Handelshøyskolen BI

GRA 19703 Master Thesis

Thesis Master of Science 100% - W

Predefinert informasjon

Startdato:	09-01-2023 09:00 CET	**Termin:**	202310
Sluttdato:	03-07-2023 12:00 CEST	**Vurderingsform:**	Norsk 6-trinns skala (A-F)
Eksamensform:	T		
Flowkode:	202310||11184||IN00||W||T		
Intern sensor:	(Anonymisert)		

Deltaker

Navn:	Zhizhen Li og Xiaoxiao Tao

Informasjon fra deltaker

Tittel *:	Employee Turnover Prediction with Supervised Machine Learning
Navn på veileder *:	Jan Kudlicka

Inneholder besvarelsen konfidensielt materiale?:	Nei	**Kan besvarelsen offentliggjøres?:**	Ja

Gruppe

Gruppenavn:	(Anonymisert)
Gruppenummer:	137
Andre medlemmer i gruppen:	

- Master Thesis -

Employee Turnover Prediction with Supervised Machine Learning

Students:
Xiaoxiao Tao, Zhizhen Li

Supervisor:
Jan Kudlicka

Hand-in Date:
27.06.2023

Program:
Master of Science in Business Analytics

Table of Contents

Acknowledgments

We would like to express our sincere gratitude to all those who have supported and contributed to the completion of this thesis.

First and foremost, we would like to thank our supervisor Jan Kudlicka, for his guidance, expertise, and continuous support throughout the entire period. His valuable insights and feedback have been instrumental in shaping this thesis.

We would also like to acknowledge the support and resources provided by the BI Norwegian Business School and we are grateful for the knowledge and skills we have gained during our time at this esteemed institution.

We would like to extend our appreciation to our family and friends who have provided encouragement, assistance, and valuable discussions throughout this journey. Their support and camaraderie have been invaluable in making this thesis a reality.

Although it is not possible to mention everyone individually, we would like to express our heartfelt appreciation to all those who have contributed in their own way to the completion of this thesis.

Thank you all for your invaluable support and encouragement.

List of Abbreviation

ANOVA	Analysis of Variance
DT	Decision Tree
dfB	Degree of Freedom Between Groups
dfW	Degree of Freedom Within Groups
FN	False Negative
FP	False Positive
FPR	False Positive Rate
GBDT	Gradient Boosting Decision Tree
HR	Human Resources
IT	Information Technology
KCV	K-fold Cross Validation
KNN	K-Nearest Neighbor
LDA	Linear Discriminant Analysis
LR	Logistic Regression
MCC	Matthews Correlation Coefficient
MLP	Multi-Layer Perception
MSB	Mean Square Between
MSW	Mean Square Within
NB	Naïve Bayes
NN	Neural Networks
PNN	Probabilistic Neural Network

RF	Random Forest
ROC-AUC	Area Under the Receiver Operating Characteristic Curve
SSB	Between-group Sum of Squares
SSW	Within-group Sum of Squares
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
TPR	True Positive Rate
XGBoost	Extreme Gradient Boosting

1. Introduction

1.1 Problem Formulation

Employee turnover is a pervasive issue faced by organizations across various industries (Korff et al., 2015). When employees decide to leave a company, it not only disrupts daily operations but also imposes costs associated with recruiting and training new talent (Cascio & Boudreau, 2011; Matthew & Kung, 2007). As a result, accurately predicting and understanding employee turnover has become a critical objective for many businesses. The advent of machine learning has opened up new possibilities for predicting and analyzing employee turnover. By leveraging vast amounts of data, organizations can develop sophisticated models that effectively forecast the likelihood of an employee leaving the company. These predictive models enable organizations to proactively address turnover risks, devise targeted retention and succession strategies, and create a more stable and successful work environment (Chanodkar et al., 2019; Perryer et al., 2010). This thesis primarily focuses on tackling the challenge of employee turnover prediction using supervised machine learning models. By utilizing four widely recognized supervised learning models, namely Decision Tree (DT), Random Forest (RF), Gradient Boosting Decision Tree (GBDT), and Extreme Gradient Boosting (XGBoost), we conduct a comparative analysis to determine the model that demonstrates the highest predictive power for employee turnover. Through our analysis, we aim to contribute to the field of employee turnover prediction by advancing the understanding of the factors that drive turnover and developing effective prediction models.

1.2 Contributions

This section highlights the significant contributions of our thesis in the field of predicting employee turnover using supervised machine learning models. Our thesis makes the following key contributions:

Comprehensive Analysis: Our thesis undertakes a thorough examination of the factors influencing employee turnover and employs four widely recognized supervised machine learning models to achieve accurate turnover predictions. Through a meticulous analysis of demographic features and the application of

robust methodologies, our study offers a comprehensive understanding of the predictive models involved in employee turnover.

Dataset with Broad Coverage: Our study utilizes a dataset with extensive coverage, encompassing diverse industries and countries. This dataset enables us to capture a more comprehensive understanding of employee turnover dynamics and facilitates the generalization of our findings to different contexts.

Comparative Evaluation of Supervised Learning Models: We compare the performance of four popular supervised learning models, including DT, RF, GBDT, and XGBoost. By evaluating their predictive abilities and identifying the model with the highest performance, we offer valuable insights into the most effective model for predicting employee turnover.

Addressing Previous Limitations: We overcome limitations identified in previous research, such as limited generalization, feature selection, and metrics selection. By employing advanced techniques and methodologies, we aim to enhance the accuracy and reliability of employee turnover prediction models.

Practical Implications: Our thesis provides practical implications for organizations in developing effective retention or succession strategies. By identifying the factors contributing to employee turnover and accurately predicting turnover, organizations can take proactive measures to mitigate turnover risks and enhance their workforce management practices.

Overall, our study contributes to the existing body of knowledge by offering a comprehensive analysis of employee turnover prediction, addressing previous limitations, and providing practical insights for organizations to make informed decisions regarding their workforce management strategies.

1.3 Structure of the Thesis

This thesis is organized into eight chapters, each focusing on a specific aspect of predicting employee turnover using supervised machine learning models. The following provides an overview of the structure and content of the thesis:

Chapter 1 presents the problem formulation, emphasizing the importance of accurately predicting employee turnover. We delve into the significance of this topic and elaborate on the contributions our work brings to the field.

Chapter 2 delves into the background information related to employee turnover. We examine the impact of employee voluntary turnover on organizations and emphasize the benefits of predicting employee turnover. Additionally, we examine the role of machine learning in addressing this issue and provide a summary of previous studies conducted in this domain. We also discuss the limitations of prior research and outline our efforts to overcome them in this study.

Chapter 3 focuses on the methodology employed in our study. We describe the statistical techniques used for feature selection, including Analysis of Variance (ANOVA), Chi-square Test, and Correlation Analysis. We also delve into the four supervised learning models utilized in our study: Decision Tree (DT), Random Forest (RF), Gradient Boosting Decision Tree (GBDT), and Extreme Gradient Boosting (XGBoost). Furthermore, we discuss the evaluation metrics used to assess the performance of these models.

Chapter 4 provides detailed information about the data used in our study. We discuss the data source, data structure, and the process of data cleaning. Additionally, we present a descriptive analysis of the data and explain the steps taken for data preprocessing, including categorical data encoding and data splitting. We also elaborate on the feature selection methods applied, namely ANOVA, Chi-square Test, and Correlation Analysis.

Chapter 5 presents the training procedures for each of the machine learning models utilized in our study, including the hyperparameter tuning process. We thoroughly evaluate each model, examining their performance both with and without hyperparameter tuning.

Chapter 6 presents the overall results of our study. We discuss the performance of each model and highlight the business value derived from predicting employee turnover. In addition, we identify potential avenues for future research and discuss possible extensions and improvements to the methods and models employed in this study.

Chapter 7 includes additional supplementary material, such as a step-by-step process for conducting feature selection, hyperparameter tuning, and the K-fold Cross Validation (KCV) results for each of the models utilized.

Chapter 8 contains a comprehensive list of the references cited throughout the thesis, ensuring proper attribution of the sources consulted.

By following this structured approach, we aim to provide a comprehensive analysis of predicting employee turnover using supervised machine learning models.

2. Background

In this chapter, we delve into the background of employee turnover prediction and its various aspects. We begin by introducing the concept of employee turnover (2.1) and exploring its implications within organizations. Subsequently, we examine the impact of voluntary employee turnover (2.2), highlighting the significant effects it can have on businesses. Recognizing the importance of predicting and mitigating turnover, we then discuss the benefits of such predictions (2.3), emphasizing the potential advantages for organizations. To facilitate these predictions, we turn our attention to machine learning (2.4) and its potential in this context. Specifically, we explore supervised learning models (2.5) as a powerful tool for predicting employee turnover. Lastly, we address the limitations of previous research (2.6), recognizing the need for further advancements in this area.

2.1 Introduction to Employee Turnover

Employee turnover refers to the number or the percentage of the total number of employees who leave the company and are replaced by hiring and appointing new employees to fill the vacant positions within a certain period of time (Chanodkar et al., 2019). Employees are often regarded as important assets of a company, especially those who are considered valuable by the company. Stovel and Bontis (2002) viewed employee turnover to be the loss of an organization's intellectual capital. According to Korff et al. (2015), employee turnover has been a long-standing problem in companies. Relevant organizational studies have confirmed that a variety of factors can influence a person's decision to quit a position, including job satisfaction, job performance, job security, work environment, wages, and the existence of clearly defined organizational goals (Allen & Griffeth, 1999; Al-Suraihi et al., 2021; Parker, 2014). Storey (2016) stated that employee retention and job satisfaction are interdependent and fundamental to a company's performance. Similar conclusions regarding the impact of organizational direction and support on employee job satisfaction and general commitment were made in Kim et al. (2005) 's study on corporate orientation. Additionally, a number of researchers have discussed the impact of employee demographics on their turnover decisions. Demographics such as age, gender, tenure, ethnicity, education, and marital status have been proven to be strong predictors of employee resignation

(Cotton & Tuttle, 1986; Holtom et al., 2008; Sacco & Schmitt, 2005; von Hippel et al., 2013).

Most studies categorize employee turnover into voluntary and involuntary turnover. Voluntary turnover occurs when the decision to leave the company is made primarily by the employee, including all resignation forms; involuntary turnover, which includes termination, dismissing, and other forms, refers to when the decision to leave the company is made mostly by the employer (Shaw et al., 1998). In contrast to involuntary turnover, which is predictable and manageable, voluntary turnover is often unpredictable and can have a greater impact on companies (Chanodkar et al., 2019). The analysis in this paper focuses on voluntary turnover. Additionally, according to the characteristics of the departing employee, employee turnover can also be classified into internal and external turnover, as well as skilled and unskilled turnover. Internal turnover is when employees move from one position to another within the same organization, while external turnover occurs when employees leave to work for another organization. Skilled turnover refers to the departure of highly skilled and educated employees, while unskilled turnover involves the departure of employees in positions that require untrained, unskilled, or uneducated workers (Akinyomi, 2016). These categories provide a deeper understanding of the factors and consequences of employee turnover in organizations.

2.2 Impact of Employee Voluntary Turnover

Voluntary employee turnover can negatively affect several aspects of an organization. A high turnover rate might harm the company financially because of high indirect costs, such as the cost of hiring, training, and developing new employees (Cascio & Boudreau, 2011; Matthew & Kung, 2007). In terms of company resources, training new employees requires additional time, manpower, and material resources (Bapna et al., 2012). At the same time, the productivity of the company will be impacted since new employees often need some time to familiarize themselves with the business operation (Matthew & Kung, 2007). In addition, both customer satisfaction (Kamalanabhan et al., 2009) and company reputation (Beheshtifar & Allahyary, 2012) can be influenced in a similar manner. A high rate of employee turnover is also bad for a company's reputation (Beheshtifar & Allahyary, 2012). Internally, the departure of experienced

employees can lead to low morale and disrupt ongoing work (Matthew & Kung, 2007; Punnoose & Ajit, 2016). Zhang (2016) decomposed the cost of employee turnover into two categories: explicit costs (such as hiring, training, and productivity loss), and hidden costs (morale, corporate reputation, damage to position chain, loss of opportunity, etc.). Moreover, the consequences of employee turnover can vary across industries. For example, research focused on the Information Technology (IT) sector by Shanmugam and Giri Babu (2016) highlights that high turnover leads to decreased productivity in this field. Sexton et al. (2005) conducted a thorough examination of the customer service industry, revealing that unexpected employee departures harm customer loyalty and diminish service quality. Additionally, in high-tech industries, the replacement of employees possessing specialized skill sets or domain expertise presents a significant challenge (Esmaieeli Sikaroudi et al., 2015). While there are drawbacks, employee turnover can also bring benefits such as replacing underperforming employees and fostering organizational creativity, flexibility, and adaptability (Purohit, 2016; Zhang, 2016). Overall, organizations need to carefully manage employee turnover to mitigate negative consequences and capitalize on potential advantages.

2.3 Benefits of Predicting Employee Turnover

Given the internal and external impact of employee turnover discussed in the previous section, there is no doubt that it is beneficial for companies to anticipate employee turnover. Vasantham and Swarnalatha (2015) concluded that the retention of competent employees is critical to a company's long-term health and success. By predicting employee turnover, companies can take appropriate proactive actions, such as planning for retention and succession (Chanodkar et al., 2019). If a drastic increase in employee resignation is predicted, both management and Human Resources (HR) teams can take necessary precautions in advance. As a result, companies are able to reduce or maintain employee turnover as needed, thereby increasing overall productivity and profitability. Accurate forecasts also provide companies with insights to estimate the budget for human resource management-related activities, such as cost per hire (Chanodkar et al., 2019). In addition, Punnoose and Ajit (2016) pointed out that any organization that wants to take the appropriate action to maintain its market position and accomplishment must first determine the main causes of employee attrition. Through the development of a predictive model, companies can gain valuable insights into the

key reasons behind employee turnover, provide the right incentives for employees, or find suitable personnel for future vacancies. Furthermore, employee turnover predictions can be used to formulate strategies related to productivity or expansion for the continued growth and development of the company (Perryer et al., 2010).

2.4 Machine Learning

Machine learning is a field of study in computer science that focuses on the use of data and algorithms to imitate the learning process of humans, with the aim of continuously improving its accuracy over time (Woolf, 2009). It can be categorized into several types based on different learning processes and methods. According to recent studies, these types include supervised learning, unsupervised learning, semi-supervised learning, reinforcement learning, and deep learning (Alpaydin, 2020; Géron, 2022; LeCun et al., 2015; Sutton & Barto, 2018). Supervised learning involves training models using labeled data, where the desired output is already known. Conversely, unsupervised learning involves training models on unlabeled data, where the desired output is unknown. Semi-supervised learning combines elements of both supervised and unsupervised learning, utilizing datasets that contain both labeled and unlabeled data. Reinforcement learning involves training algorithms to make decisions by leveraging feedback received from the environment. Finally, deep learning employs artificial neural networks to emulate the structure and function of the human brain, enabling the resolution of complex problems. Given that the nature of our work is to predict whether an employee will resign, and the classes in the dataset are known: all employees will be labeled as resigned (positive) or not resigned (not resigned). Therefore, the machine learning models used in our analysis are all supervised learning.

The development of machine learning models has resulted in the emergence of robust quantitative techniques that are being applied across a range of industries, including biology and medical sciences (Bakry et al., 2017; Seddik & Shawky, 2015), transportation (Mathias & Ragusa, 2017; Ye et al., 2009), and political science (Durant & Smith, 2007). In the field of human resource management, different machine learning models have been studied by researchers to improve productivity in areas such as employee performance prediction (Al-Radaide & Al Nagi, 2012), personnel selection (Chien & Chen, 2008), and recruitment system construction (Li et al., 2010). Machine learning also serves as a valuable tool for

predicting employee turnover, as demonstrated by recent studies summarized in the next chapter.

2.5 Supervised Learning Models for Employee Turnover

The use of supervised machine learning algorithms has been explored by several studies to predict employee turnover. Alao and Adeyemo (2013) investigated the use of Decision Tree (DT) algorithms in predicting employee attrition and identified some of the important factors in predicting employee attrition, including salary and tenure. Esmaieeli Sikaroudi et al. (2015) proposed using several machine learning algorithms to predict employee turnover, including DT, K-Nearest Neighbor (KNN), Multi-Layer Perception (MLP), Naïve Bayes (NB), Probabilistic Neural Network (PNN), Random Forests (RF), and Support Vector Machine (SVM). They found that the RF model achieved the best performance in predicting employee turnover, with an accuracy rate of 90%, and identified work experience as the most important factor in predicting employee turnover. Punnoose and Ajit (2016) compared seven machine learning algorithms in predicting employee turnover. These algorithms are Linear Discriminant Analysis (LDA), Logistic Regression (LR), KNN, NB, RF, SVM, and Extreme Gradient Boosting (XGBoost). The researchers found that XGBoost outperformed the other models in terms of Area Under the Receiver Operating Characteristic Curve (ROC-AUC). Zhao et al. (2018) explored machine learning algorithms including DT, Gradient Boosting Decision Tree (GBDT), KNN, LDA, LR, NB, Neural Networks (NN), RF, SVM, XGBoost, and found that GBDT achieved the best performance in predicting employee turnover using ROC-AUC. Overall, these studies highlight the potential of machine learning algorithms in predicting employee turnover and identifying factors that contribute to it.

In Table 2-1, we present a summary of recent publications (after 2010) in the field of predicting employee turnover. Researchers in these studies focused on one or more machine learning models to predict employee turnover and compared their performance using pre-selected metrics to identify the model with the best predictive ability. Building upon the insights gained from this literature review, our thesis incorporates four wildly recognized supervised learning models, namely DT, RF, GBDT, and XGBoost, to develop predictive models for employee turnover.

Table 2-1. Summary of Recent Research on Supervised Machine Learning for Predicting Employee Turnover

Study	Dataset Size	Positive %	Number of Features	Sentimental Features	Industry	Country	Supervised Machine Learning Model	Metrics	Identified Key Features
Saradhi & Palshikar, 2011	1,575	11.81	12	No	-	-	NB, RF, **SVM**	Accuracy, TN, **TP**	-
Alao & Adeyemo, 2013	4326	7.14	6	No	Education	Nigeria	**DT**	**Accuracy**, F-Measure, FP, Preccision, Recall, ROC-AUC, TP	Salary, Tenure
Esmaieeli Sikaroudi et al., 2015	-	20	14	Yes	Manufacture	Iran	DT, KNN, MLP, NB, PNN, **RF**, SVM	**Accuracy**	Number of job changing, Knowledge about the working conditions and laws, Perseverance and interest to work
Punnoose & Ajit, 2016	73,115	-	33	Yes	Retailer	United States	LDA, LR, KNN, NB, RF, SVM, **XGBoost**	Memory utilizaiton, Model run time, **ROC-AUC**	-
Khera & Divya, 2018	1,650	16.13	22	Yes	IT	India	SVM	**Accuracy**	-
Zhao et al., 2018 (1)	9,089	28.34	19	Yes	Bank	United States	DT, **GBDT**, KNN, LDA, LR, NB, NN, RF, SVM, XGBoost	Accuracy, F1 score, Preccision, Recall, **ROC-AUC**	-
Zhao et al., 2018 (2)	1,470	16.12	31	-	-	-	DT, **GBDT**, KNN, LDA, LR, NB, NN, RF, SVM, XGBoost	Accuracy, F1 score, Preccision, Recall, **ROC-AUC**	-
Chanodkar et al., 2019	1,470	19.22	-	-	-	-	AdaBoost, LR, NB, RF, **SVM**	**Accuracy**, F-Measure, FP, MCC, Precision, Recall, ROC-AUC, TP	Frequent business travel, Distance from home, Age, Overtime, Gap in promotion
Anwar Hossen et al., 2021	15,000	23.81	9	Yes	-	-	DT, KNN, MLP, NB, **RF**, SVM	**Accuracy**, ROC-AUC	Satisfaction level, Last evaluation, Number of projects

Study: Articles are sorted by publication date. Zhao et al. (2018) used two datasets in their study and therefore appears twice in this table.
Dataset size: In some articles it refers to the number of data points (i.e. number of rows), such as Punnoose and Ajit (2016). In this case, if the company collects data on a monthly basis, an employee may appear multiple times in the same dataset. In some other articles it refers to the actual number of employees. Most articles did not make a detailed distinction.
Positive %: Percentage of data labelled as resigned (the positive class).
Sentimental Features: Whether features representing employee emotions, such as satisfaction level expressed by employees, are included in the dataset.
Supervised Machine Learning Model: The supervised learning models analysed in the study, with the best model concluded in the article marked in bold.
Metrics: The metrics used to evaluate model performance in the study. The metrics use to determined the best model in the article are marked in bold.
Identified Key Features: Features identified in the study as having high predictive power for employee turnover.
"-" : When the article did not provide the required information.

2.6 Limitation of Previous Research

Despite the fact that a number of studies have focused on utilizing machine learning to develop predictive models for employee turnover, the models developed from

these studies tend to be specific to the industry or country in which they were developed, making it challenging to apply them in other contexts. One possible reason for this is the limited availability of data. Developing predictive models for employee turnover requires access to HR data that includes sensitive and confidential personal information, such as employee performance, salary, tenure, and reasons for leaving. However, companies generally prefer to keep such information private (Seth & Sethi, 2011), which makes it challenging to collect data on a large scale within one study, especially when it involves multiple countries and industries. Most of the studies summarized in Table 2-1 conducted analyses using datasets belonging to a single industry and from a single country, while others did not mention the source of their datasets. The resulting models, therefore, have a restricted capacity to be generalized. The dataset used in our thesis is provided by a global company with operations covering multiple industries. The dataset comprises ten countries where the company has the highest number of employees across the globe. The countries are China, Germany, India, Italy, the Netherlands, Norway, Poland, Spain, the United Kingdom, and the United States. Additionally, it includes nine industries, namely consultancy, cyber security, data management, energy, HR, IT, insurance, maritime and supply chain. A thorough description and descriptive analysis of the dataset are provided in Chapter 4.

Another reason for the limited generalization of previous studies can be the inclusion of sentimental features. These features, such as employee satisfaction level, job security perceptions, and peer relationships, are often incorporated into the dataset to predict employee turnover. Although, as mentioned in Chapter 2.1, studies in organizational research have demonstrated the importance of sentimental features in reflecting employees' decisions to leave their jobs, it is difficult to ensure the consistency and accuracy of such data. Companies generally send employees surveys or questionnaires to find out how they feel about the company, their job, or their colleagues (Moyes et al., 2008; Okechukwu, 2017; Saleem et al., 2010). However, the design of each company's survey is likely to be different. For instance, some companies may ask about job satisfaction, while others may ask about workload satisfaction. This means that the sentimental features included in datasets from different companies are likely to be incomparable. In addition, individuals perceive satisfaction differently, especially across different cultures (Kristensen & Johansson, 2008). As a result, analyses designed using sentimental

features are difficult to verify in other companies, let alone in a different industry or country. To mitigate this issue, the analysis in our thesis only focuses on employee demographic features such as age, gender, and tenure. These features are available to most companies and have been consistently identified as significant factors in predicting employee turnover in prior research (Cotton & Tuttle, 1986; Holtom et al., 2008; Sacco & Schmitt, 2005; von Hippel et al., 2013)

In previous studies, the evaluation of supervised learning models in predicting employee turnover often relied heavily on accuracy as the primary metric. However, this approach may have limitations when working with imbalanced datasets (Boughorbel et al., 2017; Jeni et al., 2013). Imbalanced datasets are common in employee turnover prediction studies because the number of employees who actually leave a company is typically a small fraction compared to the total number of employees. Our thesis deals with a highly imbalanced dataset, with only 2.66% of the data labeled as “resigned” (positive). In cases like these, accuracy is not a practical metric for evaluating model performance. For instance, a model can achieve 97.33% accuracy simply by predicting all instances as negative. Therefore, we use F1score and ROC-AUC as our preferred metrics in the analysis. These metrics are more suitable for handling imbalanced datasets (Cahyana et al., 2019; Zhao et al., 2018). Detailed information on the calculation of each metric can be found in Chapter 3.10.

In conclusion, previous research on employee turnover prediction using machine learning has faced challenges regarding generalizability and practical application. The specificity of the models developed for particular industries or countries has hindered their transferability to diverse organizational contexts. Our thesis addresses this limitation by utilizing a dataset encompassing multiple countries and industries. In addition, instead of relying on subjective sentimental features, we incorporate a diverse range of demographic features to enhance the robustness and reliability of our models. Previous studies also heavily relied on accuracy as the primary evaluation metric, which may not be suitable for imbalanced datasets commonly used in employee turnover prediction. To address this issue, we adopt F1score and ROC-AUC as our preferred evaluation metrics, as they provide more robust performance measures for imbalanced data. By addressing these limitations and adopting a comprehensive approach, our thesis strives to make valuable contributions to the field of employee turnover prediction using machine learning.

3. Methodology

In this chapter, we present the methodology employed to develop and evaluate our predictive models for employee turnover. We begin by discussing the feature selection techniques used, namely Analysis of Variance (ANOVA), Chi-square Test, and Correlation Analysis (3.1-3.3). These methods enable us to identify the most relevant features for predicting employee turnover. Next, we delve into the machine learning models employed in our analysis, including Decision Tree (DT), Random Forest (RF), Gradient Boosting Decision Tree (GBDT), and Extreme Gradient Boosting (XGBoost) (3.4-3.7). These models were carefully chosen based on their proven performance in predicting employee turnover and their wide adoption in the field of machine learning. In order to ensure the robustness of our models, we employ K-fold Cross Validation (KCV) (3.8) to assess their stability and generalizability. For evaluating model performance, we utilize various metrics (3.9-3.10). These include the Confusion Matrix, which provides a comprehensive overview of model predictions, and relevant metrics such as accuracy, precision, recall, and F1 score (3.10.1-3.10.4). Additionally, we employ the Area Under the Receiver Operating Characteristic Curve (ROC-AUC) to evaluate the models' discrimination power and ability to handle imbalanced datasets (3.10.5).

3.1 Analysis of Variance (ANOVA)

In our analysis, we employ Analysis of Variance (ANOVA) as a feature selection technique to identify relevant numerical features for our machine learning model. ANOVA is a statistical method that assesses the statistical significance of differences between groups (Kishore et al., 2017), in this case, the relationship between each numerical feature and the target variable, "Resigned". The steps for using ANOVA for feature selection in our analysis can be outlined as follows:

1. Data Preparation: We start by preparing our dataset, ensuring that it is properly formatted and contains the target variable and the numerical features of interest, such as Age, Tenure, and so on.
2. Grouping: We divide our dataset into two groups corresponding to the two classes: "Resigned" and "Not Resigned".
3. F-statistic Calculation: Using the grouped data, we calculate the F-statistic for each selected feature. This statistic measures the variability between the

groups compared to the variability within each group. It quantifies the extent to which the feature explains the variation in the target variable.

4. Significance Testing: We assess the statistical significance of the F-statistic by calculating the p-value. The p-value represents the probability of observing a result as extreme as, or more extreme than, the one obtained under the null hypothesis. A low p-value indicates a significant difference between the groups and suggests that the feature is informative for predicting the target variable.
5. Feature Selection: Based on the calculated p-values, we rank the features in descending order of their significance. We set the significance threshold to 0.05 and select the features with p-values below this threshold. These features are considered relevant to our model.

3.1.1 F-statistic Calculation

In Step 3 of using ANOVA for feature selection, we calculate the F-statistic for each feature. Mathematically, the F-statistic can be calculated using the following formula:

$$F = \frac{MSB}{MSW}, \tag{3-1}$$

where MSB denotes the mean square between groups and MSW denotes the mean square within groups.

To calculate MSB, we compute the sum of squares between groups (SSB) by summing the squared differences between the group means and the overall mean, and then divide it by the degrees of freedom between groups (dfB). The formula for MSB is:

$$MSB = \frac{SSB}{dfB} \tag{3-2}$$

To calculate MSW, we compute the sum of squares within groups (SSW) by summing the squared differences between each observation and its respective group mean, and then divide it by the degrees of freedom within groups (dfW). The formula for MSW is:

$$MSW = \frac{SSW}{dfW} \tag{3-3}$$

The dfB is equal to the number of groups minus one, while the dfW is equal to the total number of observations minus the number of groups.

In summary, ANOVA-based feature selection provides a statistical framework to identify the numerical features that contribute to the target variable. This helps to focus the model's learning on the informative features, potentially improving efficiency and reducing the risk of overfitting.

3.2 Chi-square Test

The Chi-square test is a statistical test used to determine if there is a significant association between two categorical variables by evaluating their independence (Thaseen et al., 2019). In our analysis, we utilize the Chi-square test for feature selection, specifically focusing on categorical features. This test allows us to identify relevant categorical features by measuring their dependence on the target variable.

The process of using the Chi-square test for feature selection is similar to that of using ANOVA, with the difference being the type of features analyzed (categorical instead of numerical) and the statistic used (Chi-square test statistic instead of F-statistic).

3.2.1 Chi-square Test Statistic Calculation

The Chi-square test statistic is calculated based on the observed and expected frequencies. Mathematically, it is calculated as follows:

$$Chi - square = \sum \frac{(Observed - Expected)^2}{Expected} \quad (3\text{-}4)$$

In this equation, the summation symbol (Σ) represents the summation operation conducted over all cells in a contingency table that captures the frequencies of the categorical feature and the target variable. For each cell, we calculate the difference between the observed frequency (the actual count in the cell) and the expected frequency (the count expected assuming independence between the feature and the target variable). We then square this difference, divide it by the expected frequency, and sum up these terms for all cells in the table.

By calculating the Chi-square test statistic, we obtain a single numerical value that measures the overall discrepancy between the observed and expected frequencies. A larger Chi-square value indicates a stronger association between the feature and the target variable, while a smaller value suggests a weaker or no association. This statistic is subsequently used to determine the degrees of freedom and calculate the p-value, enabling us to assess the significance of the association.

In summary, by applying the Chi-square test and analyzing the resulting p-values, we can identify informative categorical features for the target variable. This process helps reduce the dimensionality of the dataset and potentially enhances the efficiency and interpretability of the model.

3.3 Correlation Analysis

Correlation analysis is a statistical technique used to assess the strength and direction of the linear relationship between two numerical variables (Hauke & Kossowski, 2011). In our analysis, we use correlation analysis as an additional feature selection technique to identify the numerical features that are relevant to our target variable. Furthermore, we examine the correlations among the features themselves to gain insights into the relationships between the features and identify potential issues such as multicollinearity. The steps involved in using correlation analysis for feature selection are as follows:

1. Calculation of Correlation Coefficients: We compute the correlation coefficients between each numerical feature and the target variable using Pearson's correlation coefficient, which quantifies the linear relationship between two variables. This coefficient measures the linear relationship between two variables, ranging from -1 to 1. A value close to 1 indicates a strong positive correlation, a value close to -1 indicates a strong negative correlation and a value close to 0 suggests no or weak correlation.
2. Evaluation of Correlation Strength: We assess the strength of the correlation coefficients to identify the numerical features that exhibit a significant relationship with the target variable. Features with high absolute correlation coefficients are considered to have a stronger association with the target variable and are more likely to provide meaningful information for prediction.

3.3.1 Pearson's Correlation Coefficient Calculation

Pearson's correlation coefficient is used to quantify the linear relationship between features and the target variable. It is calculated using the formula:

$$r = \frac{\sum((X_i - \bar{X})(Y_i - \bar{Y}))}{n * \sigma_X * \sigma_Y}, \quad (3\text{-}5)$$

where r represents Pearson's correlation coefficient, X and Y represent the features and the target variable, $\bar{X}$ and $\bar{Y}$ are the mean of X and Y, n represents the number of data points in the dataset, and σ_X and σ_Y are the standard deviations of X and Y.

In summary, by utilizing correlation analysis, we can identify the numerical features that have a strong relationship with the target variable, allowing us to focus on the informative features for our predictive model. This feature selection process helps reduce dimensionality, enhance model interpretability, and potentially improve the model's performance.

3.4 Decision Tree (DT)

Decision Tree (DT) is a supervised learning algorithm that can be used for classification and regression tasks (Breiman et al., 2017). It builds a tree-like model consisting of nodes and branches, which represent decisions and their possible outcomes. Root nodes and internal nodes correspond to features, such as an employee's age, while branches represent the possible values or ranges for those features. Leaf nodes indicate class labels, which in our context would indicate whether an employee has resigned or not.

To illustrate the functionality of a DT model, we use an example of fruit classification. We start with a dataset that contains information about fruits, including their color and shape, along with labels indicating whether they are "Apple" or "Orange". The objective is to build a DT model using this dataset to classify new fruits based on their color and shape. Figure 3-1 showcases one potential structure for the DT model. To make predictions using this model, we start at the root node labeled "Color" and follow the branches based on the feature values of the new fruits. Eventually, we reach a leaf node representing either "Apple" or "Orange".

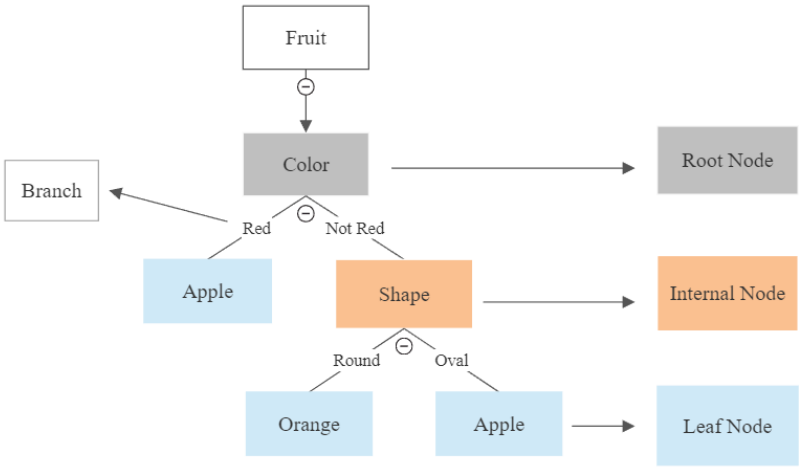


Figure 3-1 Decision Tree Structure for Fruit Classification

3.4.1 Gini Impurity

The detailed features of the DT model and its algorithm can be referred to Breiman et al. (2017) and Priyam et al. (2013). This section is dedicated to examining the splitting criterion of our final DT model. The splitting criterion determines the optimal feature and value for dividing the data at each node of the tree. The two most commonly used splitting criteria are Information Gain and Gini Impurity. In our thesis, the final DT model utilized Gini Impurity as the splitting criterion. It is calculated using the following formula:

$$Gini(S) = 1 - \sum(p(i)^2)\,, \tag{3-6}$$

where $p(i)$ represents the probability of class i appearing in the dataset S.

The resulting Gini Impurity reflects the degree of impurity or disorder within the dataset or subset. A Gini Impurity of 0 indicates a perfectly pure dataset, where all instances belong to the same class. Higher values indicate higher impurity, with 0.5 being the maximum impurity when classes are evenly distributed.

When considering a split on a specific feature, the Gini Impurity is calculated for each possible split point. The optimal split is chosen based on the split that minimizes the weighted average of the Gini Impurity for the resulting subsets. By using the Gini Impurity as the splitting criterion, the DT algorithm aims to create

splits that generate subsets with the least impurity, which leads to a more accurate and informative tree.

In summary, DT is a supervised learning algorithm that involves recursively partitioning the data based on the most informative features, ultimately creating a tree structure that can classify new instances. The selection of an appropriate splitting criterion is crucial in determining the optimal splits and overall accuracy and interpretability of the DT model.

3.5 Random Forest (RF)

Ensemble learning refers to algorithms that aggregate predictions from multiple models. Random Forest (RF) is an ensemble learning algorithm that combines multiple DTs to make predictions. RF can be applied to both classification and regression tasks, offering improved accuracy, reduced overfitting, and the ability to handle high-dimensional datasets with a large number of features (Breiman, 2001). Similar to DT, RF builds a collection of tree-like models. However, unlike a single DT, RF builds each individual tree by selecting a random subset of the dataset with replacement (known as bootstrap sampling) and using a random subset of features for nodes.

Building upon our fruit classification example, Figure 3-2 presents a potential structure for using a RF model to classify fruits. In this structure, when classifying a new fruit, each tree in the RF independently provides its prediction (e.g., “Apple” or “Orange”). The final prediction is then determined through majority voting, where the class that receives the most votes across all trees is chosen, resulting in a more reliable and accurate classification.

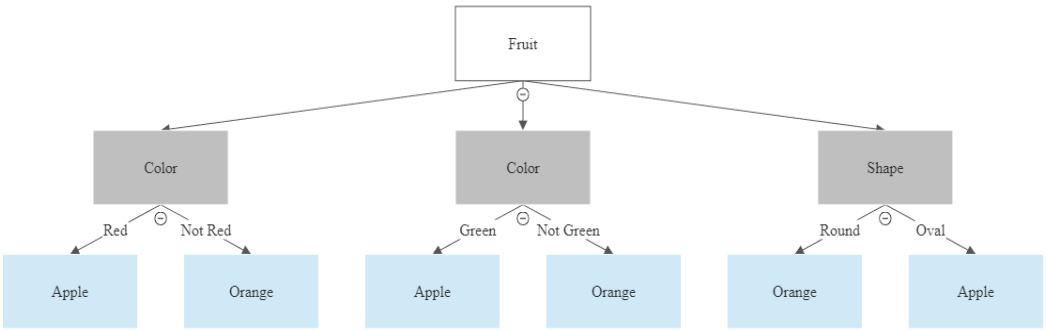


Figure 3-2 Random Forest Structure for Fruit Classification

3.5.1 Bootstrap Sampling

For the detailed features of the RF model and its algorithm, please refer to Breiman (2001) and Rodriguez-Galiano et al. (2012). This section delves into the examination of bootstrap sampling, which holds significant importance in the RF algorithm due to its crucial role in constructing individual DTs. It involves creating multiple resamples (bootstrap samples) from an original dataset by randomly selecting observations with replacement. The term "with replacement" indicates that each selected observation is returned to the dataset before the next selection, allowing the possibility of selecting the same observation more than once within a resample. The fundamental steps involved in bootstrap sampling are as follows:

1. Begin with an original dataset D, containing N observations.
2. Randomly select an observation from the original dataset and add it to a bootstrap sample.
3. Repeat Step 2 N times, with replacement, to create a bootstrap resample of the same size as the original dataset.
4. Repeat Steps 2 and 3 a total of B times to generate B bootstrap samples, denoted as D_1 , D_2 , …, D_B.
5. For each bootstrap sample, construct a decision tree using the DT algorithm.

By following these steps, we can generate multiple bootstrap samples from the original dataset and build each individual DT using each bootstrap sample, resulting in an ensemble of DTs. Additionally, for every bootstrap sample, a random subset of features is chosen from the available feature set. Typically, the subset size is smaller than the total number of features. These procedures introduce diversity among the DTs as each tree learns from a slightly different combination of data and features. Ultimately, this diversity aids in mitigating overfitting and improving the generalization capability of the RF model.

In summary, RF is an ensemble learning algorithm that combines multiple DTs to enhance prediction accuracy. By employing bootstrap sampling and random feature selection, RF ensures diversity among the individual trees, resulting in improved performance and robustness.

3.6 Gradient Boosting Decision Tree (GBDT)

Gradient Boosting Decision Tree (GBDT) is an ensemble learning algorithm that can be used for both classification and regression tasks (Friedman, 2000). GBDT aims to create a robust predictive model by iteratively combining weak learners, in this case, DTs, in a systematic manner. In comparison to RF, where trees are built independently, GBDT constructs trees in a sequential manner, with each subsequent tree designed to rectify the mistakes made by the previous trees. This iterative process enhances the model's predictive capability and reduces errors. Additionally, while both RF and GBDT utilize ensembles of DTs to make final predictions, RF relies on majority voting, whereas GBDT combines the predictions from all the trees.

The construction process of DTs in GBDT can be summarized in the following steps:

1. Begin with a single DT as the initial model.
2. Calculate the residuals or errors between the predictions of the current model and the true values of the target variable.
3. Construct a new DT specifically to predict the residuals, aiming to minimize the residuals and improve the overall model performance.
4. Update the model by adding the newly constructed DT to the ensemble, adjusting the predictions by a certain learning rate.
5. Repeat steps 2-4 until the desired number of trees is reached or the performance metric converges.

To provide a clearer understanding, Figure 3-3 presents a simplified example of using GBDT to predict a person's age. We begin with an initial DT that predicts an age of 20. Upon comparing this prediction with the true age, we calculate a residual of 10. To address this residual, a second DT is constructed specifically for predicting it, estimating a value of 6 and resulting in a residual of 4. This iterative process continues with the third tree predicting the new residual, and it persists until the fourth tree is built to handle the remaining residuals. To obtain the ultimate prediction, we sum up the predictions made by these four trees, resulting in a sum of 30, representing the final age prediction. This sequential approach of iteratively

correcting errors made by previous models allows GBDT to progressively improve its predictive power and achieve more accurate predictions.

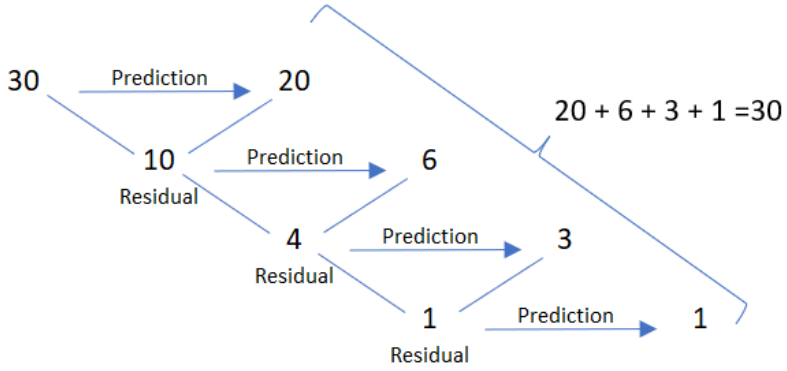


Figure 3-3 A Simplified GBDT Structure for Age Prediction

3.6.1 Loss Function

The detailed features of the GBDT model and its algorithm can be referred to Friedman (2000) and Ke et al. (2017). This section presents the loss function used in our final GBDT model. The Loss function is used to quantify and capture the discrepancy between the predicted values made by the GBDT model and the true values. It plays a crucial role in guiding the construction of subsequent DTs and optimizing the overall model performance. The choice of the loss function depends on the specific problem at hand, such as binary classification, multiclass classification, or regression. In our thesis, the final GBDT model incorporates deviance as the chosen loss function. It is calculated using the formula:

$$L(y, F(x)) = \log(1 + \exp(-2yF(x))), \tag{3-7}$$

where L is the loss function, y is the true output and $F(x)$ is the predicted output.

GBDT employs gradient descent to optimize the model. It utilizes the gradients of the loss function with respect to the predictions of the current model to guide the construction of subsequent trees. This approach ensures that each new tree is built in a direction that minimizes the loss function, leading to a more accurate and effective ensemble.

In conclusion, GBDT is a powerful ensemble learning algorithm that sequentially combines DTs to enhance prediction accuracy. Through iterative error correction, GBDT creates a strong ensemble capable of handling complex tasks.

3.7 Extreme Gradient Boosting (XGBoost)

Extreme Gradient Boosting (XGBoost) is an ensemble learning algorithm introduced by Chen and Guestrin (2016). It builds upon the principles of GBDT while incorporating several key enhancements to optimize computation efficiency and model performance. One important improvement is the normalization of the loss function, which reduces model variance and mitigates the risk of overfitting, resulting in improved stability and robustness. XGBoost also employs a sparsity-aware algorithm that efficiently handles attributes with a high occurrence of zero or missing value entries by excluding them from potential splits, enhancing overall algorithm efficiency. Additionally, XGBoost utilizes parallelizable learning to accelerate the computation of the best split, significantly reducing computational complexity and enabling faster model building without sacrificing ensemble accuracy.

3.7.1 Objective Function

For an in-depth understanding of the XGBoost model, Chen and Guestrin (2016) offer detailed insights into its algorithm. This section focuses on presenting the definition of the objective function in XGBoost, which distinguishes it from GBDT. The objective function in XGBoost, which can be derived from the loss function described in Equation 3-7, can be expressed as follows:

$$L(y, F(x)) + \Omega(F(x)) , \tag{3-8}$$

where $\Omega(F(x))$ represents the regularization term.

In comparison to GBDT, XGBoost incorporates the regularization term to control the complexity of the model. In the objective function, the loss function aims to ensure that the model fits the training data as closely as possible, the same as in GBDT. On the other hand, the regularization term promotes simpler models by penalizing complexity, reducing the impact of randomness when fitting the model with limited data. As a result, it mitigates the risk of overfitting and leads to more stable predictions from the model.

3.7.2 Taylor Expansion

Another key difference between XGBoost and GBDT is the use of a second-order Taylor expansion on the loss function. During each boosting iteration in XGBoost, Taylor expansion is used to approximate the loss function by using a polynomial expansion around a given point and making the optimization process more efficient. This allows for more efficient computation of the gradients and Hessians, which are used to update the model parameters during training.

Based on the loss function in Equation 3-8, the Taylor expansion of the loss function around a point $F(x_0)$ can be expressed mathematically as follows:

$$L(y,F(x)) \approx L\big(y,F(x_0)\big) + \big(F(x) - F(x_0)\big) * \frac{\partial L\big(y,F(x)\big)}{\partial F(x)} + \big(F(x) - F(x_0)\big)^2 * \frac{\frac{\partial^2 L(y,Fx))}{\partial F(x)^2}}{2!} + \cdots, \tag{3-9}$$

where $L(y,F(x))$ represents the loss function. The first term $L\big(y,F(x_0)\big)$ represents the loss at the point x_0. The second term $\big(F(x) - F(x_0)\big) \times \frac{\partial L\big(y,F(x)\big)}{\partial F(x)}$ represents the first-order derivative of the loss function with respect to $F(x)$ evaluated at x_0, multiplied by the difference between $F(x)$ and $F(x_0)$. The third term $\big(F(x) - F(x_0)\big)^2 \times \frac{\frac{\partial^2 L(y,Fx))}{\partial F(x)^2}}{2!}$ represents the second-order derivative of the loss function with respect to $F(x)$ evaluated at x_0, multiplied by the squared difference between $F(x)$ and $F(x_0)$, divided by 2! (which is 2 factorial).

By utilizing the Taylor expansion, XGBoost reduces the complexity of computing the loss function and its derivatives, which leads to faster training and improved efficiency. It is worth noting that the specific implementation details may vary between different versions of XGBoost, but the general idea of using Taylor expansion to approximate the loss function remains consistent.

In summary, XGBoost is a powerful ensemble learning algorithm that combines efficiency, scalability, regularization techniques, and an enhanced objective function to deliver superior classification accuracy.

3.8 K-fold Cross Validation (KCV)

K-fold cross validation (KCV) is a widely used technique for evaluating the performance of machine learning models (Anguita et al., 2012). In our analysis, we utilize KCV to select the best hyperparameters and assess the performance of the trained model. The process involves partitioning the dataset into k equal-sized folds, where k represents the desired number of folds, in our case, k is set to 10.

The steps of KCV can be summarized as follows:

1. Partitioning: The dataset is divided into k equal-sized folds, ensuring that each fold contains a representative subset of the data.
2. Training and Testing: The model is trained on $k - 1$ folds and evaluated on the remaining fold. This process is repeated k times, with each fold serving as the testing set exactly once.
3. Performance Metric: A performance metric, such as accuracy, precision, recall, or F1 score, is calculated for each iteration of the training and testing process.
4. Aggregation: The performance metrics obtained from each fold are averaged to provide an overall performance estimate of the model.

KCV is beneficial for model evaluation as it mitigates the risk of overfitting and provides a robust estimate of the model's performance on unseen data. It allows us to assess the model's ability to generalize across different subsets of the dataset and select the best hyperparameters based on the aggregated performance metrics. In our analysis, KCV serves as a valuable tool for model assessment and hyperparameter tuning, contributing to the overall reliability and validity of our results.

3.9 Confusion Matrix

A confusion matrix is a tabular representation that summarizes the performance of a classification model by showing the counts of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions (Deng et al., 2016). It provides valuable insights into the accuracy and error types of the model's predictions.

The structure of a confusion matrix is as follows:

Table 3-1. Confusion Matrix

Actual \ Predicted	**Negative**	**Positive**
Negative	True Negative (TN)	False Positive (FP)
Positive	False Negative (FN)	True Positive (TP)

*TP: Instances where the model correctly predicted the positive class.
*FN: Instances where the model incorrectly predicted the negative class, but they were actually of the positive class.
*FP: Instances where the model incorrectly predicted the positive class, but they were actually of the negative class.
*TN: Instances where the model correctly predicted the negative class.

The sum of TP, TN, FP, and FN represents the total number of instances in the dataset.

3.10 Relevant Metrics

The confusion matrix provides several performance metrics that can be derived to evaluate the model's performance, including accuracy, precision, recall, and F1 score. These metrics help assess the model's ability to correctly classify instances and identify potential imbalances or biases in the predictions.

3.10.1 Accuracy

Accuracy evaluates the overall correctness of a classification model. It represents the proportion of correct predictions out of the total number of predictions made by the model. The formula to calculate accuracy from a confusion matrix is:

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN} \quad (3\text{-}10)$$

Accuracy ranges from 0 to 1, where 1 indicates a perfect classification with no errors, and 0 indicates complete misclassification. Accuracy provides an overall measure of how well the model is able to classify instances correctly. However, it may not be suitable for imbalanced datasets, where the number of instances in different classes is significantly different. In such cases, accuracy alone may be misleading (Boughorbel et al., 2017; Jeni et al., 2013).

3.10.2 Precision

Precision focuses on the accuracy of positive predictions made by a classification model. It quantifies the ratio of TP predictions to the total number of positive predictions generated by the model. The formula to calculate precision is:

$$Precision = \frac{TP}{TP+FP} \tag{3-11}$$

Precision ranges from 0 to 1, where 1 indicates perfect precision with no FP predictions and 0 indicates complete misclassification of positive instances. Precision is particularly useful in scenarios where the cost of FP is high. It indicates how well the model is able to identify the TP instances while minimizing FP. A high precision value indicates a low rate of FP and a high level of confidence in the positive predictions made by the model.

3.10.3 Recall

Recall measures the proportion of TP predictions out of all actual positive instances in a classification problem. It quantifies the ability of a model to correctly identify positive instances. The formula to calculate recall from a confusion matrix is:

$$Recall = \frac{TP}{TP+FN} \tag{3-12}$$

Recall ranges from 0 to 1, where 1 indicates a perfect recall with no FN predictions and 0 indicates complete misclassification of positive instances. Recall is particularly important in scenarios where the cost of FN is high. It indicates how well the model captures all positive instances and minimizes FN. A high recall indicates a low rate of FN and a high level of sensitivity in detecting positive instances.

3.10.4 F1 Score

F1 score combines both precision and recall into a balanced measure of a model's performance. It provides a harmonic mean of precision and recall, giving equal importance to both metrics. The formula to calculate F1 score from precision and recall is:

$$F1 = 2 * \frac{Precision*Recall}{Precision+Recall} \tag{3-13}$$

F1 score ranges from 0 to 1, where 1 indicates the best possible performance and 0 indicates the worst. When F1 score is high, it indicates that the model has achieved a good trade-off between precision and recall. F1 score is particularly useful in scenarios where there is an imbalance between positive and negative instances in the dataset (Cahyana et al., 2019). It provides a way to assess a model's ability to achieve both high precision (minimizing FP) and high recall (minimizing FN).

3.10.5 Area Under the Receiver Operating Characteristic Curve (ROC-AUC)

Area Under the Receiver Operating Characteristic Curve (ROC-AUC) is a performance metric used to evaluate the predictive power of a binary classification model.

The ROC curve is a graphical representation of the model's performance by plotting the true positive rate (TPR) on the y-axis against the false positive rate (FPR) on the x-axis at various classification thresholds. The curve illustrates how well the model can distinguish between the positive and negative classes across different threshold settings.

The AUC refers to the area under the ROC curve. It assesses the overall performance of the model by quantifying the probability that the model will assign a higher rank to a randomly selected positive instance compared to a randomly selected negative instance.

The AUC value ranges from 0 to 1, where a value of 1 indicates a perfect classifier and a value of 0.5 suggests a random classifier (no better than random).

TPR, also known as recall, is calculated with reference to Equation 3-12. FPR is calculated as:

$$FPR = \frac{FP}{FP+TN} \tag{3-14}$$

A higher ROC-AUC score indicates that the model has a stronger ability to accurately differentiate between classes. It is particularly useful when dealing with imbalanced datasets, where the distribution of positive and negative instances is unequal (Miao & Zhu, 2020). Figure 3-4 shows an example of the ROC-AUC plot derived from the final XGBoost model of our analysis.

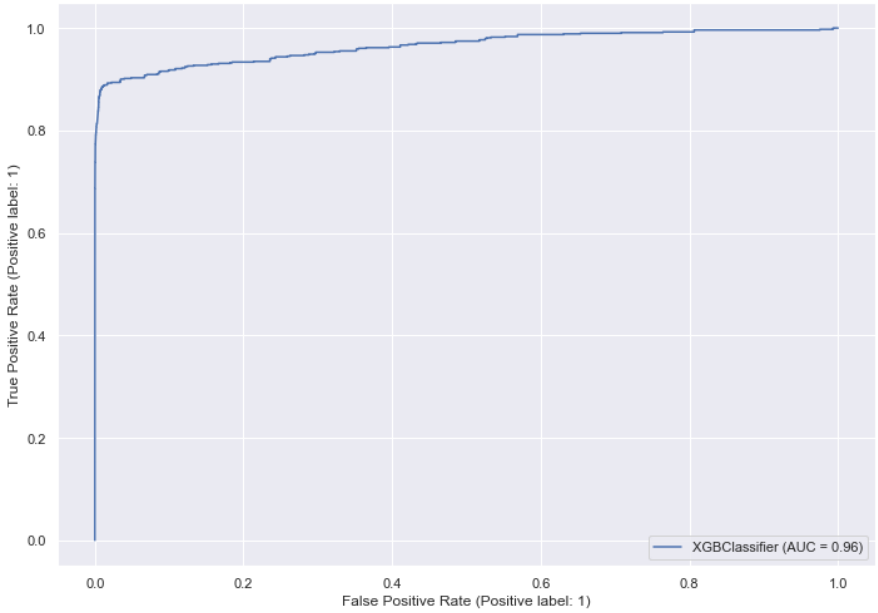


Figure 3-4 ROC-AUC Plot of the Final XGBoost Model

4. Data

In this chapter, we focus on the data used in our analysis and the various steps involved in preparing it for analysis. We begin by discussing the data source and structure (4.1). Next, we address the crucial process of data cleaning (4.2). This step involves identifying and handling missing values and inconsistencies in the dataset to ensure its quality and reliability. To gain a better understanding of the dataset, we conduct descriptive analysis (4.3), which involves summarizing and visualizing key characteristics and patterns present in the data. Subsequently, we move on to data preprocessing (4.4). This stage includes several steps such as categorical data encoding (4.4.1) to transform categorical features into numerical representations suitable for different machine learning algorithms. We also discuss data splitting (4.4.2), which involves dividing the dataset into training, validation, and testing sets to evaluate model performance accurately. Feature selection plays a crucial role in building effective predictive models. We explore different techniques for feature selection (4.5), including Analysis of Variance (ANOVA) (4.5.1), Chi-square test (4.5.2), and correlation analysis (4.5.3). These methods help identify the most relevant features that contribute to predicting employee turnover. We summarize the findings of the feature selection process (4.5.4), highlighting the selected features for further analysis. By thoroughly examining the data source, cleaning and preprocessing the dataset, and performing feature selection, we ensure the data's quality and suitability for developing accurate and reliable models for predicting employee turnover. These steps lay the foundation for the subsequent analysis and modeling stages of our work.

4.1 Data Source and Structure

The dataset used in the analysis is provided by a multinational corporation headquartered in Oslo, Norway. The company has over 10,000 employees in more than 100 countries worldwide. The dataset consists of two files: one file contains monthly employee demographic information spanning from January 2021 to June 2022, while the second file contains a list of employees who left the company between January 2021 and September 2022, including their departure dates and reasons for leaving. It is important to note that the data specifically focuses on permanent employees of the company, excluding temporary employees who have predetermined end dates for their employment. Additionally, to ensure a more

focused analysis, the dataset only includes employee data from the top 10 countries with the highest number of employees in the company. By narrowing the scope to these countries, which include China, Germany, India, Italy, Netherlands, Norway, Poland, Spain, United Kingdom, and the United States, the analysis concentrates on regions with substantial employee representation. For a comprehensive overview of the dataset's structure and content, Table 4-1 presents a detailed list of all columns included in both data files, accompanied by their respective descriptions. Furthermore, to provide a glimpse of the dataset's contents, examples of rows from the employee demographics file and the termination file can be found in Table 4-2 and Table 4-3, respectively.

Table 4-1. Description of the Dataset Used

File Name	Feature	Feature Description
Employee Demographics File	Employee ID	The personal identification number for each employee.
	Age	The age of the employee.
	Country	The country where the employee works. The countries are China, Germany, India, Italy, the Netherlands, Norway, Poland, Spain, the United Kingdom and the United States.
	Extraction date	The data extraction date. The data extraction is performed on a monthly basis from January 2021 to June 2022.
	Education	The educational background of the employees. The educations are 2 year College Level, Bachelor Level, Basic Education Level, Doctorate Level, Master Level, Not Applicable, Professional / Technical 0-3 years and Unknown.
	Gender	The gender of the employee. The genders are Female, Male and Unknown.
	Hire date	The date the employee was hired.
	Industry	The industry the employee works in. The industries are Consultancy, Cyber Security, Data Management, Energy, External, Human Resource, Information Technology, Insurance, Maritime and Supply Chain.
	Last date of promotion	The date the employee was last promoted on the Level.
	Level	The job level of the employee. The level ranges from 1 to 15, with the additional category of Unknown.
	Tenure	The length of time the employee has worked for the company, measured in years.
	Title	The job title of the employee. The titles are Auditor, Consultant, Engineer, Manager, No Title, Specialist, Support, Surveyor, Technical Assistant, Vice President and Unknown.
Employee Termination File	Employee ID	The personal identification number for each employee.
	Termination date	The date when the employee's resignation is registered. From January 2021 to September 2022.
	Termination type	The type of the employee's resignation. The types are Voluntary, Involuntary and Unknown.
	Termination reason	The reason for the employee's resignation. The reasons are Death, Dismiss, Illness, Involuntary, Transfer, Voluntary and Unknown.

Table 4-2. Sample Rows from the Employee Demographics File

Extraction date	Employee ID	Industry	Gender	Age	Tenure	Education	Title	Level	Last date of promotion	Country	Hire date
2021-01-01 00:00:00	xxx	Consultancy	Male	49	8	Master Level	Engineer	10	2019-04-01 00:00:00	Poland	2012-10-01 00:00:00
2021-01-01 00:00:00	xxx	Energy	Male	44	0	Master Level	Engineer	8	2020-09-01 00:00:00	Poland	2020-09-01 00:00:00
2021-01-01 00:00:00	xxx	Energy	Male	31	0	Master Level	Engineer	8	2020-11-01 00:00:00	Poland	2020-11-02 00:00:00
2021-02-01 00:00:00	xxx	Insurance	Female	49	26	Basic Education Level	Manager	6	2011-01-01 00:00:00	Italy	1994-10-01 00:00:00
2021-02-01 00:00:00	xxx	Insurance	Male	55	21	Master Level	Auditor	8	2012-01-01 00:00:00	Italy	1999-05-14 00:00:00
2021-02-01 00:00:00	xxx	Insurance	Female	56	24	Bachelor Level	Support	5	2018-04-01 00:00:00	Italy	1997-06-01 00:00:00

Table 4-3. Sample Rows from the Employee Termination File

Employee ID	Termination date	Termination type	Termination reason
xxx	2022-07-31 00:00:00	Voluntary	Voluntary
xxx	2022-01-31 00:00:00	Voluntary	Voluntary
xxx	2021-02-28 00:00:00	Involuntary	Transfer
xxx	2021-01-31 00:00:00	Voluntary	Voluntary
xxx	2022-03-31 00:00:00	Voluntary	Voluntary
xxx	2021-09-30 00:00:00	Involuntary	Transfer

4.2 Data Cleaning

In order to prepare the dataset for model training, we conducted several rounds of data cleaning.

Firstly, we merged the two files based on the common column, Employee ID. After merging, employees with recorded dates of resignation were identified as having left the company. The prediction period of our model was set at three months. Therefore, employees who had resigned within three months of the data extraction date were labeled as "resigned" (positive), while others were labeled as "not resigned" (negative). We determined the three-month prediction period in consultation with the company that provided the data. They highlighted that if the prediction period was too short, such as one month, the company would not have sufficient time to take action even if they knew they were going to lose employees. On the other hand, we discovered that if the prediction period was too long, it would adversely affect the model's performance. Overall, the three-month prediction period provides companies with an early warning of employee turnover and allows them enough time to take preventative action.

Secondly, we cleaned the merged file by column:

Employee ID: Given that the employee identifier is unique and does not contribute meaningful information to the prediction task, we have made the decision to exclude this column. This simplification allows us to focus on relevant features that have a more direct impact on the prediction outcome.

Exaction date: Considering that this feature merely indicates the date when the data was collected from the HR system. It is unrelated to actual resignation outcomes and does not provide direct insights into an employee's decision-making process. We have made the decision to exclude this column.

Age: The initial range of values in this column extended from -1 to 86, suggesting potential errors in data entry. Taking into account variations in retirement age across different countries, we have chosen to narrow down the range and only include employee ages between 20 and 70. This adjustment, which removed 0.25% of the data, ensures a more realistic representation of the dataset while accounting for the typical age demographics of employees.

Education: Around 8.00% of employees in our dataset have been labeled as "Not Applicable" or "Unknown" in this column, amounting to a total of 12,693 rows. Removing these rows would result in a reduction in the overall size of our dataset, which is not ideal for maintaining an adequate sample size for analysis and modeling purposes. As a result, we have concluded that excluding the "Education" column is a more favorable approach in order to ensure the integrity and reliability of our dataset.

Gender: This column contained entries labeled as "Unknown", which constituted approximately 0.04% of the data. To maintain the dataset's accuracy and reliability, we have made the decision to exclude the "Unknown" entries from our analysis.

Hire date & Tenure: After careful consideration, we have opted to remove the "Hire date" column from our dataset. This decision is made based on the fact that the information conveyed by the "Hire date" column is essentially redundant with the "Tenure" column. By removing the "Hire date" column, we can streamline our analysis and maintain data consistency by focusing on a single column to capture employee tenure information.

Industry: Although the dataset primarily focuses on registered employees within the company, it contains data on external individuals working for the company, referred to as "External". These external employees comprise 0.04% of the dataset. We have excluded the external employees' data from our analysis.

Last date of promotion: During the analysis of this column, we discovered that approximately 2.37% of the entries were missing. Upon further examination, it was determined that these blank entries were likely erroneous. Even employees who have never been promoted have their hire date recorded as the last promotion date. In order to maintain the completeness and consistency of our dataset, we have made the decision to remove these blank entries.

Level: The entry labeled as “Unknown” was identified as an error in this column. It is evident that these errors were introduced during the data input process. Consequently, we have removed these entries which accounted for 0.11% of the employees in the dataset.

Title: In this column, we have excluded the entries labeled as “No Title” and “Unknown”. These values accounted for only 0.06% of the dataset.

Termination date, Termination type & Termination reason: In alignment with our analysis focus on voluntary turnover prediction, we have exclusively included voluntary resignations in the dataset. Consequently, termination types such as dismissals, transfers, illnesses, and deaths, which fall under the category of involuntary reasons for leaving, have been deliberately excluded. Additionally, to avoid target leakage in our analysis, we have made the decision not to include the “Termination date”, “Termination type” and “Termination reason” columns in our dataset. Given that our objective is to predict an employee’s decision to leave the company, including these columns would introduce information that becomes available only after an employee has already left.

After completing the data cleaning process, the dataset contains a total of 146,885 data points. Among these, 3,912 instances are labeled as positive, representing approximately 2.66% of the dataset. These positive instances indicate individuals who have resigned within a three-month time frame. The dataset includes eight features: Age, Country, Gender, Industry, Last date of promotion, Level, Tenure, and Title. These features provide valuable information about the individuals in the dataset and can be used for further analysis and modeling. Additionally, there is one target column, “Resigned”, which serves as an indicator variable, indicating whether an employee has resigned or not within the specified time period.

4.3 Descriptive Analysis

The dataset used in our analysis is thoroughly examined through the following graphs, offering a comprehensive descriptive analysis. Figure 4-1 illustrates the complete dataset encompassing all employees, providing insights into various aspects. On the other hand, Figure 4-2 focuses specifically on the data of employees who have resigned, allowing for a more targeted examination of this subgroup.

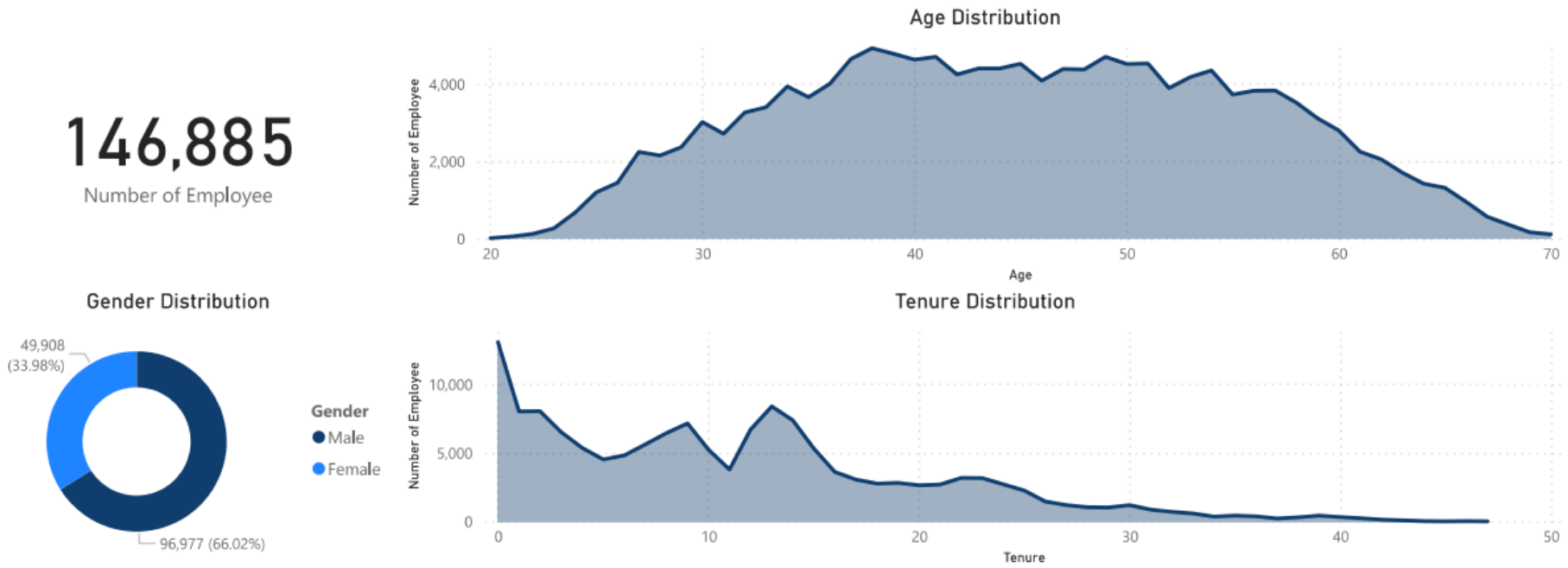


Figure 4-1 Descriptive Analysis of All Employees

Country	Number of Employee	Percentage of Total
Norway	33,896	23.08%
United States	23,950	16.31%
Germany	19,941	13.58%
United Kingdom	15,737	10.71%
China	13,807	9.40%
Poland	12,126	8.26%
Netherlands	8,642	5.88%
India	7,006	4.77%
Italy	6,083	4.14%
Spain	5,697	3.88%
Total	**146,885**	**100.00%**

Industry	Number of Employee	Percentage of Total
Energy	50,124	34.12%
Maritime	39,108	26.62%
Information Technology	17,813	12.13%
Insurance	16,144	10.99%
Consultancy	10,289	7.00%
Supply Chain	5,744	3.91%
Human Resource	3,548	2.42%
Cyber Security	2,412	1.64%
Data Management	1,703	1.16%
Total	**146,885**	**100.00%**

Title	Number of Employee	Percentage of Total
Engineer	46,777	31.85%
Support	32,649	22.23%
Manager	27,792	18.92%
Consultant	15,271	10.40%
Surveyor	12,248	8.34%
Auditor	7,264	4.95%
Specialist	2,702	1.84%
Technical Assistant	1,661	1.13%
Vice President	521	0.35%
Total	**146,885**	**100.00%**

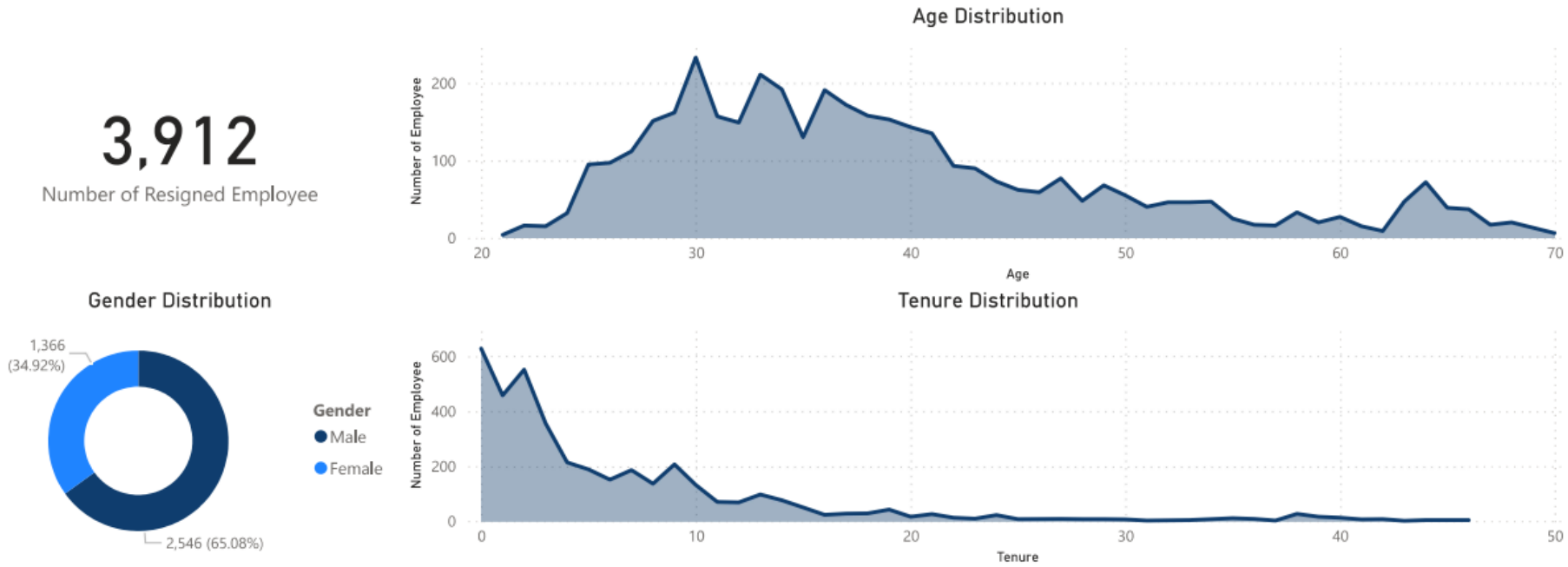


Figure 4-2 Descriptive Analysis of Resigned Employees

Country	Number of Employee	Percentage of Total
United States	980	25.05%
Norway	917	23.44%
Poland	464	11.86%
United Kingdom	394	10.07%
India	257	6.57%
Germany	254	6.49%
Spain	239	6.11%
Netherlands	191	4.88%
China	141	3.60%
Italy	75	1.92%
Total	**3,912**	**100.00%**

Industry	Number of Employee	Percentage of Total
Energy	1,776	45.40%
Information Technology	632	16.16%
Maritime	503	12.86%
Insurance	361	9.23%
Consultancy	252	6.44%
Supply Chain	183	4.68%
Human Resource	93	2.38%
Data Management	64	1.64%
Cyber Security	48	1.23%
Total	**3,912**	**100.00%**

Title	Number of Employee	Percentage of Total
Engineer	1,421	36.32%
Support	1,004	25.66%
Consultant	678	17.33%
Manager	427	10.92%
Auditor	165	4.22%
Surveyor	127	3.25%
Technical Assistant	52	1.33%
Specialist	34	0.87%
Vice President	4	0.10%
Total	**3,912**	**100.00%**

4.4 Data Preprocessing

4.4.1 Categorical Feature Encoding

In order to ensure that the same dataset can be used for training different machine learning models, we performed categorical feature encoding during the data preprocessing stage. We encoded the categorical features using the "LabelEncoder" function from the "sklearn.preprocessing" module in Python. The code used for feature encoding can be found in Appendix 7.1. This process allows us to convert categorical data into a numeric format, as the DT and RF algorithms in the scikit-learn library for Python do not support categorical variables (Pedregosa et al., 2011). By applying the LabelEncoder function, we transformed each unique category within categorical features, including Industry, Gender, Title, and Country, into a corresponding unique integer. This encoding procedure guarantees that the data is in a suitable format that can be effectively utilized by various machine learning models for both training and analysis purposes.

4.4.2 Data Splitting

To ensure a robust evaluation process for model performance, we conducted data splitting during the data preprocessing stage using the "train_test_split" function from the "sklearn.model_selection" module. The code used for data splitting can be found in Appendix 7.2. The dataset is randomly divided into three subsets: the training set, the validation set, and the testing set. The training set is allocated 60% of the total data, while both the validation and testing sets accounted for 20% each. Additionally, the proportion of positive instances in all three subsets remains consistent at 2.66%, which is the same as the proportion in the original dataset. This stratified splitting ensures a balanced representation of the target variable across the subsets. By splitting the data into distinct sets, we can avoid using the validation and testing sets for feature selection and model training. This ensures that the model is evaluated on previously unseen instances, minimizing the risk of biased evaluation and information leakage. Moreover, having a separate validation set allows us to fine-tune the model's hyperparameters without compromising the integrity of the results. Data splitting plays a crucial role in providing a more realistic estimation of the model's generalization capabilities and enhances the

overall reliability of our analysis. Table 4-4 presents the distribution of positive and negative instances across different subsets after data splitting.

Table 4-4. Resigned vs. Unresigned Individuals in Different Subsets

Training Set	Resigned	2,347
	Unresigned	85,784
Validation Set	Resigned	782
	Unresigned	28,595
Testing Set	Resigned	782
	Unresigned	28,595

4.5 Feature Selection

In order to train our machine learning model with the most significant and influential features from the dataset, we applied three widely recognized (Chandrashekar & Sahin, 2014; J. Li et al., 2017; Khalid et al., 2014) feature selection techniques: Analysis of Variance (ANOVA), Chi-square test and correlation analysis. Each technique was selected based on its specific strengths and suitability for our classification task. In this chapter, we examine the outcomes of each feature selection test and discuss their implications for our analysis. The detailed step-by-step process for each feature selection test can be found in Appendix 7.3.

4.5.1 ANOVA for Numerical Feature Selection

Table 4-5. ANOVA Test Results: Feature Significance at 5% Level

Feature	Statistically Significant at 5% Level?
Tenure	TRUE
Age	TRUE
Level	TRUE
Last date of promotion	TRUE

* Features are sorted in ascending order according to their p-value.

Table 4-5 presents the result of the ANOVA, which examined the statistical significance of each numerical feature at a 5% significance level using p-values. The analysis revealed that all the analyzed features, namely Age, Tenure, Last date of promotion, and Level, demonstrated statistical significance. This indicates that these features have a substantial impact on the target variable and possess valuable predictive power for employee turnover. These findings align with the descriptive analysis presented in Figure 4-2. For instance, the Age feature suggests that younger employees are more likely to resign, while the Tenure feature indicates that employees with shorter tenures have a higher probability of leaving their jobs. In

summary, the ANOVA reveals a strong relationship between employee turnover and all the numerical features in the dataset, including Age, Tenure, Last date of promotion, and Level.

4.5.2 Chi-square Test for Categorical Feature Selection

Table 4-6. Chi-square Test Results: Feature Significance at 5% Level

Feature	Statistically Significant at 5% Level?
Country	TRUE
Industry	TRUE
Title	TRUE
Gender	FALSE

* Features are sorted in ascending order according to their p-value.

Table 4-6 presents the results of the Chi-square test conducted for each categorical feature at a 5% significance level. The results indicate that Country, Industry, and Title show a strong association with the target variable, suggesting that these features have a significant impact on employee turnover. On the other hand, the Chi-square test reveals that the Gender feature has a limited impact on the target variable, suggesting that it may not be a strong predictor of employee turnover. The lack of significance for gender could be attributed to several possible reasons. One reason could be the dataset itself may not capture all the relevant aspects related to gender and employee turnover. Factors such as gender bias or gender-related disparities in the workplace may not be adequately represented in the available data, leading to a limited impact of gender on the prediction of employee turnover. Additionally, it is important to consider the nature of the specific industry or organizational context. Certain industries or workplaces may have a more gender-neutral or inclusive culture, where gender may have a limited influence on employee turnover compared to other factors. In conclusion, the Chi-square test results indicate that Country, Industry, and Title have a significant association with the target variable, suggesting that they play a crucial role in predicting employee turnover. However, the Gender feature shows a limited impact on the target variable, indicating that it may not be a strong predictor. It is important to recognize the contextual factors and potential variations in different datasets or organizational settings. The significance of gender as a predictor of employee turnover can vary depending on the industry, workplace culture, and specific characteristics of the dataset.

4.5.3 Correlation Analysis for Numerical Feature Selection

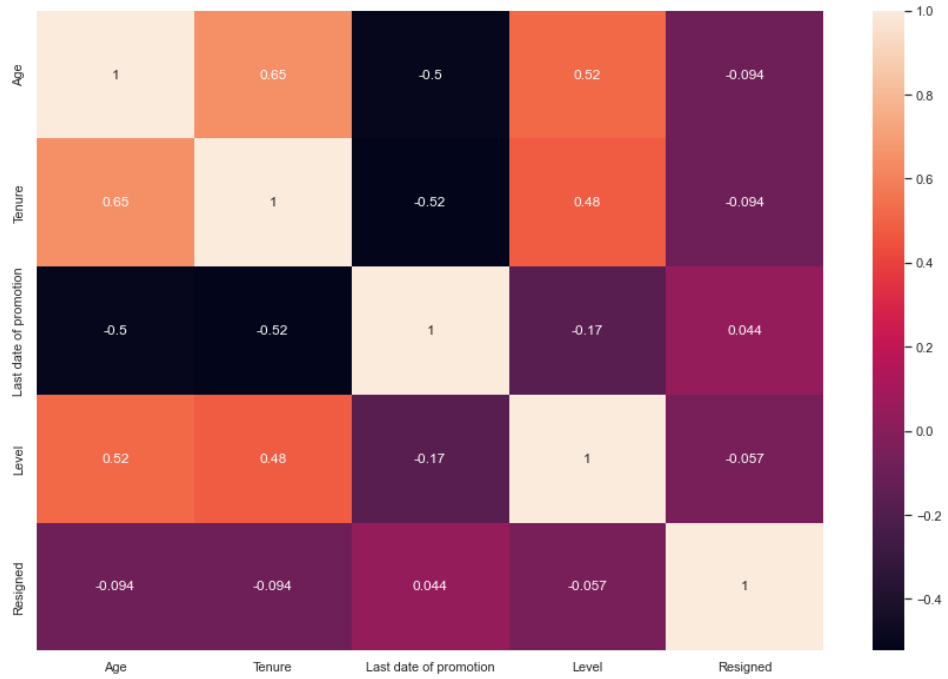


Figure 4-3 Correlation Heatmap of Numerical Features and the Target Variable

Figure 4-3 shows the correlation heatmap among all the numerical features and the target variable (Resigned). The overall low correlation between the target variable and the numerical features indicates the challenge of accurately predicting employee turnover based solely on a single variable. In addition, Age and Tenure have the highest correlation among the features themselves, with a correlation coefficient of 0.65. This is reasonable, considering that employees with longer tenures within a company are typically older on average. It is worth noting that the features generally demonstrate a low degree of correlation with each other, suggesting that each feature carries unique information and contributes independently to the prediction of employee turnover.

4.5.4 Summary

In summary, the application of ANOVA, Chi-square test, and correlation analysis revealed that Age, Tenure, Last date of promotion, Country, Level, Industry, and Title are identified as key features exhibiting a strong relationship with the target variable, Resigned. However, Gender did not demonstrate statistical significance in the Chi-square test. As a result, we made the decision to remove Gender from the dataset. However, it is important to acknowledge that the significance of Gender may vary across different contexts or datasets, and the decision to exclude it should be evaluated in consideration of the specific industry, workplace culture, and

available data. Additionally, it is important to recognize the potential biases in the dataset such as the imbalanced class distribution, where the number of employees who resigned is significantly smaller than those who did not. This class imbalance can impact the performance of the feature selection tests and should be taken into account. Overall, while the feature selection techniques provided valuable insights, it is crucial to interpret the results with caution and account for the limitations and biases inherent in the dataset.

5. Analysis

In this chapter, our focus is on the training process of the machine learning models employed in this study, namely Decision Tree (DT) (5.1), Random Forest (RF) (5.2), Gradient Boosting Decision Tree (GBDT) (5.3), and Extreme Gradient Boosting (XGBoost) (5.4). For each model, we begin by examining its performance without any hyperparameter tuning, establishing a baseline for comparison. Subsequently, we delve into the hyperparameter tuning process, providing a practical example to illustrate the methodology (5.1.2). Finally, we evaluate the model's performance after the hyperparameter tuning phase, allowing us to assess the effectiveness of the optimized configurations.

5.1 Decision Tree (DT)

In this section, we focus on the training of the Decision Tree (DT) model, which serves as the foundation for the other three models.

5.1.1 Model Performance Without Hyperparameter Tuning

We begin by evaluating the performance of the DT model without any hyperparameter tuning using the training set. This initial step allows us to explore the model's compatibility with the dataset and gain insights into its predictive capabilities. As shown in Table 5-1, the DT model demonstrates excellent performance on the training set, achieving an F1 score of 0.98. This high F1 score indicates a strong ability of the model to correctly classify the target variable based on the training data. Furthermore, the model maintains a commendable performance on the validation set, with an F1 score of 0.90. This suggests that the model generalizes well to unseen data, reinforcing its effectiveness in predicting employee turnover. It is important to note that the extremely high accuracy values obtained for both the training and validation sets can be misleading. The dataset is highly imbalanced, so accuracy alone is not the most appropriate metric for evaluating model performance in this case. Overall, the DT model without any hyperparameter tuning demonstrates a reasonable ability to classify employees into their respective turnover categories. Given these initial results, it is evident that the DT model has the potential to be a valuable tool for identifying employees at risk of turnover. However, further improvements can be made through hyperparameter tuning, which is discussed in the following sections.

Table 5-1. DT Model Score (Default Hyperparameter)

	Training Set	Validation Set
Accuracy	1	0.99
Precision	0.99	0.92
Recall	0.97	0.89
F1 score	0.98	0.90

5.1.2 Hyperparameter Tuning Process (with example)

Hyperparameter tuning involves adjusting the model's parameters to find the optimal configuration that improves its performance and generalization ability. By fine-tuning the hyperparameters, we aim to strike a balance between model complexity and generalization ability. The classification DT model has a total of 12 parameters that can be adjusted (*sklearn.tree.DecisionTreeClassifier*, n.d.). After a thorough examination, we categorize these parameters into three categories based on their significance and impact on the model's performance. The three categories are as follows:

1. Hyperparameters for model tuning:
    - criterion: This parameter allows us to choose between "entropy" or "gini" as the measure of the importance of features in splitting the nodes of the tree.
    - max_depth: It limits the maximum depth of the DT. Considering the large sample size in our analysis, we choose to limit the maximum depth to prevent overfitting.
    - max_features: It controls the number of features that are considered when looking for the best split at each node of the tree. By tuning it, we can influence the randomness and diversity of the feature selection process.
    - min_samples_split: This parameter sets the minimum number of samples required in a node for it to be considered for further splitting. The default value is 2, meaning that a node will only continue to split if it contains more than 2 samples.
    - min_samples_leaf: When the number of samples assigned to a leaf node is less than the set number, the leaf node will be pruned. This can help remove some obvious noise data.
2. Hyperparameters for handling an unbalanced dataset:

- class_weight: This parameter is used to assign different weights to positive and negative samples when the dataset is highly imbalanced. In our case, where the dataset is highly imbalanced, we have adjusted this parameter to account for the unequal distribution of classes.

3. Other hyperparameters.

   For our analysis, the remaining parameters do not require manual adjustment, as they typically do not significantly impact the model's performance.

To fine-tune these hyperparameters, we employ the K-fold Cross Validation (KCV) and grid search to explore different combinations of hyperparameter values and identify the best-performing configuration. In Appendix 7.4, we provide a detailed description of the hyperparameter tuning process for the DT model, including the range of values considered for each hyperparameter and the evaluation metrics used to assess the performance of different parameter configurations. Here, we will demonstrate the process of tuning the "max_depth" parameter as an example. The "max_depth" parameter determines the maximum depth of the DT, which controls the complexity of the model.

First to narrow down the range of values for "max_depth", we use KCV with 5 folds and the F1 score as the scoring function. We examine the range starting from 10 to 100, with increments of 10. Figure 5-1 shows the F1 score as a function of different depths. The result shows that the F1 score reaches its peak when the depth is around 40. Based on this finding, we further narrow our search range to focus on values between 30 and 50. Figure 5-2 shows the F1 score as a function of different "max_depth" values within this range. It appears that the F1 score initially increases with the increase in the depth and reaches its highest value at a "max_depth" of 35, with a score of 0.78 on the training set. This indicates that setting the tree depth to 35 achieves the optimal balance between capturing relevant information from the data and preventing overfitting. Using the value of 35 for the "max_depth" parameter, we can move on to adjust the other parameters to further optimize the model's performance.

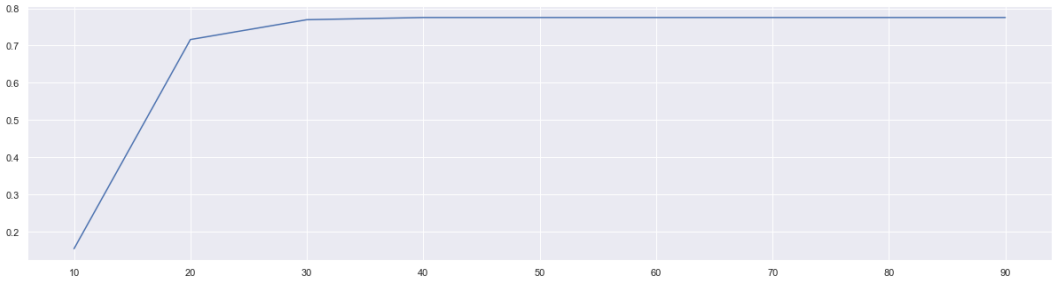


Figure 5-1 Cross-Validation Search of «max_depth» (Range 10 to 100)

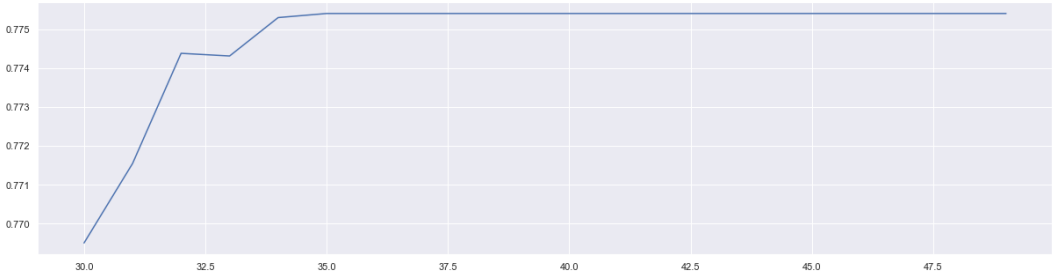


Figure 5-2 Cross-Validation Search of «max_depth» (Range 30 to 50)

After conducting hyperparameter tuning for the DT model, we obtain the final hyperparameter configuration, which is listed in Table 5-2.

Table 5-2. Hyperparameter Configuration for the DT Model

Hyperparameter	Value
class_weight	None
ccp_alpha	0.0
criterion	gini
max_depth	35
max_features	2
max_leaf_nodes	None
min_impurity_decrease	0.0
min_samples_leaf	1
min_samples_split	2
min_weight_fraction_leaf	0.0
random_state	42
spliter	best

5.1.3 Model Performance After Hyperparameter Tuning

To evaluate the performance of the final DT model, we first conduct KCV on the training set to assess its stability. The detailed KCV process can be found in Appendix 7.5. The average F1 score on the training set is 0.79, indicating a reasonably good performance in classifying the training data. Additionally, the average ROC-AUC score is 0.86, suggesting a relatively robust performance. To further evaluate the model, we compare its performance on the testing set before and after hyperparameter tuning. Table 5-3 presents the model score before and after tuning. The precision of the model increases from 0.92 to 0.95, indicating a

higher accuracy in predicting positive instances. The recall remains the same at 0.89, and the F1 score increases by 0.02. To gain more insight, we examine the confusion matrix, which provides a detailed breakdown of the model's predictions. As shown in Table 5-4, the model correctly predicts 14 additional instances (TN+TF) compared to the model without hyperparameter tuning. Additionally, the number of FP predictions decreases by 16 and the number of FN decreases by 7, indicating a reduction in misclassifications. Although the overall improvement in the performance metrics may appear subtle, a closer examination of the confusion matrix reveals that the model's predictions have become more accurate and aligned with the true labels.

Table 5-3. DT Model Score on Testing Set

	Default Hyperparameter	**Tunned Hyperarameter**
Accuracy	0.99	0.99
Precision	0.92	0.95
Recall	0.89	0.89
F1 score	0.90	0.92

Table 5-4. Confusion Matrix on Testing Set of the DT Model

Default Hyperparameter		**Predicted Label**	
		0 (Not Resigned)	1 (Resigned)
True Label	0 (Not Resigned)	28,519	76
	1 (Resigned)	176	606
Tunned Hyperarameter		**Predicted Label**	
		0 (Not Resigned)	1 (Resigned)
True Label	0 (Not Resigned)	28,526	60
	1 (Resigned)	169	613

In summary, the DT model proves to be highly suitable for predicting employee turnover using the available dataset. Through the process of hyperparameter tuning, the model demonstrates slightly improved performance, including higher accuracy in predicting positive instances and a reduction in misclassifications. These enhancements further enhance the model's effectiveness in identifying employees at risk of turnover.

5.2 Random Forest (RF)

In this section, we delve into the training of the Random Forest (RF) model, which builds upon the foundation of the DT model. By leveraging the power of ensemble learning, RF enhances the predictive performance and generalization ability compared to a single DT.

5.2.1 Model Performance Without Hyperparameter Tuning

To assess the compatibility of the RF model with the dataset, we initially evaluate the model performance using the default parameters. As shown in Table 5-5, the results are highly promising. The model achieves an F1 score of 0.98 on the training set and 0.92 on the validation set. These scores surpass the performance of the DT model, which is reasonable considering that RF is built upon DT. To explore the potential for further improvement, we proceed with hyperparameter tuning to optimize the RF model's performance.

Table 5-5. RF Model Score (Default Hyperparameter)

	Training Set	Validation Set
Accuracy	1	0.99
Precision	0.99	0.98
Recall	0.98	0.87
F1 score	0.98	0.92

5.2.2 Hyperparameter Tuning Process

Compared to the DT model, the RF model introduces 6 additional hyperparameters (*sklearn.ensemble.RandomForestClassifier*, n.d.). In the process of hyperparameter tuning, we have chosen to focus on tuning several specific hyperparameters that we found to be crucial for enhancing the model's performance. These hyperparameters include class_weight, criterion, max_depth, max_features, min_samples_split, and min_samples_leaf, which are introduced in the DT model. In addition, we also considered the following new hyperparameters introduced by the RF model:

n_estimators: This hyperparameter controls the number of trees in the forest. Increasing the number of trees typically leads to an improvement in the performance of the model. However, there comes a point where adding more trees no longer significantly increases accuracy, and the computational cost of training the model increases.

bootstrap: It determines whether to use bootstrap samples when building each tree in the random forest. Bootstrap sampling involves randomly sampling the training dataset with replacement, which can introduce diversity and improve the model's generalization ability.

oob_score: When bootstrap is set to True, this hyperparameter allows us to use out-of-bag data to evaluate the model's performance. Out-of-bag samples are data

points that are not included in the bootstrap sample used for training a particular tree.

The hyperparameter tuning process, including the range of values explored for each parameter and the evaluation metrics used to assess the performance of various parameter configurations, is provided in detail in Appendix 7.6. The final hyperparameter configuration of the RF model is listed in Table 5-6.

Table 5-6. Hyperparameter Configuration for the RF Model

Hyperparameter	Value
bootstrap	True
ccp_alpha	0.0
class_weight	None
criterion	gini
max_depth	31
max_features	auto
max_leaf_nodes	None
max_samples	None
min_impurity_decrease	0.0
min_samples_leaf	1
min_samples_split	2
min_weight_fraction_leaf	0.0
n_estimators	169.0
n_jobs	None
oob_score	False
random_state	42
verbose	0.0
warm_start	False

5.2.3 Model Performance After Hyperparameter Tuning

After conducting hyperparameter tuning, we first perform KCV to assess the stability of the final RF model. The detailed procedures of the KCV can be found in Appendix 7.7. The average F1 score and ROC-AUC on the training set are 0.81 and 0.98, respectively, indicating the model's stability and its ability to generalize well across different folds of the data. Subsequently, we evaluate the performance of the RF model on the testing set before and after hyperparameter tuning, as shown in Table 5-7. Comparing the results with the DT model, the RF model after hyperparameter tuning does not show a significant improvement. One possible explanation for this observation is that the RF model with default hyperparameters already demonstrates satisfactory performance, leaving limited room for further enhancement. The F1 score is 0.92, indicating a strong overall performance in classifying positive and negative instances. The model has a high precision of 0.98,

indicating a low rate of FP predictions, and a relatively low recall of 0.87, suggesting a slightly higher rate of FN predictions. This finding is consistent with the confusion matrix shown in Table 5-8, where the number of FP predictions is only 13, while the number of FN is 221.

Table 5-7. RF Model Score on Testing Set

	Default Hyperparameter	Tunned Hyperarameter
Accuracy	0.99	0.99
Precision	0.98	0.98
Recall	0.87	0.87
F1 score	0.92	0.92

Table 5-8. Confusion Matrix on Testing Set of the RF Model

Default Hyperparameter		**Predicted Label**	
		0 (Not Resigned)	1 (Resigned)
True Label	0 (Not Resigned)	28,582	13
	1 (Resigned)	776	6
Tunned Hyperarameter		**Predicted Label**	
		0 (Not Resigned)	1 (Resigned)
True Label	0 (Not Resigned)	28,582	13
	1 (Resigned)	221	561

In summary, the RF model proves to be highly effective for predicting employee turnover. Even without hyperparameter tuning, the model exhibits strong performance using the dataset at hand.

5.3 Gradient Boosting Decision Tree (GBDT)

In this section, we focus on the training of the Gradient Boosting Decision Tree (GBDT) model, which is a powerful ensemble learning model that improves upon the performance of a single DT.

5.3.1 Model Performance Without Hyperparameter Tuning

To assess the compatibility between the GBDT model and the dataset, we begin by evaluating the model's performance using the default hyperparameters. However, the obtained results, presented in Table 5-9, indicate subpar performance compared to the DT and RF models. Both the training set and validation set show an F1 score of 0.50, suggesting that the GBDT model faces challenges in accurately identifying positive instances. These findings highlight the need for further optimization and hyperparameter tuning to better adapt the model to the distinct characteristics of the dataset.

Table 5-9. GBDT Model Score (Default Hyperparameter)

	Training Set	Validation Set
Accuracy	0.97	0.97
Precision	0.97	0.88
Recall	0.51	0.50
F1 score	0.50	0.50

5.3.2 Hyperparameter Tuning Process

The classification GBDT model has a total of 20 hyperparameters that can be adjusted (*sklearn.tree.GradientBoostingClassifier*, n.d.). After a thorough evaluation, we identified several critical hyperparameters that significantly impact model performance. The hyparameters we focus on include learning_rate, n_estimators, loss, subsample, min_samples_split, min_samples_leaf and max_depth. By fine-tuning these parameters, we aim to optimize the performance and predictive capabilities of the model. In addition to the previously mentioned hyperparameters, there are several new hyperparameters that require interpretation. These hyperparameters include:

n_estimators: This hyperparameter refers to the number of boosting stages or iterations that GBDT will perform during the training process. In other words, it represents the number of DTs that will be sequentially added to the ensemble. Increasing the number of trees allows the model to learn more complex relationships within the data. However, it also increases the computational cost and the risk of overfitting.

learning_rate: When a new tree is added to the model, its purpose is to correct the mistakes made by the sum of the previous trees. This hyperparameter determines the contribution of each individual tree to the final outcome. By adjusting the learning_rate, we control the weight or influence of each tree in the ensemble. A smaller learning_rate means each tree has a smaller impact on the final prediction, while a larger learning_rate allows each tree to have a stronger influence.

loss: For classification models, there are two options for the loss function: the log-likelihood loss function “deviance” and the exponential loss function “exponential”.

subsample: This parameter represents the fraction of samples that will be used for fitting each individual base learner. It is important to note that the subsampling

technique used here is different from that of RF. While RF employs replacement sampling, where samples are randomly selected with replacement, the subsampling in GBDT does not involve putting back the samples.

In Appendix 7.8, we provide a detailed description of the hyperparameter tuning process for the GBDT model, including the range of values considered for each hyperparameter and the evaluation metrics used to assess the performance of different parameter configurations. The finalized hyperparameter configuration of the GBDT model is presented in Table 5-10.

Table 5-10. Hyperparameter Configuration for the GBDT Model

Hyperparameter	Value
ccp_alpha	0.0
criterion	friedman_mse
init	None
learning_rate	0.22
loss	deviance
max_depth	20.0
max_features	None
max_leaf_nodes	None
min_impurity_decrease	0.0
min_samples_leaf	1.0
min_samples_split	2.0
min_weight_fraction_leaf	0.0
n_estimators	8000
n_iter_no_change	None
random_state	42
subsample	1.0
tol	0.0001
validation_fraction	0.1
verbose	0.0
warm_start	False

5.3.3 Model Performance After Hyperparameter Tuning

To evaluate the performance of the final GBDT model, we first conduct KCV on the training set. The results show an average F1 score of 0.82 and an average ROC-AUC of 0.94. These scores provide strong evidence of the GBDT model's effectiveness in distinguishing between positive and negative instances. For a detailed account of the KCV process, please refer to Appendix 7.9. We also compare the model's performance on the testing set before and after hyperparameter tuning. As shown in Table 5-11, the F1 score experiences a significant boost from 0.50 to 0.93, indicating an improved balance between precision and recall. Consistently, the recall score exhibits a substantial increase

from 0.50 to 0.89. To gain a better insight, we analyze the confusion matrix. Table 5-12 reveals a significant enhancement in the predictive performance of the final GBDT model. It accurately identifies an additional 586 instances (TN+TF) compared to the original model. In addition, there is a notable reduction of 605 instances in FN predictions. However, it should be noted that there is a slight increase of 19 instances in FP predictions.

Table 5-11. GBDT Model Score on Testing Set

	Default Hyperparameter	Tunned Hyperarameter
Accuracy	0.97	0.99
Precision	0.92	0.98
Recall	0.50	0.89
F1 score	0.50	0.93

Table 5-12. Confusion Matrix on Testing Set of the GBDT Model

Default Hyperparameter		**Predicted Label**	
		0 (Not Resigned)	1 (Resigned)
True Label	0 (Not Resigned)	28,594	1
	1 (Resigned)	776	6
Tunned Hyperarameter		**Predicted Label**	
		0 (Not Resigned)	1 (Resigned)
True Label	0 (Not Resigned)	28,575	20
	1 (Resigned)	171	611

In summary, the process of hyperparameter tuning had a transformative impact on the GBDT model, resulting in significant enhancements in its performance. Through the fine-tuning of hyperparameters, the model shows greater proficiency in correctly identifying positive instances and reducing FN predictions. These improvements make the tuned GBDT model a more reliable and effective tool for predicting employee turnover.

5.4 Extreme Gradient Boosting (XGBoost)

In this section, we explore the training of the Extreme Gradient Boosting (XGBoost) model, which builds upon the foundation of the GBDT model and is known for its efficiency, scalability, and high performance.

5.4.1 Model Performance Without Hyperparameter Tuning

We begin by evaluating the performance of the XGBoost model on the training set using its default hyperparameters. The results, as displayed in Table 5-13, show

relatively low F1 scores of 0.66 on the training set and 0.59 on the validation set. Although these scores are slightly better than the GBDT model with default hyperparameters, it is evident that the XGBoost model without hyperparameter tuning struggles to accurately identify a considerable number of positive cases. To enhance the model's performance, we proceed with hyperparameter tuning.

Table 5-13. XGBoost Model Score (Default Hyperparameter)

	Training Set	Validation Set
Accuracy	0.98	0.98
Precision	0.99	0.97
Recall	0.60	0.56
F1 score	0.66	0.59

5.4.2 Hyperparameter Tuning Process

The classification XGBoost model has a total of 29 hyperparameters that can be adjusted to optimize its performance (XGBoost Parameters, n.d.). In our analysis, we focus on testing hyperparameters related to tree construction, boosting process, and regularization. To identify the optimal hyperparameter configuration, we systematically explore various combinations. Through this process, we discover that the most influential parameters affecting the model's performance are learning_rate, n_estimators, subsample, and max_depth, which are introduced in the GBDT model. In addition, we also consider the following new hyperparameters introduced by the XGBoost model:

booster: This parameter provides two choices: gbtree and gblinear. When selecting gbtree, the model employs a tree structure to process the data, allowing for non-linear relationships and interactions to be captured. On the other hand, selecting gblinear utilizes a linear model, which assumes a linear relationship between the input features and the target variable.

min_child_weight: This parameter specifies the minimum sum of sample weights required for a leaf node to be created during the tree-building process. This parameter is used to control the complexity of the tree and prevent the model from creating leaf nodes with very few samples.

gamma: When a node is considered for splitting, the loss function is calculated before and after the split. The node will only be split if the loss function decreases by an amount greater than or equal to the specified gamma value.

The hyperparameter tuning process, including the range of values explored for each parameter and the evaluation metrics used to assess the performance of various parameter configurations, is provided in detail in Appendix 7.10. Table 5-14 shows the hyperparameter configuration for the final XGBoost model.

Table 5-14. Hyperparameter Configuration for the XGBoost Model

Hyperparameter	Value
objective	binary:logistic
use_label_encoder	True
base_score	0.5
booster	gbtree
colsample_bylevel	1
colsample_bynode	1
colsample_bytree	1
enable_categorical	False
gamma	0
gpu_id	-1
importance_type	None
learning_rate	0.22
max_delta_step	0
max_depth	12
min_child_weight	1.0
missing	nan
n_estimators	2800
n_jobs	16
num_parallel_tree	1
predictor	auto
random_state	42
reg_alpha	0
reg_lambda	1
scale_pos_weight	1
subsample	1
tree_method	extract
validate_parameters	1
verbosity	None

5.4.3 Model Performance After Hyperparameter Tuning

To evaluate the performance of the final XGBoost model, we first conduct KCV to assess its stability. A detailed description of the KCV process can be found in Appendix 7.11. The average F1 score on the training set is 0.83, indicating a strong performance in classifying the training data. Moreover, the average ROC-AUC score of 0.94 suggests a robust performance. To further evaluate the model, we compare its performance on the testing set before and after hyperparameter tuning. Table 5-15 provides the model scores before and after tuning. The precision of the model remains consistent at 0.97, while the recall increases from 0.56 to 0.89. This increase signifies an improved ability to correctly identify positive instances. The

better balance between high precision and high recall is further demonstrated by the F1 score, which rises from 0.59 to 0.93. To gain a better insight, we examine the confusion matrix, which provides a detailed breakdown of the model's predictions. As shown in Table 5-16, we observe that compared to the model without hyperparameter tuning, the tuned model correctly predicts an additional 518 instances (TN+TF), while the number of FN predictions decreases by 528. These improvements indicate a significant enhancement in the model's performance.

Table 5-15. XGBoost Model Score on Testing Set

	Default Hyperparameter	**Tunned Hyperarameter**
Accuracy	0.98	0.99
Precision	0.97	0.97
Recall	0.56	0.89
F1 score	0.59	0.93

Table 5-16. Confusion Matrix on Testing Set of the XGBoost Model

Default Hyperparameter		**Predicted Label**	
		0 (Not Resigned)	1 (Resigned)
True Label	0 (Not Resigned)	28,593	2
	1 (Resigned)	703	70
Tunned Hyperarameter		**Predicted Label**	
		0 (Not Resigned)	1 (Resigned)
True Label	0 (Not Resigned)	28,574	21
	1 (Resigned)	175	607

In summary, the hyperparameter tuning process had a significant impact on the XGBoost model. The tuned XGBoost model exhibits enhanced abilities to accurately identify positive instances and reduce FN predictions. The notable increase in the F1 score highlights the model's improved balance between high precision and high recall, leading to more accurate and dependable predictions. Consequently, the tuned XGBoost model proves to be a highly reliable and effective tool for predicting employee turnover.

6 Result and Conclusion

In this chapter, we present the results and conclusions derived from our study on predicting employee turnover using machine learning models. Specifically, we focus on three key aspects: the model results (6.1), the business value derived from our findings (6.2), and suggestions for future extensions and improvements (6.3). By examining these aspects, we aim to provide a comprehensive overview of the significance and implications of our thesis in the field of employee turnover prediction.

6.1 Model Result

Despite the Decision Tree (DT) and Random Forest (RF) models demonstrating reasonable performance even without hyperparameter tuning, they do not measure up to the performance of the final Gradient Boosting Decision Tree (GBDT) and Extreme Gradient Boosting (XGBoost) models with the available dataset. While both GBDT and XGBoost initially exhibit relatively poor performance with the default settings, the process of hyperparameter tuning significantly enhances their performance. On the other hand, the process of hyperparameter tuning does not lead to a significant change in the performance of the DT and RF models. One possible explanation for this difference in performance improvement is that the DT and RF models already operate near their optimal performance with the default settings, leaving limited room for further enhancement. This difference could also be attributed to the inherent characteristics of the models. GBDT and XGBoost are ensemble methods that sequentially add DTs to correct the errors made by previous models. This iterative process allows them to effectively learn complex patterns and relationships in the data. In contrast, DT and RF models do not have this boosting capability. Overall, the difference in performance before and after hyperparameter tuning highlights the importance of optimizing the hyperparameters for boosting-based models like GBDT and XGBoost. It also emphasizes the potential limitations of DT and RF models in capturing complex patterns in the data.

When comparing the performance of the final GBDT and XGBoost models, both models achieve a high F1 score of 0.93. The only minor difference in performance is that the GBDT model exhibits slightly higher precision (0.01), which indicates its proficiency in accurately identifying positive instances. However, there is a

significant difference in training time between the two models. The XGBoost model takes approximately 2.7 minutes to train on our training dataset, while the GBDT model requires around 100.7 minutes. This substantial difference in training time can be attributed to the underlying algorithms and implementation details of the two models. XGBoost is specifically designed to optimize the performance of GBDT through various algorithmic enhancements, including parallelization techniques. These optimizations allow XGBoost to efficiently process large datasets and expedite the training process.

In conclusion, our analysis of the different supervised learning models for employee turnover prediction reveals that the GBDT and XGBoost models outperform the DT and RF models. This highlights the importance of hyperparameter tuning for boosting-based models and exposes the potential limitations of DT and RF models in capturing complex patterns in the data. When comparing GBDT and XGBoost, both models achieve a high F1 score of 0.93, with GBDT exhibiting slightly higher precision. However, there is a significant discrepancy in training time, with XGBoost being considerably faster due to its algorithmic optimizations. The choice between GBDT and XGBoost should consider the trade-off between slightly higher precision and faster training time.

6.2 Business Value

The application of machine learning models to predict employee turnover offers important business value to organizations. In this section, we will focus on the final GBDT model as an example to showcase the potential business benefits of implementing a predictive model. By analyzing the confusion matrix of the GBDT model on the testing set, we can observe its slightly better predictive performance compared to the final XGBoost model, with 5 additional correct predictions.

6.2.1 Feature Importance

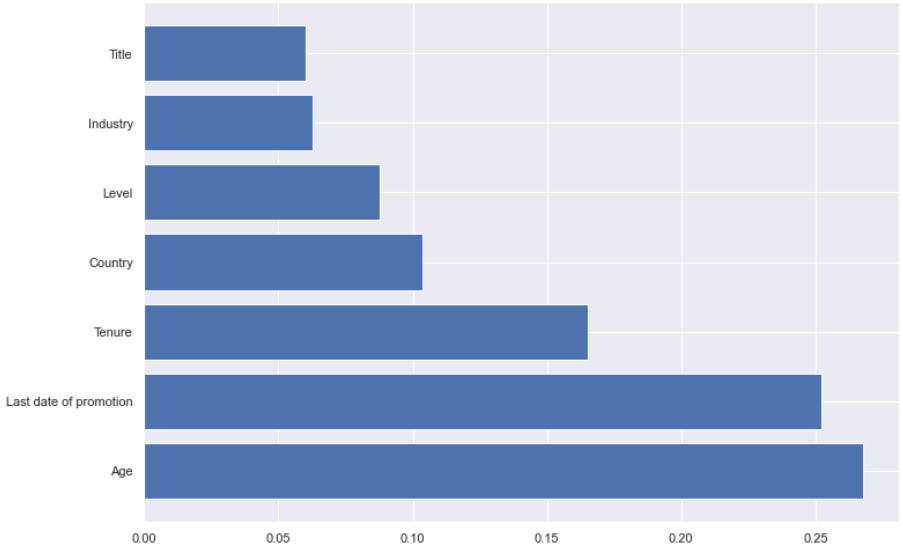


Figure 6-1 Feature Importance for the Final GBDT Model

The feature importance plot in Figure 6-1 represents the relative importance of each feature in the final GBDT model. It reveals that "Age" as the most influential feature contributes approximately 26.8% to the overall importance. This indicates that age significantly affects the output of the classifier. By integrating this finding with the descriptive analysis depicted in Figure 4-2, we can conclude that age plays a critical role in predicting employee turnover, with younger individuals exhibiting a higher tendency to leave. This understanding of age's impact can enable organizations to develop targeted strategies catering to the specific needs and concerns of different age groups. For instance, implementing mentorship programs or offering tailored career development opportunities for younger employees may help improve their job satisfaction and increase retention rates. Organizations can also create age-specific initiatives to foster a supportive work environment and address any age-related challenges that may contribute to turnover (Naim & Lenka, 2018). By leveraging the insights gained from the feature importance analysis, organizations can make informed decisions on resource allocation and implement targeted interventions to effectively address the impact of important features on employee turnover.

6.2.2 Profit Matrix

Employee turnover is associated with indirect costs, such as recruitment, onboarding, and training expenses, as discussed in Chapter 2.2. Employee turnover prediction models offer valuable insights to organizations by identifying potential

risks and facilitating proactive measures. To gain a better understanding of the financial impact of such models, a profit matrix can be constructed, which defines the costs and benefits associated with different prediction outcomes. Table 6-1 presents an example of the profit matrix, illustrating the potential costs and benefits for each prediction type. It should be noted that these estimates are derived from an interview with the HR department of the company that provided the dataset for our analysis and serve as simplified examples. The actual costs and benefits will vary depending on the specific circumstances of each organization, including factors such as industry, country, and company policies.

Table 6-1. Profit Matrix

		Predicted Label	
		0 (Not Resigned)	1 (Resigned)
True Label	0 (Not Resigned)	$0	-$10,000
	1 (Resigned)	-$25,000	+$5,000

The numbers in Table 6-1 for each prediction type align with the following interpretations:

True Negative (TN): The model predicts an employee will stay and they do. There is no associated cost or benefit as the business continues as usual. The net profit would be $0.

True Positive (TP): The model predicts an employee will leave and they do. The cost could be seen as the expenses associated with hiring and training a replacement. The benefit could be the cost savings from potentially avoiding a period of low productivity or the costs associated with a sudden departure. If we assume the cost to replace an employee is $10,000, and we successfully manage to avoid a productivity loss worth $15,000, the net profit would be +$5,000.

False Positive (FP): The model predicts an employee will leave, but they stay. The cost might be the unnecessary expenditure on hiring or training a replacement. Using the same numbers, if we spend $10,000 preparing for a departure that does not happen and there is no productivity gain, our net profit would be -$10,000.

False Negative (FN): The model predicts an employee will stay, but they leave. This could incur costs due to productivity loss, cost to hire and train a replacement, and potential overtime for other employees. Assuming these costs amount to $25,000 in total, the net profit would be -$25,000.

Based on the estimated profit matrix, we can quantify the profit resulting from the implementation of a predictive model. For instance, if we use the data from June 2022 in our dataset as an illustrative example. In this particular month, there are 231 employees who are projected to resign within the next three months out of a total of 8,738 employees in the company. By comparing the costs associated with the baseline model (representing the absence of a predictive model) to those of the final GBDT model, we can assess the financial impact and potential profit achieved through the implementation of the final GBDT model.

For the baseline model:

True Negatives (TN):

$$8{,}507 * \$0 = \$0 \tag{6-1}$$

False Negatives (FN):

$$231 * (-\$25{,}000) = -\$5{,}775{,}000 \tag{6-2}$$

False Positives (FP):

$$0 * (-\$10{,}000) = \$0 \tag{6-3}$$

True Positives (TP):

$$0 * \$5{,}000 = \$0 \tag{6-4}$$

Summing up these values, we get:

$$\$0 + (-\$5{,}775{,}000) + \$0 + \$0 = -\$5{,}775{,}000 \tag{6-5}$$

Therefore, the estimated cost based on the absence of a predictive model is $5,775,000.

The predictions from the final GBDT model on this dataset are TN are 8507, TP are 223, FP are 0, and FN are 8. We can calculate the associated costs:

Based on the predictions generated by the final GBDT model on this dataset, we have 8,507 TN, 223 TP, 8 FN, and 0 FP. With this information, we can proceed to calculate the associated costs:

True Negatives (TN):

$$8{,}507 * \$0 = \$0 \tag{6-6}$$

False Negatives (FN):

$$8 * (-\$25{,}000) = -\$200{,}000 \tag{6-7}$$

False Positives (FP):

$$0 * (-\$10{,}000) = \$0 \tag{6-8}$$

True Positives (TP):

$$223 * \$5{,}000 = \$1{,}115{,}000 \tag{6-9}$$

Summing up these values, we get:

$$\$0 + (-\$200{,}000) + \$0 + \$1{,}115{,}000 = \$915{,}000 \tag{6-10}$$

The estimated profit based on the final GBDT model is $915,000.

In this case, the implementation of the final GBDT model in this company results in a profit gain of $915,000 for the three months following June 2022 instead of incurring a cost of $5,775,000. This outcome demonstrates the substantial financial benefits that can be achieved by implementing an effective predictive model to identify and manage turnover risks. By leveraging the insights gained from a predictive model, organizations can optimize their workforce management strategies, mitigate turnover risks, and effectively reduce recruitment and training costs. This ultimately leads to substantial cost reductions, providing tangible business value and improved financial performance for organizations.

6.3 Future Extension and Improvement

While this study provides valuable insights into predicting employee turnover using supervised machine learning models, there are several avenues for future research and potential improvements to enhance the effectiveness of these models. Here, we outline some potential areas of focus for future extensions and improvements:

Integration of additional data sources: Expanding the dataset to include more diverse and comprehensive sources of data, such as external factors like industry

trends and economic indicators, can enrich the predictive models. Incorporating these additional variables can provide a more holistic understanding of employee turnover and improve the accuracy of predictions.

Feature engineering: Exploring advanced feature engineering techniques can help in identifying and creating more informative features that capture the complex relationships and interactions among variables. Techniques like feature interaction, feature scaling, and dimensionality reduction can enhance the predictive power of the models and uncover hidden patterns within the data.

Incorporating temporal analysis: Employee turnover patterns can exhibit temporal dependencies, such as seasonality or trends over time. By incorporating temporal analysis techniques, such as time series modeling or recurrent neural networks, into the prediction models, organizations can better capture the dynamic nature of employee turnover and improve the accuracy of long-term forecasts.

Continuous model monitoring and updating: Employee turnover dynamics can change over time due to various internal and external factors. Therefore, establishing a system for continuous model monitoring and updating is crucial. Regularly evaluating the model's performance, incorporating new data, and retraining the models can ensure their reliability and effectiveness in real-world scenarios.

By addressing these future extensions and improvements, organizations can enhance their employee turnover prediction capabilities, enabling them to make more informed decisions regarding retention strategies, succession planning, and overall human resource management.

7 Appendix

7.1 Code for Categorical Feature Encoding

The following code snippet demonstrates the process of encoding categorical features into numerical representations:

```
from sklearn import preprocessing
industry_le =
preprocessing.LabelEncoder().fit(df["Industry"].unique())
df["Industry"] = industry_le.transform(df["Industry"])
title_le = preprocessing.LabelEncoder().fit(df["Title"].unique())
df["Title"] = title_le.transform(df["Title"])
country_le =
preprocessing.LabelEncoder().fit(df["Country"].unique())
df["Country"] = country_le.transform(df["Country"])
df[["Industry", "Title", "Country"]] = df[["Industry", "Title",
"Country"]].astype("int64")
```

7.2 Code for Data Splitting

The following code is used to splitting the dataset into training, validation and testing set:

```
from sklearn.model_selection import train_test_split
from collections import Counter
y = df["Resigned"]
X = df.drop("Resigned", axis=1)

# split training and test data.
X_train, X_test, y_train, y_test = train_test_split(X, y,
random_state=42, test_size = .40, stratify = y)
print('train shape %s' % Counter(y_train))

# split test data into a valuation set and a holdout set
X_value, X_test, y_value, y_test = train_test_split(X_test, y_test,
random_state=42, test_size = .5, stratify = y_test)
```

7.3 Process for Feature Selection

7.3.1 ANOVA for Numerical Feature Selection Process

The following code snippet demonstrates the process of encoding the “Last date of promotion” column and selecting numerical features for data splitting in preparation for ANOVA:

```
df["Last date of promotion"] = pd.to_datetime(df["Last date of
promotion"]).view("int64")

X_num = df[["Age", "Tenure","Last date of promotion","Level"]]
y = df["Resigned"]

# split training and test data.
X_train, X_test, y_train, y_test = train_test_split(X_num, y,
random_state=42, test_size = .40, stratify = y)
print('train shape %s' % Counter(y_train))

# split test data into a valuation set and a holdout set
X_value, X_test, y_value, y_test = train_test_split(X_test, y_test,
random_state=42, test_size = .5, stratify = y_test)
print("value/test shape %s" % Counter(y_test))
```

The following code snippet demonstrates the implementation of ANOVA:

```
from sklearn.feature_selection import SelectKBest, f_classif
selector = SelectKBest(f_classif, k=4)
selector.fit(X_train, y_train)

p_values = pd.Series(selector.pvalues_, index= X_train.columns)
p_values.sort_values(ascending = True , inplace = True)
print(p_values<=0.05)
```

The output of the ANOVA is as follows:

```
Tenure                    True
Age                       True
Level                     True
Last date of promotion    True
dtype: bool
```

7.3.2 Chi-square test for Categorical Feature Selection Process

To prepare the data for the Chi-square test, the following code was used:

```
X_num = df[["Age", "Tenure","Last date of promotion","Level"]]
y = df["Resigned"]

# split training and test data.
X_train, X_test, y_train, y_test = train_test_split(X_cat, y,
random_state=42, test_size = .40, stratify = y)
print('train shape %s' % Counter(y_train))

# split test data into a valuation set and a holdout set
X_value, X_test, y_value, y_test = train_test_split(X_test, y_test,
random_state=42, test_size = .5, stratify = y_test)
print("value/test shape %s" % Counter(y_test))
```

The following code snippet demonstrates the implementation of Chi-square test:

```
from sklearn.feature_selection import chi2
# Create and fit selector
selector= SelectKBest(chi2, k=4)
selector.fit(X_train, y_train)

p_values = pd.Series(selector.pvalues_, index= X_train.columns)
p_values.sort_values(ascending = True , inplace = True)
print(p_values<=0.05)
```

The output of the Chi-square test is as follows:

```
Country      True
Industry     True
Title        True
Gender      False
dtype: bool
```

7.3.3 Correlation Analysis for Numerical Feature Selection Process

The code snippet below demonstrates the implementation of correlation analysis, which follows the same data preparation process as ANOVA:

```
import seaborn as sns
%matplotlib inline
sns.set(rc = {'figure.figsize':(15,10)})
train = pd.concat([X_train, y_train], axis = 1)
corr = train.corr()
sns.heatmap(corr, annot=True,
            xticklabels=corr.columns.values,
            yticklabels=corr.columns.values)
```

7.4 Hyperparameter Tuning Process for the DT Model

- **criterion**

For criterion, we have two options to choose from. We used grid search to determine which criterion yielded the highest F1 score. The code used for this purpose is as follows:

```
criterion = ["gini", "entropy"]
parameter = dict(criterion = criterion)
grid_search = GridSearchCV(model, parameter, cv = 5, verbose = 1,
n_jobs = -1, scoring = "f1")
grid_search.fit(X_train, y_train)
print(grid_search.best_params_)
```

```
print('''best score = {:.2f}'''.format(grid_search.best_score_))
```

The output of our analysis indicates that the “gini” criterion performed the best, as it resulted in the highest F1 score:

```
{'criterion': 'gini'}
best score = 0.78
```

- **class_weight**

In a similar manner to the criterion search, we employed grid search to determine the optimal setting for the class weight. The code used is:

```
class_weight = [None, "balanced"]
parameter = dict(class_weight = class_weight)
grid_search = GridSearchCV(model, parameter, cv = 5, verbose = 1,
n_jobs = -1, scoring = "f1")
grid_search.fit(X_train, y_train)
print(grid_search.best_params_)
print('''best score = {:.2f}'''.format(grid_search.best_score_))
```

The output shows:

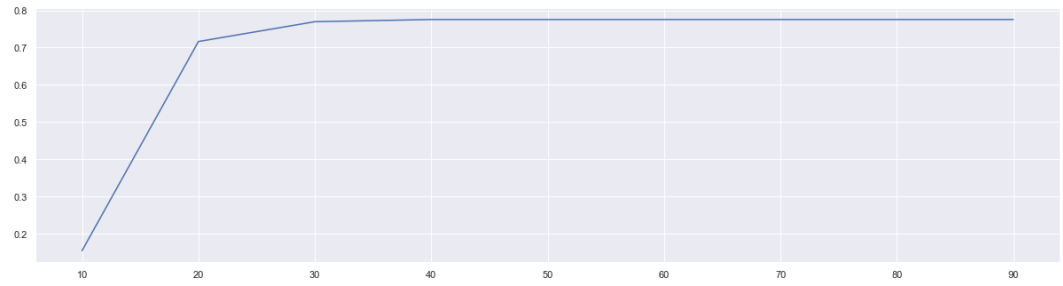
```
{'class_weight': None}
best score = 0.78
```

- **max_depth**

To search for the best max_depth, we implemented a for loop combined with cross-validation methodology, and plot the scores as a function of this parameter. The code used is:

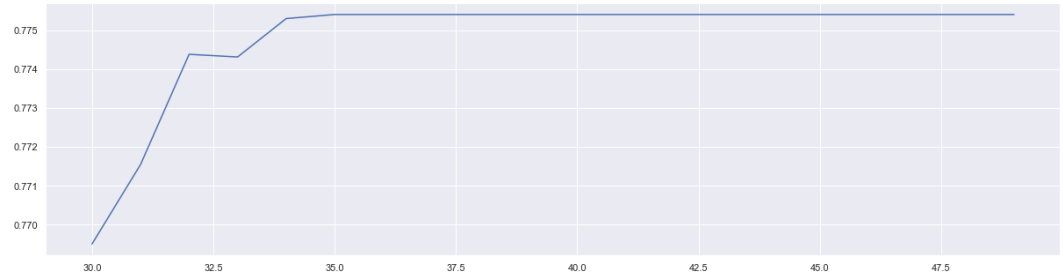
```
ScoreAll = []
for i in range():
    DT = DecisionTreeClassifier(max_depth = i, random_state = 42)
    score = cross_val_score(DT, X_train, y_train, cv=5, scoring =
"f1").mean()
    ScoreAll.append([i,score])
ScoreAll = np.array(ScoreAll)
max_score = np.where(ScoreAll==np.max(ScoreAll[:,1]))[0][0]
print("best parameter & score:",ScoreAll[max_score])
plt.figure(figsize=[20.7,5.27])
plt.plot(ScoreAll[:,0],ScoreAll[:,1])
plt.show()
```

We first search within `range(10,100,10)`, and the output shows:




```
best parameter & score: [40.          0.7754104]
```

After refining our search to `range(30,50)`, the output displayed the following results:




```
best parameter & score: [35.          0.7754104]
```

- **max_features**

Given that we have 7 features in the dataset, we used grid search to explore the optimal max_features from 1 to 7, and visualized the changes in F1 score with different max_features values using a heatmap:

```
max_features = [None, 1, 2, 3, 4, 5, 6, 7]
parameter = dict(max_features = max_features)
model = DecisionTreeClassifier(max_depth=35, random_state=42)
grid_search = GridSearchCV(model, parameter, cv = 5, verbose = 1,
n_jobs = -1, scoring = "f1")
grid_search.fit(X_train, y_train)
dt = pd.DataFrame(grid_search.cv_results_)
dt.param_ max_features = dt.param_ max_features.astype(str)
table = pd.pivot_table(dt, values='mean_test_score', index='param_
max_features)
sns.heatmap(table)
print(grid_search.best_params_)
print('''best score = {:.2f}'''.format(grid_search.best_score_))
```

The output shows:

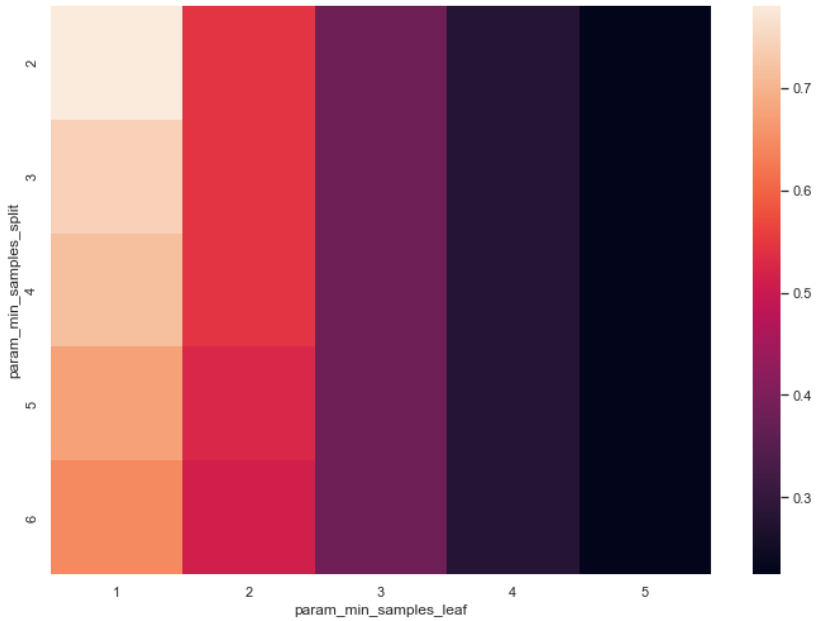

```
{'max_features': 2}
best score = 0.78
```

- **min_samples_leaf & min_samples_split**

Considering the interactivity between min_samples_leaf and min_samples_split, we decided to tune them together. Utilizing grid search, we explored various combinations of these parameters and obtained the corresponding F1 score. Subsequently, we generated a heatmap to visualize the relationship between the different parameter combinations and the F1 score. The code used is as follows:

```
min_samples_split = [2, 3, 4, 5, 6]
min_samples_leaf = [1, 2, 3, 4, 5]
parameter = dict(min_samples_split = min_samples_split,
min_samples_leaf = min_samples_leaf)
model = DecisionTreeClassifier(max_depth=35, max_features=2,
random_state=42)
grid_search = GridSearchCV(model, parameter, cv = 5, verbose = 1,
n_jobs = -1, scoring = "f1")
grid_search.fit(X_train, y_train)
dt = pd.DataFrame(grid_search.cv_results_)
dt.param_min_samples_split = dt.param_min_samples_split.astype(str)
dt.param_min_samples_leaf = dt.param_min_samples_leaf.astype(str)
table = pd.pivot_table(dt, values='mean_test_score',
index='param_min_samples_split', columns='param_min_samples_leaf')
sns.heatmap(table)
print(grid_search.best_params_)
print('''best score = {:.2f}'''.format(grid_search.best_score_))
```

The output shows:




```
{'min_samples_leaf': 1, 'min_samples_split': 2}
best score = 0.78
```

7.5 KCV for the DT Model

To assess the model's performance, we utilized K-fold Cross Validation (KCV) with 10 folds on the training set. Firstly, we obtained 10 F1 score through this process and calculated their average value. Subsequently, we repeated the same procedure to obtain and average the ROC-AUC scores. This approach allowed us to gain a comprehensive understanding of the model's performance across different folds. The code used is as follows:

```
f1_scores = cross_val_score(model_1, X_train, y_train, cv=10,
scoring='f1')
f1_scores
mean(f1_scores)
roc_auc = cross_val_score(model_1, X_train, y_train, cv=10,
scoring='roc_auc')
roc_auc
mean(roc_auc)
```

The output for F1 score is:

```
array([0.84140969, 0.80742459, 0.77674419, 0.76738609, 0.80652681,
0.78983834, 0.8       , 0.75238095, 0.78983834, 0.81132075])

mean: 0.7942869750760719
```

The output for ROC-AUC is:

```
array([0.91950086, 0.89136138, 0.88028983, 0.86765636, 0.88739452,
       0.88314714, 0.88937014, 0.86391362, 0.88504884, 0.89009639])

mean: 0.885777907843261
```

7.6 Hyperparameter Tuning Process for the RF Model

- **criterion**

The code snippet below demonstrates the use of grid search to determine the criterion that yields the highest F1 score:

```
criterion = ["gini", "entropy"]
parameter = dict(criterion = criterion)
model = RandomForestClassifier(random_state=42)
grid_search = GridSearchCV(model, parameter, cv = 5, verbose = 1,
n_jobs = -1, scoring = "f1")
grid_search.fit(X_train, y_train)
print(grid_search.best_params_)
print('''best score = {:.2f}'''.format(grid_search.best_score_))
```

The output of our analysis indicates that the “gini” criterion performed the best, resulting in the highest F1 score.

```
{'criterion': 'gini'}
best score = 0.78
```

- **bootstrap**

In a similar manner to the criterion search, we employed grid search to determine the optimal setting for the bootstrap parameter. The range for bootstrap is as follows:

```
bootstrap = ["True", "False"]
```

The output shows:

```
{'bootstrap': 'True'}
best score = 0.78
```

- **oob_score**

In a similar manner to the criterion search, we employed grid search to determine the optimal setting for the oob_score parameter. The range for oob_score is as follows:

```
bootstrap = ["True", "False"]
```

The output shows:

```
{'oob_score': 'True'}
best score = 0.78
```

- **class_weight**

In a similar manner to the criterion search, we employed grid search to determine the optimal setting for the class_weight parameter. The range for class_weight is as follows:

```
class_weight = [None, "balanced"]
```

The output shows:

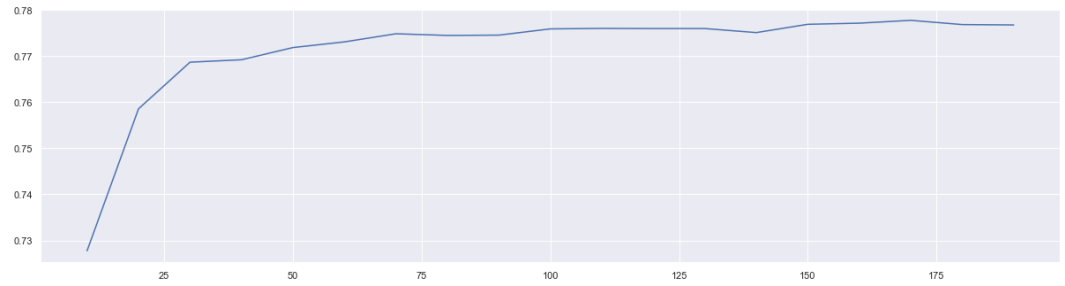
```
{'class_weight': None}
best score = 0.78
```

- **n_estimators**

The code used to search for the best n_estimators using a for loop combined with cross-validation methodology and plot the scores as a function of this parameter is as follows:

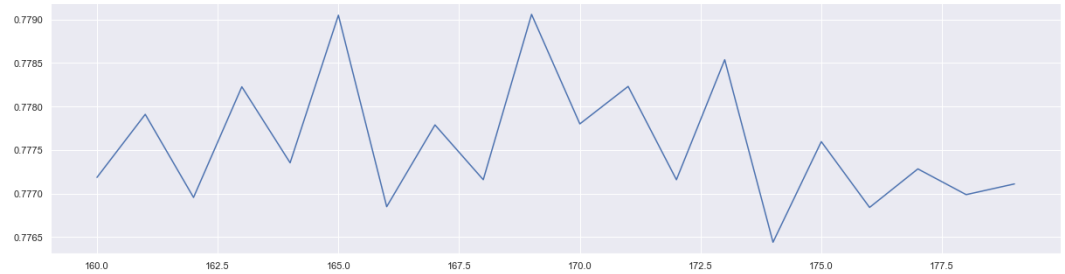
```
ScoreAll = []
for i in range(10,200,10):
    RF = RandomForestClassifier(n_estimators = i, oob_score=True,
random_state = 42)
    score = cross_val_score(RF, X_train, y_train, cv=5, scoring =
"f1").mean()
    ScoreAll.append([i,score])
ScoreAll = np.array(ScoreAll)

max_score = np.where(ScoreAll==np.max(ScoreAll[:,1]))[0][0]
print("best parameter & score:",ScoreAll[max_score])
# print(ScoreAll[,0])
plt.figure(figsize=[20.7,5.27])
plt.plot(ScoreAll[:,0],ScoreAll[:,1])
plt.show()
```

Based on the implemented code, the output for the `range (10, 200, 10)` shows the scores as a function of the n_estimators parameter is:




```
best parameter & score: [170.           0.77780047]
```

After refining the search to `range (160, 180)`, the output displayed the following results for the scores as a function of the n_estimators parameter:



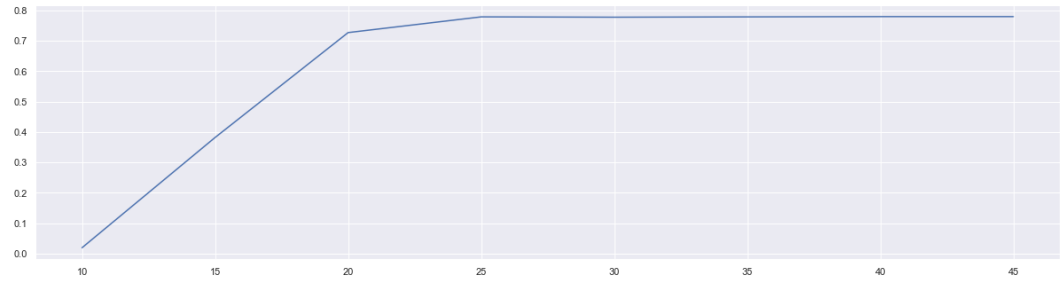

```
best parameter & score: [169.           0.77905861]
```

- **max_depth**

To search for the best max_depth, we implemented a for loop combined with cross-validation methodology, and plot the scores as a function of this parameter. The code used is:

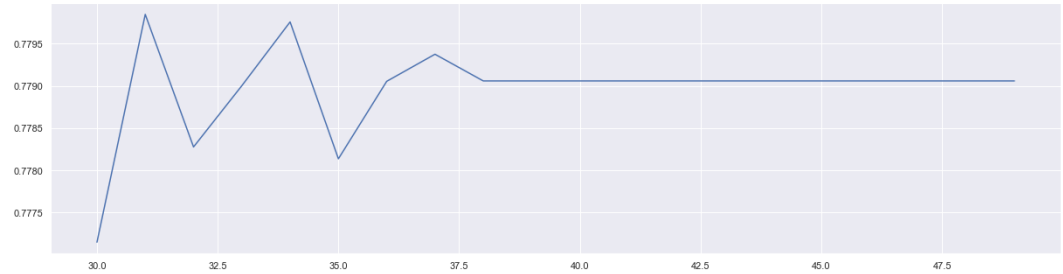
```
ScoreAll = []
for i in range(10, 50, 5):
    RF = RandomForestClassifier(n_estimators = 169, oob_score=True,
max_depth = i, random_state = 42)
    score = cross_val_score(RF, X_train, y_train, cv=5, scoring =
"f1").mean()
    ScoreAll.append([i,score])
ScoreAll = np.array(ScoreAll)

max_score = np.where(ScoreAll==np.max(ScoreAll[:,1]))[0][0]
print("best parameter & score:",ScoreAll[max_score])

plt.figure(figsize=[20.7,5.27])
plt.plot(ScoreAll[:,0],ScoreAll[:,1])
plt.show()
```

We first search within `range(10,50,10)`, and the output shows:




```
best parameter & score: [40.         0.7754104]
```

After refining our search to `range(30,50)`, the output displayed the following results:




```
best parameter & score: [31.          0.77984613]
```

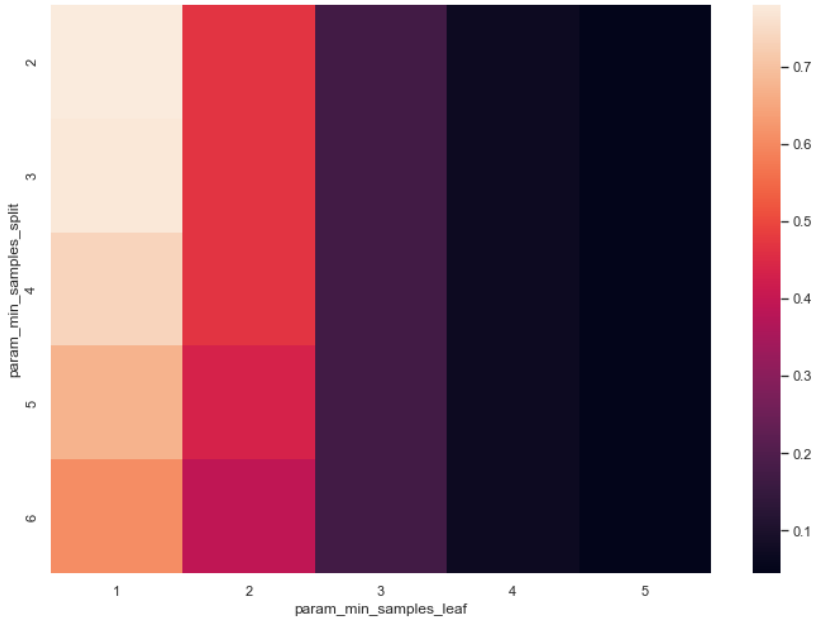
- **min_samples_leaf & min_samples_split**

Considering the interactivity between min_samples_leaf and min_samples_split, we decided to tune them together. Using grid search, we explored various combinations of these parameters and obtained the corresponding F1 score. Subsequently, we generated a heatmap to visualize the relationship between the different parameter combinations and the F1 score. The code used for this purpose is as follows:

```
min_samples_split = [2, 3, 4, 5, 6]
min_samples_leaf = [1, 2, 3, 4, 5]
parameter = dict(min_samples_split = min_samples_split,
min_samples_leaf = min_samples_leaf)
model = RandomForestClassifier(n_estimators=169, max_depth=31,
oob_score=True, random_state=42)
grid_search = GridSearchCV(model, parameter, cv = 5, verbose = 1,
n_jobs = -1, scoring = "f1")
grid_search.fit(X_train, y_train)
rf = pd.DataFrame(grid_search.cv_results_)
rf.param_min_samples_split = rf.param_min_samples_split.astype(str)
rf.param_min_samples_leaf = rf.param_min_samples_leaf.astype(str)
table = pd.pivot_table(rf, values='mean_test_score',
index='param_min_samples_split', columns='param_min_samples_leaf')
sns.heatmap(table)
```

```
print(grid_search.best_params_)
print('''best score = {:.2f}'''.format(grid_search.best_score_))
```

The output shows:



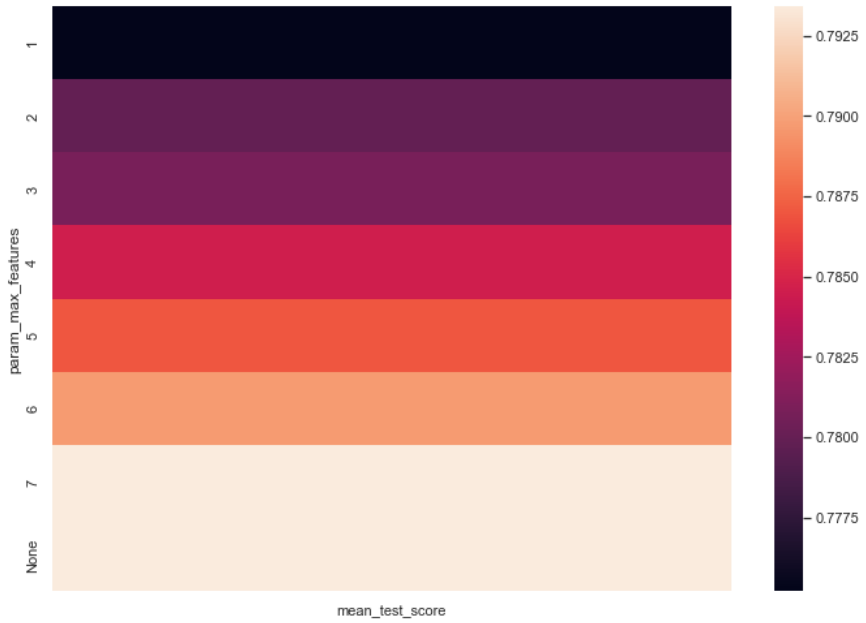

```
{'min_samples_leaf': 1, 'min_samples_split': 2}
best score = 0.78
```

- **max_features**

Given that we have 7 features in the dataset, we used grid search to explore the optimal max_features from 1 to 7. We then visualized the changes in F1 score with different max_features values using a heatmap. The code used for this process is as follows:

```
max_features = [None, 1, 2, 3, 4, 5, 6, 7]
parameter = dict(max_features = max_features)
model = RandomForestClassifier(n_estimators=169, max_depth=31,
oob_score=True, random_state=42)
grid_search = GridSearchCV(model, parameter, cv = 5, verbose = 1,
n_jobs = -1, scoring = "f1")
grid_search.fit(X_train, y_train)
rf = pd.DataFrame(grid_search.cv_results_)
rf.param_ max_features = rf.param_ max_features.astype(str)
table = pd.pivot_table(rf, values='mean_test_score', index='param_
max_features)
sns.heatmap(table)
print(grid_search.best_params_)
print('''best score = {:.2f}'''.format(grid_search.best_score_))
```

The output shows:



7.7 KCV for the RF Model

The following code snippet demonstrates the implementation of KCV for the final RF model:

```
f1_scores = cross_val_score(model_1, X_train, y_train, cv = 10,
scoring= "f1")
f1_scores
f1_scores.mean()
roc_auc = cross_val_score(model_1, X_train, y_train, cv = 10,
scoring= "roc_auc")
roc_auc
roc_auc.mean()
```

The output for F1 score is:

```
array([0.85377358, 0.82926829, 0.82089552, 0.79207921, 0.80604534,
       0.79301746, 0.80589681, 0.76214834, 0.7970297 , 0.81572482])
f1_scores.mean() 0.8075879066494746
```

The output for ROC-AUC is:

```
array([0.9813508 , 0.99114639, 0.98858298, 0.97771616, 0.98578724,
       0.97750728, 0.97800038, 0.97489446, 0.98029819, 0.98444338])
roc_auc.mean() 0.9819727262753244
```

7.8 Hyperparameter Tuning Process for the GBDT Model

- **n_estimator & learning rate**

To identify the optimal values for two key parameters, namely n_estimators and learning_rate, we employed grid search methodology. Through multiple runs with different parameter ranges, we determined that the final range for n_estimators is set between 5000 and 10000, while the range for learning_rate is defined as 0.20 to 0.25. The following code snippet showcases the implementation of the grid search process:

```
n_estimators = [5000, 6000, 7000, 8000, 9000, 10000]
learning_rate = [0.25, 0.24, 0.23, 0.22, 0.21, 0.20]
parameter1 = dict(n_estimators = n_estimators, learning_rate =
learning_rate)
Model = GradientBoostingClassifier(random_state = 42)
grid_search1 = GridSearchCV(Model, parameter1, cv = 5, verbose = 1,
n_jobs = -1, scoring = "f1")
grid_search1.fit(X_train, y_train)
print(grid_search1.best_params_)
print('''best score = {:.2f}'''.format(grid_search1.best_score_))
```

The output shows:

```
{'learning_rate': 0.22, 'n_estimators': 8000}
best score = 0.67
```

- **loss**

The code snippet below demonstrates the use of grid search to determine the loss parameter that yields the highest F1 score:

```
loss = ['deviance', 'exponential']
parameter = dict(loss = loss)
model = GradientBoostingClassifier(n_estimators = 8000,
learning_rate = 0.22, random_state = 42)
grid_search = GridSearchCV(model, parameter, cv = 5, verbose = 1,
n_jobs = -1, scoring = "f1")
grid_search.fit(X_train, y_train)
print(grid_search.best_params_)
print('''best score = {:.2f}'''.format(grid_search.best_score_))
```

The output of our analysis indicates that the “deviance” criterion performed the best, resulting in the highest F1 score.

```
{'loss': 'deviance'}
best score = 0.67
```

- **subsample**

To find the best value for the subsample parameter, we utilized a for loop in conjunction with cross-validation methodology. The following code snippet outlines the implementation:

```
ScoreAll = []
for i in np.arange(0.1,1,0.1):
    GB = GradientBoostingClassifier(n_estimators = 8000,
learning_rate = 0.22, max_depth = i, random_state = 42)
    score = cross_val_score(GB, X_train, y_train, cv=5, scoring =
"f1").mean()
    ScoreAll.append([i,score])
ScoreAll = np.array(ScoreAll)

max_score = np.where(ScoreAll==np.max(ScoreAll[:,1]))[0][0]
print("best parameter & score:",ScoreAll[max_score])
```

The output shows:

```
best parameter & score: [1.         0.6718683]
```

- **min_samples_leaf & min_samples_split**

Using grid search, we conducted an exploration of various combinations of the min_samples_leaf and min_samples_split hyperparameters. By systematically evaluating different values for these hyperparameters, we were able to identify the best combination that has the highest F1 score. The following code snippet outlines the implementation:

```
min_samples_split = [2, 3, 4, 5, 6]
min_samples_leaf = [1, 2, 3, 4, 5]
parameter = dict(min_samples_split = min_samples_split,
min_samples_leaf = min_samples_leaf)
model = GradientBoostingClassifier(n_estimators = 8000,
learning_rate = 0.22, random_state = 42)
grid_search = GridSearchCV(model, parameter, cv = 5, verbose = 1,
n_jobs = -1, scoring = "f1")
grid_search.fit(X_train, y_train)
print(grid_search.best_params_)
print('''best score = {:.2f}'''.format(grid_search.best_score_))
```

The output shows:

```
{'min_samples_leaf': 1, 'min_samples_split': 2}
best score = 0.68
```

- **max_depth**

To find the best value for the max_depth parameter, we utilized a for loop in conjunction with cross-validation methodology. The following code snippet outlines the implementation:

```
ScoreAll = []
for i in range(1,50):
    GBT = GradientBoostingClassifier(n_estimators = 8000,
learning_rate = 0.22, max_depth = i, random_state = 42)
    score = cross_val_score(GBT, X_train, y_train, cv=10, scoring =
"f1").mean()
    ScoreAll.append([i,score])

ScoreAll = np.array(ScoreAll)
max_score = np.where(ScoreAll==np.max(ScoreAll[:,1]))[0][0]
print("best parameter & score:",ScoreAll[max_score])
```

The output shows:

```
best parameter & score: [20.         0.8133697]
```

7.9 KCV for the GBDT Model

The following code snippet demonstrates the implementation of KCV for the final GBDT model:

```
f1_scores = cross_val_score(Model_f1, X_train, y_train, cv = 10,
scoring= "f1")
f1_scores
f1_scores.mean()
roc_auc = cross_val_score(Model_f1, X_train, y_train, cv = 10,
scoring= "roc_auc")
roc_auc
roc_auc.mean()
```

The output for F1 score is:

```
array([0.87414188, 0.84309133, 0.82857143, 0.79805353, 0.82014388,
0.81730769, 0.8156682 , 0.78640777, 0.81235154, 0.82352941])

f1_scores.mean() 0.8219266670539529
```

The output for ROC-AUC is:

```
array([0.95525913, 0.96359676, 0.94545392, 0.92682041, 0.93703487,
0.93633789, 0.94069143, 0.93040361, 0.93488712, 0.95091327])

roc_auc.mean() 0.9421398392936672
```

7.10 Hyperparameter Tuning Process for the XGBoost Model

- **booster**

For booster, we have two options to choose from. We used grid search to determine which criterion yielded the highest F1 score. The code used for this purpose is as follows:

```
criterion = ["gbtree", "gblinear"]
parameter = dict(criterion = criterion)
model= XGBClassifier(random_state = 42)
grid_search = GridSearchCV(model, parameter, cv = 5, verbose = 1,
n_jobs = -1, scoring = "f1")
grid_search.fit(X_train, y_train)
print(grid_search.best_params_)
print('''best score = {:.2f}'''.format(grid_search.best_score_))
```

The output of our analysis indicates that the “gbtree” criterion performed the best, as it resulted in the highest F1 score:

```
 {'criterion': 'gbtree'}
best score = 0.60
```

- **n_estimator & learning_rate**

To identify the optimal values for two key parameters, namely n_estimators and learning_rate, we employed grid search methodology. Through multiple runs with different parameter ranges, we determined that the final range for n_estimators is set between 2500 and 3000, while the range for learning_rate is defined as 0.20 to 0.25. The following code snippet showcases the implementation of the grid search process:

```
n_estimators = [2500, 2600, 2700, 2800, 2900, 3000]
learning_rate = [0.25, 0.24, 0.23, 0.22, 0.21, 0.20]
parameter1 = dict(n_estimators = n_estimators, learning_rate =
learning_rate)
model= XGBClassifier(random_state = 42)
grid_search1 = GridSearchCV(model, parameter1, cv = 5, verbose = 1,
n_jobs = -1, scoring = "f1")
grid_search1.fit(X_train, y_train)
print(grid_search1.best_params_)
print('''best score = {:.2f}'''.format(grid_search1.best_score_))
```

The output shows:

```
{'learning_rate': 0.22, 'n_estimators': 2800}
best score = 0.80
```

- **subsample**

To find the best value for the subsample parameter, we utilized a for loop in conjunction with cross-validation methodology. The following code snippet outlines the implementation:

```
ScoreAll = []
for i in np.arange(0.1,1,0.1):
    XGB = XGBClassifier(n_estimators = 2800, learning_rate = 0.22,
random_state = 42)
    score = cross_val_score(XGB, X_train, y_train, cv=5, scoring =
"f1").mean()
    ScoreAll.append([i,score])
ScoreAll = np.array(ScoreAll)

max_score = np.where(ScoreAll==np.max(ScoreAll[:,1]))[0][0]
print("best parameter & score:",ScoreAll[max_score])
```

The output shows:

```
best parameter & score: [1.         0.7978683]
```

- **min_child_weight**

To find the best value for the min_child_weight parameter, we utilized a for loop in conjunction with cross-validation methodology. The following code snippet outlines the implementation:

```
min_child_weight = [1, 2, 3, 4, 5, 6]
parameter = dict(criterion = criterion)
model = XGBClassifier(n_estimators = 2800, learning_rate = 0.22,
random_state = 42)
grid_search = GridSearchCV(model, parameter, cv = 5, verbose = 1,
n_jobs = -1, scoring = "f1")
grid_search.fit(X_train, y_train)
print(grid_search.best_params_)
print('''best score = {:.2f}'''.format(grid_search.best_score_))
```

The output shows:

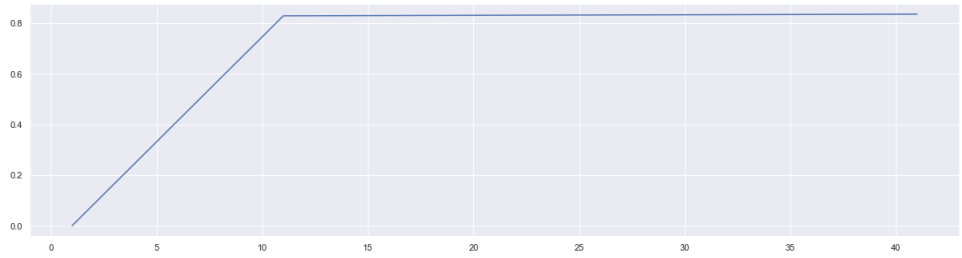
```
best parameter & score: [1.         0.80]
```

- **max_depth**

To find the best value for the max_depth parameter, we utilized a for loop in conjunction with cross-validation methodology. The following code snippet outlines the implementation:

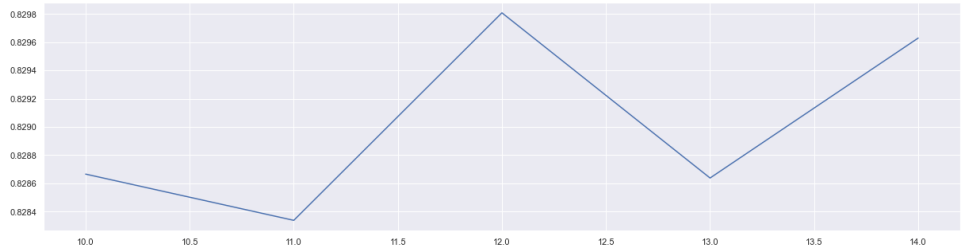
```
ScoreAll = []
for i in range(1,50,10):
    xgb = XGBClassifier(n_estimators = 2800, learning_rate = 0.22,
max_depth = i, random_state = 42)
    score = cross_val_score(xgb, X_train, y_train, cv=10, scoring =
"f1").mean()
    ScoreAll.append([i,score])
ScoreAll = np.array(ScoreAll)

max_score = np.where(ScoreAll==np.max(ScoreAll[:,1]))[0][0]
print("best parameter & score:",ScoreAll[max_score])
```

We first search within `range(10,50,10)`, and the output shows:




```
best parameter & score: [10.          0.82864885]
```

After refining our search to `range (10,15)`, and the output displayed the following results:




```
best parameter & score: [12.          0.82980737]
```

- **gamma**

To find the best value for the gamma parameter, we utilized a for loop in conjunction with cross-validation methodology. The following code snippet outlines the implementation:

```
ScoreAll = []
for i in np.arange(0,1,0.1):
    XGB = XGBClassifier(n_estimators = 2800, learning_rate = 0.22,
max_depth = 12, random_state = 42)
    score = cross_val_score(XGB, X_train, y_train, cv=5, scoring =
"f1").mean()
    ScoreAll.append([i,score])
ScoreAll = np.array(ScoreAll)

max_score = np.where(ScoreAll==np.max(ScoreAll[:,1]))[0][0]
print("best parameter & score:",ScoreAll[max_score])
```

The output shows:

```
best parameter & score: [0.         0.83040745]
```

7.11 KCV for the XGBoost Model

The following code snippet demonstrates the implementation of KCV for the final XGBoost model:

```
f1_scores = cross_val_score(model_1, X_train, y_train, cv=10,
scoring='f1')
f1_scores
roc_auc = cross_val_score(model_1, X_train, y_train, cv=10,
scoring='roc_auc')
roc_auc
```

The output for F1 score is:

```
array([0.86836028, 0.85781991, 0.8377724 , 0.80295567, 0.84107579,
0.81265207, 0.84160757, 0.82409639, 0.82211538, 0.8321513 ])
mean(f1_scores)
0.8340606742621919
```

The output for ROC-AUC is:

```
array([0.96682622, 0.96901946, 0.96071928, 0.93057635, 0.95593081,
0.9515222 , 0.95299108, 0.94000759, 0.9535633 , 0.96570271])
mean(roc_auc)
0.9546858993543491
```

8. References

Akinyomi, O. J. (2016). Labour turnover: Causes, consequences and prevention. *Fountain University Journal of Management and Social Sciences*, *5*(1), 105–112.

Alao, D., & Adeyemo, A. B. (2013). Analyzing employee attrition using decision tree algorithms. *Computing, Information Systems, Development Informatics and Allied Research Journal*, *4*(1), 17–28.

Allen, D. G., & Griffeth, R. W. (1999). Job performance and turnover: A review and integrative multi-route model. *Human Resource Management Review*, *9*(4), 525–548. https://doi.org/10.1016/s1053-4822(99)00032-7

Alpaydin, E. (2020). *Introduction to machine learning*. MIT press.

Al-Radaide, Q. A., & Al Nagi, E. (2012). Using data mining techniques to build a classification model for predicting employees performance. *International Journal of Advanced Computer Science and Applications*, *3*(2), 144–151. https://doi.org/10.14569/ijacsa.2012.030225

Al-Suraihi, W. A., Samikon, S. A., Al-Suraihi, A. H. A., & Ibrahim, I. (2021). Employee turnover: Causes, importance and retention strategies. *European Journal of Business and Management Research*, *6*(3), 1–10. https://doi.org/10.24018/ejbmr.2021.6.3.893

Anguita, D., Ghelardoni, L., Ghio, A., Oneto, L., & Ridella, S. (2012). The 'K' in K-fold Cross Validation. *Computational Intelligence*.

Anwar Hossen, M., Hossain, E., Zahereel Ishwar, A. K., & Siddika, F. (2021). Ensemble method based architecture using random forest importance to predict employee's turn over. *Journal of Physics: Conference Series*, *1755*. https://doi.org/10.1088/1742-6596/1755/1/012039

Bakry, U., Ayeldeen, H., Ayeldeen, G., & Shaker, O. (2017). Classification of liver fibrosis patients by multi-dimensional analysis and SVM classifier: An egyptian case study. *Proceedings of SAI Intelligent Systems Conference (IntelliSys) 2016*, *15*, 1085–1095. https://doi.org/10.1007/978-3-319-56994-9_75

Bapna, R., Langer, N., Mehra, A., Gopal, R., & Gupta, A. (2012). Human capital investments and employee performance: An analysis of IT services industry. *Management Science*, *59*(3), 641–658. https://doi.org/10.1287/mnsc.1120.1586

Beheshtifar, M., & Allahyary, M. H. (2012). Study the relationship among organizational reputation with organizational commitment and employees' turnover intention. *International Research Journal of Applied and Basic Sciences*, *6*(10), 1467–1478.

Boughorbel, S., Jarray, F., & El-Anbari, M. (2017). Optimal classifier for imbalanced data using matthews correlation coefficient metric. *PloS One*, *12*(6). https://doi.org/10.1371/journal.pone.0177678

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (2017). Classification and regression trees. In *Routledge eBooks*. Routledge. https://doi.org/10.1201/9781315139470

Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1), 5–32. https://doi.org/10.1023/A:1010933404324

Cahyana, N., Khomsah, S., & Aribowo, A. S. (2019). Improving imbalanced dataset classification using oversampling and gradient boosting. In *2019 5th International Conference on Science in Information Technology (ICSITech)*. IEEE. https://doi.org/10.1109/icsitech46713.2019.8987499

Cascio, W., & Boudreau, J. (2011). *Investing in people: Financial impact of human resource initiatives* (2nd ed.). FT Press.

Chandrashekar, G., & Sahin, F. (2014). A survey on feature selection methods. *Computers & Electrical Engineering*, *40*(1), 16–28. https://doi.org/10.1016/j.compeleceng.2013.11.024

Chanodkar, A., Changle, R., & Mamtani, D. (2019). Prediction of employee turnover in organizations using machine learning algorithms. *Prestige International Journal of Management and Research*, *12*(1/2), 222–226.

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. https://doi.org/10.1145/2939672.2939785

Chien, C. F., & Chen, L. F. (2008). Data mining to improve personnel selection and enhance human capital: A case study in high-technology industry. *Expert Systems With Applications*, *34*(1), 280–290. https://doi.org/10.1016/j.eswa.2006.09.003

Cotton, J. L., & Tuttle, J. M. (1986). Employee turnover: A meta-analysis and review with implications for research. *Academy of Management Review*, *11*(1), 55–70. https://doi.org/10.5465/amr.1986.4282625

Deng, X., Liu, Q., Deng, Y., & Mahadevan, S. (2016). An improved method to construct basic probability assignment based on the confusion matrix for classification problem. *Information Sciences*, 340–341, 250–261. https://doi.org/10.1016/j.ins.2016.01.033

Durant, K. T., & Smith, M. D. (2007). Predicting the political sentiment of web log posts using supervised machine learning techniques coupled with

feature selection. *Advances in Web Mining and Web Usage Analysis*, *4811*, 187–206. https://doi.org/10.1007/978-3-540-77485-3_11

Esmaieeli Sikaroudi, A., Ghousi, R., & Sikaroudi, A. (2015). A data mining approach to employee turnover prediction (case study: Arak automotive parts manufacturing). *Journal of Industrial and Systems Engineering*, *8*(4), 106–121.

Friedman, J. H. (2000). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 1189–1232. https://www.jstor.org/stable/2699986

Géron, A. (2022). *Hands-on machine learning with Scikit-Learn, Keras, and Tensorflow*. O'Reilly Media, Inc.

Hauke, J., & Kossowski, T. (2011). Comparison of Values of Pearson's and Spearman's Correlation Coefficients on the Same Sets of Data. *QUAGEO*, 30(2), 87–93. https://doi.org/10.2478/v10117-011-0021-1

Ho, T. K. (1995). Random decision forests. *Proceedings of 3rd International Conference on Document Analysis and Recognition*, *1*, 278–282. https://doi.org/10.1109/ICDAR.1995.598994

Holtom, B. C., Mitchell, T. R., Lee, T. W., & Eberly, M. B. (2008). Turnover and retention research: A glance at the past, a closer review of the present, and a venture into the future. *The Academy of Management Annals*, *2*(1), 231–274. https://doi.org/10.1080/19416520802211552

Jeni, L. A., Cohn, J. F., & De La Torre, F. (2013). Facing imbalanced data--recommendations for the use of performance metrics. *2013 Humaine Association Conference on Affective Computing and Intelligent Interaction*, 245–251. https://doi.org/10.1109/acii.2013.47

Kamalanabhan, T. J., Prakash Sai, L. P., & Mayuri, D. (2009). Employee engagement and job satisfaction in the information technology industry. *Psychological Reports*, *105*(3), 759–770. https://doi.org/10.2466/pr0.105.3.759-770

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T. (2017). LightGBM: A highly efficient gradient boosting decision tree. In *Advances in neural information processing systems* (Vol. 30, pp. 3149–3157).

Khalid, S., Khalil, T., & Nasreen, S. (2014). A survey of feature selection and feature extraction techniques in machine learning. *2014 Science and Information Conference*, 372–378. https://doi.org/10.1109/sai.2014.6918213

Khera, S. N., & Divya. (2018). Predictive modelling of employee turnover in indian IT industry using machine learning techniques. *Vision*, *23*(1), 12–21. https://doi.org/10.1177/0972262918821221

Kim, W. G., Leong, J. K., & Lee, Y. K. (2005). Effect of service orientation on job satisfaction, organizational commitment, and intention of leaving in a casual dining chain restaurant. *International Journal of Hospitality Management*, *24*(2), 171–193. https://doi.org/10.1016/j.ijhm.2004.05.004

Kishore, R. A., Sanghadasa, M., & Priya, S. (2017). Optimization of segmented thermoelectric generator using Taguchi and ANOVA techniques. *Scientific Reports*, 7(1), Article 1. https://doi.org/10.1038/s41598-017-16372-8

Korff, V. P., Balbo, N., Mills, M., Heyse, L., & Wittek, R. (2015). The impact of humanitarian context conditions and individual characteristics on aid worker retention. *Disasters*, *39*(3), 522–545. https://doi.org/10.1111/disa.12119

Kristensen, N., & Johansson, E. (2008). New evidence on cross-country differences in job satisfaction using anchoring vignettes. *Labour Economics*, *15*(1), 96–117. https://doi.org/10.1016/j.labeco.2006.11.001

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84–90.

LeCun, Y., Bengio, Y., & Hinton, G. E. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., & Liu, H. (2017). Feature selection: A data perspective. *ACM Computing Surveys*, *50*(6), 94. https://doi.org/10.1145/3136625

Li, Y. M., Lai, C. Y., & Kao, C. P. (2010). Building a qualitative recruitment system via SVM with MCDM approach. *Applied Intelligence*, *35*(1), 75–88. https://doi.org/10.1007/s10489-009-0204-9

Mathias, H. D., & Ragusa, V. R. (2017). Micro aerial vehicle path planning and flight with a multi-objective genetic algorithm. *Proceedings of SAI Intelligent Systems Conference (IntelliSys) 2016*, *15*, 107–124. https://doi.org/10.1007/978-3-319-56994-9_8

Matthew, O., & Kung, M. C. (2007). The cost of employee turnover. *Industrial Management*, *49*(1), 14–19.

Moyes, G. D., Shao, L. P., & Newsome, M. (2008). Comparative analysis of employee job satisfaction in the accounting profession. *Journal of Business &Amp; Economics Research (JBER)*, *6*(2). https://doi.org/10.19030/jber.v6i2.2392

Naim, M. F., & Lenka, U. (2018). Development and retention of generation y employees: a conceptual framework. *Employee Relations*, *40*(2), 433–455. https://doi.org/10.1108/er-09-2016-0172

Okechukwu, W. (2017). Influence of training and development, employee performance on job satisfaction among the staff. *Journal of Technology Management and Business*, *4*(1).

Parker, S. K. (2014). Beyond motivation: Job and work design for development, health, ambidexterity, and more. *Annual Review of Psychology*, *65*, 661–691. https://doi.org/10.1146/annurev-psych-010213-115208

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, *12*(85), 2825–2830.

Perryer, C., Jordan, C., Firns, I., & Travaglione, A. (2010). Predicting turnover intentions: The interactive effects of organizational commitment and perceived organizational support. *Management Research Review*, *33*(9), 911–923. https://doi.org/10.1108/01409171011070323

Priyam, A., Gupta, R., Rathee, A., Srivastava, S., & Abhijeeta. (2013). Comparative analysis of decision tree classification algorithms. *International Journal of Current Engineering and Technology*, *3*(2), 334–337.

Punnoose, R., & Ajit, P. (2016). Prediction of employee turnover in organizations using machine learning algorithms: A case for extreme gradient boosting. *International Journal of Advanced Research in Artificial Intelligence*, *5*(9), 22–26. https://doi.org/10.14569/ijarai.2016.050904

Purohit, M. (2016). A study on - employee turnover in IT sector with special emphasis on wipro and infosys. *Journal of Business and Management*, *18*(4), 47–51. https://doi.org/10.9790/487X-1804014751

Rodriguez-Galiano, V. F., Ghimire, B., Rogan, J., Chica-Olmo, M., & Rigol-Sánchez, J. (2012). An assessment of the effectiveness of a random forest classifier for land-cover classification. *ISPRS Journal of Photogrammetry and Remote Sensing*, *67*, 93–104. https://doi.org/10.1016/j.isprsjprs.2011.11.002

Sacco, J. M., & Schmitt, N. (2005). A dynamic multilevel model of demographic diversity and misfit effects. *Journal of Applied Psychology*, *90*(2), 203–231. https://doi.org/10.1037/0021-9010.90.2.203

Saleem, R., Mahmood, A., & Mahmood, A. (2010). Effect of work motivation on job satisfaction in mobile telecommunication service organizations of pakistan. *International Journal of Business and Management*, *5*(11), 213–222. https://doi.org/10.5539/ijbm.v5n11p213

Saradhi, V. V., & Palshikar, G. K. (2011). Employee churn prediction. *Expert Systems With Applications*, *38*(3), 1999–2006. https://doi.org/10.1016/j.eswa.2010.07.134

Seddik, A. F., & Shawky, D. M. (2015). Logistic regression model for breast cancer automatic diagnosis. *2015 SAI Intelligent Systems Conference (IntelliSys)*. https://doi.org/10.1109/intellisys.2015.7361138

Seth, M., & Sethi, D. (2011). Human resource outsourcing: Analysis based on literature review. *International Journal of Innovation, Management and Technology*, *2*(2).

Sexton, R. S., McMurtrey, S., Michalopoulos, J. O., & Smith, A. M. (2005). Employee turnover: A neural network solution. *Computers & Operations Research, 32*(10), 2635–2651. https://doi.org/10.1016/j.cor.2004.06.022

Shanmugam, R., & Giri Babu, N. (2016). Assessment of employee attrition among IT employees. *International Journal of Applied Engineering Research, 11*(5), 3449–3453.

Shaw, J. D., Delery, J. E., Jenkins, G. D., & Gupta, N. (1998). An organization-level analysis of voluntary and involuntary turnover. *Academy of Management Journal, 41*(5), 511–525. https://doi.org/10.2307/256939

sklearn.ensemble.GradientBoostingClassifier. (n.d.). Scikit-learn. Retrieved May 26, 2023, from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html

sklearn.ensemble.RandomForestClassifier. (n.d.). Scikit-learn. Retrieved May 15, 2023, from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

sklearn.tree.DecisionTreeClassifier. (n.d.). Scikit-learn. Retrieved May 12, 2023, from https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier

Storey, D. J. (2016). *Understanding the small business sector (routledge library editions: Small business)* (1st ed.). Routledge. https://doi.org/10.4324/9781315544335

Stovel, M., & Bontis, N. (2002). Voluntary turnover: Knowledge management – friend or foe? *Journal of Intellectual Capital*, *3*(3), 303–322. https://doi.org/10.1108/14691930210435633

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (Vol. 16). MIT press.

Thaseen, I. S., Kumar, Ch. A., & Ahmad, A. (2019). Integrated Intrusion Detection Model Using Chi-Square Feature Selection and Ensemble of Classifiers. *Arabian Journal for Science and Engineering*, 44(4), 3357–3368. https://doi.org/10.1007/s13369-018-3507-5

Vasantham, S. T., & Swarnalatha, C. (2015). Need and importance of employee retention. *International Journal in Management & Social Science*, *3*(8), 415–417.

Von Hippel, C., Kalokerinos, E. K., & Henry, J. D. (2013). Stereotype threat among older employees: Relationship with job attitudes and turnover intentions. *Psychology and Aging*, *28*(1), 17–27. https://doi.org/10.1037/a0029825

Woolf, B. P. (2009). *Building intelligent interactive tutors: Student-centered strategies for revolutionizing e-learning* (Vols. 221–297). Morgan Kaufmann. https://doi.org/10.1016/B978-0-12-373594-2.00007-1

XGBoost Parameters. (n.d.). XGBoost Documentation. Retrieved May 30, 2023, from https://xgboost.readthedocs.io/en/stable/parameter.html#

Ye, Q., Zhang, Z., & Law, R. (2009). Sentiment classification of online reviews to travel destinations by supervised machine learning approaches. *Expert Systems With Applications*, *36*(3), 6527–6535. https://doi.org/10.1016/j.eswa.2008.07.035

Zhang, Y. (2016). A review of employee turnover influence factor and countermeasure. *Journal of Human Resource and Sustainability Studies*, *04*(02), 85–91. https://doi.org/10.4236/jhrss.2016.42010

Zhao, Y., Hryniewicki, M. K., Cheng, F., Fu, B., & Zhu, X. (2018). Employee turnover prediction with machine learning: A reliable approach. *Proceedings of SAI Intelligent Systems Conference*, *869*, 737–758. https://doi.org/10.1007/978-3-030-01057-7_56