Project Report

Project Name: Anticipating Business Bankruptcy

Team ID: Team – 592444

Team Members:

- Devanshu Gupta
- Jaivardhan Tamminana
- Taniya Hussain
- Prakhar Vishwas

Index

Back to Index

INTRODUCTION

Project Overview

The project on “Anticipating Business Bankruptcy” is aimed at developing a machine learning model that can predict the likelihood of a business going bankrupt based on various financial and non-financial factors. This project is critical for financial institutions, investors, and business stakeholders who want to assess the risk associated with lending to or investing in a particular company. By leveraging historical data and advanced machine learning techniques, the goal is to provide a predictive tool that can help stakeholders make more informed decisions and manage risk effectively.

Purpose

The primary purpose of this project is to mitigate the economic and financial risks associated with business bankruptcies. Bankruptcy can have far-reaching consequences, affecting not only the companies themselves but also their employees, creditors, and the broader economy. By developing predictive models for anticipating business bankruptcy, the project serves several essential purposes:

- **Risk Mitigation:** The project aims to help businesses and financial institutions proactively identify companies at risk of bankruptcy. This enables stakeholders to take preventive measures, such as restructuring, refinancing, or asset protection, to minimize the adverse effects of bankruptcy.
- **Investment Decision Support:** Investors, both individual and institutional, can use the predictive models to make more informed investment decisions. Identifying bankruptcy risks can help them avoid investing in companies with a higher likelihood of financial distress.
- **Credit Risk Assessment:** Banks and financial institutions can use the models to assess the creditworthiness of their clients and manage their lending portfolios more effectively. This can reduce the likelihood of loan defaults and financial losses.
- **Regulatory Compliance:** Regulatory bodies can benefit from this project by using the models to monitor and regulate companies in their jurisdiction. Anticipating business bankruptcy can help regulators enforce compliance with financial regulations and prevent systemic risks.
- **Economic Stability:** By helping to prevent and manage business bankruptcies, the project contributes to the overall economic stability of a region or country. Reducing the frequency of bankruptcy filings can lead to fewer job losses and a healthier business environment.

Back to Index

LITERATURE SURVEY

Existing problem

Anticipating business bankruptcy is a challenging problem that has significant implications for financial stability and risk management. The existing methods for predicting business bankruptcy typically rely on traditional financial ratios and expert judgment. These methods often have limitations in terms of accuracy, scalability, and the ability to handle large datasets with diverse features. Machine learning techniques offer the potential to improve the accuracy of bankruptcy prediction by analysing a broader range of data sources and identifying complex patterns and relationships.

References

To understand the existing research and methodologies related to anticipating business bankruptcy, the project will refer to a range of academic papers, articles, and books on the subject. Some relevant references may include:

- Altman, E. I. (1968). Financial ratios, discriminant analysis and the prediction of corporate bankruptcy. The Journal of Finance, 23(4), 589-609.
- Ohlson, J. A. (1980). Financial ratios and the probabilistic prediction of bankruptcy. Journal of Accounting Research, 18(1), 109-131.
- Zmijewski, M. E. (1984). Methodological issues related to the estimation of financial distress prediction models. Journal of Accounting Research, 22(1), 59-82.

Problem Statement Definition

The problem statement for this project is defined as follows:

“Develop a machine learning model that can accurately predict the likelihood of business bankruptcy based on historical financial and non-financial data. The model should analyze a diverse range of features, including financial ratios, industry-specific factors, market sentiment, and macroeconomic indicators. The goal is to provide a reliable tool for assessing bankruptcy risk, allowing financial institutions and investors to make informed decisions about lending, investment, and risk management.”

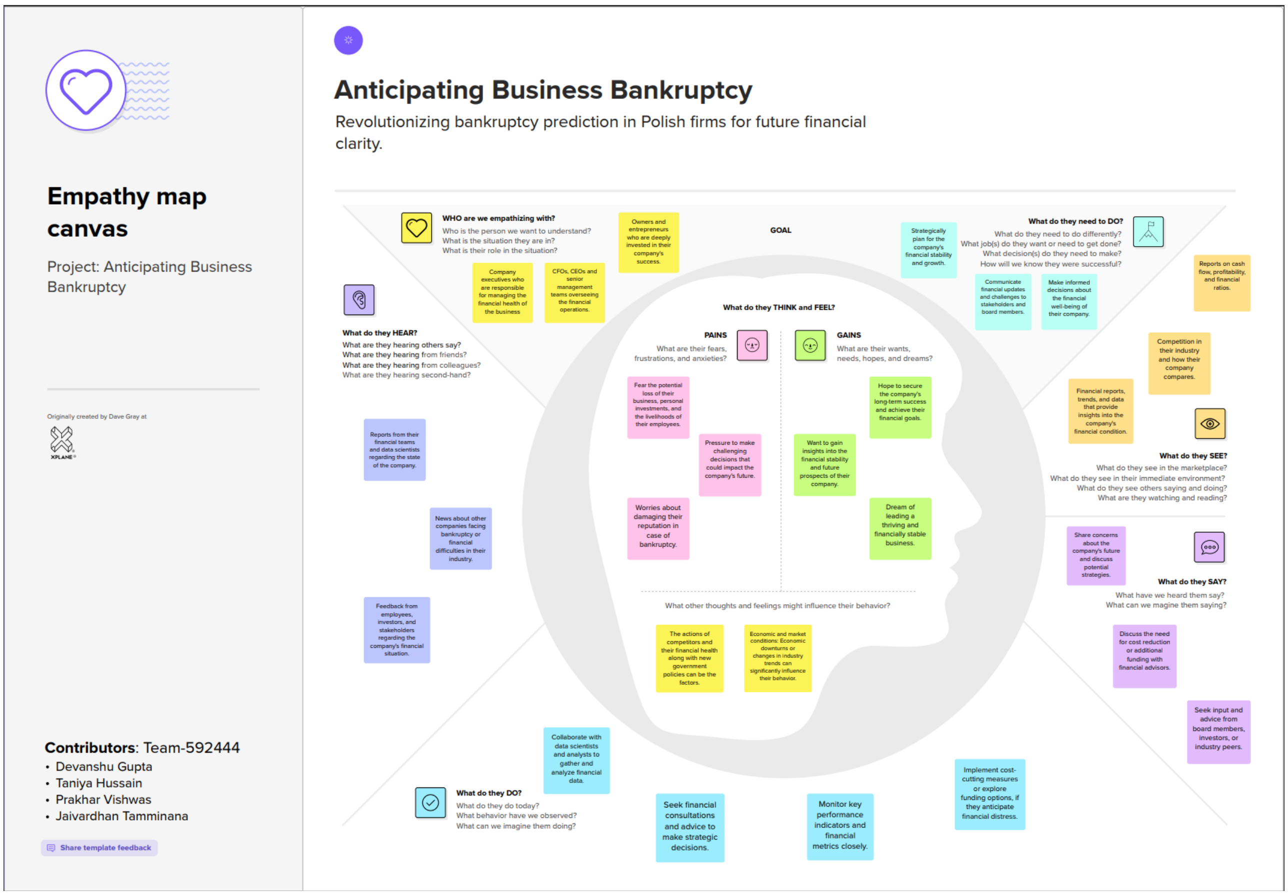
The project will focus on collecting and preprocessing relevant data, selecting appropriate machine learning algorithms, training and fine-tuning the model, and evaluating its performance using appropriate metrics. Additionally, the model will be designed to handle real-time data for ongoing risk assessment and decision support.

Back to Index

IDEATION AND PROPOSED SOLUTION

Empathy Map Canvas

An empathy map canvas is a visual tool and framework used in design thinking and user experience (UX) design to help teams gain a deeper understanding of users and their perspectives. It typically consists of a canvas divided into sections, with each section representing various aspects of the user's experience, thoughts, feelings, and needs. The canvas is used to gather and organize insights about users and to create a shared understanding within a team. It often includes sections for capturing what the user says, thinks, does, and feels, as well as their pain points, needs, and aspirations. The empathy map canvas serves as a collaborative tool to foster empathy, align team members, and guide the design process towards creating more user-centered and meaningful solutions.



Empathy map canvas

Project: Anticipating Business Bankruptcy

Originally created by Dave Gray at XPLANE

Contributors: Team-592444
- Devanshu Gupta
- Taniya Hussain
- Prakhar Vishwas
- Jaivardhan Tamminana

Share template feedback

Anticipating Business Bankruptcy

Revolutionizing bankruptcy prediction in Polish firms for future financial clarity.

WHO are we empathizing with?
Who is the person we want to understand?
What is the situation they are in?
What is their role in the situation?

- Company executives who are responsible for managing the financial health of the business
- CFOs, CEOs and senior management teams overseeing the financial operations.
- Owners and entrepreneurs who are deeply invested in their company's success.

GOAL

What do they need to DO?
What do they need to do differently?
What job(s) do they want or need to get done?
What decision(s) do they need to make?
How will we know they were successful?

- Strategically plan for the company's financial stability and growth.
- Communicate financial updates and challenges to stakeholders and board members.
- Make informed decisions about the financial well-being of their company.

What do they SEE?
What do they see in the marketplace?
What do they see in their immediate environment?
What do they see others saying and doing?
What are they watching and reading?

- Reports on cash flow, profitability, and financial ratios.
- Competition in their industry and how their company compares.
- Financial reports, trends, and data that provide insights into the company's financial condition.

What do they SAY?
What have we heard them say?
What can we magine them saying?

- Share concerns about the company's future and discuss potential strategies.
- Discuss the need for cost reduction or additional funding with financial advisors.
- Seek input and advice from board members, investors, or industry peers.

What do they DO?
What do they do today?
What behavior have we observed?
What can we imagine them doing?

- Collaborate with data scientists and analysts to gather and analyze financial data.
- Seek financial consultations and advice to make strategic decisions.
- Monitor key performance indicators and financial metrics closely.
- Implement cost-cutting measures or explore funding options, if they anticipate financial distress.

What do they HEAR?
What are they hearing others say?
What are they hearing from friends?
What are they hearing from colleagues?
What are they hearing second-hand?

- Reports from their financial teams and data scientists regarding the state of the company.
- News about other companies facing bankruptcy or financial difficulties in their industry.
- Feedback from employees, investors, and stakeholders regarding the company's financial situation.

What do they THINK and FEEL?

PAINS
What are their fears, frustrations, and anxieties?

- Fear the potential loss of their business, personal investments, and the livelihoods of their employees.
- Pressure to make challenging decisions that could impact the company's future.
- Worries about damaging their reputation in case of bankruptcy.

GAINS
What are their wants, needs, hopes, and dreams?

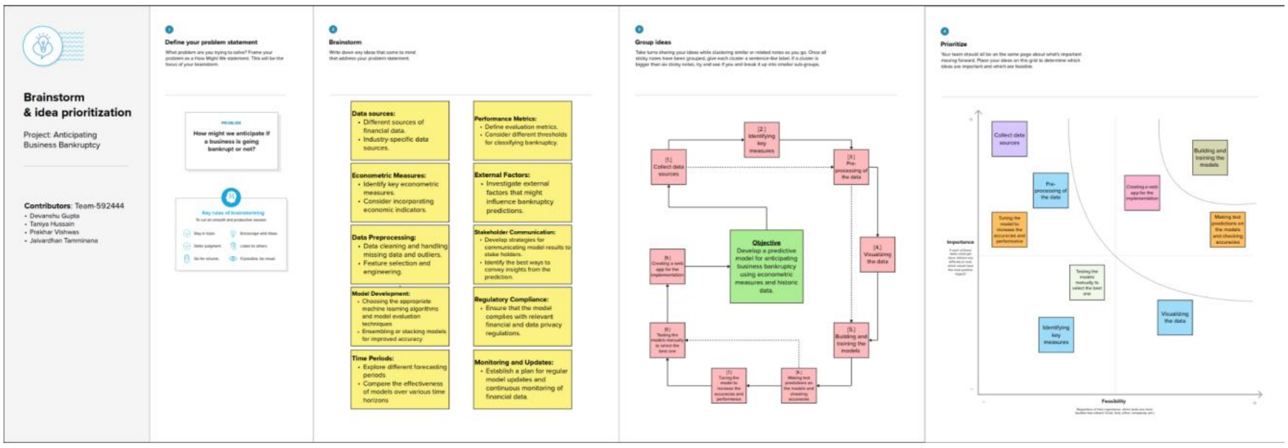
- Hope to secure the company's long-term success and achieve their financial goals.
- Want to gain insights into the financial stability and future prospects of their company.
- Dream of leading a thriving and financially stable business.

What other thoughts and feelings might influence their behavior?

- The actions of competitors and their financial health along with new government policies can be the factors.
- Economic and market conditions: Economic downturns or changes in industry trends can significantly influence their behavior.

Ideation & Brainstorming

A brainstorming idea prioritization map is a visual tool or framework used to assess, rank, and prioritize a set of ideas generated during a brainstorming session or ideation process. It helps individuals or teams determine which ideas are most promising and worth pursuing further. Typically, this map consists of various criteria or factors that are important for evaluating and selecting ideas. Participants assign scores, rankings, or labels to each idea based on these criteria. The result is a structured and visually organized representation of ideas, highlighting those with the highest potential or alignment with project goals. It aids in decision-making, resource allocation, and focuses on the most valuable concepts or projects.


Brainstorm & idea prioritization

Project: Anticipating Business Bankruptcy

Contributors: Team-592444
- Devanshu Gupta
- Taniya Hussain
- Prakhar Vishwas
- Jaivardhan Tamminana

Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

PROBLEM

How might we anticipate if a business is going bankrupt or not?

Key rules of brainstorming

To run an smooth and productive session

Stay in topic.	Encourage wild ideas.
Defer judgment.	Listen to others.
Go for volume.	If possible, be visual.

Brainstorm

Write down any ideas that come to mind that address your problem statement.

Data sources:
- Different sources of financial data.
- Industry-specific data sources.

Performance Metrics:
- Define evaluation metrics.
- Consider different thresholds for classifying bankruptcy.

Econometric Measures:
- Identify key econometric measures.
- Consider incorporating economic indicators.

External Factors:
- Investigate external factors that might influence bankruptcy predictions.

Data Preprocessing:
- Data cleaning and handling missing data and outliers.
- Feature selection and engineering.

Stakeholder Communication:
- Develop strategies for communicating model results to stake holders.
- Identify the best ways to convey insights from the prediction.

Model Development:
- Choosing the appropriate machine learning algorithms and model evaluation techniques
- Ensembling or stacking models for improved accuracy

Regulatory Compliance:
- Ensure that the model complies with relevant financial and data privacy regulations.

Time Periods:
- Explore different forecasting periods
- Compare the effectiveness of models over various time horizons

Monitoring and Updates:
- Establish a plan for regular model updates and continuous monitoring of financial data.

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

[1] Collect data sources
[2] Identifying key measures
[3] Pre-processing of the data
[4] Visualizing the data
[5] Building and training the models
[6] Making test predictions on the models and checking accuracies
[7] Tuning the model to increase the accuracies and performance
[8] Testing the models manually to select the best one
[9] Creating a web app for the implementation
Objective
Develop a predictive model for anticipating business bankruptcy using econometric measures and historic data.

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

Importance
If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?
Feasibility
Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)
Collect data sources
Pre-processing of the data
Tuning the model to increase the accuracies and performance
Creating a web app for the implementation
Building and training the models
Making test predictions on the models and checking accuracies
Testing the models manually to select the best one
Identifying key measures
Visualizing the data

REQUIREMENT ANALYSIS

Requirement analysis for a Business Bankruptcy Prediction System involves understanding the specific needs and expectations of users to create a system that accurately assesses bankruptcy risks. The requirements can be broadly categorised into Functional and Non-Functional Requirements.

Functional Requirement

Functional requirements focus on the system's core capabilities, defining what the system must do to fulfil users' needs. In the context of bankruptcy prediction, functional requirements include:

1. Data Collection and Preprocessing:
   - The system must efficiently collect historical financial data and external market indicators.
   - Data preprocessing should handle missing values, outliers, and inconsistencies, ensuring clean and reliable datasets for analysis.
2. Feature Selection and Model Training:
   - The system should select pertinent features from the data, optimizing the accuracy of bankruptcy predictions.
   - Machine learning algorithms need to be trained with selected features to learn patterns from historical data.
3. Prediction Generation:
   - The trained model must process new data inputs and generate bankruptcy risk scores based on learned patterns.
   - Predictions should be precise, offering actionable insights to users for decision-making.
4. Result Presentation:
   - Prediction results, including risk scores and contributing factors, must be presented to users through a user-friendly interface.
   - The system should provide detailed explanations for risk factors, enabling users to interpret the predictions effectively.

Ensuring these functional aspects align with users' expectations is vital for the system's success.

Back to Index

Non-functional Requirements

Non-functional requirements define the system's qualities, such as performance, security, and usability. In the case of a Business Bankruptcy Prediction System, non-functional requirements include:

1. Performance:
    - The system must handle a substantial volume of data efficiently, ensuring quick response times even during peak usage.
    - Predictions should be generated within an acceptable time frame to support real-time decision-making.
2. Security:
    - Data security is paramount. The system must employ robust encryption and authentication mechanisms to protect sensitive financial data.
    - User access control mechanisms should be in place to ensure that only authorized personnel can access specific functionalities.
3. Scalability:
    - The system should be scalable, capable of accommodating growing datasets and an expanding user base without compromising performance.
    - Scalability measures should be in place to handle increased computational demands as the system usage grows.
4. Compliance:
    - The system must adhere to legal and ethical standards, ensuring compliance with industry regulations and data protection laws.
    - Regular compliance audits and adherence to industry best practices are essential to maintaining legal integrity.
5. Usability:
    - The user interface must be intuitive and user-friendly, catering to users with varying levels of technical expertise.
    - Clear visualization and reporting tools should be provided to enhance user experience and facilitate ease of interpretation.

By addressing these requirements, the system aims to provide accurate, secure, and user-friendly bankruptcy predictions, thereby assisting businesses and financial professionals in making informed decisions.

Back to Index

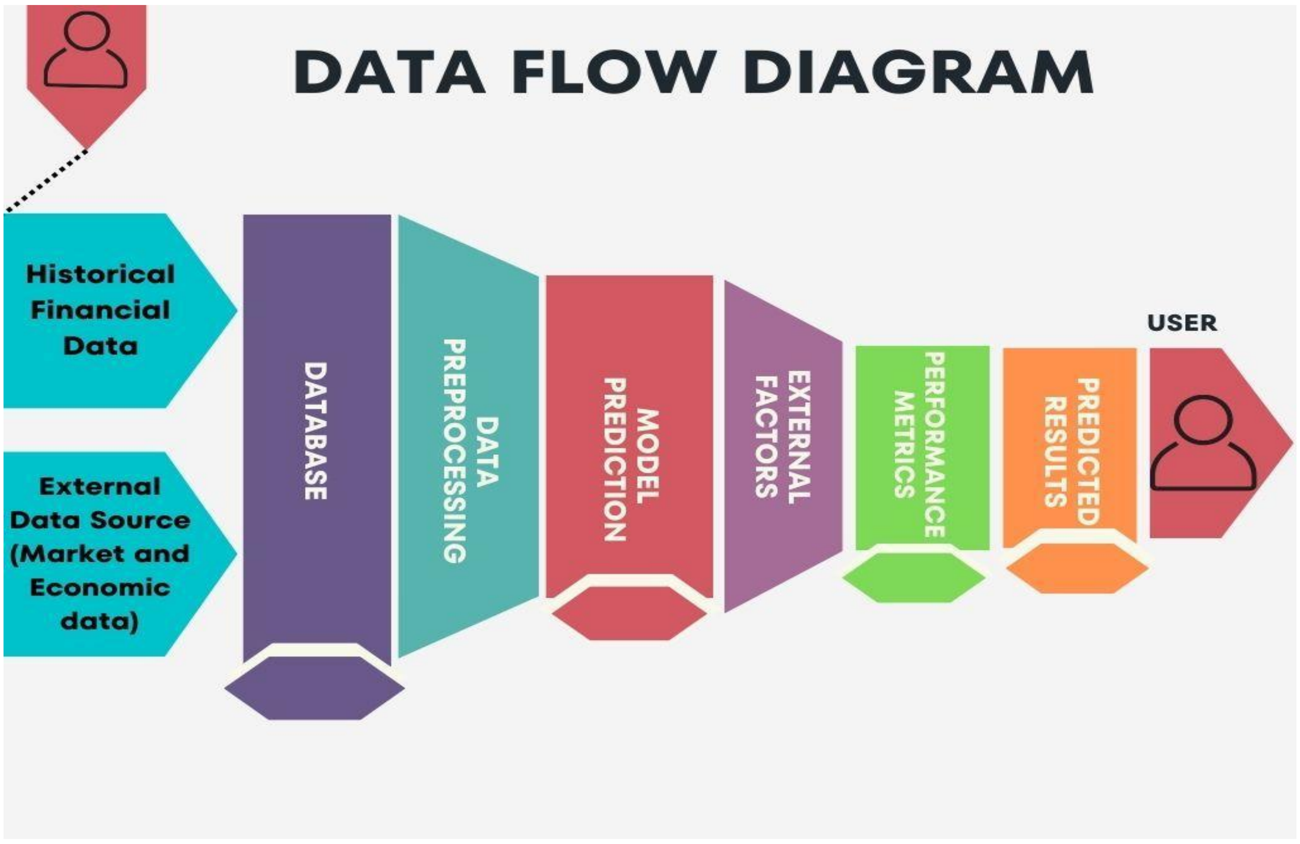
PROJECT DESIGN

Data Flow Diagrams & User Stories

The Data Flow Diagram (DFD) at level 0 involves representing a system's major: Processes, Data Flows, Data Stores, and External Entities.

1. External Entities:
   - User - Interacts with the system by providing input data and receiving prediction results.
   - DataBase – Represents external storage where historical financial data and model configurations are stored.
   - External Data Source- Provides additional market and economic data used in the prediction.
   - Prediction Results - Displays the result of the bankruptcy predictions to the user.
2. Processes:
   - Data Collection -Gathers historical financial data from the Database and additional market data from the External Data Source.
   - Data Preprocessing-Cleans and preprocesses the collected data, handling missing values and outliers. Feature Selection -Selects relevant features from the pre-processed data for the machine learning model.
   - Model Training-Uses the selected features to train a machine learning model for bankruptcy prediction.
   - Model Prediction -Applies the trained model to new data to generate bankruptcy predictions.
   - Result Presentation -Formats the prediction results and presents them to the user through the Prediction Results Output.
3. Data Stores:
   - Historical Financial Data-Contains past financial records retrieved from the Database.
   - Market and Economic Data-Stores additional data obtained from the External Data Source.
   - Pre-processed Data-Data cleaned and prepared for feature selection and model training.
   - Trained Model-Machine learning model trained for bankruptcy prediction.
   - Prediction Results- Results generated by the model, including risk scores and related information.
4. Data Flows:
   - Historical Financial Data Flow- Data flows from the Database to Data Collection for analysis.
   - External Data Flow-Additional market and economic data flow from the External Data Source to Data Collection.
   - Pre-processed Data Flow-Cleaned data moves from Data Collection to Data Preprocessing.
   - Model Training Flow-Selected features are used in Model Training to create the Trained Model.
   - Prediction Data Flow-New data is processed by Prediction Generation using the Trained Model to generate Prediction Results.
   - Result Presentation Flow-Formatted results flow from Prediction Generation to the Prediction Results Output for user viewing.

Back to Index



User stories are concise, user-focused descriptions of software features or functionalities, written from the perspective of end-users. They serve as a tool to capture the desired behaviours and outcomes of a system without delving into extensive technical details. Each user story typically includes a user type, a feature description, acceptance criteria, and a priority level. In our Business Bankruptcy Prediction System, here is how we have used them:

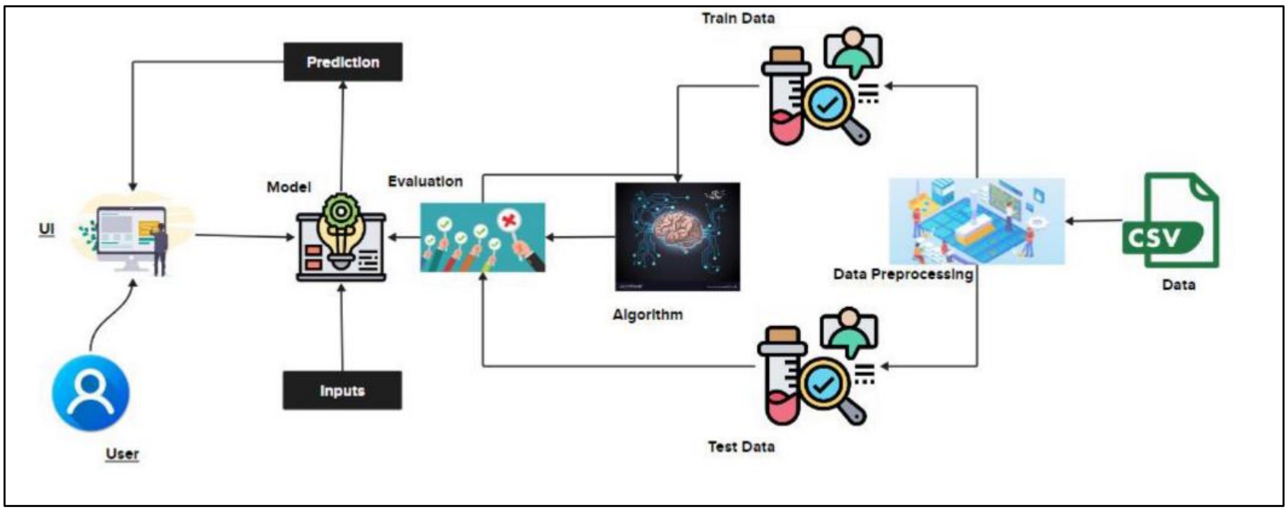
1. *User Alignment*: User stories have enabled us to align our development efforts with the needs and expectations of various users such as data Scientists, stakeholders, financial analysts, business owners, and others.
2. *Requirement Definition*: Each user story defines a specific feature or function that users require. For instance, "As a Researcher, I want to look through historical financial data for analysis."
3. *Acceptance Criteria*: User stories include acceptance criteria, which clearly outline the conditions that must be met for a user story to be considered complete. This ensures that the delivered feature aligns with user expectations. For example, "The system allows uploading CSV files containing financial data, and the data is successfully processed and stored."
4. *Prioritization*: We have assigned priority levels to user stories, allowing us to focus on the most critical features first. This aids in incremental development and the delivery of valuable functionality early in the development process.
5. *Enhanced Communication*: User stories facilitate clear and concise communication among the development team, stakeholders, and users. They provide a common language for discussing requirements and expectations.

Back to Index

Solution Architecture

Solutions Architecture is a critical aspect of designing a robust and effective Business Bankruptcy Prediction System.

1. *User Interface (UI)*: The UI of the Business Bankruptcy Prediction System has been meticulously designed to offer users an intuitive and seamless experience. It provides a user-friendly dashboard where users can upload CSV files containing financial data effortlessly. The UI focuses on clarity and simplicity, ensuring that users, including financial analysts and business owners, can easily navigate through the system. Visualization tools and interactive elements have been incorporated to enhance the interpretability of bankruptcy predictions, allowing users to comprehend complex data insights immediately.
2. *Model and Algorithm Selection*: In the Solutions Architecture, careful consideration was given to the selection of machine learning algorithms for bankruptcy prediction. Various algorithms, including logistic regression, decision trees, and neural networks, were evaluated to determine their effectiveness in capturing the underlying patterns in financial data. After rigorous evaluation, the appropriate algorithm was chosen based on its accuracy, scalability, and ability to manage the complexity of the dataset.
3. *Data Preprocessing from CSV*: Dealing with CSV data requires robust preprocessing techniques. The Solutions Architecture incorporates a data preprocessing pipeline tailored for CSV files. This process involves handling missing values, removing outliers, and normalizing data to ensure consistency. Additionally, categorical data in CSV files is encoded appropriately to make it compatible with machine learning algorithms. The preprocessing stage ensures that the data fed into the model is clean, standardized, and ready for effective analysis and prediction.
4. *Evaluation Strategy*: The Solutions Architecture includes a comprehensive evaluation strategy to assess the performance of the prediction model. Various metrics such as accuracy, precision, recall, and F1-score are utilized to measure the model's effectiveness. Cross-validation techniques are employed to validate the model's robustness and reliability across different datasets. Regular evaluation cycles are implemented, allowing the system to continuously learn from new data and adapt to evolving patterns, ensuring the accuracy and relevance of the bankruptcy predictions over time.



Back to Index

PROJECT PLANNING AND SCHEDULING

Technical Architecture

In the context of business bankruptcy anticipation, our technical architecture emphasizes the following:

1. *Scalability*: Given the dynamic nature of financial data and the need to analyse extensive datasets, the system is designed to scale horizontally and vertically. This architecture allows the system to handle an increasing volume of data efficiently, ensuring it remains responsive to user demands and market fluctuations.
2. *Data Integration*: The architecture supports the integration of external market and economic data sources. For instance, real-time stock market data, economic indicators, and industry-specific metrics are integrated seamlessly to enhance prediction accuracy.
3. *Reliability*: To ensure accurate predictions, the architecture incorporates redundant systems and data backup mechanisms. This safeguards against system failures and data loss, critical in the context of business bankruptcy where data integrity is paramount.

For instance, the technical architecture allows the system to integrate and analyse historical financial data with real-time stock market data, providing users with up-to-the-minute insights into their company's financial health and potential bankruptcy risks.

Sprint Planning & Estimation

Sprint planning and estimation are pivotal in the development of the Business Bankruptcy Prediction System.

1. *User Stories*: User stories, derived from the anticipation of critical financial indicators and risk factors, drive sprint planning. For example, "As a financial analyst, I want to analyse cash flow trends to predict bankruptcy risks."
2. *Sprint Goals*: Sprints are organized around achieving specific bankruptcy prediction goals. These goals are based on the relative importance of user stories, aligning with the anticipation of the most critical risk factors.
3. *Estimation*: During sprint planning sessions, the development team estimates the complexity and effort required for each user story, assigning story points. For instance, a user story related to analysing debt ratios is estimated based on the anticipated complexity.

For Instance, using insights from bankruptcy anticipation, sprint planning assigns higher priority and more resources to user stories related to financial indicators like liquidity ratios, which are known to be strong predictors of bankruptcy.

Back to Index

Sprint Delivery Schedule

Sprint delivery schedules are a fundamental aspect of project management for the Business Bankruptcy Prediction System.

1. *Duration Optimization*: Sprint durations are optimized to strike a balance between rapid feature delivery and comprehensive testing. Shorter sprints allow the team to respond quickly to user needs and emerging risk factors.
2. *Sprint Reviews*: At the end of each sprint, stakeholders conduct reviews to evaluate the delivered features. This process ensures alignment with the anticipated requirements, providing an opportunity for real-time adjustments.
3. *Retrospectives*: Regular retrospectives after each sprint allow the team to reflect on their performance and adapt the project plan. Insights from bankruptcy anticipation help refine the sprint delivery schedule based on evolving needs and user feedback.

For instance, the project schedules shorter sprints to deliver critical features quickly, responding to newly identified risk factors or market trends that could impact bankruptcy predictions.

In conclusion, by aligning technical architecture, sprint planning, and sprint delivery schedules with anticipated bankruptcy risks, the project aims to provide users with timely and effective tools for financial risk management and decision-making.

Back to Index

CODING AND SOLUTIONING

The coding phase assumes a pivotal and all-encompassing role in the development of the machine learning model designed for the critical task of bankruptcy prediction within the scope of this project. This phase is a complex and multifaceted process that involves a series of indispensable steps and methodological approaches, all of which are intricately interconnected to establish the foundational framework for achieving highly precise and reliable forecasts. In this dedicated section of our project documentation, we embark on an extensive and in-depth exploration of the Python coding process, meticulously crafted for this specific purpose. By harnessing the immense capabilities of widely acclaimed libraries and leveraging innovative techniques, we embark on a comprehensive journey with the overarching goal of unravelling the intricate complexities inherent in transforming raw financial data into actionable predictive insights. Within this section, we not only provide a comprehensive high-level overview of the coding process but also offer meticulous and detailed step-by-step guidance to ensure a profound and comprehensive understanding of this critical component that underpins the entire project.

Importing the Libraries

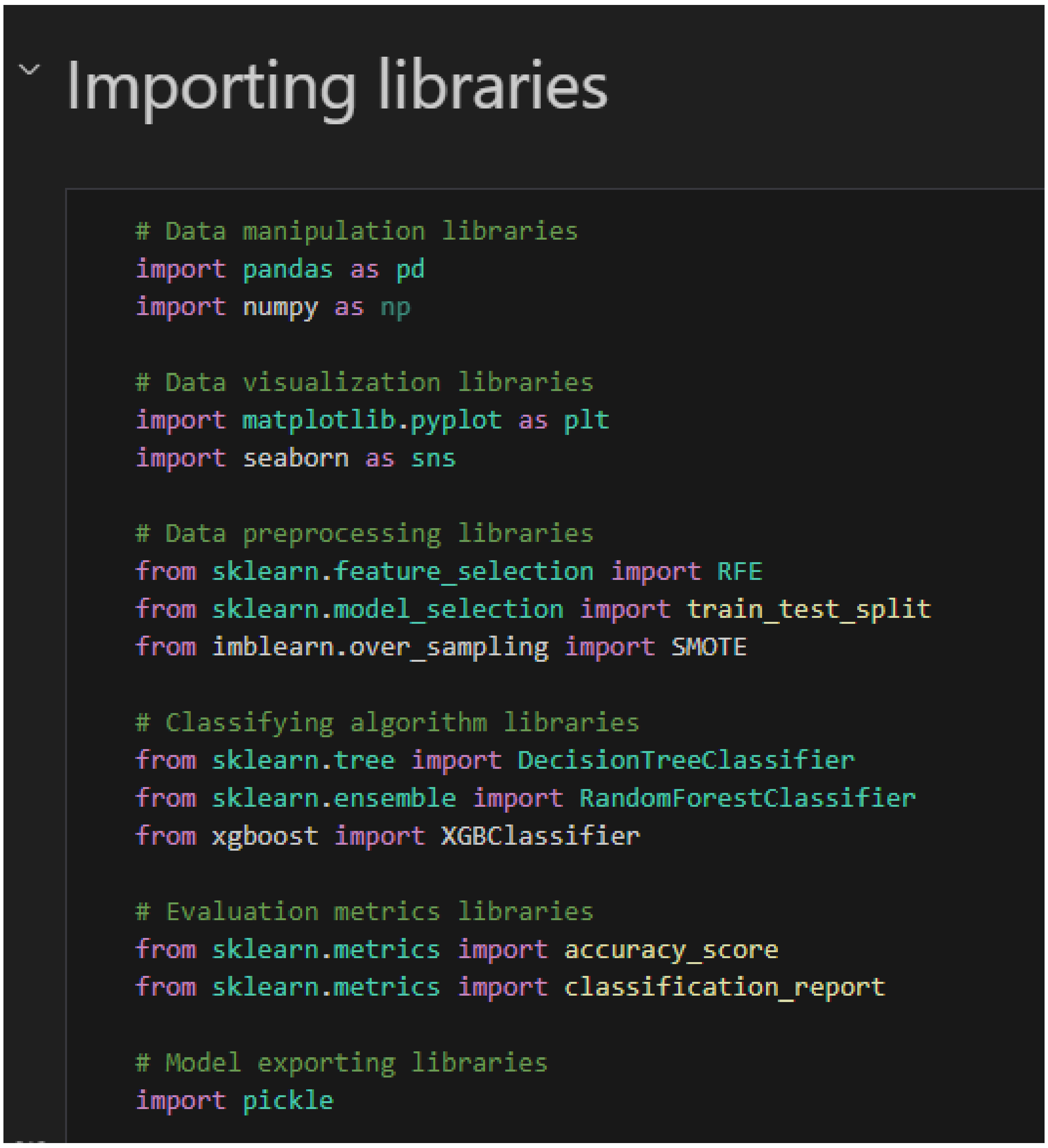
Various libraries played a fundamental role in the development of our machine learning model for bankruptcy prediction. These libraries provide essential functionalities for data manipulation, visualization, preprocessing, classification algorithms, evaluation metrics, and model exporting. Let us take a closer look at each of these libraries.

The following libraries were imported to aid us in creating the model:

1. `` `pandas` ``: A powerful data manipulation tool that allows us to work with structured data, providing data structures and functions for data cleaning, transformation, and analysis.
2. `` `numpy` ``: A fundamental library for numerical operations in Python, providing support for arrays and mathematical functions, making it indispensable for scientific and numerical computations.
3. `` `matplotlib` ``: A widely used data visualization library in Python that enables the creation of several types of graphs and charts, making it crucial for visually analysing data.
4. `` `seaborn` ``: A data visualization library built on top of Matplotlib, which simplifies the creation of attractive and informative statistical graphics.
5. `` `RFE from sklearn.feature_selection` ``: A feature selection technique from the scikit-learn library that helps identify the most relevant features for our model by recursively fitting the model with different subsets of features.
6. `` `train_test_split from sklearn.model_selection` ``: Essential for dividing our dataset into training and testing subsets, enabling us to assess our model's performance effectively.
7. `` `SMOTE from imblearn.over_sampling` ``: Synthetic Minority Over-sampling Technique (SMOTE) is used to address class imbalance in our dataset by generating synthetic examples of the minority class.
8. `` `DecisionTreeClassifier from sklearn.tree` ``: Decision Tree is a supervised learning algorithm used for classification tasks, where it creates a tree-like model to make decisions.
9. `` `RandomForestClassifier from sklearn.ensemble` ``: Random Forest is an ensemble learning method that combines multiple decision trees to improve prediction accuracy and reduce overfitting.
10. `` `XGBClassifier from xgboost` ``: XGBoost is a popular gradient boosting library that offers high-performance classification and regression capabilities.

Back to Index

11. `accuracy_score from sklearn.metrics`: A common metric used to evaluate the performance of classification models by measuring the ratio of correctly predicted instances to the total number of instances.
12. `classification_report from sklearn.metrics`: Provides a comprehensive summary of classification metrics such as precision, recall, F1-score, and support for each class in a classification problem.
13. `pickle`: Used for serializing and deserializing Python objects, allowing us to save and load our trained machine learning models for future use.



Importing libraries

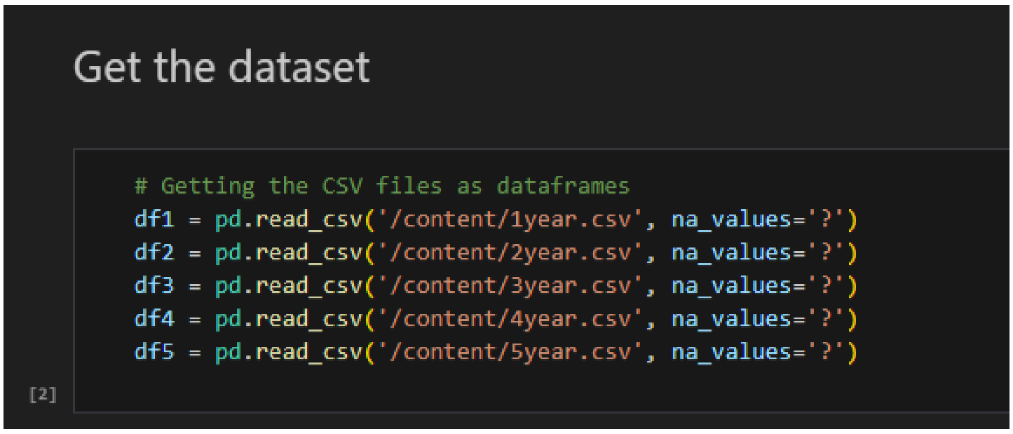
```
# Data manipulation libraries
import pandas as pd
import numpy as np

# Data visualization libraries
import matplotlib.pyplot as plt
import seaborn as sns

# Data preprocessing libraries
from sklearn.feature_selection import RFE
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE

# Classifying algorithm libraries
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier

# Evaluation metrics libraries
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report

# Model exporting libraries
import pickle
```

These imported libraries and modules serve as the building blocks of our bankruptcy prediction model, providing the tools and functionalities needed for data analysis, model development, and performance evaluation.

Back to Index

Read the Dataset

In this code snippet, we make use of the powerful Pandas library and its `read_csv` function to import a set of distinct CSV files, each denoted by '1year.csv,' '2year.csv,' '3year.csv,' '4year.csv,' and '5year.csv.' These CSV files contain structured data that we aim to analyze. To ensure accurate data processing, we apply a specific data handling technique by setting 'na_values' to '?'—this instructs Pandas to interpret question marks as missing data, allowing us to maintain data integrity during the import process. This step is particularly important for reliable and consistent data analysis.

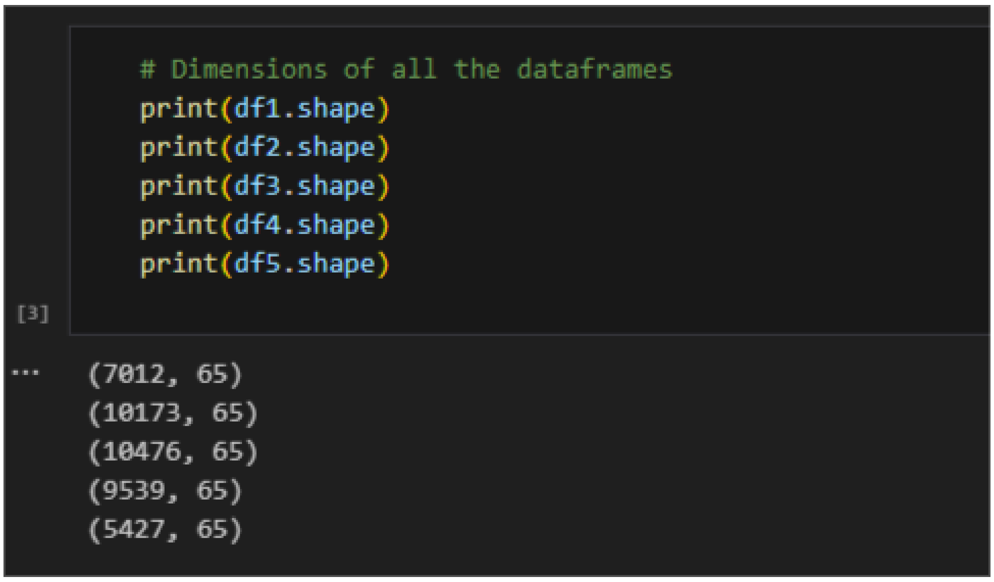


Get the dataset

```
# Getting the CSV files as dataframes
df1 = pd.read_csv('/content/1year.csv', na_values='?')
df2 = pd.read_csv('/content/2year.csv', na_values='?')
df3 = pd.read_csv('/content/3year.csv', na_values='?')
df4 = pd.read_csv('/content/4year.csv', na_values='?')
df5 = pd.read_csv('/content/5year.csv', na_values='?')
```

[2]

By subsequently printing the dimensions (number of rows and columns) of each dataframe, we gain valuable insights into the size and structure of the datasets, which serves as a foundational step in our data exploration and analysis journey.

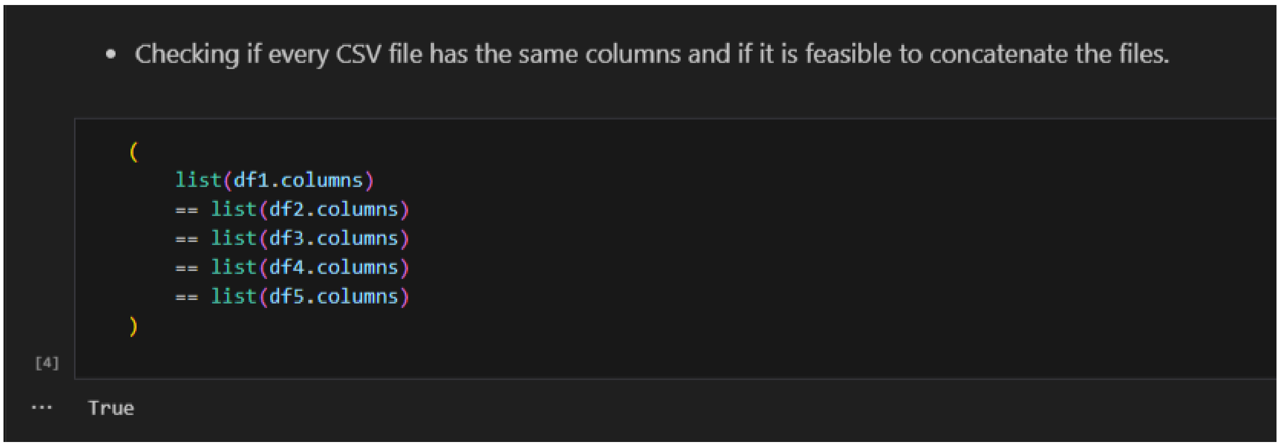


```
# Dimensions of all the dataframes
print(df1.shape)
print(df2.shape)
print(df3.shape)
print(df4.shape)
print(df5.shape)
```

[3]

```
(7012, 65)
(10173, 65)
(10476, 65)
(9539, 65)
(5427, 65)
```

We verify whether all five provided dataframes share the same attributes and target variable. If they do, we can concatenate these datasets into a unified large dataframe, simplifying our data management and analysis.

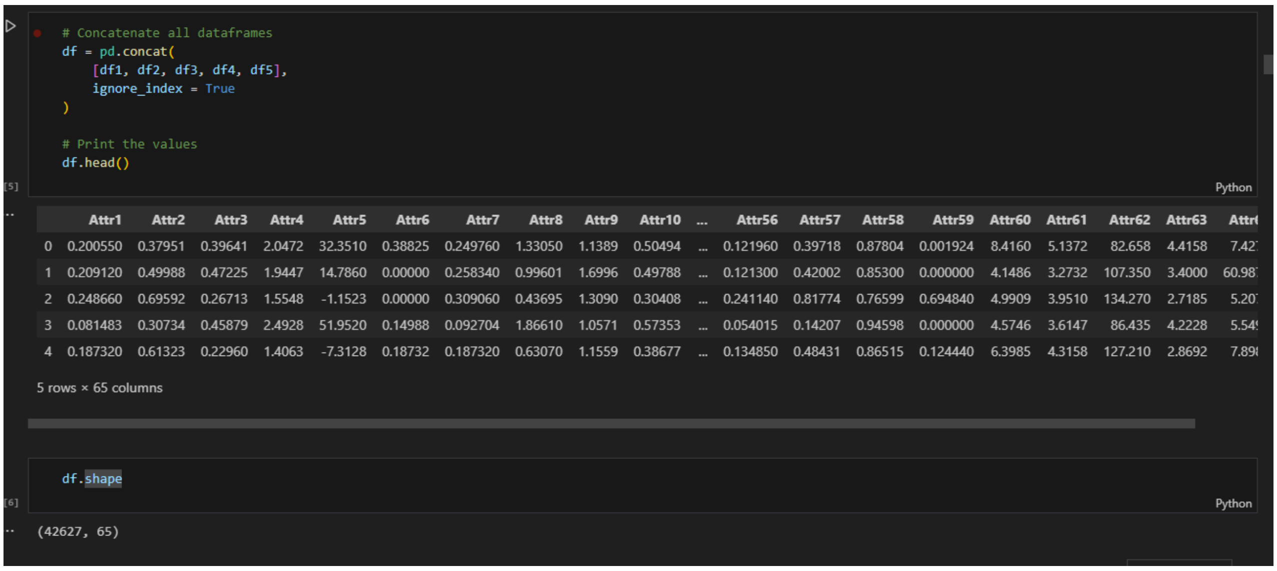


- Checking if every CSV file has the same columns and if it is feasible to concatenate the files.

```
[4]
(
    list(df1.columns)
    == list(df2.columns)
    == list(df3.columns)
    == list(df4.columns)
    == list(df5.columns)
)
```

```
True
```

We consolidate the five datasets into a single dataset, given that they share identical properties and a common target variable. This consolidation enables more extensive analysis on a larger dataset, enhancing the depth of our observations.



```
[5]
# Concatenate all dataframes
df = pd.concat(
    [df1, df2, df3, df4, df5],
    ignore_index = True
)

# Print the values
df.head()
```

	Attr1	Attr2	Attr3	Attr4	Attr5	Attr6	Attr7	Attr8	Attr9	Attr10	...	Attr56	Attr57	Attr58	Attr59	Attr60	Attr61	Attr62	Attr63	Attr
0	0.200550	0.37951	0.39641	2.0472	32.3510	0.38825	0.249760	1.33050	1.1389	0.50494	...	0.121960	0.39718	0.87804	0.001924	8.4160	5.1372	82.658	4.4158	7.42
1	0.209120	0.49988	0.47225	1.9447	14.7860	0.00000	0.258340	0.99601	1.6996	0.49788	...	0.121300	0.42002	0.85300	0.000000	4.1486	3.2732	107.350	3.4000	60.98
2	0.248660	0.69592	0.26713	1.5548	-1.1523	0.00000	0.309060	0.43695	1.3090	0.30408	...	0.241140	0.81774	0.76599	0.694840	4.9909	3.9510	134.270	2.7185	5.20
3	0.081483	0.30734	0.45879	2.4928	51.9520	0.14988	0.092704	1.86610	1.0571	0.57353	...	0.054015	0.14207	0.94598	0.000000	4.5746	3.6147	86.435	4.2228	5.54
4	0.187320	0.61323	0.22960	1.4063	-7.3128	0.18732	0.187320	0.63070	1.1559	0.38677	...	0.134850	0.48431	0.86515	0.124440	6.3985	4.3158	127.210	2.8692	7.89

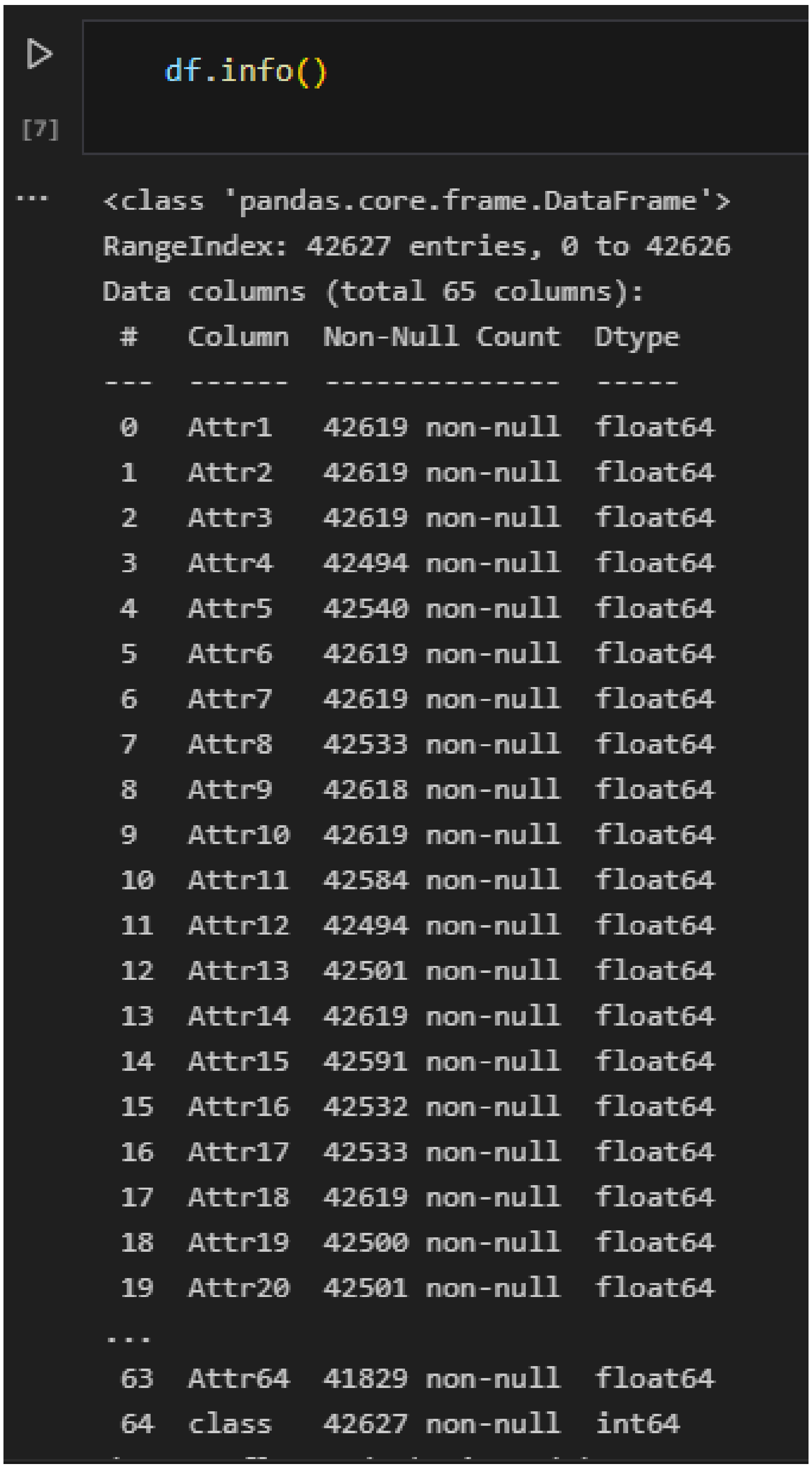
5 rows × 65 columns

```
[6]
df.shape
```

```
(42627, 65)
```

Data Preprocessing

The `df.info()` method provides a concise summary of our dataframe, including the number of non-null entries in each column, data types, and memory usage. It offers valuable insights into the dataset's composition, enabling quick assessment of missing data and an overview of column types. This function is a fundamental tool for initial data exploration and quality check.

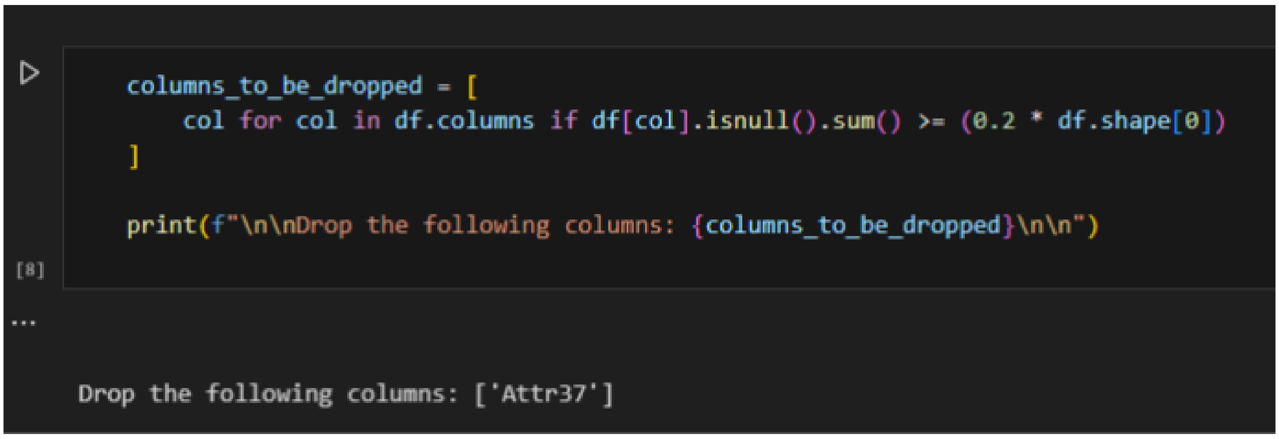


```
[7]
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 42627 entries, 0 to 42626
Data columns (total 65 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   Attr1    42619 non-null  float64
 1   Attr2    42619 non-null  float64
 2   Attr3    42619 non-null  float64
 3   Attr4    42494 non-null  float64
 4   Attr5    42540 non-null  float64
 5   Attr6    42619 non-null  float64
 6   Attr7    42619 non-null  float64
 7   Attr8    42533 non-null  float64
 8   Attr9    42618 non-null  float64
 9   Attr10   42619 non-null  float64
 10  Attr11   42584 non-null  float64
 11  Attr12   42494 non-null  float64
 12  Attr13   42501 non-null  float64
 13  Attr14   42619 non-null  float64
 14  Attr15   42591 non-null  float64
 15  Attr16   42532 non-null  float64
 16  Attr17   42533 non-null  float64
 17  Attr18   42619 non-null  float64
 18  Attr19   42500 non-null  float64
 19  Attr20   42501 non-null  float64
...
 63  Attr64   41829 non-null  float64
 64  class    42627 non-null  int64
```

Back to Index

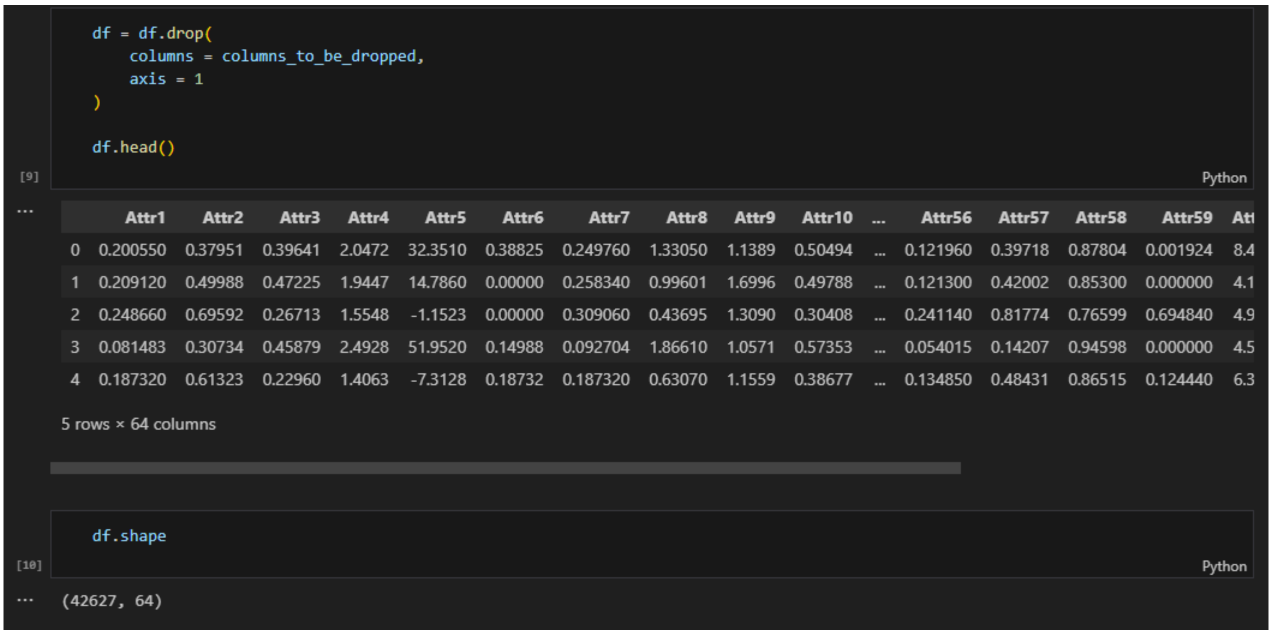
We identify and prepare to drop columns from our Pandas dataframe, 'df,' that exhibit a significant amount of missing data. We create a list called 'columns_to_be_dropped' using a list comprehension, which iterates through the columns of 'df.' For each column, it checks the number of missing values using `df[col].isnull().sum()` and compares it to a threshold of 20% of the total number of rows in 'df,' calculated as (0.2 * `df.shape[0]`). If a column has missing values equal to or exceeding this threshold, it is added to the 'columns_to_be_dropped' list.



```
[8]
columns_to_be_dropped = [
    col for col in df.columns if df[col].isnull().sum() >= (0.2 * df.shape[0])
]

print(f"\n\nDrop the following columns: {columns_to_be_dropped}\n\n")
```

```
...

Drop the following columns: ['Attr37']
```

We eliminate designated columns from the Pandas dataframe 'df' using the 'drop' method. 'columns_to_be_dropped' contains the column names to be removed. This action enhances data quality and streamlines processing.



```
[9]
df = df.drop(
    columns = columns_to_be_dropped,
    axis = 1
)

df.head()
```

Python

	Attr1	Attr2	Attr3	Attr4	Attr5	Attr6	Attr7	Attr8	Attr9	Attr10	...	Attr56	Attr57	Attr58	Attr59	Att
0	0.200550	0.37951	0.39641	2.0472	32.3510	0.38825	0.249760	1.33050	1.1389	0.50494	...	0.121960	0.39718	0.87804	0.001924	8.4
1	0.209120	0.49988	0.47225	1.9447	14.7860	0.00000	0.258340	0.99601	1.6996	0.49788	...	0.121300	0.42002	0.85300	0.000000	4.1
2	0.248660	0.69592	0.26713	1.5548	-1.1523	0.00000	0.309060	0.43695	1.3090	0.30408	...	0.241140	0.81774	0.76599	0.694840	4.9
3	0.081483	0.30734	0.45879	2.4928	51.9520	0.14988	0.092704	1.86610	1.0571	0.57353	...	0.054015	0.14207	0.94598	0.000000	4.5
4	0.187320	0.61323	0.22960	1.4063	-7.3128	0.18732	0.187320	0.63070	1.1559	0.38677	...	0.134850	0.48431	0.86515	0.124440	6.3

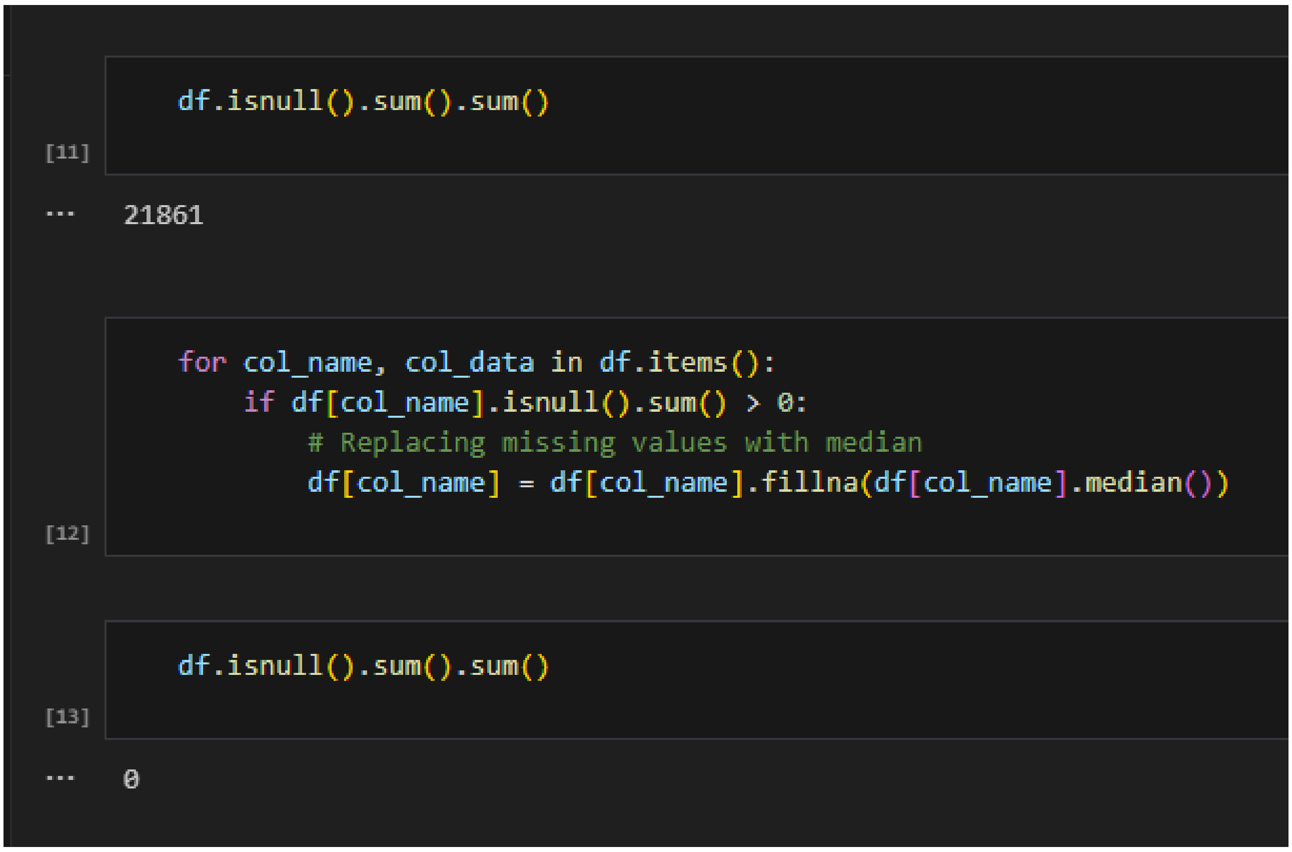
5 rows × 64 columns

```
[10]
df.shape
```

Python

```
...  (42627, 64)
```

Ensuring data integrity is a fundamental step in data preprocessing. Replacing missing values is crucial to prevent biases that can affect statistical analyses and lead to erroneous conclusions. Imputing missing values allows us to maximize the utility of available data, facilitating robust modelling and decision-making while minimizing data loss. To identify and quantify missing data, we employ the `df.isnull()` function. The number of missing values in each column is determined using `df.isnull().sum()`, and the total count of missing values across the entire dataset is computed with `df.isnull().sum().sum()`. To address missing values, we opt for imputing with the median of each column. The median is a robust measure of central tendency, less susceptible to the influence of outliers. Its position as the middle value when data is sorted makes it a reliable choice for imputation, enhancing data quality and analysis accuracy.



```
[11]  df.isnull().sum().sum()

...   21861
```

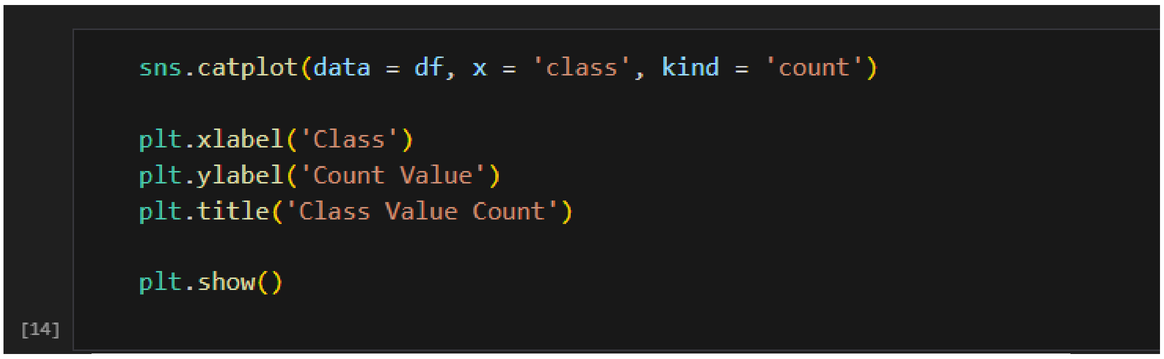
```
[12]  for col_name, col_data in df.items():
          if df[col_name].isnull().sum() > 0:
              # Replacing missing values with median
              df[col_name] = df[col_name].fillna(df[col_name].median())
```

```
[13]  df.isnull().sum().sum()

...   0
```

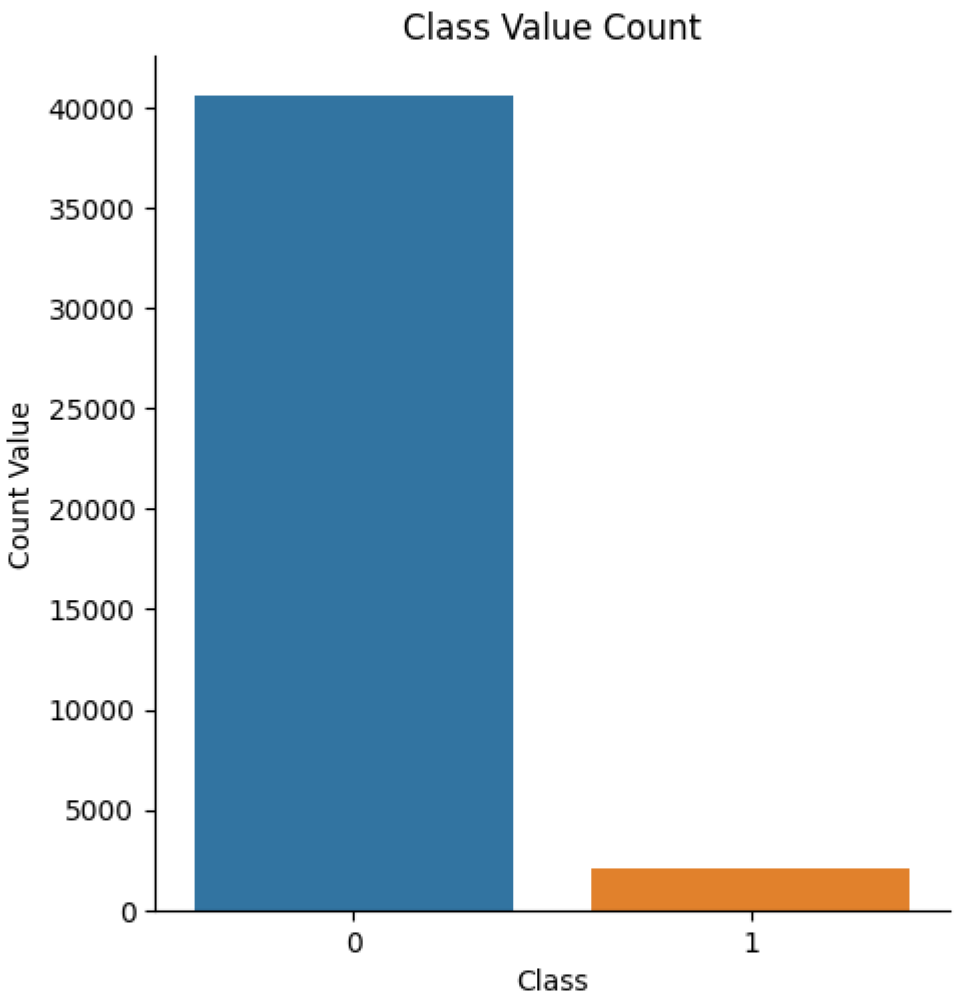
Data Visualization

Visual analysis is the process of exploring and comprehending data via the use of visual representations such as charts, plots, and graphs. It is a method for quickly identifying patterns, trends, and outliers in data, which can aid in gaining insights and making sound decisions.

This code employs Seaborn to create a categorical plot that visualizes the distribution of classes in the 'class' column in the dataset, with the x-axis representing the 'class' variable and the y-axis showing count values. The graph highlights a significant class imbalance, with class 0 dominating the dataset. This imbalance underscores the importance of implementing data balancing strategies to ensure fair and accurate model training and predictions, effectively addressing the challenges posed by imbalanced data and enhancing overall model performance.

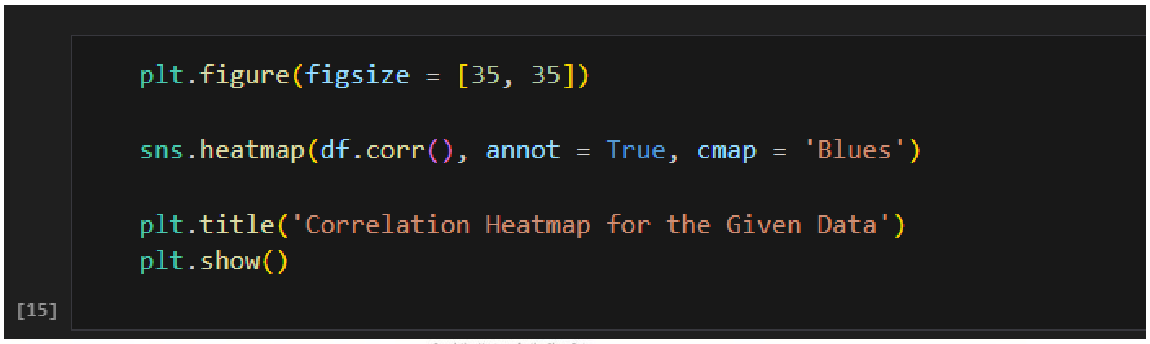


```
sns.catplot(data = df, x = 'class', kind = 'count')

plt.xlabel('Class')
plt.ylabel('Count Value')
plt.title('Class Value Count')

plt.show()
[14]
```



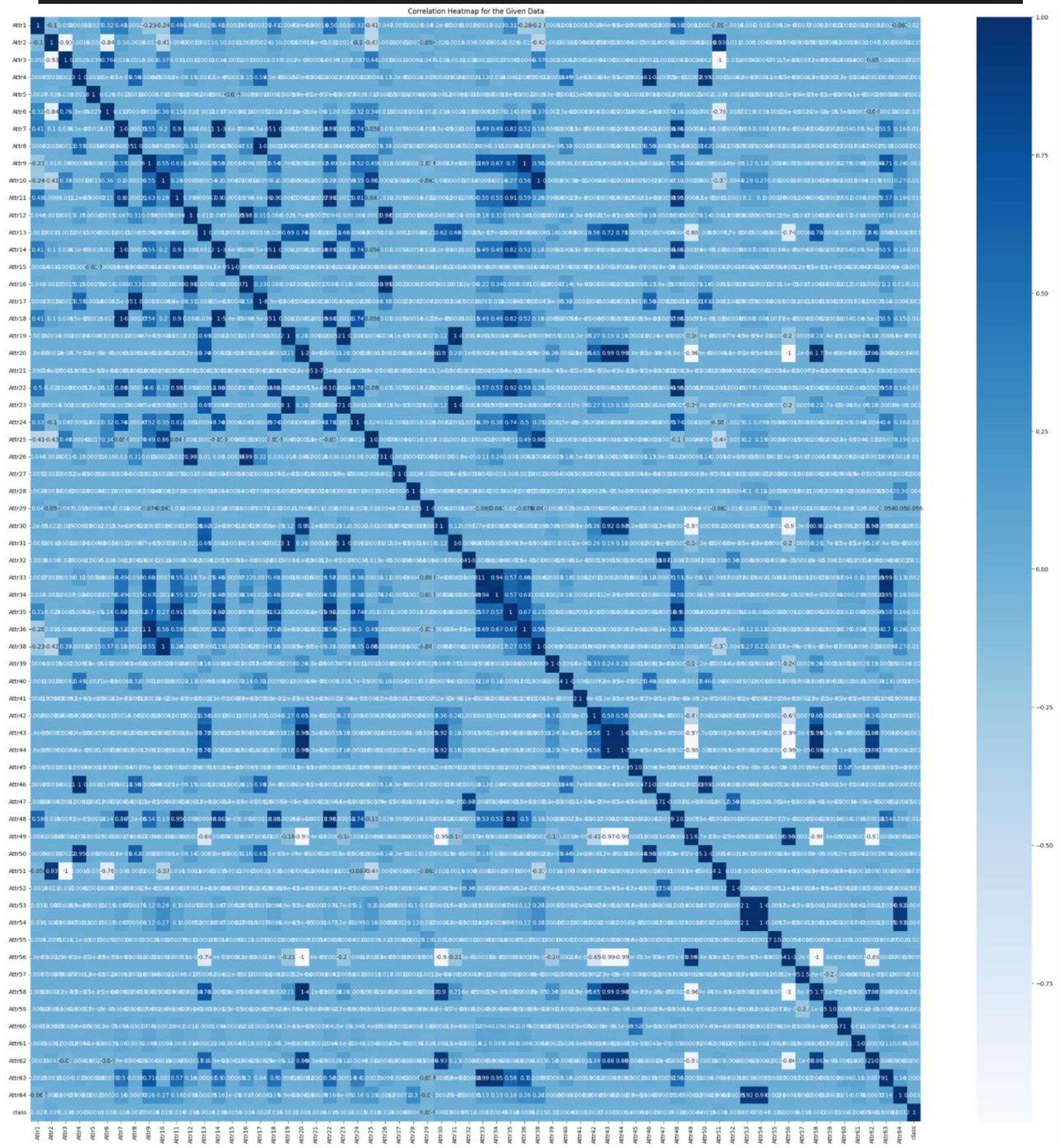
Back to Index

Next, we generate a heatmap using Matplotlib and Seaborn to visualize the correlation between variables in the dataset. The 'figsize' parameter defines the size of the plot, and the heatmap is annotated to display correlation values. The colour map 'Blues' is applied to represent the strength of correlations, with darker shades indicating stronger relationships. This heatmap serves as a valuable tool for exploring how variables interact with each other, assisting in feature selection and understanding the data's underlying patterns.

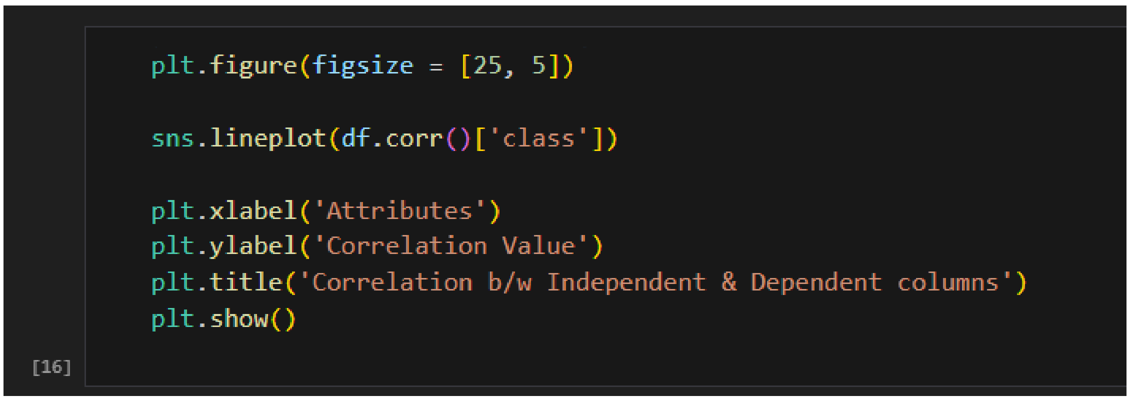


```
plt.figure(figsize = [35, 35])

sns.heatmap(df.corr(), annot = True, cmap = 'Blues')

plt.title('Correlation Heatmap for the Given Data')
plt.show()
```

[15]

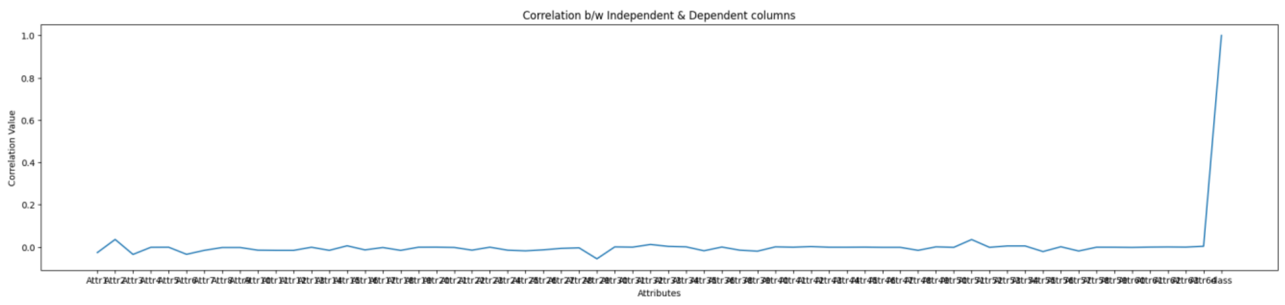


A line plot is created using Matplotlib and Seaborn to visualize the correlation between independent attributes and the dependent 'class' variable. The 'figsize' parameter determines the plot's dimensions, and the 'sns.lineplot' function is used to display the correlations. The x-axis represents the attributes, the y-axis shows the correlation values. This plot provides insights into the relationships between individual attributes and the target variable, aiding in feature selection and understanding the impact of attributes on the target variable.



```
plt.figure(figsize = [25, 5])

sns.lineplot(df.corr()['class'])

plt.xlabel('Attributes')
plt.ylabel('Correlation Value')
plt.title('Correlation b/w Independent & Dependent columns')
plt.show()
```

[16]

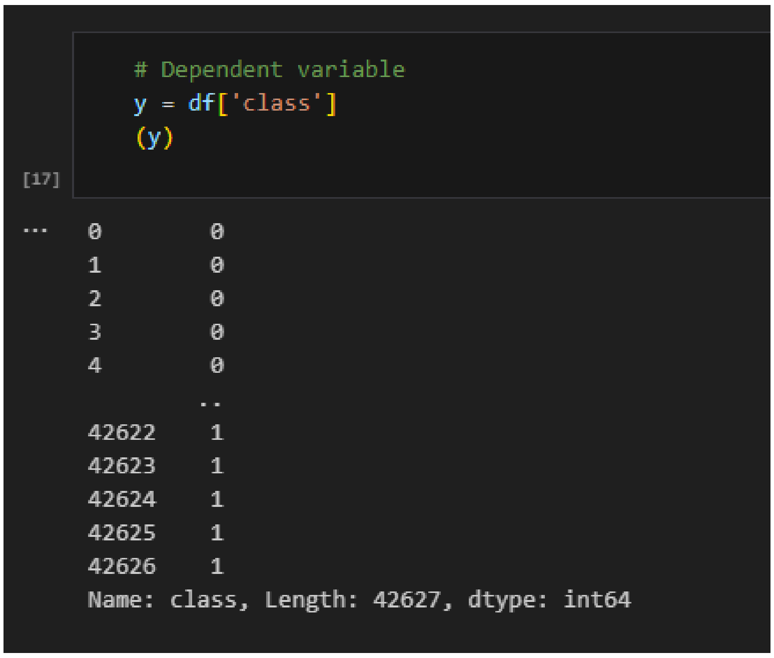


Back to Index

X and Y Splitting

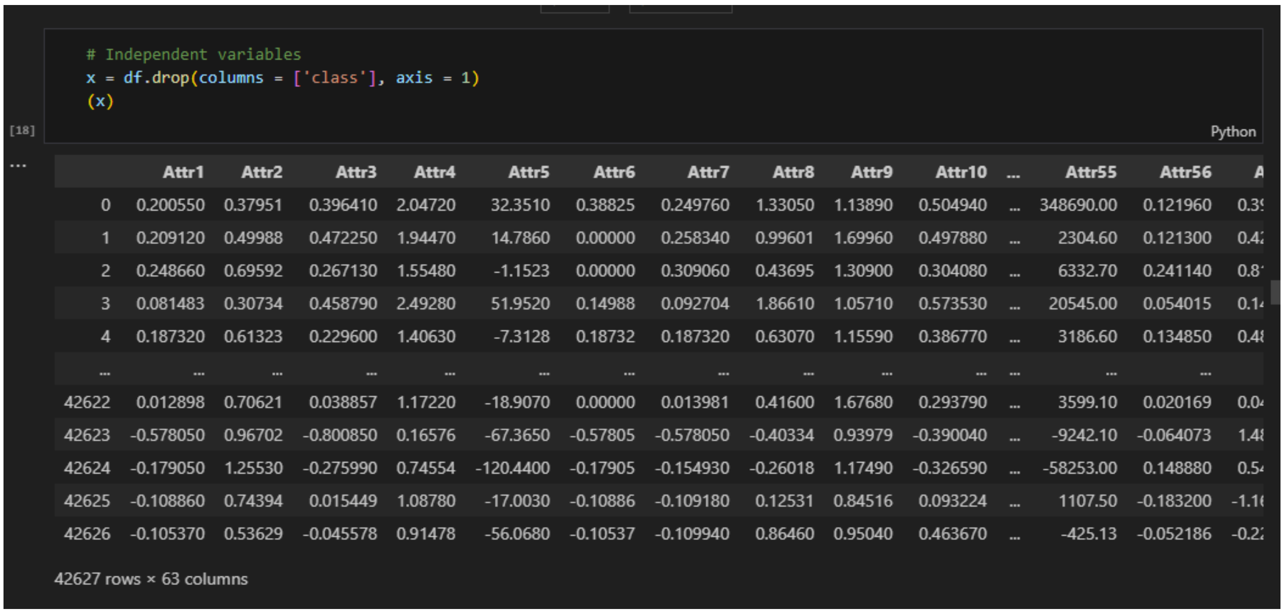
In machine learning, an "x-y split" is a crucial step that separates a dataset into two distinct components: the independent variables (often denoted as 'X') and the dependent variable or target (typically denoted as 'Y'). The 'X' component comprises the input features or attributes that the model uses to make predictions, while 'Y' represents the variable the model aims to predict. This separation is essential for model training, as it allows the algorithm to learn from the relationships between 'X' and 'Y' and later make accurate predictions based on unseen 'X' values. It also facilitates rigorous model evaluation by testing its ability to generalize to new data points.

The dependent variable, 'class,' is assigned to 'y,' while the independent variables are assigned to 'x' after excluding the 'class' column. This division simplifies the process of training and validating models, preventing data leakage and enabling rigorous testing of a model's generalization to new, unseen data. It ensures precise predictions and reliable performance evaluation, a fundamental practice in machine learning and data analysis.



```
[17]
# Dependent variable
y = df['class']
(y)
```

```
0        0
1        0
2        0
3        0
4        0
        ..
42622    1
42623    1
42624    1
42625    1
42626    1
Name: class, Length: 42627, dtype: int64
```



```
[18]
# Independent variables
x = df.drop(columns = ['class'], axis = 1)
(x)
```

	Attr1	Attr2	Attr3	Attr4	Attr5	Attr6	Attr7	Attr8	Attr9	Attr10	...	Attr55	Attr56	A
0	0.200550	0.37951	0.396410	2.04720	32.3510	0.38825	0.249760	1.33050	1.13890	0.504940	...	348690.00	0.121960	0.3
1	0.209120	0.49988	0.472250	1.94470	14.7860	0.00000	0.258340	0.99601	1.69960	0.497880	...	2304.60	0.121300	0.4
2	0.248660	0.69592	0.267130	1.55480	-1.1523	0.00000	0.309060	0.43695	1.30900	0.304080	...	6332.70	0.241140	0.8
3	0.081483	0.30734	0.458790	2.49280	51.9520	0.14988	0.092704	1.86610	1.05710	0.573530	...	20545.00	0.054015	0.1
4	0.187320	0.61323	0.229600	1.40630	-7.3128	0.18732	0.187320	0.63070	1.15590	0.386770	...	3186.60	0.134850	0.4
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
42622	0.012898	0.70621	0.038857	1.17220	-18.9070	0.00000	0.013981	0.41600	1.67680	0.293790	...	3599.10	0.020169	0.0
42623	-0.578050	0.96702	-0.800850	0.16576	-67.3650	-0.57805	-0.578050	-0.40334	0.93979	-0.390040	...	-9242.10	-0.064073	1.4
42624	-0.179050	1.25530	-0.275990	0.74554	-120.4400	-0.17905	-0.154930	-0.26018	1.17490	-0.326590	...	-58253.00	0.148880	0.5
42625	-0.108860	0.74394	0.015449	1.08780	-17.0030	-0.10886	-0.109180	0.12531	0.84516	0.093224	...	1107.50	-0.183200	-1.1
42626	-0.105370	0.53629	-0.045578	0.91478	-56.0680	-0.10537	-0.109940	0.86460	0.95040	0.463670	...	-425.13	-0.052186	-0.2

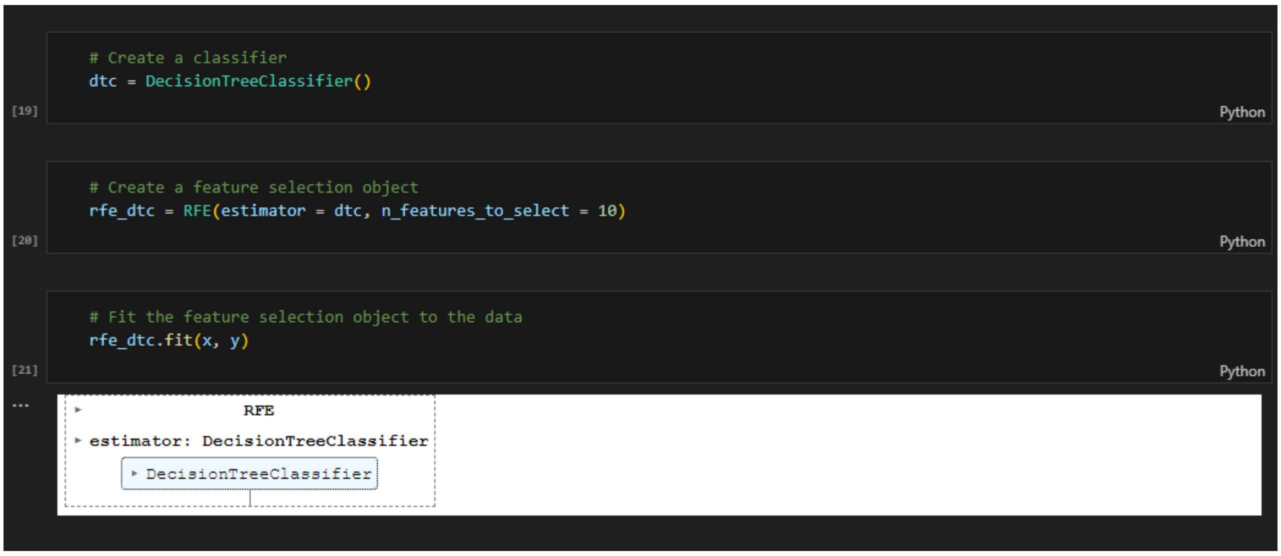
42627 rows × 63 columns

Feature Selection

Feature selection plays a pivotal role in elevating the performance of machine learning models. It offers several advantages, including dimensionality reduction, faster model training, and improved generalization. By singling out the most informative features, it effectively reduces noise and guards against overfitting. Additionally, feature selection enhances model interpretability, concentrating on the most relevant attributes, which, in turn, aids in better decision-making and deeper data insights. In sum, it streamlines the modelling process, yielding more precise and efficient results.

We utilize three machine learning classifiers – Decision Tree, Random Forest, and XGBoost – in combination with feature selection techniques to enhance model performance. Each classifier is paired with a feature selection object utilizing the Recursive Feature Elimination (RFE) method to identify the top ten influential features. By fitting these objects to input features (X) and the target variable (Y), we systematically uncover the most impactful attributes. This simplifies the dataset, improves interpretability, and often leads to higher model accuracy. Each classifier, with feature selection, supports data-driven decision-making and delivers more efficient model results.

Decision Tree Classifier



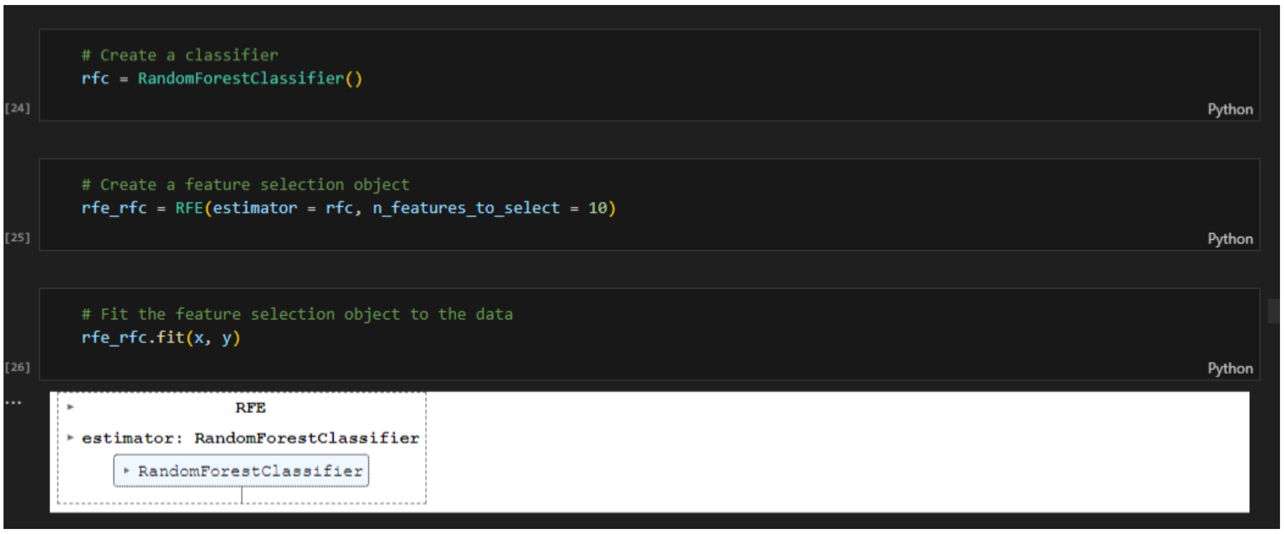
```
# Create a classifier
dtc = DecisionTreeClassifier()
```
[19] Python

```
# Create a feature selection object
rfe_dtc = RFE(estimator = dtc, n_features_to_select = 10)
```
[20] Python

```
# Fit the feature selection object to the data
rfe_dtc.fit(x, y)
```
[21] Python

```
RFE
estimator: DecisionTreeClassifier
DecisionTreeClassifier
```

Random Forest Classifier



```
# Create a classifier
rfc = RandomForestClassifier()
```
[24] Python

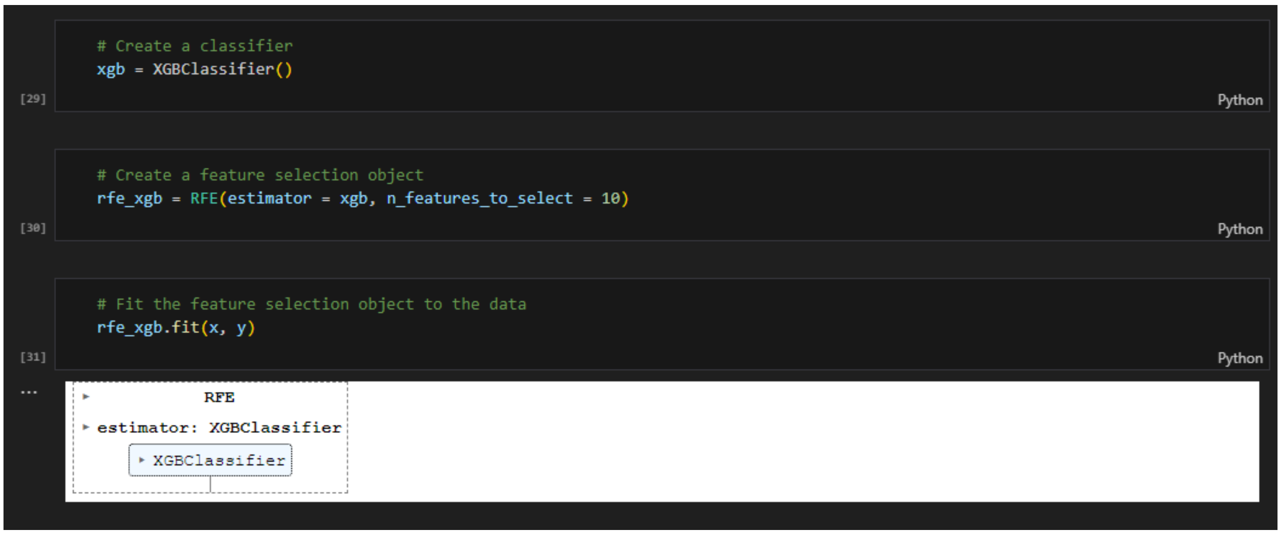
```
# Create a feature selection object
rfe_rfc = RFE(estimator = rfc, n_features_to_select = 10)
```
[25] Python

```
# Fit the feature selection object to the data
rfe_rfc.fit(x, y)
```
[26] Python

```
RFE
estimator: RandomForestClassifier
RandomForestClassifier
```

Back to Index

Extreme Gradient Boosting Classifier



```
[29]
# Create a classifier
xgb = XGBClassifier()
```

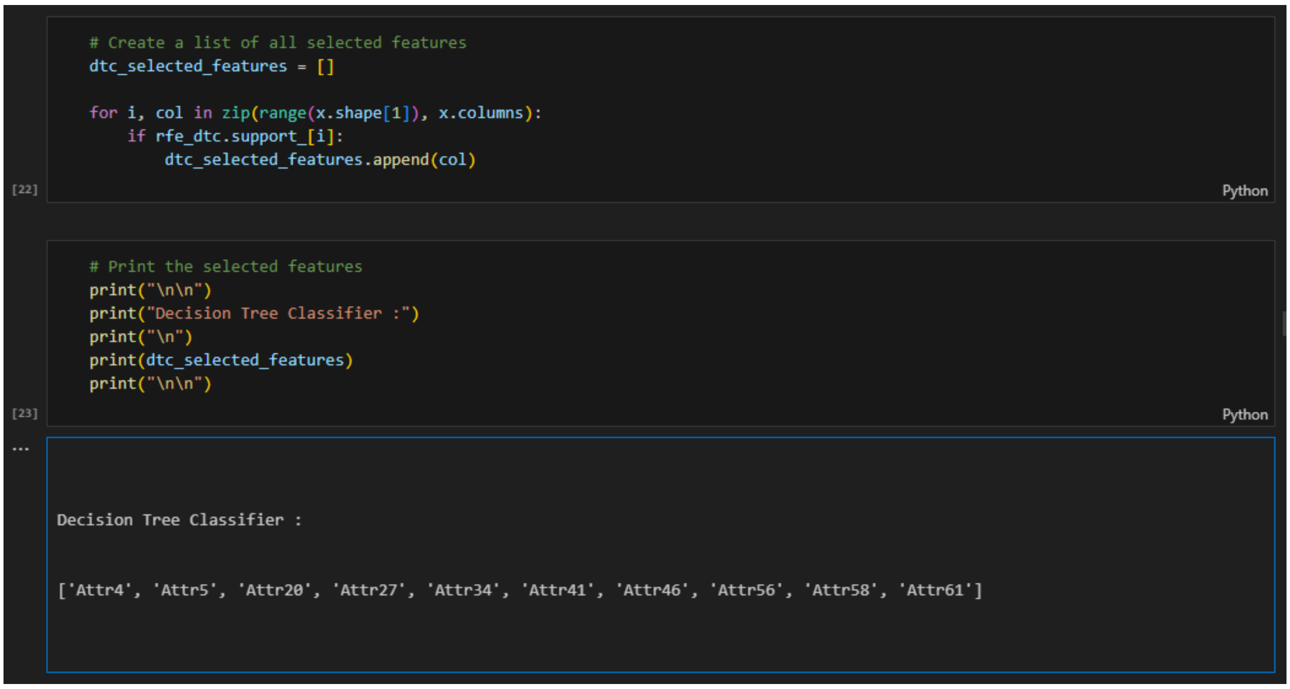
```
[30]
# Create a feature selection object
rfe_xgb = RFE(estimator = xgb, n_features_to_select = 10)
```

```
[31]
# Fit the feature selection object to the data
rfe_xgb.fit(x, y)
```

```
RFE
estimator: XGBClassifier
XGBClassifier
```

We create lists to store selected features for each of the three machine learning classifiers – Decision Tree, Random Forest, and XGBoost. For each classifier, we iterate through the columns of the input features and check the Boolean values stored in the respective RFE support arrays (e.g., rfe_dtc.support_) to identify which features were selected as the top ten most influential. We then print the selected features for each classifier, allowing us to easily access and analyze the key attributes that significantly impact model performance. This process ensures transparency and assists in feature selection, contributing to more effective and efficient model outcomes.

Decision Tree Classifier



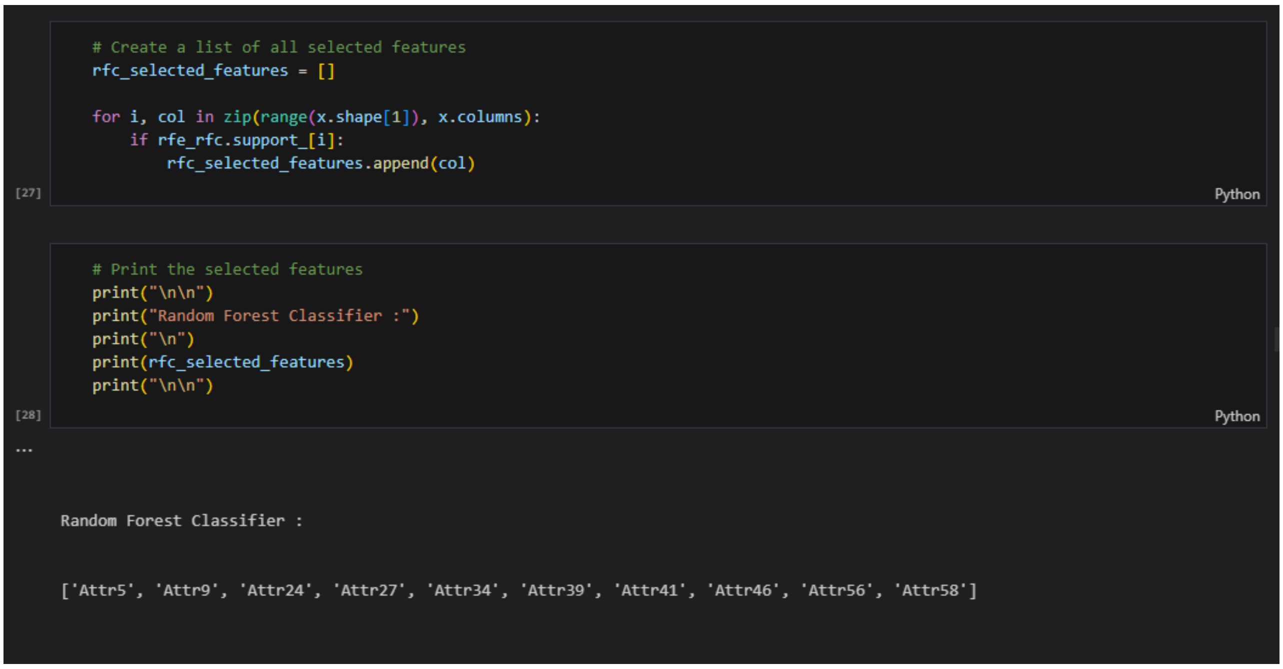
```
[22]
# Create a list of all selected features
dtc_selected_features = []

for i, col in zip(range(x.shape[1]), x.columns):
    if rfe_dtc.support_[i]:
        dtc_selected_features.append(col)
```

```
[23]
# Print the selected features
print("\n\n")
print("Decision Tree Classifier :")
print("\n")
print(dtc_selected_features)
print("\n\n")
```

```
Decision Tree Classifier :

['Attr4', 'Attr5', 'Attr20', 'Attr27', 'Attr34', 'Attr41', 'Attr46', 'Attr56', 'Attr58', 'Attr61']
```

Back to Index

Random Forest Classifier



```
# Create a list of all selected features
rfc_selected_features = []

for i, col in zip(range(x.shape[1]), x.columns):
    if rfe_rfc.support_[i]:
        rfc_selected_features.append(col)
```

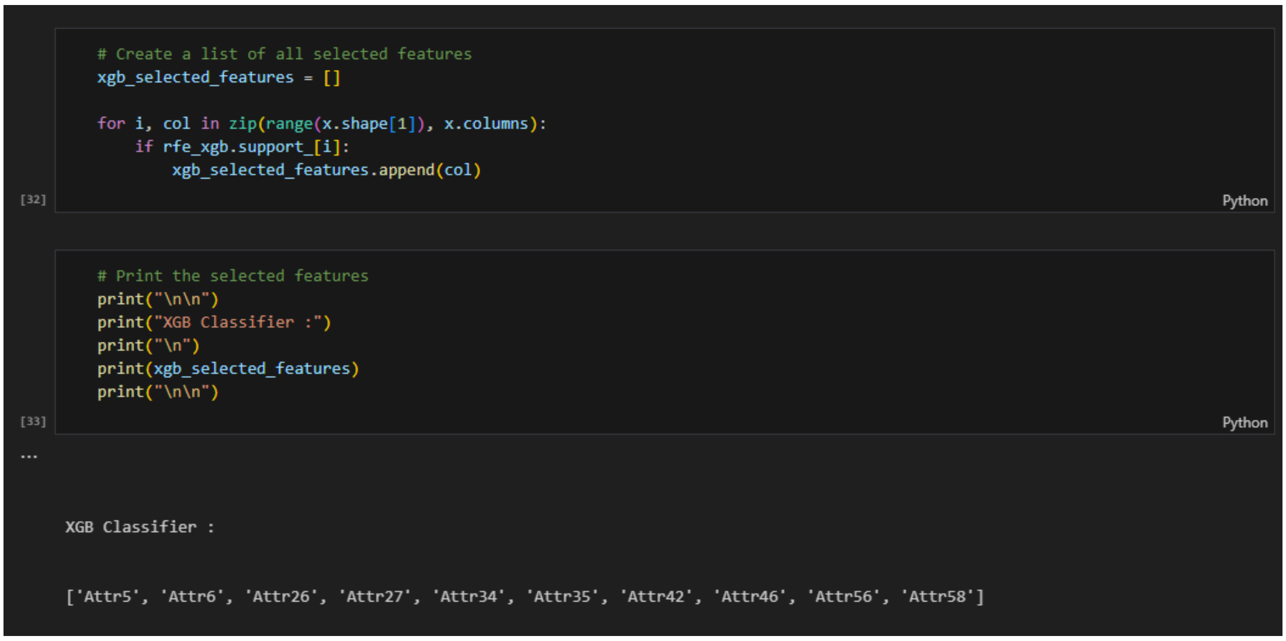
[27] Python

```
# Print the selected features
print("\n\n")
print("Random Forest Classifier :")
print("\n")
print(rfc_selected_features)
print("\n\n")
```

[28] Python

```
...

Random Forest Classifier :


['Attr5', 'Attr9', 'Attr24', 'Attr27', 'Attr34', 'Attr39', 'Attr41', 'Attr46', 'Attr56', 'Attr58']
```

Extreme Gradient Boosting Classifier



```
# Create a list of all selected features
xgb_selected_features = []

for i, col in zip(range(x.shape[1]), x.columns):
    if rfe_xgb.support_[i]:
        xgb_selected_features.append(col)
```

[32] Python

```
# Print the selected features
print("\n\n")
print("XGB Classifier :")
print("\n")
print(xgb_selected_features)
print("\n\n")
```

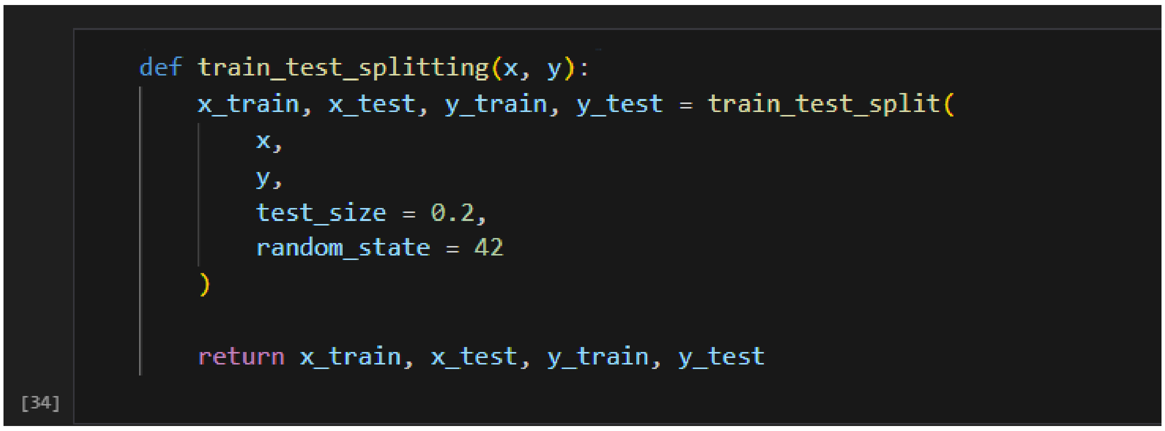
[33] Python

```
...

XGB Classifier :


['Attr5', 'Attr6', 'Attr26', 'Attr27', 'Attr34', 'Attr35', 'Attr42', 'Attr46', 'Attr56', 'Attr58']
```

Back to Index

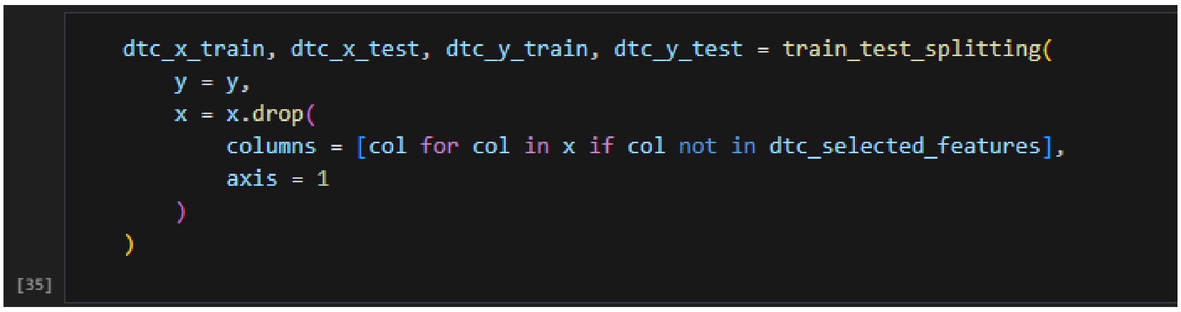
Train-test Split

We defined a custom function named `train_test_splitting` for splitting data into training and testing sets. It takes input features 'x' and target variables 'y' as input. Using the `train_test_split` function from Scikit-Learn, it divides the data into training and testing sets with a 20% test size and a specified random seed (42 in this case) to ensure reproducibility. The function returns the resulting training and testing data for both input features and target variables. This function simplifies the process of data splitting, a critical step in machine learning model development.



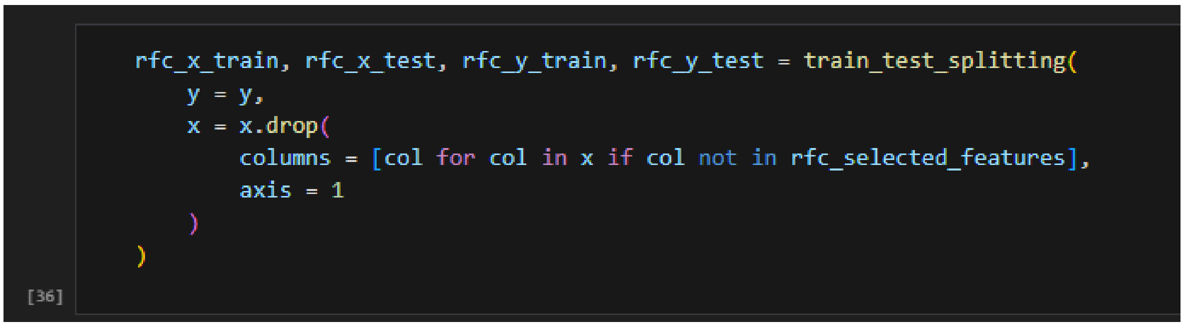
```
def train_test_splitting(x, y):
    x_train, x_test, y_train, y_test = train_test_split(
        x,
        y,
        test_size = 0.2,
        random_state = 42
    )

    return x_train, x_test, y_train, y_test
[34]
```

Decision Tree Classifier



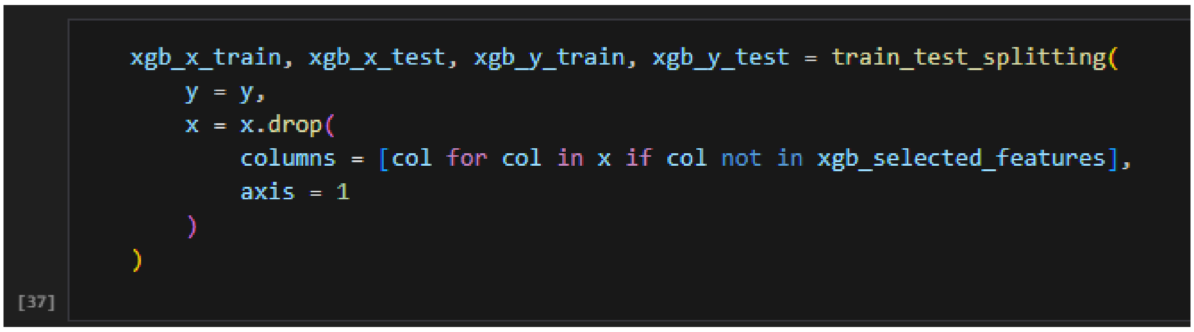
```
dtc_x_train, dtc_x_test, dtc_y_train, dtc_y_test = train_test_splitting(
    y = y,
    x = x.drop(
        columns = [col for col in x if col not in dtc_selected_features],
        axis = 1
    )
)
[35]
```

Random Forest Classifier



```
rfc_x_train, rfc_x_test, rfc_y_train, rfc_y_test = train_test_splitting(
    y = y,
    x = x.drop(
        columns = [col for col in x if col not in rfc_selected_features],
        axis = 1
    )
)
[36]
```

Extreme Gradient Boosting Classifier

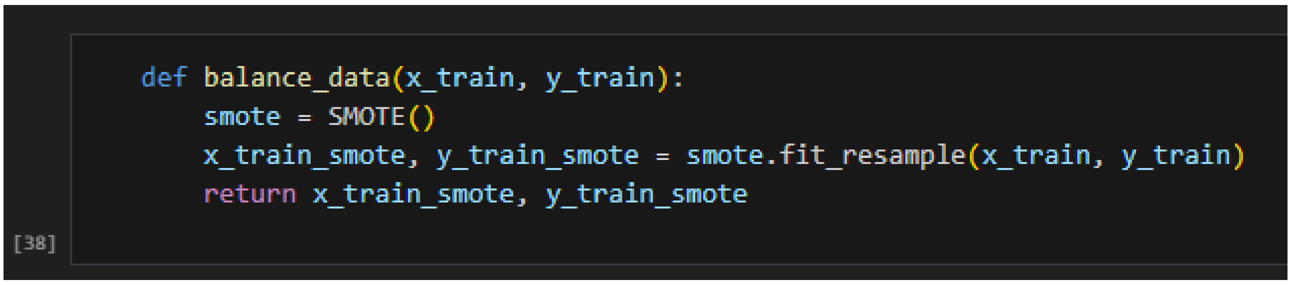


```
xgb_x_train, xgb_x_test, xgb_y_train, xgb_y_test = train_test_splitting(
    y = y,
    x = x.drop(
        columns = [col for col in x if col not in xgb_selected_features],
        axis = 1
    )
)
[37]
```

Back to Index

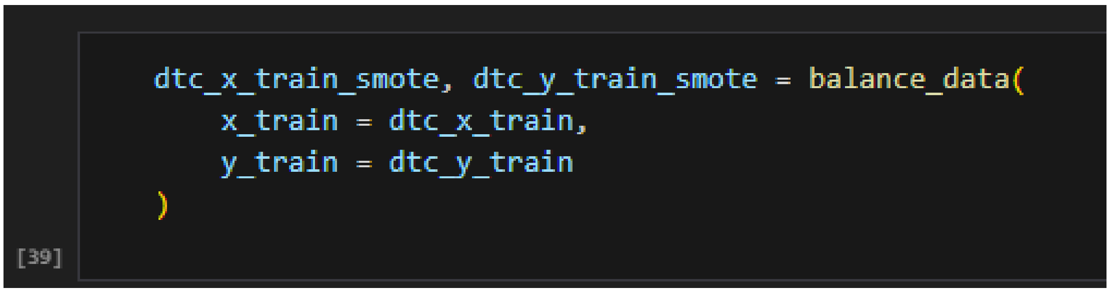
Handling Imbalanced Data

This code defines a function named `balance_data` used to balance the training data. It employs the Synthetic Minority Over-sampling Technique (SMOTE) to balance the class distribution. The input features 'x_train' and target variables 'y_train' are provided to the function. SMOTE is applied to create synthetic samples of the minority class to balance it with the majority class. The function returns the balanced training data, represented as 'x_train_smote' and 'y_train_smote. Balancing the data is crucial for addressing class imbalances and improving the model's ability to make accurate predictions.



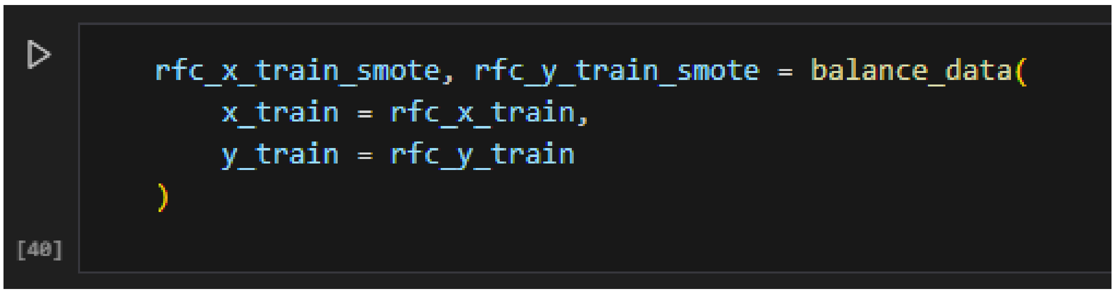
```python
def balance_data(x_train, y_train):
    smote = SMOTE()
    x_train_smote, y_train_smote = smote.fit_resample(x_train, y_train)
    return x_train_smote, y_train_smote
```

Decision Tree Classifier



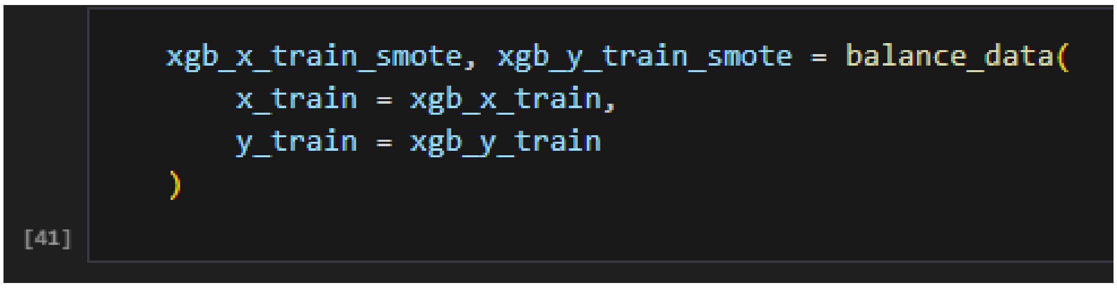
```python
dtc_x_train_smote, dtc_y_train_smote = balance_data(
    x_train = dtc_x_train,
    y_train = dtc_y_train
)
```

Random Forest Classifier



```python
rfc_x_train_smote, rfc_y_train_smote = balance_data(
    x_train = rfc_x_train,
    y_train = rfc_y_train
)
```

Extreme Gradient Boosting Classifier



```python
xgb_x_train_smote, xgb_y_train_smote = balance_data(
    x_train = xgb_x_train,
    y_train = xgb_y_train
)
```

Back to Index

Model Building

In machine learning, model building refers to the process of training a predictive algorithm or model with a labelled dataset. This entails giving the model features and target variables, allowing it to discover patterns and relationships in the data. Model construction frequently entails fine-tuning hyperparameters, selecting an effective algorithm, and optimizing its performance. The goal is to develop a robust and reliable predictive model capable of making educated predictions on fresh, previously unknown data, hence simplifying data-driven decision-making in a variety of applications.

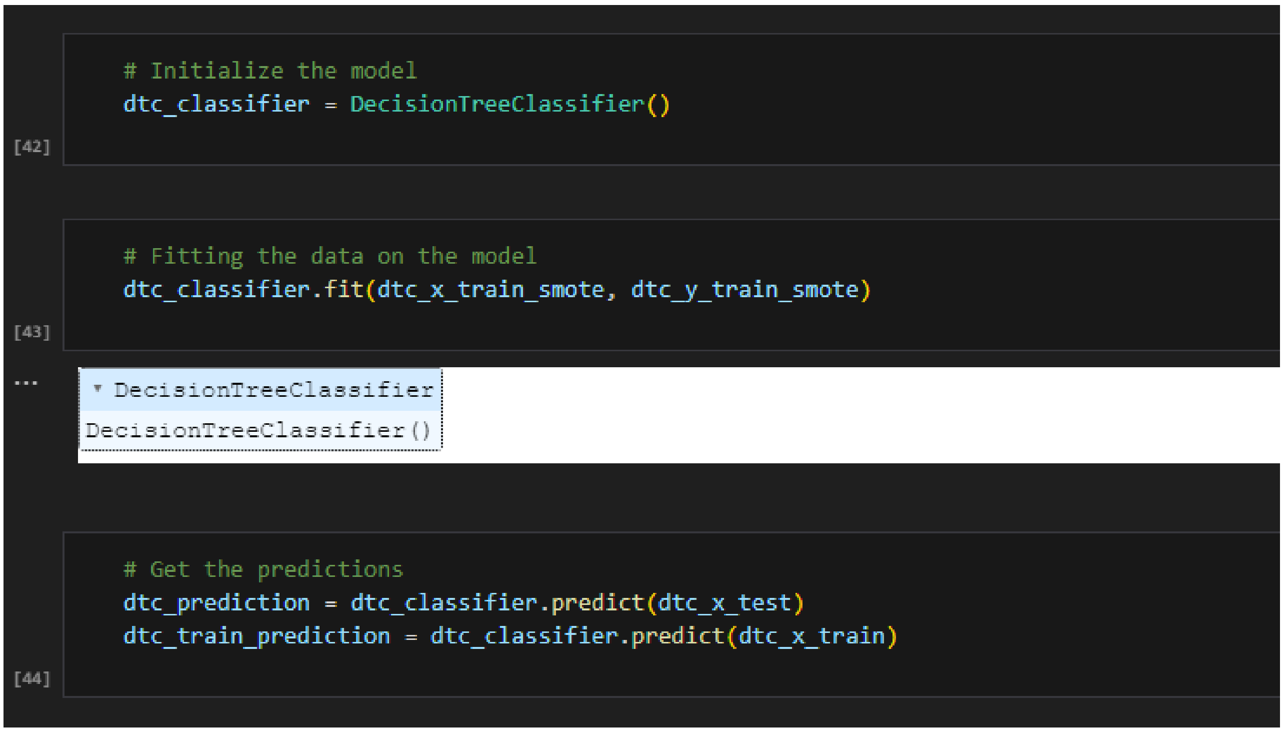
Decision Tree Classifier

The model is initialized by instantiating the `DecisionTreeClassifier` class, creating the `dtc_classifier` instance.

Next, the model is fitted with the training data. The training data, previously balanced using SMOTE, is provided as `dtc_x_train_smote` and `dtc_y_train_smote`. This fitting process involves training the model to recognize patterns and relationships within the data.

Subsequently, the trained model is employed to make predictions on two separate datasets. The first dataset, `dtc_x_test`, represents the test data used for evaluating the model's performance. Predictions are generated for this dataset and stored in the variable `dtc_prediction`. The second dataset, `dtc_x_train`, serves as the training data for reference, and predictions for this dataset are saved in the variable `dtc_train_prediction`.

These predictions can be further assessed to gauge the model's accuracy, performance, and its ability to generalize when applied to both the test and training data. This process is fundamental for evaluating the model's effectiveness and ensuring its suitability for the intended classification task.



```
[42]
# Initialize the model
dtc_classifier = DecisionTreeClassifier()
```

```
[43]
# Fitting the data on the model
dtc_classifier.fit(dtc_x_train_smote, dtc_y_train_smote)
```

```
...
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

```
[44]
# Get the predictions
dtc_prediction = dtc_classifier.predict(dtc_x_test)
dtc_train_prediction = dtc_classifier.predict(dtc_x_train)
```

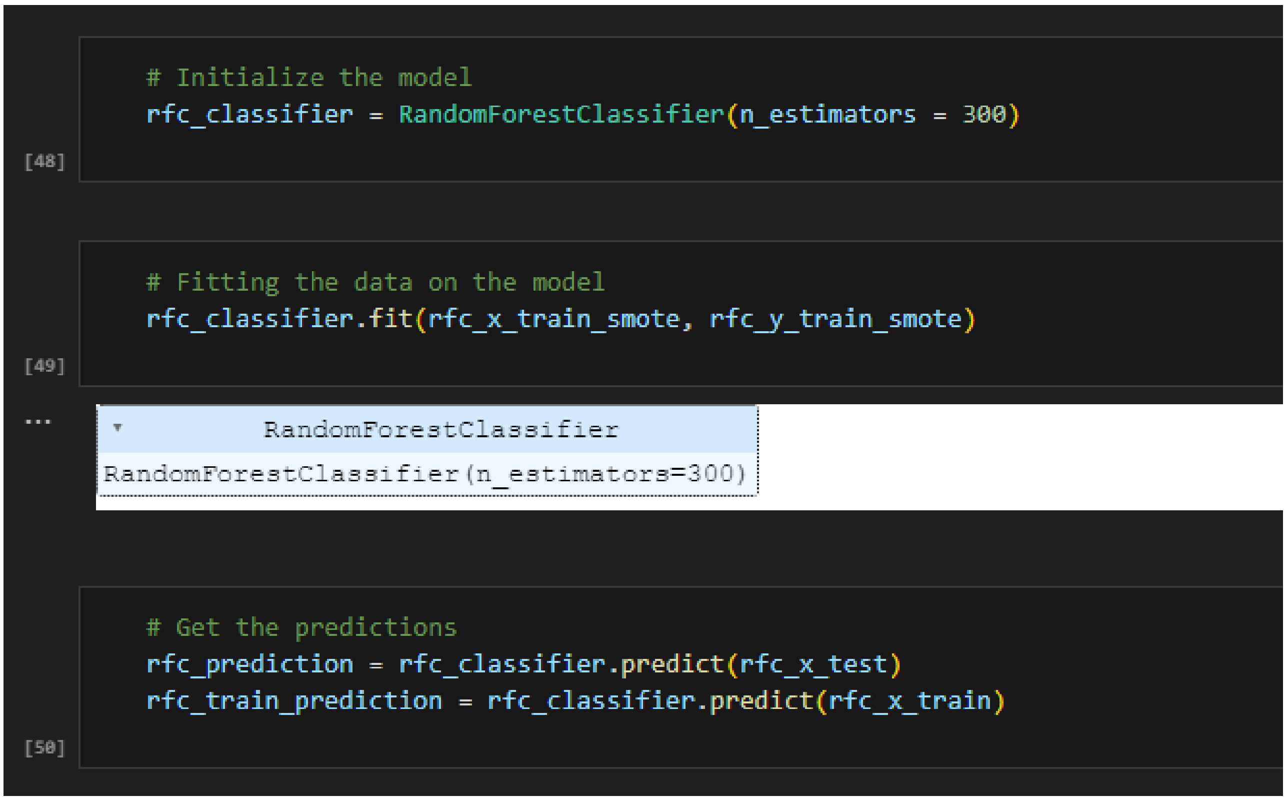
Random Forest Classifier

The model is initialized by creating an instance of the `RandomForestClassifier` class. This instance is configured with three hundred decision trees in the ensemble, denoted by the `n_estimators` parameter. The initialized model is assigned to the variable `rfc_classifier`.

Following that, the model is trained with the previously balanced training data, which underwent SMOTE balancing. The training data consists of two components: `rfc_x_train_smote`, representing the input features, and `rfc_y_train_smote`, representing the target variable. This training phase enables the model to learn from the provided data, capturing underlying patterns and relationships.

Subsequently, the trained Random Forest model is utilized to make predictions on two distinct datasets. The first dataset, `rfc_x_test`, acts as the test data, serving to evaluate the model's performance. Predictions for this dataset are stored in the variable `rfc_prediction`. The second dataset, `rfc_x_train`, serves as the training data for reference, and predictions for this dataset are recorded in the variable `rfc_train_prediction`.

These predictions play a vital role in assessing the model's accuracy, overall performance, and its capability to generalize to unseen data. This process is essential for evaluating the model's effectiveness and suitability for the binary classification task at hand.



```
# Initialize the model
rfc_classifier = RandomForestClassifier(n_estimators = 300)
[48]
```

```
# Fitting the data on the model
rfc_classifier.fit(rfc_x_train_smote, rfc_y_train_smote)
[49]
...
RandomForestClassifier
RandomForestClassifier(n_estimators=300)
```

```
# Get the predictions
rfc_prediction = rfc_classifier.predict(rfc_x_test)
rfc_train_prediction = rfc_classifier.predict(rfc_x_train)
[50]
```

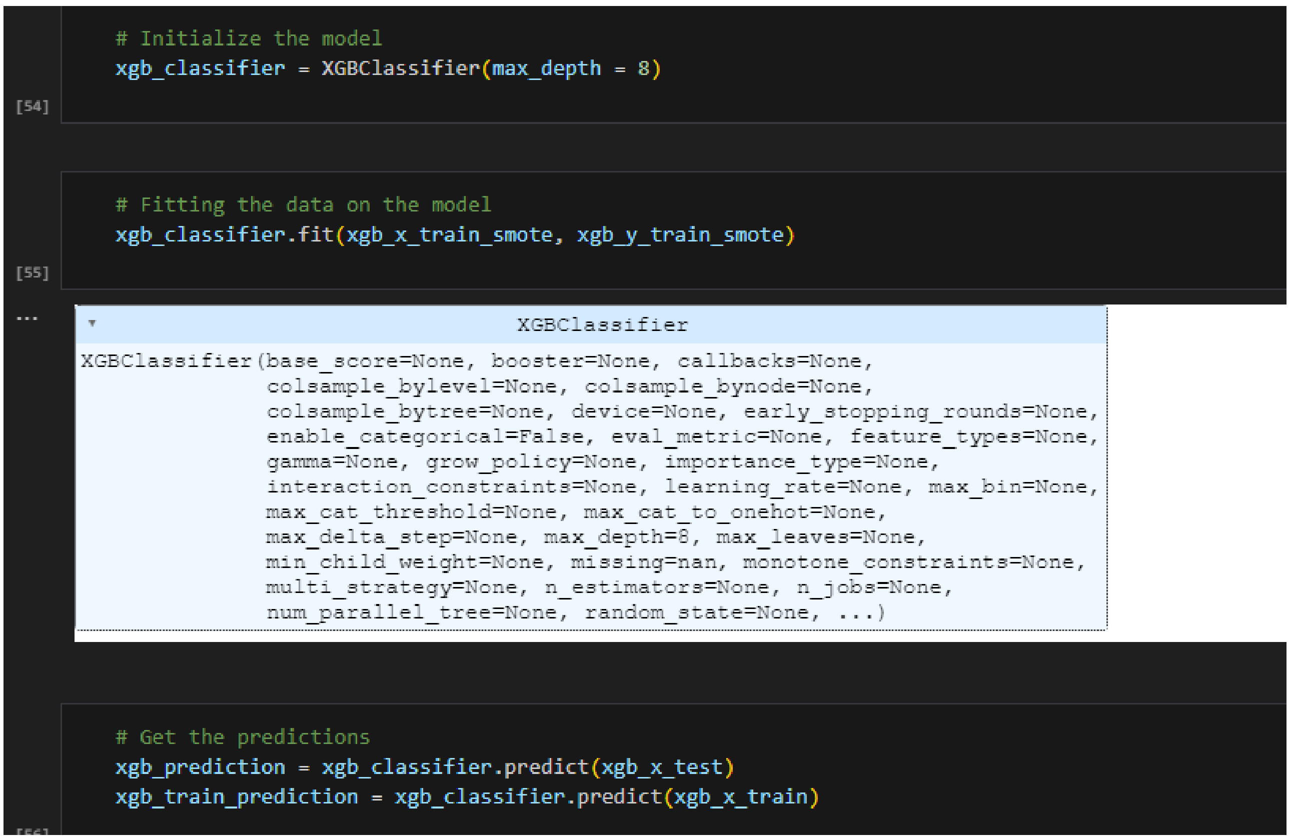
Extreme Gradient Boosting Classifier

The XGBoost Classifier model is initialized by creating an instance of the `XGBClassifier` class. In this instance, the `max_depth` parameter is configured to eight, controlling the maximum depth of decision trees within the ensemble. The initialized model is stored in the variable `xgb_classifier`.

Following the initialization, the model is trained using the provided training data, which has undergone SMOTE balancing. This training data consists of two key components: `xgb_x_train_smote` for input features and `xgb_y_train_smote` for the target variable. The training process enables the model to learn from the data, capturing intricate patterns and relationships.

Subsequently, the trained XGBoost model is leveraged to generate predictions for two distinct datasets. The first dataset, `xgb_x_test`, is designated as the test data, serving as the basis for assessing the model's performance. Predictions for this dataset are stored in the variable `xgb_prediction`. The second dataset, `xgb_x_train`, represents the training data used as a reference, and predictions for this dataset are retained in the variable `xgb_train_prediction`.

These predictions play a crucial role in evaluating the model's accuracy, overall performance, and its capacity to generalize when presented with new, unseen data. This evaluation process is pivotal in determining the model's effectiveness and suitability for the binary classification task at hand.



```
[54]
# Initialize the model
xgb_classifier = XGBClassifier(max_depth = 8)
```

```
[55]
# Fitting the data on the model
xgb_classifier.fit(xgb_x_train_smote, xgb_y_train_smote)
```

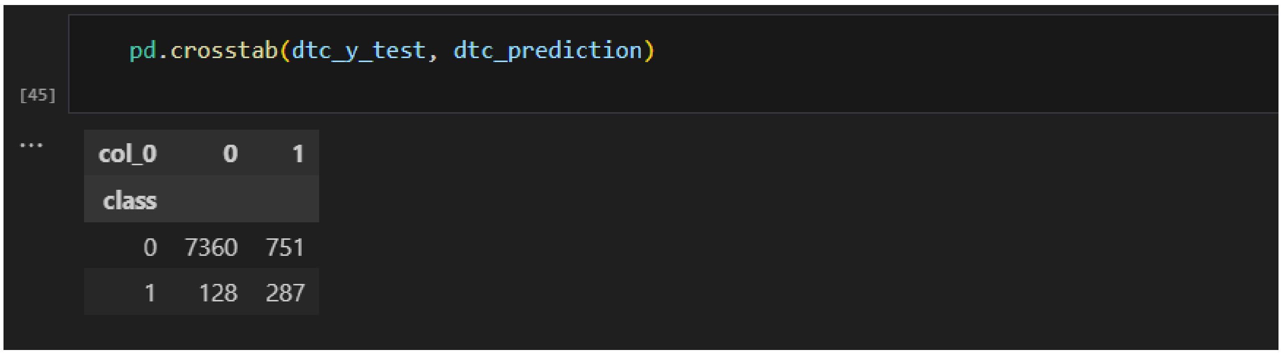
```
XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=8, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
              num_parallel_tree=None, random_state=None, ...)
```

```
# Get the predictions
xgb_prediction = xgb_classifier.predict(xgb_x_test)
xgb_train_prediction = xgb_classifier.predict(xgb_x_train)
```

PERFORMANCE TESTING

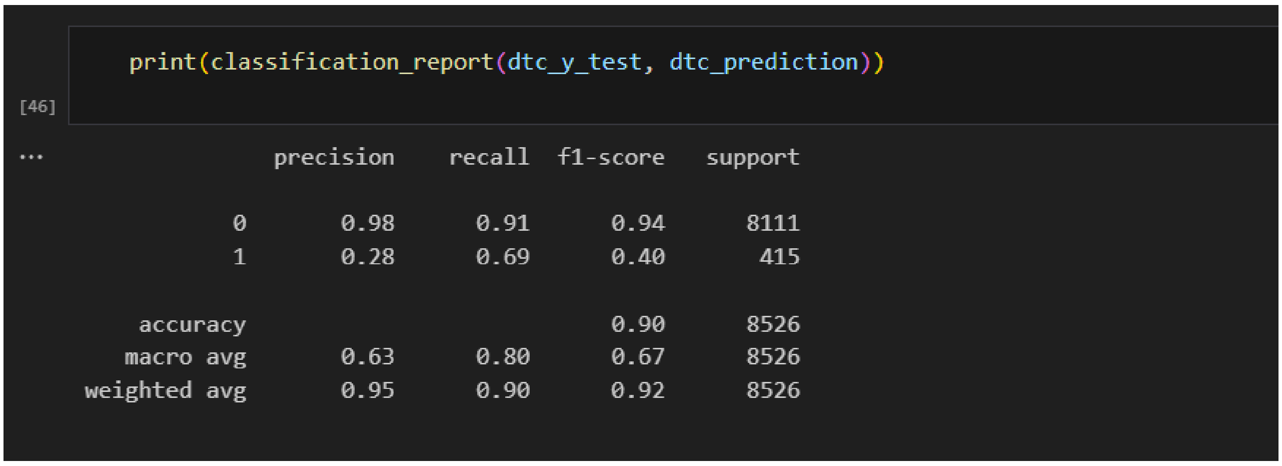
Machine learning performance testing involves evaluating the efficiency and efficacy of ML models and algorithms. It evaluates variables such as training time, prediction speed, and resource utilization to ensure that models match application requirements while optimizing computational resources. For implementing ML systems that fulfil real-world restrictions and produce trustworthy results, performance testing is critical. The accuracy scores of the Decision Tree Classifier, Random Forest Classifier, and Extreme Gradient Boosting Classifier are listed below.

Decision Tree Classifier

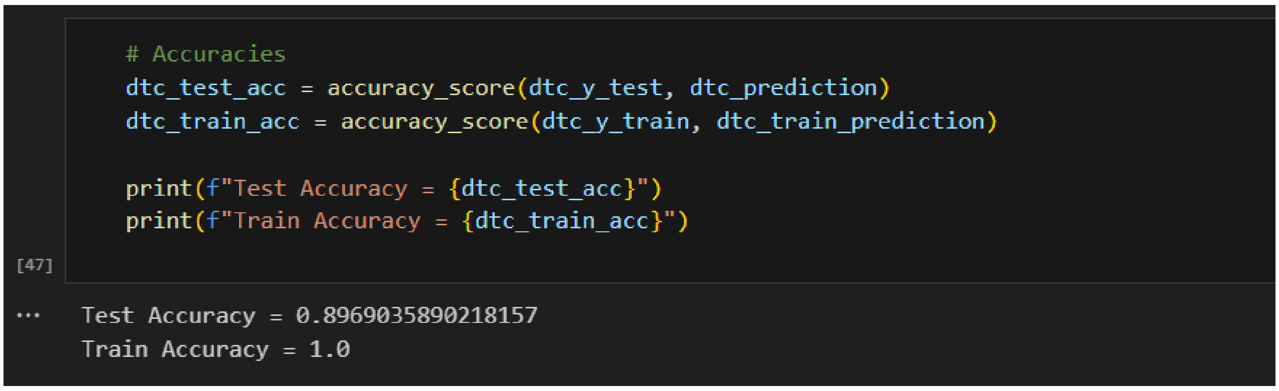


```
[45]  pd.crosstab(dtc_y_test, dtc_prediction)
```

col_0	0	1
class		
0	7360	751
1	128	287

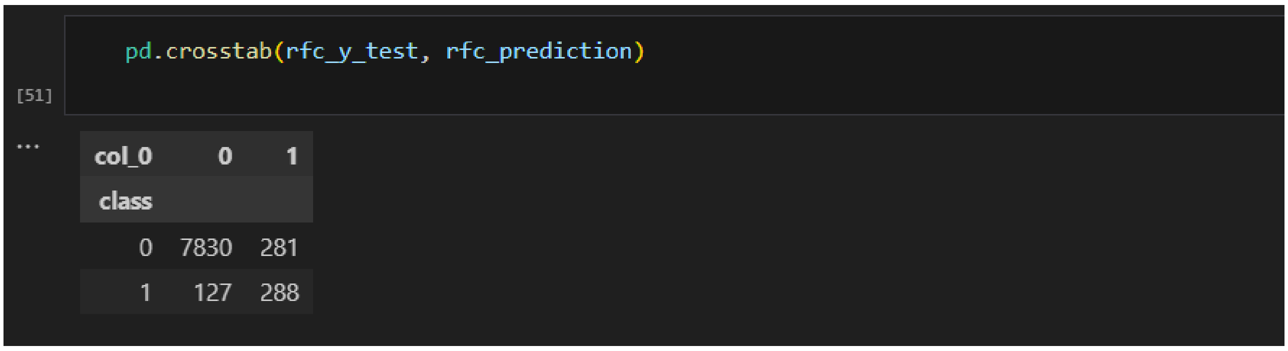


```
[46]  print(classification_report(dtc_y_test, dtc_prediction))

              precision    recall  f1-score   support

           0       0.98      0.91      0.94      8111
           1       0.28      0.69      0.40       415

    accuracy                           0.90      8526
   macro avg       0.63      0.80      0.67      8526
weighted avg       0.95      0.90      0.92      8526
```



```
# Accuracies
dtc_test_acc = accuracy_score(dtc_y_test, dtc_prediction)
dtc_train_acc = accuracy_score(dtc_y_train, dtc_train_prediction)

print(f"Test Accuracy = {dtc_test_acc}")
print(f"Train Accuracy = {dtc_train_acc}")
[47]

Test Accuracy = 0.8969035890218157
Train Accuracy = 1.0
```

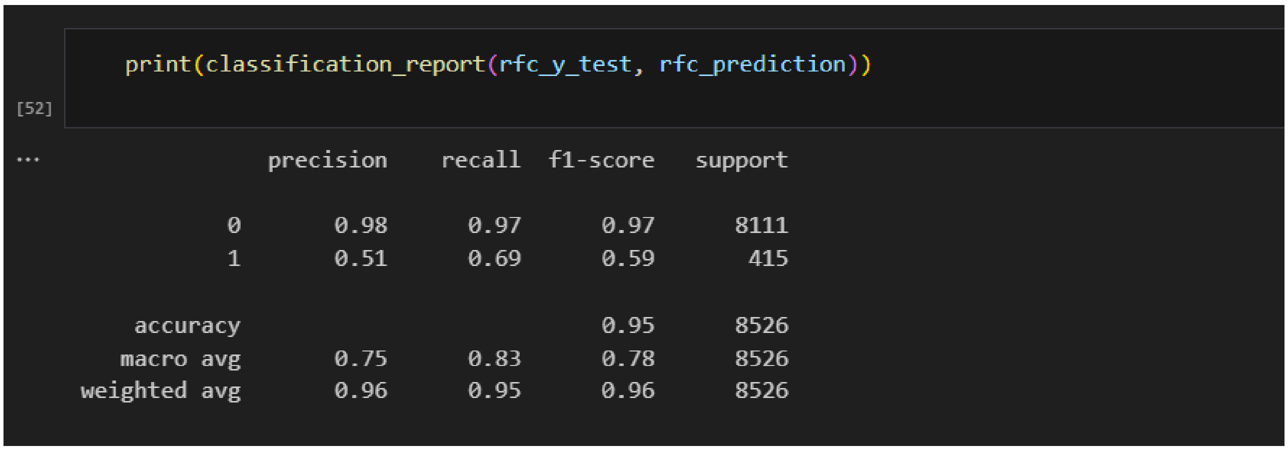
Random Forest Classifier



```
pd.crosstab(rfc_y_test, rfc_prediction)
```

[51]

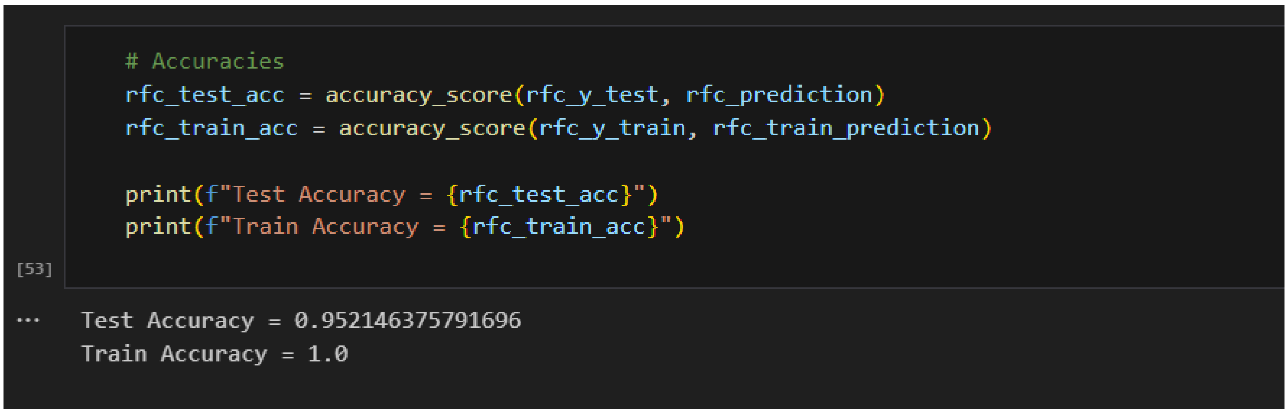
col_0	0	1
class		
0	7830	281
1	127	288



```
print(classification_report(rfc_y_test, rfc_prediction))
```

[52]

```
              precision    recall  f1-score   support

           0       0.98      0.97      0.97      8111
           1       0.51      0.69      0.59       415

    accuracy                           0.95      8526
   macro avg       0.75      0.83      0.78      8526
weighted avg       0.96      0.95      0.96      8526
```



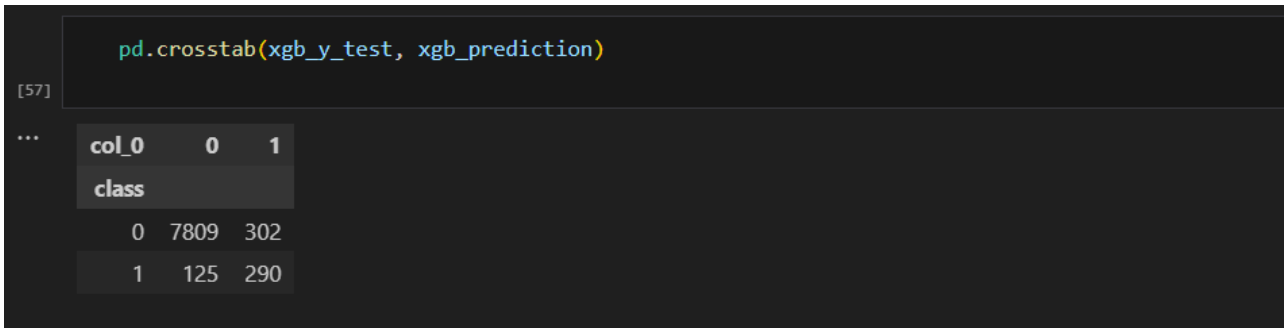
```
# Accuracies
rfc_test_acc = accuracy_score(rfc_y_test, rfc_prediction)
rfc_train_acc = accuracy_score(rfc_y_train, rfc_train_prediction)

print(f"Test Accuracy = {rfc_test_acc}")
print(f"Train Accuracy = {rfc_train_acc}")
```

[53]

```
Test Accuracy = 0.952146375791696
Train Accuracy = 1.0
```

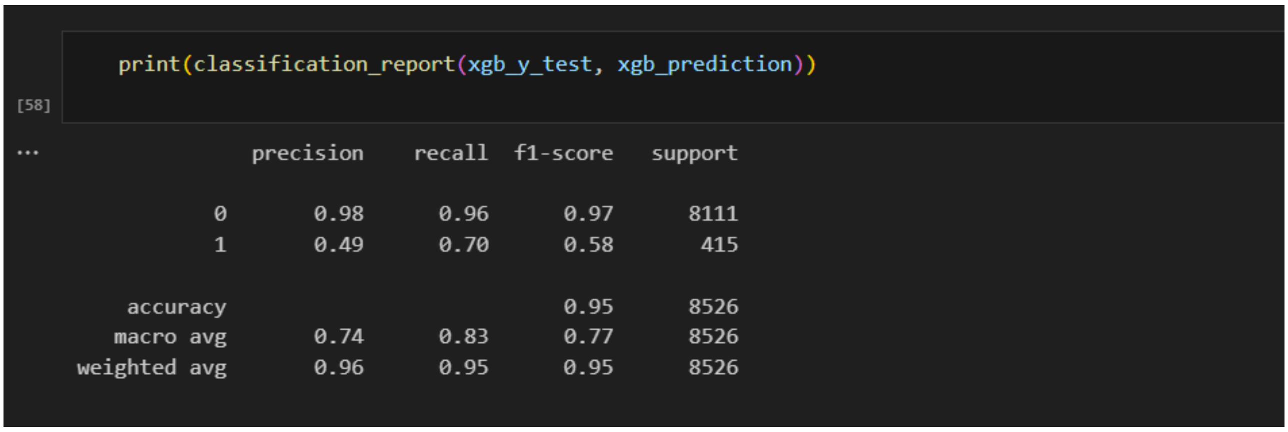
Back to Index

Extreme Gradient Boosting Classifier



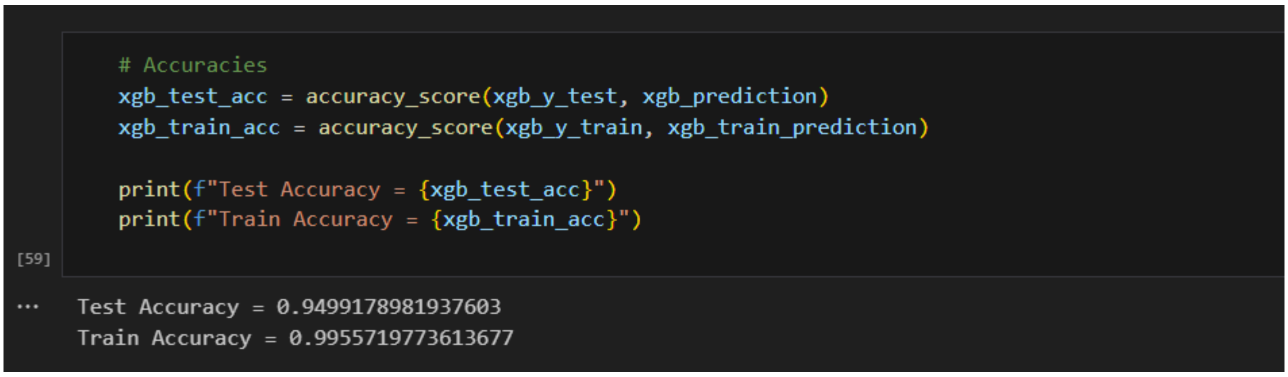
```
[57]
pd.crosstab(xgb_y_test, xgb_prediction)
```

col_0	0	1
class		
0	7809	302
1	125	290



```
[58]
print(classification_report(xgb_y_test, xgb_prediction))
```

```
              precision    recall  f1-score   support

           0       0.98      0.96      0.97      8111
           1       0.49      0.70      0.58       415

    accuracy                           0.95      8526
   macro avg       0.74      0.83      0.77      8526
weighted avg       0.96      0.95      0.95      8526
```

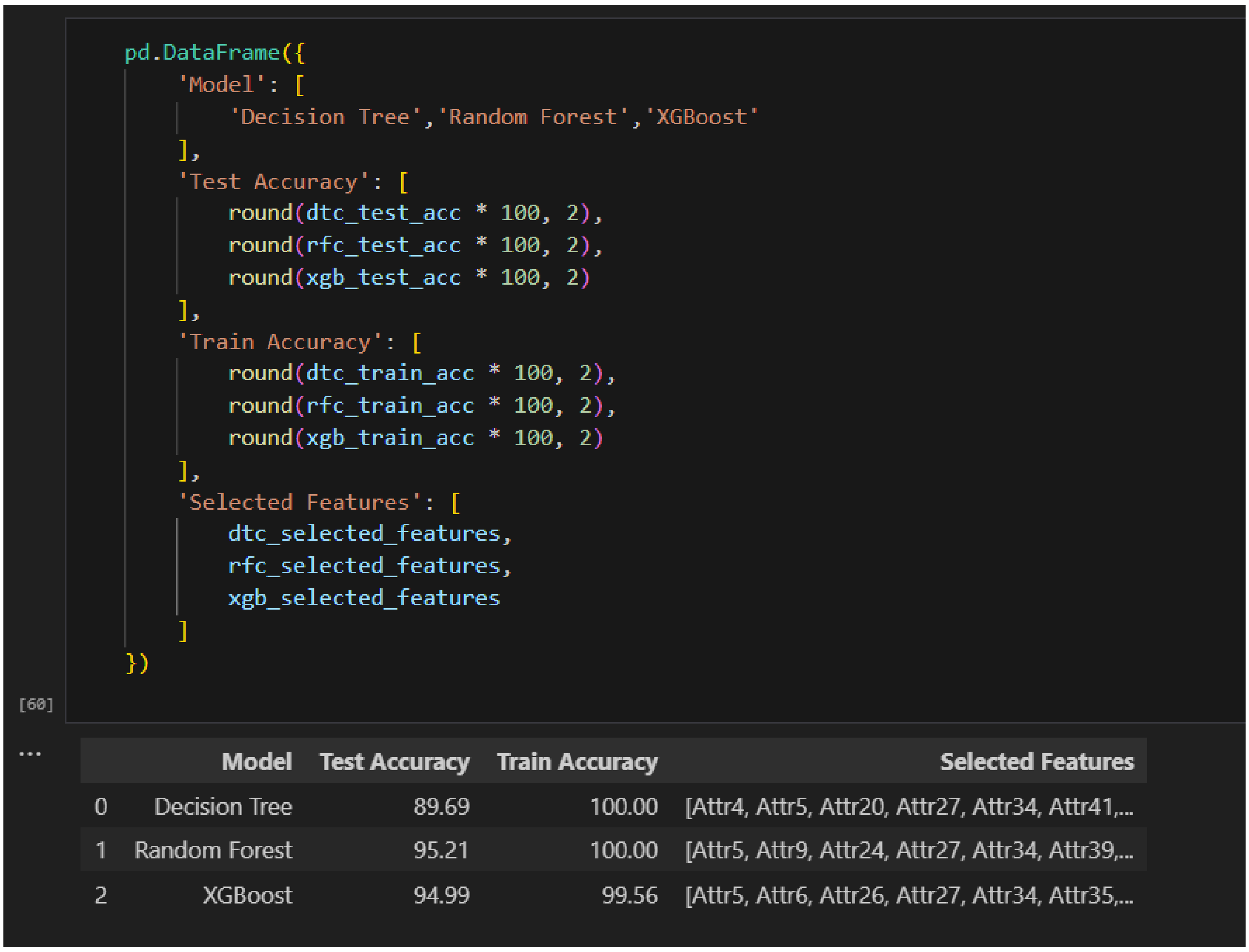


```
[59]
# Accuracies
xgb_test_acc = accuracy_score(xgb_y_test, xgb_prediction)
xgb_train_acc = accuracy_score(xgb_y_train, xgb_train_prediction)

print(f"Test Accuracy = {xgb_test_acc}")
print(f"Train Accuracy = {xgb_train_acc}")
```

```
Test Accuracy = 0.9499178981937603
Train Accuracy = 0.9955719773613677
```

Back to Index

Comparison



```
pd.DataFrame({
    'Model': [
        'Decision Tree','Random Forest','XGBoost'
    ],
    'Test Accuracy': [
        round(dtc_test_acc * 100, 2),
        round(rfc_test_acc * 100, 2),
        round(xgb_test_acc * 100, 2)
    ],
    'Train Accuracy': [
        round(dtc_train_acc * 100, 2),
        round(rfc_train_acc * 100, 2),
        round(xgb_train_acc * 100, 2)
    ],
    'Selected Features': [
        dtc_selected_features,
        rfc_selected_features,
        xgb_selected_features
    ]
})
```

[60]

	Model	Test Accuracy	Train Accuracy	Selected Features
0	Decision Tree	89.69	100.00	[Attr4, Attr5, Attr20, Attr27, Attr34, Attr41,...
1	Random Forest	95.21	100.00	[Attr5, Attr9, Attr24, Attr27, Attr34, Attr39,...
2	XGBoost	94.99	99.56	[Attr5, Attr6, Attr26, Attr27, Attr34, Attr35,...

Based on the evaluation of the three classifiers, Decision Tree Classifier (DTC), Random Forest Classifier (RFC), and Extreme Gradient Boosting Classifier (XGBoost), it is evident that each classifier exhibits varying levels of performance on both the test and training data. The Decision Tree Classifier achieves an impressive accuracy of 90% on the test data and 100% on the training data, indicating robust performance. The Random Forest Classifier excels further with a test accuracy of 95% and full training data accuracy. Notably, the Extreme Gradient Boosting Classifier showcases remarkable accuracy, with a near-ideal 94.99% on the test data and a high 99.56% on the training data, demonstrating strong generalization capabilities.

Considering these results, it becomes clear that the *Extreme Gradient Boosting Classifier* stands out as the most promising choice. Its ability to maintain a high test accuracy while avoiding overfitting, as seen in the training data, signifies its robustness and suitability for the binary classification task at hand. Therefore, we conclude that the Extreme Gradient Boosting Classifier is the most effective and reliable classifier for this specific problem, providing the best results among the evaluated models.

Back to Index

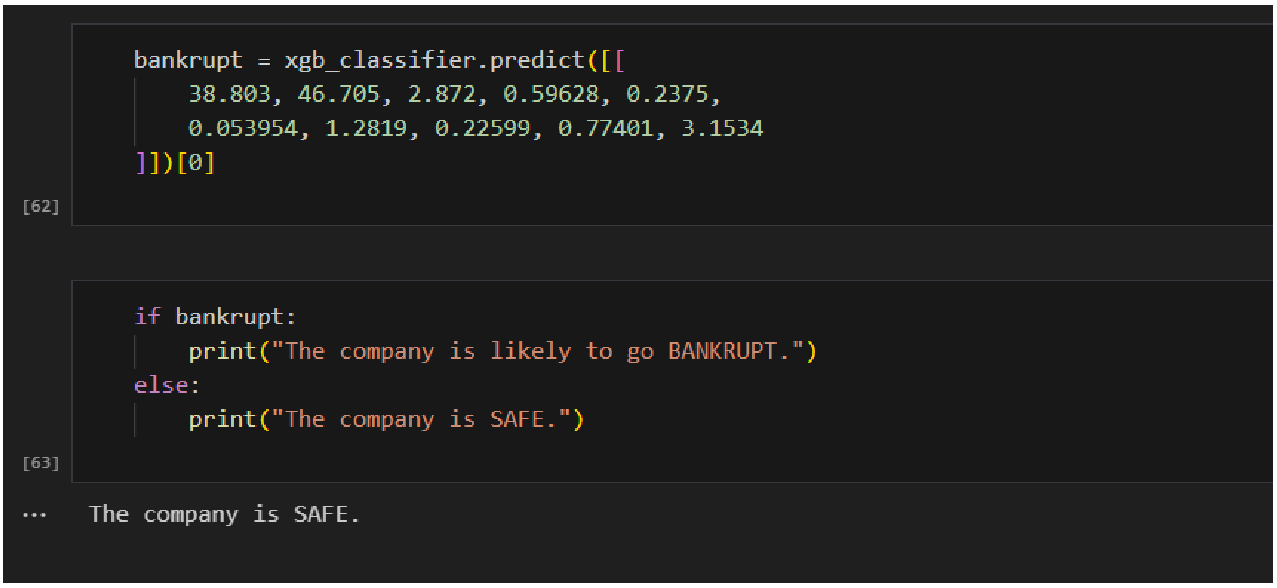
MODEL DEPLOYMENT

Model Testing

The trained XGBoost Classifier model, denoted as `xgb_classifier`, is utilized to make a prediction for a set of input features. These input features, represented as numerical values, are fed into the model for analysis. The model generates a prediction that determines the likelihood of a company going bankrupt or remaining financially stable.

After the prediction is made, the code evaluates the result. If the prediction suggests that the company is likely to go bankrupt (i.e., `bankrupt` evaluates to `True`), it prints a message stating "The company is likely to go BANKRUPT." Conversely, if the prediction indicates that the company is safe (i.e., `bankrupt` evaluates to `False`), it prints "The company is SAFE."

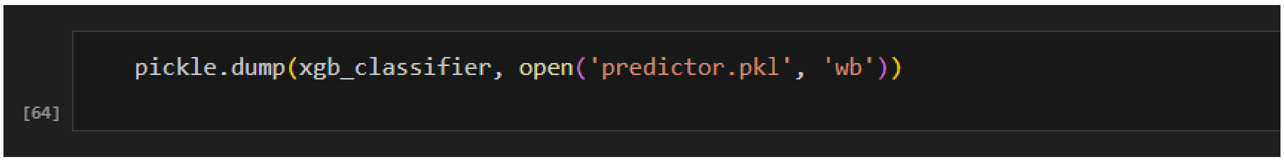
This demonstrates how the trained model can be applied to real-world data to provide insights or predictions, making it a valuable tool for assessing the financial health of companies.



```
bankrupt = xgb_classifier.predict([[
    38.803, 46.705, 2.872, 0.59628, 0.2375,
    0.053954, 1.2819, 0.22599, 0.77401, 3.1534
]])[0]
[62]

if bankrupt:
    print("The company is likely to go BANKRUPT.")
else:
    print("The company is SAFE.")
[63]
...  The company is SAFE.
```

Exporting the Model

This code snippet carries out the process of serializing and saving an XGBoost Classifier model into a binary file named 'predictor.pkl.' The primary purpose of this action is to preserve the trained model for future use. The `pickle.dump` function is employed to perform this serialization, and it specifically targets the `xgb_classifier` model, which contains all the learned patterns and information from the training phase. The 'wb' mode, used with the `open` function, ensures that the file is opened in binary write mode, allowing the binary data representation of the model to be written into the 'predictor.pkl' file. This saved model can be reloaded later to make predictions or conduct further analysis without the necessity of retraining the model from scratch.

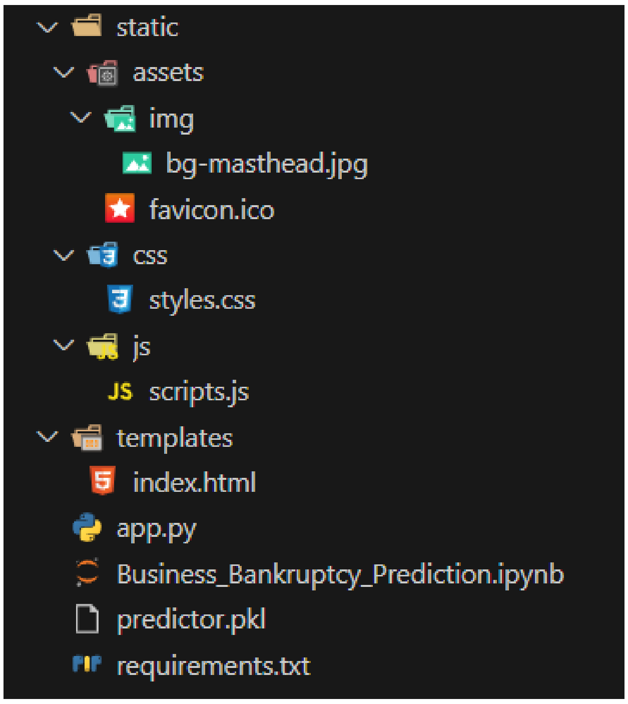


```
pickle.dump(xgb_classifier, open('predictor.pkl', 'wb'))
[64]
```

Back to Index

Web Application

The project structure is as follows:



```
static
  assets
    img
      bg-masthead.jpg
    favicon.ico
  css
    styles.css
  js
    scripts.js
templates
  index.html
app.py
Business_Bankruptcy_Prediction.ipynb
predictor.pkl
requirements.txt
```

At its core, we can find the "static" directory. Nested within this directory is the "assets" subfolder, housing a valuable collection of assets that contribute to the application's visual appeal and functionality. Notably, the "img" subdirectory within "assets" contains "bg-masthead.jpg," a key image resource serving as the background for the web interface. Furthermore, the "favicon.ico" file, located directly within "assets," functions as the browser tab's iconic representation. The "css" directory stores "styles.css," a CSS file crucial for defining the visual aesthetics and styling of the web application. In parallel, the "js" directory houses "scripts.js," a JavaScript file responsible for incorporating client-side scripting features into the web application.

The "templates" directory, as the name suggests, serves as the repository for HTML templates utilized by the Flask web application. Notably, "index.html" is the primary and only HTML template, defining the structure and content of the web application's main page.

At the heart of the project is "app.py," a Python script that leverages the Flask web framework to create, run, and manage the web application. This script connects all the elements, ensuring the smooth execution of the application and serving the model's predictions to users. The "predictor.pkl" file is central to the predictive capabilities of the application, containing a serialized machine learning model ready for use in making predictions related to business bankruptcy.

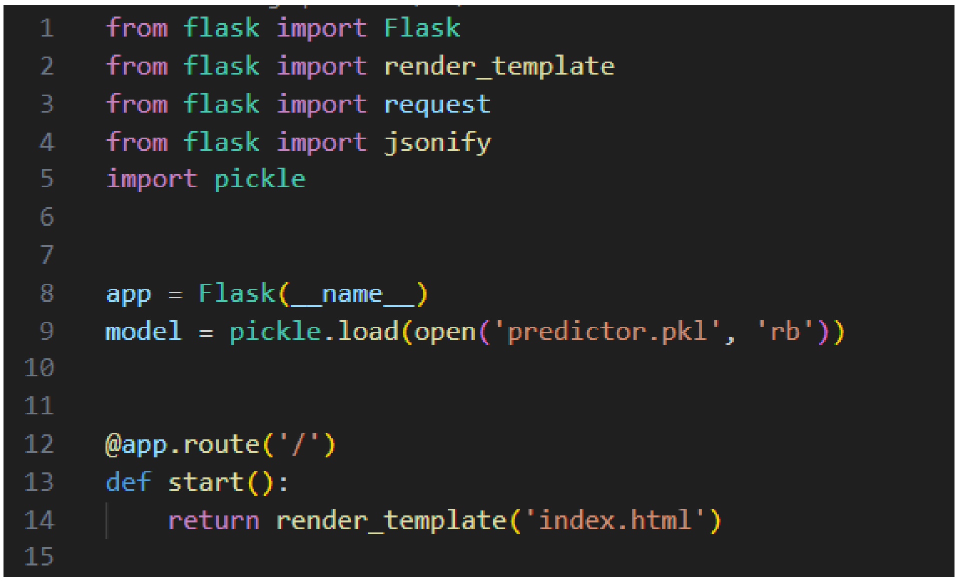
"Business_Bankruptcy_Prediction.ipynb" is the Jupyter Notebook, that includes code, data analysis, and project documentation.

Finally, "requirements.txt" plays a critical role in specifying the project's dependencies and their version compatibility, guaranteeing that the necessary libraries and packages are installed for seamless application functionality.

Back to Index

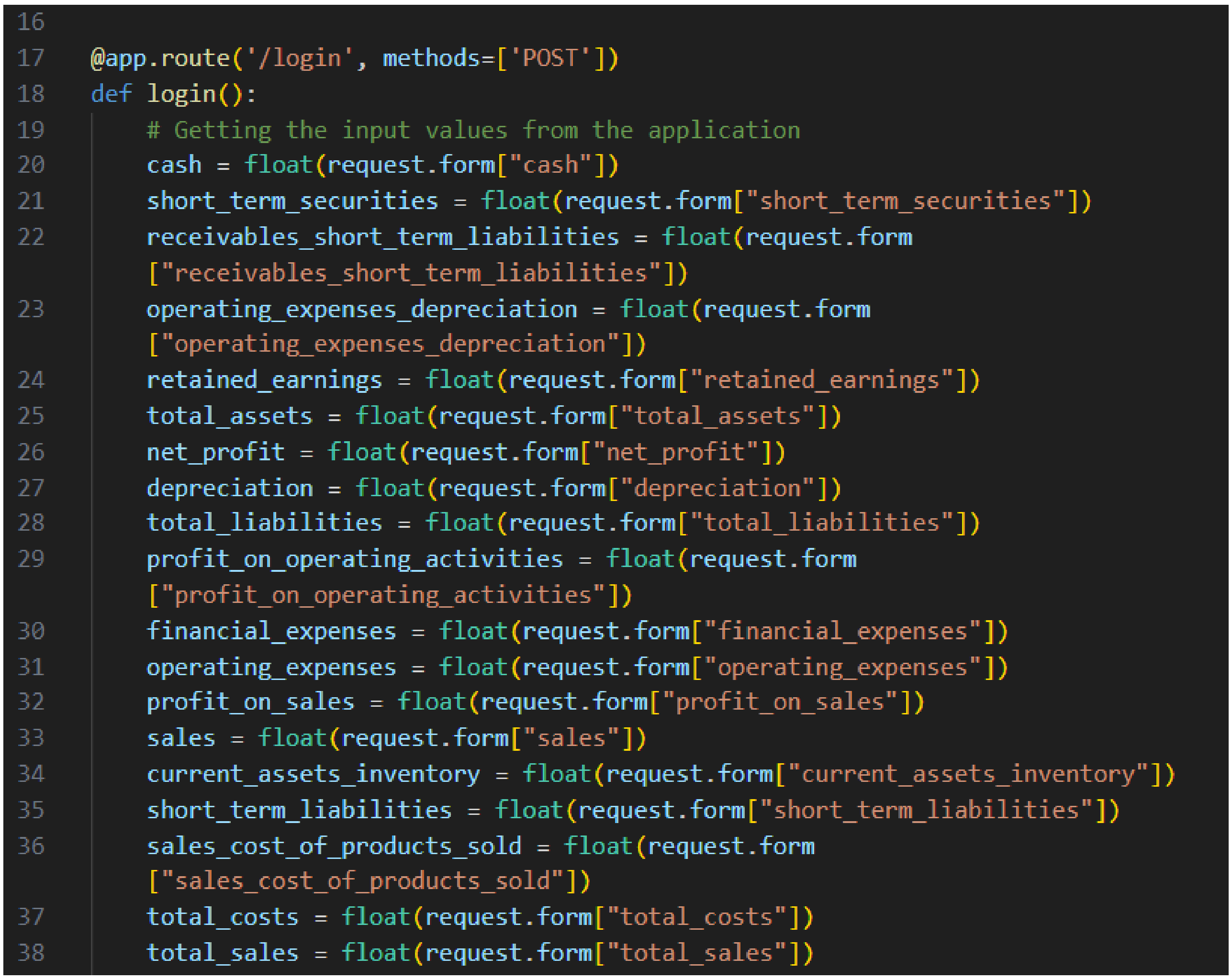
Flask App

In this Flask web application, a server is established using Flask, and a pre-trained machine learning model is loaded from 'predictor.pkl' using `pickle.load()`. The application has a single route, '@app.route('/')', serving as the web app's main page, where users access the 'index.html' template for interaction. This Flask application forms the core infrastructure for deploying a web interface to leverage the loaded model for making predictions.

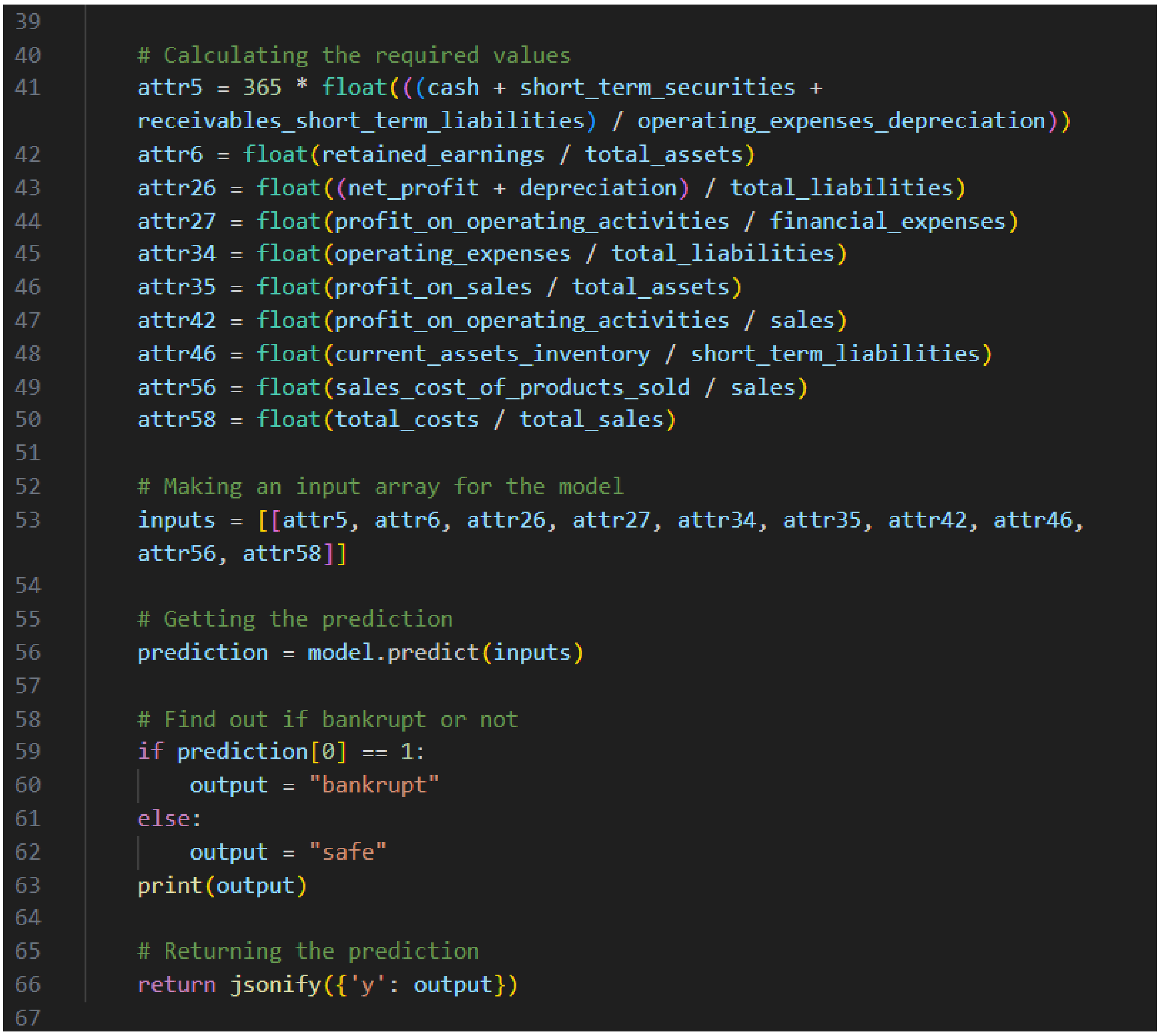


```python
from flask import Flask
from flask import render_template
from flask import request
from flask import jsonify
import pickle


app = Flask(__name__)
model = pickle.load(open('predictor.pkl', 'rb'))


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

```

The Flask route facilitates the interaction between the user interface and the machine learning model, enabling real-time bankruptcy predictions for companies based on the provided financial data.



```python

@app.route('/login', methods=['POST'])
def login():
    # Getting the input values from the application
    cash = float(request.form["cash"])
    short_term_securities = float(request.form["short_term_securities"])
    receivables_short_term_liabilities = float(request.form
    ["receivables_short_term_liabilities"])
    operating_expenses_depreciation = float(request.form
    ["operating_expenses_depreciation"])
    retained_earnings = float(request.form["retained_earnings"])
    total_assets = float(request.form["total_assets"])
    net_profit = float(request.form["net_profit"])
    depreciation = float(request.form["depreciation"])
    total_liabilities = float(request.form["total_liabilities"])
    profit_on_operating_activities = float(request.form
    ["profit_on_operating_activities"])
    financial_expenses = float(request.form["financial_expenses"])
    operating_expenses = float(request.form["operating_expenses"])
    profit_on_sales = float(request.form["profit_on_sales"])
    sales = float(request.form["sales"])
    current_assets_inventory = float(request.form["current_assets_inventory"])
    short_term_liabilities = float(request.form["short_term_liabilities"])
    sales_cost_of_products_sold = float(request.form
    ["sales_cost_of_products_sold"])
    total_costs = float(request.form["total_costs"])
    total_sales = float(request.form["total_sales"])
```



```python

    # Calculating the required values
    attr5 = 365 * float(((cash + short_term_securities + receivables_short_term_liabilities) / operating_expenses_depreciation))
    attr6 = float(retained_earnings / total_assets)
    attr26 = float((net_profit + depreciation) / total_liabilities)
    attr27 = float(profit_on_operating_activities / financial_expenses)
    attr34 = float(operating_expenses / total_liabilities)
    attr35 = float(profit_on_sales / total_assets)
    attr42 = float(profit_on_operating_activities / sales)
    attr46 = float(current_assets_inventory / short_term_liabilities)
    attr56 = float(sales_cost_of_products_sold / sales)
    attr58 = float(total_costs / total_sales)

    # Making an input array for the model
    inputs = [[attr5, attr6, attr26, attr27, attr34, attr35, attr42, attr46, attr56, attr58]]

    # Getting the prediction
    prediction = model.predict(inputs)

    # Find out if bankrupt or not
    if prediction[0] == 1:
        output = "bankrupt"
    else:
        output = "safe"
    print(output)

    # Returning the prediction
    return jsonify({'y': output})

```

This Flask route, defined as `/login` and configured to accept POST requests, serves as the endpoint for a predictive application. Upon receiving data from the application's user interface, it processes the input values and employs the trained machine learning model for bankruptcy prediction.

Data Retrieval: The incoming data is obtained from the user interface, which includes various financial attributes such as cash, short-term securities, receivables, and more. These values are accessed using the `request.form` object and are converted to floating-point numbers for numerical analysis.

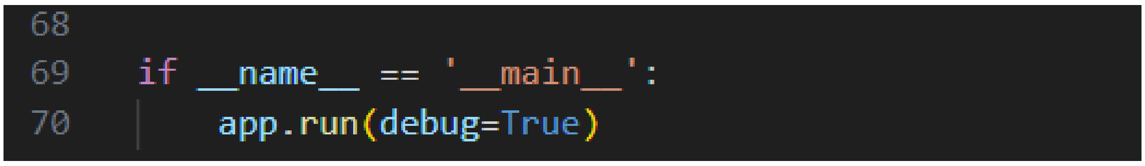
Attribute Calculation: Several derived attributes are calculated based on the input values. These attributes, such as `attr5`, `attr6`, and others, play a crucial role in assessing the financial health of a company. They are computed using mathematical formulas that consider the financial data provided.

Data Preparation: The calculated attributes are organized into an input array, `inputs`, which is structured in a manner suitable for the machine learning model's input requirements.

Prediction: The machine learning model, loaded earlier, is employed to predict the likelihood of the company going bankrupt. The input array `inputs` is fed into the model, and the prediction is obtained. If the prediction is equal to 1, it signifies a prediction of bankruptcy; otherwise, it indicates a prediction of financial stability.

Result Handling: The prediction outcome is stored as either "bankrupt" or "safe," based on the model's assessment. This result is then returned in a JSON format using the `jsonify` function. Additionally, it is printed to the server's console for reference.

The last section checks whether the current Python script is being run as the main program or if it's being imported as a module into another script. If it is the main program, the script proceeds to execute the Flask application with debugging mode enabled. Debugging mode provides detailed error information and automatically refreshes the application when code changes occur. This part of the code ensures that the Flask application runs when the script is executed directly, making it ready for local testing and development.



```python

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

Back to Index

ADVANTAGES & DISADVANTAGES

To understand the project better, we need to look at both its strong points and its potential drawbacks. Let us now explore what makes the project great and what challenges it might face. The project has several advantages, and a few of them are as follows.

1. Risk Mitigation:
    - This project serves as a powerful risk mitigation tool for businesses.
    - By accurately predicting financial vulnerabilities and distress, companies can take proactive measures to address potential issues.
    - This includes adjusting their financial strategies, securing additional funding, or making operational changes to avert bankruptcy.
2. Informed Decision-Making:
    - In today's complex and volatile business landscape, informed decision-making is critical.
    - The project's predictive model equips businesses with the insights they need to make data-driven financial decisions.
    - It helps them allocate resources more efficiently and seize opportunities while mitigating risks.
3. Economic Stability:
    - Beyond individual businesses, the project indirectly contributes to economic stability.
    - Preventing business bankruptcies safeguards jobs, investments, and overall economic prosperity.
    - The ripple effect of preserving employment and economic activities can be significant, promoting a stable and prosperous society.
4. Industry Benchmarking:
    - Industry benchmarking tools offered by the project provide businesses with a competitive edge.
    - By comparing their financial performance against industry peers, they gain a comprehensive understanding of their market positioning.
    - This competitive intelligence empowers businesses to make strategic decisions and stay ahead in their respective sectors.
5. Educational Resource:
    - The inclusion of educational resources within the platform is an advantage for users.
    - Businesses and professionals can access articles, webinars, and resources related to financial management and bankruptcy prevention.
    - This educational component not only enhances the project's value but also fosters financial literacy and empowerment among its user base.
6. Global Relevance:
    - As the project expands to serve global companies, it gains enhanced relevance and utility.
    - The global economy is interconnected, and the ability to assess financial health and bankruptcy risks on a global scale is invaluable.
    - By offering its services internationally, the project positions itself as a versatile and globally

relevant tool.

7. Real-Time Monitoring:
   - The inclusion of real-time monitoring through interactive dashboards represents a proactive approach to financial health management.
   - Businesses can respond swiftly to financial warning signs, preventing prolonged financial distress.
   - This real-time feature is invaluable in ensuring a company's long-term financial health.
8. Compliance Solutions:
   - Regulatory compliance is a vital aspect of financial management.
   - The project's assistance in regulatory compliance helps businesses reduce the risk of regulatory issues, fines, or penalties.
   - It streamlines the often complex process of adhering to financial regulations, further enhancing a company's legal and financial standing.
9. Data Diversity:
   - The project benefits from data diversity by integrating information from Emerging Markets Information Service (EMIS), which covers emerging markets globally.
   - This diversity enhances the robustness of the dataset, offering a broader perspective on financial dynamics and bankruptcy prediction.
10. Customized Models:
    - The ability to develop customized models for specific industries is an advantage for the project.
    - Different sectors have unique financial dynamics and risk factors.
    - Customization enhances the accuracy of predictions, allowing businesses to receive tailored insights that align closely with their industry-specific challenges and opportunities.

However, it also has some disadvantages, which can be understood from the points given below.

1. Data Limitations:
   - A potential limitation of the project is its dataset's focus on Polish companies.
   - While it provides valuable insights into the financial health of these businesses, it may have limited immediate applicability to companies in other geographical regions, potentially restricting its global reach.
2. Data Privacy Concerns:
   - The handling of sensitive financial data introduces legitimate concerns related to data privacy and security.
   - To address these concerns, the project must implement robust data protection measures, which can be resource-intensive and challenging.
3. Market Dynamics:
   - Predicting bankruptcy is inherently challenging due to the rapidly changing nature of market conditions and unforeseen external factors that may impact a company's financial health.
   - These external variables, such as economic downturns, shifts in consumer behaviour, or changes in industry regulations, can introduce uncertainty into predictions.

Back to Index

4. Cost of Development:
    - Expanding and enhancing the platform, introducing new features, and maintaining data privacy and security may require significant financial investments.
    - Developing and maintaining a comprehensive financial platform is resource-intensive and must be managed efficiently.
5. User Adoption:
    - Convincing businesses and financial professionals to adopt the platform may be a challenge.
    - Businesses are often cautious about adopting new financial tools and may require effective marketing and outreach efforts to demonstrate the platform's value and efficacy.

Back to Index

CONCLUSION

In summary, our project has emerged as a crucial and ever-relevant endeavour in the field of economics and financial analysis. As bankruptcy prediction remains a perennial subject of interest, this project was conceived with the aim of developing a robust predictive model that integrates diverse econometric measures to foresee the financial condition of firms. This undertaking serves the overarching purpose of assessing the financial health of companies and providing invaluable insights into their prospects in the context of long-term operations within the market. By focusing on a dataset specific to Polish companies, collected from the Emerging Markets Information Service (EMIS), the project offers a comprehensive analysis of bankruptcy prediction over the years 2000-2012. This dataset, combined with the meticulous categorization based on forecasting periods, provides a solid foundation for financial analysis. The inclusion of financial rates from the first year of the forecasting period, along with corresponding class labels indicating bankruptcy status, enhances the project's predictive accuracy and utility.

This project not only underscores the importance of financial health in the corporate world but also represents a significant milestone in the ongoing quest for effective bankruptcy prediction. With its rich dataset and evolving predictive models, this project is poised to be a valuable resource for businesses, investors, regulatory authorities, and financial professionals. It is a testament to the enduring relevance of economic analysis and serves as a stepping stone for future advancements in the field of financial prediction. As we move forward, the project is primed for expansion and enhancement, offering a promising future in supporting businesses in making informed decisions, safeguarding their financial well-being, and contributing to a broader economic stability.

Back to Index

FUTURE SCOPE

Looking ahead, our project holds tremendous promise for growth and development. Our ambition goes far beyond simply predicting financial distress; we are transforming into a comprehensive platform that addresses a multitude of needs, and the possibilities are boundless.

In the future, a primary focus will be on enhancing our predictive models. This will involve ongoing refinement through advanced machine learning and artificial intelligence. The objective is to make our predictions even more accurate, thus ensuring our users receive reliable insights into financial distress. We are exploring the integration of real-time financial data, which can substantially elevate the precision of our forecasts. Furthermore, we are considering crafting tailored models for specific industries to provide even sharper and more industry-specific predictions.

Another exciting development on the horizon is the incorporation of AI-powered chatbots into our platform. These chatbots will act as friendly guides, ready to interpret bankruptcy predictions, provide detailed explanations, and offer practical advice. This transformation will turn our platform into a trusted financial advisor, bridging the gap between data and actionable insights, making it more user-friendly and accessible.

Expanding our data sources is also a significant part of our future. We intend to widen our dataset to encompass global companies, extending beyond our current focus on Polish businesses. This expansion will enrich our analysis by ensuring data accuracy and diversity. Collaborating with multiple data providers will be instrumental in maintaining the quality and reliability of our predictions while opening the door to new insights.

Our future vision extends beyond prediction to encompass comprehensive risk management. We aim to provide businesses with a suite of tools to identify potential financial vulnerabilities and offer guidance on managing risks effectively. The introduction of interactive dashboards will empower users to monitor their company's financial health in real-time, facilitating timely decisions and actions to mitigate potential issues proactively.

In addition, we are actively exploring the simplification of regulatory compliance for businesses. Complying with financial regulations can be intricate and challenging, and we aspire to offer solutions that streamline this process. These solutions will not only reduce the risk of regulatory issues but also contribute to the seamless operation of businesses in a complex regulatory landscape.

Furthermore, we are planning to introduce industry benchmarking tools in the future. This will enable companies to evaluate their financial performance in comparison to industry peers, providing valuable insights for businesses striving to maintain a competitive edge. This feature will contribute to a comprehensive understanding of a company's financial health within the broader industry context.

Knowledge-sharing is an integral part of our strategy for the future. We are gearing up to create an educational hub within our platform, offering a treasure trove of articles, webinars, and resources on financial management and bankruptcy prevention. Our goal is to evolve into a valuable learning resource, going beyond being a mere prediction tool. This initiative will empower businesses with essential knowledge and equip them with the tools and strategies needed to make informed financial decisions.

We also have global aspirations, aiming to reach international markets by translating our platform into multiple languages. This will make our platform accessible to businesses worldwide, extending our global reach and impact. The ability to interact with our platform in their native language will empower businesses in various regions to leverage our insights effectively.

A user-friendly mobile application is in the pipeline for the future. This application will offer on-the-

Back to Index

go access to financial insights and bankruptcy predictions, enhancing user convenience. It will also provide push notifications for critical financial alerts, ensuring users stay informed, even when they are not actively using the platform.

User feedback remains a cornerstone of our future efforts, and we actively seek it. Collaboration with experts in the field, analysts, and regulatory bodies is another avenue for improvement. This ensures that our predictions stay aligned with industry best practices and remain at the forefront of financial analysis.

Premium services are also on our radar for the future. These services will present advanced features and personalized financial guidance, catering to a wide array of business needs. By offering these premium services, we aim to provide tailored solutions that meet the unique requirements of businesses, regardless of their size or industry.

In summary, our vision for the future of "Anticipating Business Bankruptcy" transcends mere prediction. It is about empowering businesses globally, offering the insights and tools they need to succeed in an increasingly complex financial landscape. We are dedicated to adapting to the evolving financial landscape and providing services that align with the dynamic needs of our user community. Our journey is about fortifying the financial well-being of businesses and facilitating smart, data-driven decision-making. The future is indeed promising, and we are enthusiastic about being part of it as we evolve and expand our services to better serve our diverse user base.

Back to Index

APPENDIX

All the essential project files, including those used to develop the web application, machine learning model, and Flask app, are conveniently located in our GitHub repository. You can easily access these resources within the 'project' folder, ensuring a comprehensive view of our project's components and enabling collaboration and exploration.

- Navigate to the main repository from here: Link
- Navigate to the project folder from here: Link
- Navigate to the final submission folder from here: Link

Back to Index