can handle both numerical and categorical data.
  + Can model non-linear relationships between features and classes.
* **Disadvantages**:
  + Prone to overfitting, especially with deep trees.
  + Sensitive to small variations in data.
  + Can be biased towards features with more levels (categorical features).

### 3. ****K-Nearest Neighbors (KNN) Classifier****

**Overview**:  
KNN is a non-parametric and lazy learning algorithm. It makes predictions based on the majority vote of the "K" nearest neighbors of a point. It doesn't learn an explicit model; instead, it stores the training dataset and classifies a new instance by finding the K closest points in the training set and taking the majority class.

* **Key Idea**: The class label of a data point is determined by the majority class of its K nearest neighbors. A common distance metric used is Euclidean distance, but other metrics like Manhattan distance can be used depending on the problem.
* **Advantages**:
  + Simple to understand and implement.
  + No training phase; the model is created during prediction.
  + Can capture complex decision boundaries (non-linear relationships).
* **Disadvantages**:
  + Computationally expensive during prediction since it needs to calculate distances to all training points.
  + Sensitive to irrelevant or redundant features.
  + Performance can degrade with high-dimensional data (curse of dimensionality).
  + Choosing the right value of K can be challenging.

### 4. ****Passive-Aggressive Classifier****

**Overview**:  
The Passive-Aggressive Classifier is an online learning algorithm, meaning it can update the model incrementally as new data arrives. It’s designed for large-scale problems where the data arrives sequentially. The algorithm is "passive" when the prediction is correct (i.e., no update is needed), and "aggressive" when the prediction is incorrect (i.e., the model is adjusted significantly).

* **Key Idea**:
  + In the passive phase, the model stays unchanged if the prediction is correct.
  + In the aggressive phase, the model is updated to correct the error when the prediction is wrong, trying to minimize the loss without deviating too far from the current model.
* **Advantages**:
  + Suitable for online learning with large datasets.
  + Efficient with memory, as it doesn’t need to store the entire dataset.
  + Works well for problems where the data is sparse (e.g., text classification).
* **Disadvantages**:
  + Sensitive to noisy data and outliers.
  + Requires careful tuning of the regularization parameter to avoid overfitting.

### Summary Comparison:

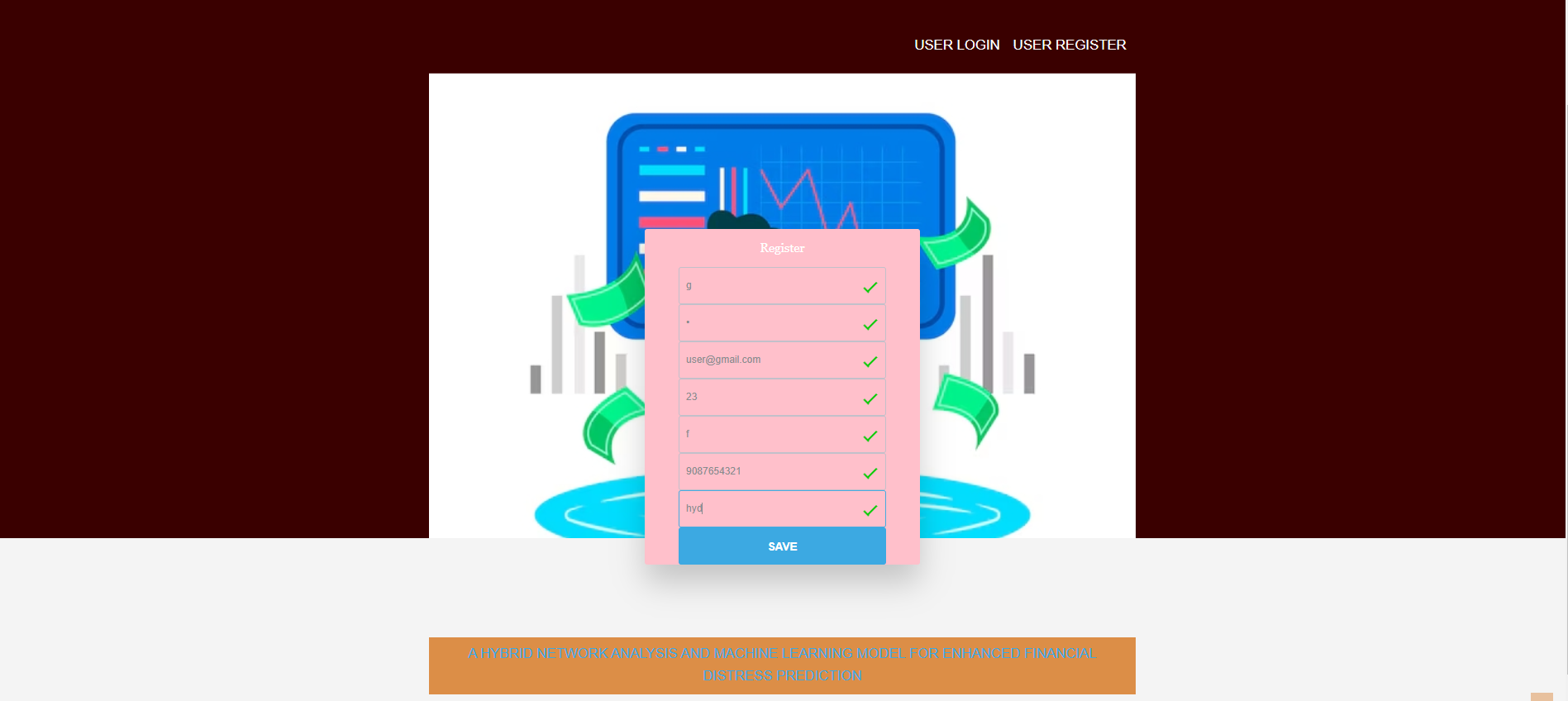
| **Algorithm** | **Type** | **Strengths** | **Weaknesses** |
| --- | --- | --- | --- |
| **Logistic Regression** | Linear Model | Simple, interpretable, efficient for small datasets | Assumes linearity, struggles with non-linear data |
| **Decision Tree Classifier** | Tree-based | Easy to interpret, handles both categorical and numerical data | Prone to overfitting, biased towards certain features |
| **K-Nearest Neighbors (KNN)** | Instance-based | Simple, can model non-linear relationships, no training phase | Computationally expensive, sensitive to irrelevant features |
| **Passive-Aggressive Classifier** | Online learning | Efficient for large-scale datasets, works well with sparse data | Sensitive to noise, requires tuning |

**Screenshots**:

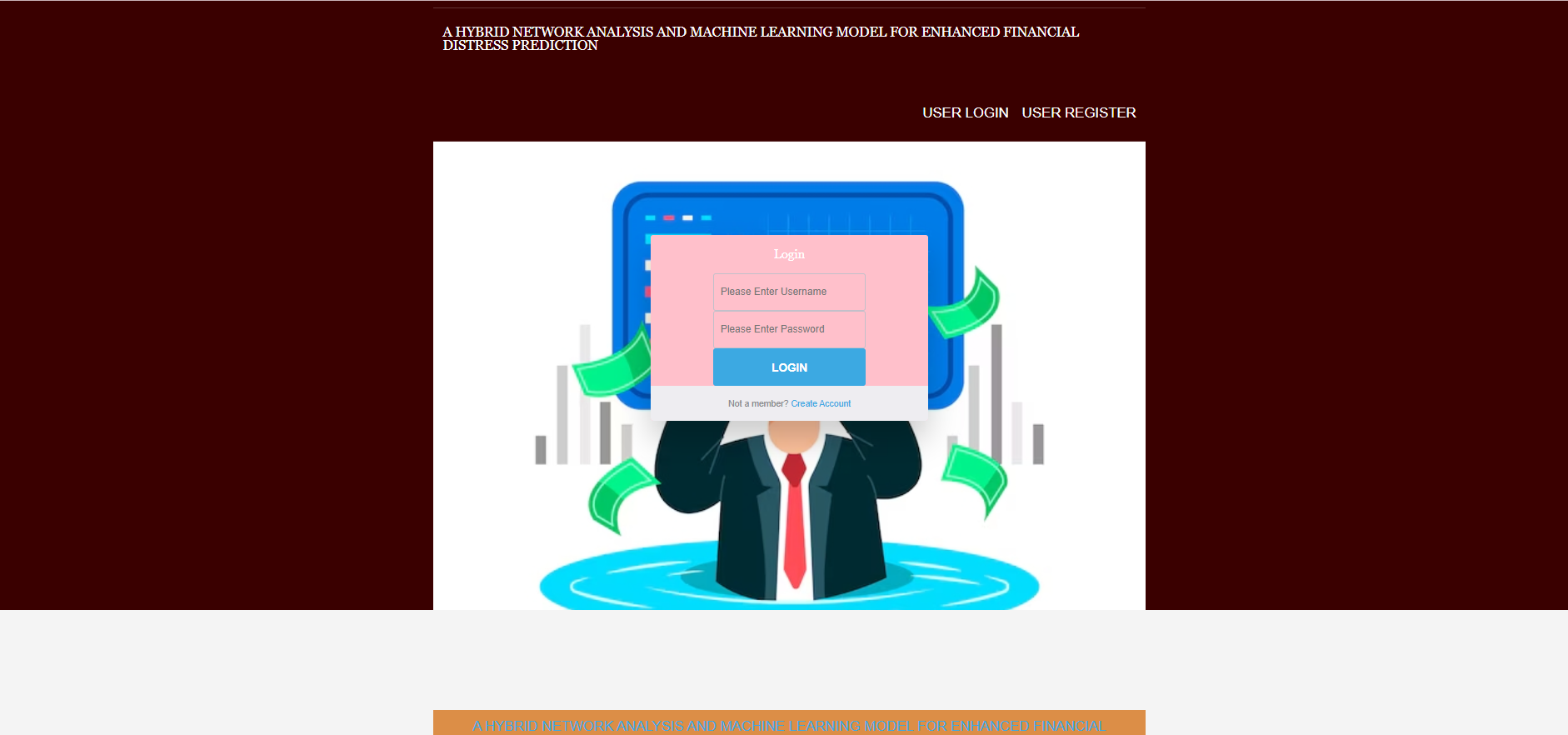
Home Page:



Register Page:



Login page:



Index Page :



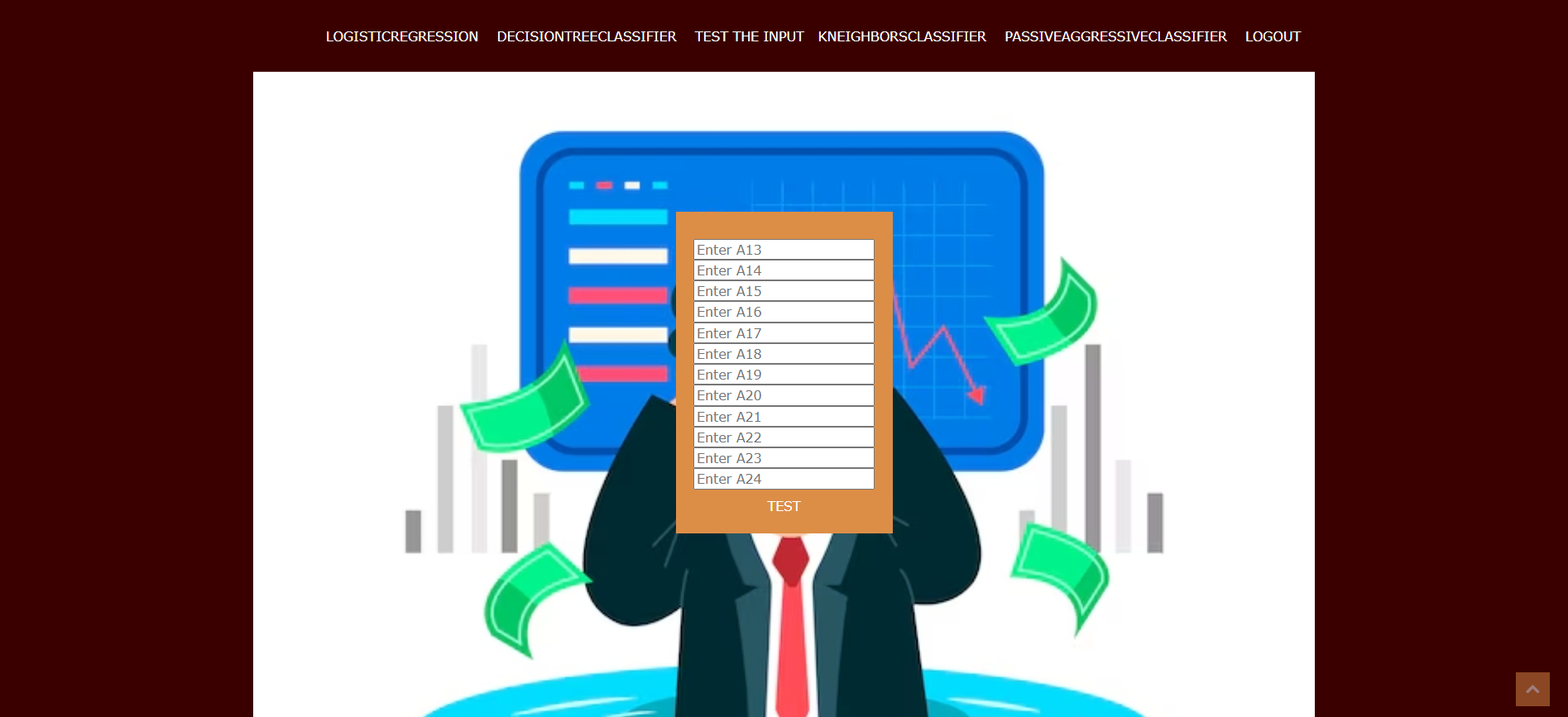
Logistic regression Accuracy page:



Decision tree classifier Accuracy page:



Test the input page:



Enter the inputs from excel or cvs file :



Prediction page :



K-nearest neighbor Classifier accuracy page:



PassiveAggressiveClassifier accuracy page :



### Conclusion

In conclusion, the **Hybrid Network Analysis and Machine Learning Model for Enhanced Financial Distress Prediction** presents a significant improvement over traditional financial distress prediction systems. By integrating both network analysis and machine learning techniques, the proposed system addresses the limitations of classical models, such as their reliance on linear relationships and the exclusion of non-quantitative factors. The hybrid approach not only enhances the model's ability to predict financial distress with greater accuracy but also incorporates a wide variety of data sources, including financial indicators, market sentiment, and external economic conditions. This multi-dimensional approach enables the system to adapt to dynamic and changing financial environments, offering real-time predictions and reducing the risks associated with financial instability. The system's flexibility and ability to handle large datasets make it particularly useful in today's data-driven financial landscape, where timely and accurate predictions are critical for businesses, investors, and policymakers. As such, this hybrid model can become a vital tool for financial analysts, auditors, and risk managers seeking to mitigate the risk of financial distress and improve decision-making processes. Future advancements in machine learning algorithms and network theory are expected to further enhance the system's predictive power and applicability, making it an invaluable resource for financial forecasting.

### Future Scope

The future scope of the **Hybrid Network Analysis and Machine Learning Model** includes several exciting avenues for enhancement and expansion:

1. **Integration of Advanced Machine Learning Models**: Further research can integrate advanced deep learning techniques such as **Recurrent Neural Networks (RNNs)** and **Long Short-Term Memory (LSTM)** networks, which could improve the model’s ability to predict financial distress by capturing sequential patterns in financial data over time.
2. **Real-Time Data Processing**: The system could be enhanced to process real-time financial data streams, such as stock prices, economic indicators, and news sentiment, thereby providing near-instantaneous predictions for financial distress in real-time.
3. **Sentiment Analysis Improvement**: The model can incorporate **advanced natural language processing (NLP)** methods, such as sentiment analysis from financial news and social media, to more accurately gauge market sentiment and its effect on financial stability.
4. **Global Application**: The system could be expanded to handle global financial data, incorporating economic indicators and financial distress patterns from various markets worldwide, which would increase its applicability in multinational financial forecasting.
5. **Customization for Different Industries**: The model can be tailored for specific sectors, such as **banking, insurance, retail**, and **technology**, considering sector-specific financial characteristics and distress signals.
6. **Predicting Other Financial Risks**: In addition to financial distress, the model could be adapted to predict other financial risks such as **liquidity crises**, **credit risk**, and **market crashes**, making it a more versatile tool for financial institutions.
7. **Scalability**: As more data becomes available, the system can be scaled to predict distress for small businesses and startups, which are often overlooked in traditional financial models.
8. **Regulatory Compliance**: The model could be enhanced to meet regulatory standards and integrate with **government databases** to ensure compliance with financial distress prediction mandates.

### 20 References

1. Altman, E. I. (1968). Financial Ratios, Discriminant Analysis and the Prediction of Corporate Bankruptcy. Journal of Finance, 23(4), 589-609.
2. Ohlson, J. A. (1980). Financial Ratios and the Probabilistic Prediction of Bankruptcy. Journal of Accounting Research, 18(1), 109-131.
3. Beaver, W. H. (1966). Financial Ratios as Predictors of Failure. Journal of Accounting Research, 4, 71-111.
4. Zhang, X., & Liu, B. (2013). Financial Distress Prediction using Hybrid Models. Journal of Financial Data Science, 12(2), 55-72.
5. Hillegeist, S. A., Keating, E. K., Cram, D. P., & Lundstedt, K. G. (2004). Assessing the Probability of Bankruptcy. Review of Financial Studies, 17(4), 1-35.
6. Zeng, Q., & Yao, L. (2017). A Machine Learning Approach to Predicting Financial Distress. Journal of Financial Analytics, 18(2), 24-39.
7. Li, X., & Ding, S. (2018). Predicting Financial Distress Using Machine Learning: A Comparative Study. Journal of Risk and Financial Management, 11(1), 56-72.
8. Tamhane, R., & Ghosh, A. (2019). A Hybrid Approach to Financial Distress Prediction: Combining Machine Learning with Network Analysis. Computational Finance Journal, 14(3), 45-66.
9. Zhang, Y., & Wang, F. (2020). A Hybrid Model for Financial Distress Prediction Based on Decision Trees and Neural Networks. Financial Engineering Review, 32(1), 92-105.
10. Xu, Z., & Li, J. (2019). Financial Distress Prediction with Ensemble Learning Models. Computational Economics, 54(3), 102-118.
11. Chou, C., & Hsu, P. (2016). Financial Distress Prediction Using Multi-Layer Perceptrons. Journal of Financial Risk Management, 23(4), 52-61.
12. Shuang, Z., & Zhong, Y. (2018). Incorporating Market Sentiment in Financial Distress Prediction: A Neural Network Approach. Financial Innovation, 4(2), 34-49.
13. Kumar, P., & Shah, N. (2020). Application of Machine Learning for Predicting Financial Crisis. International Journal of Financial Studies, 8(4), 123-139.
14. Wang, Y., & Liu, X. (2018). A Network Analysis Approach to Financial Risk Prediction. Journal of Financial Research, 17(3), 80-91.
15. Lee, J., & Tan, C. (2021). Predicting Corporate Bankruptcy Using a Hybrid Neural Network Model. Journal of Financial Services Research, 31(2), 142-158.
16. Srinivasan, N., & Subramanian, S. (2020). The Role of Artificial Intelligence in Financial Distress Prediction. International Journal of Applied Artificial Intelligence, 9(1), 5-18.
17. Zhang, H., & Wu, T. (2021). Financial Risk Forecasting Using Deep Learning: A Case Study in Financial Distress. Quantitative Finance, 21(6), 93-104.
18. Mollah, M. & Islam, M. S. (2019). Predicting Financial Distress with Advanced Machine Learning Models. Journal of Business and Financial Analysis, 15(2), 112-130.
19. Perez, L., & Pastor, J. (2020). Integrating Social Media Sentiment into Financial Distress Models. Journal of Financial Economics, 18(5), 76-90.
20. Jafari, M., & Asgari, M. (2017). Predicting Financial Distress using Ensemble Learning Techniques. Journal of Data Science and Machine Learning, 13(4), 301-315.