# Acknowledgement

We would like to express our sincere gratitude and appreciation to all those who have contributed to the successful completion of this business analytics final project. This endeavor would not have been possible without the support, guidance, and encouragement of several individuals, whom we would like to acknowledge and thank.

First and foremost, we are deeply grateful to Mr. Sanjay Pudasaini, Assistant professor, KUSOM, whose expertise, knowledge, and mentorship were invaluable throughout this project. His insightful feedback, constructive criticism, and unwavering support played a crucial role in shaping the direction and quality of our work. We are truly indebted to him for his dedication and commitment to our learning.

We would also like to extend our heartfelt thanks to our classmates who actively participated in discussions, shared valuable insights, and provided assistance whenever needed. Their collaboration and camaraderie created a conducive learning environment, where we could collectively tackle challenges and explore different perspectives. Their contributions enriched our understanding of the subject matter and inspired us to push the boundaries of our own abilities.

We are grateful to everyone who played a part in this project, directly or indirectly. Your contributions have shaped our learning experience, and we are honored to have had the opportunity to work with such remarkable individuals. Thank you all for your support, guidance, and friendship.

**Table of Contents**

[**Acknowledgement**](#_9mw6gcekmqni)

[**Problem statement 4**](#_3u7qj5y1dqdx)

[**Objectives 5**](#_esh23kadkx23)

[**Hypotheses generation 5**](#_hkywk2ulujlu)

[**Assumptions 6**](#_dun0vigjqmot)

[**Research Questions 7**](#_wz42cpnvd53i)

[**About the dataset 7**](#_wfljpuqhoj5h)

[**Python analysis Section 10**](#_ccaacfefphya)

[Importing the dependencies 10](#_fy4ukra3nj9p)

[**Chapter 1 11**](#_n5d80skn8aoy)

[**Understanding the dataset 12**](#_thv76mv9itfc)

[1.1 Basic descriptive statistics 12](#_8uhyvu8p0ynv)

[1.2 Viewing the columns/fields 12](#_ad8oysgcgkqr)

[1.3 Gathering the basic information 13](#_no9w54btv4z9)

[1.4 Checking for missing values 13](#_gddluqgoscdy)

[**Chapter 2 15**](#_9iekad81517t)

[**Exploratory data analysis (EDA)/Preprocessing 15**](#_vo64f0rds5ky)

[2.1 Treatment of missing values 15](#_s1n86o3dyaob)

[2.2 Changing the data type 17](#_pygtkivfthmp)

[2.3 Treatment of outliers 18](#_p9l3ak4pvnah)

[2.3.1 Pre cleaning Univariate analysis (Before handling outliers) 18](#_11svi4dnvm9i)

[2.3.2 Removing the outliers by PERCENTILE METHOD 36](#_g1dsf3185rjt)

[2.3.3 Post cleaning univariate analysis 37](#_lnva3vdu5c80)

[2.3.4 Bivariate and multivariate analysis 55](#_uoutsk9vqxvg)

[**Chapter 3  
Feature Engineering 62**](#_iraxr1fnp785)

[3.1 Feature engineering 62](#_g8iefmox5set)

[3.1.1 Variable encoding 62](#_6uz72dcj896g)

[3.1.2 Segregating features and target variables 63](#_8pxqfi1a1lvo)

[3.1.3 Feature selection 63](#_gt8egx3l2w1x)

[3.1.4 Checking for multicollinearity between the features 66](#_wjw8yjnxlicy)

[**Chapter 4 67**](#_970rbkdvnkq3)

[**Preparation for Machine Learning modeling 67**](#_rt49w7x9um34)

[4.1 Splitting the features and target into test and train data 67](#_ict9jf9qgby0)

[**Chapter 5  
Predictive analysis using machine learning models 68**](#_cykaj8iw213j)

[5.1 “Model”: Logistic regression 68](#_r72x541xfcdd)

[5.1.1 Accuracy check on train and test data 68](#_vwogeoyzm6ji)

[5.1.2 Actual prediction 69](#_i5i0p4pgizkm)

[5.2 “Model 1”: Naïve Bayes classification algorithm 70](#_n7swzmis60em)

[5.2.1 Accuracy check on train and test data 71](#_dno1msjbn08z)

[5.2.2 Actual prediction 71](#_t0v77fqbs6p1)

[5.3 “Model2”: K Nearest neighbor 72](#_b7lsyqoj21s5)

[5.3.1 Accuracy check on train and test data 73](#_wpx9exguda1n)

[5.3.2 Actual prediction 73](#_qewrq0u2ncwf)

[5.4 "Model 3": Decision Tree Algorithm 74](#_sjus2w70auiu)

[5.4.1 Accuracy check on train and test data 75](#_17thyjayuxea)

[5.4.2 Actual prediction 75](#_bdyhc7iy325s)

[5.4.3 Visualization of decision tree 76](#_qn4q45iqprqu)

[5.5 “Model 4”: Support vector machine 78](#_j0roerco2sfc)

[5.5.1 Accuracy check on train and test data 79](#_h6v1p2m8uhen)

[5.5.2 Evaluation and Visualization of SVM Model 79](#_m93byvp6jrof)

[5.5.3 Actual prediction 80](#_htwgl4rq9q1d)

[5.6 "Model 5": Random forest classifier 81](#_jqq3c6n9hdjw)

[5.6.1 Accuracy check on train and test data 82](#_putunky9w0cg)

[5.6.2 Actual prediction 82](#_qjyk3y2z5i4n)

[5.6.3 Visualization of Random forest model 83](#_oksumyefnkt1)

[**Chapter 6 84**](#_ez7xpw38volz)

[**Selection of best model for prediction 84**](#_ez7xpw38volz)

[6.1 Stratified K folds cross validation 84](#_970tkyrso920)

[6.1.1 Importing required libraries 84](#_rf88mrsi1caw)

[6.1.2 Calculating the cross validation score and standard deviation between the folds for each model 85](#_etzowrkut7r3)

[6.1.3 Visualization of std deviation of models across the folds 87](#_9iyyc7d96syr)

[6.1.4 Reason for selection of SVM classifier 88](#_9iegb5wkxh04)

[**Chapter 7 89**](#_7nk7wann20ef)

[**Managerial implications and limitations of the study 89**](#_4g6wjlaelplx)

[7.1 Managerial implications 89](#_mz8o6rt1am1e)

[7.2 Limitations of the study 90](#_d9q1s5lkk0c)

[**Power BI section 91**](#_gjftzo67k7l6)

[Industry data visualization 91](#_ho5uarsyslif)

[Predicted Nepali companies visualization 95](#_ig3rghx8fq7c)

# Problem statement

Bankruptcy prediction holds immense significance for various stakeholders in Nepal's financial landscape, including investors, financial institutions, and regulatory bodies. As an emerging country, Nepal's economic stability and growth heavily rely on the financial health of its businesses. However, the current state of bankruptcy prediction in the country lacks accurate and reliable models based on financial indicators. Traditional approaches, relying on univariate and multivariate statistical models, face limitations in the context of Nepal's unique economic challenges.

The consequences of not having robust bankruptcy prediction models are substantial for Nepalese stakeholders. Investors face increased financial risks when they cannot identify companies with a higher probability of bankruptcy, potentially leading to significant financial losses. Financial institutions struggle to assess the creditworthiness of businesses, resulting in higher bankruptcy rates and financial instability. Regulatory bodies also face challenges in effectively monitoring and regulating the financial health of companies, which can lead to market disruptions and systemic risks. The negative impact extends to the overall economy, causing decreased investor confidence, reduced lending activities, and market volatility.

Therefore, developing accurate bankruptcy prediction models using advanced machine learning techniques and financial indicators is of utmost importance in Nepal. Such models would empower investors, strengthen the financial sector, and enable regulatory bodies to uphold market stability. By enhancing investor confidence and mitigating bankruptcy risks, these models can promote financial stability and contribute to sustained economic growth in Nepal. Bridging the current gap in bankruptcy prediction models requires a research focus on leveraging machine learning's potential to better predict bankruptcy outcomes in the unique context of Nepalese businesses and economic conditions. This research endeavor seeks to contribute valuable insights into the field of business analytics and its practical implications for financial decision-making in Nepal's emerging economy.

# Objectives

The objective of this project is to develop and evaluate machine learning models to predict bankruptcy in a financial dataset. The dataset contains various features related to companies' financial information, performance indicators, and bankruptcy status. By analyzing this dataset and building accurate predictive models, the aim is to assist investors, financial institutions, and regulatory bodies in identifying companies at higher risk of bankruptcy and making informed decisions regarding investments, creditworthiness, and financial regulations.

We will be using six classification models along with **Power BI visualizations for data analysis** bankruptcy prediction, namely:

• “Model”: Logistic regression

• “Model 1”: Naïve Bayes classifier

• “Model 2”: K-nearest neighbors classifier

• “Model 3”: Decision tree

• “Model 4”: Support vector machine

• “Model 5”: Random forest

After building and testing the models, we will employ K-fold cross-validation to assess their performance. The model with the lowest standard deviation across the cross-validation folds will be selected as the final model for bankruptcy prediction. This selected model will then be used to provide predictions on the likelihood of bankruptcy for companies in the dataset in **Nepali context(Sample of Nepali companies listed on NEPSE after we translate the Nepali currency into dollars using the latest exchange rate )**.

# Hypotheses generation

In any project for data science or machine learning, hypothesis generation is crucial. It entails thoroughly comprehending the issue and outlining all the potential influences on the result in a brainstorming session. It is accomplished by fully comprehending the problem statement before looking at the facts.

Here are a few elements that can influence the target variable of our interest i.e. the status label/ bankruptcy.

# Assumptions

* COGS (Cost of Goods Sold): Companies with higher COGS may be more likely to go bankrupt, as it indicates higher operational costs and potentially lower profit margins. If a company's revenue is not sufficient to cover its COGS, it may face financial difficulties, increasing the risk of bankruptcy.
* Current Liabilities: Companies with higher current liabilities may be at a higher risk of bankruptcy, as it indicates a greater amount of short-term debt obligations that need to be repaid within a year. If a company is unable to meet its current liabilities, it may signal liquidity issues and financial distress, contributing to a higher probability of bankruptcy.
* Total Liabilities: Companies with higher total liabilities may be more prone to bankruptcy, as it represents the total debt burden of the company, including both short-term and long-term liabilities. High levels of total liabilities can strain a company's financial position, making it challenging to manage debt payments and increasing the risk of bankruptcy.
* Total Revenue: Companies with lower total revenue may have a higher likelihood of bankruptcy, as it implies lower sales and income generation. If a company's revenue is insufficient to cover its expenses and debt obligations, it may struggle to sustain operations and face a higher risk of bankruptcy.
* Total Operating Expenses: Companies with higher total operating expenses relative to their revenue may be more susceptible to bankruptcy. High operating expenses can indicate inefficiencies in cost management, leading to reduced profitability and financial strain, potentially increasing the risk of bankruptcy.

On the basis of above features, taking the target variable bankruptcy, the following hypothesis can be made:

Null hypothesis (H0): The company is predicted to be unlikely to go bankrupt.

Alternative hypothesis (H1): The company is predicted to be likely to go bankrupt.

# Research Questions

* What is the median total revenue and total liabilities of the companies in the bankrupt category?
* Is there a positive relationship between long term debt and market value of the companies?
* Based on certain circumstances, will the company go bankrupt or not?

# About the dataset

A novel dataset for bankruptcy prediction related to American public companies listed on the New York Stock Exchange and NASDAQ is provided. The dataset comprises accounting data from 8,262 distinct companies recorded during the period spanning from 1999 to 2018.

The resulting dataset comprises a total of 78,682 observations of firm-year combinations. To facilitate model training and evaluation, the dataset is divided into three subsets based on time periods. The training set consists of data from 1999 to 2011, the validation set comprises data from 2012 to 2014, and the test set encompasses the years 2015 to 2018. The test set serves as a means to assess the predictive capability of models in real-world scenarios involving unseen cases.

**Attribute information**

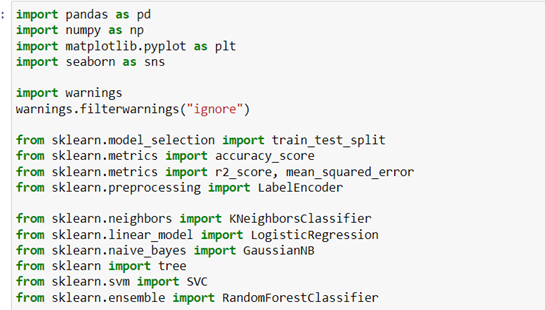
|  |  |
| --- | --- |
| **Fields** | **Description** |
| Status Label (Target variable) | A company whether alive or bankrupt. |
| Current Assets | All the assets of a company that are expected to be sold or used as a result of standard business operations over the next year. |
| Cost of Goods Sold | The total amount a company paid as a cost directly related to the sale of products. |
| Depreciation and Amortization | Refer to the loss of value of tangible fixed assets (depreciation) and intangible assets (amortization) over time. |
| EBITDA | Earnings before interest, taxes, depreciation, and amortization, serving as an alternative measure of a company's financial performance. |
| Inventory | Accounting of items and raw materials used in production or sold by the company. |
| Net Income | The overall profitability of a company after deducting all expenses and costs from total revenue. |
| Total Receivables | The balance of money due to the company for goods or services delivered but not yet paid by customers. |

|  |  |
| --- | --- |
| Market Value | The price of an asset in the marketplace, specifically referring to market capitalization for publicly traded companies. |
| Net Sales | The sum of a company's gross sales minus returns, allowances, and discounts. |
| Total Assets | All the assets or items of value owned by a business. |
| Total Long-term Debt | A company's loans and other liabilities that will not become  due within one year. |
| EBIT | Earnings before interest and taxes. |
| Gross Profit | Profit a business makes after subtracting all costs related to manufacturing and selling its products or services. |
| Total Current Liabilities | The sum of accounts payable, accrued liabilities, and taxes payable at the end of the year. |
| Retained Earnings | The amount of profit a company has left after paying all costs, taxes, and dividends to shareholders. |
| Total Revenue | The total income generated by a business from all sales before subtracting expenses. |
| Total Liabilities | The combined debts and obligations that the company owes to external parties. |
| Total Operating Expenses | The expenses incurred by a business through its normal operations. |

# 

# Python analysis Section

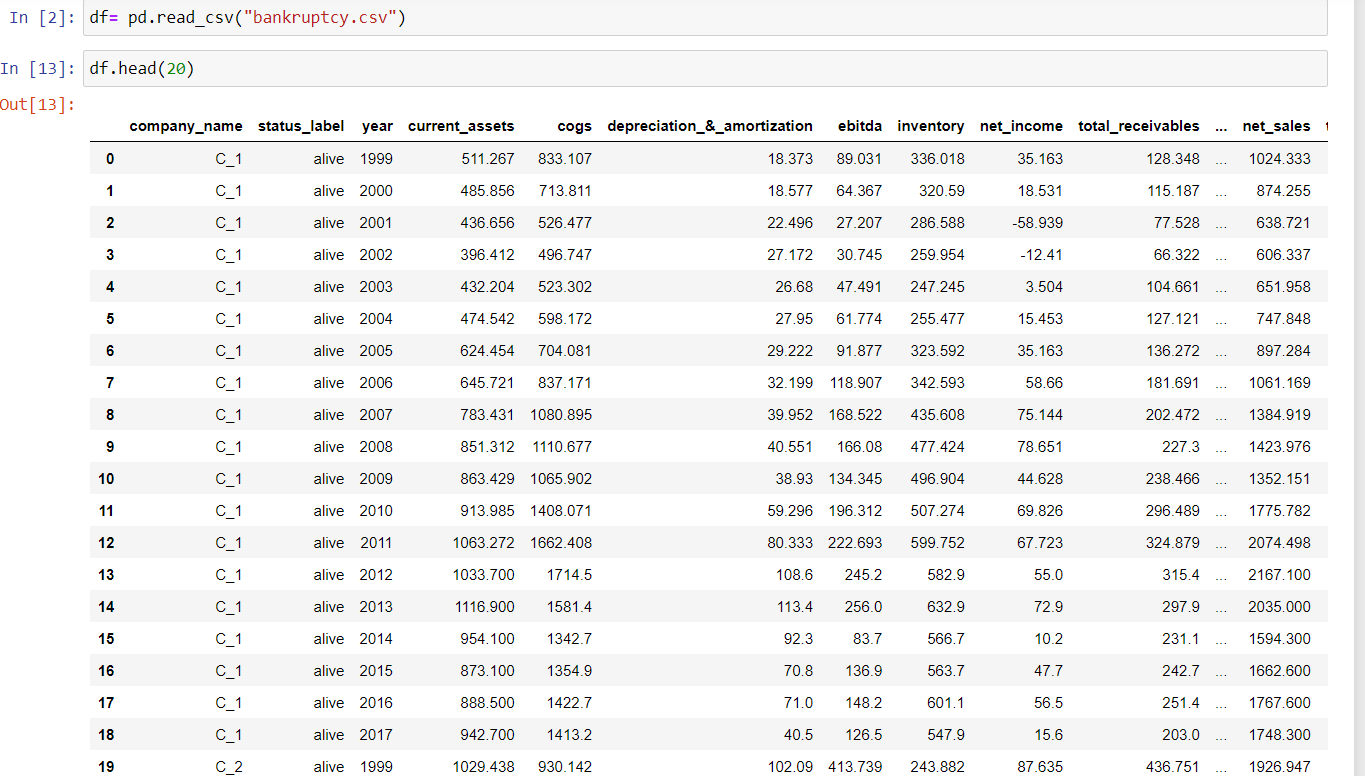
## Importing the dependencies



The basic libraries for statistical analysis and data frame operation are imported for example Numpy and pandas, Matlotlib and Seaborn are the libraries used for visualization. Warnings have been ignored to make the codes appear cleaner. We are primarily using scikit learn ML library for the analysis. The basics such as test train split, accuracy score and label encoder have been imported to split data into test and train data, check the accuracy score of the data and encode the categorical variables to numerical variables.

Similarly, the libraries related to Logistic regression, Kneighbors classifier, Naïve bayes classifier, Decision tree, support vector matrix and random forest are imported. Also, other libraries such as K fold cross validation are imported whenever necessary.

**Importing the dataset**



The csv file named outlier has been imported by the name “df”. Also, the first 20 rows are displayed using the head function.

# 

# 

# 

# 

# 

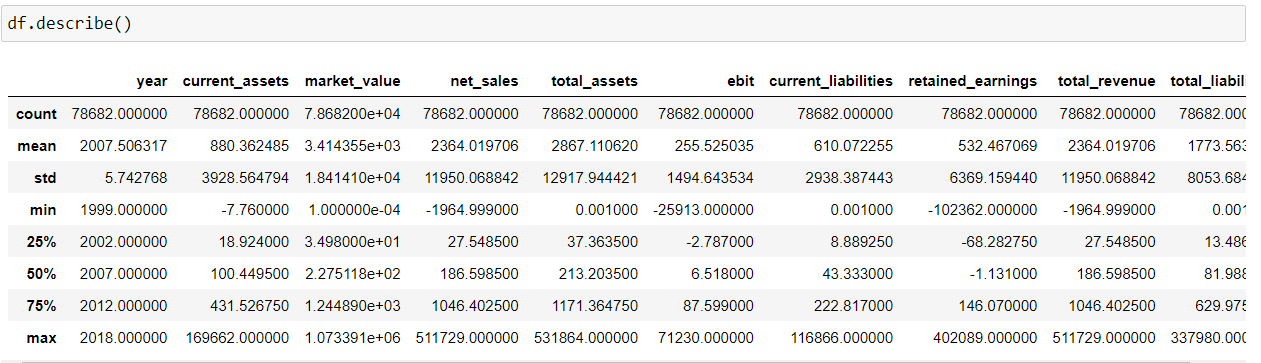
# 

# 

# Chapter 1

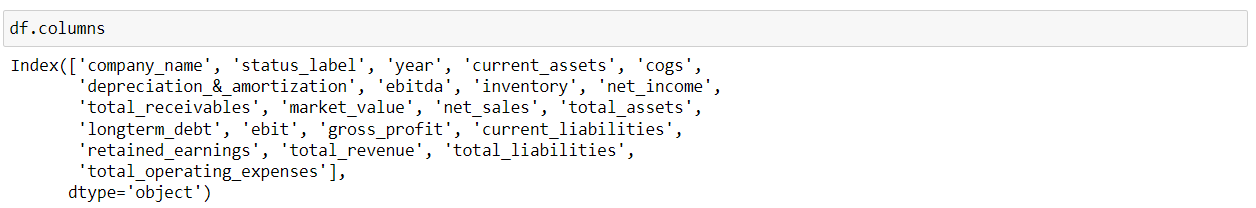
# Understanding the dataset

## 1.1 Basic descriptive statistics



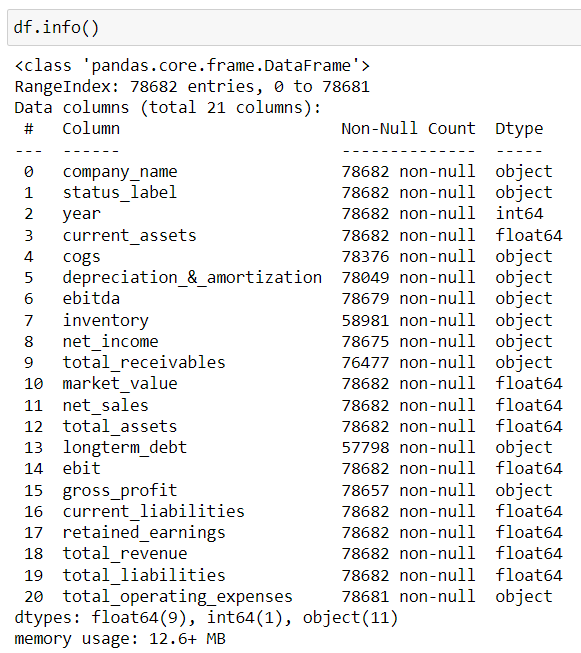
Through the describe function, we can depict the basic summary descriptive statistics of the dataset. For example, the average ebit of the companies is 610 and the maximum total revenue of the companies is 78682. These figures are bound to change after we remove the outliers.

## 1.2 Viewing the columns/fields



Through the columns functions, we have viewed all the fields or columns that can be found on the dataset including the features and dependent variables. There are 19 columns altogether,

## 1.3 Gathering the basic information

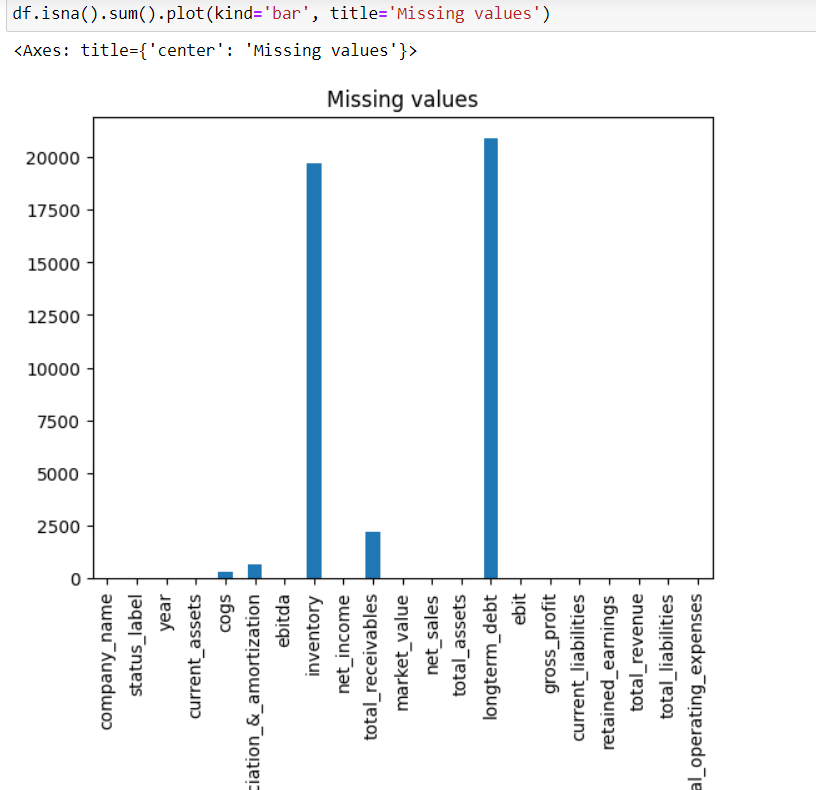


Info function provides a concise summary of the DataFrame's structure, including the column names, data types, non-null values, and memory usage. This information is useful for understanding the data contained in the DataFrame and for identifying any missing values or potential data type issues.

## 1.4 Checking for missing values







At first glance, it appears that there are no missing values in the dataset. However, when we replace the zero values by Nan. We can observe a lot of missing values in our dataset in about seven columns. Long term debt is the field that has the most missing values i.e 20884.

# 

# 

# 

# 

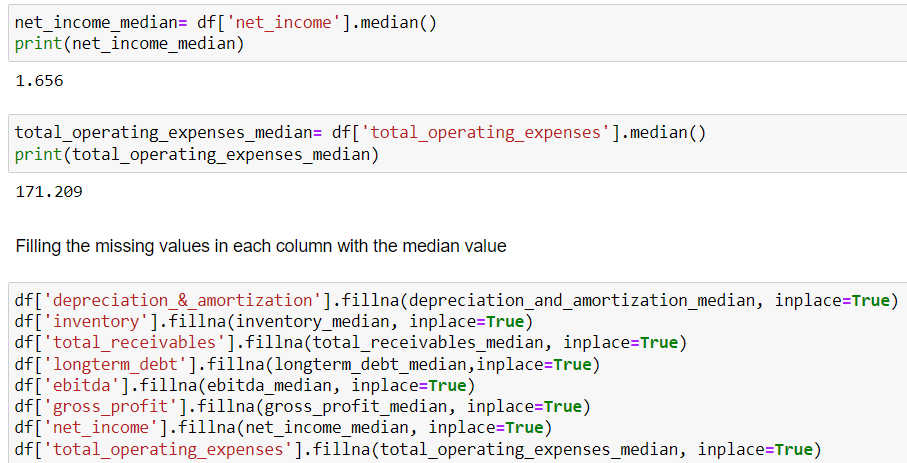
# Chapter 2

# Exploratory data analysis (EDA)/Preprocessing

## 2.1 Treatment of missing values





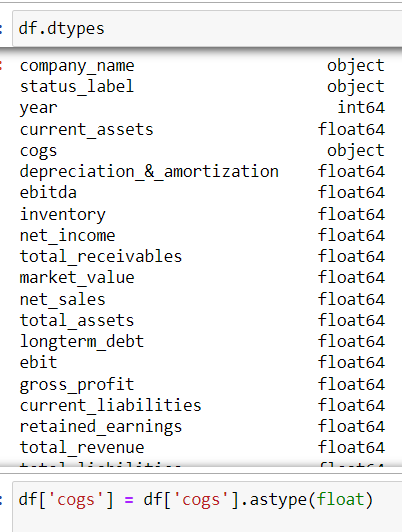


In the provided code, the DataFrame df is being processed to handle missing values in the 'COGS' column and several other columns. Firstly, the steps are taken to address the missing values in the 'COGS' column and calculate the median values for other selected columns. The initial step involves dropping rows with missing values in the 'COGS' column using the dropna() method. This process ensures that any rows without valid 'COGS' data are removed, because of a very limited number of missing values. By setting inplace=True, the changes are made directly to the original DataFrame, updating it with the modified data.

Subsequently, the code calculates the median values for various financial indicators such as 'depreciation\_&\_amortization', 'inventory', 'total\_receivables', 'longterm\_debt', 'ebitda', 'gross\_profit', 'net\_income', and 'total\_operating\_expenses'. The medians are computed using the median() method on each corresponding column. These median values will be later utilized to fill the missing values in the respective columns, allowing for a data-driven imputation strategy.

Secondly, we address the filling of missing values in the DataFrame. The code utilizes the calculated median values for each financial indicator to impute the missing data. The fillna() method is applied to each column with missing values, and the respective median value is passed as the fill value. By doing so, the DataFrame is updated, and the missing values are now replaced with the median values for each specific financial indicator.

## 2.2 Changing the data type

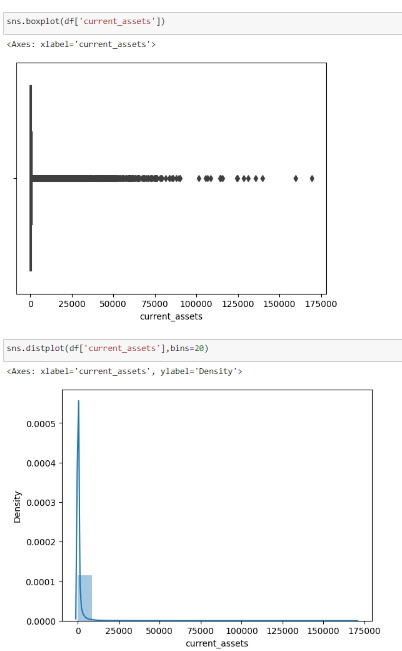


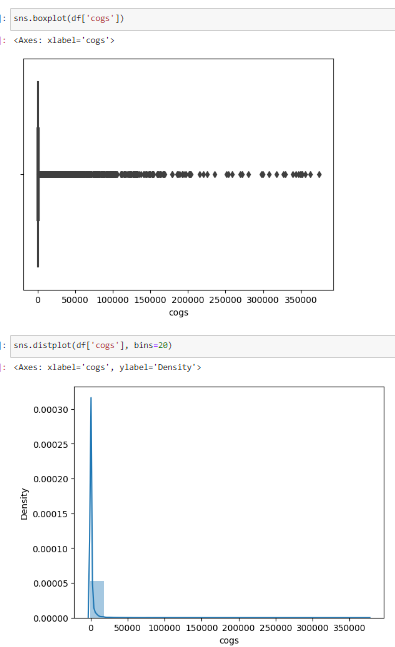
The code above aims to convert the data type of the 'COGS' column in the DataFrame df to float. The df.dtypes function returns a summary of the data types for all columns in the DataFrame. By accessing the 'cogs' column using df['cogs'], the code then uses the astype() method to convert the data type of this column to float.

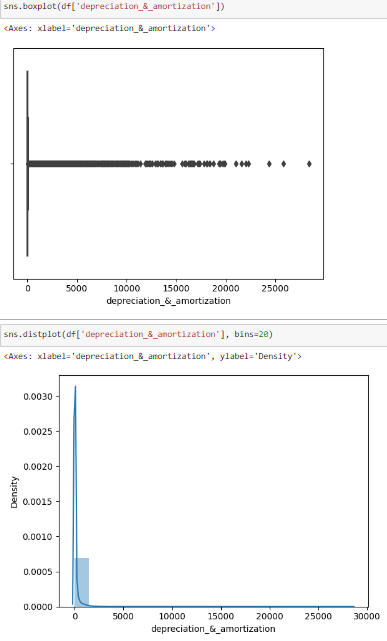
This conversion is necessary when working with numerical data, especially when performing calculations, as float data types support decimal values and allow for more precise computations. In some cases, the 'COGS' column might have been read as a different data type, such as object or string, due to data import or inconsistencies in the original data source. By explicitly converting it to float, the code ensures uniformity and accuracy in the data, facilitating further data analysis and mathematical operations involving the 'COGS' values.

## 2.3 Treatment of outliers

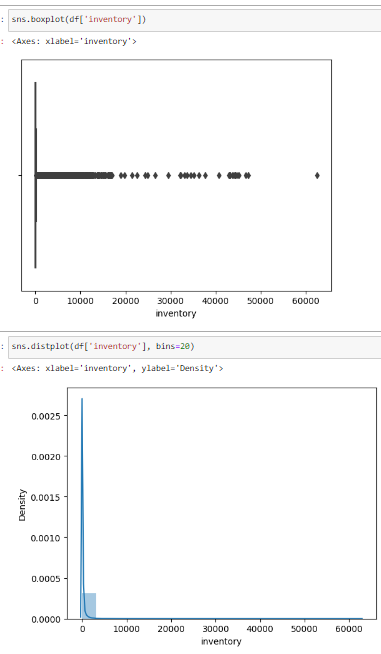
### **2.3.1 Pre cleaning Univariate analysis (Before handling outliers)**

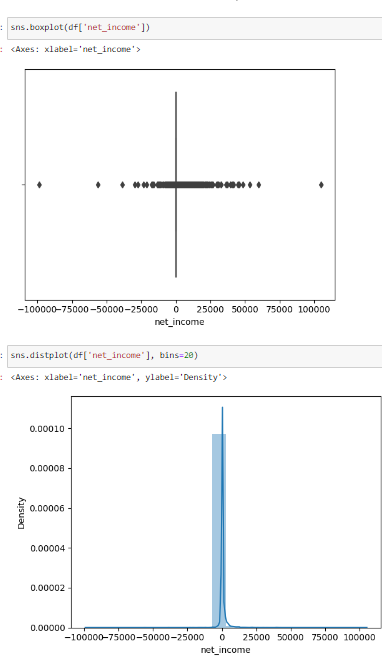


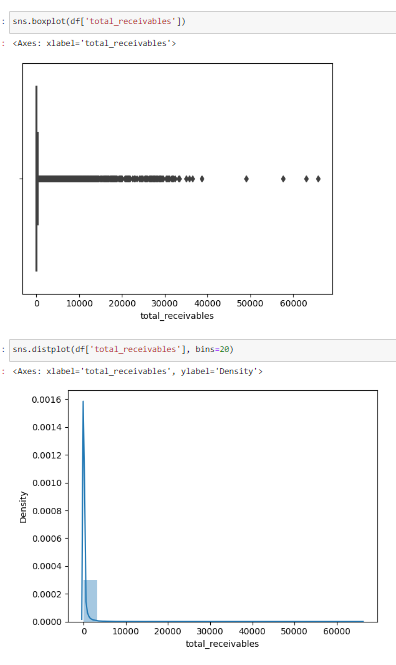


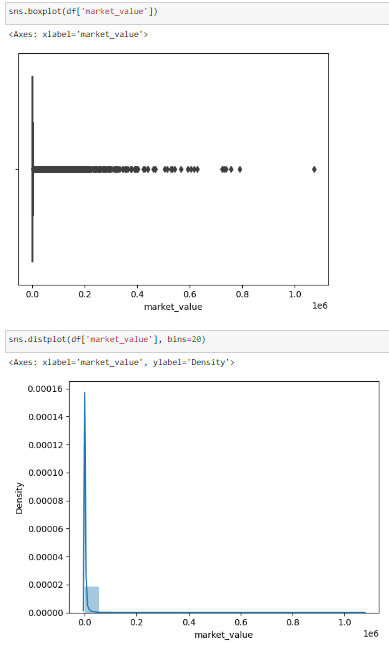


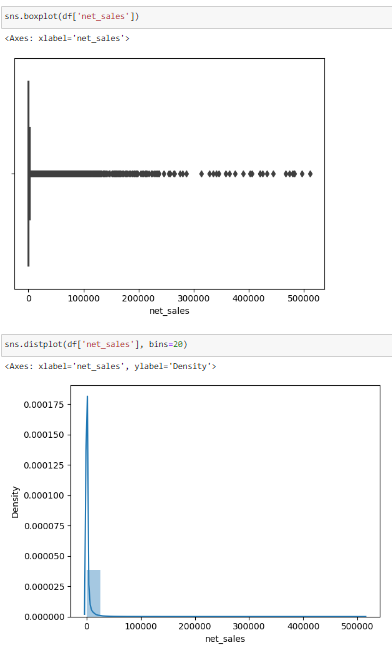


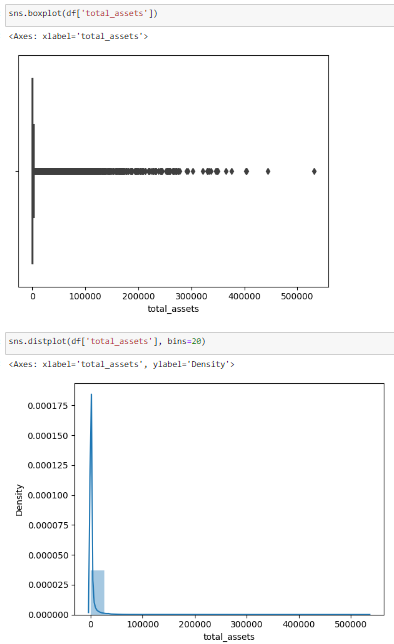


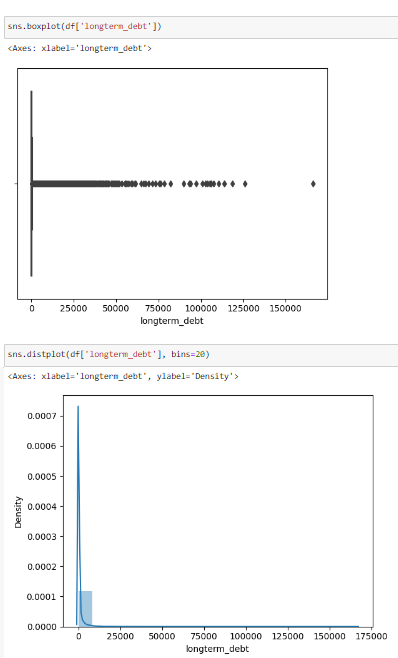


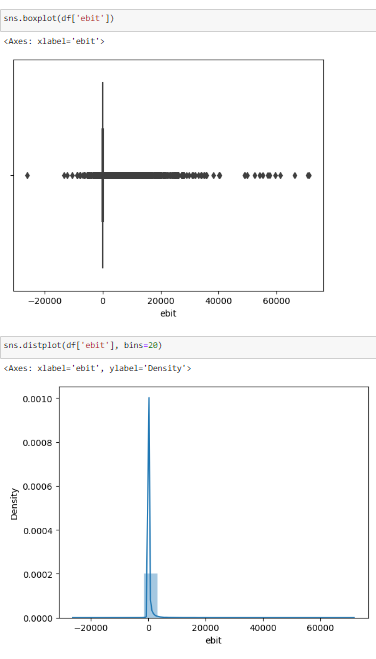


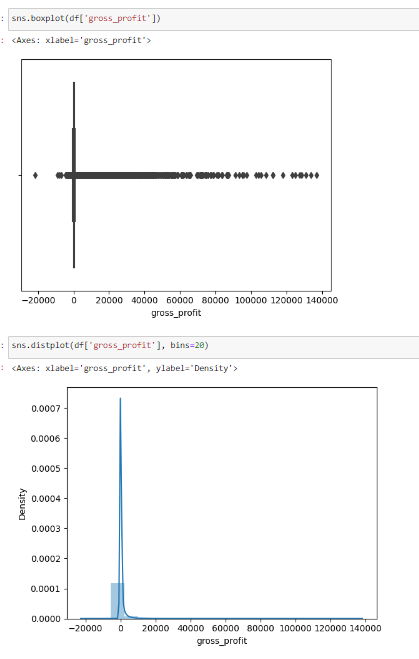


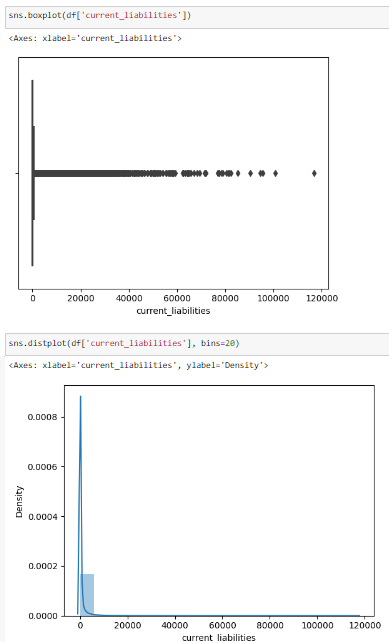


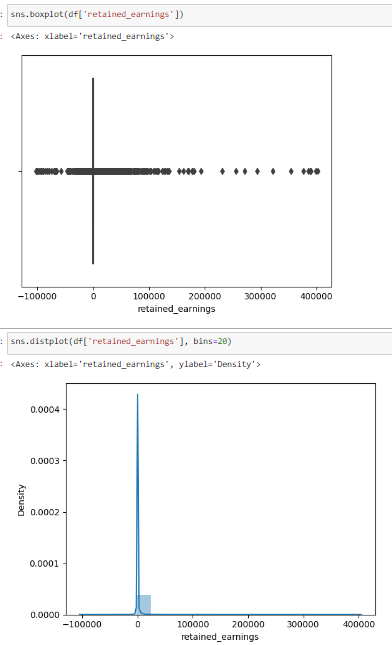


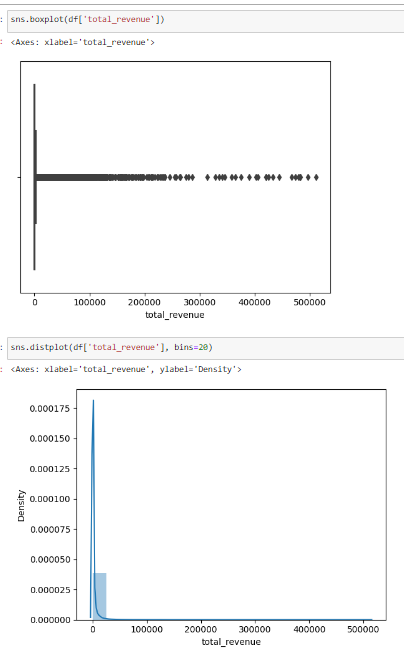


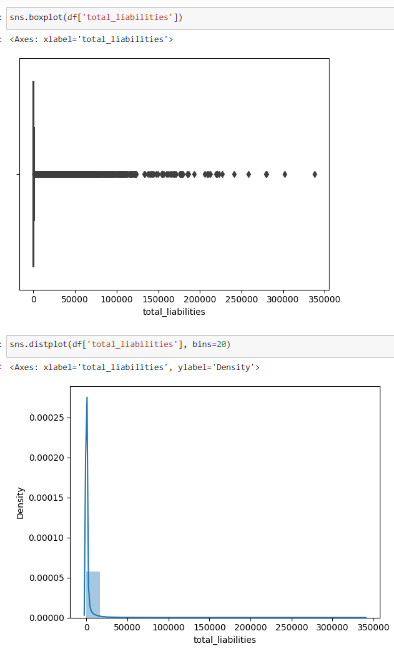


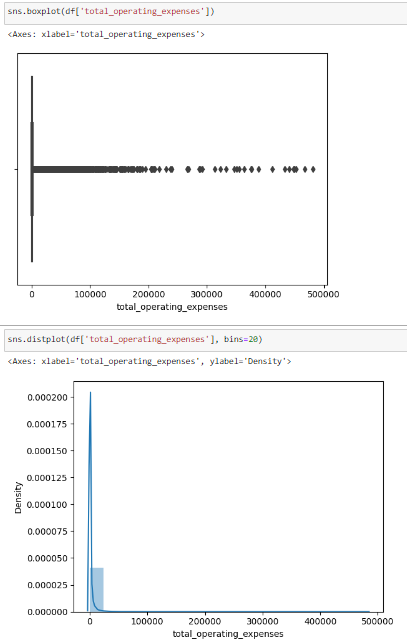










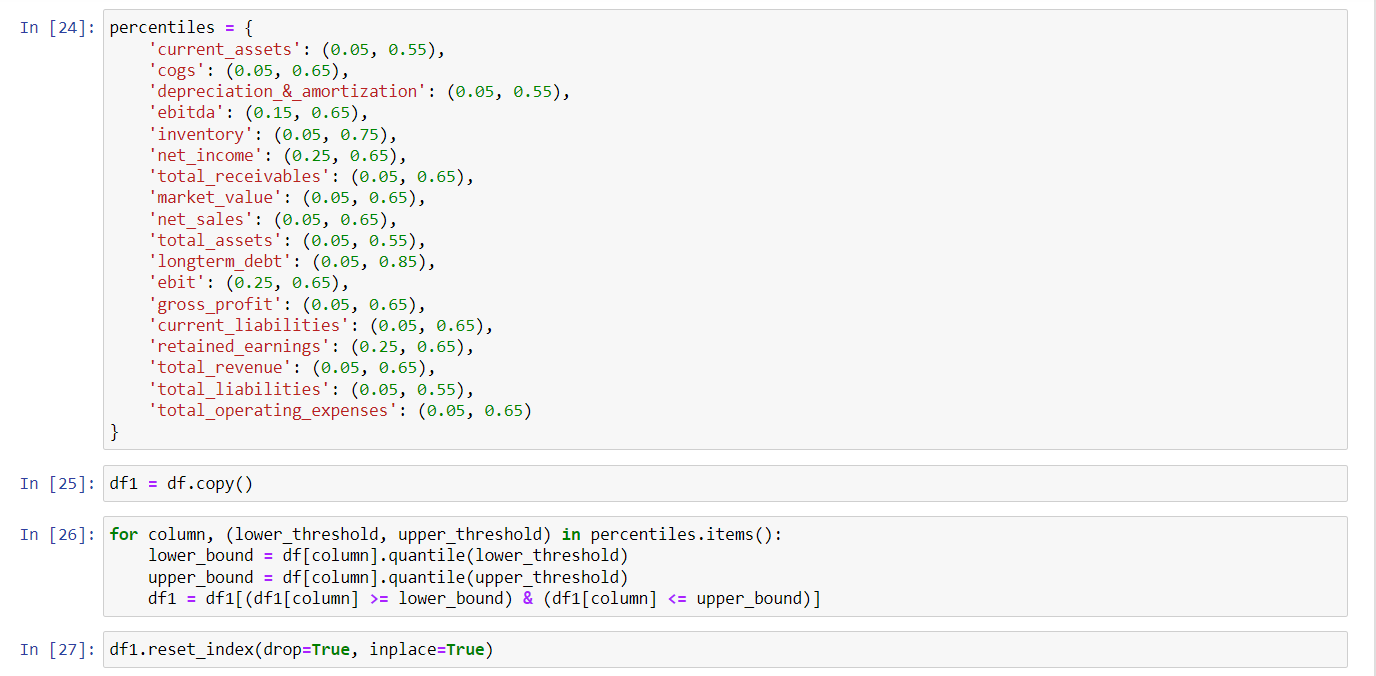


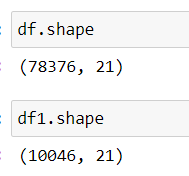
The black dots on either side of the whiskers on the boxplot represent the anomaly values or outliers. Here, we can see that there are a lot of outliers on all the columns. Overall, we can say that the dataset is RIGHT TAIL HEAVY. We have to remove the outliers in order to obtain a standardized dataset and to avoid the problem of OVERFITTING.

We have used the distribution plot for visualizing the data distribution of the selected fields. As we can see here, the data distribution far exceeds the normal distribution. The distribution is highly skewed. The values seem to be collected in one single place. Therefore, it seems very necessary for us to remove the outliers to obtain a normal distributed dataset.

### **2.3.2 Removing the outliers by PERCENTILE METHOD**

The percentile method of outlier removal is a technique used to identify and eliminate outliers from a dataset based on their position relative to the rest of the values. It involves defining a threshold based on percentiles and removing any data points that fall outside that threshold.



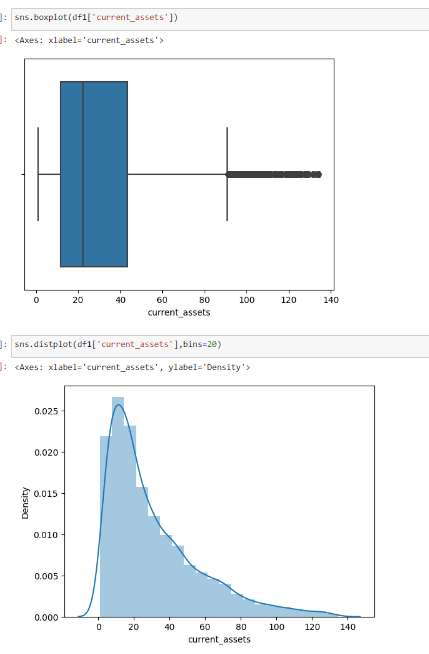


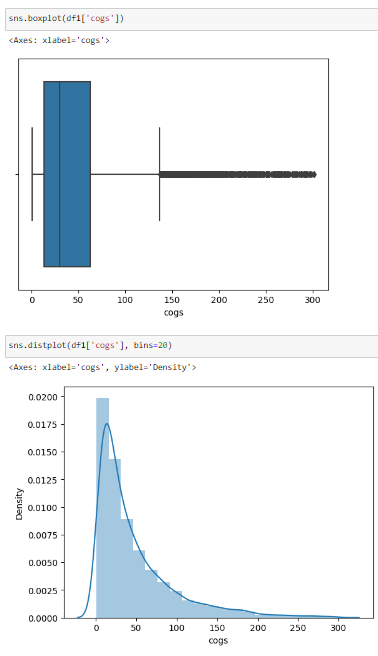
In the provided code, the objective is to create a new DataFrame df1 by filtering out the data points in the original DataFrame df based on specific percentiles defined for various financial indicators. The percentiles dictionary contains tuples specifying lower and upper percentiles for each financial indicator. The lower and upper percentiles represent the range of values that will be retained in the new DataFrame.

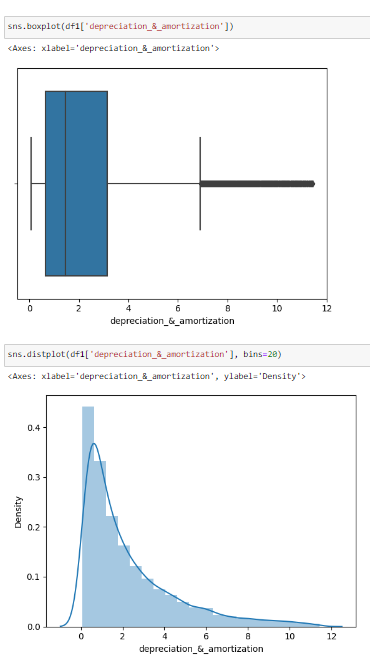
The code iterates through each column in the percentiles dictionary, and for each financial indicator, it calculates the lower and upper bounds based on the specified percentiles using the quantile() function. Subsequently, it filters df1 by selecting only the rows where the value of each financial indicator falls within the specified percentile range.

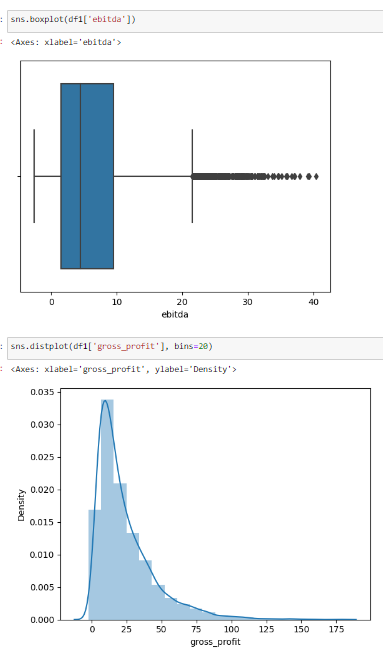
The new DataFrame df1 is created, which contains a subset of the original data, retaining only those rows where the financial indicators meet the defined percentile criteria. By resetting the index with reset\_index(drop=True, inplace=True), the row indices are reorganized sequentially, resulting in a more consistent and organized DataFrame. Finally, the code compares the shapes of the original DataFrame df and the filtered DataFrame df1 which is reduced by about 5 times, providing a quick summary of the data reduction achieved through filtering based on the specified percentiles.

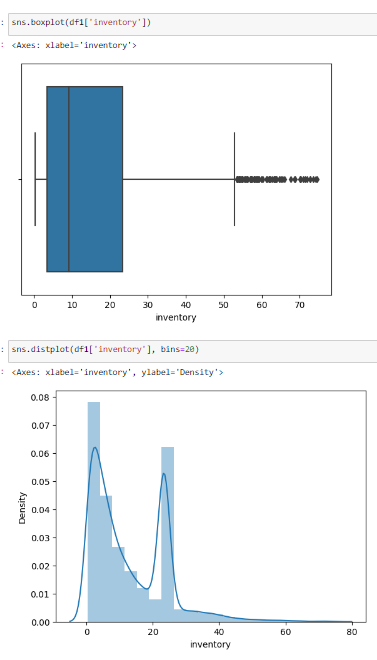
### **2.3.3 Post cleaning univariate analysis**

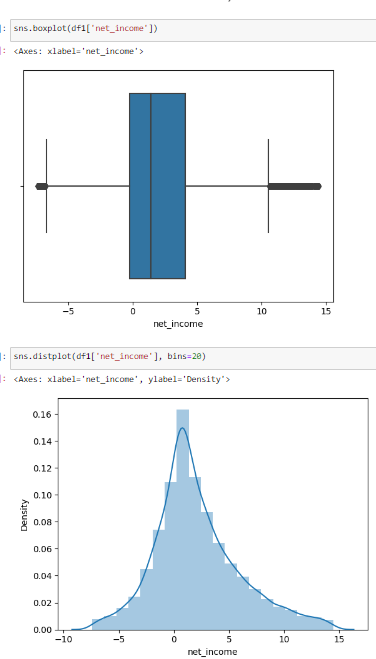


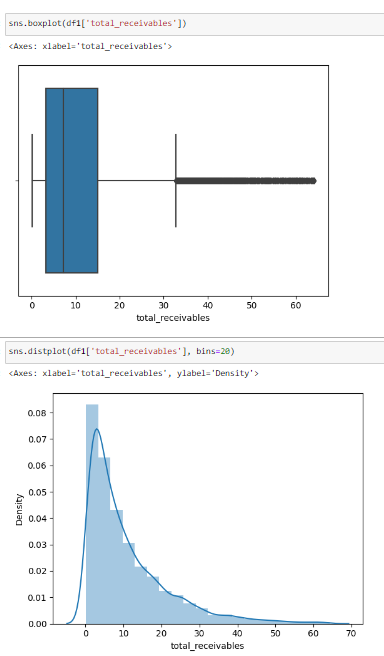


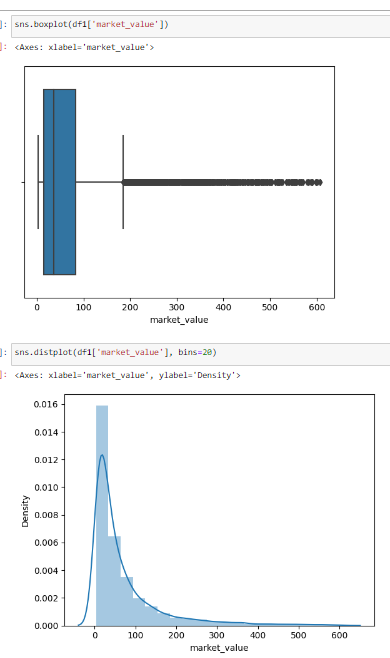


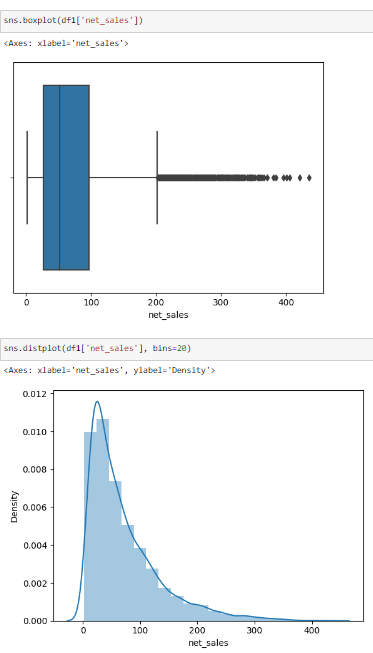


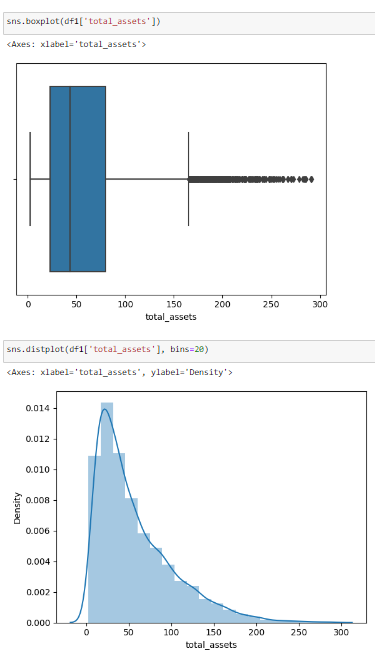


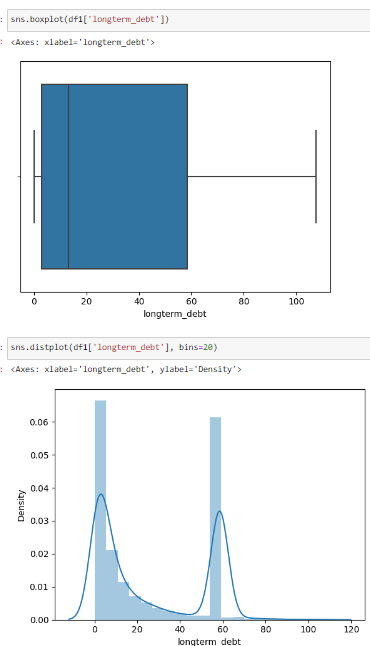




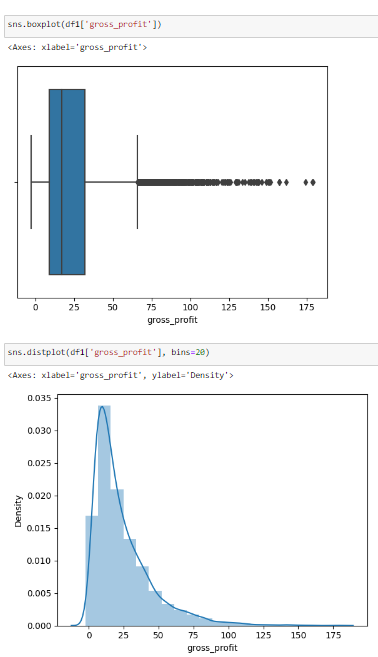


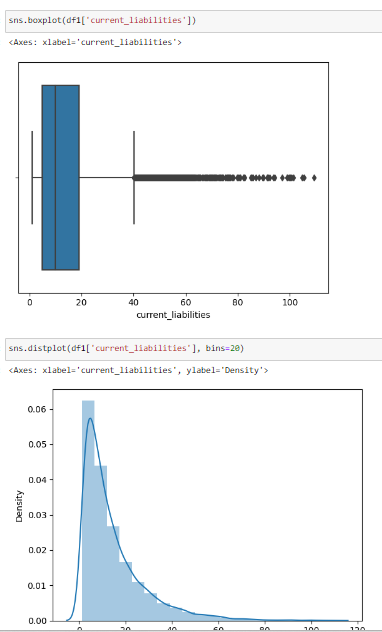


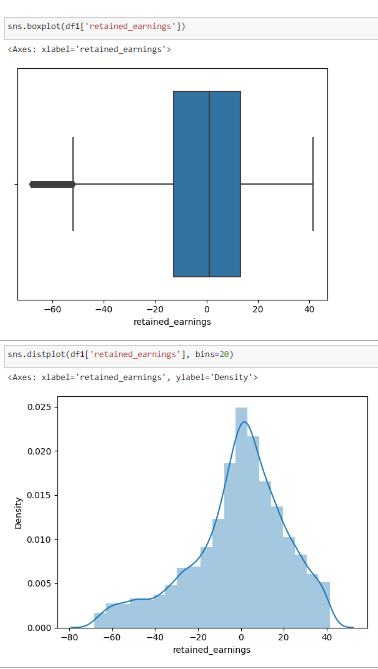


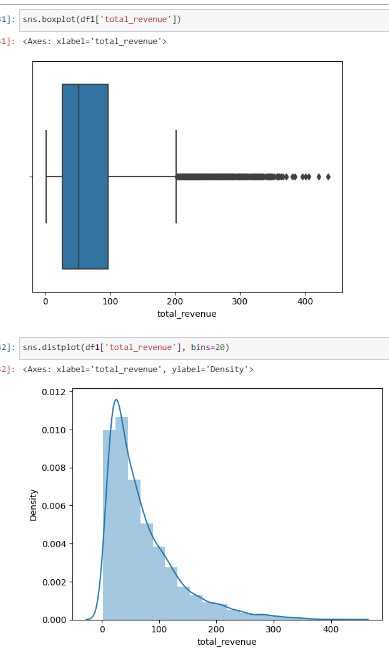


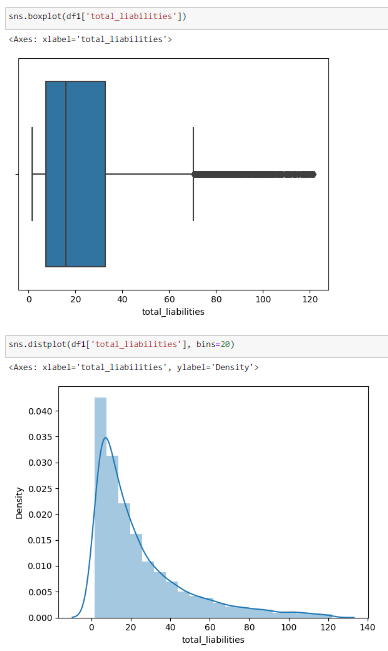


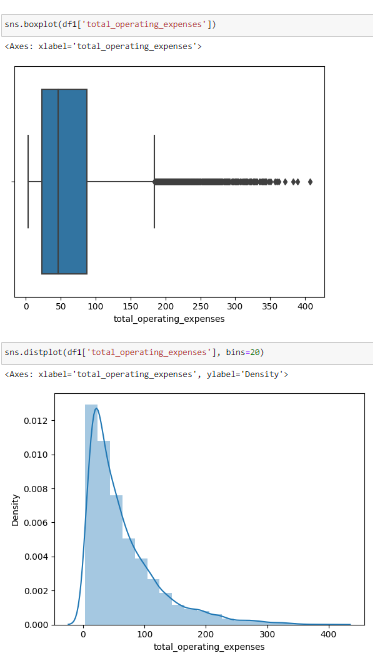










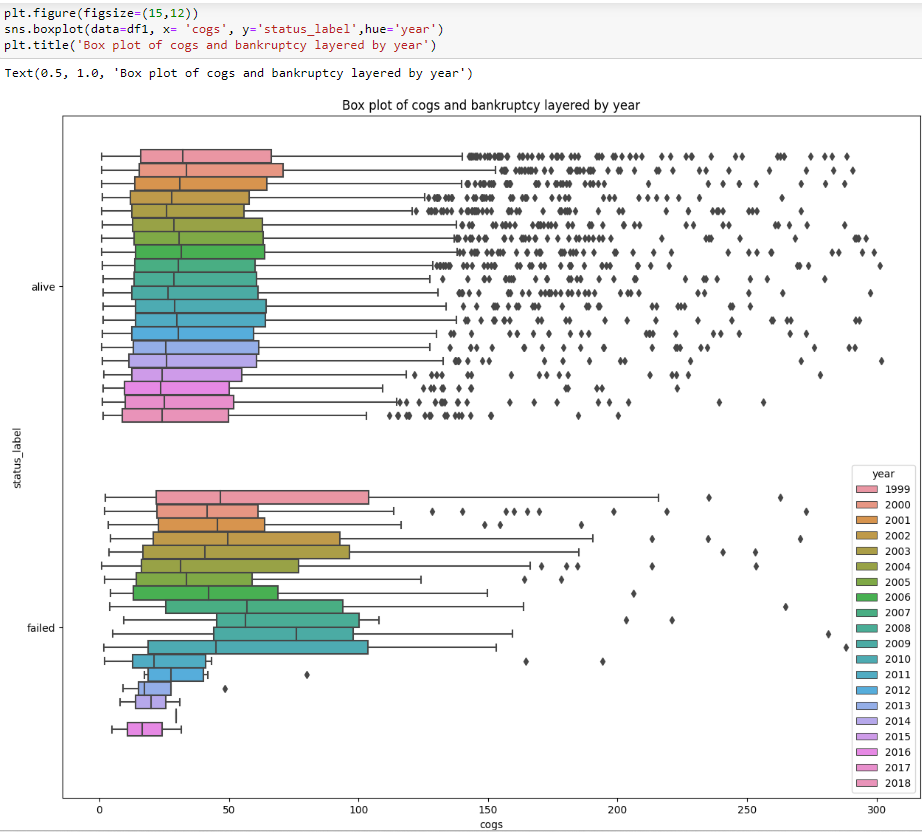
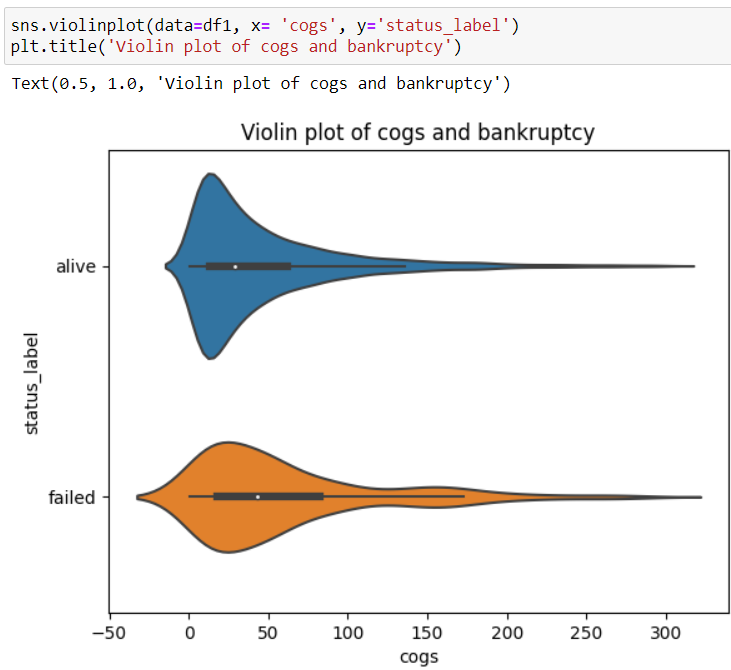


The boxplots and distribution plots after the removal of outliers are depicting a normal distribution lookalike distribution for the duration variable. Although not perfectly normally distributed, this seems to be way more standardized than Pre outlier removal’s distribution which gives us a signal that we can move ahead with our further modeling.

### **2.3.4 Bivariate and multivariate analysis**

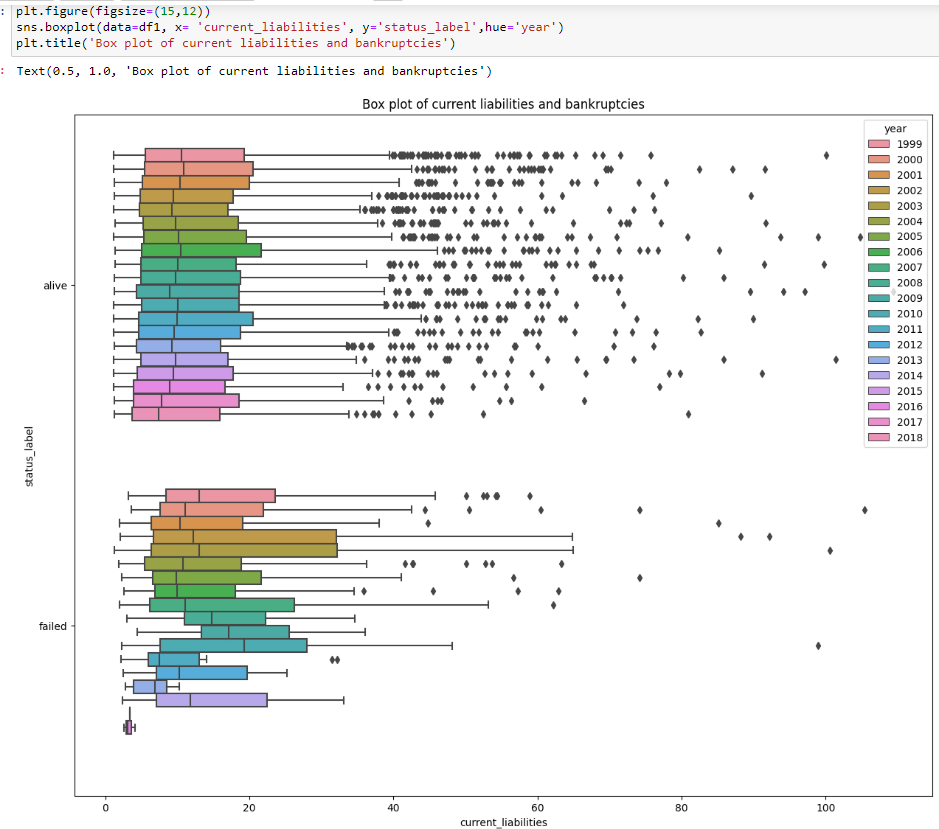
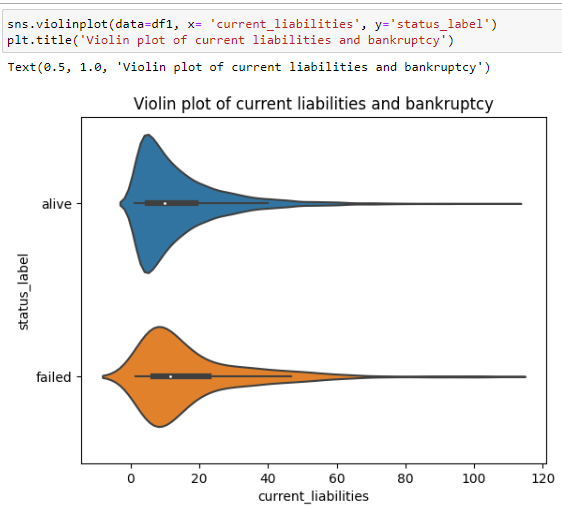
Here, we will be analyzing more than two variables unlike the univariate analysis. Also, we will be relating these analyses with the **ASSUMPTIONS** that we have made while formulating the hypothesis.

Relationship between cogs and bankruptcy



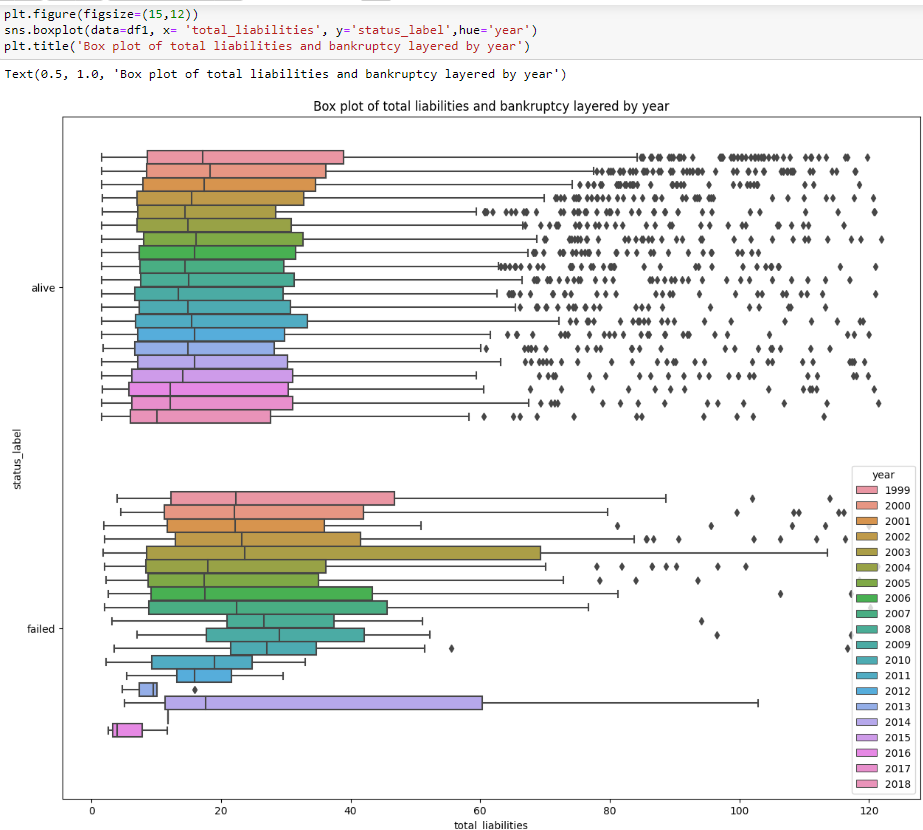
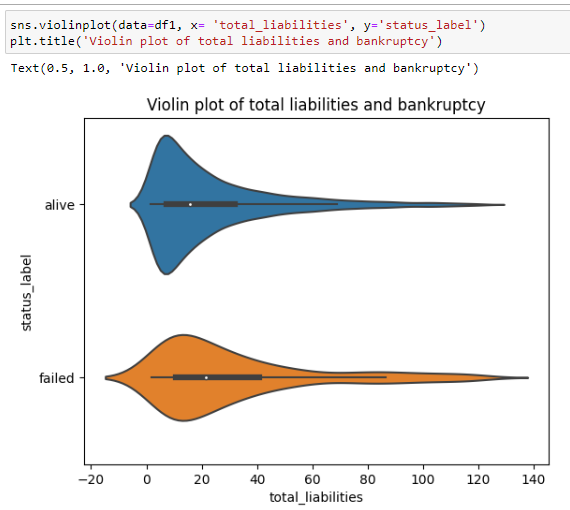
Here, The boxplot and the violin plot show that the median COGS for the bankrupt companies is higher than the median COGS for the companies that did not go bankrupt. When layered by the year, we can see that in later years, the bankrupt companies have incurred more cost of goods sold than the non bankrupt companies.

**Relationship between current liabilities and Bankruptcy**



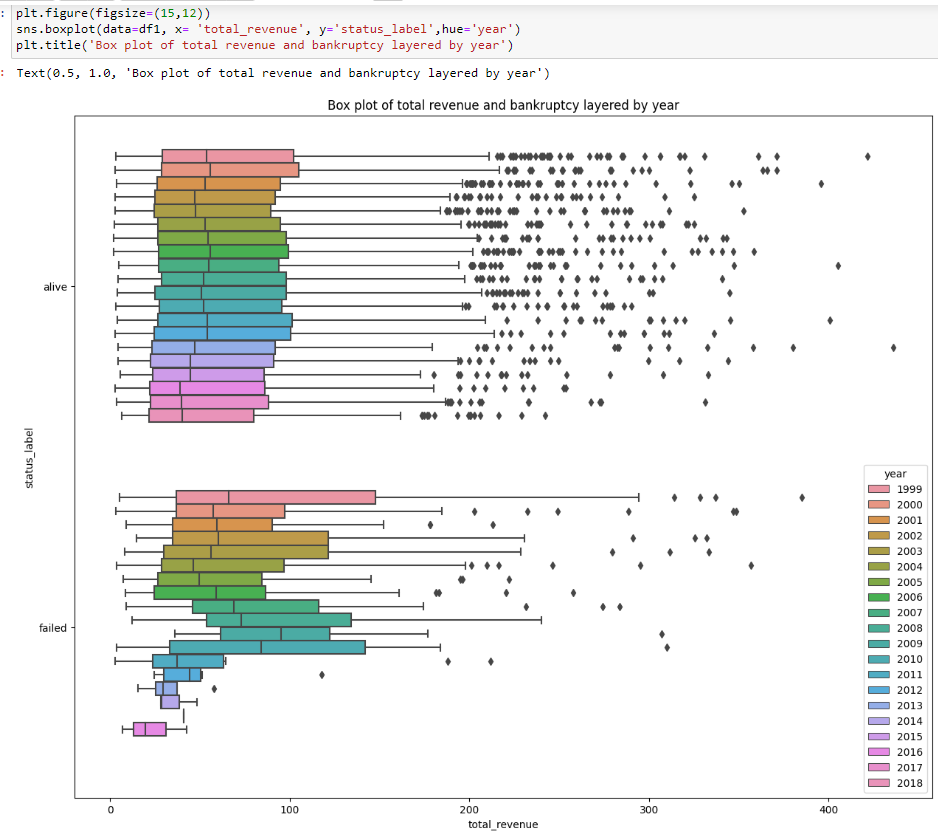
Similarly, we can see that the bankrupt companies have a higher median current liabilities than that of the non bankrupt companies especially in the latter years. The bankrupt companies have a higher median current liabilities than non-bankrupt companies which implies that the level of current liabilities may be a significant factor contributing to their financial distress and eventual bankruptcy. Current liabilities are the debts and obligations that a company needs to settle within a short period, usually within one year. These liabilities often include accounts payable, short-term loans, and other obligations that require relatively immediate payment.

**Relationship between total liabilities and Bankruptcy**



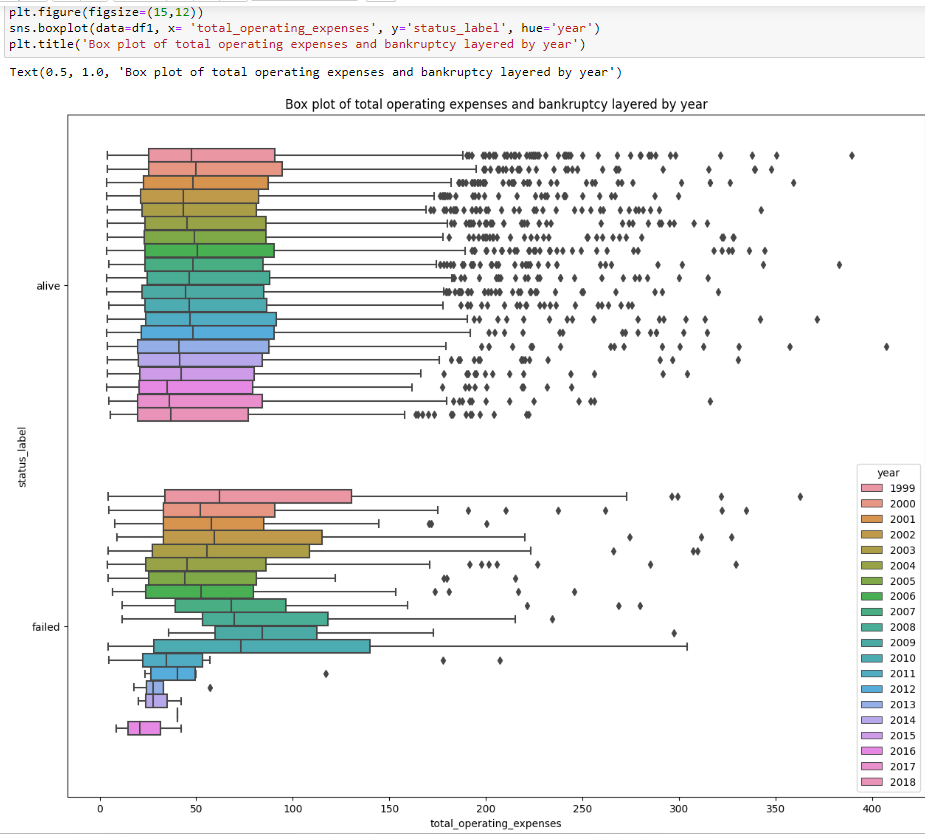
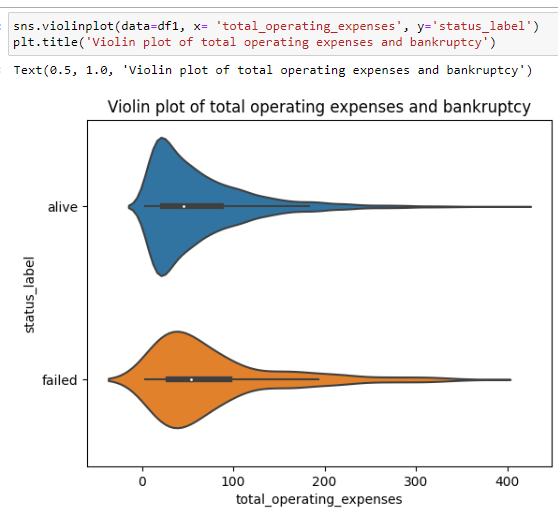
We can observe a higher total liabilities in the bankrupt companies than that of the non bankrupt companies especially in the latter years when layered with years. A higher total liabilities in bankrupt companies compared to non-bankrupt companies implies that the former may have accumulated excessive debt and financial obligations, potentially beyond their capacity to manage and repay. This could indicate poor financial management, a high dependency on debt financing, and the inability to generate sufficient income to cover their liabilities. The elevated level of total liabilities can lead to financial instability, liquidity problems, and an increased risk of default, ultimately contributing to the company's financial distress and bankruptcy.

**Relationship between total revenue and Bankruptcy**



A higher total revenue in bankrupt companies compared to non-bankrupt companies would be quite unusual and contradictory to typical bankruptcy scenarios. In a healthy business environment, successful companies usually generate higher revenues, indicating strong customer demand, effective business strategies, and positive financial performance. On the other hand, bankrupt companies typically face declining revenues or struggle to generate enough revenue to cover their expenses and debt obligations. If such a situation were observed, it might indicate an aggressive sales strategy or liberal credit terms of the bankrupt companies.

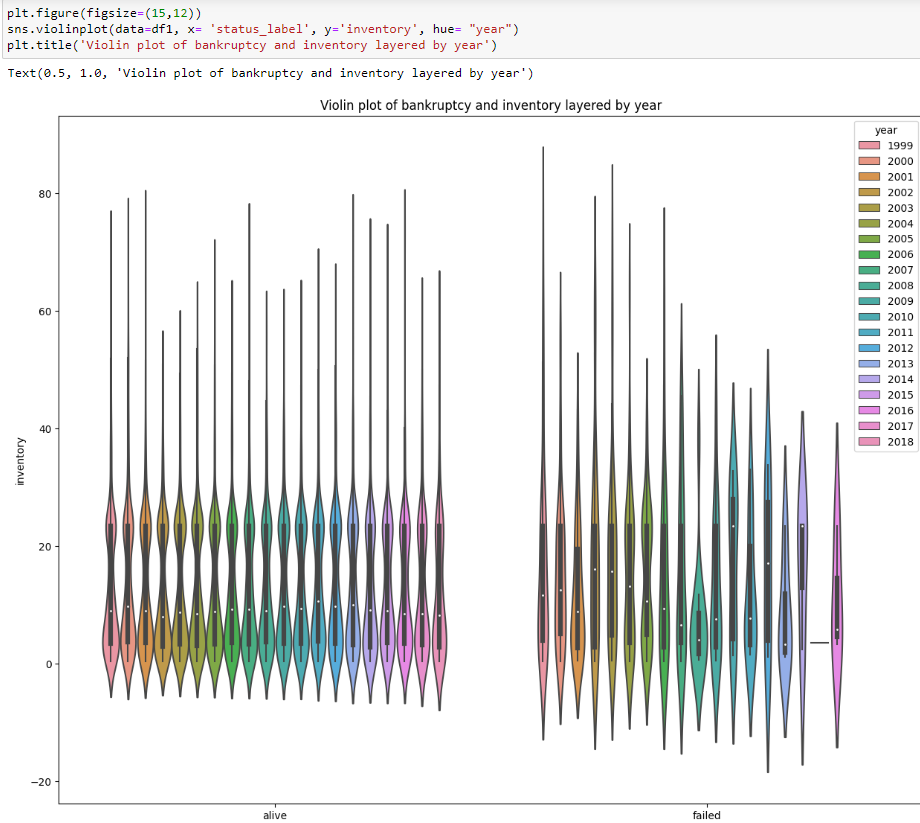
**Relationship between total operating expenses and bankruptcy**



A higher total operating expenses in bankrupt companies compared to non-bankrupt companies would be a concerning sign. Operating expenses encompass the day-to-day costs of running a business, including salaries, rent, utilities, and other expenses necessary to maintain operations. In a healthy business, operating expenses are managed efficiently to ensure profitability and sustainability.

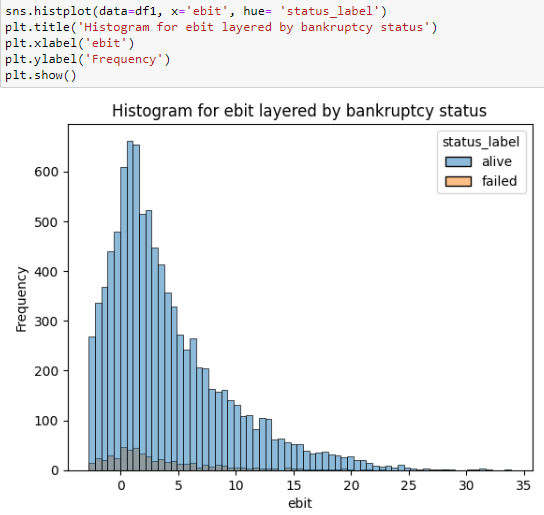
It implies that they might have been facing challenges in controlling their costs or optimizing their operations. This situation can lead to reduced profitability, cash flow problems, and financial distress. High operating expenses relative to revenue can erode profit margins and make it difficult for a company to generate enough income to cover its obligations, ultimately contributing to its financial downfall and potential bankruptcy.

**Relationship between bankruptcy and Inventory**



A higher inventory level in bankrupt companies compared to non-bankrupt companies suggests potential issues in managing inventory effectively. This could be due to poor inventory management, decreased sales, liquidity problems, challenges in adapting to market demands, or industry changes. Excessive inventory ties up capital, impacts cash flow, and can lead to financial strain, ultimately contributing to the company's financial distress and potential bankruptcy.

**Relationship between bankruptcy and EBIT**



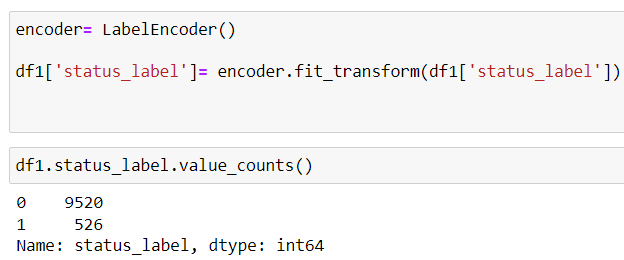
An extremely low EBIT in bankrupt companies implies significant profitability issues and financial distress. It indicates that the company is struggling to generate enough operating profit to cover expenses and debt obligations, potentially due to declining sales or inefficient operations. This low EBIT can hinder the company's ability to sustain itself, making it vulnerable to insolvency and eventual bankruptcy.

# Chapter 3 Feature Engineering

## 3.1 Feature engineering

Feature engineering is the process of transforming raw data into a more meaningful format for machine learning models. It involves techniques such as feature extraction, where relevant information is selected or derived from the data, feature transformation, which applies mathematical transformations to ensure data compatibility, and feature encoding, which converts categorical variables into numerical representations. Under feature engineering, we will be performing the following exercises.

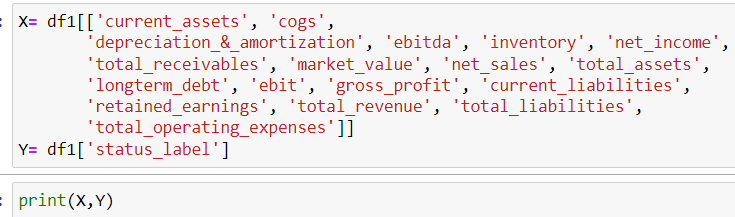
### **3.1.1 Variable encoding**



Here, a LabelEncoder object encoder is instantiated, which will be used to encode the categorical 'status\_label' column in the DataFrame df1 into numerical values. LabelEncoder is a utility class in scikit-learn that converts categorical labels into unique integer values. The categorical column 'status\_label' is transformed using the fit\_transform() method of the encoder. This process maps each unique category in the 'status\_label' column to a corresponding numerical label, ensuring that each label is encoded with a unique integer value.

The resulting numerical labels represent different categories present in the 'status\_label' column. The value\_counts() function is then applied to count the occurrences of each unique numerical label in the encoded 'status\_label' column. This provides a summary of the distribution of each category in the form of its corresponding encoded label.

### **3.1.2 Segregating features and target variables**



Here, the DataFrame df1 is being prepared for a machine learning task. The variables X and Y are created to represent the features (X) and the target variable (Y) for the task. The features X include a subset of columns from the original DataFrame df1, containing financial indicators such as 'current\_assets', 'cogs', 'depreciation\_&\_amortization', 'ebitda', 'inventory', 'net\_income', 'total\_receivables', 'market\_value', 'net\_sales', 'total\_assets', 'longterm\_debt', 'ebit', 'gross\_profit', 'current\_liabilities', 'retained\_earnings', 'total\_revenue', 'total\_liabilities', and 'total\_operating\_expenses'. These financial indicators will be used as inputs for the machine learning model. The target variable Y is defined as 'status\_label', which has been previously encoded using the LabelEncoder to convert categorical values into numerical labels. The target variable represents the status category or class that the machine learning model will attempt to predict.

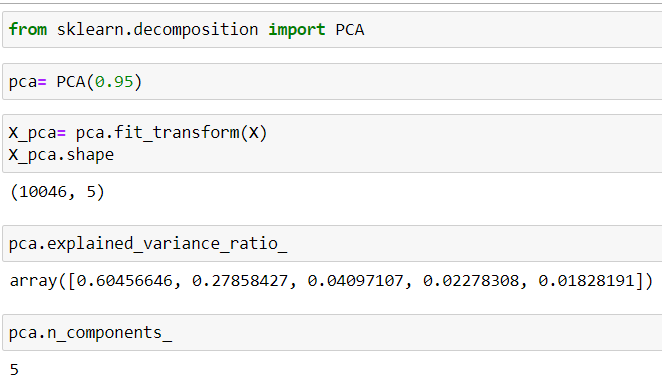
By printing X and Y, the code provides a concise display of the selected features and the target variable, offering a glimpse of the data that will be used for the subsequent machine learning task. This data preparation step is crucial for organizing the data into appropriate input and output components required for training and evaluating the machine learning model.

### **3.1.3 Feature selection**

Feature selection in data science refers to the process of identifying and selecting the most relevant and informative features or variables from a dataset that contribute the most towards predicting or explaining the target variable. By eliminating irrelevant or redundant features, feature selection reduces the dimensionality of the data, improves model performance, and enhances interpretability.

We have analyzed the features by using two main methods of feature selection. The two methods of feature selection are explained below:

**Principal Component Analysis**



Principal Component Analysis (PCA) is a dimensionality reduction technique used to transform high-dimensional data into a lower-dimensional space by identifying a set of orthogonal axes called principal components. These components capture the maximum variance in the data, enabling data scientists to represent the data with fewer dimensions while retaining the most important information and patterns.

Here, Principal Component Analysis (PCA) is utilized to reduce the dimensionality of the feature set X, which contains various financial indicators. By creating a PCA instance with a variance threshold of 0.95, the code ensures that the transformed data, represented by X\_pca, retains 95% of the variance present in the original features. The transformed X\_pca will have fewer dimensions while preserving the most important patterns in the data. The shape of X\_pca is displayed to observe the new dimensionality. The pca.explained\_variance\_ratio\_ provides an array of explained variance ratios for each principal component, indicating the proportion of variance explained by each component. The cumulative sum of these ratios should be close to 0.95, validating that the desired variance retention is achieved.

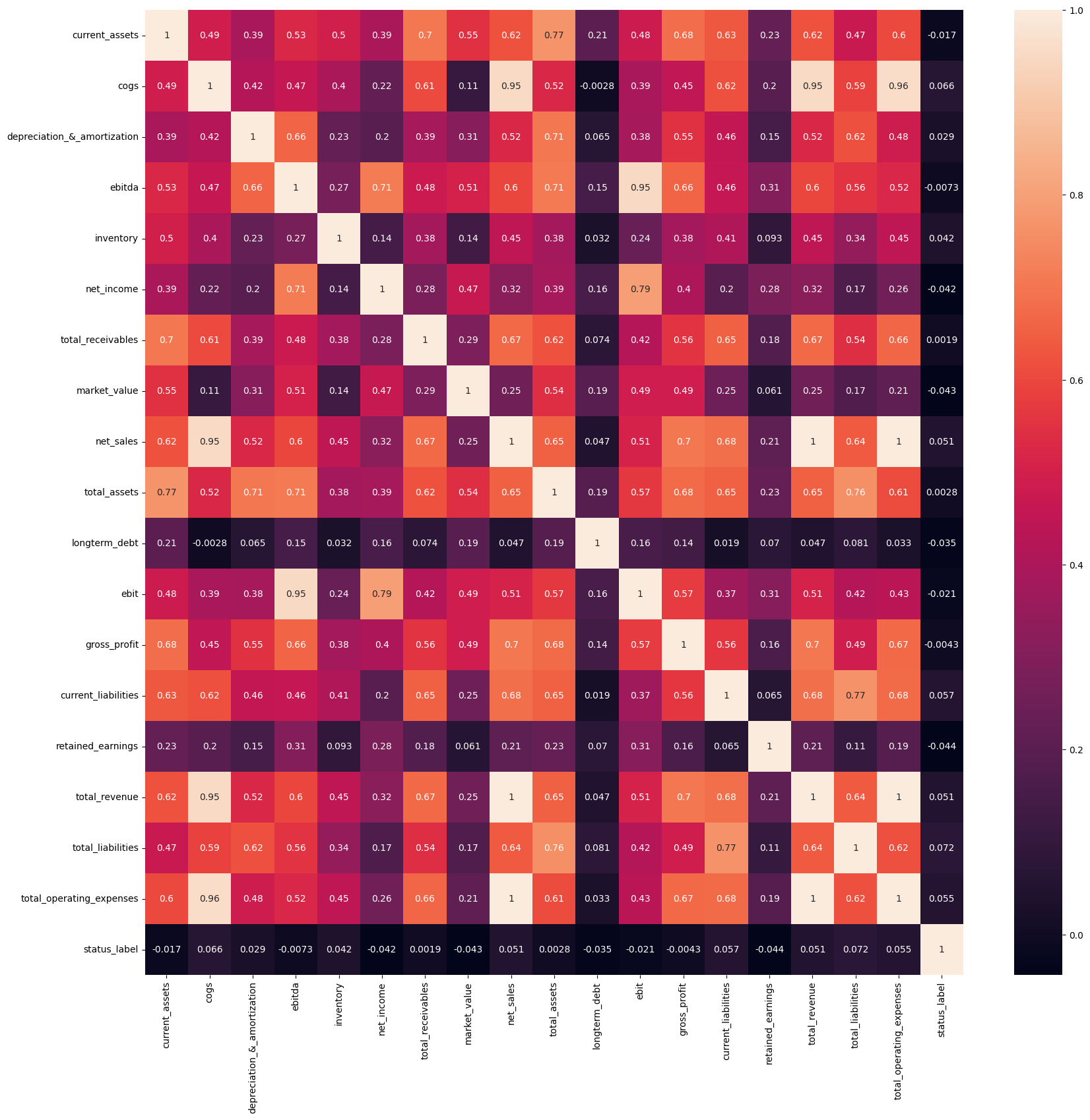
Finally, pca.n\_components\_ returns the number of principal components selected i**.e 5**, revealing the reduced dimensionality of the data after PCA transformation. Overall, PCA effectively reduces the feature set X while retaining a significant portion of its variance, enabling more efficient and insightful analysis in the reduced-dimensional space.

**Correlation method of feature selection**

The correlation method of feature selection involves selecting features that exhibit the highest correlation with the target variable. This technique measures the strength and direction of the linear relationship between each feature and the target, typically using correlation coefficients such as Pearson's correlation.

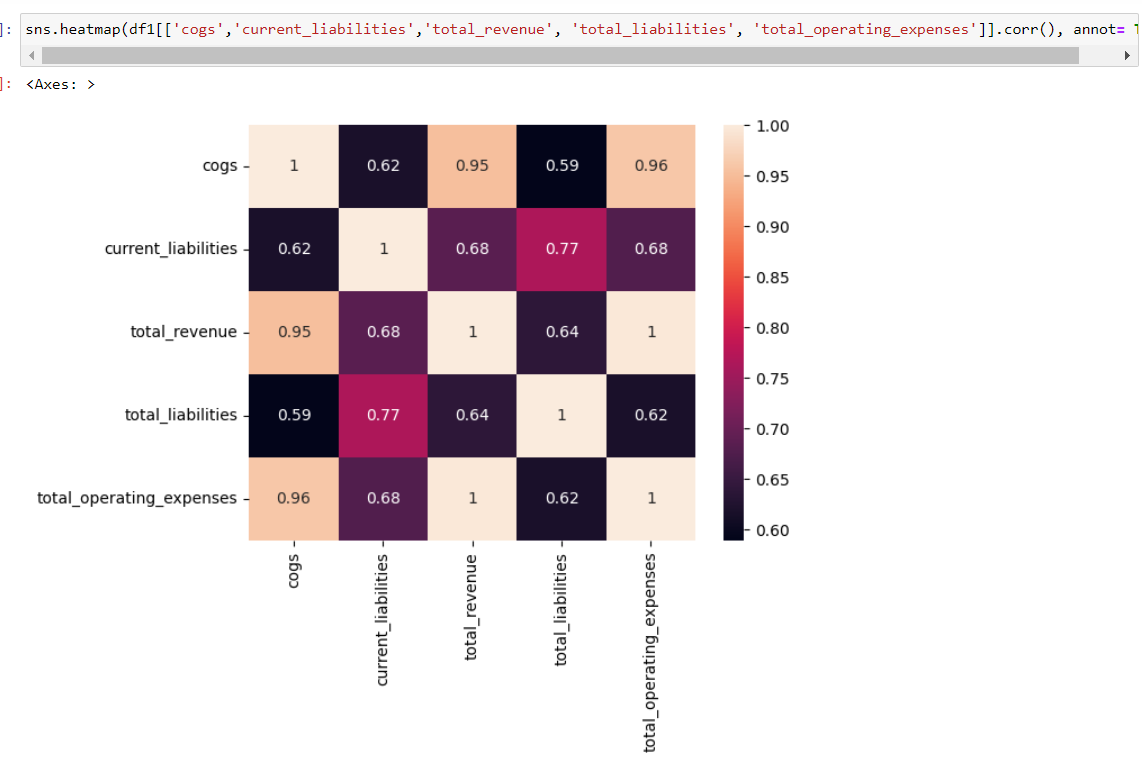
Combining the PCA with correlation method we will select 5 most correlated features in the following ways.





We will take cogs, current\_liabilities, total\_revenue, total\_liabilities and total\_operating\_expenses as our features since we have been provided with the optimal number of features to be 5 by the PCA analysis, they are the 5 most correlated to the target variable/ variable of interest.

### **3.1.4 Checking for multicollinearity between the features**

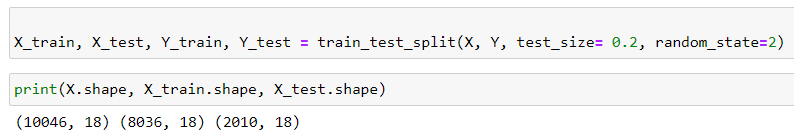


Here, we can observe significant correlation between many variables. If we observe a deviation in the test train score later, we will again use the dimensionality reduction technique to rip off some features.

# Chapter 4

# Preparation for Machine Learning modeling

## 4.1 Splitting the features and target into test and train data

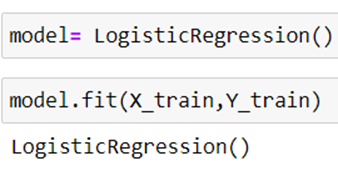
****

Here, the dataset is split into training and testing sets using the train\_test\_split function from scikit-learn. X\_train and Y\_train represent the feature and target variables, respectively, for the training set, while X\_test and Y\_test represent the feature and target variables for the testing set. The train\_test\_split function divides the original feature set X and target variable Y into two sets: one for training the machine learning model (X\_train and Y\_train) and the other for evaluating the model's performance (X\_test and Y\_test). The test\_size parameter is set to 0.2, which means that 20% of the data is reserved for testing, while 80% is used for training. The random\_state is set to 2, ensuring reproducibility of the same train-test split if the code is run multiple times. By printing X.shape, X\_train.shape, and X\_test.shape, the code provides a summary of the dimensions of the original feature set and the training and testing sets, enabling a quick assessment of the data partitioning.

# Chapter 5 Predictive analysis using machine learning models

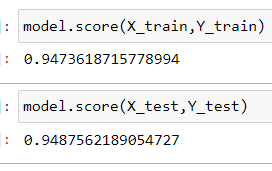
## 5.1 “Model”: Logistic regression

Logistic regression is a machine learning model used for binary classification, where it predicts the probability of an instance belonging to a particular class. It applies a sigmoid function to the linear combination of input features, transforming the output into a probability value. By applying a threshold, instances are classified into one of the classes. The model is trained by estimating optimal coefficients that minimize a specific loss function.



Here, a logistic regression model is instantiated using LogisticRegression() and assigned to the model. The fit() method is then applied to train the model on the training data (X\_train and Y\_train). This process estimates the optimal coefficients that minimize the difference between predicted and actual target values. By fitting the model to the training data, it learns the underlying patterns and relationships.

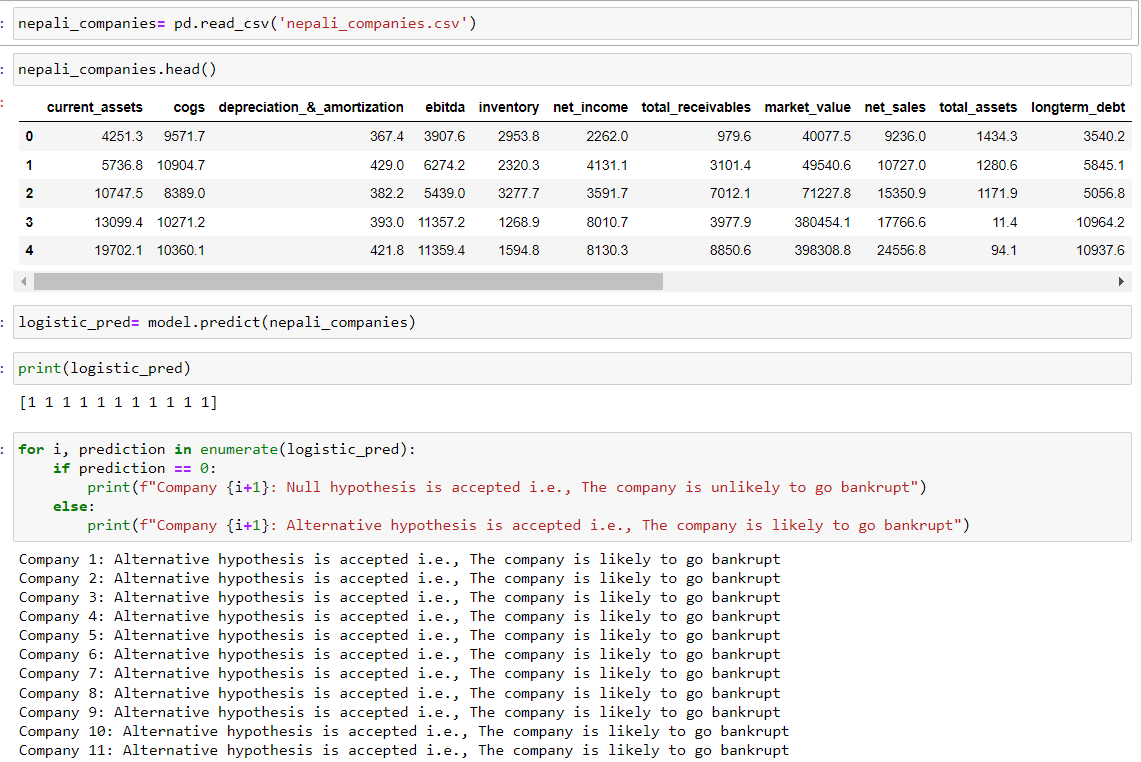
### **5.1.1 Accuracy check on train and test data**



model.score(X\_train, Y\_train) : 0.9473618715778994: This indicates that the model achieved an accuracy of approximately 94.74% on the training dataset (X\_train and Y\_train). The accuracy represents the proportion of correctly predicted target values (Y\_train) with respect to the corresponding feature values (X\_train) used during training.

model.score(X\_test, Y\_test) : 0.9487562189054727: This shows that the model achieved an accuracy of around 94.88% on the testing dataset (X\_test and Y\_test). The accuracy represents the proportion of correctly predicted target values (Y\_test) with respect to the corresponding feature values (X\_test) that were not used during training, serving as an indicator of the model's generalization performance on unseen data.

### **5.1.2 Actual prediction**



Here, a dataset containing information about Nepali companies is loaded using the Pandas library and stored in the DataFrame nepali\_companies. The dataset's initial rows are displayed using the head() method to provide a glimpse of the data. Subsequently, a logistic regression model is used to predict bankruptcy probabilities for each company in the nepali\_companies DataFrame. The predicted outcomes are stored in the array logistic\_pred.

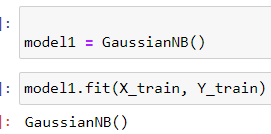
To interpret the predictions, a loop is used to iterate through each prediction in logistic\_pred. For each company (indexed by i+1), the loop checks the prediction value. If the prediction is 0, the code prints a message indicating that the null hypothesis is accepted, meaning the company is unlikely to go bankrupt. On the other hand, if the prediction is not 0 (i.e., 1), the code prints a message indicating that the alternative hypothesis is accepted, suggesting the company is likely to go bankrupt. The logistic regression model is used to classify companies into two classes: 0 (non-bankrupt) and 1 (bankrupt), based on the input features from the nepali\_companies dataset. This interpretation provides valuable insights for stakeholders, such as investors and financial institutions, to make informed decisions about the financial health and viability of the Nepali companies in question.

When we asked the model to predict the bankruptcy for a sample nepali company with details on the selected features that have been provided on the CSV dataframe, the model predicted that the Alternative hypothesis is accepted i.e., all the companies are likely to go bankrupt.

## 5.2 “Model 1”: Naïve Bayes classification algorithm

Bayes' theorem is a fundamental concept in probability theory and statistics. It provides a way to update the probability of an event based on new evidence or information.

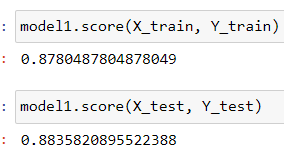
P(A|B) = (P(B|A) \* P(A)) / P(B)



Here, a Gaussian Naive Bayes model is created and assigned to the variable model1. This model is then trained using the training data (X\_train and Y\_train) by calling the fit() method on model1.

The fit() method calculates the necessary statistical parameters from the training data to build the Naive Bayes model. In the case of Gaussian Naive Bayes, it estimates the mean and variance for each feature in each class based on the provided training instances. By executing this code, the Gaussian Naive Bayes model (model1) learns the underlying patterns and relationships in the training data, enabling it to make predictions based on the input features.

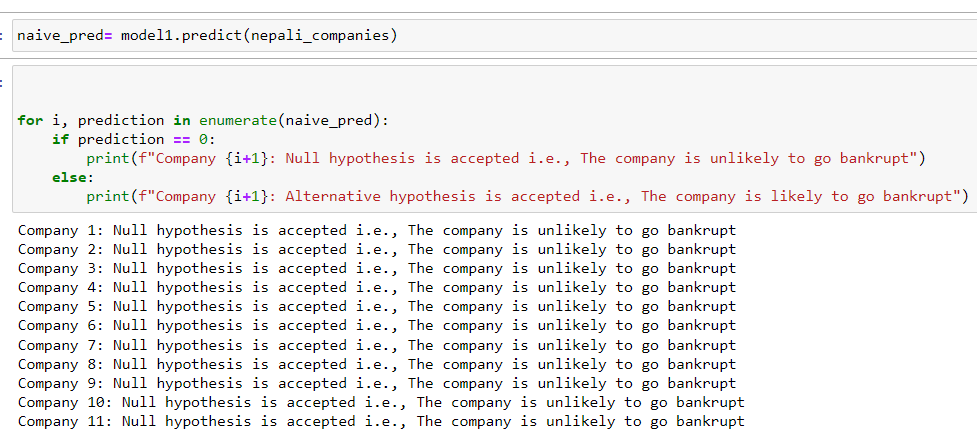
### **5.2.1 Accuracy check on train and test data**



model1.score(X\_train, Y\_train) 0.8780487804878049: This indicates that model1 achieved an accuracy of approximately 87.80% on the training dataset (X\_train and Y\_train). The accuracy represents the proportion of correctly predicted target values (Y\_train) with respect to the corresponding feature values (X\_train) used during training.

model1.score(X\_test, Y\_test) 0.8835820895522388: This shows that model1 achieved an accuracy of around 88.36% on the testing dataset (X\_test and Y\_test). The accuracy represents the proportion of correctly predicted target values (Y\_test) with respect to the corresponding feature values (X\_test) that were not used during training. This indicates how well model1 is able to generalize its predictions to new, unseen data

### **5.2.2 Actual prediction**

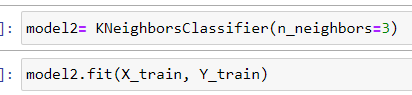


In the provided code, a machine learning model named model1 is used to predict bankruptcy probabilities for Nepali companies in the nepali\_companies dataset. The model's predictions are stored in the array naive\_pred. Subsequently, a loop is employed to iterate through each prediction in naive\_pred. For each company (indexed by i+1), the loop checks the prediction value. If the prediction is 0, the code prints a message indicating that the null hypothesis is accepted, meaning the company is unlikely to go bankrupt. Otherwise, if the prediction is not 0 (i.e., 1), the code prints a message indicating that the alternative hypothesis is accepted, suggesting the company is likely to go bankrupt. The model's predictions provide valuable insights into the likelihood of bankruptcy for each Nepali company, allowing stakeholders to make informed decisions about their financial health and viability.

When we asked the model to predict the bankruptcy for a sample nepali company with details on the selected features that have been provided on the CSV dataframe, the model predicted that the Null hypothesis is accepted i.e., all the companies are unlikely to go bankrupt.

## 5.3 “Model2”: K Nearest neighbor

K-Nearest Neighbors (KNN) is a machine learning algorithm that predicts the class or value of a data point based on the classes or values of its neighboring data points. It uses a distance metric to find the K nearest neighbors in the training data and assigns the majority class or average value of those neighbors as the prediction for the new data point. KNN is a versatile and intuitive algorithm, applicable to both classification and regression tasks, and is useful when decision boundaries are not well-defined or when there is limited training data.



Here, a K-Nearest Neighbors (KNN) model is created and assigned to the variable model2. The model is initialized with n\_neighbors=3, which sets the number of neighbors considered for prediction to 3. The KNN model is then trained using the training data (X\_train and Y\_train) by calling the fit() method on model2. During the training process, the model learns the relationships between the input features in X\_train and their corresponding target labels in Y\_train, enabling it to make predictions based on the nearest neighbors in the feature space.

### 

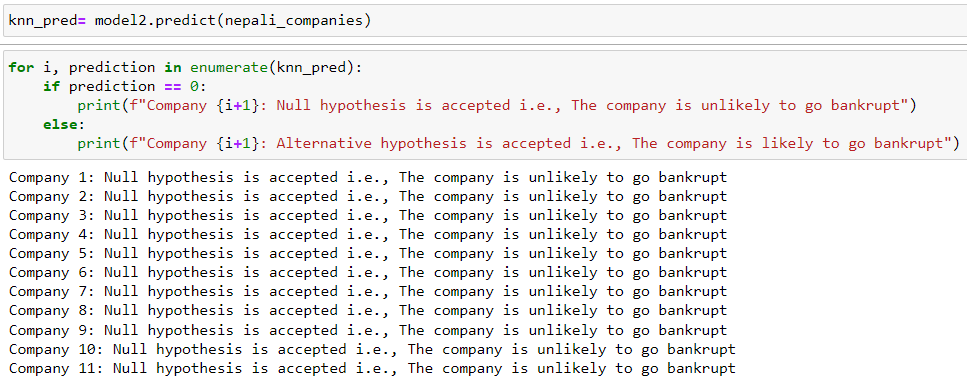
### **5.3.1 Accuracy check on train and test data**



model2.score(X\_train, Y\_train) 0.9695121951219512: This indicates that model2 achieved an accuracy of approximately 96.95% on the training dataset (X\_train and Y\_train). The accuracy represents the proportion of correctly predicted target values (Y\_train) with respect to the corresponding feature values (X\_train) used during training.

model2.score(X\_test, Y\_test) 0.9482587064676616: This shows that model2 achieved an accuracy of around 94.83% on the testing dataset (X\_test and Y\_test). The accuracy represents the proportion of correctly predicted target values (Y\_test) with respect to the corresponding feature values (X\_test) that were not used during training. This indicates how well model2 is able to generalize its predictions to new, unseen data.

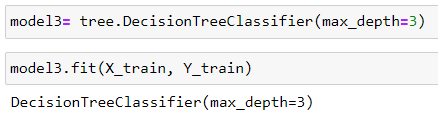
### **5.3.2 Actual prediction**



In the provided code, a k-Nearest Neighbors (KNN) machine learning model model2 is used to predict bankruptcy probabilities for Nepali companies in the nepali\_companies dataset. The model's predictions are stored in the array knn\_pred. A loop is then employed to iterate through each prediction in knn\_pred. For each company (indexed by i+1), the loop checks the prediction value. As observed from the output, all predictions are 0, which means the null hypothesis is accepted for each company. The null hypothesis states that the company is unlikely to go bankrupt. This implies that according to the KNN model's predictions, none of the companies in the nepali\_companies dataset are predicted to go bankrupt. This interpretation provides important insights for stakeholders, suggesting that, based on the model's analysis, the assessed Nepali companies are less likely to face bankruptcy.

## 5.4 "Model 3": Decision Tree Algorithm

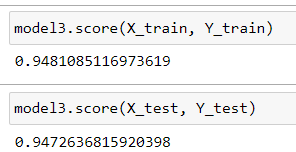
A decision tree algorithm is a machine learning algorithm that builds a predictive model in the form of a tree structure. It is a popular algorithm for both classification and regression tasks. The decision tree algorithm partitions the input data into subsets based on various attributes and makes decisions at each node of the tree based on these attributes. Each internal node represents a test on an attribute, each branch represents the outcome of the test, and each leaf node represents the final decision or the predicted outcome.



Here, the variable model3 is assigned to an instance of the DecisionTreeClassifier class from the tree module. This class is part of the scikit-learn library, which is a popular machine learning library in Python. The DecisionTreeClassifier is a specific implementation of the decision tree algorithm for classification tasks. It creates a decision tree model that can be used to classify new instances based on their features.

The parameter max\_depth=3 is passed to the DecisionTreeClassifier constructor. This parameter sets the maximum depth of the decision tree. The depth of a tree refers to the length of the longest path from the root node to any leaf node. By setting max\_depth=3, we are limiting the depth of the tree to 3 levels. This means that the decision tree will have a maximum of 3 decision nodes, which helps to control the complexity and overfitting of the model.

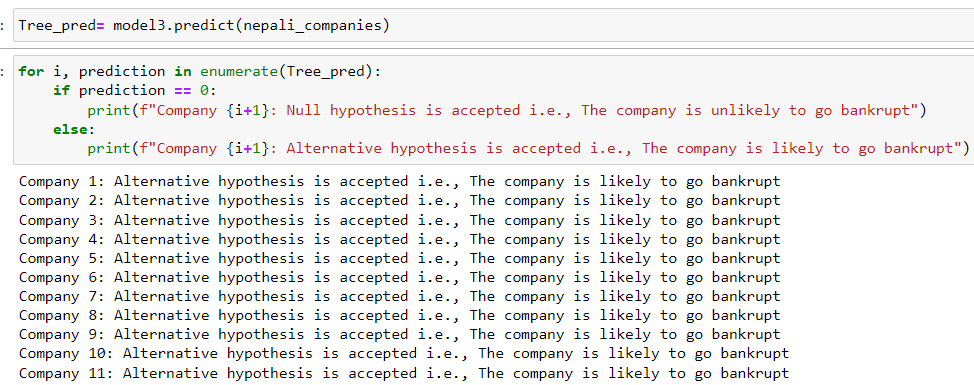
### **5.4.1 Accuracy check on train and test data**



model3.score(X\_train, Y\_train) 0.9481085116973619: This indicates that model3 achieved an accuracy of approximately 94.81% on the training dataset (X\_train and Y\_train). The accuracy represents the proportion of correctly predicted target values (Y\_train) with respect to the corresponding feature values (X\_train) used during training.

model3.score(X\_test, Y\_test) 0.9472636815920398: This shows that model3 achieved an accuracy of around 94.73% on the testing dataset (X\_test and Y\_test). The accuracy represents the proportion of correctly predicted target values (Y\_test) with respect to the corresponding feature values (X\_test) that were not used during training. This indicates how well model3 is able to generalize its predictions to new, unseen data.

### **5.4.2 Actual prediction**



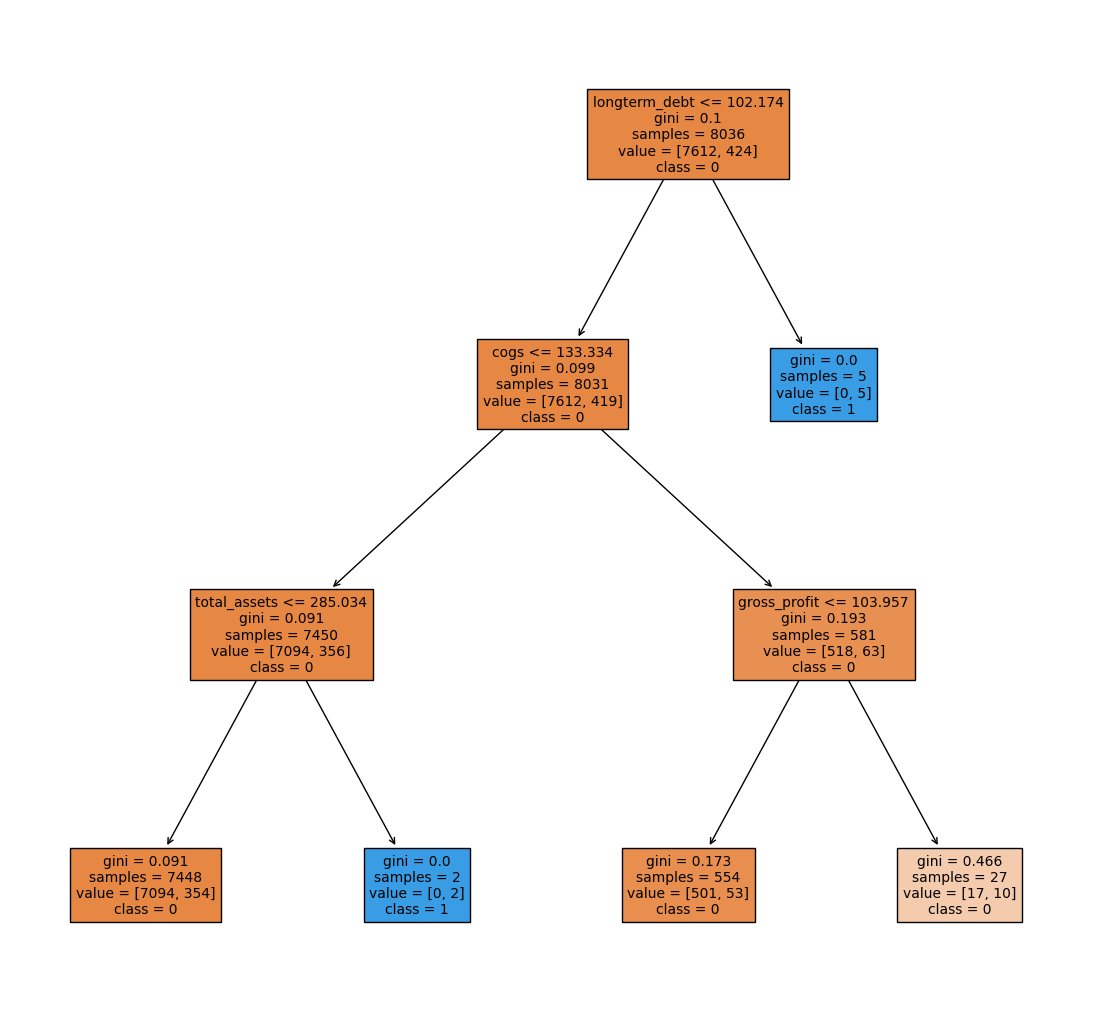
In the provided code, a decision tree machine learning model model3 is used to predict bankruptcy probabilities for Nepali companies in the nepali\_companies dataset. The model's predictions are stored in the array Tree\_pred. A loop is then employed to iterate through each prediction in Tree\_pred. For each company (indexed by i+1), the loop checks the prediction value. As observed from the output, all predictions are 1, which means the alternative hypothesis is accepted for each company.

The alternative hypothesis states that the company is likely to go bankrupt. This implies that, according to the decision tree model's predictions, all the companies in the nepali\_companies dataset are predicted to go bankrupt. This interpretation provides significant insights for stakeholders, suggesting that, based on the model's analysis, the assessed Nepali companies are more likely to face bankruptcy.

### **5.4.3 Visualization of decision tree**



The above code imports the plot\_tree function from the tree module of the scikit-learn library. This function allows us to visualize and plot a decision tree model. By importing this function, we gain access to a useful tool that helps in understanding the structure and decision-making process of a decision tree by providing a graphical representation of the tree's nodes, branches, and leaf nodes.

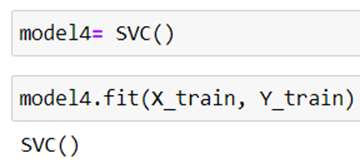


The text in the image provides some information about the variables that the classifier uses to make its predictions. The variables are "gross profit" and "total assets." The values for these variables are shown in parentheses after each node. The class that the classifier predicts is shown in the text on the right side of the tree.

For example, the first node in the tree asks if the gross profit is greater than 10,510. If the answer is yes, then the classifier predicts class D. If the answer is no, then the classifier goes to the next node, which asks if the total assets are greater than 285,014. If the answer is yes, then the classifier predicts class C. If the answer is no, then the classifier predicts class B.

## 5.5 “Model 4”: Support vector machine

The support vector machine (SVM) algorithm is a powerful supervised machine learning technique used for classification and regression tasks. SVM aims to find an optimal hyperplane in a high-dimensional feature space that best separates different classes of data points. It achieves this by constructing a decision boundary that maximizes the margin, i.e., the distance between the decision boundary and the closest data points from each class. SVM can handle both linearly separable and non-linearly separable datasets by using various kernel functions that transform the original input space into a higher-dimensional space where linear separation is possible. During training, SVM identifies a subset of data points called support vectors that play a crucial role in defining the decision boundary. In the case of classification, new instances are classified based on which side of the decision boundary they fall on. Let’s try to envision the support vector machine to fit in our dataset and variable of interest.



Here, model4 is assigned to an instance of the SVC class, which stands for Support Vector Classifier. The SVC class is part of the scikit-learn library and represents the implementation of the Support Vector Machine (SVM) algorithm for classification tasks. By creating model4 as an instance of SVC(), we are initializing a SVM classifier without specifying any hyperparameters. This allows the classifier to use default settings for parameters like the kernel function, regularization parameter, and so on.

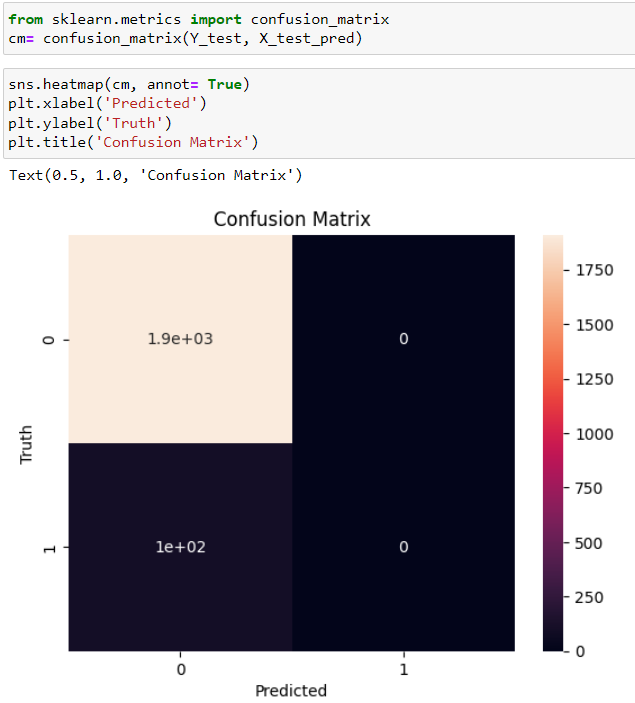
The code then proceeds to train the model4 SVM classifier using the fit() method by providing the labeled training data (X\_train for features and Y\_train for labels). This process involves finding the optimal hyperplane that maximizes the margin between different classes of data points, using the SVM algorithm's optimization principles.

### **5.5.1 Accuracy check on train and test data**

model4.score(X\_train, Y\_train) 0.947237431557989: This indicates that model4 achieved an accuracy of approximately 94.72% on the training dataset (X\_train and Y\_train). The accuracy represents the proportion of correctly predicted target values (Y\_train) with respect to the corresponding feature values (X\_train) used during training.

model4.score(X\_test, Y\_test) 0.9492537313432836: This shows that model4 achieved an accuracy of around 94.93% on the testing dataset (X\_test and Y\_test). The accuracy represents the proportion of correctly predicted target values (Y\_test) with respect to the corresponding feature values (X\_test) that were not used during training. This indicates how well model4 is able to generalize its predictions to new, unseen data.

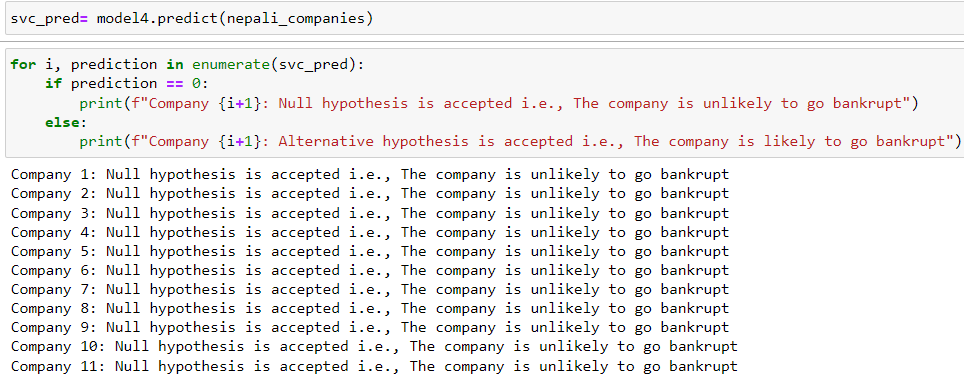
### **5.5.2 Evaluation and Visualization of SVM Model**



Here, the confusion\_matrix() function from scikit-learn's metrics module is imported. The confusion matrix is then computed by passing the true target values (Y\_test) and the predicted values (X\_test\_pred) to the confusion\_matrix() function, and the resulting matrix is assigned to cm. To visualize the confusion matrix, sns.heatmap() from the seaborn library is used. The cm matrix is passed as the data to be visualized, and the annot=True parameter is set to display the numerical values within the heatmap. Additional plot formatting is performed using matplotlib: plt.xlabel('Predicted') sets the x-axis label as "Predicted", plt.ylabel('Truth') sets the y-axis label as "Truth", and plt.title('Confusion Matrix') sets the title of the plot as "Confusion Matrix".

The confusion matrix here can be interpreted in a simple way. For 1900 times, The logistic regression model here predicted the output to be (0) and the truth was 0. Similarly, 102 times, the truth was 0 and the model predicted to be 1. We can clearly see a very low error rate.

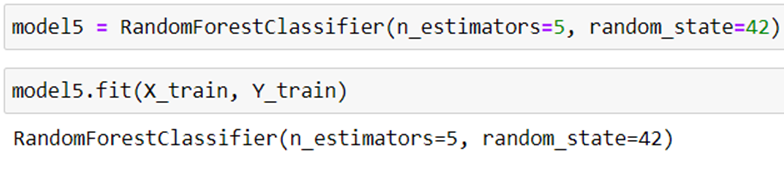
### **5.5.3 Actual prediction**



In the provided code, a Support Vector Machine (SVM) classifier machine learning model model4 is used to predict bankruptcy probabilities for Nepali companies in the nepali\_companies dataset. The model's predictions are stored in the array svc\_pred. A loop is then employed to iterate through each prediction in svc\_pred. For each company (indexed by i+1), the loop checks the prediction value. As observed from the output, all predictions are 0, which means the null hypothesis is accepted for each company. The null hypothesis states that the company is unlikely to go bankrupt. This implies that, according to the SVM model's predictions, none of the companies in the nepali\_companies dataset are predicted to go bankrupt. This interpretation provides valuable insights for stakeholders, suggesting that, based on the model's analysis, the assessed Nepali companies are less likely to face bankruptcy.

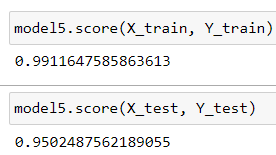
## 5.6 "Model 5": Random forest classifier

The random forest classifier is a versatile and powerful supervised machine learning algorithm used for classification tasks. It combines the principles of ensemble learning and decision trees to make predictions. In a random forest, multiple decision trees are constructed using random subsets of the training data and random subsets of the input features. Each decision tree independently makes predictions, and the final prediction is determined through a voting mechanism, where the class with the most votes is chosen. This ensemble approach helps to reduce overfitting, improve generalization, and increase accuracy. Random forests are robust to noise, handle high-dimensional data well, and can capture complex relationships between variables.



Here, we used the Random Forest classifier from scikit-learn library and created an instance of it called "model5". We specified that the random forest should consist of 5 decision trees by setting the parameter "n\_estimators" to 5. Additionally, we set the random seed to 42 using the "random\_state" parameter for reproducibility purposes. Next, we trained the model on the training data, where "X\_train" represents the input features and "Y\_train" represents the corresponding target labels. By calling the "fit" method on the model and passing the training data, we allowed the model to learn patterns and relationships between the input features and their corresponding labels.

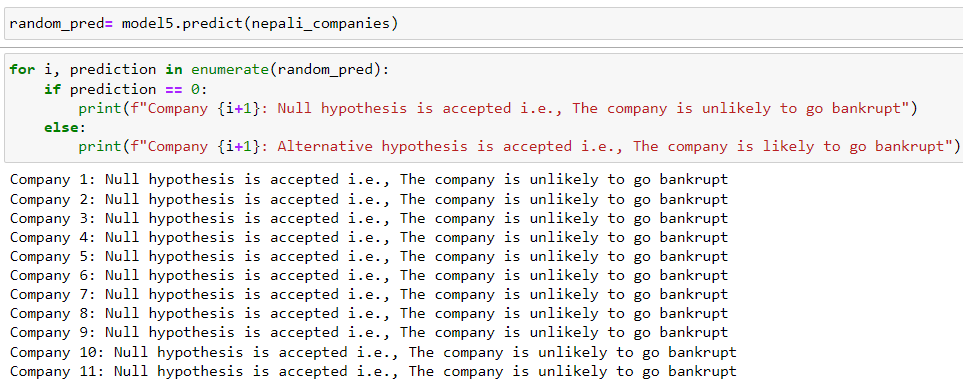
### **5.6.1 Accuracy check on train and test data**



model5.score(X\_train, Y\_train) 0.9911647585863613: This indicates that model5 achieved an accuracy of approximately 99.12% on the training dataset (X\_train and Y\_train). The accuracy represents the proportion of correctly predicted target values (Y\_train) with respect to the corresponding feature values (X\_train) used during training.

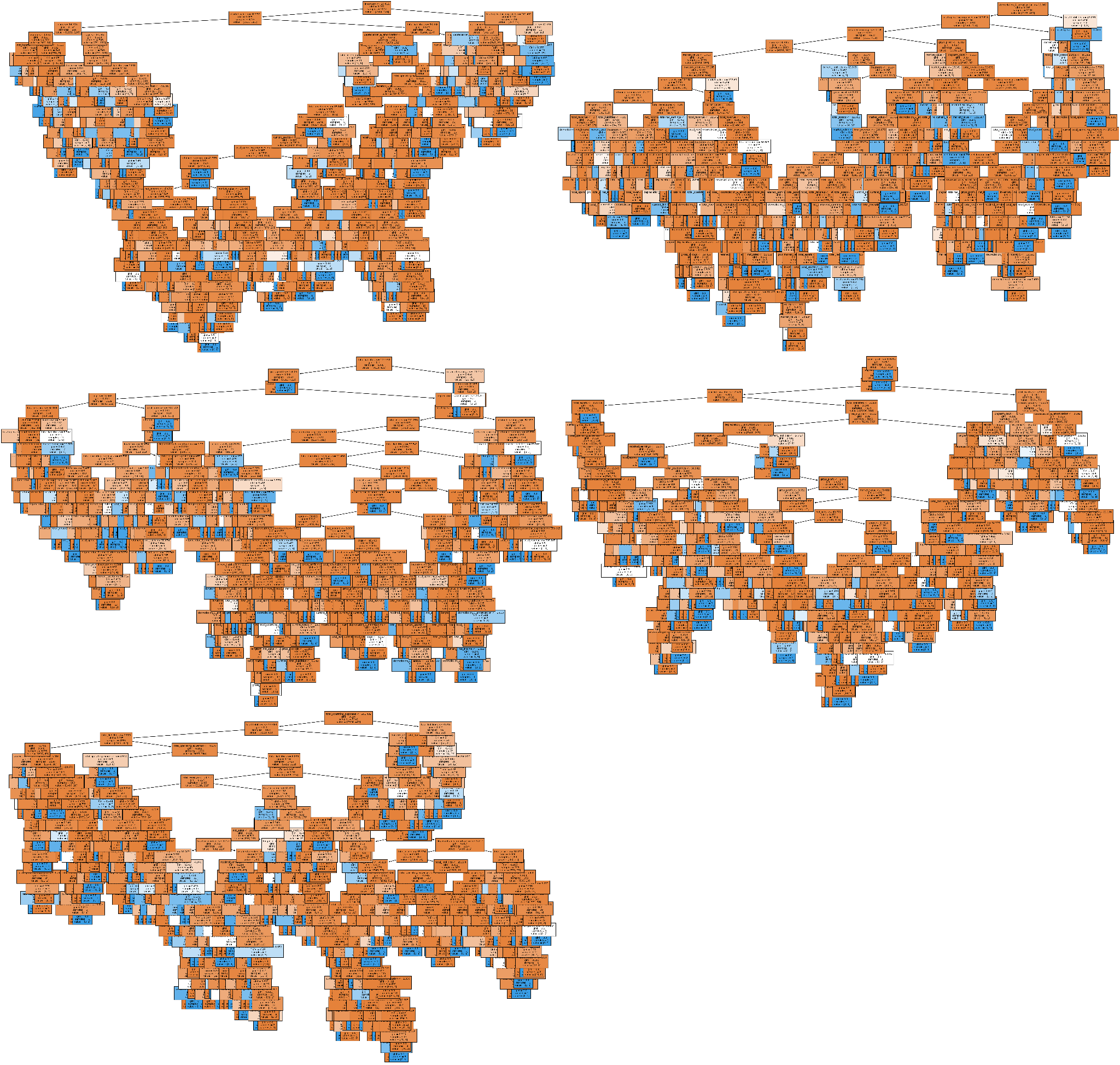
model5.score(X\_test, Y\_test) 0.9502487562189055: This shows that model5 achieved an accuracy of around 95.02% on the testing dataset (X\_test and Y\_test). The accuracy represents the proportion of correctly predicted target values (Y\_test) with respect to the corresponding feature values (X\_test) that were not used during training. This indicates how well model5 is able to generalize its predictions to new, unseen data.

### **5.6.2 Actual prediction**



In the provided code, a Random Forest classifier machine learning model model5 is used to predict bankruptcy probabilities for Nepali companies in the nepali\_companies dataset. The model's predictions are stored in the array random\_pred. A loop is then employed to iterate through each prediction in random\_pred. For each company (indexed by i+1), the loop checks the prediction value. As observed from the output, all predictions are 0, which means the null hypothesis is accepted for each company. The null hypothesis states that the company is unlikely to go bankrupt. This implies that, according to the Random Forest model's predictions, none of the companies in the nepali\_companies dataset are predicted to go bankrupt. This interpretation provides valuable insights for stakeholders, suggesting that, based on the model's analysis, the assessed Nepali companies are less likely to face bankruptcy.

### **5.6.3 Visualization of Random forest model**



In the above figure, Lets observe the first decision tree in the forest, we can observe the boxes with lighter colors signifying a higher gini coefficient.It simply means that there is higher IMPURITY there.

# Chapter 6

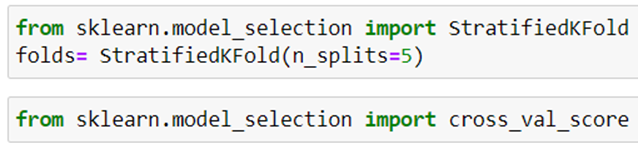
# Selection of best model for prediction

## 6.1 Stratified K folds cross validation

Cross-validation is a valuable technique used in data science to select the best model among multiple candidate models. It involves dividing the dataset into several subsets or folds, where each fold is used as a validation set while the remaining folds are used for training. The models are then trained and evaluated on each fold, and their performance metrics, such as accuracy or mean squared error, are recorded. By averaging the performance across all folds, cross-validation provides an objective measure of how well each model generalizes to unseen data. This enables data scientists to compare and select the model with the highest average performance, helping to identify the most effective model for a given task.

We will be selecting the model with the lowest standard deviation in the cross validation scores. The standard deviation is a measure of the dispersion or spread of a dataset. In this context, the standard deviation of the cross-validation scores indicates the degree of variation in the model's performance across the different folds. A lower standard deviation suggests more consistent performance across the folds, while a higher standard deviation implies greater variability in the model's performance. In this case, the relatively low standard deviation value suggests that the model's performance is consistent and stable across the cross-validation folds.

### **6.1.1 Importing required libraries**

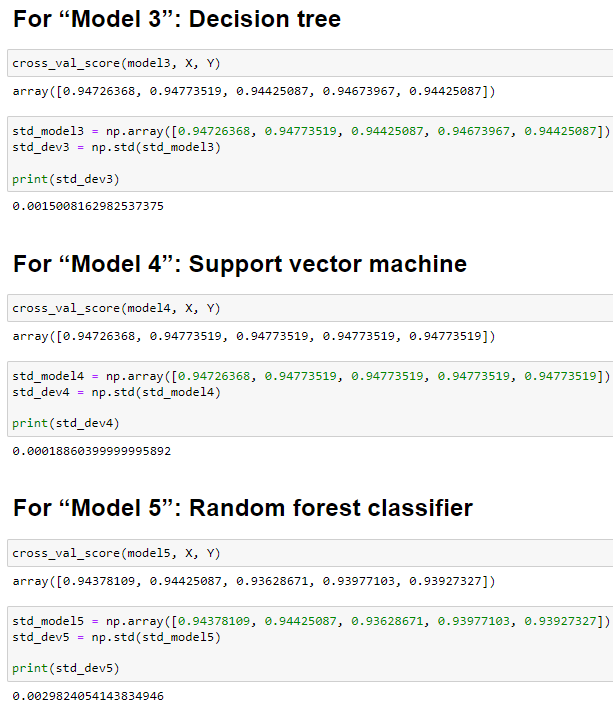


Here, the StratifiedKFold class from the sklearn.model\_selection module. StratifiedKFold is a cross-validation method that splits the dataset into multiple folds while preserving the distribution of class labels. In this case, it is used to create an instance of StratifiedKFold with 5 splits (folds).

Next, the code snippet imports the cross\_val\_score function from the same module. This function is used to perform cross-validation by splitting the dataset into folds and evaluating a model's performance on each fold. It takes as input a machine learning model, the dataset, and a cross-validation strategy (in this case, the StratifiedKFold object) to perform the evaluation. The output of cross\_val\_score is an array of scores, where each score represents the performance of the model on a particular fold.

### **6.1.2 Calculating the cross validation score and standard deviation between the folds for each model**



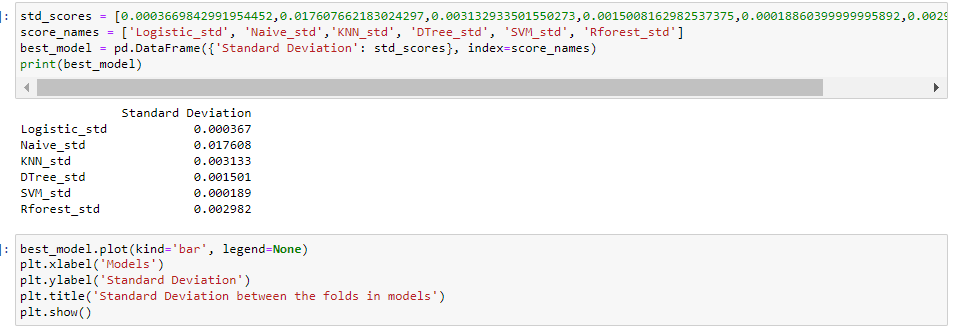


Here, 6 different classification algorithms, "Logistic Regression," "Naive Bayesian Regression," "K Nearest Neighbor," "Decision Tree," and "Support Vector Machine," as well as a "Random Forest Classifier," have been evaluated using cross-validation to estimate their performance consistency.

The cross\_val\_score function from scikit-learn is used to perform k-fold cross-validation on each model, where k is the number of folds (default is 5). For each model, the function returns an array of accuracy scores obtained in each fold. The numpy.std function is then used to calculate the standard deviation of these accuracy scores. The standard deviation serves as a measure of how much the accuracy scores vary across the different folds.

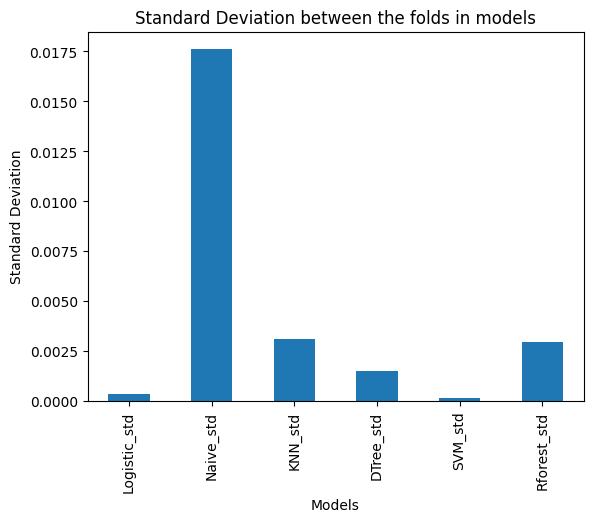
Interpreting the results, a lower standard deviation indicates that the model's performance is relatively consistent across different folds, suggesting that it generalizes well to different subsets of the data. On the other hand, a higher standard deviation implies that the model's performance varies significantly depending on the training and testing data splits, indicating possible instability and inconsistency in the predictions.

### **6.1.3 Visualization of std deviation of models across the folds**



Here, we created a DataFrame named best\_model using pandas. The DataFrame consists of a single column labeled "Standard Deviation" that contains the values from the std\_scores list. Each value corresponds to the standard deviation of a specific model's performance across different folds. The index of the DataFrame is set using the score\_names list, which contains the names of the respective models.

Next, we created a bar chart using the plot function of the best\_model DataFrame. The kind parameter is set to 'bar', indicating that we want to create a bar chart. Additional labels and title are added to the plot using the xlabel, ylabel, and title functions from plt.



### **6.1.4 Reason for selection of SVM classifier**

Here, this visualization allows for a quick comparison of the models' performance stability, **with lower bar heights indicating more consistent performance and higher bar heights indicating more variability. Among all the models, SVM has the lowest standard deviation across the folds which means that the predictions will have a higher and consistent accuracy through this model. Therefore, SVM will be selected as our final model for prediction.**

# 

# 

# 

# 

# Chapter 7

# Managerial implications and limitations of the study

## 7.1 Managerial implications

Based on the BIVARIATE AND MULTIVARIATE DATA ANALYSIS, the following managerial implications have been decided as a part of evidence based decision making.

* **Efficient Cost Control and Profitability Analysis:** Companies should prioritize effective cost control measures to manage operating expenses and ensure a healthy EBIT. Regularly analyzing profitability by assessing revenue, COGS, and operating expenses can help identify areas for improvement, cost-saving opportunities, and ensure the company's financial sustainability.
* **Proactive Debt Management:** As higher total liabilities were observed in bankrupt companies, it is crucial for businesses to manage their debt levels prudently. Companies should aim to strike a balance between debt and equity financing and avoid overleveraging. Regularly reviewing and optimizing the debt structure can reduce the risk of financial distress and improve financial flexibility.
* **Optimized Inventory Management:** Since higher inventory levels were associated with bankrupt companies, optimizing inventory management is essential. Companies should implement effective inventory tracking systems, adopt just-in-time practices where suitable, and closely monitor inventory turnover to ensure proper stock levels aligned with market demand. This can help free up capital, improve cash flow, and reduce the risk of obsolescence. Example, Just in Time inventory system could be haelpful.
* **Revenue Diversification and Market Adaptation:** Given the unusual finding of higher total revenue in bankrupt companies, companies should focus on diversifying their revenue streams and being responsive to changing market conditions. Diversification can help mitigate risks associated with dependency on a single revenue source, while adaptability can allow the company to remain competitive and resilient in evolving business environments.
* **Early Warning System and Financial Risk Assessment:** Developing an early warning system to track key financial indicators, such as current liabilities, revenue trends, EBIT margins, and inventory turnover, can help identify warning signs of potential financial distress. Conducting regular financial risk assessments can enable management to take timely corrective actions and implement appropriate strategies to avoid bankruptcy.

## 7.2 Limitations of the study

* **Probable Sample Bias:**

In the context of predicting companies' bankruptcy, sample bias refers to the possibility that the dataset used for analysis may not accurately represent the entire population of companies in the target region or industry. If the dataset primarily includes companies from specific sectors, sizes, or geographic locations, it may not provide a comprehensive view of the overall business landscape. As a result, the analysis could be biased and the findings may not be generalizable to the entire population of companies in the region or industry. To address this concern, it is essential to ensure that the dataset used for bankruptcy prediction is diverse and representative of various business sectors and sizes in the target country, such as Nepal. This can be achieved by carefully selecting a well-rounded and balanced dataset that includes companies from different industries and regions.

* **Data Quality:**

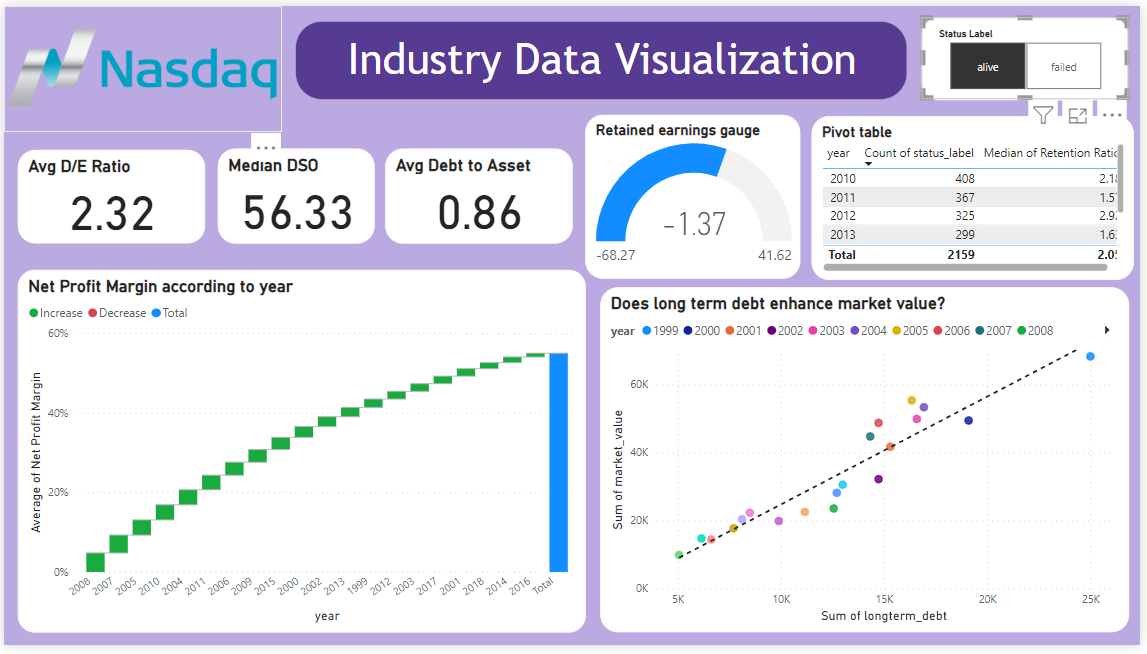
Data quality is critical in the context of bankruptcy prediction as inaccurate or incomplete information can lead to biased or unreliable results. It is essential to ensure that the dataset used for analysis is clean, consistent, and free from errors. Missing values or inconsistencies in financial indicators and other variables can impact the accuracy of bankruptcy predictions. For instance, if important financial ratios or balance sheet data have missing values or errors, it could lead to incorrect assessments of a company's financial health and bankruptcy risk. Therefore, thorough data cleansing and validation processes are essential to enhance the reliability of the analysis and predictions.

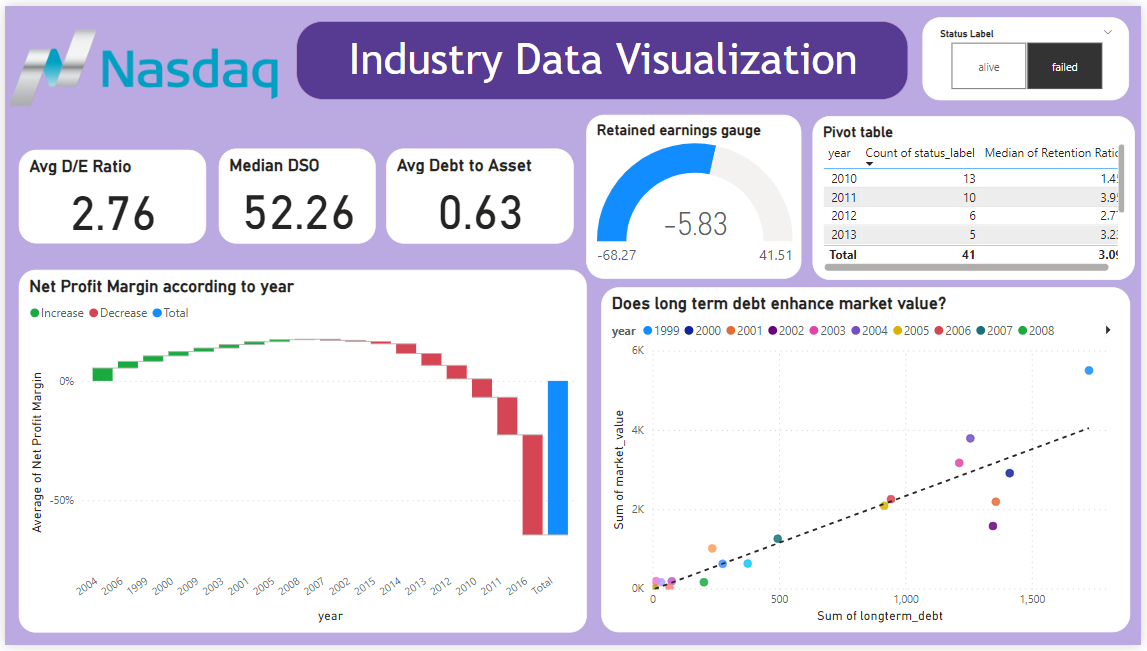
* **Exogenous Factors:**

In the context of bankruptcy prediction, exogenous factors refer to external influences or events that could affect a company's financial health and potential bankruptcy risk. These factors might not be captured within the dataset used for analysis, but they can significantly impact the financial performance of companies. For example, changes in government policies, economic conditions, or industry-specific regulations can influence companies' profitability and financial stability, leading to potential bankruptcies. It is crucial to be aware of these external factors and consider them while interpreting the bankruptcy prediction results. However, since these external factors might not be directly available in the dataset, additional research and domain knowledge are required to account for their potential effects and improve the accuracy of the bankruptcy prediction models.

# Power BI section

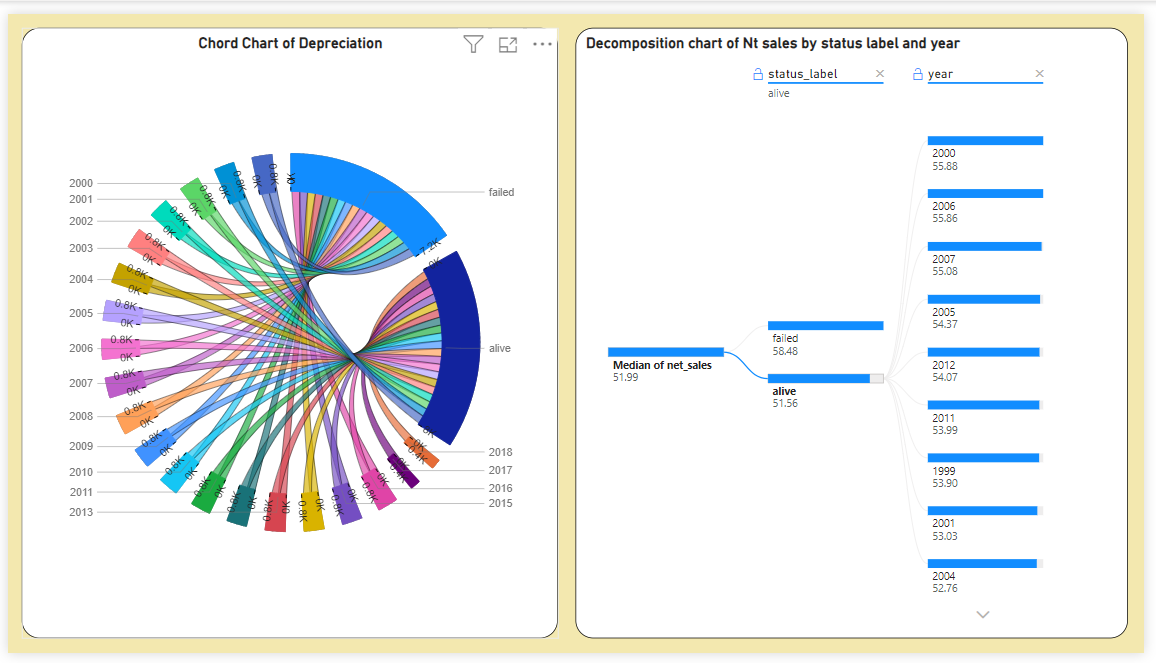
## Industry data visualization



****

This dashboard contains visualizations

* The average debt-to-equity ratio (D/E ratio) for companies listed on the Nasdaq Composite is 2.76. This means that, on average, companies listed on the Nasdaq have $2.76 in debt for every $1 in equity.
* The median D50 (50th percentile) D/E ratio for companies listed on the Nasdaq Composite is 52.26. This means that half of the companies listed on the Nasdaq have a D/E ratio of 52.26 or higher.
* This means that the average debt-to-asset ratio for all the companies in this industry is 0.85. A debt-to-asset ratio of 0.85 is considered to be in the good range. This means that the company has a relatively low amount of debt compared to its assets. This is a good sign for the company's financial health, as it means that they are less likely to default on their debts.
* The waterfall chart shows the net profit margin for all the companies from the year 1999 to 2016. It can be seen that the profit margin is decreasing but is still positive for the active companies but can be seen that it is in the negative range for the failed companies from the year 2002.
* The gauge chart shows the retained earnings range from its minimum value to the maximum value along with the average value of retained earnings.
* The correlation between long-term debt and market value is shown to be positive in the long term so it does suggest that long term debt enhances the market value of the companies



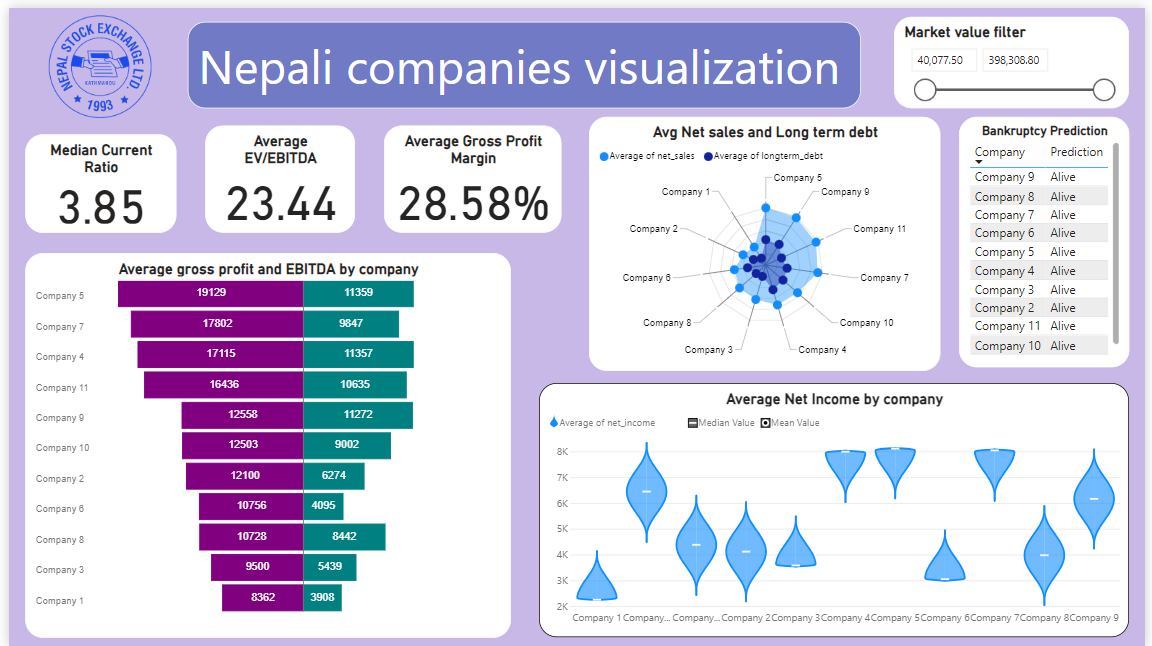
This dashboard gives further insight into the companies data as the decomposition chart shows the breakdown of net sales by the status of the companies and then the companies itself.

The chord chart shows the breakdown of depreciation on the basis of year and the status of the companies.



This is the tooltip for the company's dashboard . This contains a line chart about the average operating expenses for all the companies . It can be seen that C-2708 has the highest operating expenses among all the companies while c-167 has the lowest operating expenses.

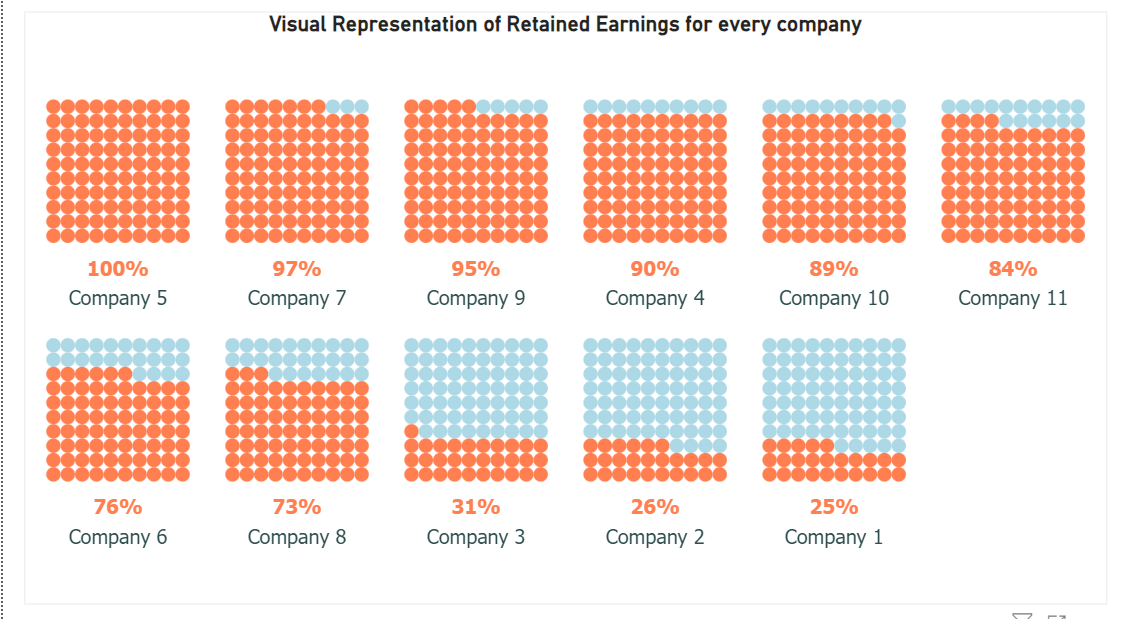
## Predicted Nepali companies visualization



* The median current ratio is a measure of a company's liquidity. A current ratio of 3.85 means that the company has 3.85 times its current liabilities in current assets. This is considered to be a healthy current ratio, as it means that the company has enough cash and other liquid assets to pay its current liabilities.
* The average EV/EBITDA is a measure of a company's valuation. EV/EBITDA is calculated by dividing the company's enterprise value by its earnings before interest, taxes, depreciation, and amortization (EBITDA). An EV/EBITDA of 23.44 means that the company is valued at 23.44 times its EBITDA. This is considered to be a high valuation, as it means that investors are paying a lot for the company's future earnings.
* The average gross profit margin is a measure of the company's profitability. A gross profit margin of 28.58% means that the company keeps 28.58% of its revenue after paying for the cost of goods sold. This is considered to be a good gross profit margin, as it means that the company is making a healthy profit on each sale.
* The tornado chart shows the average gross profit and EBITDA for 10 companies. The companies are ranked in ascending order of average gross profit, with Company 1 having the highest average gross profit and Company 10 having the lowest average gross profit.

The graph shows that Company 5 has the highest average gross profit, at 19,129. This means that Company 5 keeps 19,129 of every 100 dollars it takes in revenue after paying for the cost of goods sold.

* The radar chart shows the average net sales and long-term debt for 10 companies. The companies are ranked in ascending order of average net sales, with Company 1 having the highest average net sales and Company 10 having the lowest average net sales.



The graph shows the percentage of retained earnings for 10 companies. The companies are ranked in descending order of retained earnings, with Company 5 having the highest percentage of retained earnings and Company 1 having the lowest percentage of retained earnings.