# Detecting Collusion in Public Procurement: A Comparative Study of Machine Learning Models

Ruchika Barot

A Thesis

in

The Department

of

Supply Chain & Business Technology Management

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Supply Chain Management at

Concordia University

Montréal, Quebec, Canada

November 2023

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared,

By: Ruchika Barot

Entitled: Detecting Collusion in Public Procurement: A Comparative Study of Machine Learning Models

and submitted in partial fulfillment of the requirements for the degree of

**Master of Supply Chain Management**

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final Examining Committee:

| | |
|---|---|
| Dr. Danielle Morin | Chair |
| Dr. Danielle Morin | Examiner |
| Dr. Mohsen Farhadloo | Examiner |
| Dr. Chaher Alzaman | Supervisor |

Approved by: Dr. Satyaveer S. Chauhan, Graduate Program Director

Date: December 20th, 2023

Dean: Dr. Anne-Marie Croteau

## ABSTRACT

Detecting Collusion in Public Procurement: A Comparative Study of Machine Learning Models

Ruchika Barot

Detecting collusion in public procurement is critical to ensure fair and transparent practices in government acquisitions. Bid collusion in auctions poses a major challenge in public procurement by causing unfair price hikes through unlawful cooperation among competing firms, consistently affecting the overall supply chain. This study uses machine learning methods to investigate collusion in public procurement processes. It delves deeply into exploring multiple machine learning models such as random forests, extra tree classifiers, support vector classifiers, Neural Networks, Gradient Boosting, and various combinations of models for collusion detection. First, the models were trained using available data, followed by the inclusion of screening variables derived from bid information as additional features. The additional features were fed to the models, which went through fine-tuning of parameters. Additionally, comparative analyses were carried out to evaluate the merits and drawbacks of each model. Metrics including Accuracy, balanced accuracy, precision, recall, F1-score, and ROC-AUC score were evaluated, providing a comprehensive evaluation framework. Various settings were used to compare which set of inputs gives the highest accuracy in collusion detection. The ROC-AUC analysis brought forward crucial insights, particularly regarding models' abilities to minimize false positives while maximizing true positives. Models like Random Forest and Gradient Boosting demonstrated superior performance, showcasing lower false positive rates—a crucial aspect when identifying collusion in public procurement. Additionally, the study underscores the significance of feature engineering in collusion detection. Specifically, attributes like screens - CV, SPD, DIFFP, RD, SKEW, KURTO, and KS significantly aid algorithms in processing data effectively to identify collusion patterns. The outcomes of this study carry significant implications for both the specific domain under investigation and the broader field of collusion detection. Ultimately, this research provides a valuable guide for policymakers, procurement officers, and data scientists, offering valuable insights into the effective machine learning techniques tailored for detecting collusion in public procurement.



**Keywords:** Collusion detection, public procurement, Machine learning models, Feature engineering

## ACKNOWLEDGEMENT

I'm incredibly thankful to Dr. Chaher Alzaman, my thesis supervisor, for his invaluable guidance and unwavering support throughout this journey. Having such a dedicated and responsive mentor has been a true privilege. His expertise was pivotal in the development of this work. This journey, both academically and personally transformative, owes its success to the support and encouragement of various individuals. I deeply thank my family for their unwavering patience, understanding, and constant inspiration throughout this thesis. Their unwavering belief in my abilities has always been my driving force. Additionally, I'm grateful to my friends for their valuable insights and contributions to this academic endeavor.

# TABLE OF CONTENTS

## List of figures

## List of Tables

## Acronyms

| | |
|---|---|
| **ANN** | Artificial neural network |
| **AUC** | Area under the curve |
| **CRISP-DM** | Cross Industry Standard Process for Data Mining |
| **KNN** | K-nearest neighbours |
| **LR** | logistic regression |
| **MLP** | Multi-Layer Perceptron |
| **NN** | Neural Network |
| **PCA** | Principal Component Analysis |
| **RBF** | Radial basis function |
| **ReLU** | Rectified linear unit |
| **ROC** | Receiver operating characteristic |
| **SGD** | Stochastic Gradient Descent |
| **SOM** | Self-Organizing map |
| **SVM** | Support Vector Machine |
| **SVC** | Support Vector Classifier |
| **KNN** | K-nearest neighbours |
| **GB** | Gradient Boosting |
| **RF** | Random Forest |

# Chapter 1 INTRODUCTION

Supply chain management encompasses a network of interconnected stages and entities involved in fulfilling consumer demands. As shown in Figure 1.1, this network expands from suppliers and manufacturers to distributors, retailers, and eventually the consumers themselves. It entails coordinating, managing, and refining operations across these elements to streamline efficiency and meet consumer needs. At every phase of the supply chain, there's a consistent acquisition of a variety of supplies and services that extends from the inception to the completion of the supply chain. Procurement refers to the process of acquiring goods or services from external sources, encompassing sourcing, negotiation through auction, purchasing, and managing supplier relationships. Procurement plays a crucial role in supply chain management by ensuring timely access to quality resources, optimizing costs, and mitigating risks associated with suppliers (Monczka, 2009). Within the domain of public procurement, the government stands as the consumer in figure 1.1, ultimately serving the citizens of the nation. Public procurement stands as an integral part of the supply chain, characterized by its systematic approach to government acquisitions, intending to utilize public funds efficiently and transparently (Rodríguez et al., 2022). Public procurement emerges as a robust mechanism to accomplish economic, environmental, technological, and social goals. The process of public procurement is often intricate and resource-intensive, demanding substantial investments. Public Procurement is susceptible to corruption due to the high value of funds involved, which frequently constitutes a significant share, typically falling within the range of 10% to 20% of GDP. Government spending, as per the Organization for Economic Cooperation and Development (OECD) 2017, constitutes a noteworthy 12% of global GDP (Bosio et al., 2022).


**Figure 1.1:** Supply chain management network

Public procurement carries a high risk of potential illegality throughout the entire process of acquiring goods and services. A significant challenge in public procurement process is Bid rigging, also known as collusion in auctions which involves illegal agreements between rival companies to boost their profits. Collusion, or bid rigging, is malpractice where competing firms form unlawful agreements to inflate prices, limiting fair competition and distorting the procurement process (Rustiarini et al., 2019). These agreements often lead to coordinated (non-competitive) price hikes, and pose a persistent challenge in the public sector, especially during the procurement of high-cost capital works. This practice undermines the integrity of the procurement process, hindering innovation and quality while risking budget inefficiencies. Therefore, there's a necessity for a robust detection methodology capable of identifying collusion among participants engaged in public procurement processes. We aim to develop methodologies for detecting collusion within Italy's public procurement system and identify which auctions might be collusive.

### 1.1 Research Problem Description

Collusion detection in public procurement is crucial as it safeguards fairness, transparency, and efficiency in government acquisitions. By preventing bid manipulation and price-fixing among suppliers, it ensures a level playing field, ultimately saving costs, curbing corruption, and optimizing the use of taxpayers' money. Detecting collusion fosters trust, accountability, and healthy competition, crucial for maintaining integrity in procurement processes and maximizing

the value derived from public spending, thus benefiting both suppliers and the public. Which also leads to better supplier selection and quality of supply or work delivered.

This research focuses on improving the accuracy of identifying collusion in public procurement. It does this by using advanced machine learning techniques commonly used in various industries. Specifically, this master's thesis aims to boost the effectiveness of current collusion detection methods by refining parameters and optimizing features. This enhanced approach is expected to yield better overall performance in identifying collusion instances. We seek to explore several research questions. Firstly, within the realm of Italian public procurement data, our primary aim is the identification of collusive auctions. Secondly, we aim to evaluate the impact of optimizing hyperparameters on the accuracy of our predictive results. Additionally, our goals include determining the extent to which feature engineering contributes to enhanced accuracy and assessing whether hyperparameter tuning improves the accuracy of collusion detection in public procurement.

### 1.2 Structure of Thesis

The thesis will consist of seven main parts. The second chapter, named "Literature Review," will thoroughly examine current methodologies used in detecting collusion in various domains including public procurement. It sets the groundwork for this study by discussing how collusion detection is approached in academia. Chapter three, "Dataset," will cover where the data comes from, how it's processed, and how features are created. Subsequently, Methodology," the following chapter, summarises the models utilized, their structures, and the techniques applied for screening variables. Chapter 5 covers the model framework, diverse settings utilized, and various error metrics employed. Chapter 6 provides the results of each model. Finally, the last chapter, "Conclusion," will sum up the main findings, research limitation, and suggest areas for future research.

## Chapter 2 LITERATURE REVIEW

The goal of this chapter is to examine different methods to detect collusion in public procurement practices. This section delves into how collusion is detected in various domains. Recently, the rise of machine learning has opened new possibilities for identifying collusion across various sectors. These algorithms can sift through extensive data sets, analyze patterns, and flag irregularities that might signal collusion. Leveraging data-driven approaches holds the potential to boost openness and responsibility in public procurement processes. In this literature review, we aim to present a comprehensive look at existing research concerning the identification of collusion in public procurement, specifically focusing on the use of machine learning algorithms. We'll explore the main ideas, techniques, challenges, and achievements in this developing area, highlighting how machine learning could serve as a valuable tool in combating collusion and corruption in public procurement. Additionally, this chapter discusses the use of screening variables and different evaluation metrics in detection and surveys previous research that employed these methods. This chapter offers a detailed analysis of the latest strategies and approaches shaping collusion detection based on the most pertinent studies available.

## 2.1 Collusion in Public Procurement Practices and impact

Public procurement plays a vital role in the supply chain, holding significant importance in its operations and overall functionality. A significant problem in public procurement is bid rigging or collusion, which disrupts fair competition and transparency. These practices not only impede efforts aimed at improving the supply chain but also lead to the selection of subpar suppliers, a decline in product quality, inflated prices, and a loss of transparency. The prevalent forms of procurement fraud and corruption often involve bid-rigging, collusion between vendors and employees, and collusion between vendors.

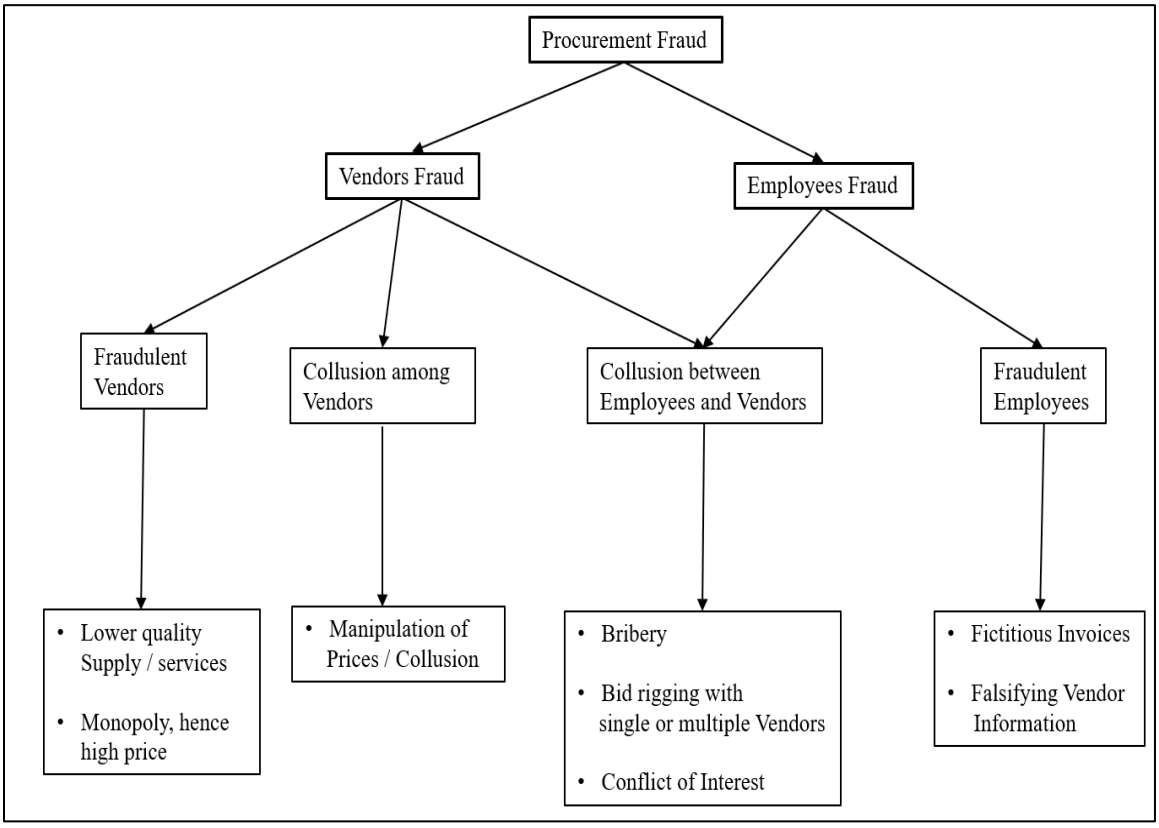


**Figure 2.1:** Fraud taxonomy related to procurement fraud by (Dhurandhar et al., 2015)

Figure 2.1 outlines the stages of procurement where fraud, encompassing bid rigging, bribery, false invoicing, and collusion, can occur. When several bidders, frequently rivals, work together covertly to manipulate the bidding process for their mutual gain, this is referred to as collusion in public procurement. Which undermines the principles of fair competition, transparency, and efficiency that are the foundation of public procurement. Collusive practices among bidders often emerge as a profound and pressing concern in public procurement auctions (Pfizer & Jakobsson, 2007). The impact of corruption on average amounts to 5% of the total value of public procurement, which translates to about 14% of the European Union's GDP, or EUR 1.9 trillion within the EU; which is why efforts have been put in the field of corruption definition and detecting suspicious actions (Dahlberg, 2001). A World Bank study identifies critical technology trends to combat public sector fraud and corruption. These include big data, artificial intelligence,

biometrics, blockchain, FinTech, and IoT, showcasing the potential for technological innovation. The OECD estimates that eliminating bid rigging could lower procurement costs by 20% or even more (Public Procurement - OECD, 2012). Creating pre-emptive approaches to detect bid-rigging schemes is of paramount significance for competition and procurement authorities worldwide.

### 2.2 Corruption/ anomaly detection methods in different sectors:

The initial efforts in identifying anomalies were undertaken within the realms of the telecommunications, insurance, and banking sectors. This process demanded a substantial investment of time and expertise spanning multiple domains, such as legal, financial, commercial, and more (Bolton & Hand, 2002). It's crucial to highlight that the research in this domain primarily centers on the development of predictive models, as well as the identification of relationships between economic entities and contracting agencies. Fundamentally, this is a multifaceted subject, encompassing statistical techniques, a variety of data mining methodologies, and the integration of machine learning. The literature demonstrates the use of two widely recognized approaches, specifically supervised and unsupervised learning methods. These approaches diverge in their target variables, with supervised learning relying on well-defined output variables, and unsupervised learning operating without predetermined variables, making it suitable for anomaly detection. Table 2.1 recapitulates the key features of both types of learning.

**Table 2.1:** Key features for algorithms

| Algorithm type | Key Features |
|---|---|
| Supervised | This method analyzes samples of data (input/ output pairs) that have been categorized beforehand (labeled data) to establish the mapping function between them. |
| Unsupervised | This method uncovers patterns in input data without requiring prior knowledge of output data labels (using unlabeled data). |

In the context of anomaly detection for financial data, two significant domains of application are network security and fraud detection. Network security is concerned with identifying atypical patterns within the vast and ever-changing landscape of network traffic, amidst a multitude of standard signals (García et al., 2009). In contrast, fraud detection primarily centers on the detection of unusual alterations in spending behavior over time, often relying on the expertise and experience of investigators. In both application areas, the fundamental approach to anomaly detection is consistent: it involves the establishment of a baseline that characterizes "normal" conditions, followed by a comparison of individual observations against this baseline to flag and identify behaviors or patterns that deviate from the norm. This process serves as the foundation for robust anomaly detection in financial contexts. Broadly, most studies indicate that

fraud detection models typically involve a series of distinct steps for supervised machine learning as shown in Figure 2.2.

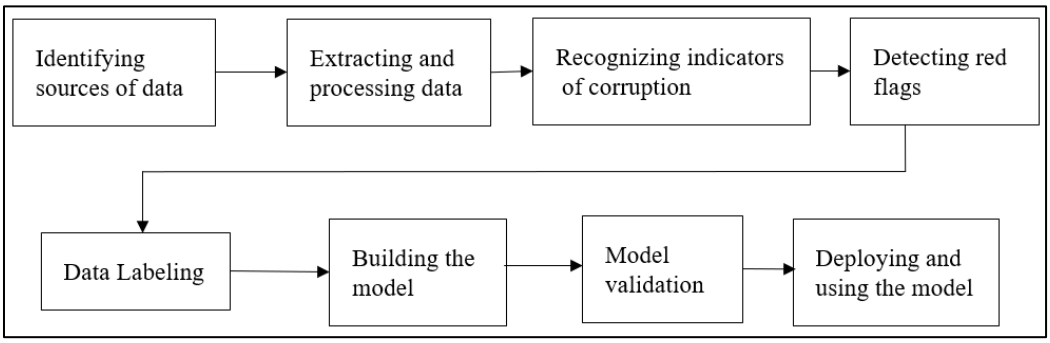


**Figure 2.2:** Fraud detection model

New approaches are being constantly under investigation and, various statistics and machine-learning initiatives have exhibited promising outcomes in the realm of fraud detection (C. Phua et al., 2013). Commonly used methods in studies include linear and logistic regression, neural networks, and Naive Bayes for classification and clustering. In simple terms, these models are trained on past data and aim to create an early warning system. This system can offer goverments advance information about the risks when they make agreements with potentially risky businesses (Gallego et al., 2021). Most frequently, the prevalent approaches, as mentioned earlier, center on supervised learning or classification methods when there is labeled data at hand. However, certain researchers have effectively applied unsupervised learning, demonstrating the ability to identify patterns and extract valuable hidden insights even when labeled data is lacking (Bolton & Hand, 2019). Bolton & Hand employed unsupervised techniques for anomaly detection in a study dating back to 2001, which centered on the detection of credit card fraud (Bolton & Hand, 2019). This research illustrated the feasibility of identifying patterns and revealing valuable hidden insights, even when labeled data was unavailable.

### 2.3 Collusion detection methods in public procurement:

The detection of corruption in public procurement has emerged as a significant global concern in recent times. Given the substantial volume of services and financial transactions involved in public procurement, it has become imperative to identify and prevent any instances of corrupt practices. Identifying collusion is a difficult task that calls for the use of a wide range of statistical methods, different data mining techniques, and machine learning methodology. Numerous studies in auction theory show that bidders' cost structures greatly impact how they compete or collude. McAfee and McMillan were the first to examine collusion in bid rotation schemes without compensation between cartel members (Mcafee & Mcmillan, 1992). Their auction model didn't tie the winner to past auctions. Aoyagi and Skrzypacz, expanded this model, exploring repeated collusion in evolving bid rotation schemes (Aoyagi, 2003). Other research has investigated how collusion happens and its impact in actual procurement auctions (Harrington, 2004). There are very few empirical-based collusion detection models available in the literature. One of the early attempts to develop an empirically based model was by Porter and Zona. They aimed to calculate the likelihood of a bidder winning given some known observable cost elements.

The goal of that approach was to predict the price range of the next (competitive) bids, not to identify collusion per se (Porter & Zona, 1999).

The model introduced by Signor et al., is a probabilistic approach (Signor et al., 2023a). Signor et al.'s model conducts a dual-level analysis of submitted bids. Firstly, it assesses whether the overall distribution of bids adheres to a predefined reference scenario, such as a Lognormal distribution. Additionally, it approximates the location of this distribution, specifically the absolute order of bid magnitudes, using historical auction data whenever pre-tender estimates (PTE) information is accessible. As a result, the model evaluates the deviation of submitted bids from the PTE (Signor et al., 2023a).

Signor et al. have introduced a probabilistic model for analyzing bids in a more straightforward manner (Signor et al., 2023a). This model operates at two levels. First, it checks if the overall distribution of bids matches a standard pattern, like a Lognormal distribution. Furthermore, it estimates the location of this distribution, which refers to the actual bid values, based on historical auctions when pre-tender estimate (PTE) data is accessible. Consequently, the model assesses how far the submitted bids deviate from these estimates. Secondly, the model's probabilistic approach deals with the spread of the lowest bid using order statistics theory. In simpler terms, it evaluates whether the lowest bid (the potential winner) could have realistically come from the same distribution as in the previous step. In this method, the actual winning bid is compared to the lowest order statistic, which is the smallest value drawn from a reference distribution. If there's a substantial difference, it indicates that the bid may not be genuinely competitive (Signor et al., 2023). This probabilistic method is robust, but its effectiveness depends on having reliable PTE data from past honest auctions and the auction currently under examination.

A technique to evaluate markets for suspected cartel activities was developed by identifying disparities in bid behavior between competitive and non-competitive firms (Bajari & Ye, 2003). If firms are competitive, they should submit proposals that are not influenced by those of their rivals. Conversely, when collusion is suspected, bids may exhibit correlations, often attributed to the submission of "phantom" bids aimed at simulating a competitive environment. The hypothesis of conditional independence can be empirically examined, and if it is found to be false, collusion is a more likely explanation. Although the model exhibits notable constraints in academic research, including a significant dependence on the selected functional form during regression analysis, heightened susceptibility to missing data, and vulnerability to manipulation by cartels with a deep understanding of its inner workings, such as coordinated cover bids. Fortunately, after Bajari and Ye's (Bajari & Ye, 2003) research, an increased availability of public data regarding public contracts and competitor information has emerged. This data presented promising opportunities for enhancing collusion detection through machine learning techniques.

The research conducted by Rodríguez et al. explores the use of machine learning algorithms for spotting potential collusion in public procurement auctions (García Rodríguez et al., 2022). The paper focuses on leveraging these algorithms to detect suspicious behaviors among auction participants, ultimately aiming to enhance collusion detection and ensure fairness in these auction processes. The study likely highlights the machine learning models' adeptness in analyzing vast data sets, identifying irregularities that might signify collusion among bidders. However, inherent limitations in this approach are anticipated to be addressed. These limitations may arise

due to the complexity of auction dynamics, the critical need for diverse and high-quality data, and the adaptability of colluding parties in masking their actions. The effectiveness of machine learning models might heavily rely on comprehensive data availability and adherence to legal and ethical data usage. In conclusion, while emphasizing the potential of machine learning in detecting collusion, the study acknowledges the need for ongoing improvements and interdisciplinary collaborations. It also underscores the importance of hyperparameter tuning and exploring other models to increase the accuracy.

Imhof's model marked a significant step forward by using machine learning (ML) to examine bids and spot collusion (Huber & Imhof, 2019). Imhof used a limited set of Screening Variables (SV) in a dataset related to road construction in Switzerland. Furthermore, Imhof employed two types of ML algorithms: (1) Lasso regression and an 'Ensemble method' that combines multiple algorithms through a weighted average, and (2) bagged regression trees, random forests, and neural networks (Huber & Imhof, 2019). These Screening Variables played a crucial role in preventing and detecting bid rigging in public procurement and competitive markets (Huber & Imhof, 2019). Similarly, in their exploration of collusion detection using machine learning in public procurement auctions, Rodríguez et al. integrated Screening Variables (SV), crucial in identifying potential collusion among auction participants. The incorporation of Screening Variables aligns with their pursuit to enhance the precision and effectiveness of collusion detection mechanisms within these auction processes (Rodríguez et al., 2022). In our study, we plan to build upon Imhof's approach by leveraging a similar dataset and screening variables but will explore a wider array of machine learning models, diverse settings, and rigorous hyperparameter tuning to assess and enhance performance.

To assess the performance of the machine learning models for classification tasks, it is essential to establish specific error metrics. The most frequently employed error metrics in the classification tasks include accuracy, precision, recall, balanced accuracy, and the F1 score (Sokolova & Lapalme, 2009). Furthermore, the Receiver Operating Characteristic (ROC) curve visually represents a binary classifier's performance. The area under this curve (AUC) offers a comprehensive assessment of the model's ability to distinguish between two classes, providing a more nuanced evaluation of its capabilities (Sokolova & Lapalme, 2009). In our study, we plan to use accuracy, precision, balanced accuracy, and ROC-AUC score to assess the performance of all models. Previous work in collusion detection within public procurement primarily relied on accuracy, precision, and balanced accuracy for assessing performance. However, the ROC graph and AUC score offer metrics independent of threshold values, which most researchers haven't incorporated alongside accuracy, precision, and balanced accuracy specially within public procurement domain. This study incorporates all these metrics, including ROC-AUC, to assess classifier performance. This research underscores the significance of carefully choosing evaluation metrics.

### 2.4 Summary table

A comprehensive summary table consolidating the research conducted in this domain provides a concise overview of recent findings. This table systematically presents various methodologies, evaluation metrics, data processing strategies, and hyperparameter tuning methods

employed across these studies, aiming to furnish a holistic comprehension of the current research landscape. Most studies as shown in table 2.3, did not opt for hyperparameter tuning or feature engineering. While Decarolis & Giorgiantonio, and Rodríguez focused on feature engineering without engaging in hyperparameter tuning (Decarolis & Giorgiantonio, 2022;Rodríguez et al., 2022). This highlights the need for a more comprehensive consideration of tuning practices in the field. One finding from this collection of studies is the absence of the ROC-AUC score as an evaluation metric. Instead, the predominant focus was on metrics like accuracy or balanced accuracy.

**Table 2.2:** Summary of Literature Review Table

| RESEARCH | METHODS | INPUT DATA | TUNING/ SOFTWARE |
|---|---|---|---|
| Dhurandhar et al., 2015 | Text analysis, ranking, unsupervised learning | RFx data | Social network analysis |
| Decarolis & Giorgiantonio, 2022 | Lasso, ridge regression, and RF. | Italian dataset managed by the ANAC | Feature engineering/ tuning not done |
| Ghedini Ralha & Sarmento Silva, 2012 | Clustering, association rules, multi-agent approach | Brazilian Public Procurement | Not available |
| Bolton & Hand, 2013 | Behavioral outlier detection, unsupervised learning | Credit card data | Peer Group Analysis |
| Domingos et al., 2017 | CRISP-DM | Brazilian Public Procurement | Not available |
| Signor et al., 2023 | Probabilistic methods | Brazilian Federal Police's ongoing “Operation Car Wash” investigation data | Not available |
| Torres-Berru & Batista, 2021 | Clustering(K-Means), SOM, SVM, PCA | Public Procurement System (SERCOP) of Ecuador | Technologies: Python Scikit-learn library, MiniSom, AZURE Machine Learning/ tuning not done |
| Wallimann et al., 2022 | RF | See-Gaster and Graubünden in Switzerland procurement data | Feature engineering/ tuning not done |
| Conley & Decarolis, 2016 | Standard hierarchical clustering algorithm | City of Turin dataset | Not available |
| Rodríguez et al., 2022 | SGD, Extra Trees, RF, Ada Boost, Gradient Boosting, SVC (C- | Public procurement open data from Brazil, Italy, Japan, | Technologies: Python, Used |

| | | | |
|---|---|---|---|
| | Support Vector classification), KNN, MLP, Bernoulli NB and Gaussian Naive Bayes. Technologies: Python and Scikit-learn library. | Switzerland, and USA | Screening variables/ Tuning not done |
| Rabuzin & Modrušan, 2019 | NLP, NB, LR, SVM. | Electronic Public Procurement of Croatia | Technologies: Python / Tuning not done |
| Arief & Asnar, 2017 | NB, Bayesian networks, decision trees, and neural network. | Indonesian Public Procurement | Tuning not done |

Despite extensive research in collusion detection, the utilization of machine learning in this domain is still in its early stages. It's crucial to note numerous limitations, particularly the absence of screening variables in many studies. Even among those that do consider such variables, the lack of parameter tuning in machine learning models is apparent. These limitations highlight the need for further research and methodological advancements in detecting collusion in public procurement. To address this gap, my work distinguishes itself from prior studies by meticulously fine-tuning parameters across multiple models and integrating screening variables with machine learning techniques. The objective is to investigate whether this approach enhances the current capability to identify bid-rigging cartels. We'll employ a range of evaluation metrics—such as accuracy, balanced accuracy, precision, and ROC-AUC score—to thoroughly assess and compare the performance of diverse models.

# Chapter 3 DATASET

To assess the collusive detection capabilities of machine learning algorithms under various conditions, we acquired public procurement datasets from Italy, covering the period between 2000-2003. The dataset has been obtained from the study conducted by García Rodríguez in 2022 (Rodríguez et al., 2022).

## 3.1 Data Information

The Italian dataset encompasses auctions related to road construction projects conducted by the municipal authorities of Turin. A quantitative description of the datasets is presented in Table 3.1. It is important to note that these datasets have been examined and/or provided by public institutions, specifically the courts of justice in Italy. Consequently, we presume the data's reliability and trustworthiness (Rodríguez et al., 2022). The auctions within this dataset are based on the Average Bid Auction (ABA) method, where the contract is awarded to the bidder whose bid is closest to a trimmed average bid (Conley & Decarolis, 2016b). This auction mechanism has the potential to incentivize collusion among bidders, as they may coordinate their bids with other participants with the aim of manipulating the overall bid distribution (Conley & Decarolis, 2016b).

**Table 3.1:** Summary of Italian Road Construction Procurement Dataset (2000-2003)

| General Information | Road Construction public procurement dataset from Italy |
|---|---|
| Year | 2000-2003 |
| No. of auctions | 278 |
| No. of bids | 20,286 |
| Average No. bids per auction | 72.97 |
| Total number of bids | 20,286 |
| Collusive bids | 8085 (39.86%) |
| Competitive bids | 12,201 (60.14%) |
| Aggregated total | €11,520,750,772 |
| Aggregated collusive | €7,911,773,729 (68.67%) |
| Aggregated Competitive | €3,608,977,044 (31.33%) |

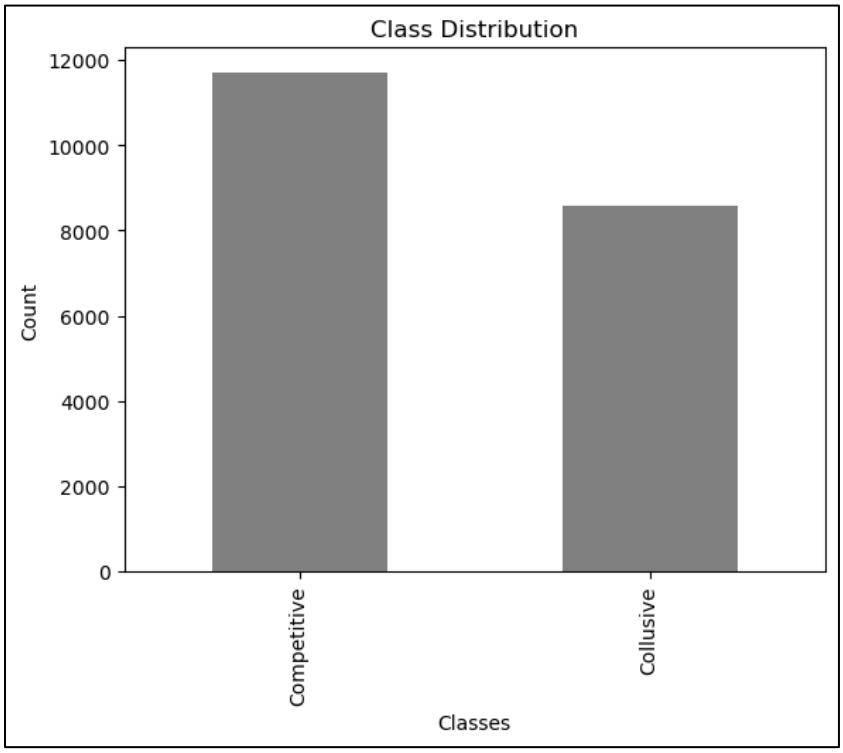


**Figure 3.1:** Class distribution

Figure 3.1 presents the class distribution for both categories. The dataset utilized in this study exhibits a minor imbalance, with one class representing collusive bids, accounting for

39.86% (8,085 instances), and the other class representing competitive bids, constituting 60.14% (12,201 instances). Collusive bids involve unethical collaboration among bidders, aiming to manipulate the bidding process by agreeing not to compete or artificially inflating prices. In contrast, competitive bids signify an open and fair competition where bidders independently submit their bids for a contract by offering their best prices. In table 3.2, a tender is an official request for competitive bids from potential suppliers or service providers. A Pre-Tender Estimate (PTE) is an early assessment of the anticipated project cost before the formal bidding process. The winner of a tender is the entity whose bid is accepted. Cartel Name signifies the identifier for a collusive group of businesses working together to manipulate markets. A bid value represents the amount submitted by a bidder in response to a tender, while the number of bids indicates the total count of proposals submitted by different entities, reflecting the level of competition for the opportunity. A sample of data used for this research is shown in table 3.2:

**Table 3.2:** Sample of dataset

| Tender | Difference Bid/PTE | Winner | Competitors | Legal_entity_type | Site | Capital | Pre-Tender Estimate (PTE) | Bid_value | Cartel_name | Collusive_competitor | Number_bids |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 33.41122 | 0 | 80170392 | 3 | 255.39 | 80000 | 1187851 | 79097540 | 1 | 1 | 63 |
| 0 | 30.11963 | 0 | 105230056 | 10 | 35.067 | 520000 | 1187851 | 83007458 | 0 | 1 | 63 |
| 0 | 28.58136 | 0 | 128540358 | 7 | 164.948 | 80000 | 1187851 | 84834694 | 0 | 1 | 63 |
| 0 | 44.84511 | 0 | 133910935 | 10 | 307.1731 | 1463245 | 1187851 | 65515784 | 0 | 1 | 63 |
| 0 | 27.57113 | 0 | 140990409 | 5 | 251.896 | 80000 | 1187851 | 86034696 | 1 | 1 | 63 |
| 0 | 45.33199 | 0 | 155570021 | 10 | 36.599 | 200000 | 1187851 | 64937443 | 0 | 1 | 63 |
| 0 | 34.31618 | 0 | 159170042 | 11 | 37.83399 | 50000 | 1187851 | 78022583 | 0 | 1 | 63 |
| 0 | 35.10052 | 0 | 167670041 | 10 | 60.73399 | 710000 | 1187851 | 77090904 | 0 | 1 | 63 |
| 0 | 28.34308 | 0 | 185120045 | 10 | 69.083 | 2000000 | 1187851 | 85117735 | 0 | 1 | 63 |
| 0 | 32.81166 | 0 | 228840278 | 7 | 250.042 | 80000 | 1187851 | 79809728 | 1 | 1 | 63 |
| 0 | 38.34854 | 0 | 238930010 | 11 | 0.347 | 208000 | 1187851 | 73232740 | 1 | 1 | 63 |
| 0 | 32.45334 | 0 | 276690823 | 11 | 991.0822 | 30988 | 1187851 | 80235359 | 1 | 1 | 63 |
| 0 | 34.17144 | 0 | 287960512 | 2 | 311.137 | 80000 | 1187851 | 78194512 | 1 | 1 | 63 |
| 0 | 30.9313 | 0 | 308900281 | 11 | 236.891 | 100000 | 1187851 | 82043315 | 0 | 1 | 63 |
| 0 | 30.75139 | 0 | 323030429 | 7 | 320.764 | 80000 | 1187851 | 82257022 | 0 | 1 | 63 |

## 3.2 Feature engineering

Feature engineering in collusion detection involves incorporating screening variables, which are specialized indices derived from bid value distributions in auctions. These Screens aid machine learning algorithms by efficiently processing auction data, enhancing the detection of collusion. While there is limited research on the performance of various Screens in collusive datasets, these indices offer a means not only to flag potential collusion in auctions but also to identify sustained collusive patterns among specific bidders (Imhof & Wallimann, 2021) . Screens typically involve statistical indices calculated directly from bid values or after selecting specific bids, such as the lowest and highest bids. Their ease of calculation and demonstrated effectiveness in machine learning algorithms make Screens valuable tools in detecting abnormally high bids and improving overall model performance. Total of seven screening variables- CV, SPD, DIFFP, RD, KURT, SKEW & KSTEST, were calculated using the formulas mentioned in section 3.3 and added to the raw data to improve the collusion capabilities of Machine Learning models. Post the incorporation of all six screens into the raw data, the final data used for modelling is presented in Table 3.3 as follows:

**Table 3.3:** Dataset snapshot after adding screens

| Tender | Difference | Winner | Competitors | Legal_enti | Site | Capital | Pre-Tende | Bid_value | Cartel_na | Collusive_ | Number_t | CV | SPD | DIFFP | RD | KURT | SKEW | KSTEST |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 33.41122 | 0 | 80170392 | 3 | 255.39 | 80000 | 1187851 | 79097540 | 1 | 1 | 63 | 0.0863 | 0.6147 | 0.0045 | 0.0469 | 2.4209 | 0.287 | 0.3725 |
| 0 | 30.11963 | 0 | 105230056 | 10 | 35.067 | 520000 | 1187851 | 83007458 | 0 | 1 | 63 | 0.0863 | 0.6147 | 0.0045 | 0.0469 | 2.4209 | 0.287 | 0.3725 |
| 0 | 28.58136 | 0 | 128540358 | 7 | 164.948 | 80000 | 1187851 | 84834694 | 0 | 1 | 63 | 0.0863 | 0.6147 | 0.0045 | 0.0469 | 2.4209 | 0.287 | 0.3725 |
| 0 | 44.84511 | 0 | 133910935 | 10 | 307.1731 | 1463245 | 1187851 | 65515784 | 0 | 1 | 63 | 0.0863 | 0.6147 | 0.0045 | 0.0469 | 2.4209 | 0.287 | 0.3725 |
| 0 | 27.57113 | 0 | 140990409 | 5 | 251.896 | 80000 | 1187851 | 86034696 | 1 | 1 | 63 | 0.0863 | 0.6147 | 0.0045 | 0.0469 | 2.4209 | 0.287 | 0.3725 |
| 0 | 45.33199 | 0 | 155570021 | 10 | 36.599 | 200000 | 1187851 | 64937443 | 0 | 1 | 63 | 0.0863 | 0.6147 | 0.0045 | 0.0469 | 2.4209 | 0.287 | 0.3725 |
| 0 | 34.31618 | 0 | 159170042 | 11 | 37.83399 | 50000 | 1187851 | 78022583 | 0 | 1 | 63 | 0.0863 | 0.6147 | 0.0045 | 0.0469 | 2.4209 | 0.287 | 0.3725 |
| 0 | 35.10052 | 0 | 167670041 | 10 | 60.73399 | 710000 | 1187851 | 77090904 | 0 | 1 | 63 | 0.0863 | 0.6147 | 0.0045 | 0.0469 | 2.4209 | 0.287 | 0.3725 |
| 0 | 28.34308 | 0 | 185120045 | 10 | 69.083 | 2000000 | 1187851 | 85117735 | 0 | 1 | 63 | 0.0863 | 0.6147 | 0.0045 | 0.0469 | 2.4209 | 0.287 | 0.3725 |
| 0 | 32.81166 | 0 | 228840278 | 7 | 250.042 | 80000 | 1187851 | 79809728 | 1 | 1 | 63 | 0.0863 | 0.6147 | 0.0045 | 0.0469 | 2.4209 | 0.287 | 0.3725 |
| 0 | 38.34854 | 0 | 238930010 | 11 | 0.347 | 208000 | 1187851 | 73232740 | 1 | 1 | 63 | 0.0863 | 0.6147 | 0.0045 | 0.0469 | 2.4209 | 0.287 | 0.3725 |
| 0 | 32.45334 | 0 | 276690823 | 11 | 991.0822 | 30988 | 1187851 | 80235359 | 1 | 1 | 63 | 0.0863 | 0.6147 | 0.0045 | 0.0469 | 2.4209 | 0.287 | 0.3725 |
| 0 | 34.17144 | 0 | 287960512 | 2 | 311.137 | 80000 | 1187851 | 78194512 | 1 | 1 | 63 | 0.0863 | 0.6147 | 0.0045 | 0.0469 | 2.4209 | 0.287 | 0.3725 |
| 0 | 30.9313 | 0 | 308900281 | 11 | 236.891 | 100000 | 1187851 | 82043315 | 0 | 1 | 63 | 0.0863 | 0.6147 | 0.0045 | 0.0469 | 2.4209 | 0.287 | 0.3725 |
| 0 | 30.75139 | 0 | 323030429 | 7 | 320.764 | 80000 | 1187851 | 82257022 | 0 | 1 | 63 | 0.0863 | 0.6147 | 0.0045 | 0.0469 | 2.4209 | 0.287 | 0.3725 |

### 3.3 Screening Variables in Bid Rigging Detection:

Screening variables play a pivotal role in identifying bid rigging within procurement procedures by identifying potentially suspicious bidding behavior. These screening variables are integrated as features or attributes in machine learning models to predict or describe certain characteristics of bid distributions. These screens can help machine learning algorithms process auction information more efficiently to detect collusion (Huber et al., 2019.). Screening variables function as initial early warning systems in detecting collusion, pinpointing tenders requiring additional investigation. They offer quantitative evidence that either corroborates or questions the notion of collusion. When integrated into machine learning models, these variables enable detection algorithms to prioritize tenders which have suspicious patterns for more extensive examination. Feinstein in 1985 discovered that within highway construction contracts in North Carolina, bid rigging was associated with a significantly lower coefficient of variation, indicating reduced bid dispersion (Feinstein et al., 1985). Abrantes-Metz in 2012 offered evidence indicating manipulation within daily bank quotes used for Dollar Libor calculation by leveraging the coefficient of variation (Abrantes-Metz et al., 2006). Conversely, Jimenez and Perdiguero in 2012 demonstrated that markets with restricted competition showcased decreased price variability and elevated prices (Jiménez & Perdiguero, 2012). There are three primary types of screening variables used in bid rigging detection:

1. **Variance Screens:** It evaluates the spread or dispersion of bid prices within a tender, where a high variance signifies a broader range of bid values.

**Significance**: During collusion or bid rigging, colluding bidders frequently submit artificially inflated or suppressed bids, aiming to manipulate prices. Such manipulation often results in higher bid price variance compared to genuinely competitive tenders.

2. **Asymmetry Screens:** Asymmetry screens center on the shape of the bid distribution, scrutinizing whether it displays skewness toward one side or maintains symmetry.

**Significance**: Bid rigging cartels may tactically submit bids that skew the distribution towards a specific outcome. Detecting such asymmetry aids in recognizing potential collusion.

3. **Uniformity Screens:** Uniformity screens assess the evenness of bids across different bid categories. Any deviations from uniformity can raise suspicion.

**Significance**: Bid rigging schemes frequently entail collusion in lot or category allocations among conspiring bidders. Identifying deviations from uniform bidding behavior can signal potential collusion.

The following section provides an explanation of the screens used in this thesis, encompassing their definitions and respective formulas:

### 3.3.1 CV

first variance screen is the coefficient of variation (CV), a scale-invariant statistic which is formally defined as follows: where $sd_t$ is standard deviation and $b_t$ the mean of the bids in some tender t.

$$CV_t = \frac{sd_t}{\bar{b}_t} \tag{3.1}$$

### 3.3.2 SPD

The spread (SPD), which is calculated as follows, is another screen associated with the support of the bids. Where $b_{max,t}$ denotes the maximum bid and $b_{min,t}$ the minimum bid in some tender t (Wallimann et al., 2022).

$$SPD_t = \frac{b_{max,t} - b_{min,t}}{b_{min,t}} \tag{3.2}$$

### 3.3.3 DIFFP

The percentage difference (DIFFP) is one screen that looks for manipulations in the discrepancies between the two lowest bids, where $b_{min,t}$ is the lowest bid and $b_{2t}$ the second lowest bid in some tender t.

$$DIFFP_t = \frac{b_{2t} - b_{min,t}}{b_{min,t}} \tag{3.3}$$

### 3.3.4 RD

The relative distance (RD) can be calculated by replacing the denominator in equation (DIFFP), which is the standard deviation of unsuccessful bids as an alternative method for quantifying the difference. $sd_{losingbids,t}$ measures the spread of losing bids around the mean.

$$RD_t = \frac{b_{2t} - b_{min,t}}{sd_{losingbids,t}} \quad 3.4$$

### 3.3.5 SKEW

Another screening criterion is the skewness statistic (SKEW), a commonly used measure of distribution symmetry. where $n_t$ denotes the number of the bids, $b_{it}$ the i th bid, $sd_t$ the standard deviation of the bids, and $\bar{b}_t$ the mean of the bids in tender t.

$$SKEW_t = \frac{n_t}{(n_t - 1)(n_t - 2)} \sum_{i=1}^{n_t} \left( \frac{b_{it} - \bar{b}_t}{sd_t} \right)^3 \quad 3.5$$

### 3.3.6 KURTO

The kurtosis statistic (KURTO) is a statistical measure used to analyze bidding behavior and detect potential collusion or manipulation of bids within a competitive procurement process. Kurtosis is a statistical term that relates to the shape of the probability distribution of a dataset. In this context, it is employed to identify irregular patterns in bidding behavior that may suggest anticompetitive practices. where $b_{it}$ denotes the bid i in tender t, $n_t$ the number of bids in tender t, $sd_t$ the standard deviation of bids, and $\bar{b}_t$ the mean of bids in that tender.

$$KURT_t = \frac{n_t(n_t + 1)}{(n_t - 1)(n_t - 2)(n_t - 3)} \sum_{i=1}^{n_t} \left( \frac{b_{it} - \bar{b}_t}{sd_t} \right)^4 - \frac{3(n_t - 1)^3}{(n_t - 2)(n_t - 3)} \quad 3.6$$

### 3.3.7 KS

The nonparametric Kolmogorov-Smirnov statistic (KS) is used to assess the uniformity in the distribution of bids(Wallimann et al., 2022b) . While competitive bids may not inherently exhibit uniformity, we hypothesize that coordination further disrupts the bid distribution's uniformity. This change can be detected by alterations in the KS statistic. where $n_t$ is the number of bids in a tender, $i_t$ is the rank of a bid.

$$KSTEST_t = max(D_t^+, D_t^-) \; with \; D_t^+ = max_i \left( \frac{b_{it}}{sd_t} - \frac{i_t}{n_t + 1} \right), \quad D_t^- = max_i \left( \frac{i_t}{n_t + 1} - \frac{b_{it}}{sd_t} \right) \quad 3.7$$

# Chapter 4 METHODOLOGY

In this section, we aim to offer an extensive overview of the different models utilized in our study. A total of eight ML algorithms were applied for the comparative analysis. We will also explore the hyperparameter tuning for each model and tuned parameters for best performance. Additionally, we will elaborate on the essential software tools and libraries that played a crucial role in fine-tuning the models and preparing the data. We employed classification methods that are suitable for anomaly detection. Machine learning methods can be used to detect collusion, with each auction being categorized as either "collusive" or "competitive." The algorithms are tasked with performing binary classification for each auction instance. These diverse algorithms collectively address the task of binary classification, aiming to identify collusion in auction scenarios through the application of various machine-learning approaches.

## 4.1 Support Vector Classifier

The support-vector classifier is a novel machine-learning approach designed for solving classification problems. Finding a hyperplane that best divides the training data into two categories is the foundation of support vector classifiers. As shown in figure 4.1, a hyperplane is a line that divides the observed points into two classes in a two-dimensional space. Based on the side of the hyperplane they fall on, observations from the test data are mapped into the space for class prediction. To enhance the accuracy of classification, it's desirable for these data points to be as distant as possible from the hyperplane in the training data, as for these data points confidence in their producing a correct classification will be high. The margin refers to the distance between the hyperplane and the closest data point within either of the two distinct classes. Giving a greater chance of new data being correctly classified, the algorithm chooses a hyperplane with the goal of achieving the greatest possible margin. However, the idea of the hyperplane as a line is a simplification, as a linear hyperplane might perform poorly when the data points are not separable with a line. Support vector machines offer the support vector classifier by enlarging the feature space using kernels and mapping the inputs into high-dimensional feature spaces (Cortes et al., 1995). To distinguish between different classes of data points on a dimensional space, SVM computes the "margin maximum classifier" (Marsland, 2009). The largest circle with no data points formed around a classification border is known as a maximum margin. Support vectors are the nearest data points found next to this margin. The classification of these vectors is thought to be the most challenging. As a result, they are employed as "support" while drawing a categorization model's boundary. The procedure outlined by Marsland in 2009 for a support vector machine for Classification in classification tasks can be summarized as follows (Marsland, 2009).

As shown in figure 4.1, a new instance x is classified by support vector classifier based on its separation from the hyperplane H, which is positioned in the centre of a maximum margin, and the support vector $x_i$. A weight vector $\vec{w}_i$ is positioned perpendicular to the hyperplane, and the class prediction $y_i$ for a new instance is determined by its coefficient on the weight vector. where $y_i$ denotes the class prediction (+1 or -1 in a binary classification), $\vec{w}_i$ represents the weight vectors, K is the kernel function. The calculation of the decision function for support vector classifier is carried out as:

$$f(x) = \sum y_i \vec{w}_i K(x, x_i) \quad 4.1$$

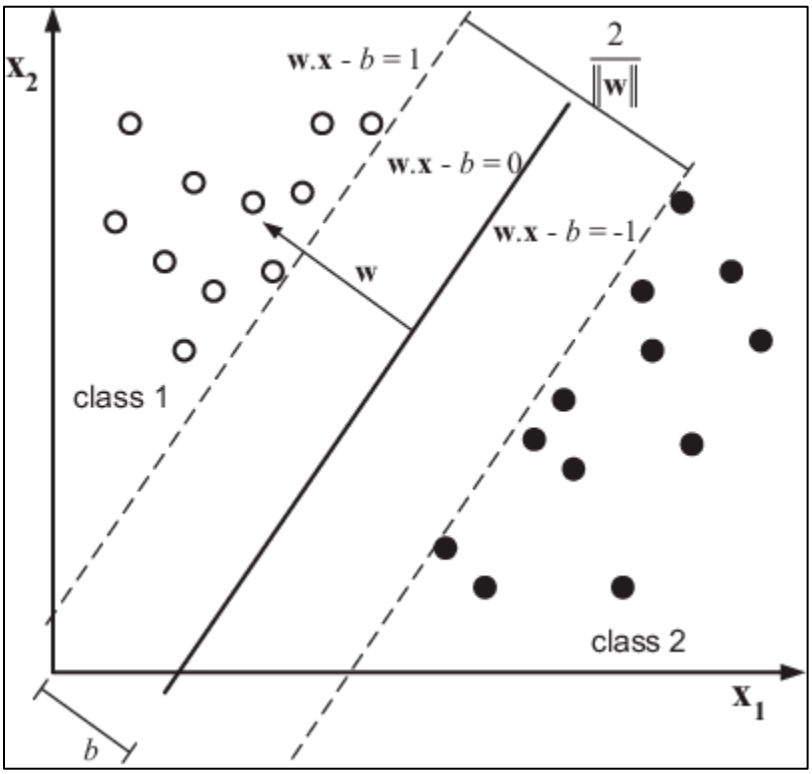


**Figure 4.1:** SVM separating hyperplane.

The baseline model was established using the default settings from the scikit-learn Python library. Within this model configuration, the regularization parameter defaults to one, while the kernel type is defined as "rbf". which stands for Radial Basis Function; it is a kernel function used in Support Vector Machine (SVM) algorithms for various machine learning tasks, including classification. This kernel function is effective in transforming data into higher-dimensional space, allowing SVM models to classify non-linearly separable data by creating non-linear decision boundaries. Subsequently, a grid search methodology was employed to fine-tune these parameters across a range of values. Specifically, for the "rbf" kernel, the regularization parameter spanned from 0.1 to 10, while the gamma ranged between 0.001 to 1 as part of the optimization process.

### 4.2 Random Forest:

Random Forest, a pioneering ensemble learning method, was introduced by Leo Breiman in 2001. This method, shown in figure 4.2, combines the power of multiple decision trees to create a robust, versatile, and high-performance predictive model (Breiman, 2001). Random forest models, which are based on decision trees, have significantly increased prediction accuracy as compared to a single tree by developing 'n' number of trees; each tree in the training set is picked randomly without replacement (Breiman, 2001). A decision tree is a tree-like structure, uses nodes

to represent classification criteria. During training, data moves through internal nodes based on tests until reaching a leaf. Next, a metric is calculated to determine whether the leaf needs to be split and if so, which splitting criterion should be applied. The Gini index and entropy through information gain are the most used measurements. Records go through the tree throughout the testing phase until they come to a leaf (Quinlan, 1986). When using tree-based algorithms for predictions beyond the sample, there exists a balance between bias and variance. By increasing the number of splits, we decrease bias, making the model more adaptable. However, this approach leads to smaller regions and higher variability in test data. To address excessive variance, random forests take multiple subsamples from the training set and build decision trees. In this process, random forest only considers a subset of features at each split to decrease the connection between tree structures across subsamples, aiming to reduce prediction variations (Merentitis et al., 2014).

To determine which feature to split on at each node, the entropy is computed. Entropy measures the homogeneity of the subset data; if entropy equals one then the class labels are equally divided while an entropy of zero means the sample is completely homogeneous (Eq. 4.2). As in the case of binary classification with only two labels, if the split resulted in the class labels being all 1 or 0, then the entropy will be zero.

$$Entropy = -p * \log 2(p) - q * \log 2(q) \qquad 4.2$$

The procedure outlined by Ruppert in 2004 for a random forest in classification tasks can be summarized as follows (Ruppert, 2004).

1. For b = 1 to B:

   (a) Draw a bootstrap sample $Z^*$ of size N from the training data.
   (b) Grow a random-forest tree $T_b$ to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size $n_{min}$ is reached.
      i. Select m variables at random from the p variables.
      ii. Pick the best variable/split-point among the m.
      iii. Split the node into two daughter nodes.

2. Output the ensemble of trees $\{T_b\}_1^B$.

   Classification: Let $\hat{c}_b(x)$ be the class prediction of the $b_{th}$ random-forest tree.

   4.3

   Then $\hat{C}_{Rf}^B(x)$ = majority vote $\{\hat{c}_b(x)\}_1^B$

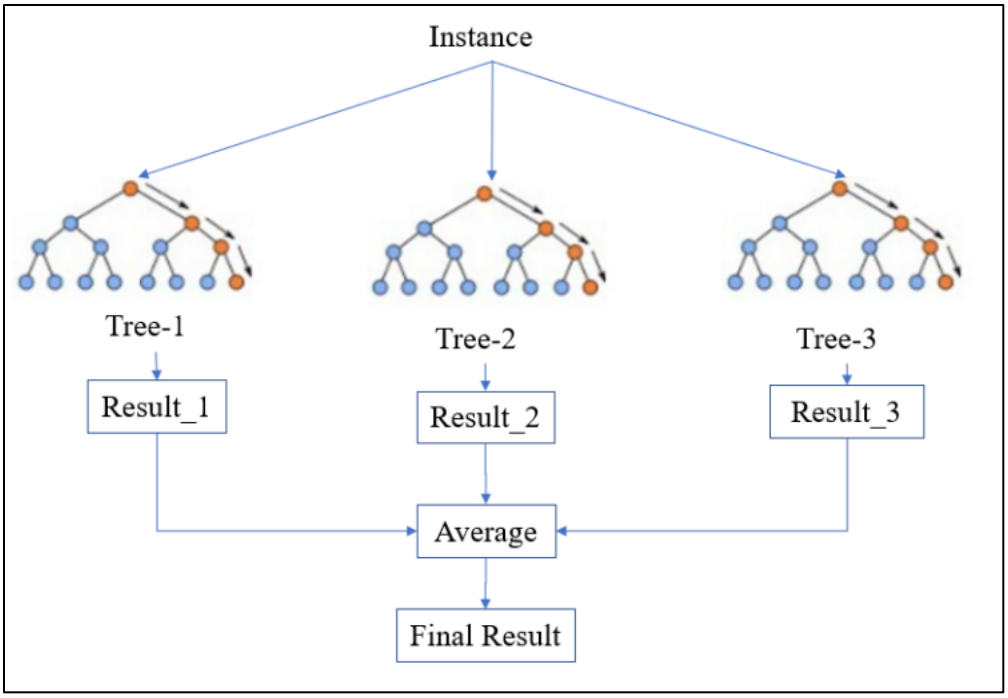


**Figure 4.2:** Ensemble of decision trees (random forest).

The scikit library and the built-in parameters were used to build the basic random forest classifier. The initial model was constructed using scikit-learn's default settings, employed a default criterion, Gini, to measure the quality of a split. The criteria involved a minimum requirement of two samples to split an internal node, one sample for a leaf node, and no restrictions on the maximum depth of each tree within the forest, consisting of one hundred trees. Later, through grid search, these parameters underwent fine-tuning by exploring various values.

### 4.3 KNN: K-nearest neighbours

The k -nearest neighbors (KNN) algorithm is a simple and effective supervised learning technique, used for classification and regression. Evelyn Fix and Joseph Hodges first demonstrated the k-nearest neighbours non-predicting classification algorithm (Fix & Hodges, 1989); which Thomas Cover later expanded on (Altman, 1992). The simple way to assess the similarity of two objects is using Euclidean distance, as outlined by (Dong et al., 2019). This involves comparing the shared attributes of the objects, calculating similarity by summing the squared differences between each term or attribute. The Euclidean distance is then derived from this process. A larger k value results in a broader, less sensitive function with fewer nearby influences. KNN classifies an observation based on its k nearest neighbours. The algorithm operates by considering a proportion of 'k' cases near the observations. For instance, as shown in figure 4.3, when 'k' equals 3, it selects the three nearest data points, and for 'k' equal to 7, it expands to the seven nearest data points. The algorithm then categorizes the new data point based on the majority class within this selected group, represented here by the dark circle. The distances that define an observation's vicinity are Euclidean and are calculated according to the dimensions of the independent variables. It basically puts more than one neighbor and calculates the distance using Euclidean distance, the method which can be calculated with the following equation (Ruppert, 2004):

$$d_e = \sqrt{(p_1 - p_2)^2 - (q_1 - q_2)^2} \quad 4.4$$

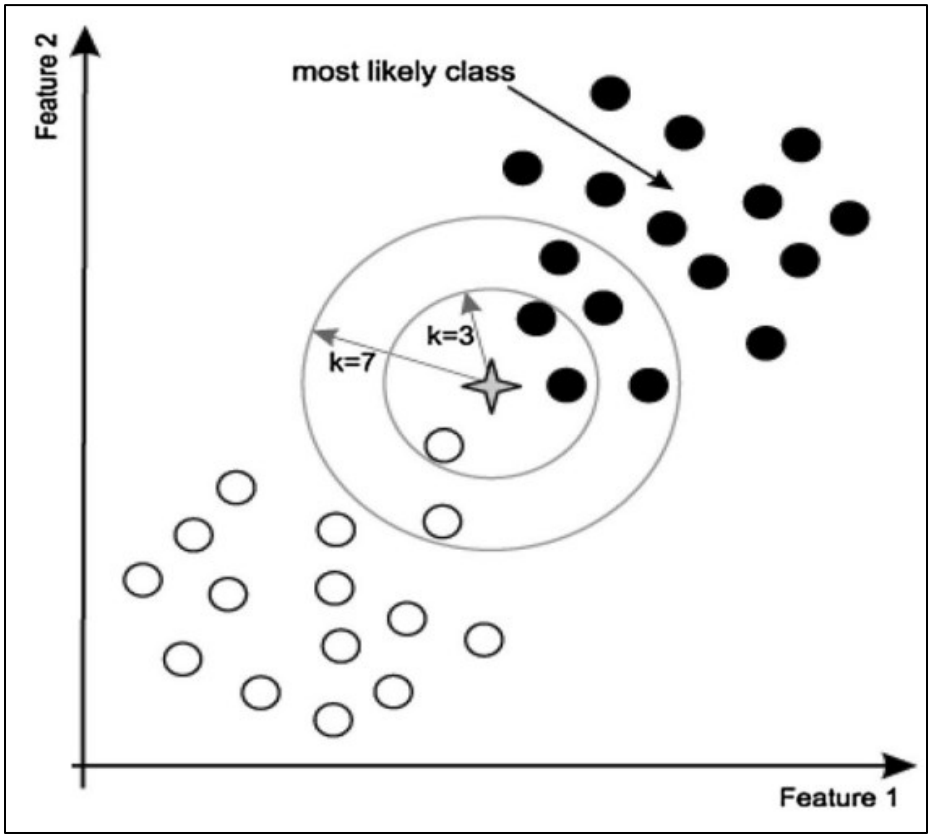


**Figure 4.3:** K-nearest neighbours classification (Sampathkumar et al., 2020)

We built the baseline k-nearest neighbor (KNN) model for classification tasks using the scikit-learn Python module. Initially, we utilized default settings, setting the number of neighbors considered for classification to five, employing uniform weights and a power parameter (p) of 2 for the Minkowski metric. To enhance the model's classification performance, we conducted parameter tuning through grid search, aiming to optimize these settings.

### 4.4 Extra Trees (Extremely Randomized Trees) Classifier:

The Extra Trees Classifier, introduced by Geurts, Ernst, and Wehenkel, is an ensemble learning method that diverges from conventional decision trees by adding an extra layer of randomness (Geurts et al., 2006). As depicted in Figure 4.4, the node divisions in the Extra Tree classifier rely not only on a random subset of features but also on randomly chosen thresholds for these divisions. This stands as the primary distinction between the Extra Tree classifier and the Random Forest classifier, along with the computational time required. Notably, Random Forest tends to take more time to process the same dataset compared to the Extra Tree classifier. The deliberate injection of randomness in extra tree classifier enhances the diversity of individual trees within the ensemble, which, combined with bagging techniques, reduces overfitting, and strengthens the model's resilience to noise in the data (Geurts et al., 2006). This unique

characteristic not only contributes to computational efficiency, owing to the method's parallelizability, but also allows the model to perform well with noisy or high-dimensional data, making it a versatile tool in various machine learning applications. Geurts, Ernst, and Wehenkel demonstrated how the Extra Trees algorithm, with its highly randomized approach to tree construction, can yield competitive performance compared to other ensemble methods like Random Forests, particularly in scenarios with noisy data or limited computational resources (Geurts et al., 2006). Summarized in figure 4.4, this class embodies a meta-estimator which fits multiple randomized trees on varied dataset subsets, employing voting or averaging techniques to enhance accuracy and control overfitting.

Furthermore, the performance of Extra-Trees is influenced by adjusting two key parameters: L (the number of trees) and $n_{min}$ (minimum sample size for node splitting). While $n_{min}$ regulates noise averaging strength, L dictates variance reduction within the forest. Notably, default values for K and $n_{min}$have been indicated as near optimal for various classification problems (Geurts et al., 2006). The classification of extra tree classifier uses Gini impurity by default as in equation 4.5 and uses Entropy as an alternative for the classification as in equation 4.6, whereas for regression, Mean Square Error and Mean Absolute Error are used:

$$GiniImpurity = \sum_{j=1}^{O} f_j(1 - f_j) \quad 4.5$$

$$Entropy = \sum_{j=1}^{O} -f_j log(f_j) \quad 4.6$$

Fj is the frequency of label j at a node and O is the number of unique labels.

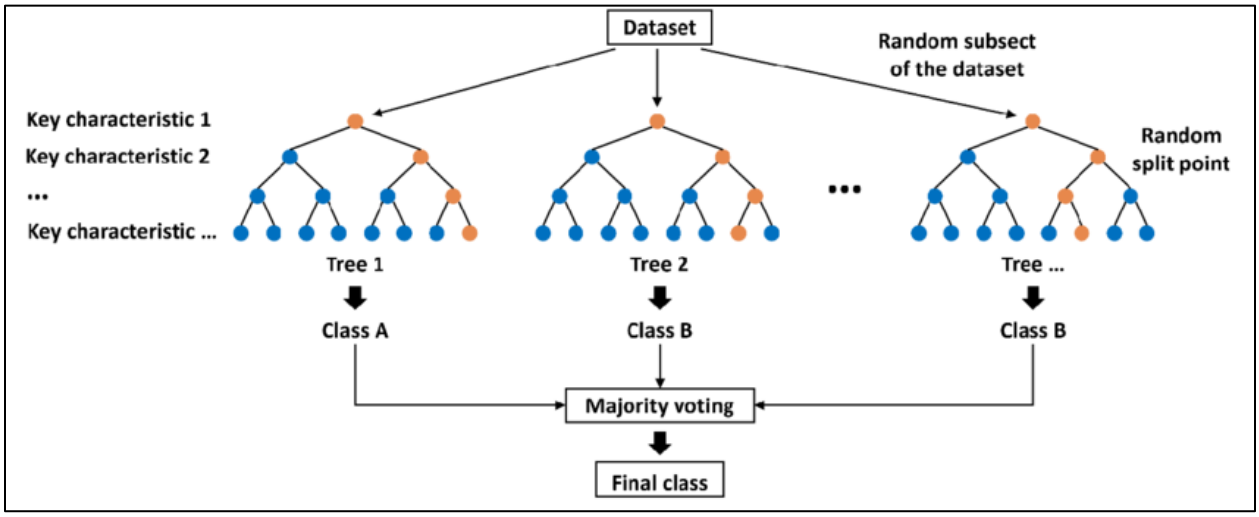


**Figure 4.4:** Structure of Extra Trees classifier

The foundational extra tree classifier was built using the Python module scikit-learn. Initially, default settings were employed, configuring 100 trees for classification, with maximum

depth set to none and a minimum sample split of 2. Subsequently, we engaged in parameter tuning via grid search to refine these settings to improve the model's classification performance.

### 4.5 Gradient Boosting Classifier:

Gradient boosting, known for its remarkable predictive abilities, has garnered extensive popularity across diverse domains as a robust machine learning algorithm. This method was initially proposed by Jerome H. Friedman in 1999 (Friedman, 2001). It involves the combination of multiple weak learners, typically decision trees, to create a strong predictive model. By iteratively correcting the errors of the previous trees, Gradient Boosting optimizes predictive accuracy and is particularly effective in handling complex, non-linear relationships within the data. It has been widely cited as an efficient algorithm for tasks such as classification, regression, and ranking (Friedman, 2001). By minimizing predefined loss functions such as squared error for regression or log loss for classification, it iteratively fits new models to the residuals of previous ones, progressively enhancing predictive accuracy by focusing on prior model mistakes as shown in figure 3.5. Post its inception, gradient boosting witnessed remarkable advancements with optimized implementations like XGBoost, LightGBM, and CatBoost, incorporating enhancements for improved performance and scalability. In this technique, the classification is dependent on the residuals of the previous iteration as shown in figure 4.5 where the impact of each feature is evaluated sequentially until a target accuracy is obtained as compared to other tree based models shown in figure 4.2 and 4.4 where no residuals are taken into consideration. The residuals are calculated by a Loss function $L(\Phi)$ that is optimized using gradient descent. The result $\Phi(X)$ is obtained by the addition of the results of the K sequential classifier functions $f_k$ as follows:

$$\hat{Y} = \Phi(X) = \sum_{k=1}^{K} f_k(X) f_k \in F \qquad 4.7$$

where $f_k$ is a decision tree, and K is the total number of iterations in the boosting algorithm.

Initially applied to regression tasks, gradient boosting quickly found substantial success in binary classification applications. It evolved to accommodate various improvements and adaptations specific to classification problems. Modifications included refined algorithms to handle class imbalances, feature importance estimation, and hyperparameter tuning tailored for classification tasks. These advancements facilitated its widespread adoption across industries for diverse classification challenges, ranging from finance to healthcare, owing to its robustness in handling complex datasets and delivering high predictive accuracy. Subsequent innovations have further refined gradient boosting techniques, enhancing their adaptability, efficiency, and applicability across a wide array of classification problems.

Using the Python module scikit-learn, the foundational gradient boosting classifier was constructed. Initially, default settings were applied, setting the learning rate to 0.1 and the number of boosting stages (n_estimators) to 100. Subsequently, parameter tuning through grid search was conducted to refine these settings, aiming to enhance the model's classification performance.

### 4.5.1 XGBoost (Extreme Gradient Boosting):

Extreme Gradient Boosting is a prominent machine learning algorithm widely recognized for its exceptional predictive capabilities and efficiency in various domains. This algorithm was developed by Tianqi Chen (Chen & Guestrin, 2016). Tianqi Chen asserts that regularisation is added to XGboost to manage the sparse data and weighted quantile sketch for tree learning to optimise the loss function. Additionally, they offer some insights that aid in the development of a quick and scalable tree boosting system (Chen & Guestrin, 2016). These insights include cache access patterns, data sharding, and data compression. XGboost outperforms most other machine learning algorithms in both speed and accuracy because of these mechanisms and insights. As shown in the figure 4.5, XGBoost employes a series of decision trees to iteratively minimize errors and optimize predictive accuracy by combining weak learners into a robust model.

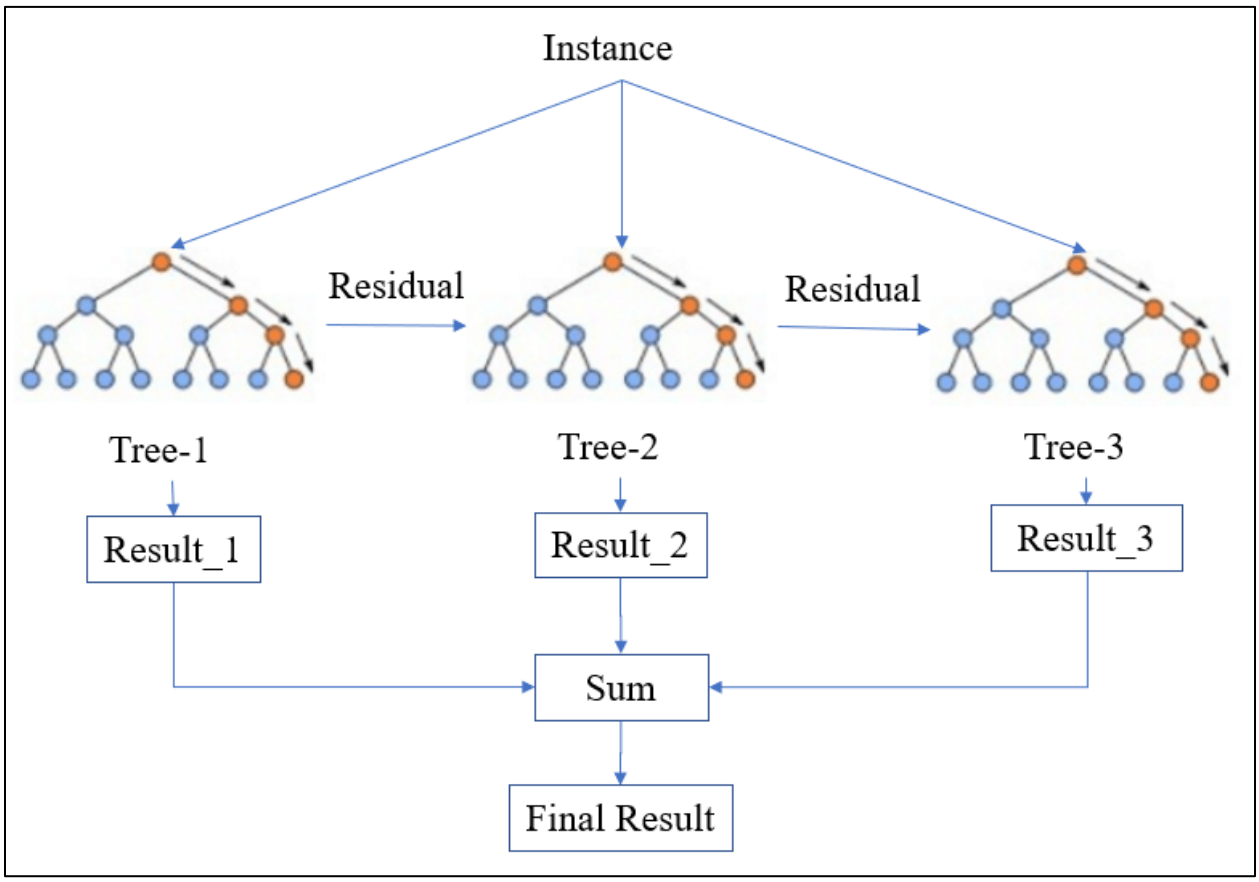


**Figure 4.5:** Simplified structure of XGBoost (Wang et al., 2021)

In this technique, the function to optimize in Step t is called the regularization term $\Omega(f_t)$ and following equation is used to calculate a Loss function $\mathcal{L}(\phi)_t$ at step t.

$$\mathcal{L}(\phi)_t = \sum l(f_{t-1} + f_t) + \Omega(f_t) \quad 4.8$$

Using the Python module scikit-learn, the foundational extreme gradient boosting classifier was constructed. Initially, default settings were applied, setting the learning rate to 0.1 and the number of boosting stages (n_estimators) to 100. Subsequently, parameter tuning through grid search was conducted to refine these settings, aiming to enhance the model's classification performance.

### 4.6 Multilayer Perceptron or Artificial Neural Network

The development of Multilayer Perceptron (MLPs), which are foundational components of modern deep learning, can be attributed to the collective research efforts of scientists and pioneers in the field of artificial intelligence, including Frank Rosenblatt's invention of the perceptron (Rosenblatt, 1960), followed by significant contributions from researchers such as Geoffrey Hinton, Yann LeCun and Yoshua Bengio (Bengio et al., 2007). The Multi Layer Perceptron (MLP) model or Feedforward Neural Network is a fundamental machine learning method that can be used to address a wide range of issues like classification, regression, and approximation. By simulating the weights of the neural connections, it contains, the network learns its behaviour. As shown in figure 4.6 a basic structure of Multilayer Perceptron consists of an input layer, an output layer, and one or more hidden layers. Each neural connection has a weight value assigned to it. The neurons in this network are fully interconnected, which means that each neuron is linked to every other neuron in the layer above it as well as every neuron in the layer below it. Weights in the neural network are adjusted using a training algorithm until desired error criteria are met, refining the network's predictions or classifications.

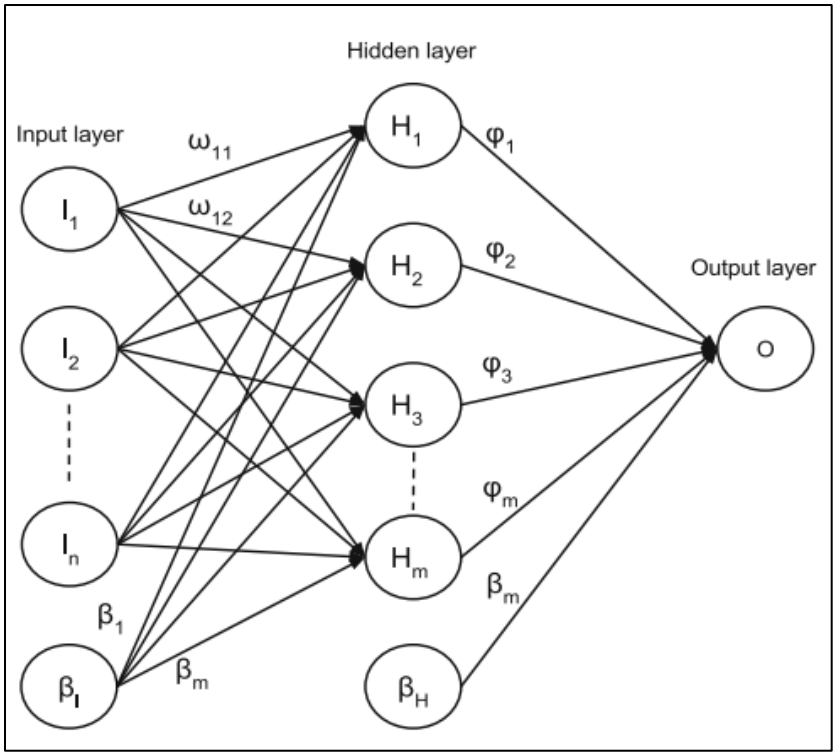


**Figure 4.6:**Basic structure of MLP (Ruppert, 2004)**.**

The activation is given by (Ruppert, 2004):

$$A^l = \sigma\,(W^l * A^{l-1} + B^l) \qquad 4.9$$

Where,
$A^{l-1}$ = Activation in layer
$W^l$= Weight matrix of layer
$B^l$ = Base matrix of layer
$\sigma$ = Activation Function

Using the Python module scikit-learn, the MLP (Multi-Layer Perceptron) classifier was built for binary classification. Initially, default settings were used, followed by parameter tuning through grid search to optimize the model's performance. The initial construction of the basic MLP classifier utilized default parameters: activation = ReLU, alpha = 0.0001, hidden layer size = 100, learning rate = constant, and max iteration = 200. Subsequent to this, a grid search was executed for parameter tuning, aimed at refining these settings to improve the model's classification performance. Which includes an input layer with 240 neurons, succeeded by three hidden layers containing 120, 70, and 35 neurons respectively, all activated by the ReLU function. For the binary classification task, the output layer comprises a single neuron activated by the sigmoid function. This summary shown in figure 4.7 offers a comprehensive overview of the neural network's design.

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_10 (Dense)            (None, 240)               1920

 dense_11 (Dense)            (None, 120)               28920

 dense_12 (Dense)            (None, 70)                8470

 dense_13 (Dense)            (None, 35)                2485

 dense_14 (Dense)            (None, 1)                 36

=================================================================
Total params: 41,831
Trainable params: 41,831
Non-trainable params: 0
```

**Figure 4.7:** Model summary of MLP used

### 4.7 Ensemble ANN-RF

Ensemble ANN-RF combines the strengths of Artificial Neural Networks (ANN) and Random Forests (RF) in a unified method for improved performance. It amalgamates the capabilities of two powerful models - Artificial Neural Networks (ANN) and Random Forests (RF) - to enhance accuracy in discerning between two distinct classes. The independent classification generated by the ANN and RF models are harmonized through a weighted combination approach. Typically, equal weights (50%) are assigned to both models, though variations in weighting strategies can be explored based on their individual performance or other pertinent considerations.

The Ensemble of ANN-RF was implemented through the Python library scikit-learn. Equal weights (50%) were allocated to both models for the final classification. The Random Forest model utilized tuned parameters, setting max features as auto, min samples leaf as 1, n_estimators as 100, and min_samples_split as 2. On the other hand, the ANN configuration included an input layer with 240 neurons, followed by three hidden layers containing 120, 70, and 35 neurons, respectively, employing the ReLU activation function. The output layer for binary classification featured a single neuron activated by the sigmoid function, alongside alpha set to 0.0001.

### 4.8 Ensemble ANN-Gradient Boost

Ensemble ANN-Gradient Boost combines the strengths of Artificial Neural Networks (ANN) and Gradient Boost in a unified method for improved performance. It amalgamates the capabilities of two powerful models - Artificial Neural Networks (ANN) and Gradient Boost - to enhance accuracy in discerning between two distinct classes. The independent classification

generated by the ANN and Gradient Boost models are harmonized through a weighted combination approach. Typically, equal weights (50%) are assigned to both models, though variations in weighting strategies can be explored based on their individual performance or other pertinent considerations.

The Ensemble of ANN- Gradient Boost was implemented through the Python library scikit-learn. Equal weights (50%) were allocated to both models for the final classification. The Gradient Boosting utilized tuned parameters, setting learning rate to 0.01 and the number of boosting stages (n_estimators) to 100. On the other hand, the ANN configuration included an input layer with 240 neurons, followed by three hidden layers containing 120, 70, and 35 neurons, respectively, employing the ReLU activation function. The output layer for binary classification featured a single neuron activated by the sigmoid function, alongside alpha set to 0.0001.

# Chapter 5 MODEL FRAMEWORK

The comprehensive framework comprises multiple distinct stages. Figure 5.1 illustrates the overall system framework. The initial stage involves dataset collection and observation. Subsequently, data preprocessing ensues, encompassing data visualization and computation of screening variables across all bids. Subsequently, the dataset is partitioned into a 80:20 ratio, allocating segments for training, and testing sets, respectively. The training set was employed in the Learning Algorithm, culminating in the development of a final model using optimization techniques. Diverse classifiers detailed in Chapter 4 were utilized within this process. Evaluation of the final model was conducted against the testing set, employing various evaluation metrics. Four distinct settings, elaborated in Table 5.1, were utilized for model assessment. A range of classification evaluation metrics, including Accuracy, Balanced Accuracy, Precision, ROC-AUC score, were applied across all models and settings.

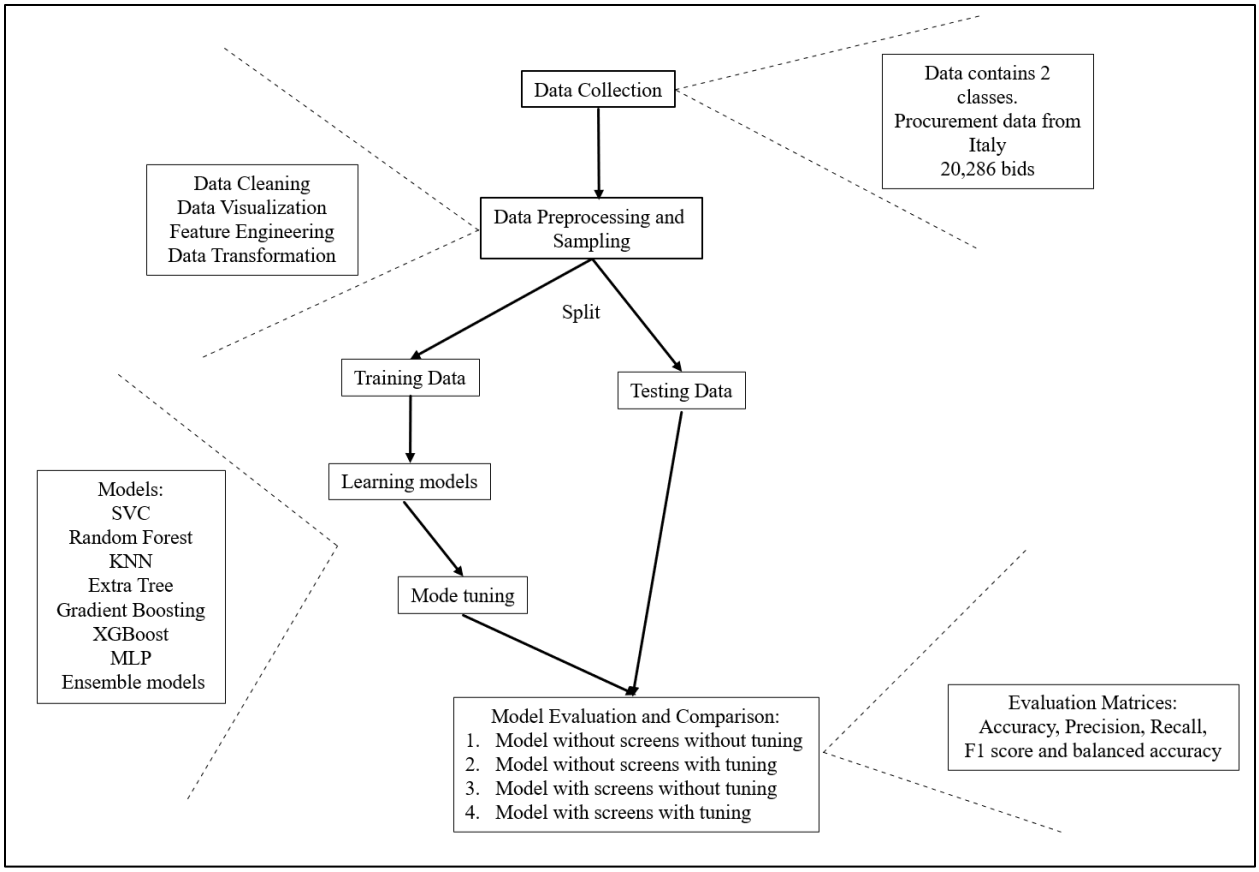


**Figure 5.1:** Overall framework employed for collusion detection.

### 5.1 Different settings

We applied machine learning methods to train and test models for detecting collusion in the public procurement dataset. Specifically, we considered eight models explained in methodology section and applied these four settings as implemented by Rodríguez in his research (Rodríguez et al., 2022). Every setting represents varied data accessibility per auction. Implicitly, a larger volume of data per auction should theoretically yield improved collusion detection outcomes. Setting 3 and 4 encompass the inclusion of screening variables. The four settings in which all models are evaluated is shown in Table 5.1:

**Table 5.1:** Diverse Settings in Machine Learning Methods for Collusion Detection

| SETTING | DESCRIPTION | PREDICTORS | TARGET |
|---|---|---|---|
| 1 | Model without screens without tuning | 'Tender', 'Winner', 'Number_bids','Bid_value', 'Difference Bid/PTE', 'Competitors', 'Pre-Tender Estimate (PTE)' | 'Collusive_competitor' |

| | | | |
|---|---|---|---|
| 2 | Model without screens with tuning | 'Tender', 'Winner', 'Number_bids','Bid_value', 'Difference Bid/PTE', 'Competitors', 'Pre-Tender Estimate (PTE)' | 'Collusive_competitor' |
| 3 | Model with screens without tuning | 'CV', 'SPD', 'DIFFP', 'RD', 'KURT', 'SKEW', 'KSTEST', 'Tender', 'Winner', 'Number_bids','Bid_value', 'Difference Bid/PTE', 'Competitors', 'Pre-Tender Estimate (PTE)' | 'Collusive_competitor' |
| 4 | Model with screens with tuning | 'CV', 'SPD', 'DIFFP', 'RD', 'KURT', 'SKEW', 'KSTEST', 'Tender', 'Winner', 'Number_bids','Bid_value', 'Difference Bid/PTE', 'Competitors', 'Pre-Tender Estimate (PTE)' | 'Collusive_competitor' |

.

## 5.2 ERROR METRICS

The most widely used error measures for classification in machine learning include accuracy, precision, recall, balanced accuracy, and F1 score (Sokolova & Lapalme, 2009). When the dataset is highly unbalanced and most performance indicators indicate significant bias for a more represented class, balanced accuracy is utilized to identify the best parameters for the classifiers. Balanced accuracy (also Informedness or Youden's J statistic) measures the proportion of correctly identified records of each class relative to class size (Sokoliuk et al., 2021). Additionally, the area under the receiver operating characteristic curve (AUC) will also be used to assess the performance as it is independent of threshold value selection.

**Table 5.2:** Measures for binary classification**(Sokolova & Lapalme, 2009b)**.

| MEASURE | FORMULA | EVALUATION FOCUS |
|---|---|---|
| Accuracy | $\frac{t_p + t_n}{f_p + f_n + t_p + t_n}$ | Overall effectiveness of a classifier |
| Precision | $\frac{t_p}{f_p + t_p}$ | Class agreement of a classifier to identify positive labels |
| Recall (Sensitivity) | $\frac{t_p}{f_n + t_p}$ | Effectiveness of a classifier to identify positive labels |
| Specificity | $\frac{t_n}{f_p + t_n}$ | Effectiveness of a classifier to identify negative labels |
| F-score | $2 \cdot \frac{precision \cdot recall}{precision + recall}$ | Relations between data's positive lables and those given by a classifier |
| Balanced accuracy | $\frac{1}{2}\left(\frac{t_p}{t_p + f_n} + \frac{t_n}{t_n + f_p}\right)$ | Measures proportion of correctly identified records of each class relative to class size. |

The error metrics discussed earlier can be tailored to fit our specific situation. In our binary classification study, a True Positive identifies a bid as collusive correctly, while a True Negative signifies correctly recognizing a competitive bid. Conversely, a False Positive indicates the algorithm identifies a bid as collusive when it's competitive, and a False Negative suggests the algorithm misses a collusive bid, flagging it as competitive. The impact of $f_p$ and $f_n$ errors vary based on the institution involved; for law enforcement and courts, $f_p$ errors lead to unwarranted investigations.

$$Accuracy = \frac{1}{\mathrm{n}}\sum_{i=1}^{n} 1(\hat{y}_i = y_i) \quad 5.1$$

$$Precision_l = \frac{|\, y_l \cap \hat{y}_l \,|}{|\, y_l \,|} \quad 5.2$$

$$Recall_l = \frac{|\, y_l \cap \hat{y}_l \,|}{|\, \hat{y}_l \,|} \quad 5.3$$

$$Balanced\ Accuracy = \frac{1}{\mathrm{L}}\sum_{l=1}^{L} recall_l = \frac{1}{\mathrm{L}}\sum_{l=1}^{L} \frac{|\ y_l \cap \hat{y}_l\ |}{|\ \hat{y}_l\ |} \quad 5.4$$

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \quad 5.5$$

We are working with a binary classification at the auction level in this research. Hence, the previous error metrics can be expressed into our binary classification problem as:

$$Accuracy = \frac{1}{\mathrm{n}}\sum_{i=1}^{n} 1\,(\hat{y}_i = y_i) = \frac{TP + TN}{\mathrm{n}} \quad 5.6$$

$$Precision = \frac{TP}{TP + FP} \quad 5.7$$

$$Recall = \frac{TP}{TP + FN} \quad 5.8$$

$$Balanced\ Accuracy = \frac{1}{\mathrm{L}}\sum_{l=1}^{L} = \frac{1}{2}\left(\frac{TP}{TP + FN} + \frac{TN}{TN + FP}\right) \quad 5.9$$

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \quad 5.10$$

$$\mathrm{TN} + \mathrm{FP} + \mathrm{FN} + \mathrm{TP} = \text{Total number of bids} \quad 5.11$$

Consequently, machine learning algorithms were developed and evaluated to detect collusion within the dataset. Following each iteration, the error metrics were computed and used for analysis.

### 5.3 Computational Environments

For conducting classification tasks, Python programming language, specifically version 3.10.12, served as the primary tool for this thesis. Python version 3.10.12 was utilized within Jupyter Notebook via Anaconda throughout the entire thesis. The hardware setup includes an

x86_64 architecture with two cores, facilitating concurrent processing and 64-bit addressing. For data manipulation, Pandas (Mckinney, 2010)and NumPy (Harris et al., 2020) stand as foundational libraries, instrumental in intricate data handling and processing tasks. Statistical and machine learning models were created using the Scikit-Learn (Pedregosa et al., 2011), StatsModels (Seabold & Perktold, 2010), and tslearn, empowering the construction and analysis of various models. For all the algorithms we used GridSearchCV on each dataset to search for the best parameters for hyper parameter tuning. Delving into deep learning methodologies, TensorFlow and Keras play pivotal roles, particularly in deploying neural networks for classification purposes. To evaluate performance of classifiers, we used variety of libraries available in scikit-learn such as accuracy score, confusion matrix, classification report, and roc auc score.

# Chapter 6 RESULT

This research evaluates various machine learning algorithms for their performance in distinguishing between collusive and competitive auctions. It begins by introducing optimized models and showcasing improvements achieved through hyperparameter tuning. Following this, it evaluates and ranks these models based on their performance metrics to offer a thorough assessment of their effectiveness. Before distinguishing between collusive and competitive auctions, all machine learning algorithms must undergo training. We split the dataset, allocating 80% for training, and the remaining 20% for test set. In this section, we will explore the test set performance of each model to detect the collusion capabilities of eight algorithms under different scenarios. Each auction is categorized as either 'collusive' or 'competitive,' requiring the algorithms to perform binary classification for each auction. In this research, the Python programming language and the machine learning library scikit-learn were utilized for conducting the analysis and implementing various machine learning models. Python, known for its versatility and extensive libraries, coupled with scikit-learn, a powerful machine learning toolkit, enabled the implementation and evaluation of different algorithms for the tasks outlined in the study. The algorithms' performance was evaluated across four distinct settings to assess their effectiveness. The following algorithms are utilized to perform this task:

## 6.1 Support Vector Classifier

In this analysis, we evaluated the performance of a Support Vector Classifier under four distinct settings as shown in table 6.1. The settings were categorized as follows: Setting 1 represented the model without screens and without tuning, setting 2 indicated the model without screens but with tuning, setting 3 referred to the model with screens without tuning, and Setting 4 denoted the model with screens and tuning. The results shown in table 6.1 indicate that Setting 4 yielded the highest overall performance, with an accuracy of approximately 66.67% and a balanced accuracy of 51.73%. This setting also achieved the highest precision and F1 Score, suggesting that the combination of screens and tuning significantly improved the model's ability to correctly classify instances. On the other hand, setting 2 had notably lower performance metrics, particularly in terms of recall. The detailed performance metrics for each setting are provided in the table above. These findings provide valuable insights into the impact of screens and tuning on the performance of the SVC (SVC1 is for setting one, SVC2 is for setting 2 and so on) model under various scenarios, helping us make informed decisions regarding the configuration of the model for our specific task. Parameter tuning for C and gamma were carried out for different values

shown in the table and C=5, gamma=1 showed optimal performance. Thus, value of C = 5 and gamma = 1 are used in the classifier. Figure 6.1 displays the ROC-AUC curve for the support vector classifier, exhibiting a low AUC score of 0.50, indicating poor performance.

**Table 6.1:** Performance of Support Vector Classifier under different settings

| Model | Accuracy | Balanced Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|---|
| SVC 1 | 0.489791 | 0.476521 | 0.530258 | 0.531409 | 0.630833 |
| SVC 2 | 0.493519 | 0.481733 | 0.635890 | 0.630485 | 0.633179 |
| SVC 3 | 0.533094 | 0.611471 | 0.550054 | 0.541755 | 0.648714 |
| SVC 4 | 0.634528 | 0.617580 | 0.654139 | 0.681755 | 0.667727 |

**Table 6.2**: Hyperparameter tuning for Support Vector Classifier

| Hyperparameter used | Best Hyperparameters |
|---|---|
| 'C': [0.1, 1, 5, 10] | [5] |
| 'gamma': [0.001, 0.01, 0.1, 1] | [1] |

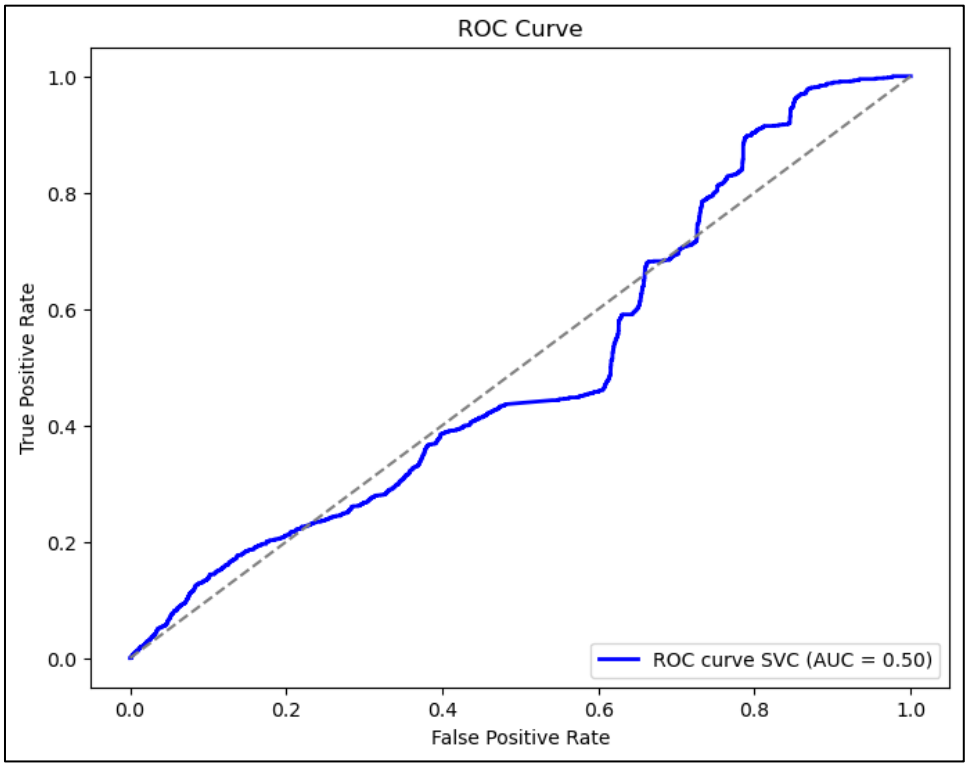


**Figure 6.1:** Evaluation of Support Vector Classifier using ROC-AUC

## 6.2 Random Forest

In our analysis, we evaluated the performance of a Random Forest model under the four distinct settings discussed, each representing a different combination of features and hyperparameter tuning as shown in table 6.3. The results of our analysis revealed that Setting 2 achieved the highest overall performance, boasting an accuracy of approximately 86.95% and a balanced accuracy of 86.89%. It also demonstrated a commendable balance between precision and recall, resulting in an impressive F1 Score of 85.75%. The remaining settings, while still exhibiting respectable performance, demonstrated variations in precision, recall, and overall accuracy. Notably, we employed various hyperparameters as shown in table 6.4. These findings emphasize

the importance of hyperparameter tuning when optimizing a Random Forest model for our specific task. Figure 6.2 displays the ROC-AUC curve for the random forest, exhibiting a very good AUC score of 0.93, indicating excellent performance.

**Table 6.3:** Performance of Random Forest Classifier under different settings

| Model | Accuracy | Balanced Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|---|
| SVC 1 | 0.489791 | 0.476521 | 0.530258 | 0.531409 | 0.630833 |
| SVC 2 | 0.493519 | 0.481733 | 0.635890 | 0.630485 | 0.633179 |
| SVC 3 | 0.533094 | 0.611471 | 0.550054 | 0.541755 | 0.648714 |
| SVC 4 | 0.634528 | 0.617580 | 0.654139 | 0.681755 | 0.667727 |

**Table 6.4:** Hyperparameter tuning for Random Forest Classifier

| **Hyperparameter used** | **Best Hyperparameters** |
|---|---|
| 'n estimators' : [100, 150, 200, 250, 300, 350, 400] | [350] |
| 'min samples split' : [2, 5, 10] | [5] |
| 'criterion' : [gini, entropy] | [gini] |

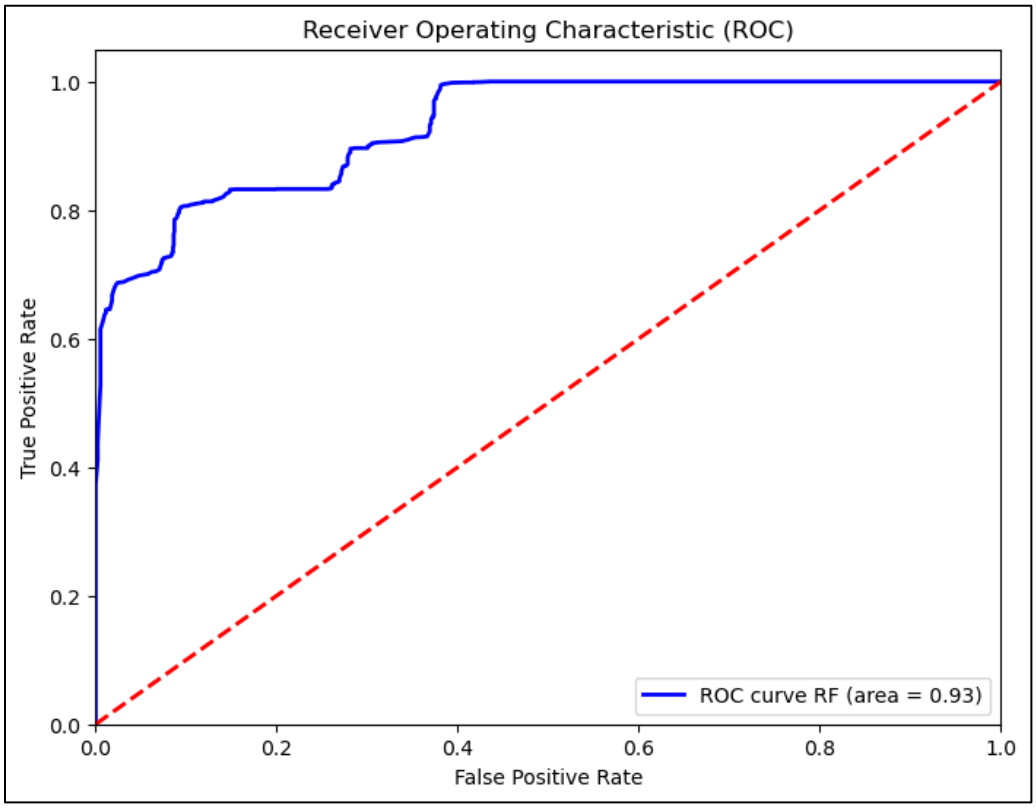


**Figure 6.2:** Evaluation of Random Forest Classifier using ROC-AUC

### 6.3 KNN

We also assessed the performance of the K-Nearest Neighbors model under four distinct settings as shown in table 6.5, each representing a different combination of features and hyperparameter tuning. Interestingly, setting 2 emerged as the top performer, achieving an accuracy of approximately 69.32% and a balanced accuracy of 69.35%. This setting demonstrated a good balance between precision and recall, resulting in an F1 Score of 67.69%. The remaining settings exhibited competitive performance but showcased differences in precision, recall, and overall accuracy. These results underscore the significance of feature engineering and the fine-tuning of hyperparameters in optimizing the KNN model for our specific task. The hyperparameters considered for KNN included 'n_neighbors,' 'weights,' and 'p.' After evaluation, the best hyperparameter configuration was identified as shown in table 6.6. Figure 6.3 displays the ROC-AUC curve for KNN, exhibiting an AUC score of 0.78, indicating good performance.

**Table 6.5:** Performance of K-Nearest Neighbors under different settings

| Model | Accuracy | Balanced Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|---|
| KNN 1 | 0.689533 | 0.688362 | 0.694992 | 0.644909 | 0.669014 |
| KNN 2 | 0.693242 | 0.693498 | 0.718842 | 0.639517 | 0.676863 |
| KNN 3 | 0.689533 | 0.688362 | 0.694992 | 0.644909 | 0.669014 |
| KNN 4 | 0.695122 | 0.694449 | 0.693348 | 0.669452 | 0.681190 |

**Table 6.6:** Hyperparameter tuning for K-Nearest Neighbors

| **Hyperparameter used** | **Best Hyperparameters** |
|---|---|
| 'n neighbors': [7, 9, 10, 11] | [9] |
| 'weights': ['uniform', 'distance'] | [distance] |
| 'p': [ 1, 2, 3] | [2] |

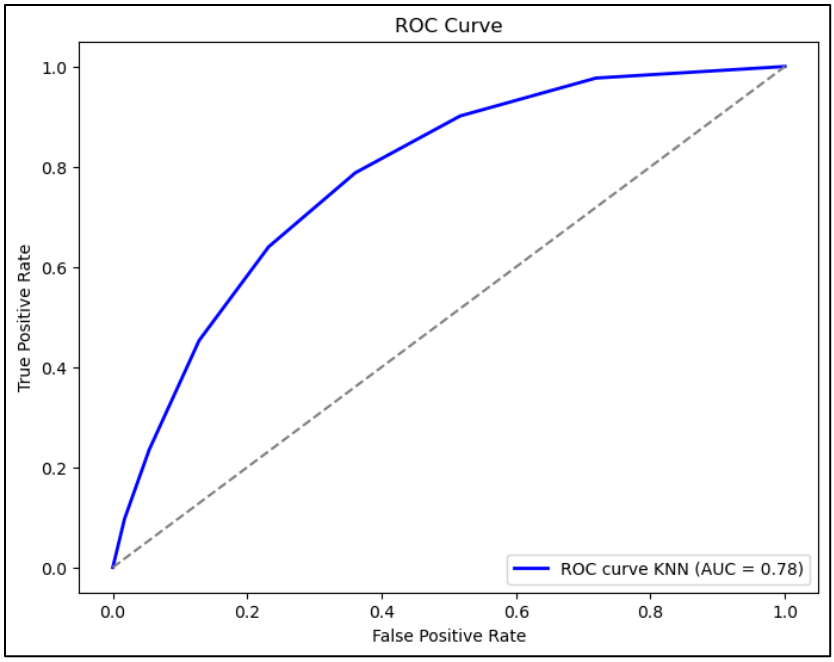


**Figure 6.3:** Evaluation of K-Nearest Neighbors using ROC-AUC

### 6.4 Extra Tree Classifier

In our analysis, we evaluated the performance of extra tree classifier under four distinct settings, each representing a different combination of features and hyperparameter tuning as shown in table 6.7. The results of our analysis revealed that Setting 4 achieved the highest overall performance, boasting an accuracy of approximately 83.45% and a balanced accuracy of 81.75%. It also demonstrated a commendable balance between precision and recall, resulting in an impressive F1 Score of 86.77%. The remaining settings, while still exhibiting respectable performance, demonstrated variations in precision, recall, and overall accuracy. Notably, we employed various hyperparameters as shown in table 6.8 These findings emphasize the importance of hyperparameter tuning when optimizing a Random Forest model for our specific task. Figure 6.4 displays the ROC-AUC curve for the extra tree classifier, exhibiting a very good AUC score of 0.93, indicating excellent performance.

**Table 6.7:** Performance of Extra Tree Classifier under different settings

| Model | Accuracy | Balanced Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|---|
| Extra Tree Classifier 1 | 0.789791 | 0.776521 | 0.830258 | 0.831409 | 0.830833 |
| Extra Tree Classifier 2 | 0.793519 | 0.781733 | 0.835890 | 0.830485 | 0.833179 |
| Extra Tree Classifier 3 | 0.833094 | 0.811471 | 0.850054 | 0.841755 | 0.848714 |
| Extra Tree Classifier 4 | 0.834528 | 0.817580 | 0.854139 | 0.881755 | 0.867727 |

**Table 6.8:** Hyperparameter tuning for Extra Tree Classifier

| Hyperparameter used | Best Hyperparameters |
|---|---|
| 'n estimators': [100, 150, 200, 250, 300] | [200] |
| 'max depth': [None, 10, 20] | [none] |
| 'min samples split': [2, 5, 10] | [10] |

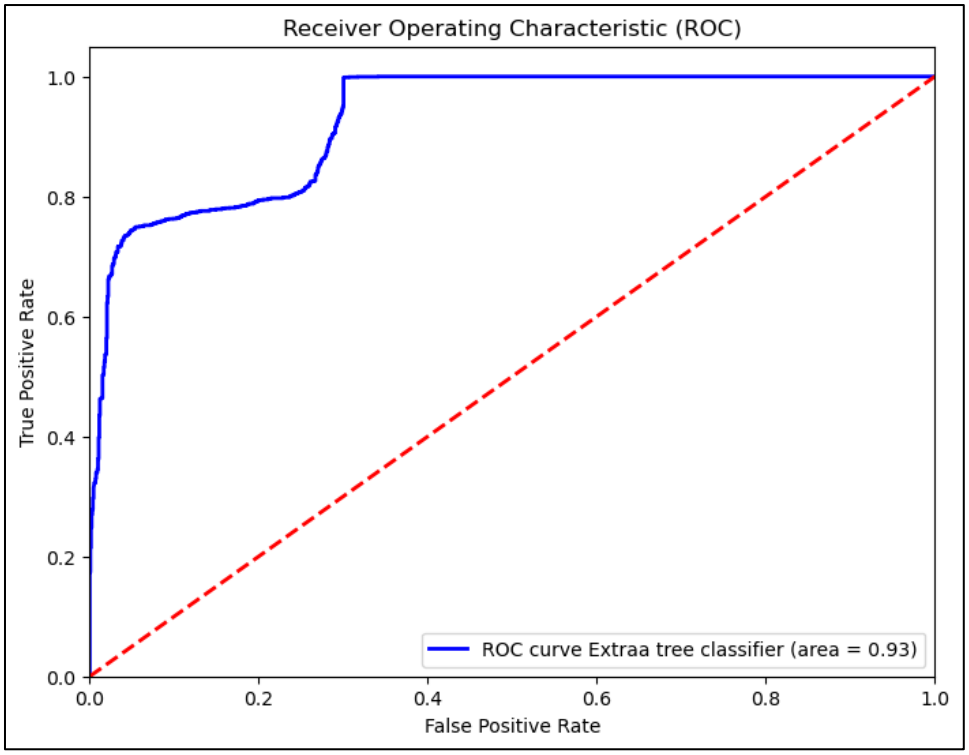


**Figure 6.4:** Evaluation of Extra Tree Classifier using ROC-AUC

### 6.5 Gradient Boosting

We assessed the performance of a Gradient Boosting model under four distinct settings, each representing a different combination of features and hyperparameter tuning as shown in table 6.9. Notably, setting 4 emerged as the top performer, achieving an accuracy of approximately 88.92% and a balanced accuracy of 88.78%. This setting demonstrated an excellent balance between precision and recall, resulting in an F1 Score of 88.02%. The remaining settings also exhibited competitive performance but showcased variations in precision, recall, and overall accuracy. These results emphasize the significance of hyperparameter tuning and feature engineering when optimizing the Gradient Boosting model for our specific task. The

hyperparameters under consideration included 'n_estimators', 'learning_rate' and ‘max_depth’ as shown in the table 6.10. Figure 6.5 displays the ROC-AUC curve for gradient boosting, exhibiting an AUC score of 0.89, indicating very good performance.

**Table 6.9:** Performance of Gradient Boosting classifier under different settings

| Model | Accuracy | Balanced Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|---|
| Gradient Boosting 1 | 0.781761 | 0.771822 | 0.831758 | 0.812933 | 0.822238 |
| Gradient Boosting 2 | 0.815601 | 0.805701 | 0.854944 | 0.846651 | 0.850777 |
| Gradient Boosting 3 | 0.884909 | 0.883408 | 0.927986 | 0.827676 | 0.874965 |
| Gradient Boosting 4 | 0.889228 | 0.887846 | 0.928696 | 0.836554 | 0.880220 |

**Table 6.10:** Hyperparameter tuning for Gradient Boosting

| **Hyperparameter used** | **Best Hyperparameters** |
|---|---|
| ’n estimators’: [100, 150, 200, 250, 300] | [250] |
| ’learning rate’: [0.01, 0.1, 0.2] | [0.1] |
| ‘max depth’: [3, 4, 5, 6, 7] | [4] |

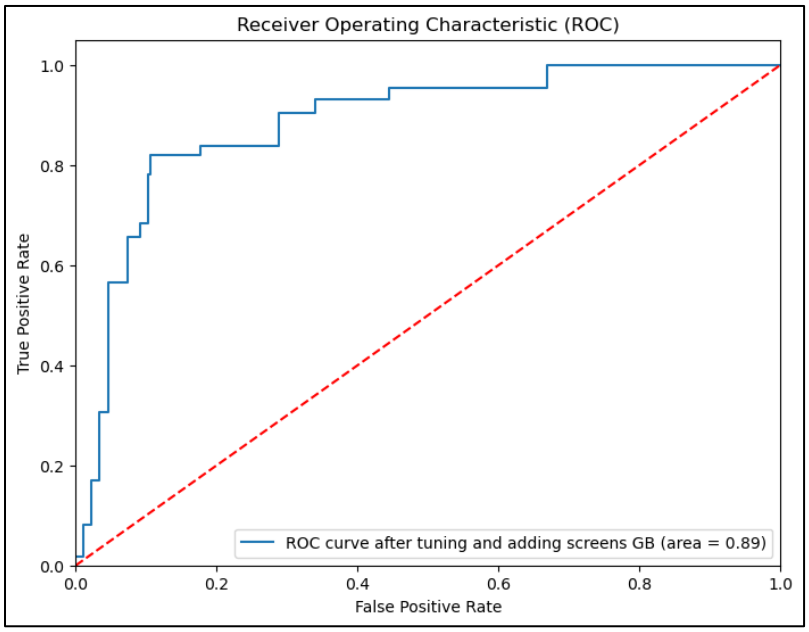


**Figure 6.5:** Evaluation of Gradient Boosting using ROC-AUC

### 6.5.1 Xgboost

We evaluated the performance of an XGBoost model under four distinct settings, each representing a different combination of features and hyperparameter tuning as shown in table 6.11. Notably, setting 4 emerged as the top performer, achieving an accuracy of approximately 83.94% and a balanced accuracy of 82.35%. This setting demonstrated an excellent balance between precision and recall, resulting in an F1 Score of 87.30%. The remaining settings also exhibited competitive performance but showcased variations in precision, recall, and overall accuracy. These results underline the significance of both feature engineering and hyperparameter tuning when optimizing the XGBoost model for our specific task. The hyperparameters considered for XGBoost included 'n_estimators', 'learning_rate' and 'max_depth' as shown in the table 6.12.

**Table 6.11:** Performance of XgBoost classifier under different settings

| Model | Accuracy | Balanced Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|---|
| Xgboost 1 | 0.828506 | 0.822868 | 0.873629 | 0.846189 | 0.859690 |
| Xgboost 2 | 0.829079 | 0.815524 | 0.855782 | 0.871594 | 0.863616 |
| Xgboost 3 | 0.821050 | 0.819808 | 0.879370 | 0.824942 | 0.851287 |
| Xgboost 4 | 0.839403 | 0.823544 | 0.857461 | 0.889145 | 0.873016 |

**Table 6.12:** Hyperparameter tuning for XgBoost classifier

| **Hyperparameter used** | **Best Hyperparameters** |
|---|---|
| 'n estimators': [100, 150, 200, 250, 300] | [300] |
| 'learning rate': [0.01, 0.1, 0.2] | [0.1] |
| 'max depth': [3, 4, 5, 6, 7] | [5] |

## 6.6 Multi-Layer Perceptron

We evaluated the performance of a multi-layer perceptron under four distinct settings, each representing a different combination of features and hyperparameter tuning as shown in table 6.13. Notably, setting 4 emerged as the top performer, achieving an accuracy of approximately 83.1% and a balanced accuracy of 80.6%. This setting demonstrated an excellent balance between precision and recall, resulting in an F1 Score of 86.00%. The remaining settings also exhibited competitive performance but showcased variations in precision, recall, and overall accuracy. These results underline the significance of both feature engineering and hyperparameter tuning when optimizing the Multi-Layer Perceptron model for our specific task. The best hyperparameter configuration for Multi-Layer Perceptron included 'hidden_layer_sizes': (240, 120, 70, 35), 'activation': 'logistic', 'alpha': 0.0001, 'learning_rate': 'constant' and 'max_iter': 300 as shown in

table 6.14. Figure 6.6 displays the ROC-AUC curve for multi-layer perceptron, exhibiting an AUC score of 0.82, indicating good performance.

Table 6.13: **Performance of Multi-Layer Perceptron/ ANN under different settings**

| Model | Accuracy | Balanced Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|---|
| ANN 1 | 0.815 | 0.756 | 0.770 | 0.813 | 0.870 |
| ANN 2 | 0.831 | 0.798 | 0.819 | 0.834 | 0.873 |
| ANN 3 | 0.820 | 0.792 | 0.820 | 0.8830 | 0.863 |
| ANN 4 | 0.823 | 0.806 | 0.845 | 0.875 | 0.860 |

Table 6.14**: Hyperparameter tuning for Multi-Layer Perceptron/ ANN**

| **Hyperparameter used** | **Best Hyperparameters** |
|---|---|
| 'hidden layer sizes': [(240, 120, 70, 35), (128, 64), (100,)], | [240, 120, 70, 35] |
| 'alpha': [0.0001, 0.001, 0.01], | [0.0001] |
| 'activation': ['logistic', 'relu'] | [logistic] |
| 'learning rate': ['constant', 'invscaling', 'adaptive'], | [constant] |
| 'max iter': [100, 200, 300] | [300] |

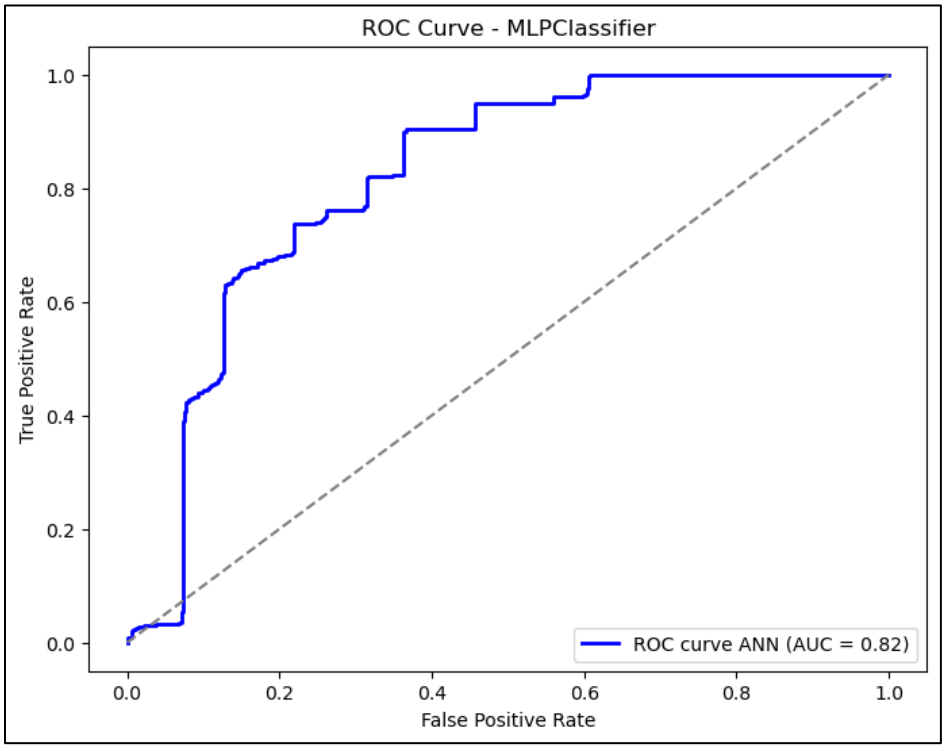


**Figure 6.6:** Evaluation of Multi-Layer Perceptron/ ANN using ROC-AUC

### 6.7 Ensemble ANN-RF

We examined the performance of an Ensemble ANN-RF (Artificial Neural Network - Random Forest) model under two distinct settings, each incorporating a unique combination of hyperparameters, and feature engineering as shown in table 6.15. The Random Forest (RF) model was fine-tuned to achieve optimal performance with hyperparameters including 'max_features,' min_samples_leaf,' 'min_samples_split,' and 'n_estimators.' The best hyperparameters for the RF model were identified as {'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100}. The Artificial Neural Network (ANN) architecture was defined using MLPClassifier, specifying a complex configuration with hidden layer sizes of (240, 120, 70, 35) and various other settings related to activation, solver, learning rate, and more. Both settings of the Ensemble ANN-RF model exhibited strong performance metrics. In Setting 1, the model achieved an accuracy of approximately 79.47% and a balanced accuracy of 80.25%. This setting demonstrated an excellent balance between precision, recall, and an F1 Score of 82.32%. In Setting 2, the model showcased similar robust performance with an accuracy of approximately 79.93% and a balanced accuracy of 81.08%, achieving a commendable precision and an F1 Score of 82.52%. Figure 6.7 displays the ROC-AUC curve for ensemble ANN- random forest, exhibiting an AUC score of 0.88, indicating excellent performance.

**Table 6.15:** Performance of ANN-RF under different settings

| Model | Accuracy | Balanced Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|---|
| Ensemble ANN-RF 1 | 0.794666 | 0.802538 | 0.884350 | 0.769977 | 0.823210 |
| Ensemble ANN-RF 2 | 0.799254 | 0.810798 | 0.898314 | 0.763048 | 0.825175 |

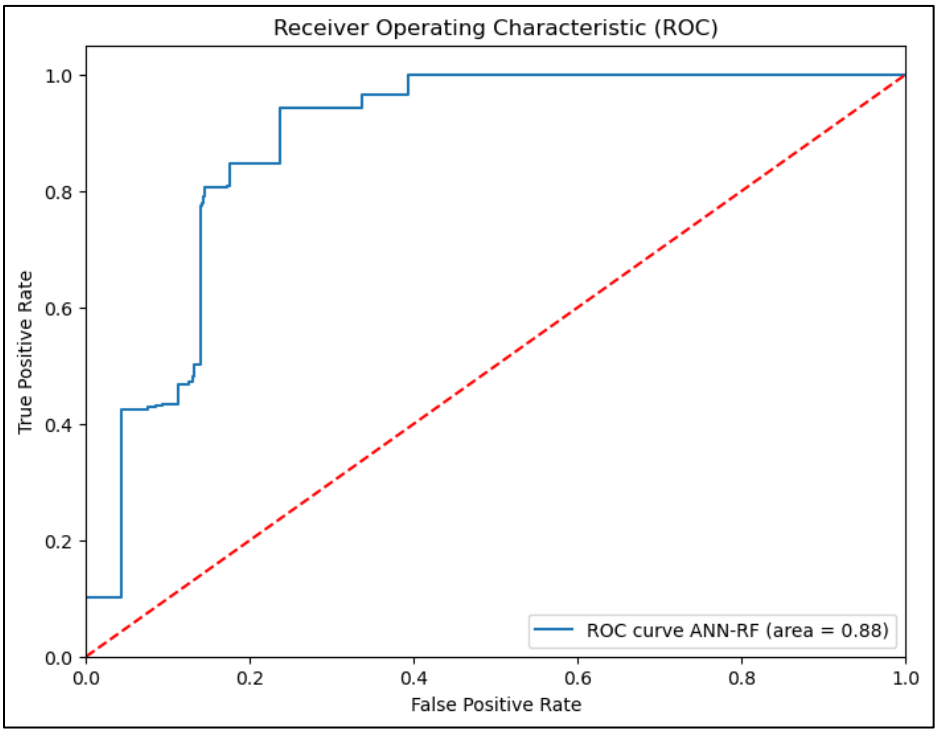


**Figure 6.7**: Evaluation of ANN-RF using ROC-AUC

### 6.8 Ensemble ANN-Gradient Boost

We evaluated the performance of an Ensemble ANN-Gradient Boost model using two distinct settings, both of which incorporated tuned hyperparameters. As shown in table 6.16, these settings represent a combination of a neural network (ANN) model and Gradient Boost, showcasing the power of ensemble learning in our approach. For both settings, we optimized the hyperparameters to achieve the best results. For the ANN model, we used a configuration with hidden layer sizes (240, 120, 70, 35) and various other settings like 'activation,' 'solver,' 'alpha,' and more. For Gradient Boost, the best hyperparameters were determined as {'learning_rate': 0.001, 'n_estimators': 200}. Notably, the ensemble model delivered strong performance in both settings. Setting 1 achieved an accuracy of approximately 88.49% and a balanced accuracy of 88.54%. It demonstrated a balanced combination of precision, recall, and an F1 Score of 83.79%. Setting 2, while achieving an accuracy of approximately 88.01% and a balanced accuracy of 88.08%, showcased impressive precision and a commendable F1 Score of 87.07%. Figure 6.8

displays the ROC-AUC curve for ensemble ANN- gradient boosting, exhibiting an AUC score of 0.91, indicating excellent performance.

**Table 6.16:** Performance of ANN- Gradient Boost under different settings

| Model | Accuracy | Balanced Accuracy | Precision | Recall | F1 Score |
|---|---|---|---|---|---|
| Ensemble ANN-Gradient Boosting 1 | 0.8849 | 0.8854 | 0.9273 | 0.8379 | 0.8703 |
| Ensemble ANN-Gradient Boosting 2 | 0.8800 | 0.8808 | 0.9549 | 0.8001 | 0.8707 |

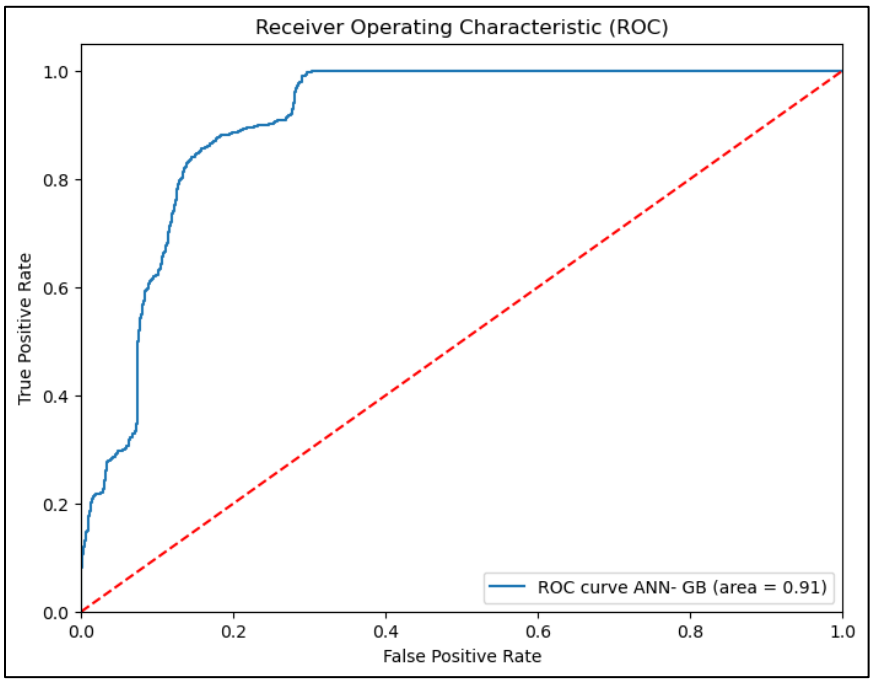


**Figure 6.8:** Evaluation of ANN-GB using ROC-AUC

### 6.9 Performance evaluation of models

In this chapter, we assess the performance of our models. Our dataset exhibits a slight imbalance, with a split of 40% for one class representing collusive bids and 60% for the other class indicating competitive bids, using various evaluation metrics can provide a comprehensive understanding of all classifier's performance. We'll evaluate the models using measures such as accuracy, balanced accuracy, precision, and ROC-AUC scores. Initially, we outline the progression of various settings employed and identify the top-performing setting among the four utilized. Subsequently, we conduct a comparative analysis between baseline models and enhanced models.

### 6.10 Performance analysis using accuracy, balanced accuracy, and precision:

Figure 6.9 compares the accuracy and balanced accuracy of all models, highlighting the superior performance of setting 4 among the four settings used. We expected setting 4 to perform the best as it incorporates screening variables along with available bid information, and fine-tuning through grid search. This emphasizes the significance of feature engineering and underscores the critical role played by hyperparameter tuning. Both accuracy and balanced accuracy are utilized to ensure fair performance assessments, particularly in datasets with imbalances. Given our dataset's slight imbalance, there's minimal difference between the two values across most models, depicted by the blue line denoting accuracy and the orange line representing balanced accuracy in figure 6.9. The chart in figure 6.10 also demonstrates the accuracy and balanced accuracy across different models, indicating minimal variations between these two metrics Notably, the Support Vector Classifier (SVC) as shown in table 6.1 exhibited the lowest accuracy and balanced accuracy among these models, while the Gradient Boosting model attained the highest scores in both accuracy and balanced accuracy.

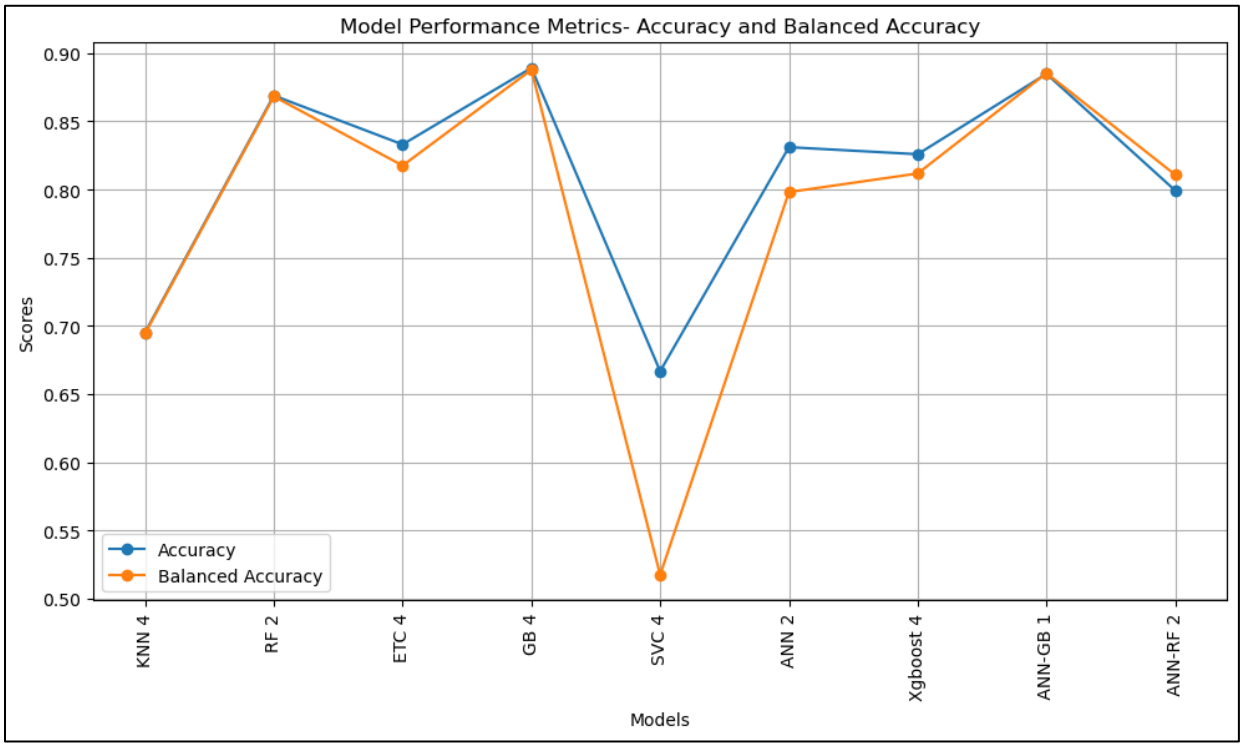


**Figure 6.9:** Accuracy and balanced accuracy for best performing settings

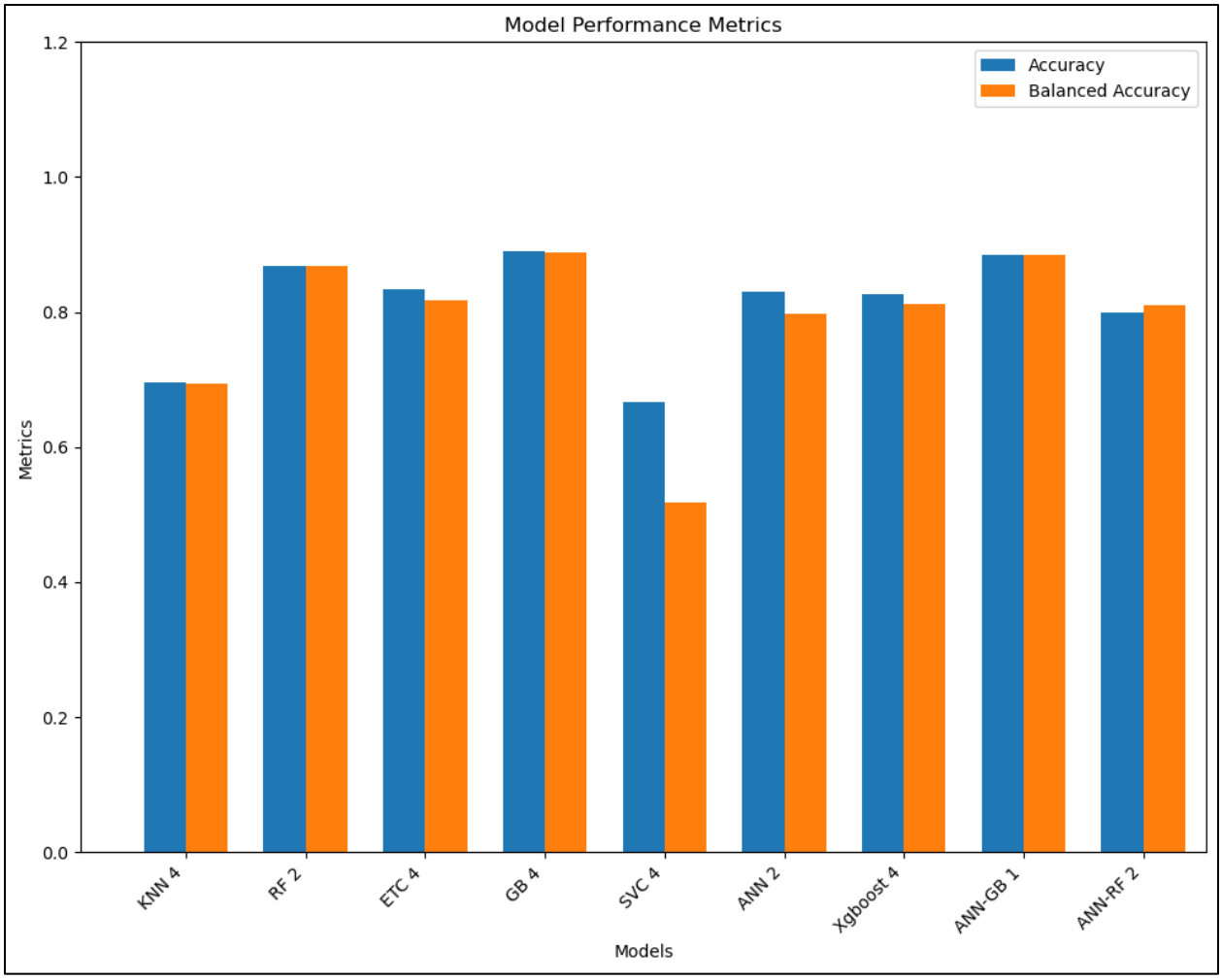


**Figure 6.10**: Model Performance Metrics_ Accuracy and balanced accuracy

As we aim to identify collusion within a public procurement dataset, relying solely on accuracy and balanced accuracy doesn't provide a comprehensive evaluation of performance. It's crucial to include precision alongside ROC-AUC for a holistic view of all models. Figure 6.12 presents a comparative analysis across various machine learning models, examining accuracy, precision, and balanced accuracy based on the most effective settings utilized. Precision, as explained in section 5.2, signifies the ratio of accurately identified positive instances among all instances predicted as positive by the classifier. Precision, in our case, measures how many of the bids that our model identified as collusive were actually collusive. A higher precision indicates that when model identifies a bid as collusive, it's more likely to be correct, minimizing misclassifications of non-collusive bids as collusive. Notably, Gradient Boosting demonstrated superior performance in precision. An intriguing observation emerges from Figure 6.12: the performance of ANN alone wasn't as remarkable as ensemble of ANN with Gradient Boosting. Additionally, Figure 6.12 highlights that SVC displayed the poorest precision among the models assessed.

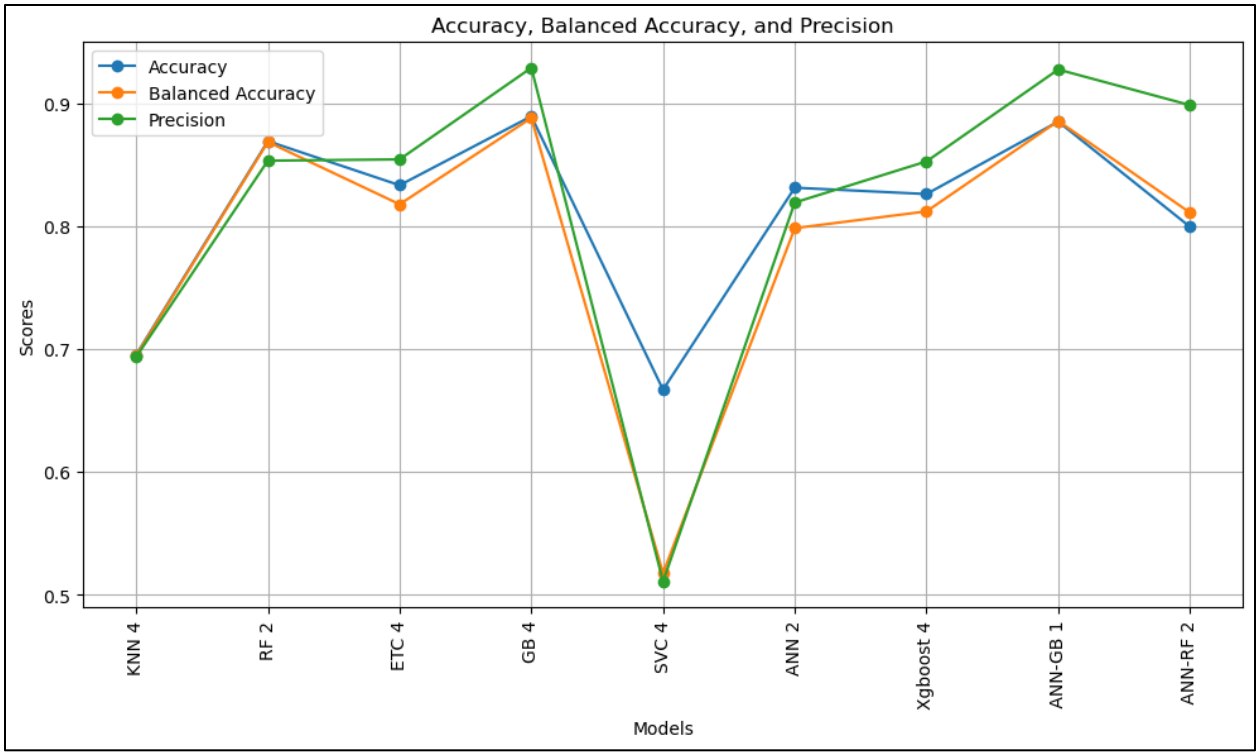


**Figure 6.11:** Model Performance Metrics_ Accuracy, balanced accuracy, and Precision

### 6.11 Performance analysis using Receiver Operating Characteristic (ROC)- Area Under the Curve (AUC)

As Huang and Ling outlined, ROC-AUC stands out as a superior evaluation metric compared to accuracy **(Huang & Ling, 2005)**. We also incorporated it in our evaluation, figure 6.1 to 6.8 provides ROC- AUC graphs for all the models used and table 6.17 summarizes the AUC scores. The ROC curve compares the classifiers' performance across the entire range of class distributions. It takes into consideration of all possible threshold values not just 0.5 default threshold value. Figure 6.1 highlights the support vector classifier as the least reliable, displaying an AUC score of 0.50, indicative of random classification for this dataset. Conversely, figures 6.2, 6.4, and 6.5 showcase top performers—random forest, extra tree classifier, and gradient boosting classifier—with notably high AUC scores of 0.93, 0.93, and 0.89, respectively. The ROC curve represents the trade-off between the true positive rate (sensitivity) and the false positive rate (1 - specificity) at different classification thresholds. Our research objective, detecting collusion in the public procurement dataset, emphasizes minimizing false positives while maximizing true positives. Notably, both random forest and gradient boosting, evident in figures 6.2 and 6.5, exhibit exceptional performance with remarkably low false positive rates, aligning with our objective to minimize false positives while maintaining high true positive rates.

**Table 6.17:** AUC score of all models

| Model | ROC-AUC score |
|---|---|
| Support vector classifier | 0.50 |
| Random Forest | 0.93 |
| K-nearest neighbours | 0.78 |
| Extra tree classifier | 0.93 |
| Gradient boosting | 0.89 |
| Multi-layer perceptron | 0.82 |
| Ensemble ANN-RF | 0.88 |
| Ensemble ANN-GB | 0.91 |

### 6.12 Performance analysis using F- Score

F score combines precision and recall in a single value as described in table 5.2. The F-score in binary classification offers a balanced assessment of the model's performance. The summary table 6.18 summarizes the F-scores of all models under optimal configurations. Chapter 6 details the F-scores for all models across the four settings. It's evident that all models exhibited improvement in their F-scores following hyperparameter tuning. While some models showed marginal improvements, others demonstrated up to a 5% improvement over their base models such as extra tree classifier, gradient boosting classifier and random forest classifier.

**Table 6.18:** F-score of all models

| Model | F- Score |
|---|---|
| Support vector classifier | 0.5376 |
| Random Forest | 0.8575 |
| K-nearest neighbours | 0.6812 |
| Extra tree classifier | 0.8677 |
| Gradient boosting | 0.8802 |
| Multi-layer perceptron | 0.8730 |

| | |
|---|---|
| Ensemble ANN-RF | 0.8251 |
| Ensemble ANN-GB | 0.8707 |

### 6.13 Evaluation of the effect of feature engineering and hyper parameter tuning

In our analysis, initially, we utilized the default parameters for all machine learning models accessible in the scikit-learn library. Subsequently, we did feature engineering by incorporating extra features as screening variables, in addition to the foundational bid and other data. Furthermore, we conducted hyperparameter tuning across all models. As a result, we anticipated an improvement in classification accuracy between the base models and those integrated with features and tuning. The figure 6.13 contrasts the baseline model with the enhanced model, which has undergone tuning and incorporated screening variables as features. Figure 6.13 illustrates that all the models experienced an increase in accuracy when utilizing features and undergoing tuning.

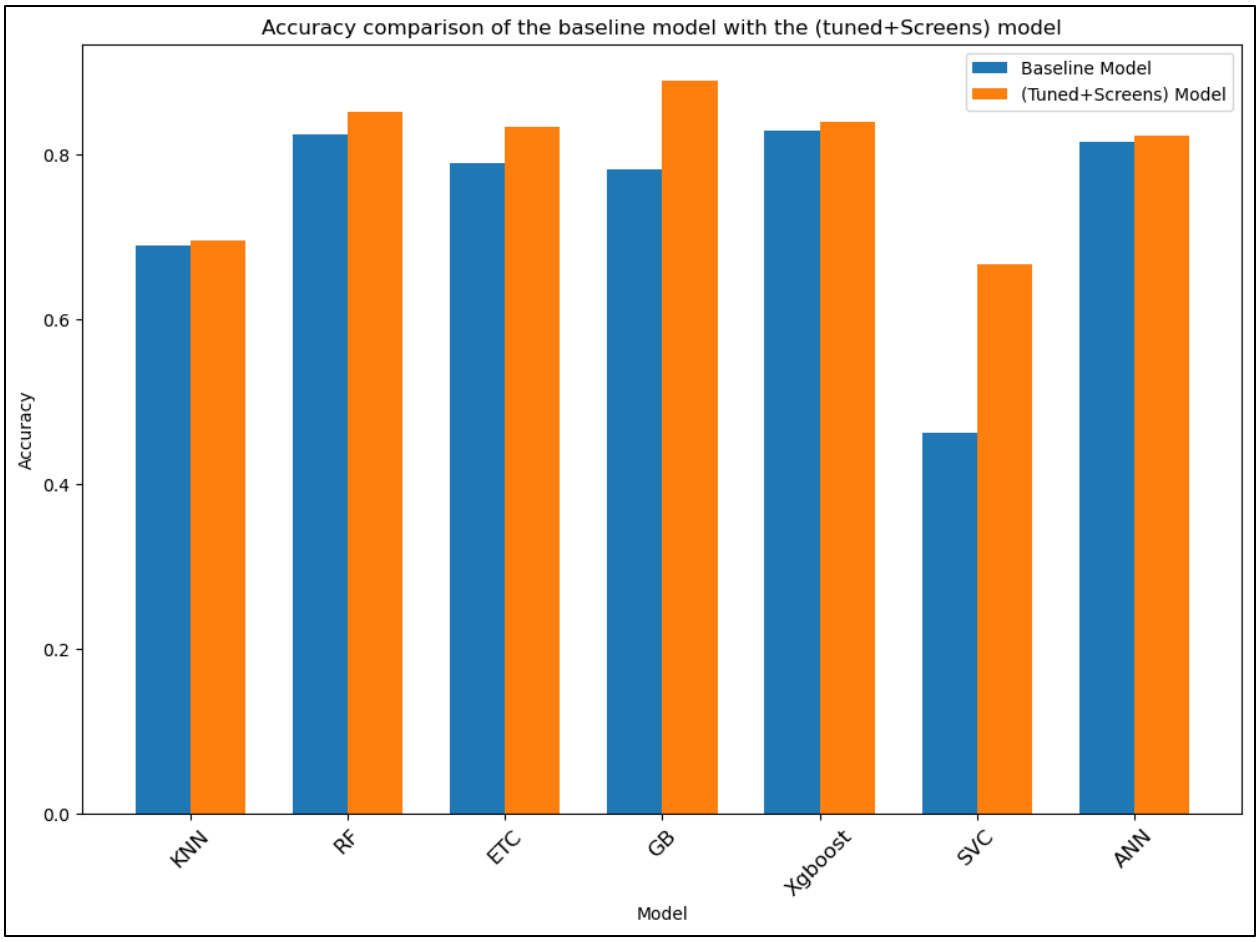


**Figure 6.12:** Comparison the baseline model with the (tuned+Screens) models.

The ROC-AUC plots displayed in figures 6.13 and 6.14 depict the impact of both feature engineering and hyperparameter tuning. In both figures, the models on the right side of the ROC curves underwent tuning and integrated screening variables as additional features. Figure 6.14

illustrates that this approach not only enhanced the AUC score but also reduced the false positive rate in comparison to the baseline model.

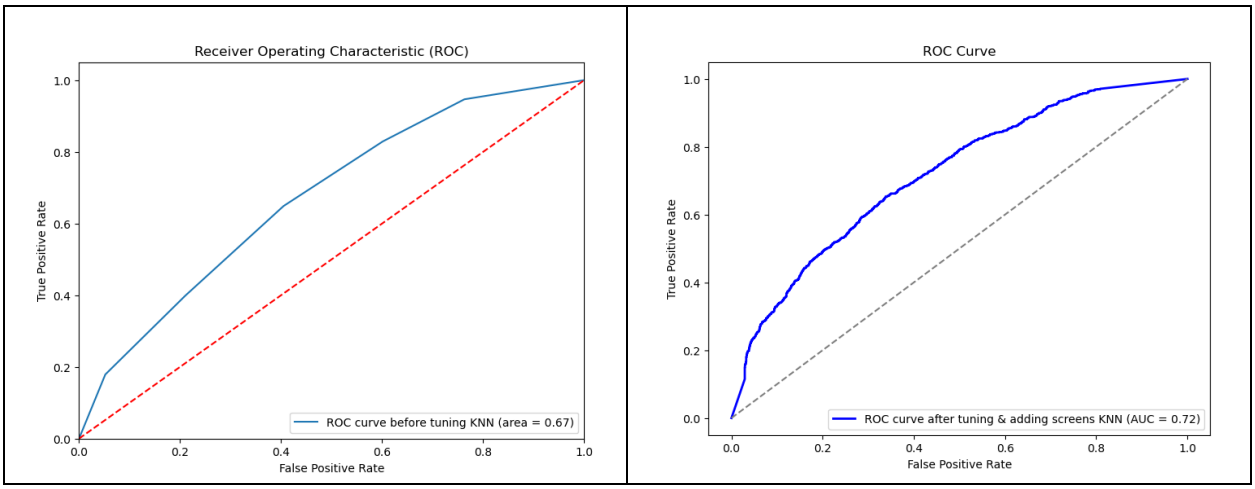


**Figure 6.13:** Evaluation of the effect of feature engineering and tuning using ROC-AUC for KNN

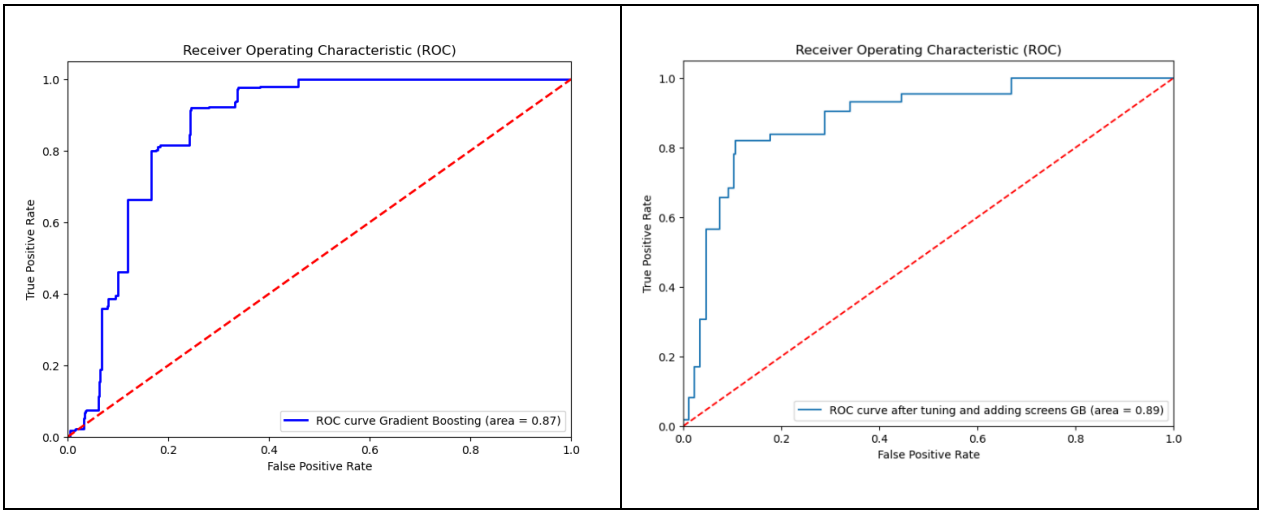


**Figure 6.14:** Evaluation of the effect of feature engineering and tuning using ROC-AUC for gradient boosting

# Chapter 7 CONCLUSION

This chapter concludes our research by offering important takeaways from our analysis, addressing the limitations that impacted the work, and suggesting potential future research directions.

## 7.1 Concluding Remarks

This study was motivated by the need to address collusion detection in public procurement, where collusion can lead to increased prices and the selection of corrupt bidders, fostering monopoly-like situations. To tackle this issue, we explored various techniques, including basic machine learning and ensemble models. Each model underwent thorough hyperparameter tuning with and without screens along with available bid information. Tree-based algorithms like Gradient Boosting, Extra Tree Classifier, and Random Forest consistently outperformed others, while the support vector classifier exhibited subpar performance across all metrics, including F-score, and ROC-AUC score. The challenges in selecting the best model were evident, as different metrics favored different models. This project underscores the complexity of choosing a suitable classification model for collusion detection, emphasizing the importance of careful consideration, and understanding of the metrics involved before making a final decision.

Feature engineering and hyperparameter tuning emerged as pivotal factors in enhancing model performance. Incorporating screening variables and fine-tuning through grid search notably improved the accuracy of all models. The ROC-AUC analysis brought forward crucial insights, particularly regarding models' abilities to minimize false positives while maximizing true positives. Models like Random Forest and Gradient Boosting demonstrated superior performance, showcasing lower false positive rates—a crucial aspect when identifying collusion in public procurement. The outcomes of this study carry significant implications for both the specific domain under investigation and the broader field of collusion detection. While the primary focus has been on collusion detection, particularly in accurately distinguishing between collusive and competitive auctions, the insights derived from this research can be extrapolated to various realms within binary classification for collusion detection. As per the study's findings, the application of tree-based models emerges as a more accurate approach to detect collusion in public procurement.

## 7.2 Contribution

Accessing public procurement data poses a considerable challenge, which is made more difficult by a small amount of academic literature on this area of collusion detection in public procurement. Very few publications offer both their findings and the associated data. This thesis investigates a variety of machine learning models that were not used before to detect collusion in the context of public procurement. Furthermore, it investigates various configurations differing in their access to bid information. This study distinguishes itself from extant literature by employing hyperparameter tuning across all machine learning models utilized herein. Notably, these models exhibited notable improvement after the tuning process. The support vector classifier and gradient boosting classifier demonstrated substantial enhancement after the tuning. In contrast, models such

as k-nearest neighbors, random forest, and multilayer perceptron exhibited improvements, however, the improvements observed in these models were not as prominent.

### 7.3 Limitations

One primary limitation of this study arises from the constraints imposed by the availability of data. The study's potential for exploration could be significantly enhanced with access to more comprehensive information on auctions and company-specific data. The incorporation of additional data has the potential to open new avenues for analysis, offering a richer understanding of participant behavior within auctions. This expanded dataset could explain critical aspects such as financial positions, bidding strategies, and potential indicators of collusion. Employing machine learning techniques on this dataset might uncover previously unnoticed connections between financial standing and procurement behavior, thereby advancing the development of more precise collusion detection models. One potential other limitation of this research is exclusive focus on machine learning techniques without considering domain-specific nuances or incorporating qualitative data from industry experts or stakeholders.

### 7.4 Future Research Directions

Future research could focus on exploring novel feature engineering techniques tailored specifically to bid timing among bidders, which can potentially reveal deeper insights into collusion dynamics within this context. Integrating auction-specific information with company financial data presents an avenue for constructing a more comprehensive dataset, offering a nuanced understanding of participant behavior within auctions. By leveraging machine learning techniques on this amalgamated dataset, future endeavors aim to uncover previously undiscovered connections between financial standing and bidding behavior, thus advancing the accuracy and effectiveness of collusion detection models within public procurement settings.

**REFERENCES**


Abrantes-Metz, R. M., Froeb, L. M., Geweke, J., & Taylor, C. T. (2006). A variance screen for collusion. International Journal of Industrial Organization, 24(3), 467–486. https://doi.org/10.1016/j.ijindorg.2005.10.003

Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. American Statistician, 46(3), 175–185. https://doi.org/10.1080/00031305.1992.10475879

Aoyagi, M. (2003). Bid rotation and collusion in repeated auctions. Journal of Economic Theory, 112(1), 79–105. https://doi.org/10.1016/S0022-0531(03)00071-1

Bajari, P., & Ye, L. (2003). Deciding Between Competition and Collusion. The Review of Economics and Statistics, 85(4), 971–989. https://doi.org/10.1162/003465303772815871

Bolton, R. J., & Hand, D. J. (n.d.-a). Unsupervised Profiling Methods for Fraud Detection.

Bolton, R. J., & Hand, D. J. (2002). Statistical Fraud Detection: A Review. In Statistical Science (Vol. 17, Issue 3).

Bosio, E., Djankov, S., Glaeser, E., & Shleifer, A. (2022). Public Procurement in Law and Practice†. American Economic Review, 112(4), 1091–1117. https://doi.org/10.1257/aer.20200738

Breiman, L. (2001). Random Forests (Vol. 45).

C. Phua, V. Lee, K. Smith, & R. Gayler. (n.d.). A Comprehensive Survey of Data Mining-based Fraud Detection Research.

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 13-17-August-2016, 785–794. https://doi.org/10.1145/2939672.2939785

Conley, T. G., & Decarolis, F. (2016a). Detecting bidders groups in collusive auctions. American Economic Journal: Microeconomics, 8(2), 1–38. https://doi.org/10.1257/mic.20130254

Conley, T. G., & Decarolis, F. (2016b). Detecting bidders groups in collusive auctions. American Economic Journal: Microeconomics, 8(2), 1–38. https://doi.org/10.1257/mic.20130254

Cortes, C., Vapnik, V., & Saitta, L. (1995). Support-Vector Networks Editor. In Machine Leaming (Vol. 20). Kluwer Academic Publishers.

Dahlberg, E. (n.d.). Legal obstacles in Member States to Single Market rules.

Decarolis, F., & Giorgiantonio, C. (2022). Corruption red flags in public procurement: new evidence from Italian calls for tenders. EPJ Data Science, 11(1). https://doi.org/10.1140/epjds/s13688-022-00325-x

Dhurandhar, A., Graves, B., Ravi, R., Maniachari, G., & Ettl, M. (2015). Big data system for analyzing risky procurement entities. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2015-August, 1741–1750. https://doi.org/10.1145/2783258.2788563

Domingos, S. L., Carvalho, R. N., Carvalho, R. S., & Ramos, G. N. (2017). Identifying it purchases anomalies in the Brazilian Government Procurement System using deep learning. Proceedings - 2016 15th IEEE International Conference on Machine Learning and Applications, ICMLA 2016, 722–727. https://doi.org/10.1109/ICMLA.2016.106

Dong, X., Peng, Q., Wu, H., Chang, Z., Yue, Y., & Zeng, Y. (2019). New principle for busbar protection based on the Euclidean distance algorithm. PLoS ONE, 14(7). https://doi.org/10.1371/journal.pone.0219320

Feinstein, J. S., Block, M. K., & Nold, F. C. (1985). Asymmetric Information and Collusive Behavior in Auction Markets (Vol. 75, Issue 3).

Fix, E., & Hodges, J. L. (1989). Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties. 57(3), 238–247.

Friedman, J. H. (2001). Greedy Function Approximation: A Gradient Boosting Machine. 29(5), 1189–1232.

Gallego, J., Rivero, G., & Martínez, J. (2021). Preventing rather than punishing: An early warning model of malfeasance in public procurement. International Journal of Forecasting, 37(1), 360–377. https://doi.org/10.1016/j.ijforecast.2020.06.006

García Rodríguez, M. J., Rodríguez-Montequín, V., Ballesteros-Pérez, P., Love, P. E. D., & Signor, R. (2022). Collusion detection in public procurement auctions with machine learning algorithms. Automation in Construction, 133. https://doi.org/10.1016/j.autcon.2021.104047

García-Teodoro, P., Díaz-Verdejo, J., Maciá-Fernández, G., & Vázquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. Computers and Security, 28(1–2), 18–28. https://doi.org/10.1016/j.cose.2008.08.003

Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. Machine Learning, 63(1), 3–42. https://doi.org/10.1007/s10994-006-6226-1

Ghedini Ralha, C., & Sarmento Silva, C. V. (2012). A multi-agent data mining system for cartel detection in Brazilian government procurement. Expert Systems with Applications, 39(14), 11642–11656. https://doi.org/10.1016/j.eswa.2012.04.037

Harrington, J. E. (2004). Detecting Cartels *. http://hdl.handle.net/10419/72037

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., … Oliphant, T. E.

(2020). Array programming with NumPy. In Nature (Vol. 585, Issue 7825, pp. 357–362). Nature Research. https://doi.org/10.1038/s41586-020-2649-2

Huang, J., & Ling, C. X. (n.d.). Using AUC and Accuracy in Evaluating Learning Algorithms. http://www.computer.org/publications/dlib

Huber, M., & Imhof, D. (2019). Machine learning with screens for detecting bid-rigging cartels. International Journal of Industrial Organization, 65, 277–301. https://doi.org/10.1016/j.ijindorg.2019.04.002

Huber, M., Imhof, D., & Ishii, R. (n.d.). Faculté des sciences économiques et sociales et du management Wirtschafts-und sozialwissenschaftliche Fakultät Transnational machine learning with screens for flagging bid-rigging cartels Transnational machine learning with screens for flagging bid-rigging cartels. http://www.oecd.org/competition/cartels/fightingbidriggininpublicprocurement.htm.

Huber, M., Imhof, D., & Ishii, R. (2022). Transnational machine learning with screens for flagging bid-rigging cartels. Journal of the Royal Statistical Society. Series A: Statistics in Society, 185(3), 1074–1114. https://doi.org/10.1111/rssa.12811

Imhof, D., & Wallimann, H. (2021). Detecting bid-rigging coalitions in different countries and auction formats. International Review of Law and Economics, 68. https://doi.org/10.1016/j.irle.2021.106016

Jiménez, J. L., & Perdiguero, J. (2012). Does Rigidity of Prices Hide Collusion? Review of Industrial Organization, 41(3), 223–248. https://doi.org/10.1007/s11151-012-9337-9

Larochelle, H., Erhan, D., Courville, A., Bergstra, J., & Bengio, Y. (n.d.). An Empirical Evaluation of Deep Architectures on Problems with Many Factors of Variation.

Lecture Notes in Artificial Intelligence 4304 Subseries of Lecture Notes in Computer Science. (n.d.).

Mcafee, R. P., & Mcmillan, J. (1992). Bidding Rings (Vol. 82, Issue 3).

Mckinney, W. (n.d.). pandas: a Foundational Python Library for Data Analysis and Statistics. http://pandas.sf.net

Merentitis, A., Debes, C., & Heremans, R. (2014). Ensemble learning in hyperspectral image classification: Toward selecting a favorable bias-variance tradeoff. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 7(4), 1089–1102. https://doi.org/10.1109/JSTARS.2013.2295513

Monczka, R. M. (2009). Purchasing and supply chain management. South-Western.

Pedregosa FABIANPEDREGOSA, F., Michel, V., Grisel OLIVIERGRISEL, O., Blondel, M., Prettenhofer, P., Weiss, R., Vanderplas, J., Cournapeau, D., Pedregosa, F., Varoquaux, G., Gramfort, A., Thirion, B., Grisel, O., Dubourg, V., Passos, A., Brucher, M., Perrot andÉdouardand, M., Duchesnay, andÉdouard, & Duchesnay EDOUARDDUCHESNAY, Fré. (2011). Scikit-learn: Machine Learning in Python Gaël Varoquaux Bertrand Thirion Vincent Dubourg Alexandre Passos PEDREGOSA, VAROQUAUX, GRAMFORT ET AL. Matthieu Perrot. In Journal of Machine Learning Research (Vol. 12). http://scikit-learn.sourceforge.net.

Perspective, A. A. (n.d.). Chapman & Hall/CRC Machine Learning & Pattern Recognition Series Chapman & Hall/CRC Machine Learning & Pattern Recognition Series Machine Learning M AC H I N E LEARNING.

Pfizer, M. J., & Jakobsson, M. (2007). Bid-rigging in Swedish Procurement Auctions Bid Rigging in Swedish Procurement Auctions. https://www.researchgate.net/publication/228878107

Porter, R. H., & Zona, J. D. (1999). Ohio School Milk Markets: An Analysis of Bidding. The RAND Journal of Economics, 30(2), 263. https://doi.org/10.2307/2556080

Public procurement - OECD. (n.d.). Retrieved September 9, 2023, from https://www.oecd.org/gov/public-procurement/

Quinlan, J. R. (1986). Induction of Decision Trees. In Machine Learning (Vol. 1).

Rabuzin, K., & Modrušan, N. (2019). Prediction of public procurement corruption indices using machine learning methods. IC3K 2019 - Proceedings of the 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, 3, 333–340. https://doi.org/10.5220/0008353603330340

Rosenblatt, F. (1960). Perceptron Simulation Experiments. Proceedings of the IRE, 48(3), 301–309. https://doi.org/10.1109/JRPROC.1960.287598

Ruppert, D. (2004). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Journal of the American Statistical Association, 99(466). https://doi.org/10.1198/jasa.2004.s339

Rustiarini, N. W., T, S., Nurkholis, N., & Andayani, W. (2019). Why people commit public procurement fraud? The fraud diamond view. In Journal of Public Procurement (Vol. 19, Issue 4, pp. 345–362). Emerald Group Holdings Ltd. https://doi.org/10.1108/JOPP-02-2019-0012

Sampathkumar, A., Rastogi, R., Arukonda, S., Shankar, A., Kautish, S., & Sivaram, M. (2020). An efficient hybrid methodology for detection of cancer-causing gene using CSC for micro array data. Journal of Ambient Intelligence and Humanized Computing, 11(11), 4743–4751. https://doi.org/10.1007/s12652-020-01731-7

Seabold, S., & Perktold, J. (2010). Statsmodels: Econometric and Statistical Modeling with Python. In PROC. OF THE 9th PYTHON IN SCIENCE CONF. http://statsmodels.sourceforge.net/

Signor, R., Ballesteros-Pérez, P., & Love, P. E. D. (2023a). Collusion Detection in Infrastructure Procurement: A Modified Order Statistic Method for Uncapped Auctions. IEEE TRANSACTIONS ON ENGINEERING MANAGEMENT, 70(2). https://doi.org/10.1109/TEM.2021

Signor, R., Ballesteros-Pérez, P., & Love, P. E. D. (2023b). Collusion Detection in Infrastructure Procurement: A Modified Order Statistic Method for Uncapped Auctions. IEEE TRANSACTIONS ON ENGINEERING MANAGEMENT, 70(2). https://doi.org/10.1109/TEM.2021

Sokoliuk, A., Kondratenko, G., Sidenko, I., Kondratenko, Y., Khomchenko, A., & Atamanyuk, I. (2021). Machine Learning Algorithms for Binary Classification of Liver Disease. 2020 IEEE International Conference on Problems of Infocommunications Science and Technology, PIC S and T 2020 - Proceedings, 417–421. https://doi.org/10.1109/PICST51311.2020.9468051

Sokolova, M., & Lapalme, G. (2009a). A systematic analysis of performance measures for classification tasks. Information Processing and Management, 45(4), 427–437. https://doi.org/10.1016/j.ipm.2009.03.002

Sokolova, M., & Lapalme, G. (2009b). A systematic analysis of performance measures for classification tasks. Information Processing & Management, 45(4), 427–437. https://doi.org/10.1016/J.IPM.2009.03.002

Torres-Berru, Y., & Batista, V. F. L. (2021). Data mining to identify anomalies in public procurement rating parameters. Electronics (Switzerland), 10(22). https://doi.org/10.3390/electronics10222873

Universitas Udayana, Institute of Electrical and Electronics Engineers. Indonesia Section, & Institute of Electrical and Electronics Engineers. (n.d.). Proceedings of 2016 International Conference on Data and Software Engineering (ICoDSE) : Udayana University, Denpasar, Bali, Indonesia, October 26th-27th, 2016.

Wallimann, H., Imhof, D., & Huber, M. (2022a). A Machine Learning Approach for Flagging Incomplete Bid-Rigging Cartels. Computational Economics. https://doi.org/10.1007/s10614-022-10315-w

Wallimann, H., Imhof, D., & Huber, M. (2022b). A Machine Learning Approach for Flagging Incomplete Bid-Rigging Cartels. Computational Economics. https://doi.org/10.1007/s10614-022-10315-w

Wang, W., Chakraborty, G., & Chakraborty, B. (2021). Predicting the risk of chronic kidney disease (Ckd) using machine learning algorithm. Applied Sciences (Switzerland), 11(1), 1–17. https://doi.org/10.3390/app11010202