COMPANY BANKRUPTCY PREDICTION

COMPANY BANKRUPTCY PREDICTION

A mini project-2 report submitted in partial fulfillment of requirements for the award of Degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

By

CHAKALI MADHU (199X1A0543)

ARADESI PRAMOD (199X1A0535)

KARRE SHIVA KUMAR (199X1A0556)

Under the Esteemed guidance of

Sri. P. Praveen Yadav
Assistant Professor
Department of C.S.E.

Department of Computer Science and Engineering

G. PULLA REDDY ENGINEERING COLLEGE (Autonomous): KURNOOL

(Affiliated to JNTUA, ANANTAPURAMU)

2022- 2023

G. PULLA REDDY ENGINEERING COLLEGE (Autonomous): KURNOOL

(Affiliated to JNTUA, ANANTAPURAMU)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the mini project-2 work entitled 'COMPANY BANKRUPTCY PREDICTION' *is a bona fide record of work carried out by*

C. MADHU (199X1A0543)

A. PRAMOD (199XA0535)

K. SHIVA KUMAR (199X1A0556)

Under my guidance and supervision in fulfillment of the requirements for the award of degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE & ENGINEERING

Sri P. Praveen Yadav	**Dr. N. Kasi Viswanath**
Assistant Professor,	Professor & Head of the Department,
Department of C.S.E.,	Department of C.S.E.,
G. Pulla Reddy Engineering College,	G. Pulla Reddy Engineering College,
Kurnool.	Kurnool.

DECLARATION

We hereby declare that the project titled “COMPANY BANKTRUPCY PREDICTION” is the authentic work carried out by us as students of G. PULLA REDDY ENGINEERING COLLEGE (Autonomous) Kurnool, during August 2022 – December 2022 and has not been submitted elsewhere for the award of any degree or diploma in part or in full to any institute.

CHAKALI MADHU
(199X1A0543)

ARADESI PRAMOD
(199X1A0535)

KARRE SHIVA KUMAR
(199X1A0556)

ACKNOWLEDGEMENT

We wish to express our deep sense of gratitude to our project guide, **Mr.P. Praveen Yadav**, Assistant Professor of CSE Department, G. Pulla Reddy Engineering College, for his immaculate guidance, constant encouragement, and cooperation which have made it possible to bring out this project work.

We are grateful to our project in charge **Smt.T.Swathi**, Assistant Professor of CSE Department, G. Pulla Reddy Engineering College, for helping us and giving us the required information needed for our project work.

We are thankful to our Head of the Department **Dr.N.KasiViswanath**, for his wholehearted support and encouragement during the project sessions.

We are grateful to our respected Principal **Dr.B.Sreenivasa Reddy** for providing the requisite facilities and helping us in providing such a good environment.

We wish to convey our acknowledgements to all the staff members of the Computer Science and Engineering Department for giving the required information needed for our project work.

Finally, we wish to thank all our friends and well wishers who have helped us directly or indirectly during the course of this project work.

ABSTRACT

Bankruptcy is legal proceeding initiated when a person or business is unable to repay outstanding debts or obligations. The bank process begins with a petition filed by the debtor, which is most common, or on behalf of creditors, which is less common. All of the debtor’s assets are measured and evaluated, and the assets may be used to repay a portion of the outstanding debt. Due to economic difficulties, companies are facing the risk of bankruptcy. In a financial point of view, predicting if a company will bankrupt can reduce possible waste of precision resources like time and money. Bankruptcy occurs when debts which are not paid to creditors in full are forgive for the owners. If we file for bankruptcy, it may be tough for us to get a new loan if we plan to start afresh. So, it’s better to check the current status of the company and by using the current technologies the company will know that whether they will survive or not in following years based on the current economic conditions. By knowing this will enable the company to take precautionary measures. The main aim of this project is to develop a prediction model to decide whether a company will bankrupt or not in the following year. The company bankruptcy was defined based on the business regulations of the Taiwan Stock Exchange. In this project we will use various classification algorithms on bankruptcy datasets to predict bankruptcies with satisfying accuracies long before the actual event.

The algorithms are Decision Tree and Gradient Boosting. Decision Tree, it is flowchart like structure in which each internal node represents a test on feature and leaf nodes represents a class labels. Gradient Boosting, it is a popular boosting algorithm. In gradient boosting, each predictor corrects it’s predecessor’s error.

CONTENTS

1. Introduction……………………………………………………….1

1.1 Introduction………………………………………………….. 1

1.2 Motivation……………………………………………………4

1.3 Objective Of The Project……………………………………...4

1.4 Limitations Of The Project……………………………………5

1.5 Organization Of The Report…………………………………..6

2. System Specifications……………………………………...7

2.1 Hardware Requirements………………………………………8

2.2 Software Requirements……………………………………….8

3. Literature Survey…………………………………………….9

3.1 Introduction…………………………………………………..10

3.2 Existing System………………………………………………11

3.3 Proposed System……………………………………………..12

4. Design & Implementation…………………………………...13

4.1 Introduction…………………………………………………14

4.2 Machine Learning……………………………………………16

4.3 Steps Involved In Project……………………………………19

4.4 Classification Models……………………………………………..26

4.5 Dimensionality Reduction…………………………………...30

4.6 Data Visualization…………………………………………...35

4.7 Source Code………………………………………………....37

5. Webpage Creation……………………………………..44

5.1 Visual Studio Code…………………………………………45

5.2 Source Code For Indexs.html……………………………….46

5.3 Source Code For App.py……………………………………50

6. OutPut Screens…………………………………………...52

7. Testing and Validation………………………………..58

7.1 Introduction…………………………………………………59

7.2 Testing Strategies…………………………………………...59

8. Conclusion……………………………………………..62

9. References……………………………………………..64

LIST OF FIGURES

S. No	Figure No.	Figure Name	Page No.
1	Fig 3.1	Altman Z-Score Formula	10
2	Fig 4.2.1	Workflow of Machine learning	17
3	Fig 4.2.2	Classification Algorithm	19
4	Fig 4.3.1	Workflow of Implementation	20
5	Fig 4.3.2	Bankruptcy Dataset	22
6	Fig 4.4.1	Workflow of Gradient Boosting	27
7	Fig 4.4.2	Structure of Decision Tree	29
8	Fig 4.5.1	Example of PCA	32
9	Fig 4.5.2	PCA Components Dataset	34
10	Fig 4.6.1	Principal Component Analysis	35
11	Fig 4.6.2	Variation in performance of model	37
12	Fig 6.1	Decision Tree Output	54
13	Fig 6.2	Gradient Boosting Output	54
14	Fig 6.3.1	Indexs.html Code	55
15	Fig 6.3.2	Indexs.html Code	55
16	Fig 6.3.3	Indexs.html Code	56
17	Fig 6.3.4	Indexs.html Code	56
18	Fig 6.4	App.py Code	57
19	Fig 6.5	Webpage Output	58

INTRODUCTION

1. INTRODUCTION

1.1 INTRODUCTION

There are many types of companies in this world in different areas. Everyone expects profit and growth for his company. So it is essential to know its present and future status of company. So we decide it by taking into consideration all areas of the company. The parameters like industrial risk, management risk, financial flexibility, creditability, competitiveness, operating risk…etc. These are all helps us to find the status of the company. Company Bankruptcy, is a legal proceeding initiated when a person or business unable repay the outstanding debts and obligations. The bankruptcy process begins with a petition filed by the debtor, which is most common, or on behalf of creditors, which is less common. All of the debtor's assets are measured and evaluated, and the assets may be used to repay a portion of the outstanding debt. The type of bankruptcy proceedings chapter 7 or chapter 11 generally provides some clue as to whether the average investor will get back all, a by-case basis. There is also pecking order of creditors and investors, which dictates who gets paid first, second and last. Chapter 7, under chapter 7 of U.S. Bankruptcy Code, "the company stop all operations and goes completely out of business. A trustee is appointed to liquidate the company's assets, and the money is used to pay off debt, "the U.S. Securities and Exchange Commission notes. Chapter 11, the company doesn't go out of business but is allowed to reorganize. A company filling chapter 11 hopes to return to normal business operations and sound financial health in the future. This type of bankruptcy is generally filed by corporations that need time to restructure that has become unmanageable.

Company bankruptcy is often a very big problem for companies. The impact of bankruptcy can cause losses to elements of the company such as owners, investors, employees, and consumers. One way to prevent bankruptcy is to predict the possibility of bankruptcy based on the company's financial data. Therefore, this study aims to find the best predictive model or method to predict company bankruptcy using the dataset from companies bankruptcy.

Company Bankruptcy Prediction, it takes into consideration the areas of the company and tells whether the company will go bankrupt in the future or not. Bankruptcy is the concept of financial accounts. If you are one of the data science enthusiasts who started with data science after commerce then you should be aware of what bankruptcy is. By using machine learning algorithms we can train a model to predict whether a company or a legal person will become bankrupt in future or not. In the section below, I will take you through a machine learning tutorial on how to train a model for the task of bankruptcy of a company by using the Python programming language.

Predicting bankruptcy of companies has been a hot subject of focus for many economists. The rationale for developing and predicting the financial distress of a company is to develop a predictive model used to forecast the financial condition of a company by combining several econometric variables of interest to the researcher. The importance of the area is due in part to the relevance for creditors and investors in evaluating the likelihood that a firm may go bankrupt. Bankruptcy prediction is the problem of detecting financial distress in businesses which will lead to eventual bankruptcy. Bankruptcy prediction has been studied since at least 1930s. The early models of bankruptcy prediction employed univariate statistical models over financial ratios. The univariate models were followed by multi-variate statistical models such as the famous Altman Z-score model. The recent advances in the field of Machine learning have led to the adoption of Machine learning algorithms for bankruptcy prediction. Machine Learning methods are increasingly being used for bankruptcy prediction using financial ratios. Bankruptcy is a state of insolvency wherein the company or the person is not able to repay the creditors the debt amount. Bankruptcy prediction is of importance to the various stakeholders of the company as well as the society on the whole. The research shall analyse the financial statements and market data of these companies. We shall then try to determine how far back these models are able to predict that the companies would get into financial distress. The major contribution of our study will be to identify a suitable model for bankruptcy prediction in the Indian context.

Machine learning methods are increasingly being used to predict company bankruptcy. Comparative studies carried out on selected methods to determine their suitability for predicting company bankruptcy have demonstrated high levels of prediction accuracy for the extreme gradient boosting method in this area. This method is resistant to outliers and relieves the researcher from the burden of having to provide missing data. The aim of this study is to assess how the elimination of outliers from data sets affects the accuracy of the extreme gradient boosting method in predicting company bankruptcy. The added value of this study is demonstrated by the application of the extreme gradient boosting method in bankruptcy prediction based on data free from the outliers reported for companies which continue to operate as a going concern. The research results indicate that it is possible to increase the detection rate for bankrupt companies by eliminating the outliers reported for companies which continue to operate as a going concern from data sets.

1.2 MOTIVATION

The main intention for us in doing project is, Company Bankruptcy Prediction is the most important part for predict whether their company or organization will go bankrupt or not. Apparently, the accurate prediction of the result is the most important task to be carried out. In this project we build a machine learning model using the Ensemble Classifiers.

There are many researchers who have worked and are working on creating and developing different prediction models using regression that will predict the probability of chance of bankruptcy but they did not show good outcomes on unseen data which may have false prediction on the real time data.

In Prediction accuracy is more important since the model have good accuracy it gives better prediction, based on the prediction business man can know the whether they company will go bankrupt or not in the following year. In this, we have motivated to improve the accuracy of the project by selecting a best model that has a good accuracy score and implemented Webpage Using Flask there by user can give inputs to the system and it predicts and gives output to the user.

1.3 OBJECTIVE OF THE PROJECT

After the introduction and summary above, now we are trying to discuss possible innovative change and future trend in bankruptcy prediction using machine learning techniques. One particular trend is the diversification of data sources. Former bankruptcy prediction papers would usually use numerical data, such as financial statement data, accounting data. Now the textual data, like news or public report even some comments from experts, are used to do prediction. It generates a new concept called Multiple-Source Heterogeneous Data.

By checking the bankruptcy the company knows its position like strong areas and weaker areas of the company. After knowing the strong and well areas the company will work on those areas. So we can prevent the company from incurring losses by taking precautionary measures. To find out all this we will test the Bankruptcy Model. So in this project we develop a high accurate model which gives us good result. After knowing the bankruptcy result, the company can focus on its problem and develop.

KEY TAKEAWAYS

- Bankruptcy is legal proceeding carried out to allow individuals or businesses freedom from their debts, while simultaneously providing creditors an opportunity for repayment.
- Bankruptcy is handled in federal courts, and rules are outlined in the U.S. Bankruptcy Code.
- For instance, Chapter 11 bankruptcy allows businesses to reorganize and remerge while Chapter 7 relates to individual bankruptcy.
- Bankruptcy can allow you a fresh start, but it will stay on your credit reports for a number of years and make it difficult to borrow in the future.

After occurring bankruptcy, the company may lose their company or assets. Sometimes, people or companies may want to avoid bankruptcy, and there are several alternatives that may be able to reduce your debt obligations. For example company ABC has bankruptcy then it works on weaker areas and to rectify those, solve the problems in those areas and improve the strategies in that area to overcome from failures. So, there is chance to company will get out from the bankruptcy in the following years. So it’s better to know the current status of the company.

1.4 LIMITATIONS OF THE PROJECT

- The main point of all researches are carried out is that only predicts the chance of Company Bankruptcy Prediction but the result not exactly correct.

- It’s not possible thing to predict the exact result of the company future it may also depends on the some external factors of the company.

- So, In this project we try to implement the model which is our level best and it predicts the maximum of accurate results.

1.5 ORGANIZATION OF THE REPORT

The first chapter deals with the introduction of the Company Bankruptcy Prediction, objective of this project, existing systems and proposed systems of the project. The objective of the project explains about main goal of the project. The existing systems explains about presently working systems and proposed system is about what we are coming up with in the future when compared to the present systems. The second chapter deals with system specifications required for developing the project, it includes hardware and software specifications. The third chapter gives preview about design and implementation with source code, output screens, testing and validation. The fourth chapter end with the conclusion of this project.

SYSTEM SPECIFICATIONS

2. SYSTEM SPECIFICATIONS

2.1 HARDWARE SPECIFICATIONS

- Processor: intel i3
- RAM: 4GB
- Hard disk: 40 GB or above

2.2 SOFTWARE SPECIFICATION

- Operating System: Windows
- Programming Language: Python
- Platform: Anaconda Navigator (Jupyter Notebook)

LITERATURE SURVEY

3. LITERATURE SURVEY

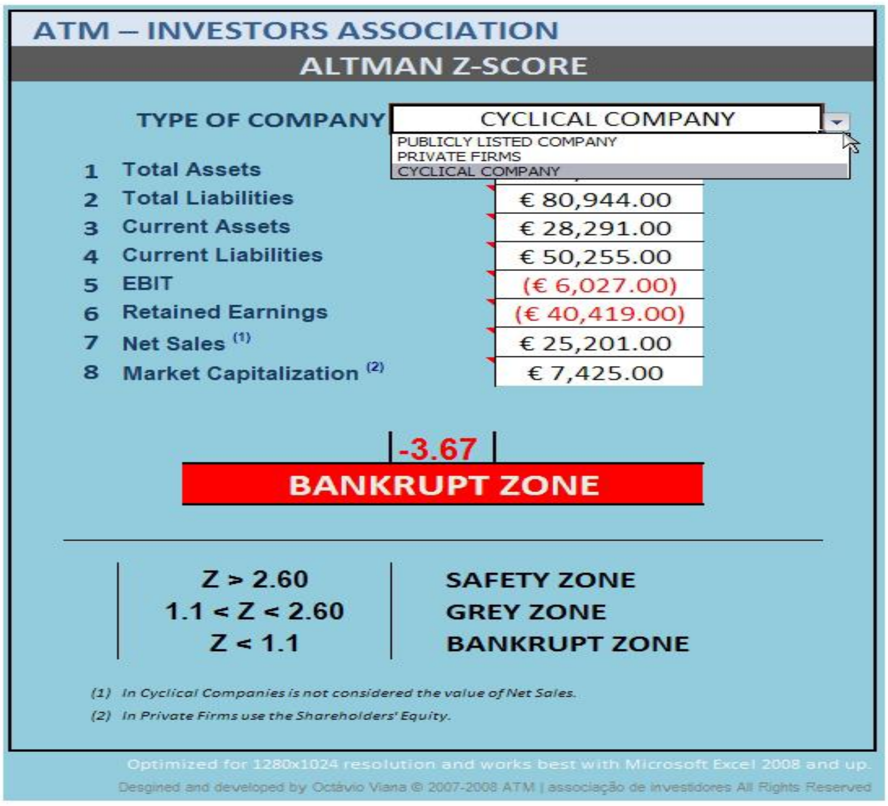
3.1 INTRODUCTION

1. In this project we use Machine Learning, it is the art & science of programming computers to learn from data. Machine Learning is the study that gives computers the ability to learn without being explicitly programmed.

2. In Machine learning we use both Decision Tree and Gradient Boosting Algorithm to build a model for company bankruptcy.

3. The dataset we collected from the Taiwan Economic Journal for the years 1999 to 2009. Company bankruptcy was defined based on the business regulations of the Taiwan Stock Exchange.

4. From the figure 3.1, it is the first formal multiple variable analysis, Edward I._Altman implemented one formula to find company bankruptcy in 1968.


ATM – INVESTORS ASSOCIATION

ALTMAN Z-SCORE

TYPE OF COMPANY: CYCLICAL COMPANY

1	Total Assets	
2	Total Liabilities	€ 80,944.00
3	Current Assets	€ 28,291.00
4	Current Liabilities	€ 50,255.00
5	EBIT	(€ 6,027.00)
6	Retained Earnings	(€ 40,419.00)
7	Net Sales (1)	€ 25,201.00
8	Market Capitalization (2)	€ 7,425.00

-3.67

BANKRUPT ZONE

Z > 2.60	SAFETY ZONE
1.1 < Z < 2.60	GREY ZONE
Z < 1.1	BANKRUPT ZONE

(1) In Cyclical Companies is not considered the value of Net Sales.

(2) In Private Firms use the Shareholders' Equity.

Optimized for 1280x1024 resolution and works best with Microsoft Excel 2008 and up.

Desgined and developed by Octávio Viana © 2007-2008 ATM | associação de investidores All Rights Reserved

Fig 3.1 Altman Z-Score Formula

5. The formula for Z-Score is Z = 1.2*X1 + 1.4*X2 + 3.3*X3 + 0.6*X4 + 1*X5.

6. In this project, we need to get lot of information (parameters) about the company from the client.

3.2 EXISTING SYSTEM

The history of bankruptcy prediction includes application of numerous statistical tools which gradually became available, and involves deepening appreciation of various pitfalls in early analyses. Research is still published that suffers pitfalls that have been understood for many years.

- In 1968, in the first formal multiple variable analysis, Edward I. Altman applied multiple discriminant analysis within a pair-matched sample. One of the most prominent early models of bankruptcy prediction is the Altman Z-score, which is still applied today.
- Zikeba et al. (2016) prepared the Polish companies bankruptcy dataset and prepared an ensemble Boosted Decision Tree model with synthetic feature generation for the prediction task. It used accuracy as the metric to evaluate their models-however, given such a high class imbalance, a model that is completely biased towards ‘not bankrupt’ would also have an accuracy of about 95%.
- Quynh and Lan Phuong (2020) proposed a new approach to take best ensemble models as on “fair voting”, and a procedure to handle missing values using Random Forest algorithm.
- Ren (2020) searched for common financial indicators of bankruptcy by comparing diverse financial markets in China and Poland.
- Bankruptcy prediction is the art of predicting bankruptcy and various measures of financial distress of public firms. It is a vast area of finance and accounting research. The importance of the area is due in part to the relevance for creditors and investors in evaluating the likelihood that a firm may go bankrupt.

Bankruptcy prediction is the art of predicting bankruptcy and various measures of financial distress of public firms. It is a vast area of finance and accounting research. The importance of the area is due in part to the relevance for creditors and investors in evaluating

the likelihood that a firm may go bankrupt. These are all existing models and procedures to find the bankruptcy of company and we have some more methods and also we can implement the bankruptcy using some additional models.

3.3 PROPOSED SYSTEM

In this project we have used the one of the ensemble technique, already there some existing ensemble techniques are there, so we using Gradient Boosting Classification Model. Gradient Boosting is a popular boosting algorithm. In gradient boosting, each predictor corrects its predecessor's error. Gradient Boosting gave us a good accurate result. We also implement Decision Tree Algorithm to compare the results with Gradient Boosting Algorithm to conclude the best accurate model. Decision Tree is a flowchart –like structure in which each internal node represents a test on feature and leaf nodes represents a class labels

For Future Prospective

We plan to extend our project by adding Frontend technologies to that project. In this user manually enter the different parameters of company and get the result. We have to make the project user friendly, it can be understandable to everyone and simple to operate. We may introduce a good and efficient web page to our project.

DESIGN AND IMPLEMENTATION

4. DESIGN AND IMPLEMENTATION

4.1 INTRODUCTION

Software Requirements are the starting points of the software developing activity. As system grew complex it became evident that the goal of the entire system cannot be easily comprehended. Hence the need for the requirements phase arose. The software is initiated by the client's needs. The SRS is the means of translating the ideas of the minds of the clients (the input) into a formal document (the output of the requirement phase).

SOFTWARE REQUIREMENTS

- Python
- Pickle
- Scikitlearn
- Pandas
- Plotly
- Numpy
- Flask

Python:

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands. Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol68, Small Talk, and Unix shell and other scripting languages. Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL). Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress. Python's features include − Easy-to-learn − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.

Pickle

Python pickle module is used for serializing and de-serializing python object structures. The process to converts any kind of python objects (list, dict, etc.) into byte streams (0s and 1s) is called pickling or serialization or flattening or marshalling. We can converts the byte stream (generated through pickling) back into python objects by a process called as unpickling. In real world sceanario, the use pickling and unpickling are widespread as they allow us to easily transfer data from one server/system to another and then store it in a file or database.

Scikitlearn

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

Pandas

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. Pandas is mainly used for data analysis. Pandas allows importing data from various file formats such as comma-separated values, JSON, SQL, Microsoft Excel. Pandas allows various data manipulation operations such as merging, reshaping, selecting, as well as data cleaning, and data wrangling features.

Numpy

Numpy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. It can be utilized to perform a number of mathematical operations on arrays such as trigonometric, statistical, and algebraic routines.

Plotly

Plotly's Python graphing library makes interactive, publication-quality graphs. Examples of how to make line plots, scatter plots, area charts, bar charts, error bars, box plots, histograms, heatmaps, subplots, multiple-axes, polar charts, and bubble charts.

Flask

Flask is a web application framework written in Python. It was developed by Armin Ronacher, who led a team of international Python enthusiasts called Poocco. Flask is based on the Werkzeg WSGI toolkit and the Jinja2 template engine. Both are Pocco projects. it’s a Python module that lets you develop web applications easily. It’s has a small and easy-to-extend core: it’s a microframework that doesn’t include an ORM (Object Relational Manager) or such features.

4.2 MACHINE LEARNING

In the real world, we are surrounded by humans who can learn everything from their experiences with learning capability, and we have computers or machines which work on our instructions. But can a machine also learn from experiences or past data like a human does? So here comes the role of Machine Learning.

The term Machine Learning was coined by Arthur Samuel in 1959, an American pioneer in the field of computer gaming and artificial intelligence and stated that “it gives computers the ability to learn without being explicitly programmed”. And in 1997, Tom Mitchell gave a “well-posed” mathematical and relational definition that “A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.

Machine Learning (ML) is the scientific study of algorithms and statistical models that computer systems use to perform a specific task without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of artificial intelligence. From the figure 4.2.1, Machine learning algorithms build a mathematical model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to perform the task. Machine learning algorithms are used in a wide variety of applications, such as email filtering and computer vision, where it is difficult or infeasible to develop a conventional algorithm for effectively performing the task. Machine learning is closely related to computational statistics, which focuses on making predictions using computers.

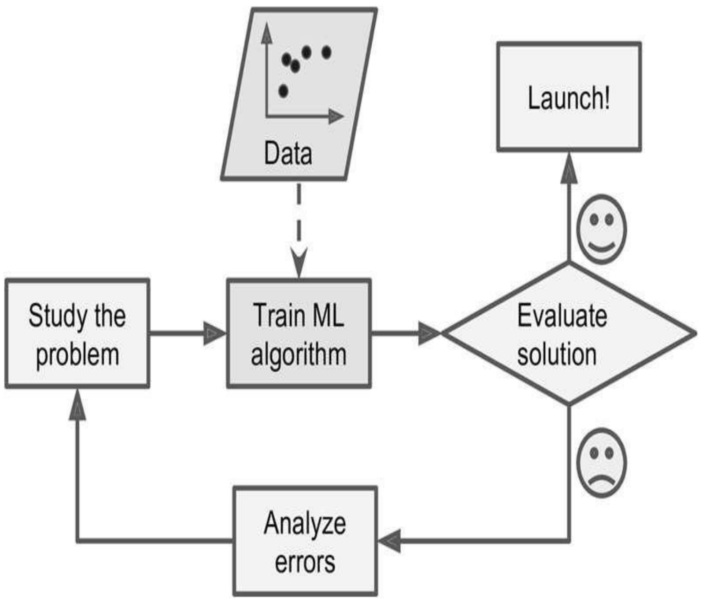


Fig 4.2.1 Workflow of Machine Learning

TYPES OF MACHINE LEARNING

Machine Learning has found its applications in almost every business sector. There are several algorithms used in machine learning that help you build complex models. Each of these algorithms in machine learning can be classified into a certain category. The different types machine learning algorithms.

- Supervised Machine Learning
- Unsupervised Machine Learning
- Reinforcement Learning

Supervised Machine Learning

Supervised learning is a type of machine learning that uses labeled data to train machine learning models. In labeled data, the output is already known. The model just needs to map the inputs to the respective outputs. Supervised learning algorithms take labeled inputs and map them to the known outputs, which means you already know the target variable. Now, let's focus on the training process for the supervised learning method. Supervised Learning methods need external supervision to train machine learning models. Hence, the name supervised. They need guidance and additional information to return the desired result. In the real-world, supervised learning can be used for Risk Assessment, Image classification, Fraud Detection, spam filtering, etc. Supervised learning can be further divided into two types of problems which are regression and classification.

Regression

Regression algorithms are used if there is a relationship between the input variable and the output variable. It is used for the prediction of continuous variables, such as Weather forecasting, Market Trends, etc. Below are some popular Regression algorithms which come under supervised learning.

Classification

The Classification algorithm is a Supervised Learning technique that is used to identify the category of new observations on the basis of training data. From the figure 4.2.2, In Classification, a program learns from the given dataset or observations and then classifies new observation into a number of classes or groups. Such as**,** Yes or No, 0 or 1, Spam or Not Spam, cat or dog, etc. Classes can be called as targets/labels or categories. The main goal of the Classification algorithm is to identify the category of a given dataset, and these algorithms are mainly used to predict the output for the categorical data. The algorithm which implements the classification on a dataset is known as a classifier. There are two types of Classifications. In this project we use Decision Tree Classification algorithm.

- **Binary Classifier:** If the classification problem has only two possible outcomes, then it is called as Binary Classifier. Examples: YES or NO, MALE or FEMALE, SPAM or NOT SPAM, CAT or DOG, etc.
- **Multi-class Classifier:** If a classification problem has more than two outcomes, then it is called as Multi-class Classifier. Example: Classifications of types of crops, Classification of types of music.

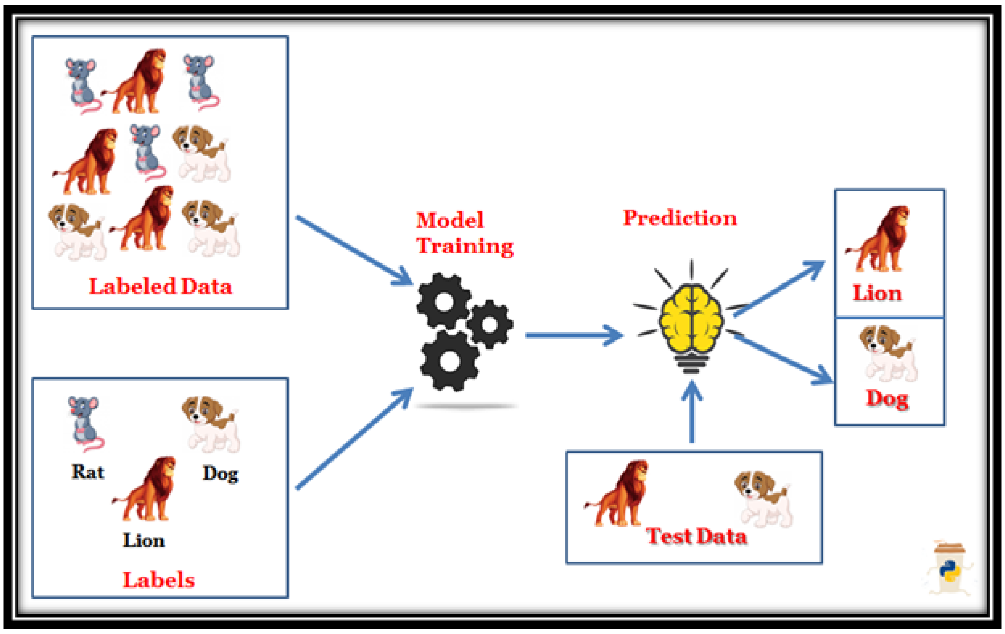


Fig 4.2.2 Classification Algorithm

In this project we are using binary classifier, which is whether the bank will go BANKRUPT or NOT BANKRUPT in the following year.

Design analysis involves how the system is being performed?, What are the specific requirements for the designing of system?, what all problems exist in the present system? What must be done to solve the problem in a efficient way?, module organization, requirement analysis and feasibility study.

4.3 STEPS INVOLVED IN PROJECT IMPLEMENTATION

The figure 4.3.1 shows the various levels involved in the project implementation.

1. Importing the python libraries
2. Loading the dataset
3. Pre-processing the data
4. Train and fit the data to the model
5. Evaluate the result
6. Apply Principal Component Analysis (PCA)
7. Results comparison
8. Connect to the web Page

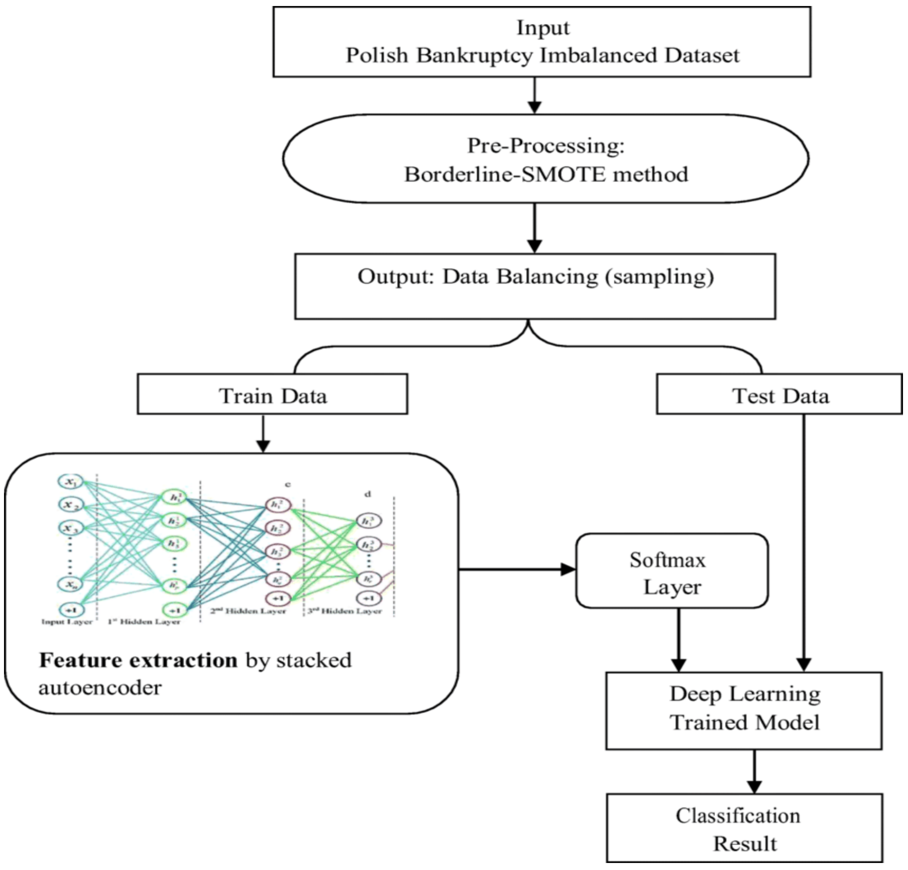

Input
Polish Bankruptcy Imbalanced Dataset
Pre-Processing:
Borderline-SMOTE method
Output: Data Balancing (sampling)
Train Data
Test Data
Feature extraction by stacked autoencoder
Softmax Layer
Deep Learning Trained Model
Classification Result

Fig 4.3.1 Workflow of Implementation

IMPORTING THE PYTHON LIBRARIES

Here we can import the all required libraries into the environment work space.

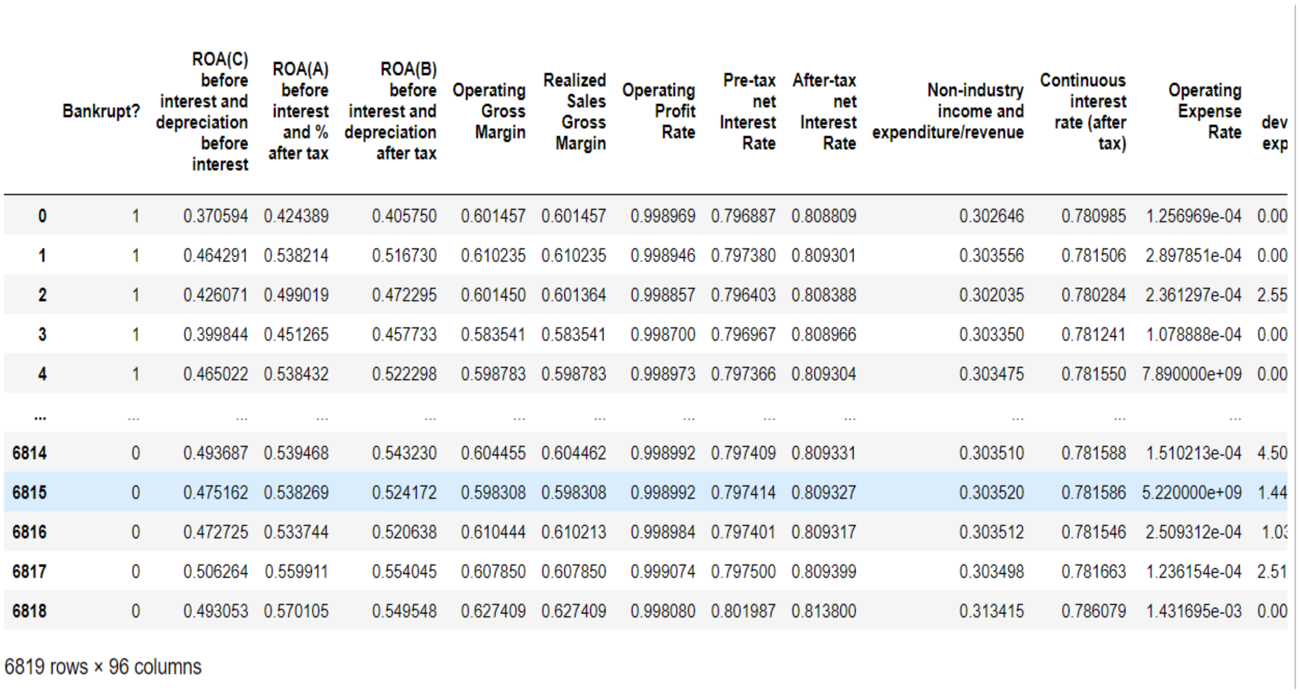
```
import numpy as np

import pandas as pd

pd.set_option('display.max_columns', None)

import plotly.express as px

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import GradientBoostingClassifier

from sklearn.decomposition import PCA

import warnings

warnings.filterwarnings(action='ignore')
```

LOADING THE DATASET

The data were collected from the Taiwan Economic Journal for the years 1999 to 2009. Company bankruptcy was defined based on the business regulations of the Taiwan Stock Exchange.

From the figure 4.3.2 shows, the dataset which contains 6819 entries and 96 columns. All the features related to different areas in the company.

```
data = pd.read_csv('data.csv')

datapca = data.copy()

data
```



	Bankrupt?	ROA(C) before interest and depreciation before interest	ROA(A) before interest and % after tax	ROA(B) before interest and depreciation after tax	Operating Gross Margin	Realized Sales Gross Margin	Operating Profit Rate	Pre-tax net Interest Rate	After-tax net Interest Rate	Non-industry income and expenditure/revenue	Continuous interest rate (after tax)	Operating Expense Rate	dev exp
0	1	0.370594	0.424389	0.405750	0.601457	0.601457	0.998969	0.796887	0.808809	0.302646	0.780985	1.256969e-04	0.00
1	1	0.464291	0.538214	0.516730	0.610235	0.610235	0.998946	0.797380	0.809301	0.303556	0.781506	2.897851e-04	0.00
2	1	0.426071	0.499019	0.472295	0.601450	0.601364	0.998857	0.796403	0.808388	0.302035	0.780284	2.361297e-04	2.55
3	1	0.399844	0.451265	0.457733	0.583541	0.583541	0.998700	0.796967	0.808966	0.303350	0.781241	1.078888e-04	0.00
4	1	0.465022	0.538432	0.522298	0.598783	0.598783	0.998973	0.797366	0.809304	0.303475	0.781550	7.890000e+09	0.00
...	...	...	...	...	...	...	...	...	...	...	...	...	
6814	0	0.493687	0.539468	0.543230	0.604455	0.604462	0.998992	0.797409	0.809331	0.303510	0.781588	1.510213e-04	4.50
6815	0	0.475162	0.538269	0.524172	0.598308	0.598308	0.998992	0.797414	0.809327	0.303520	0.781586	5.220000e+09	1.44
6816	0	0.472725	0.533744	0.520638	0.610444	0.610213	0.998984	0.797401	0.809317	0.303512	0.781546	2.509312e-04	1.0
6817	0	0.506264	0.559911	0.554045	0.607850	0.607850	0.999074	0.797500	0.809399	0.303498	0.781663	1.236154e-04	2.51
6818	0	0.493053	0.570105	0.549548	0.627409	0.627409	0.998080	0.801987	0.813800	0.313415	0.786079	1.431695e-03	0.00

6819 rows × 96 columns

Fig 4.3.2 Bankruptcy Dataset

All the parameters of the dataset

0 Bankrupt?
1 ROA(C) before interest and depreciation before interest
2 ROA(A) before interest and % after tax
3 ROA(B) before interest and depreciation after tax
4 Operating Gross Margin
5 Realized Sales Gross Margin
6 Operating Profit Rate
7 Pre-tax net Interest Rate
8 After-tax net Interest Rate
9 Non-industry income and expenditure/revenue
10 Continuous interest rate (after tax)
11 Operating Expense Rate
12 Research and development expense rate
13 Cash flow rate
14 Interest-bearing debt interest rate
15 Tax rate (A)

16 Net Value Per Share (B)
17 Net Value Per Share (A)
18 Net Value Per Share (C)
19 Persistent EPS in the Last Four Seasons
20 Cash Flow Per Share
21 Revenue Per Share (Yuan ¥)
22 Operating Profit Per Share (Yuan ¥)
23 Per Share Net profit before tax (Yuan ¥)
24 Realized Sales Gross Profit Growth Rate
25 Operating Profit Growth Rate
26 After-tax Net Profit Growth Rate
27 Regular Net Profit Growth Rate
28 Continuous Net Profit Growth Rate
29 Total Asset Growth Rate
30 Net Value Growth Rate
31 Total Asset Return Growth Rate Ratio
32 Cash Reinvestment %
33 Current Ratio
34 Quick Ratio
35 Interest Expense Ratio
36 Total debt/Total net worth
37 Debt ratio %
38 Net worth/Assets
39 Long-term fund suitability ratio (A)
40 Borrowing dependency
41 Contingent liabilities/Net worth
42 Operating profit/Paid-in capital
43 Net profit before tax/Paid-in capital
44 Inventory and accounts receivable/Net value
45 Total Asset Turnover
46 Accounts Receivable Turnover
47 Average Collection Days
48 Inventory Turnover Rate (times)
49 Fixed Assets Turnover Frequency
50 Net Worth Turnover Rate (times)
51 Revenue per person
52 Operating profit per person
53 Allocation rate per person
54 Working Capital to Total Assets
55 Quick Assets/Total Assets
56 Current Assets/Total Assets
57 Cash/Total Assets
58 Quick Assets/Current Liability
59 Cash/Current Liability
60 Current Liability to Assets
61 Operating Funds to Liability
62 Inventory/Working Capital
63 Inventory/Current Liability
64 Current Liabilities/Liability

65	Working Capital/Equity
66	Current Liabilities/Equity
67	Long-term Liability to Current Assets
68	Retained Earnings to Total Assets
69	Total income/Total expense
70	Total expense/Assets
71	Current Asset Turnover Rate
72	Quick Asset Turnover Rate
73	Working capitcal Turnover Rate
74	Cash Turnover Rate
75	Cash Flow to Sales
76	Fixed Assets to Assets
77	Current Liability to Liability
78	Current Liability to Equity
79	Equity to Long-term Liability
80	Cash Flow to Total Assets
81	Cash Flow to Liability
82	CFO to Assets
83	Cash Flow to Equity
84	Current Liability to Current Assets
85	Liability-Assets Flag
86	Net Income to Total Assets
87	Total assets to GNP price
88	No-credit Interval
89	Gross Profit to Sales
90	Net Income to Stockholder's Equity
91	Liability to Equity
92	Degree of Financial Leverage (DFL)
93	Interest Coverage Ratio (Interest expense to EBIT)
94	Net Income Flag
95	Equity to Liability

The dataset contains more number of features. If we apply frontend technology to the all features it’s difficult to enter the all records and it’s not a efficient way to implement the model by considering the all features. So here we need to drop the some columns to better understanding and also achieve better performance.

Here I drop maximum of 85 columns, I used only the features which play major role for finding the company bankruptcy prediction.

Source Code:

```
data.drop(data.columns[[1,2,3,5,6,7,8,9,10,11,12,14,16,17,18,19,20,21,22,23,24,25,26,27,28,
29,30,31,32,33,34,35,36,37,39,40,41,42,43,44,46,47,48,49,50,51,52,53,54,55,56,58,60,61,62,
63,64,65,66,67,68,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,90,91,92,93,95]],
axis=1, inplace=True)
```

```
 0  Bankrupt?
 1   Operating Gross Margin
 2   Cash flow rate
3   Tax rate (A)
4   Net worth/Assets
5   Total Asset Turnover
6   Cash/Total Assets
7   Cash/Current Liability
8   Total income/Total expense
9   Total expense/Assets
10   Gross Profit to Sales
```

PREPROCESSING THE DATA

Data pre-processing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model. When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So for this, we use data pre-processing task.

A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data pre-processing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model. There are many important steps in data pre-processing, such as data cleaning, data transformation, and feature selection. Data cleaning and transformation are methods used to remove outliers and standardize the data so that they take a form that can be easily used to create a model.

Here I split the data into both train data and test data. In this splitting I used 70% of data to the training and 30% of the data to the test data using train_test_split.

```
def preprocess_inputs(df):
    df = df.copy()
    df = df.drop(df.columns[[11]], axis=1)
    y = df['Bankrupt?']
    X = df.drop('Bankrupt?', axis=1)
    X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, shuffle=True, random_state=1)
    return X_train, X_test, y_train, y_test
X_train, X_test, y_train, y_test = preprocess_inputs(data)
```

4.4 CLASSIFICATION OF MODELS

In this project we use Gradient Boosting Classifier and for supporting we also add Decision Tree Classifier.

GRADIENT BOOSTING

What is Boosting?

Boosting is a technique to combine weak learners and convert them into strong ones with the help of Machine Learning algorithms. It uses ensemble learning to boost the accuracy of a model. Ensemble learning is a technique to improve the accuracy of Machine Learning models. There are two types of ensemble learning:

In Machine Learning, we use gradient boosting to solve classification and regression problems. It is a sequential ensemble learning technique where the performance of the model improves over iterations. From the figure 4.4.1, this method creates the model in a stage-wise fashion. It infers the model by enabling the optimization of an absolute differentiable loss function. As we add each weak learner, a new model is created that gives a more precise estimation of the response variable.

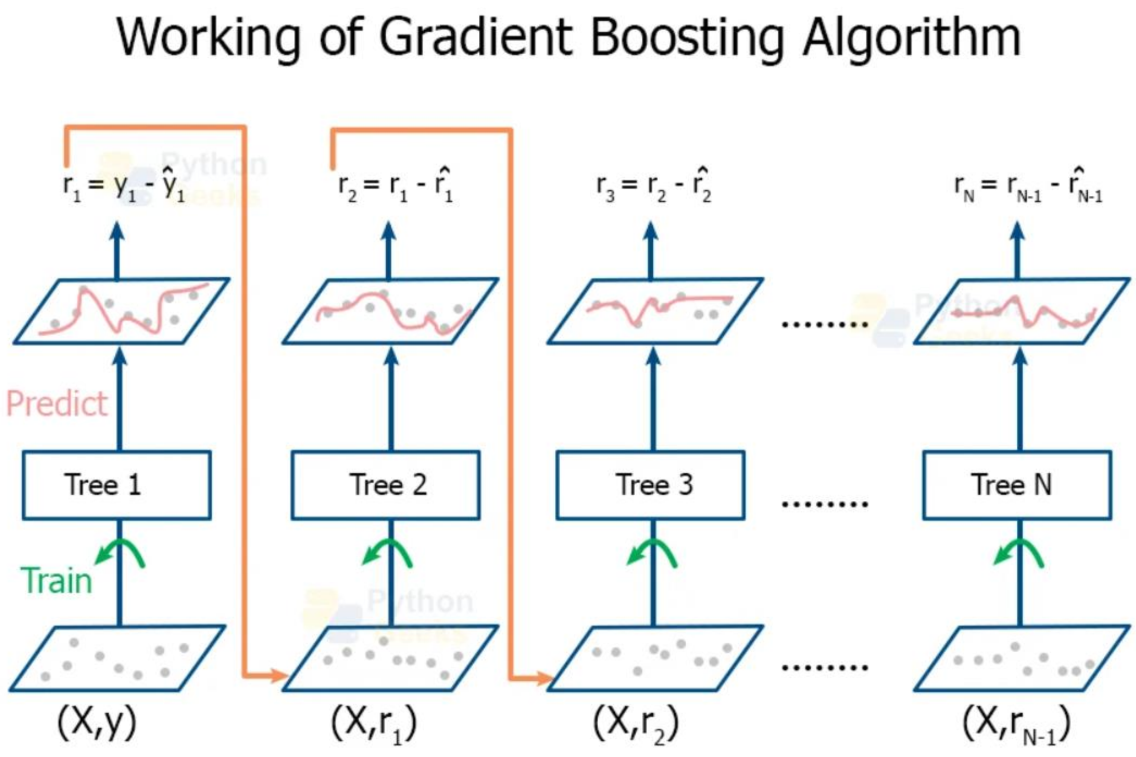


Fig 4.4.1 Work flow of Gradient Boosting

The gradient boosting algorithm requires the below components to function.

1. **Loss function**: To reduce errors in prediction, we need to optimize the loss function. Unlike in AdaBoost, the incorrect result is not given a higher weightage in gradient boosting. It tries to reduce the loss function by averaging the outputs from weak learners.

2. **Weak learner**: In gradient boosting, we require weak learners to make predictions. To get real values as output, we use regression trees. To get the most suitable split point, we create trees in a greedy manner, due to this the model overfits the dataset.

3. **Additive model**: In gradient boosting, we try to reduce the loss by adding decision trees. Also, we can minimize the error rate by cutting down the parameters. So, in this case, we design the model in such a way that the addition of a tree does not change the existing tree.

Finally, we update the weights to minimize the error that is being calculated. Gradient boosting is a highly robust technique for developing predictive models. It applies to several risk functions and optimizes the accuracy of the model's prediction. It also resolves multicollinearity problems where the correlations among the predictor variables are high.

```
modelg = GradientBoostingClassifier()

modelg.fit(X_train, y_train)

modelg_pred = modelg.predict(X_test)

print("Gradient Boosting trained.")
```

```
Gradient Boosting trained.
```

```
result = modelg.score(X_test, y_test)

print("Gradient Boosting" + ": {:.2f}%".format(result * 100))
```

```
Gradient Boosting: 96.48%
```

DECISION TREE

- Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

- In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, where as Leaf nodes are the output of those decisions and do not contain any further branches.

- The decisions or the test are performed on the basis of features of the given dataset.

- It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.

- From the figure 4.4.2, it is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.

- In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm.

- A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into sub trees.

- Below diagram explains the general structure of a decision tree.

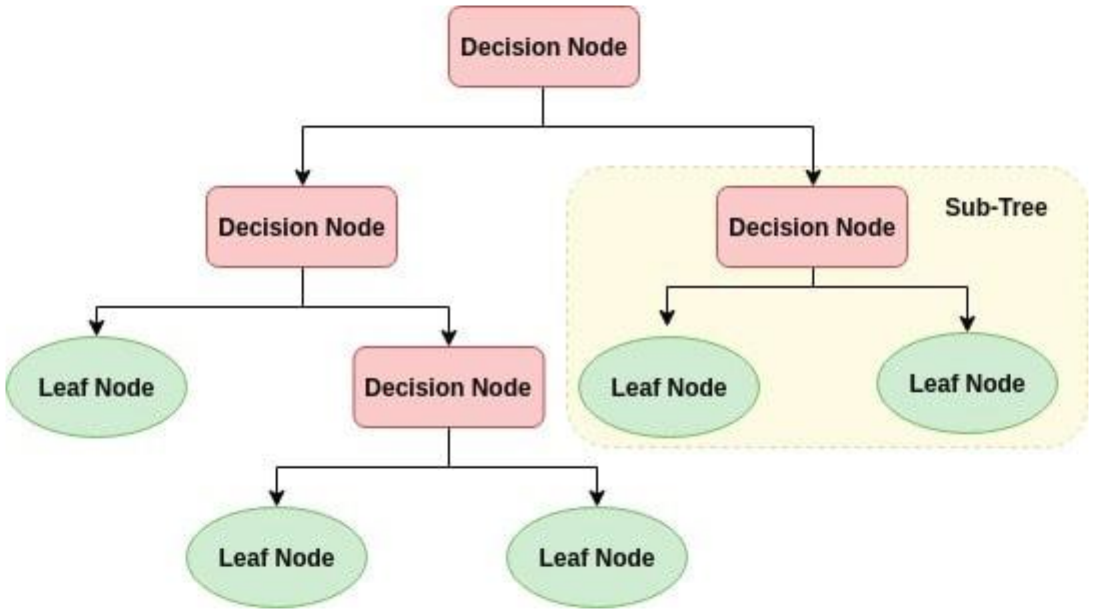


Fig 4.4.2 Structure of Decision Tree

There are various algorithms in Machine learning, so choosing the best algorithm for the given dataset and problem is the main point to remember while creating a machine learning model. Below are the two reasons for using the Decision tree. Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand. λ The logic behind the decision tree can be easily understood because it shows a tree-like structure.

modeld = DecisionTreeClassifier()

modeld.fit(X_train, y_train)

```
model_pred = modeld.predict(X_test)

print("Decision Tree trained.")
```

```
Decision Tree trained.
```

```
result = modeld.score(X_test, y_test)

print("Decision Tree" + ": {:.2f}%".format(result * 100))
```

```
Decision Tree: 94.77%
```

EVALUATE THE RESULTS

Gradient Boosting, which gives accuracy score 96.48%. Gradient Boosting model is perform well on both train and test data.

Decision Tree, which gives accuracy score 94.77%. Decision Tree model is also perform well on both train and test data.

Both models are gives accurate results and perform well but Gradient Boosting performance is slightly more than the Decision Tree. So we classify that the Gradient Boosting model is perform better than the Decision Tree.

4.5 DIMENSIONALITY REDUCTION

In machine learning classification problems, there are often too many factors on the basis of which the final classification is done. These factors are basically variables called features. The higher the number of features, the harder it gets to visualize the training set and then work on it. Sometimes, most of these features are correlated, and hence redundant. This is where dimensionality reduction algorithms come into play. Dimensionality reduction is the process of reducing the number of random variables under consideration, by obtaining a set of principal variables. It can be divided into feature selection and feature extraction.

Feature selection, Feature selection is a means of selecting the input data set's optimal, relevant features and removing irrelevant features.

Feature extraction, this method transforms the space containing too many dimensions into a space with fewer dimensions. This process is useful for keeping the whole information while using fewer resources during information processing.

PRINCIPAL COMPONENT ANALYSIS

The Principal Components are a straight line that captures most of the variance of the data. They have a direction and magnitude. Principal components are orthogonal projections (perpendicular) of data onto lower-dimensional space. Now that you have understood the basics of PCA, let’s look at the next topic on PCA in Machine Learning.

The Principal Component Analysis is a popular unsupervised learning technique for reducing the dimensionality of data. It increases interpretability yet, at the same time, it minimizes information loss. It helps to find the most significant features in a dataset and makes the data easy for plotting in 2D and 3D. PCA helps in finding a sequence of linear combinations of variables.

The algorithm can be used on its own, or it can serve as a data cleaning or data pre-processing technique used before another machine learning algorithm. On its own, PCA is used across a variety of use cases.

1. Visualize multidimensional data
2. Compress information.
3. Simplify complex business decisions.
4. Clarify convoluted scientific processes.

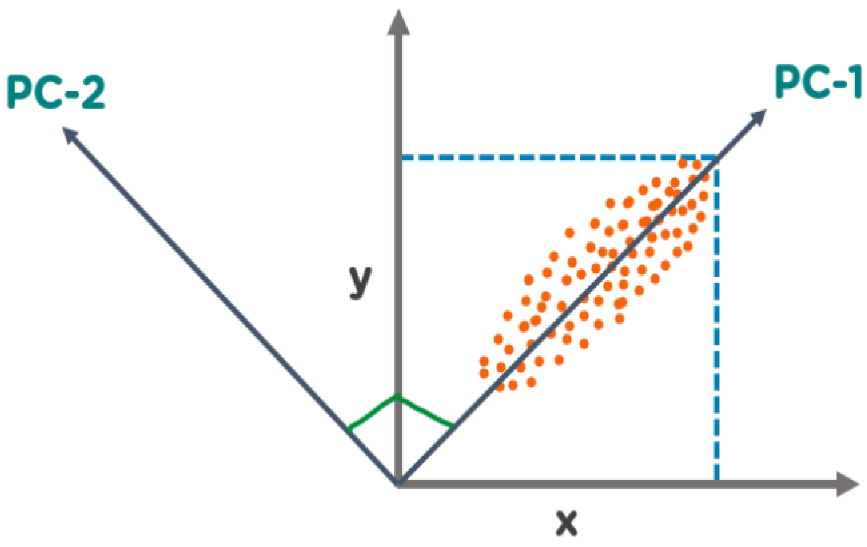


Fig 4.5.1 Example of PCA

From the figure 4.5.1, we have several points plotted on a 2-D plane. There are two principal components. PC1 is the primary principal component that explains the maximum variance in the data. PC2 is another principal component that is orthogonal to PC1.

STANDARD SCALAER

StandardScaler follows Standard Normal Distribution (SND). Therefore, it makes *mean = 0* and scales the data to unit variance. In the presence of outliers, StandardScaler does not guarantee balanced feature scales, due to the influence of the outliers while computing the empirical mean and standard deviation. This leads to the shrinkage in the range of the feature values.
By using RobustScaler(), we can remove the outliers and then use either StandardScaler or MinMaxScaler for preprocessing the dataset.

Standardize the data before performing PCA. This will ensure that each feature has a mean = 0 and variance = 1.

$$Z = \frac{x - \mu}{\sigma}$$

scaler = StandardScaler()

```
scaler.fit(X_train)

X_train=pd.DataFrame(scaler.transform(X_train),index=X_train.index,columns=X_train.col
umns)

X_test=pd.DataFrame(scaler.transform(X_test),index=X_test.index,columns=X_test.column)
```

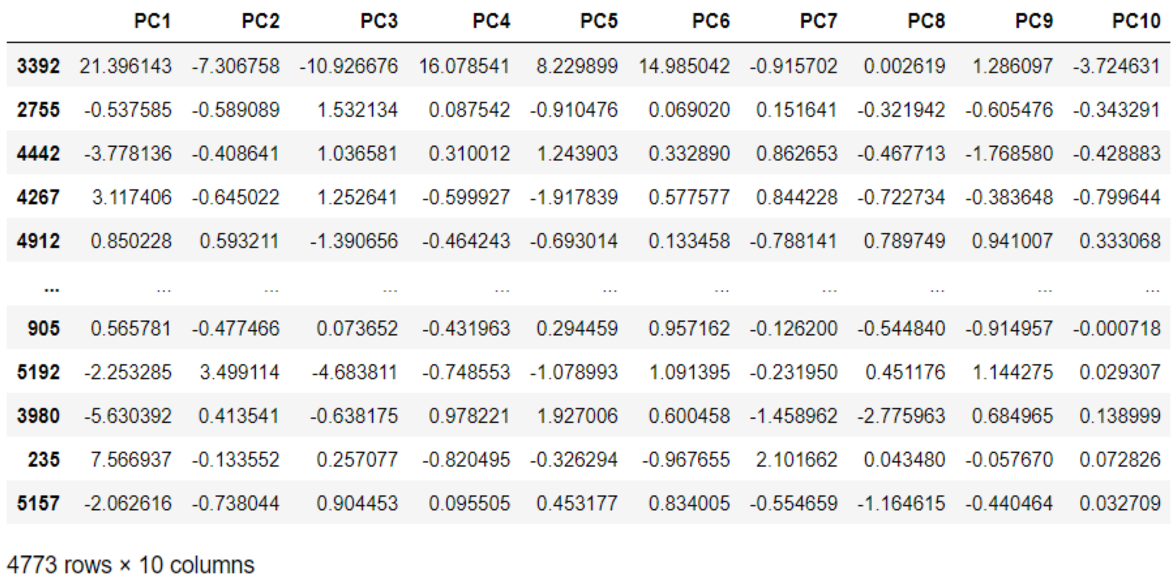
Here we apply the Standard Scaler to both train and test data to make magnitude and unit in the same range. For PCA, we are using the dataset, which contains 6819 entries and 96 columns.

When we build model for this dataset, means that applying both Gradient Boosting and Decision Tree Classifiers.

```
original_models = {

    "                         Decision Tree":DecisionTreeClassifier(),

    "                      Gradient Boosting":GradientBoostingClassifier()

}

for name,model in original_models.items():

    model.fit(X_train,y_train)

original_results = []

for name, model in original_models.items():

    result = model.score(X_test,y_test)

    original_results.append(result)

    print(name + ":{:.2f}%".format(result * 100))
Decision Tree: 95.31%

Gradient Boosting: 96.14%
```

we apply PCA for same dataset, so instead of using 96 features we use only 10 components. From the figure 4.5.2, 10 components, which have high explained variance ratio. Explained variance is a statistical measure of how much variation in a dataset can be attributed to each of the principal components (eigenvectors) generated by a PCA. In very basic terms, it refers to the amount of variability in a data set that can be attributed to each individual principal component.

```
n_components = 10

pca = PCA(n_components=n_components)

pca.fit(X_train)

X_train_reduced=pd.DataFrame(pca.transform(X_train),index=X_train.index,columns=["PC" + str(i) for i in range(1, n_components + 1)])

X_test_reduced = pd.DataFrame(pca.transform(X_test), index=X_test.index, columns=["PC" + str(i) for i in range(1, n_components + 1)])
```



	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10
3392	21.396143	-7.306758	-10.926676	16.078541	8.229899	14.985042	-0.915702	0.002619	1.286097	-3.724631
2755	-0.537585	-0.589089	1.532134	0.087542	-0.910476	0.069020	0.151641	-0.321942	-0.605476	-0.343291
4442	-3.778136	-0.408641	1.036581	0.310012	1.243903	0.332890	0.862653	-0.467713	-1.768580	-0.428883
4267	3.117406	-0.645022	1.252641	-0.599927	-1.917839	0.577577	0.844228	-0.722734	-0.383648	-0.799644
4912	0.850228	0.593211	-1.390656	-0.464243	-0.693014	0.133458	-0.788141	0.789749	0.941007	0.333068
...	...	...	...	...	...	...	...	...	...	...
905	0.565781	-0.477466	0.073652	-0.431963	0.294459	0.957162	-0.126200	-0.544840	-0.914957	-0.000718
5192	-2.253285	3.499114	-4.683811	-0.748553	-1.078993	1.091395	-0.231950	0.451176	1.144275	0.029307
3980	-5.630392	0.413541	-0.638175	0.978221	1.927006	0.600458	-1.458962	-2.775963	0.684965	0.138999
235	7.566937	-0.133552	0.257077	-0.820495	-0.326294	-0.967655	2.101662	0.043480	-0.057670	0.072826
5157	-2.062616	-0.738044	0.904453	0.095505	0.453177	0.834005	-0.554659	-1.164615	-0.440464	0.032709

4773 rows × 10 columns

Fig 4.5.2 PCA Components Dataset

4.6 DATA VISUALIZATION

Data visualization is the process of transforming large data sets into a statistical and graphical representation. It is an essential task of data science and knowledge discovery techniques to make data less confusing and more accessible.

Visualization takes a huge complex amount of data to represent charts or graphs for quick information to absorb and better understandability. It avoids hesitation on large data sets table to hold audience interest longer.

From the figure 4.6.1, we represent the graph for 10 components of PCA, which have maximum explained variance ratio.

```
pca.explained_variance_ratio_

array([0.13996931, 0.07616369, 0.05680625, 0.05395067, 0.04987216, 0.03484072,
0.03239139, 0.02416938, 0.02302857, 0.02222451])
```

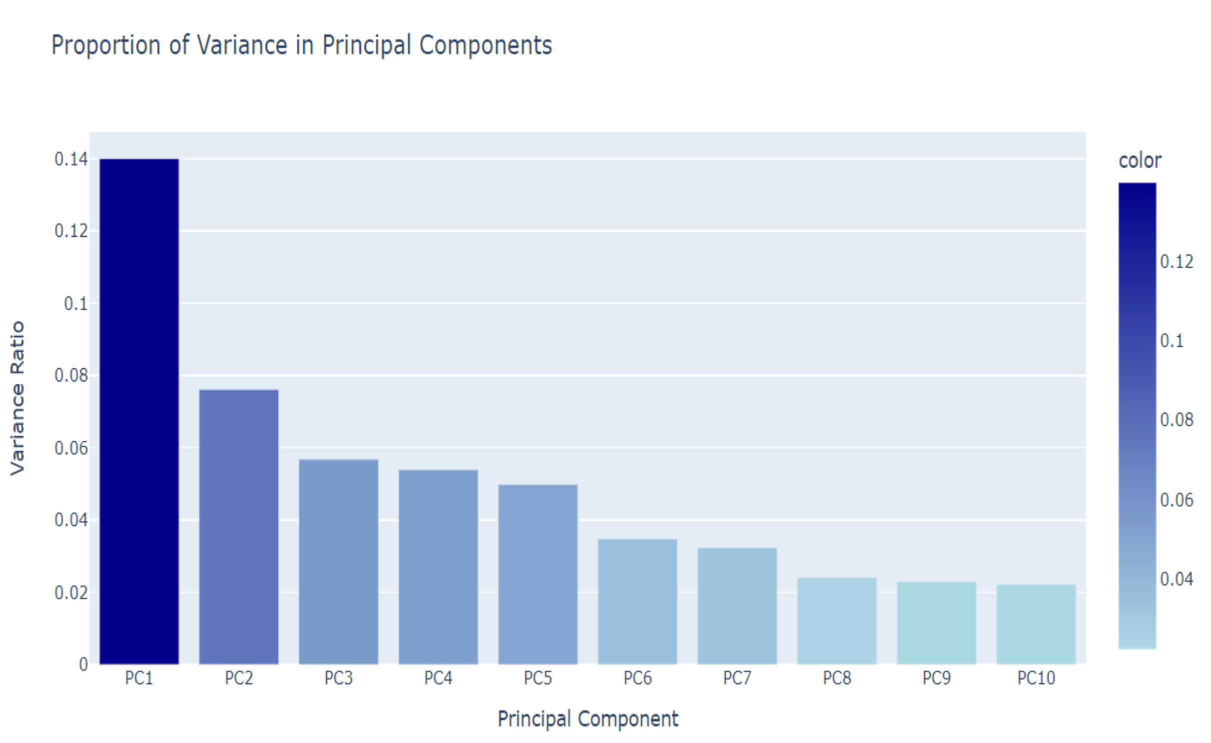


Fig 4.6.1 Principal Component Analysis

After plotting the graph, we implement the Decision Tree and Gradient Boosting for PCA data. We find the accuracy results for both models.

```
reduced_results = []

for name, model in reduced_models.items():

    result = model.score(X_test_reduced, y_test)

    reduced_results.append(result)

    print(name + ": {:.2f}%".format(result * 100))
```

```
Decision Tree: 94.67%

Gradient Boosting: 96.29%
```

The Decision Tree, which gives accuracy result as 94.67% and Gradient Boosting model gives accuracy result as 96.29%.

Now plotting graph between two situations, which are before PCA and after PCA. We compare the both accuracy results.

The Decision Tree, which has more accuracy 95.31% before PCA and accuracy after PCA has 94.67%.

The Gradient Boosting, which has accuracy 96.14% before PCA and accuracy after PCA has 96.29%.

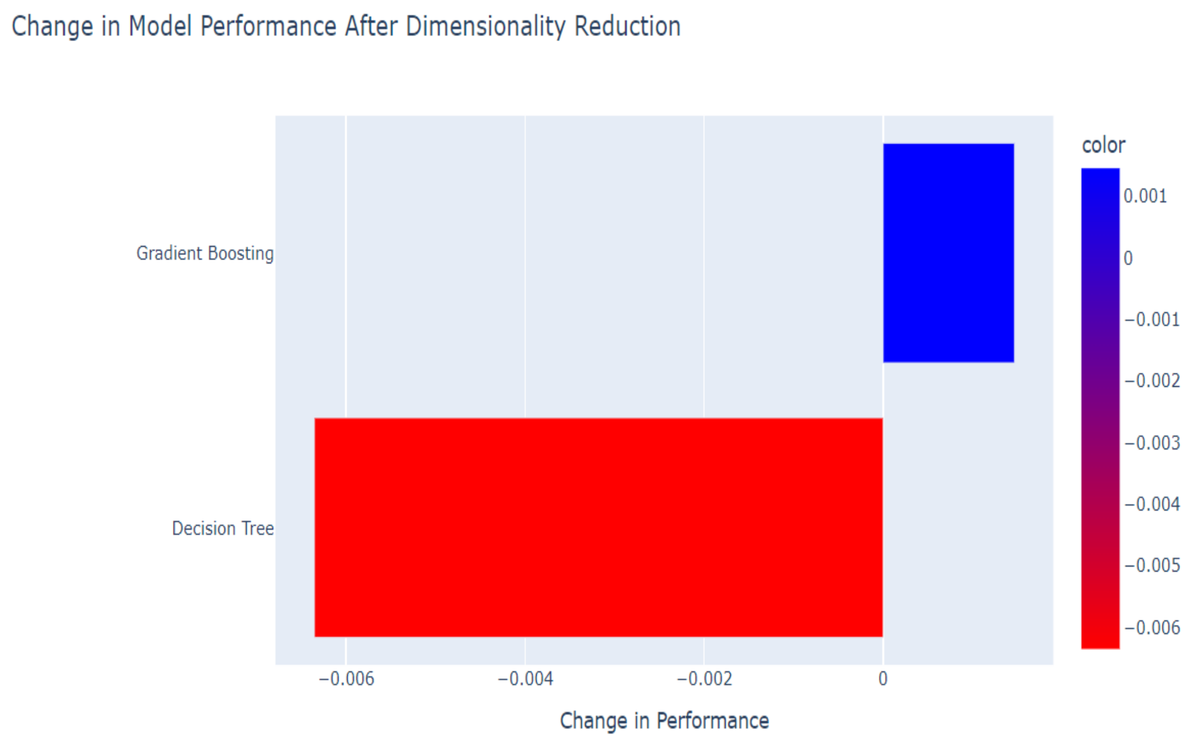


Fig 4.6.2 Variation in performance of model

From the figure 4.6.2, the change in model performance Decision Tree after Dimensionality Reduction is -0.00635. Here we get negative accuracy for Decision Tree after PCA. Every model will get high accuracy after PCA, but here we get negative performance.

The change in model performance Gradient Boosting after Dimensionality Reduction is 0.00146. Here we get positive accuracy for Gradient Boosting after PCA and which has more than before PCA. So we conclude that Gradient Boosting, which is the best accurate model than the Decision Tree.

4.7 SOURCE CODE

Here the source code of entire project.

```
#IMPORTING LIBRARIES

import numpy as np

import pandas as pd
```

```
pd.set_option('display.max_columns', None)

import plotly.express as px

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import GradientBoostingClassifier

from sklearn.decomposition import PCA

import warnings

warnings.filterwarnings(action='ignore')

#LOADING DATASET

data = pd.read_csv('data.csv')

datapca = data.copy()

data

print(data.isna().sum().sum())

print(np.isnan(data).sum().sum())

print(data.isnull().sum().sum())

data.describe()

data.info()

{column: len(data[column].unique()) for column in data.columns}

data.drop(data.columns[[1,2,3,5,6,7,8,9,10,11,12,14,16,17,18,19,20,21,22,23,24,25,26,27,28,
29,30,31,32,33,34,35,36,37,39,40,41,42,43,44,46,47,48,49,50,51,52,53,54,55,56,58,60,61,62,
63,64,65,66,67,68,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,90,91,92,93,95]],
axis=1, inplace=True)

data

data.info()

#PREPROCESSING DATASET

def preprocess_inputs(df):

    df = df.copy()

    df = df.drop(df.columns[[11]], axis=1)

    y = df['Bankrupt?']

    X = df.drop('Bankrupt?', axis=1)

    X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, shuffle=True,
random_state=1)

    return X_train, X_test, y_train, y_test

X_train, X_test, y_train, y_test = preprocess_inputs(data)

X_train, X_test, y_train, y_test = preprocess_inputs(data)

X_train

y_train

#CLASSIFICATION OF MODELS

#Gradient Boosting

modelg = GradientBoostingClassifier()

modelg.fit(X_train, y_train)

modelg_pred = modelg.predict(X_test)
```

```
print("Gradient Boosting trained.")

Gradient Boosting trained.
result = modelg.score(X_test, y_test)

print("Gradient Boosting" + ": {:.2f}%".format(result * 100))

Gradient Boosting: 96.48%



#Decision Tree

modeld = DecisionTreeClassifier()

modeld.fit(X_train, y_train)

model_pred = modeld.predict(X_test)

print("Decision Tree trained.")

Decision Tree trained.

result = modeld.score(X_test, y_test)

print("Decision Tree" + ": {:.2f}%".format(result * 100))

Decision Tree: 94.53%

y_test.value_counts() / len(y_test)

0    0.965298
1    0.034702
Name: Bankrupt?, dtype: float64
#PRINCIPAL COMPONENT ANALYSIS (PCA)

datapca

def preprocess_inputs(df):

    df = df.copy()

    #Drop single-value column

    df = df.drop(df.columns[[11]], axis=1)
    #Split df into X and Y

    y = df['Bankrupt?']

    X = df.drop('Bankrupt?', axis=1)

    # Train-test split

    X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, shuffle=True,
random_state=1)

    # Scale X

    scaler = StandardScaler()

    scaler.fit(X_train)

    X_train = pd.DataFrame(scaler.transform(X_train), index=X_train.index,
columns=X_train.columns)

    X_test = pd.DataFrame(scaler.transform(X_test), index=X_test.index,
columns=X_test.columns)

    return X_train, X_test, y_train, y_test
    return df

X_train, X_test, y_train, y_test = preprocess_inputs(datapca)

original_models = {

    "                         Decision Tree":DecisionTreeClassifier(),

    "                      Gradient Boosting":GradientBoostingClassifier()

}

for name,model in original_models.items():

    model.fit(X_train,y_train)

original_results = []

for name, model in original_models.items():

    result = model.score(X_test,y_test)
    original_results.append(result)

    print(name + ":{:.2f}%".format(result * 100))
```

```
Decision Tree:95.31%

Gradient Boosting:96.14%
```

```
n_components = 10

pca = PCA(n_components=n_components)

pca.fit(X_train)

X_train_reduced = pd.DataFrame(pca.transform(X_train), index=X_train.index, columns=["PC" + str(i) for i in range(1, n_components + 1)])

X_test_reduced = pd.DataFrame(pca.transform(X_test), index=X_test.index, columns=["PC" + str(i) for i in range(1, n_components + 1)])

X_train_reduced

X_train_reduced.describe()

pca.explained_variance_ratio_
```

```
array([0.13996931, 0.07616369, 0.05680625, 0.05395067, 0.04987216, 0.03484072,
0.03239139, 0.02416938, 0.02302857, 0.02222451])
```

```
fig = px.bar(

    x=["PC" + str(i) for i in range(1, n_components + 1)],

    y=pca.explained_variance_ratio_,

    labels={'x': "Principal Component", 'y': "Variance Ratio"},

    color=pca.explained_variance_ratio_,
    color_continuous_scale=[(0, 'lightblue'), (1, 'darkblue')],

    title="Proportion of Variance in Principal Components"
)
```

```
fig.show()

reduced_models = {
    "                    Decision Tree": DecisionTreeClassifier(),

    "                 Gradient Boosting": GradientBoostingClassifier()
}

for name, model in reduced_models.items():

    model.fit(X_train_reduced, y_train)

    print(name + " trained.")

Decision Tree trained.

Gradient Boosting trained.

reduced_results = []

for name, model in reduced_models.items():

    result = model.score(X_test_reduced, y_test)

    reduced_results.append(result)

    print(name + ": {:.2f}%".format(result * 100))

Decision Tree: 94.67%

Gradient Boosting: 96.29%

fig = px.bar(

    x=np.subtract(reduced_results, original_results),

    y=original_models.keys(),

    orientation='h',

    labels={'x': "Change in Performance", 'y': "Model"},

    color=np.subtract(reduced_results, original_results),

    color_continuous_scale=[(0, 'red'), (1, 'blue')],

    title="Change in Model Performance After Dimensionality Reduction"
)
fig.show()
```

WEBPAGE CREATION

5. WEBPAGE CREATION

5.1 VISUAL STUDIO CODE

Visual Studio Code is a source-code editor made by Microsoft for windows, Linux and macOS by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax, highlighting, code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional that add additional functionality.

Visual Studio Code is a source-code editor that can be used with a variety of programming languages, including Java, JavaScript, Go, Node.js, Python and C++. It is based on the Electron framework, which is used to develop Node.js Web application that run on the Blink layout engine.

Here we are using frontend technologies like HTML and CSS. We connect our Gradient Boosting model with webpage. The Gradient Boosting model gives accurate results to us.

HTML and CSS, these technologies are very helpful for us while creating webpage. We also implement backend technology to project to make relationship between our Gradient Boosting model and webpage. We are using pickle module for connecting the model with the web page. When we start our model it will automatically generates one raw file in the folder which is related to the web page creation.

HTML

HTML stands for HyperText Markup Language. It is a standard markup language for web page creation. It allows the creation and structure of sections, paragraphs, and links using HTML elements (the building blocks of a web page) such as tags and attributes.

- Web development. Developers use HTML code to design how a browser displays web page elements, such as text, hyperlinks, and media files.

- Internet navigation. Users can easily navigate and insert links between related pages and websites as HTML is heavily used to embed hyperlinks.
- Web documentation. HTML makes it possible to organize and format documents, similarly to Microsoft Word.

CSS

Cascading **S**tyle **S**heets, fondly referred to as CSS, is a simple design language intended to
simplify the process of making web pages presentable.
CSS handles the look and feel part of a web page. Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, layout designs, variations in display for different devices and screen sizes as well as a variety of other effects.
CSS is easy to learn and understand but it provides powerful control over the presentation of an HTML document. Most commonly, CSS is combined with the markup languages HTML or XHTML.

All the code is implemented in the visual studio code, python is used to make relationship between flask and model. Here we can see the code snippets of our web page work.

```
import pickle

pickle_out = open("modelg.pkl","wb")

pickle.dump(modelg, pickle_out)

pickle_out.close()
```

5.2 SOURCE CODE FOR INDEXS.HTML

```
<!DOCTYPE html>
<html>
  <head>
    <title>Company Bankruptcy Prediction</title>
    <style>
      html, body {
      min-height: 100%;
```

```
 padding: 0;
 margin: 0;
 font-family: Roboto, Arial, sans-serif;
 font-size: 14px;
 color: rgb(46, 111, 139);
   }
 h1 {
 margin: 0 0 20px;
 font-weight: 400;
 color: rgb(88, 46, 139);
 }
 label {
 margin: 0 0 0px;
 font-weight: 50;
 color: red;
 font-size: 18px;
 }
 p {
 margin: 0 0 5px;
 }
 .main-block {
 display: flex;
 flex-direction: column;
 justify-content: center;
 align-items: center;
 min-height: 75vh;
 background-image: url("static/img/background image.jpg");
background-size: cover; /* Resize the background image to cover the entire container */

 }
 form {
 padding: 25px;
 margin: 25px;
 width: 50%;
 box-shadow: 0 2px 5px #f5f5f5;
 background: #f5f5f5;
 border: 2px solid rgb(115, 64, 118);
     border-radius: 8px;
     }
 input, textarea {
 width: calc(50% - 18px);
 padding: 8px;
 margin-bottom: 20px;
 border: 2px solid seagreen;
 border-radius: 4px;
 outline: none;
 }
```

```
    input::placeholder {
    color: rgb(168, 171, 171);
    }
    input {
      display: block;
    }
    button {
    width: 50%;
    padding: 10px;
    border: none;
    border-radius: 8px;
    background: seagreen;
    font-size: 16px;
    font-weight: 400;
    color: #fff;
    }
    button:hover {
    background: green;
    }
    @media screen and (max-width: 600px) {
    .main-block {
    flex-direction: row;
    }
    }
    .left-part, form {
    width: 50%;
    }
    .result{
      position: absolute;
      visibility: visible;
      top: 70%;
      right: 25%;
      width: 15%;
      height: 15%;
      border: 3px solid seagreen;
      border-radius: 5px;
      box-shadow: 0 2px 5px #f5f5f5;
      text-align: center;
   }
  </style>
</head>
<body>
  <div class="main-block">
    <div class="left-part">
    </div>
    <center><h1><b>Company Bankruptcy Prediction</b></h1></center>
```

```
        <p>Bankruptcy is a legal proceeding initiated when a person or business is unable to
repay outstanding debts or obligations. </p>
        <p><i>Accurate model Company Bankruptcy Prediction</i></p>
      <form action="{{url_for('predict')}}" method="post">
        <section>
        <p>This Web Application is processed by a Machine Learning Algorithm to predict
probability that whether the company will go bankrupt or not in following years. </p>
        <p><i>Note that this model is 96.48% accurate</i></p>
        <div class="info">
          <p><b>Please Enter values for the following:</b></p><br>
          <label><i>Enter Your OPG Score : </i></label><br><br>
          <input type="number" name="Operating Gross Margin" step="0.00001"
placeholder="OPG" required>
          <label><i> Enter your CFR Rating : </i></label><br><br>
          <input type="number"  name="Cash flow rate" step="0.00001" placeholder="CFR"
required>
          <label><i> Enter Your TRA Rating : </i></label><br><br>
          <input type="number" name="Tax rate (A)" step="0.00001" placeholder="TRA"
required>
          <label><i> Enter Your NWA : </i></label><br><br>
          <input type="number" name="Net worth/Assets" step="0.00001"
placeholder="NWA" required>
          <label><i>Enter TAT : </i></label><br><br>
          <input type="number" name="Total Asset Turnover" step="0.00001"
placeholder="TAT" required>
          <label><i>Enter CTA : </i></label><br><br>
          <input type="number" name="Cash/Total Assets" step="0.00001" placeholder="CTA"
required>
          <label><i>Enter CCL : </i></label><br><br>
          <input type="number" name="Cash/Current Liability" step="0.00001"
placeholder="CCL" required>
          <label><i>Enter TITE : </i></label><br><br>
          <input type="number" name="Total income/Total expense" step="0.00001"
placeholder="TITE" required>
          <label><i>Enter TEA : </i></label><br><br>
          <input type="number" name="Total expense/Assets" step="0.00001"
placeholder="TEA" required>
          <label><i>Enter GPS : </i></label><br><br>
          <input type="number" name="Gross Profit to Sales" step="0.00001"
placeholder="GPS" required>
        </div>
        <center><button type="submit" value="predict">Predict</button></center>
        <div class = "result">
           <h2>Result</h2>
           <p>{{result}}</p>
        </div>
        </section>
```

```
     </form>
        <footer>
           <p><i>Developed By Team</i></p>
    </footer>
  </body>
</html>
```

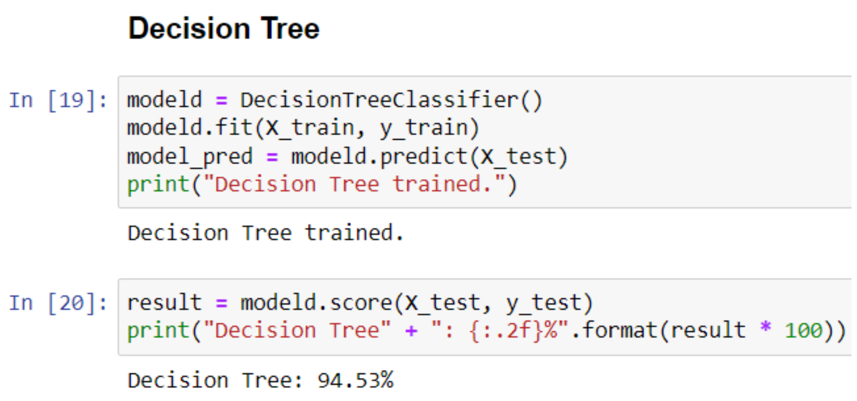
5.3 SOURCE CODE FOR APP.PY

```
from flask import Flask, render_template,request,jsonify
import pickle
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

app  = Flask(__name__)

#Load the pickle model
model=pickle.load(open('modelg.pkl','rb'))

@app.route('/')
def index():
    return render_template('indexs.html')


@app.route('/predict', methods=['GET','post'])
def predict():
    Operating_Gross_Margin = float(request.form['Operating Gross Margin'])
    Cash_flow_rate = float(request.form['Cash flow rate'])
    Tax_rate_A = float(request.form['Tax rate (A)'])
    Net_worth_Assets = float(request.form['Net worth/Assets'])
    Total_Asset_Turnover = float(request.form['Total Asset Turnover'])
    Cash_Total_Assets = float(request.form['Cash/Total Assets'])
    Cash_Current_Liability = float(request.form['Cash/Current Liability'])
    Total_income_Total_expense = float(request.form['Total income/Total expense'])
    Total_expense_Assets = float(request.form['Total expense/Assets'])
    Gross_Profit_to_Sales = float(request.form['Gross Profit to Sales'])

    predict =
model.predict([[Operating_Gross_Margin,Cash_flow_rate,Tax_rate_A,Net_worth_Assets,To
tal_Asset_Turnover,Cash_Total_Assets,Cash_Current_Liability,Total_income_Total_expens
e,Total_expense_Assets,Gross_Profit_to_Sales]])

    output = predict[0]

    return render_template('indexs.html',result=output)
```

```
if __name__ == "__main__":
    app.run(debug=True)
```

OUTPUT SCREENS

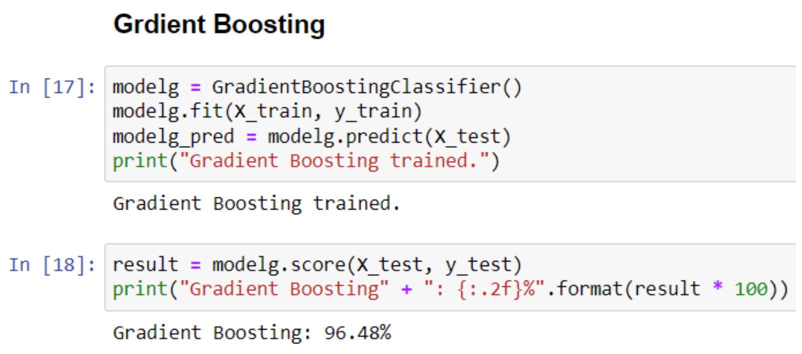
6. OUTPUT SCREENS



Decision Tree

```
In [19]: modeld = DecisionTreeClassifier()
         modeld.fit(X_train, y_train)
         model_pred = modeld.predict(X_test)
         print("Decision Tree trained.")

         Decision Tree trained.

In [20]: result = modeld.score(X_test, y_test)
         print("Decision Tree" + ": {:.2f}%".format(result * 100))

         Decision Tree: 94.53%
```

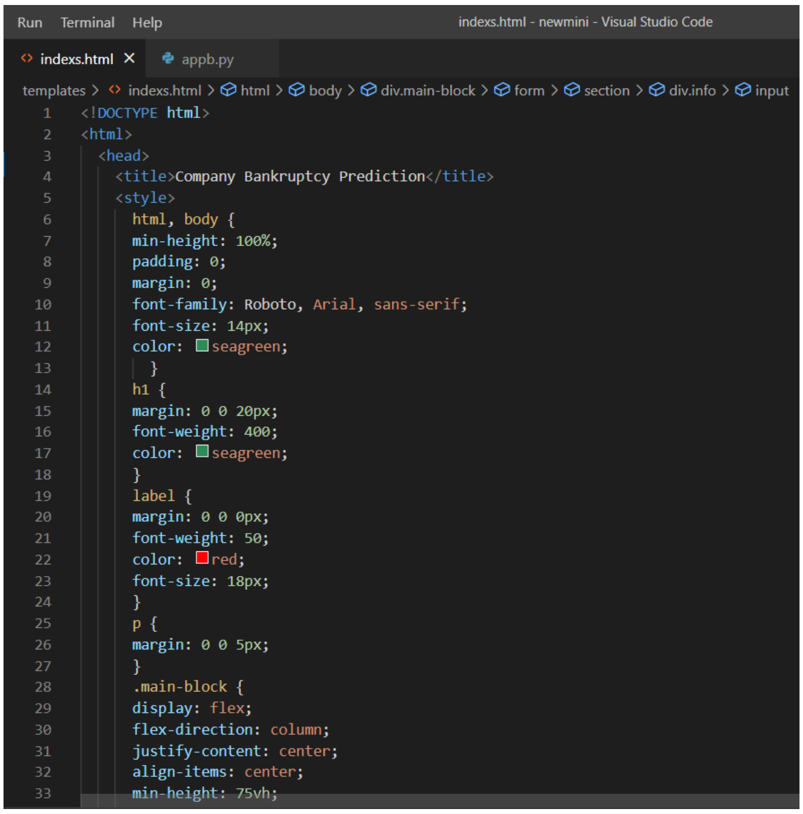
Fig 6.1 Decision Tree Output



Grdient Boosting

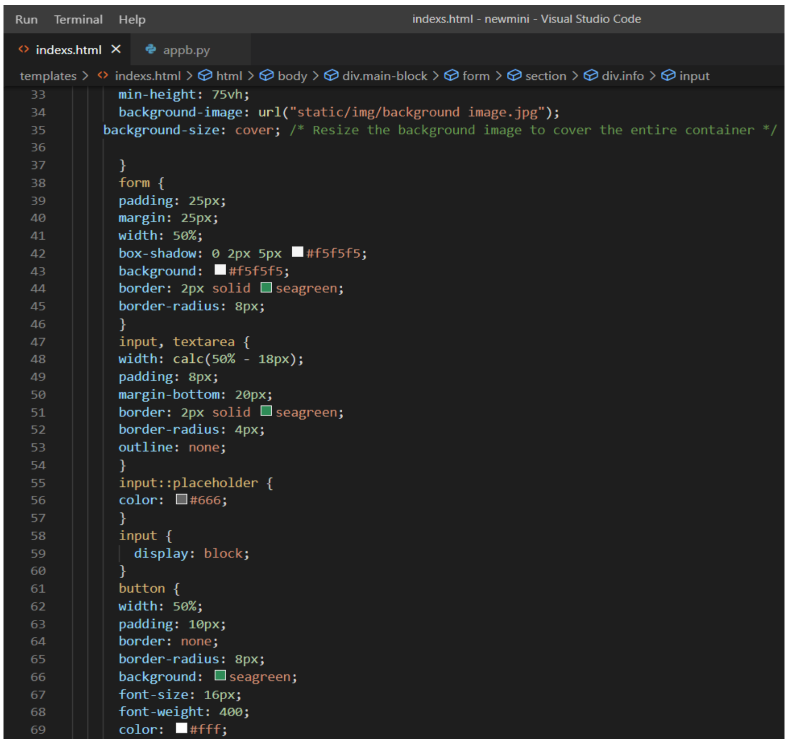
```
In [17]: modelg = GradientBoostingClassifier()
         modelg.fit(X_train, y_train)
         modelg_pred = modelg.predict(X_test)
         print("Gradient Boosting trained.")

         Gradient Boosting trained.

In [18]: result = modelg.score(X_test, y_test)
         print("Gradient Boosting" + ": {:.2f}%".format(result * 100))

         Gradient Boosting: 96.48%
```

Fig 6.2 Gradient Boosting Output


Run Terminal Help
indexs.html - newmini - Visual Studio Code
indexs.html
appb.py
templates > indexs.html > html > body > div.main-block > form > section > div.info > input

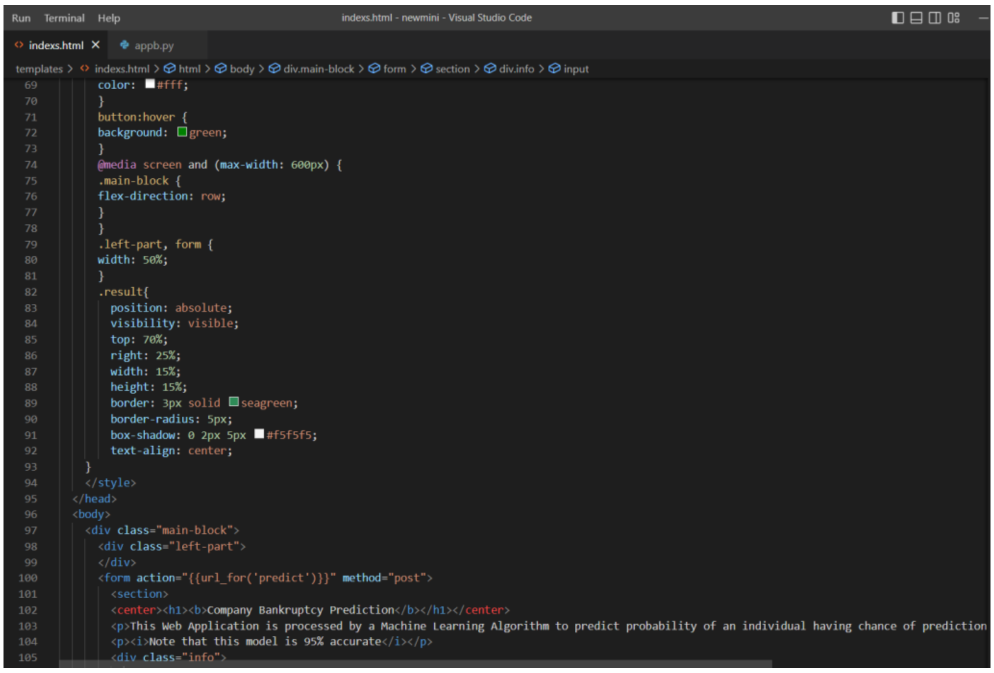
```
<!DOCTYPE html>
<html>
  <head>
    <title>Company Bankruptcy Prediction</title>
    <style>
      html, body {
      min-height: 100%;
      padding: 0;
      margin: 0;
      font-family: Roboto, Arial, sans-serif;
      font-size: 14px;
      color: seagreen;
        }
      h1 {
      margin: 0 0 20px;
      font-weight: 400;
      color: seagreen;
      }
      label {
      margin: 0 0 0px;
      font-weight: 50;
      color: red;
      font-size: 18px;
      }
      p {
      margin: 0 0 5px;
      }
      .main-block {
      display: flex;
      flex-direction: column;
      justify-content: center;
      align-items: center;
      min-height: 75vh;
```

Fig 6.3.1 Indexs.html code


Run Terminal Help
indexs.html - newmini - Visual Studio Code
indexs.html
appb.py
templates > indexs.html > html > body > div.main-block > form > section > div.info > input

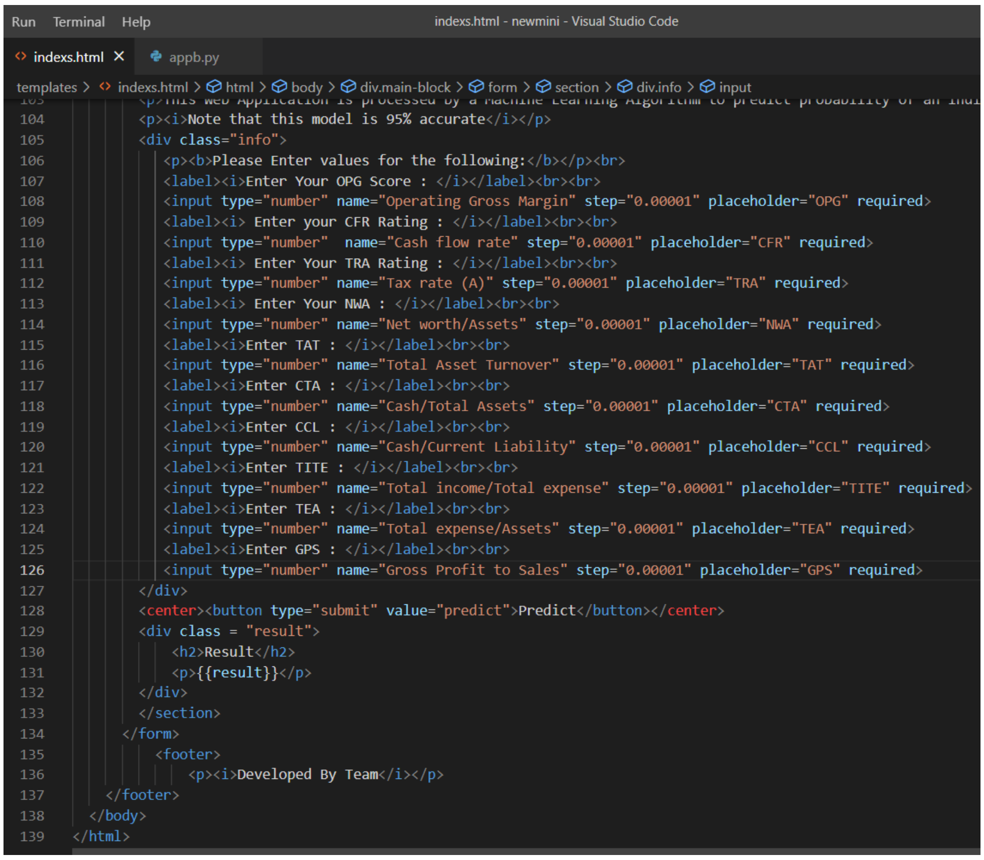
```
      min-height: 75vh;
      background-image: url("static/img/background image.jpg");
    background-size: cover; /* Resize the background image to cover the entire container */

      }
      form {
      padding: 25px;
      margin: 25px;
      width: 50%;
      box-shadow: 0 2px 5px #f5f5f5;
      background: #f5f5f5;
      border: 2px solid seagreen;
      border-radius: 8px;
      }
      input, textarea {
      width: calc(50% - 18px);
      padding: 8px;
      margin-bottom: 20px;
      border: 2px solid seagreen;
      border-radius: 4px;
      outline: none;
      }
      input::placeholder {
      color: #666;
      }
      input {
        display: block;
      }
      button {
      width: 50%;
      padding: 10px;
      border: none;
      border-radius: 8px;
      background: seagreen;
      font-size: 16px;
      font-weight: 400;
      color: #fff;
```

Fig 6.3.2 Indexs.html code



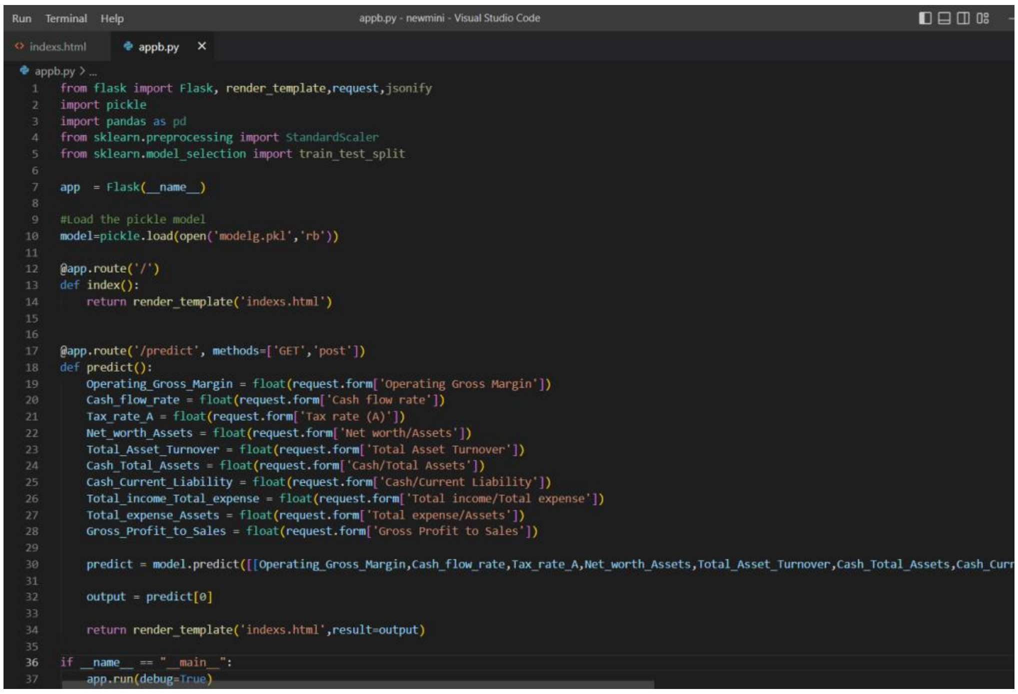
```
        color: #fff;
        }
        button:hover {
        background: green;
        }
        @media screen and (max-width: 600px) {
        .main-block {
        flex-direction: row;
        }
        }
        .left-part, form {
        width: 50%;
        }
        .result{
          position: absolute;
          visibility: visible;
          top: 70%;
          right: 25%;
          width: 15%;
          height: 15%;
          border: 3px solid seagreen;
          border-radius: 5px;
          box-shadow: 0 2px 5px #f5f5f5;
          text-align: center;
      }
      </style>
    </head>
    <body>
      <div class="main-block">
        <div class="left-part">
        </div>
        <form action="{{url_for('predict')}}" method="post">
          <section>
          <center><h1><b>Company Bankruptcy Prediction</b></h1></center>
          <p>This Web Application is processed by a Machine Learning Algorithm to predict probability of an individual having chance of prediction
          <p><i>Note that this model is 95% accurate</i></p>
          <div class="info">
```

Fig 6.3.3 Indexs.html code



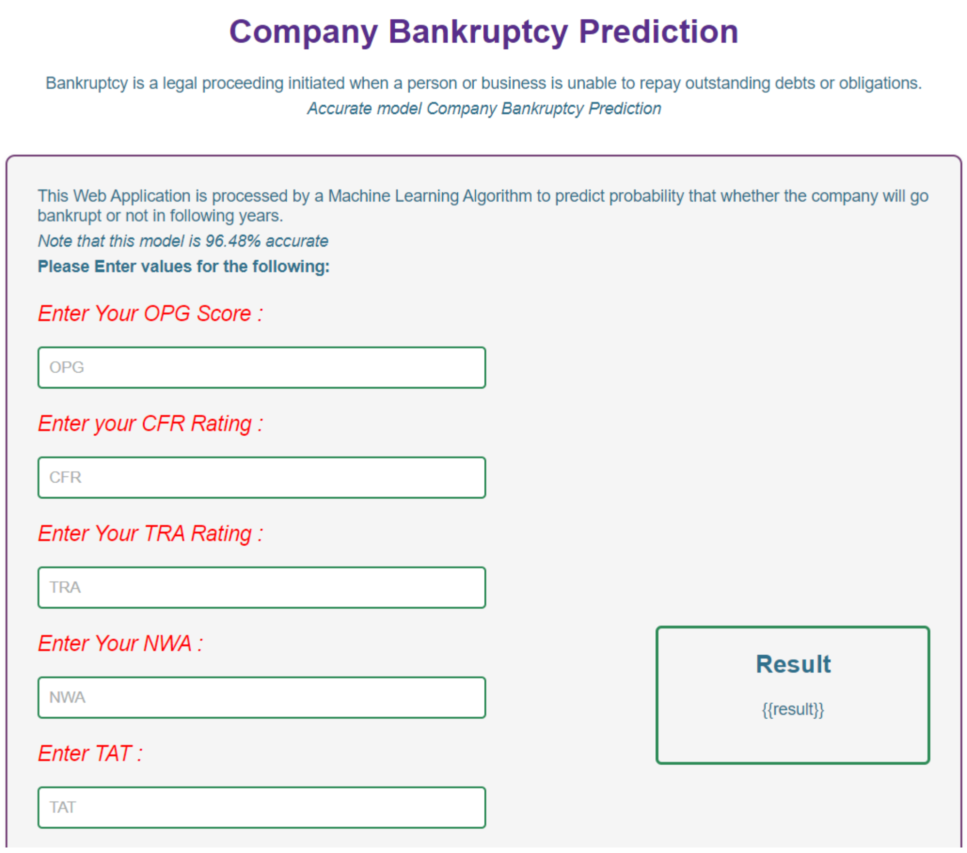
```
          <p><i>Note that this model is 95% accurate</i></p>
          <div class="info">
            <p><b>Please Enter values for the following:</b></p><br>
            <label><i>Enter Your OPG Score : </i></label><br><br>
            <input type="number" name="Operating Gross Margin" step="0.00001" placeholder="OPG" required>
            <label><i> Enter your CFR Rating : </i></label><br><br>
            <input type="number"  name="Cash flow rate" step="0.00001" placeholder="CFR" required>
            <label><i> Enter Your TRA Rating : </i></label><br><br>
            <input type="number" name="Tax rate (A)" step="0.00001" placeholder="TRA" required>
            <label><i> Enter Your NWA : </i></label><br><br>
            <input type="number" name="Net worth/Assets" step="0.00001" placeholder="NWA" required>
            <label><i>Enter TAT : </i></label><br><br>
            <input type="number" name="Total Asset Turnover" step="0.00001" placeholder="TAT" required>
            <label><i>Enter CTA : </i></label><br><br>
            <input type="number" name="Cash/Total Assets" step="0.00001" placeholder="CTA" required>
            <label><i>Enter CCL : </i></label><br><br>
            <input type="number" name="Cash/Current Liability" step="0.00001" placeholder="CCL" required>
            <label><i>Enter TITE : </i></label><br><br>
            <input type="number" name="Total income/Total expense" step="0.00001" placeholder="TITE" required>
            <label><i>Enter TEA : </i></label><br><br>
            <input type="number" name="Total expense/Assets" step="0.00001" placeholder="TEA" required>
            <label><i>Enter GPS : </i></label><br><br>
            <input type="number" name="Gross Profit to Sales" step="0.00001" placeholder="GPS" required>
          </div>
          <center><button type="submit" value="predict">Predict</button></center>
          <div class = "result">
              <h2>Result</h2>
              <p>{{result}}</p>
          </div>
          </section>
        </form>
            <footer>
                <p><i>Developed By Team</i></p>
      </footer>
    </body>
  </html>
```

Fig 6.3.4 Indexs.html code



```python
from flask import Flask, render_template,request,jsonify
import pickle
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

app  = Flask(__name__)

#Load the pickle model
model=pickle.load(open('modelg.pkl','rb'))

@app.route('/')
def index():
    return render_template('indexs.html')


@app.route('/predict', methods=['GET','post'])
def predict():
    Operating_Gross_Margin = float(request.form['Operating Gross Margin'])
    Cash_flow_rate = float(request.form['Cash flow rate'])
    Tax_rate_A = float(request.form['Tax rate (A)'])
    Net_worth_Assets = float(request.form['Net worth/Assets'])
    Total_Asset_Turnover = float(request.form['Total Asset Turnover'])
    Cash_Total_Assets = float(request.form['Cash/Total Assets'])
    Cash_Current_Liability = float(request.form['Cash/Current Liability'])
    Total_income_Total_expense = float(request.form['Total income/Total expense'])
    Total_expense_Assets = float(request.form['Total expense/Assets'])
    Gross_Profit_to_Sales = float(request.form['Gross Profit to Sales'])

    predict = model.predict([[Operating_Gross_Margin,Cash_flow_rate,Tax_rate_A,Net_worth_Assets,Total_Asset_Turnover,Cash_Total_Assets,Cash_Curr

    output = predict[0]

    return render_template('indexs.html',result=output)

if __name__ == "__main__":
    app.run(debug=True)
```

Fig 6.4 app.py code



Company Bankruptcy Prediction

Bankruptcy is a legal proceeding initiated when a person or business is unable to repay outstanding debts or obligations.
Accurate model Company Bankruptcy Prediction

This Web Application is processed by a Machine Learning Algorithm to predict probability that whether the company will go bankrupt or not in following years.
Note that this model is 96.48% accurate
Please Enter values for the following:

Enter Your OPG Score :
OPG

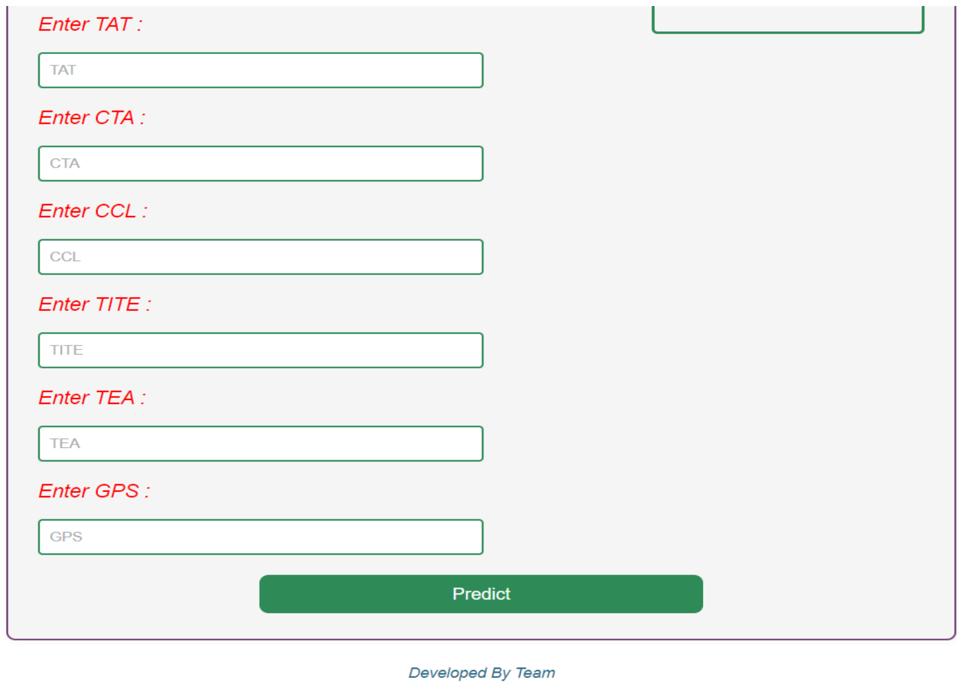
Enter your CFR Rating :
CFR

Enter Your TRA Rating :
TRA

Enter Your NWA :
NWA

Enter TAT :
TAT

Result
{{result}}



Enter TAT :
TAT

Enter CTA :
CTA

Enter CCL :
CCL

Enter TITE :
TITE

Enter TEA :
TEA

Enter GPS :
GPS

Developed By Team

Fig 6.5 Web page output

TESTING AND VALIDATION

7. TESTING AND VALIDATION

7.1 Introduction

The completion of a system is achieved only after it has been thoroughly tested. Though this gives a feel the project is completed, there cannot be any project without going through this stage. Hence in this stage it is decided whether the project can undergo the real time environment execution without any break downs, therefore a package can be rejected even at this stage.

Testing is a set of activities that can be planned in advance and conducted systematically. The proposed system is tested in parallel with the software that consists of its own phases of analysis, implementation, testing and maintenance.

7.2 Testing Strategies

A Strategy for software testing integrates software test cases into a series of well-planned steps that result in the successful construction of software. Software testing is a broader topic for what is referred to as Verification and Validation. Verification refers to the set of activities that ensure that the software correctly implements a specific function. Validation refers the set of activities that ensure that the software that has been built is traceable to customer's requirements

Unit Testing

Unit testing focuses verification effort on the smallest unit of software design that is the module. Using procedural design description as a guide, important control paths are tested to uncover errors within the boundaries of the module. The unit test is normally white box testing oriented and the step can be conducted in parallel for multiple modules.

Integration Testing

Integration testing is a systematic technique for constructing the program structure, while conducting test to uncover errors associated with the interface. The objective is to take unit tested methods and build a program structure that has been dictated by design.

Top-down Integration

Top-down integrations is an incremental approach for construction of program structure.

Modules are integrated by moving downward through the control hierarchy, beginning with the main control program. Modules subordinate to the main program are incorporated in the structure either in the breath-first or depth-first manner.

Bottom-up Integration

This method as the name suggests, begins construction and testing with atomic modules i.e., modules at the lowest level. Because the modules are integrated in the bottom-up manner the processing required for the modules subordinate to a given level is always available and the need for stubs is eliminated.

Validation Testing

At the end of integration testing software is completely assembled as a package. Validation testing is the next stage, which can be defined as successful when the software functions in the manner reasonably expected by the customer. Reasonable expectations are those defined in the software requirements specifications. Information contained in those sections form a basis for validation testing approach.

System Testing

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system. Although each test has a different purpose, all work to verify that all system elements have been properly integrated to perform allocated functions.

Security Testing

Attempts to verify the protection mechanisms built into the system.

Performance Testing

This method is designed to test runtime performance of software within the context of an integrated system**.**

CONCLUSION

8. CONCLUSION

Bankruptcy prediction models may use many different data and techniques. The latest studies regarding bankruptcy prediction used different kinds of variables and cutting-edge techniques. In this study, by applying a decision tree, and Gradient Boosting classification algorithms on the financial data of Taiwan dataset, we achieve a global bankruptcy prediction accuracy of above 95%. Compared to past analysis of the same dataset with different machine learning algorithms, we achieve a slight improvement in the bankruptcy cases and a improvement in solvency cases. We recognize the limitation in the prediction accuracy as arising from the significant overlap in the feature space between financial variables belonging to bankrupt and solvent companies. However, our study does not report significant differences in results in terms of prediction accuracy, regardless of the technique used. Moreover, a significant prediction accuracy rate is achieved by using financial ratios that are easily obtainable for most firms. In addition to its contribution to the academic literature, this study is of high interest for bankers who want to assess the probability of bankruptcy (and therefore of non-reimbursement) of firms requesting loans without having to compute many financial ratios and collect non-financial data.

REFERENCES

9. REFERENCES:

- Altman EI (1968) The Prediction of Corporate Bankruptcy: A Discriminant Analysis. *J Financ* 23: 193-194.

- Ohlson JA (1980) Financial Ratios and the Probabilistic Prediction of Bankruptcy. *J Account Res* 18: 109-131. doi: 10.2307/2490395

- A Review of Bankruptcy Prediction Studies: 1930-Present.

- Altman E I. Financial ratios, discriminant analysis and the prediction of corporate bankruptcy[J]. The journal of finance, 1968, 23(4): 589-609.

- Beaver W H. Financial ratios as predictors of failure[J]. Journal of accounting research, 1966: 71-111.

- Barboza F, Kimura H, Altman E. Machine learning models and bankruptcy prediction[J]. Expert Systems with Applications, 2017, 83: 405-417.

- Ohlson J A. Financial ratios and the probabilistic prediction of bankruptcy[J]. Journal of accounting research, 1980: 109-131.

- Lin W Y, Hu Y H, Tsai C F. Machine learning in financial crisis prediction: a survey[J]. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 2011, 42(4): 421-436

- https://www.sciencedirect.com/topics/computer-science/bankruptcy-prediction

- https://www.kaggle.com/datasets/fedesoriano/company-bankruptcy-prediction

- https://en.wikipedia.org/wiki/Category:Bankruptcy

- https://en.wikipedia.org/wiki/Bankruptcy_prediction#:~:text=Bankruptcy%20prediction%20is%20the%20art,of%20finance%20and%20accounting%20research.

- https://epublications.marquette.edu/