NOVA IMS Information Management School

MGI

Mestrado em Gestão de Informação
Master Program in Information Management

Predicting Account Receivables Outcomes with Machine-Learning

Susana Lopes da Costa Rebelo

Project Work presented as partial requirement for obtaining the Master's degree in Information Management

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

NOVA Information Management School

Instituto Superior de Estatística e Gestão de Informação

Universidade Nova de Lisboa

PREDICTING ACCOUNT RECEIVABLES OUTCOMES WITH MACHINE-LEARNING

by

Susana Lopes da Costa Rebelo (M20180146)

Project Work presented as partial requirement for obtaining the Master's degree in Information Management, with a specialization in Knowledge Management and Business Intelligence

Advisor: Professor Doutor Roberto Henriques

Co Advisor: Mestre Miguel Batista

July 2021

ACKNOWLEDGEMENTS

This work is dedicated to my family who has always shown immense support in anything I do.

I would like to thank my advisor Roberto Henriques for advising me throughout this project, reviewing my work and letting me present it.

To Miguel Batista, my co-advisor, thank you for introducing me to interesting topics, for guiding me while developing ideas, and for always being there when I needed. And to my co-worker and friend Joana, thank you for helping me throughout this project with precious advice and a revision of my work.

To my parents, thank you for letting me get the education most kids would only dream about. Thank you for always being there to tell me I can do much more than I usually think I can. This thesis would not be written if it was not for you.

To my friends, who have always been by my side to support me, even when away because of the pandemic, and whom I miss so much.

And finally, to Nuno, for supporting me, for always putting a smile on my face, and for being my companion in this crazy journey that is life.

ABSTRACT

The Account Receivables (AR) of a company are considered an important determinant of a company's Cash Flow – the backbone of a company's financial performance or health. It has been proved that by efficiently managing the money owed by customers for goods and services (AR), a company can avoid financial difficulties and even stabilize results in moments of extreme volatility. The aim of this project is to use machine-learning and data visualization techniques to predict invoice outcomes and provide useful information and a solution using analytics to the collection management team. Specifically, this project demonstrates how supervised learning models can classify with high accuracy whether a newly created invoice will be paid earlier, on-time or later than the contracted due date. It is also studied how to predict the magnitude of the delayed payments by classifying them into interesting, delayed categories for the business: up to 1 month late, from 1 to 3 months late and delayed for more than 3 months. The developed models use real-life data from a multinational company in the manufacturing and automation industries and can predict payments with higher accuracy than the baseline achieved by the business.

KEYWORDS

Cash Flow; Account Receivables; Machine-Learning; LightGBM

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

LIST OF ACRONYMS AND ABBREVIATIONS

AR Account Receivables

CRISP-DM Cross Industry Standard Process for Data Mining

KDD Knowledge Discovery in Databases

SEMMA Sample, Explore, Modify, Model, Assess

OvA One-vs-All

OvO One-vs-One

ML Machine-learning

LR Logistic Regression

DT Decision Trees

KNN K-Nearest Neighbors

SVM Support Vector Machines

RF Random Forests

GBM Gradient Boosting Machines

SMOTE Synthetic Minority Oversampling Technique

LGBM LightGBM

1. INTRODUCTION

This project describes some of the work performed on the Analytics and Business Intelligence internship at Siemens. The project's goal was to provide a predictive analysis of the company's payment data using state-of-the-art analytic tools and deliver actionable insights and values to the clients. The analytics team at Siemens -who integrated the student - is an important asset within the company, that provides various data analytics solutions to make informed business decisions. Because Siemens' portfolio is vast, the team takes projects from internal customers from diverse areas and various data types, such as financial transactional data, industry-related and others. Hence, the type of problems the team tries to solve are varied and directly related to the customer's needs.

For this reason, the team is composed of Data Science professionals with backgrounds and skills such as computer and biomedical engineering, physics, and business. The projects developed during this internship were to support the financial and management business areas for Siemens' entities in countries such as Portugal, The Netherlands, Poland and Canada. Even though Siemens is a multinational company, an essential key feature to the success of these projects is the effort of close contact with the customers that are the domain experts and with other IT teams, namely database experts. In the context of the Covid-19 pandemic and as part of the 2020's corporate goals, Siemens invested heavily in increasing the acceptance by its workforce on digitalization of the workplace. This project was also a part of this effort to combine less data-driven and IT heavy departments into collaborating with ones with more expertise to increase data literacy improving essential business processes – in this case the order-to-cash process – with new technology trends.

Organizations such as Siemens have continuously gathered more data on their businesses. The need to analyze data and use it to become better at making decisions has never been so crucial (Abbott, 2014). The customers of this project were several business areas in Siemens from different countries, who have demonstrated interest in improving their financial reports by taking advantage of on-going trends for digitalization and the team's expertise with machine-learning. More specifically, several times our customers manually overestimated and underestimating the capacity of collection of accounts of some of their customers, i.e., wrongly predicting if the customers were going to pay their obligations on-time or not. This led to several problems when providing reliable estimates of collections and performing actions to remind the faulty payments. Overall, it was trusted that with more sophisticated data analytics techniques, the information provided on the accounts receivables spectrum there could be improved, and these processes optimized.

The main goal of this project was to use machine-learning techniques with transactional data of individual invoices to predict the behavior of the customers towards their payment obligations - the account receivables of a business. This implies predicting whether the customers would pay earlier, on-time or later than the payments are due. The motivation of this project was to help the business areas at Siemens that required a service to gather more accurate insights on accounts receivables. Consequently, finding a suitable model based on historical data for increasing the accuracy of the accounts receivables turnover prediction is of utmost importance for the company to improve the management of the collections. To know beforehand what would be the outcome of a particular invoice is the actual value this works will provide to the business. This way, the cash collectors will move closer to deciding the correct strategy to collect such invoices based on the insight. For example, if at the issuing time of an invoice, the model flags it as probably going to be paid later than its due

date, the collecting strategy should be adjusted to be stricter. This could be achieved by having the cash collector calling more often the customer with that invoice for reminding, for instance.

The contribution of this work reflects significantly on specific topics:

- Providing a new approach for analyzing invoices and identifying customer behavior more automatically than before;
- Make use of existing historical data for machine-learning model to identify such behaviors - providing value with data resources previously not used to a great extent;
- In depth analysis of the current state of the accounts receivables related data;
- Comparison of several algorithms to the problem at hand;
- Comparison of different types of classification problems according to different business objectives;
- Overall development of an interesting framework with the ability to scale to other countries and use cases.

This project aims to solve this problem using widely accepted data mining methodologies and visualization techniques. For data cleaning and preparation, methods of identifying outliers and treating missing data were used. Also, various transformations on the dataset were performed to guarantee it would provide information for the model. When data was prepared, predictive techniques were used to classify payments' timing class. The methods and their performance on the data were compared amongst each other, and then the most adequate methods were discussed.

This project stands out as it was intended to be flexible enough for easy scalability to other countries. When it was being developed, internal Siemens clients demonstrated increased interest. These clients were not data scientists but were increasingly interested in understanding how machine-learning projects develop. For this reason, well-known and tested techniques were used being their business value was assessed on every phase of the project.

From a technical standpoint the objectives drawn for this project were the following:

The first phase comprised data extraction, analysis, and preparation: After understanding the business requirements, the company's database was assessed, and the necessary scope of data necessary was selected. An exploratory data analysis was performed in order to provide a more detailed description of the data. The dataset was also interpreted and cleaned. This included using data cleaning techniques such as identifying outliers, treating missing values, and reporting unusual findings. Finally, the data was transformed by changing features to more acceptable formats and generating new ones.

A second phase comprised the experimentation and comparison of several machine-learning algorithms for the binary and multiclass classification problem. Upon this phase, the best performing model was selected and prepared for deployment.

This document comprises five sections: introduction, literature review, methodology, results, and conclusions. The following section comprises an overview of the existing literature to complement the

data on predicting financial indicators and Cash Flow. The methodology section details the processes and techniques used throughout the project and is followed by the results section, where models are evaluated, and the results compared. Finally, the conclusions are addressed along with some limitations encountered, and future work is discussed.

2. LITERATURE REVIEW

2.1. CASH FLOW AND ACCOUNT RECEIVABLES COLLECTION MANAGEMENT

Studying the methods to evaluate corporate financial performance, Profit, and Cash Flow are some of the most notorious metrics. At a corporate level, Cash Flow is a metric that measures the flow of money, as the name indicates. This means, the balance between the money that effectively enters the company (cash inflow) and the money that leaves (cash outflow). It can be thought of as the company's blood because its flow is utterly essential for its survival, and if it runs out, the result is death (or insolvency) (Tracy & Tracy, 2012). Another commonly referenced metric is profit, which calculates the revenues subtracted by expenses; it indicates whether the company is running at a profit or a loss. A profit or a positive cash flow cannot alone indicate good health; this is because a company can have profit at the end of the year but negative cash flow. This may mean that sales are increasing, but they are not being paid in time by the customer, resulting in the company not having money to pay the rest of its obligations (Stobierski, 2020).

Money inflows and outflows can be of different categories according to their operating, financial and investment cycles (Vernimmen, Le Fur, Dallochio, Salvi, & Quiry, 2017). In table 2.1 some examples are presented:

	Cash Inflows Examples	***Cash Outflows Examples***
Operations	Receiving a payment from a good or service to a customer	Paying a good or service to a supplier
Investment	Proceeds from disposal of a property or equipment	Payments for a property rent or equipment sold (longer than operating cycle)
Financing	Contracting debt	Debt liquidation

Table 2.1 - Examples of Cash Inflows and Cash Outflows commonly considered when calculating Cash Flow.

One example of the operations cycle could be purchasing tools from a supplier (outflow) to perform factory machinery maintenance services and receiving the payment for this service (inflow) by the customer. To estimate the Cash Flow at a corporate level is to consider all the cash inflows and outflows throughout a period. Considering the operation cycle, one can estimate the Operating Cash Flow, which results only from operating activities, and the same applies to the investment and financing categories (Vernimmen et al., 2017).

The importance of improving the Order-to-Cash process (O2C) is argued to be decisive in order also to improve cash flow management (Wright, n.d.). The O2C process presented in figure 2.1 is characterized by all the actions on the company level ranging from when the potential customer is checked for credit until the payment is received. These steps are common to most companies, and usually and the invoice information is stored in ERP systems (Korotina, Mueller, & Debortoli, 2015). The account receivables management can be located in a sub-process inside the O2C, Invoice-to-Cash (I2C). This process will only start once the invoice is created and sent to the customer and its duration depend on the collection effectiveness. (Zeng, Melville, Lang, Boier-Martin, & Murphy, 2008)

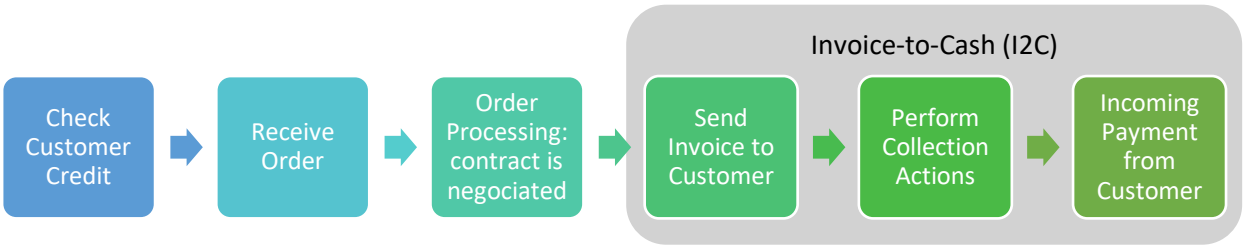


Figure 2.1 - The Order-to-Cash (O2C) and Invoice-to-Cash (I2C) processes based on Wright, n.d..

This is where the account receivables come in, which is the object of this study. Account receivables is the amount of money due to the company for goods or services delivered to customers but not yet paid by them (Investopedia, 2021). Companies grant their customers credit to allow them to pay some days, weeks or even months after the invoice is contracted. This lagging time is a determining factor of the operations cycle length of the company (Vernimmen et al., 2017). In financial transactions theory, the customer is usually compelled to delay the payment as much possible, while the company tries collecting it sooner. The contracts between the two entities are always different and determined by several factors. One example is the credit rating of the customer and seller - a better credit rating of the customer allows it to negotiate better credit and to contract better terms such as cheaper capital and discounts (Pfohl & Gomm, 2009). For this reason, inflows from account receivables usually come later and sometimes they can even default, not entering the cashflow calculation at the expected time. It is important to correctly manage the account receivables to guarantee the good health of the cash flow, thus monitoring receivables and these lagging times are an essential step in estimating cash flow in any industry. Account receivables management in particular, is regarded as one of the most important actions to guaranteeing finance stability at a corporate level (Navon, 1996)

Before moving on, some critical concepts in collection management are reviewed: in a transaction contract, the **payment term** is simply the negotiated number of days for the invoice to be paid (from invoice created date to due date), usually 30, 45 or 60 days, and the **due date** is the maximum date when the invoice should be paid. For instance, if an invoice is created on the 1st of June and the payment term is 30 days, then the due date is July 1st; an **overdue (or late) invoice** is a bill that is late in their payment, i.e., the invoice's due date has passed, and the customer has not yet made the payment. In contrast, **outstanding invoices** are not paid yet but are not late either, i.e., their due dates lie in the future. Invoices can also be **paid in advance** – in other words earlier than the agreed due date. These advance payments usually attribute more risk to the customer who can pay for the service or good but not receive it at the end (Jones, 2018). Some choose to pay in advance for several reasons, namely, to have a good reputation and better relationship with the company.

Most similar studies refer to several collection metrics used to evaluate the collection of account receivables performance. They are also useful to understand important concepts in the present study (Peiguang, 2015; Zeng et al., 2008)

1. Average Delinquent Days (ADD): Average time in days that invoices are overdue. The average difference in days between the invoice's due date and the payment date.
2. Days Sales Outstanding (DSO): Average time in days that invoices are left outstanding, i.e., waiting to be paid but not yet late. It helps to estimate the average time the business takes to collect its invoices. Monitoring the trend of DSO over time is usually helpful to detect any red

flags, and comparing DSOs with the industry average, it is helpful to understand if the company is performing above or below its competitors in collection management (Jones, 2018). The formula to calculate DSO is the following:

$$DSO = \frac{Accounts\ Receivable * Number\ of\ Days\ in\ period}{Total\ Credit\ Sales\ in\ period}$$

Equation 2.1 - Days Sales Outstanding (DSO)

3. Collective Effectiveness Index (CEI): The percentual amount of account receivables collected in a given period, over the total amount of account receivables available to collect in that period (Credit Guru Inc). It informs the collection effectiveness of a company in a given period. For instance, a CEI of 50% in June means that half of the account receivables available to be collected in June were successfully converted into cash. By not considering sales, CEI could be a more accurate collective metric than DSO, especially for those businesses with seasonal spikes in sales.

$$CEI = \frac{(Start\ Receivables + Total\ Credit\ Sales\ in\ period - End\ Total\ Receivables) * 100}{Start\ Receivables + Total\ Credit\ Sales\ in\ period - End\ Current\ Receivables}$$

Equation 2.2 - Collective Effectiveness Index (CEI)

In a typical company, collection management is usually the responsibility of cash collectors. They monitor company's account receivables, typically with manual steps and techniques that do not guarantee efficiency and are prone to error. They tend to overlook smaller customers favoring larger customers or more risky payments, and prioritize invoices with larger values. They take interventive actions which habitually will be corrective rather than preventive due to time scarcity and lack of better systems. Different actions are taken based on the severity of the lateness of the payment. Soft actions like notices of late invoices by email and calls are usually done on the first days of the late invoice. More severe actions like letters of demand are sent a couple of months after the invoice due date has passed. In very severe cases, the debt can be considered lost, and the company will ensure that no more transactions with the same customer are made, sometimes even proceeding to legal action (Cheong, Cheong, & Shi, 2018). The timeline of these actions varies by company, industry and even sometimes customers. Even though less common in most companies, some preventive actions are also taken, such as offering discounts to accelerate payment and negotiating penalty fees for the late payments when the contract is stipulated (Euler Hermes, 2021).

An important tool in collection management are the A/R Aging reports. They are structured tables where the collectors can periodically monitor account receivables; they are useful to keep track of delinquent invoices for an extended period. Usually, ageing buckets are created, for instance, outstanding (not late), 0-30 days overdue, 31-60 days overdue, 61-90 days overdue, 90 + days overdue (Accounting Tools, 2021).

Automating the process of account receivables and try to improve cash flows estimations are important steps nowadays (Bussy, 2018), and indeed, there are already several companies that took the opportunity to develop software for account receivables management. For instance, Oracle sells NetSuite (NetSuite, 2021) with functionalities for tracking accounts and payments, namely invoice

issuing, invoice tracking, automatic actions, and visualization dashboards. However, there is no information on using machine-learning predictions. Other companies such as YayPay by Quadrient (Martin, 2021) already offer consulting services incorporating machine-learning to predict invoice outcomes. However, all of these solutions are usually more costly, require extra time to implement external software, and could be less flexible than in-house projects as they are often part of a more extensive product offering.

2.2. DATA MINING AND MACHINE-LEARNING

Data mining is the extraction of knowledge from data (Witten, Frank, Hall, & Pal, 2016). Nowadays, it is an important topic as the amount of data keeps increasing overwhelmingly. Every personal choice is nowadays recorded, from a simple swipe on an application to the purchase in a supermarket. In companies also financial data is recorded into databases to factory data by the minute hosted on the cloud. It is estimated that around 2,000,000,000,000,000,000 bytes of data are generated in a day across all industries (Conall, 2021). Mining this data "flood" to find important patterns also becomes increasingly challenging but finding the proper techniques is more important than ever. Data Mining is an umbrella term used to describe a set of tasks rather than a specific method to extract these insights from databases. In a business environment, it can start by being just describing existing data with data visualization and statistics to reach valuable conclusions. However, as a company and its data structures mature, it can involve estimation and classification to predict characteristics or even future outcomes. The use of computer programs to process data, and specifically Machine-learning algorithms, are valuable tools to perform some of these data mining tasks (Abbott, 2014; Larose & Larose, 2015).

Computers were for several years perceived as machines that obey specific instructions, but this perception has been changing with the emerging of concepts such as machine-learning. Machine-learning is a technology used to mine knowledge from data, and it concerns the ability of a program to learn patterns from examples or historical data. It is born out of the necessity to automate the data discovery process to make it faster and more accurate, leading even to pattern discovery that would hardly be solved by human capability (Mitchell, 1997). Even so, it is inspired by the human learning experience. Humans tackle a problem by making mistakes, correcting them and, in the process, learning from them. As humans get more life experience and are confronted with different life situations, they adapt their behaviour to perfect their actions. Likewise, a machine-learning program will solve a problem by learning the input given and correcting errors along the way until an optimal level of performance or plateau resulting from increased experience is reached. Machine-learning is a field growing in popularity, not only because of the volume of data (or Big Data) but also due to the major technological advancements. Some examples of these advancements are data storage methods, increasing computing power, accessibility of data manipulation tools like open-source software. Also, it is foreseen that the field will expand in the future as new advances gain pace, namely in the deep learning area (Fradkov, 2020).

The various applications of Machine-learning in real-life problems are considered pivotal in today's society. Pedro Domingos, the author of The Master Algorithm (Domingos, 2015) theorizes that *"[M]achine learning will bring about not just a new era of civilization, but a new stage in the evolution of life on earth."* However, the history of machine-learning (intertwined with Artificial Intelligence) takes back several years prior and has been characterized by fluctuations in popularity. The term machine-learning was first popularized by Arthur L. Samuel in the 50s with the creation of an algorithm to play checkers (Samuel, 1959). Following some ups and downs of acceptance, only in the 90s and 2000s, machine-learning started to be more widely used in various tasks, and most notably in the business world. Today, many companies can create value by leveraging data internally in creative and exciting ways, making it decisive to the business's success. A very well-known case of success is Netflix with their use of data mining to revolutionize the streaming business: namely with the creation of content based on data and a recommendation algorithm with the involvement of the data science

community (J. Bennett & Lanning, 2007; Zhou, Wilkinson, Schreiber, & Pan, 2008). Another example is American Express which uses machine-learning to detect fraud on transactions associated with their credit cards, saving the company millions of dollars in related losses. (Faden, 2021; Foote, 2019; BBVA API Market, 2015)

There are many types of machine-learning algorithms: unsupervised and supervised learning (most popular), semi-supervised (a combination of the two previous), and reinforcement learning.

In **supervised learning**, the model is given the data with the target to learn from these labelled examples from the past. After learning, the model is applied to data that it has not seen before and which is unlabeled, so it provides an estimation or prediction for the label (or target variable), the goal is simply to predict the label or target (Larose & Larose, 2015; Saravanan & Sujatha, 2018). The target can be continuous, for example, predicting the weight of someone or a house price, or categorical, like predicting if an email is spam or not, or if a specific image represents a dog, a cat or a horse. In machine-learning, these problems are called **regression** or **classification**, respectively. This project will focus on classification

Contrarily, in **unsupervised learning,** the model does not receive the labels of the examples. Instead, its goal is to find significant patterns in the input data that will enable the model to conclude its outcome. The most popular form of unsupervised learning is **cluster analysis,** where the model created groups based on the similarity pattern it can learn from the data (Larose & Larose, 2015). This is very often used to perform customer segmentation for marketing campaigns of retails stores, for example.

The potential of applying machine-learning to financial topics is vast and has been long discussed since companies have kept and curating financial records for very long periods. There is also a trend for machine-learning resources to become more broadly available and peer knowledge easily reachable. Given all this, the present study will leverage this potential and use historical data of invoices issued by a company to predict the customer's future behaviour based on the characteristics learnt from the past.

2.3. RESEARCH ON INVOICE OUTCOME PREDICTION

In this section, some successful case studies where machine-learning was applied are reviewed.

In the field of account receivables, as of the time of writing this report, there were few research studies that use machine-learning to predict the outcome on invoice level. Most studies reviewed were limited in the amount of information available about the customer because such data is very sensitive in a corporate environment. For this reason, most studies use aggregate historical variables to characterize the behavior of the customer with invoices in the database, such as: number of paid invoices, amount of delayed paid invoices and the ratio of delayed invoices per customer (Appel, Oliveira, Lima, & Dec, 2020; Peiguang, 2015; Zeng et al., 2008). The exception is the work of Ezvan which works with data from retail (Ezvan, 2018) and which uses features that describe the customer in more details such as number of employees, shop size and mailing options which unfortunately are not so readily available in most company data sources. What all studies have in common is that they all put much importance in constructing features that can describe the historical behavior of the customer, as this is most indicative of invoice outcome.

The reviewed studies used a binary target to classify invoices into on-time or delayed (Appel et al., 2020; Peiguang, 2015) but also a multiclass target that divides the late payments into more than one delayed class (Ezvan, 2018; Peiguang, 2015; Zeng et al., 2008). Of course, the target definition specification differs from study to study due to different business requirements. The most performant algorithms in these studies are ensembles of algorithms based on decision trees, for instance Random Forests (Hu, 2009; Peiguang, 2015), C4.5 (Zeng et al., 2008), LightGBM (Ezvan, 2018) and XGBoost(Appel et al., 2020).

Zeng et. al (2008) proved that using machine-learning to predict invoice delayed outcomes, reduced significantly collection-time. This study analyzed invoices of 4 different companies of diverse industries (tech and marketing). It was able to accurately predict invoices of the four companies with more than 77% of accuracy, reaching the best result of 96% of accuracy for a tech firm using a model that unified the historical data of the four firms and the C4.5 classification algorithm. They could predict new invoices into five classes (one time, 1-30 days late, 31-60 days late, 61-90 days late and + 90 days late). They were additionally using a cost learning technique to avoid misclassifying high-risk invoices (+ 90 days) which are less frequent, using a cost matrix. In the end, they developed a policy for prioritizing invoices by ordering them by their risk of delay (+90 days late being the worst and 1-30 days the best). Consequently, they could exponentially save time on invoice delays when comparing to a typical invoice-to-cash process (Zeng et al., 2008).

Following Zeng et. al (2008) approach, Appel et al. (2020) also developed an invoice prioritization policy. However, this time, with a binary target (on-time and delayed), they associated the probability of being late with the invoice amount, the rationale being, invoices with higher values and more probability of being late are riskier and should be prioritized. In their study, they use a window of time (3 months) to look back when constructing the aggregate historical customer features. These features are created in this way to avoid the effects of their distribution changing as time passes, in other words, to avoid concept drift when deploying the model. Using around 115503 invoices for training (approximately one year worth of data) from a Bank located in Latin America, they were reached up to 77% of accuracy. They pointed to developing a visual analytics tool as future development, to

support collectors visualize the invoice data and prediction results ultimately improving decision-making (Appel et al., 2020).

The focus of the work of Jean-Loup Ezvan (2008) was a five-class classification problem (on-time, 1-15 days, 16-30 days, 31-60 days and +60 days late). A cost learning technique was used to penalize false negatives. It uses the LightGBM algorithm with a customized objective function for the purpose, experimenting even other dimensions to weight the cost such as the invoice value. With around 200000 invoices from a non-tech firm, it archived an f1 score of 83.5% and an f1-score score for the most rikiest payments (+60 days late) of 88.3% (Ezvan, 2018).

In the work of Hu Peiguang (2015) developed for the Massachusetts Institute of Technology (MIT), a binary target (on-time and delayed) and a 4-class target (on-time, within 30 days, between 30 to 90 days and more than 90 days) for classification were compared. With a review of the literature on supply chain finance, the author showed how a predictive analytics problem could improve a company's financial performance. In a dataset with 21000 (roughly three months of data) invoices from a construction company located in Canada. The author started by using the unsupervised learning algorithms Principal component Analysis (PCA) and K-Means Clustering to understand how historical invoices with different outcomes differed from each other. Concluding that even though it was hard, groups of 4 to 5 types of invoice outcome could be identified. After simulating with a group of commonly used algorithms, the best performing one was Random Forests with 81.6% accuracy for the 2-class problem. For the 4-class problem the author noticed that the class imbalance of the dataset was damaging the accuracy in less populated classes. It proposed 2 solutions: attributing to each class a cost proportionally inverse to the class frequency in the random forest parameters; and using cost learning with a cost matrix which assigns cost to false positives. The second method proved to be a better trade-off between overall accuracy and the accuracy of the least frequent class accomplishing 61.9% accuracy to long delays (improving from the initial 51.9%) and overall attaining an accuracy score of 81.2% (Peiguang, 2015)

In her work, Weikun Hu (2009), reviewed other techniques to deal with the class imbalance in predictive modellings, such as traditional sampling methods like random over and under-sampling, and synthetic techniques such as Synthetic Minority Over-sampling (SMOTE) and Random Over-Sampling Examples (ROSE). It explained the importance of using metrics that go beyond accuracy in measuring the performance of a classifier, such as the Receiving Operating Characteristics (ROC) curve and Area Under the Curve (AUC) (Hu, 2009). The final proposed solution was to use a random Forests algorithm with a weighted function to address the imbalance issues, which were characterized by a 2-class target of mostly one-time payments (81.2%). A weighted function of 1:2 proved to be more efficient in predicting, adding a penalty to misclassifications of the less frequent classes, reaching 89% of specificity.

On the other end of the spectrum, we have study by Tater et. al. (2016), which also has as object the prediction of invoice outcomes but focused instead on the account payables. Contrarily to account receivables, account payables are the bills owed by the company to its suppliers for goods and services. The authors also struggled with class imbalance, having a binary classification target composed of 90% of on-time invoices (Tater, Dechu, Mani, & Maurya, 2016). For this reason, they used different metrics such as precision, recall, f1-score, average precision score, and area under the precision-recall curve.

Even though the problem was slightly different from the current study, the features used were similar in a way that importance was given to extract knowledge about the past behavior of the payer.

Similarly, to Appel et. al. (2020), a window of time to look back when computing historical features was also implemented to avoid concept drift and accommodate seasonality. One novel notion was the fact that the payment could be influenced by the number of invoices in queue to be paid at the same time as they could constitute a bottleneck and lead to delays, some features extracted for this purpose were the number of days between invoice dates, for instance. They tested several algorithms in which tree-based ones provided good results, but due to imbalance and the use of a sparse training set, the model with the best results was a Cost-Sensitive First-order Sparse Online Learning with an f1-score of 98,4% (with 84.83% of late invoices correctly predicted) (Tater et al., 2016).

Some limitations of the current studies is that little is known and, in most cases, is presented as future work the tuning of the algorithms (Ezvan, 2018). It should also be noted that the object of all the studies with the multiclass target was to focus on the delayed characteristics, mainly to promote actions to prevent bad customer behaviour. Little is studied about early payments and predicting early invoices for better forecasting and promoting better behaviour. Most studies also admit that the simulation could benefit from more data such as invoice dispute information (Hu, 2009), customer information such as revenue and margins (Ezvan, 2018; Peiguang, 2015). Most business interpretation of late on-time and late differs, which was our clients' case at Siemens. Thus, having customized solutions for different businesses and companies is of the utmost importance.

In table 2.2, a summary of all the reviewed articles related to invoice outcome prediction is presented.

	REFERENCE	TITLE	PROBLEM	ALGORITHMS TESTED
1	**(Zeng et al., 2008)**	*Using Predictive Analysis to Improve Invoice-to-cash*	Multiclass (5 class)	PART ★ C4.5 Boosting Logistic Regression Naïve Bayes
2	**(Hu, 2009)**	*Overdue Invoice Forecasting and Data Mining*	Binary (On-time, Delayed)	Weighted Random Forests ★ Random Forests Neural Network K-Nearest Neighbors Decision Trees
3	**(Peiguang, 2015)**	*Predicting and Improving Invoice-to-Cash Collection Through Machine-learning*	Binary (On-time, Delayed) & Multiclass (4 class)	Random Forests ★ AdaBoost Support Vector Machines Logistic Regression Classification Decision Trees
4	**(Ezvan, 2018)**	*Predicting late payment of an invoice*	Multiclass (5 class)	LightGBM ★
5	**(Appel et al., 2020)**	*Optimize Cash Collection: Use Machine-learning to Predicting Invoice Payment*	Binary (On-time, Delayed)	XGBoost ★ Random Forests Logistic Regression Naive Bayes K-Nearest Neighbors Decision Trees
6	**(Tater et al., 2016)**	*Prediction of Invoice Payment Status in Account Payable Business*	Binary (On-time, Delayed)	Cost-Sensitive First-order Sparse Online Learning ★ LibLinear Boosted Trees Random forest Logistic classification SVM Balanced Bagging Decision Support Vector Machines Balanced Bagging Logistic Regression Balanced Bagging AdaBoost BBGB Neural Network

Table 2.2 - Summary table of Research on invoice outcome prediction

3. METHODOLOGY

3.1. OUTLINE

This project's methodology was developed by following the Cross-Industry Standard Approach Process to Data Mining by the CRISP-DM consortium (Pete et al., 2000). The framework comprises five standard phases of work, which are reflected in this report. It started with the Business Understanding phase, where the context and motivation of the problem were explained. Next, the Data Understanding phase was reflected and a deeper look into the dataset was taken. Following that, comes the description of the various techniques of data preparation used where it becomes clear how the dataset was cleansed and how variables were created and transformed. Finally, the Modelling and Evaluation phases – where the machine-learning algorithms are explained, and so is the respective approach to the modelling problem and the metrics used to evaluate its results. A CRISP-DM's framework requires that each step's outcomes are required at the beginning of the next step (Larose & Larose, 2015) and its application to this project is summarized in figure 3.1, bellow.

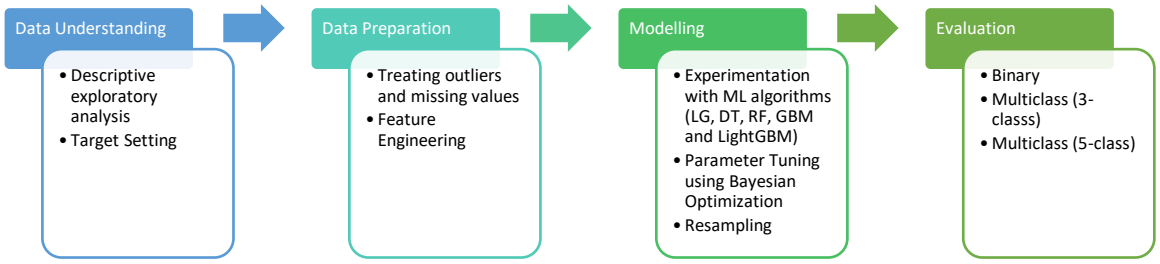


Figure 3.1 - Methodology Outline

This CRISP-DM approach provides the structure necessary for the efficient management of the project among the team members, such as task assignment and documentation (Wirth, 2000). When choosing a framework for data mining, several options were considered such as the KDD, conceptualized in (Fayyad et al, 1996) and SEMMA by the SAS institute. According to Azevedo and Santos (2008), each approach has its own benefits. Both SEMMA and CRISP-DM are inspired in the KDD process and the choice between both is usually context and system-specific (Azevedo & Santos, 2008). Because one of the most important assets was the customer, CRISP-DM proved to be an essential tool to manage customer expectations, - as suggested in literature (Wirth, 2000). This is due to its easy interpretability and familiarity, since most projects in the Siemens Analytics Lab are developed following the same approach.

3.2. DATA UNDERSTANDING

3.2.1. Collecting the Initial data

The dataset was collected from the corporate using an in-memory, column-oriented and relational database – the SAP HANA Data lake. SAP HANA is acknowledged as a good service mainly for transactional query processing in several real-world problems and applications (Hale, 2018). This database consists of tables that are standard to the whole company and which are populated from several ERP systems with transactional data, i.e., data that describes a business event. This centralized service of storing and sourcing data provided by SAP HANA Data lake is common in most team projects, hence the importance of being familiar with SAP standard table fields.

In this case, the project's focus is upon the payment outcome of an invoice emitted by Siemens to a customer, so the level of detail extracted was on the invoice level. An invoice is a document emitted by the seller that describes a product or service being sold. In the dataset, each row will correspond to a transaction or invoice, which naturally has as its main characteristics its value, and the date it was created, the customer that incurred in the cost and the date it is due to be pay. Since the action being studied was the payment of accounts receivables, it was clear that the customer's and invoice's characteristics would be important to leverage the prediction. For this reason, to extract the final dataset, customer master data tables and document level tables are used.

The first raw dataset was queried from Hana Data lake. This subset included not only the required data to be used as input to the model but also other columns important for the business for reporting purposes.

The table below shows some of the Standard SAP tables queried and their respective description according to SAP Learnx (Leanx, s.d.):

Table Name	Description
BSEG	Table that stores accounting records, each line representing one invoice under certain characteristics (expense or earning).
KNA1	Table that stores customer information.
BKPF	Table that stores the headers of the accounting documents present in BSEG.
KNKK	Table that stores credit data on each customer.

Table 3.1 - Description of the standard tables used from SAP HANA retrieved from SAP Learnx

3.2.2. Describing the data

A critical analysis of the dataset was to discuss the meaning and type of variables at hand. Historically, practitioners would classify variables into a few groups. The most popular approach is the phycologist S. S. Stephens' 4 data scales: ratio, interval, ordinal and nominal (Stevens, 1946). In recent years, as the field grew, machine-learning specialists went even further to create variable type subgroups. The 7 Data Types framework (Hale, 2018) qualifies data into seven categories: useless, nominal, binary, ordinal, count, time, and interval. Using this taxonomy proved to be useful for deciding on data preparation options more easily.

This dataset presented several challenges in terms of variables. Firstly, most fields in the dataset were categorical features with many levels, they are nominal data with no numerical relationship between

levels e.g., customer number, customer name, profit centre, payment terms. Most of them are keys that execute a relationship to the tables with their labels. Most of the labels the author was not granted permission to access. The most important numerical feature is VALUE_LC: the value of the transaction in the local currency. There are also some indicative dates such as DUE_DATE, calculated based on the payment terms, and the payment date (AUGDT) used when calculating the target.

Another important notion is that because Siemens supplies exclusively to companies (B2C), when referring to customers, we speak of companies and not individuals. Many machine-learning problems that include customer features will provide variables such as gender and age. However, this is not the case, instead, there are company-specific variables such as credit rating and industry groups.

Attributes	*Invoice nr 5795523420-001-0010-022*
MANDT	022
BUKRS	0010
BELNR	5795523420
BUZEI	001
GJAHR	2017
HKONT	0024110100
ALTKT	0024110000
KUNNR	0000445577
PRCTR	SEM
BUTXT	Siemens Canada Limited
GSBER	20
KNA1_LAND1	CA
KNA1_ORT01	WINNIPEG
REGIO	MB
KUKLA	NG
INDUSTRY	4663
KTOKD	Z333
KNA1_NAME1	[redacted]
KNA1_NAME2	null
ZTERM	Z215
VTEXT	Aff. Distributor with Discount
FDTAG	2017-05-25 00:00:00
AUGDT	2017-08-15 00:00:00
CPUDT	2017-05-25 00:00:00
BUDAT	2017-05-25 00:00:00
DUE_DATE_SOURCE	ZBD1T
DUE_DATE	2017-06-15 00:00:00
VALUE_LC	18594
VALUE_EUR	12364
LOCAL_CURRENCY	CAD
SKFBT	17708.56055
BLART	RV
SHKZG	H
KLIMK	2200000
CTLPC	500
RATINGMETHOD	AU14
HISTORICRATING	R4-
CURRENTRATING	R4-
DBEKR	1750000
KDGRP	PF-BAROB
GRUPP	
DATASET	CA

Figure 3.2 - Example of an invoice (with masked numbers for data privacy)

3.2.3. Target Setting

For this problem, the target had to be calculated using existing fields from the extracted dataset. The main goal is that the machine-learning models should be able to predict the invoice's outcome. This means that when a new invoice is registered in the system, the model should predict, based on certain invoice's characteristics, whether that invoice will be paid on-time, later than the date it is due or earlier. It would also be interesting to know, in the case where the invoice is late, how much delay will the customer incur with the invoice.

In this project, several options of target constructions were explored. The main rationale behind them was to classify the outcome of an invoice and make a classification task for the algorithm. In the end, three approaches were investigated: a binary classification problem (delayed, not delayed) and two multiclass classification problems (in the case of on-time, late and early and late payment deals).

In the binary outcome case, it was evaluated whether the invoice will be late or not, so the model will try to classify each invoice into two classes: class 0 if the invoice payment is predicted not to be delayed and one if the invoice payment is predicted to be delayed. This is a crucial problem to investigate because it will provide more ways of measuring the error from different perspectives when compared to the multiclass problem. This is addressed in a specific dedicated section for the metrics used (section 3.4.1.3 Evaluation Metrics). However, it will not go into much more detail. For instance, it will not tell the level of delay the invoice will be paid late if it is in fact, delayed. The column for the binary case is referred to as TARGET_2C.

Both from literature and authentic experiences, on-time payments can be considered in different manners. Some consider it as those payments paid on the date of the due date (Zeng et al., 2008), but others consider a window of days after which are still considered as on-time and are "forgiven" to the customer (Appel et al., 2020). The difference behind the different types of definitions of "on-time" is usually related to the processing time of the payments (Appel et al., 2020). It might be related to the company itself, the nature of the business or its country. In this project, an on-time payment was considered with a window of 6 days – an on-time payment is considered if it falls three days before due until three days after the due date. This was thought of as a good definition as it would cover customers with invoices due weekends and possibly some holidays and festivities too. This definition was consulted with and approved by cash collectors at Siemens who referred to conduct a similar approach depending on the type of customer as well. With no data to back a different treatment to different customers, this default was set, and the standardization was considered an important best-practice.

In the multiclass problem with three classes, the goal is to classify the invoices early, on-time or late. Compared to the binary case, a level of complexity is added by splitting the not delayed class into on-time or early payments as both do not delay in payment but are different. It was important for the customer to identify those likely to pay earlier than the due date on the contract to define collection strategies that would reprimand bad payers and award good payers to motivate good payment behavior. It is important to recognize the best payers to guarantee their fidelity to the company when negotiating contracts or simply when communicating with them. The column for the multiclass case with three classes is referred to as TARGET_3C.

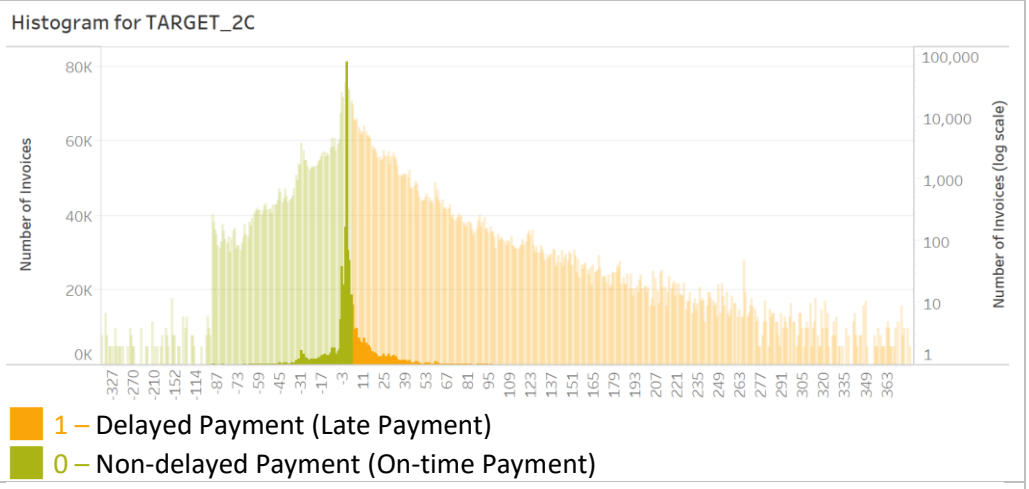
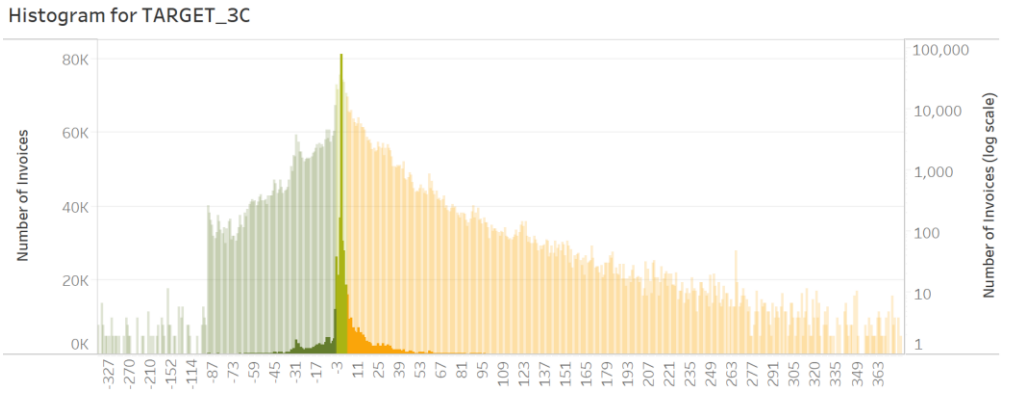
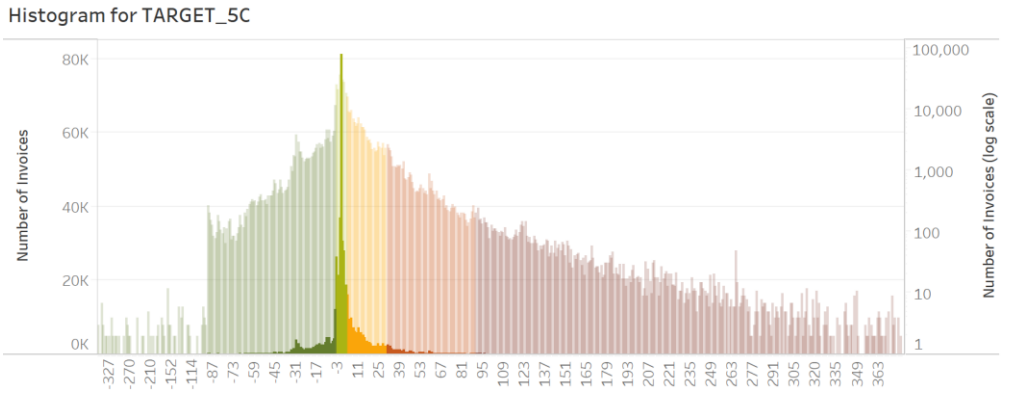
A **Binary Outcome**	 1 – Delayed Payment (Late Payment) 0 – Non-delayed Payment (On-time Payment)
B **Multiclass - 3 Classes**	 -1 – Early Payment 0 – On-Time Payment (3 days before until 3 days after due date) 1 – Late Payment
C **Multiclass – 5 Classes**	 -1 - Early Payment 0 – On-Time Payment (3 days before until 3 days after due date) 1 – Late Payment - Delayed until 1 month 2 – Very Late Payment - Delayed from 1 to 3 months 3 – Critically Late Payment - Delayed for more than 3 months

Table 3.2 - Setting the target for the three classification problems

Lastly, for the multiclass problem with 5 classes the payment classification was divided into 5 classes according to the level of delay of an invoice. This type of target was created to try to answer the question of how long the payment of an invoice will be delayed. The invoices can be classified into –1 for early payments and 0 for on-time payments similarly to the multiclass problem with two classes.

Basing on definitions from similar studies in literature, accounts receivables management best practices and using input from the business, the delayed are in turn classified into late (1) if the payment falls between 3 to 30 days after the due date; very late (2) if the payment fall late between 31 to 90 days after the due date; and critically late (3) if the payment falls late 91 days and onwards. The column for the multiclass case with five classes is referred to as TARGET_5C

As classes in a problem increase, higher accuracy is harder to achieve compared to binary classification or problems with fewer classes (Thrampoulidis, Oymak, & Soltanolkotabi, 2020). Considering also that one of the requirements of the company was to include the class of early payments, this 5-class target with three severity levels within the late payments, was considered a good trade-off. It was also considered to meet the stakeholders' needs since they generally evaluate the status of accounts receivables monthly and at the beginning of each month when the financial statements have just closed. Table 3.2 summarizes the three targets considered in this work and illustrates the distribution of the variables that provides the difference in days between the payment date and the due date of the invoice, classified by the three different targets using a color code technique.

3.2.4. Exploratory Data Analysis

The object of the dataset used in this project is the Account Receivables from the Canadian offices of Siemens Inc. In the dataset were considered **529428 closed** invoices created and paid between October of 2016 and August of 2020. Most of these invoices (66%) had a positive outcome, having **20% paid earlier** and **46% paid on-time**. Only 34% of the payments were delayed. Depicted in figure 3.3 are the the distributions of the three constructed targets, by the number of invoices.

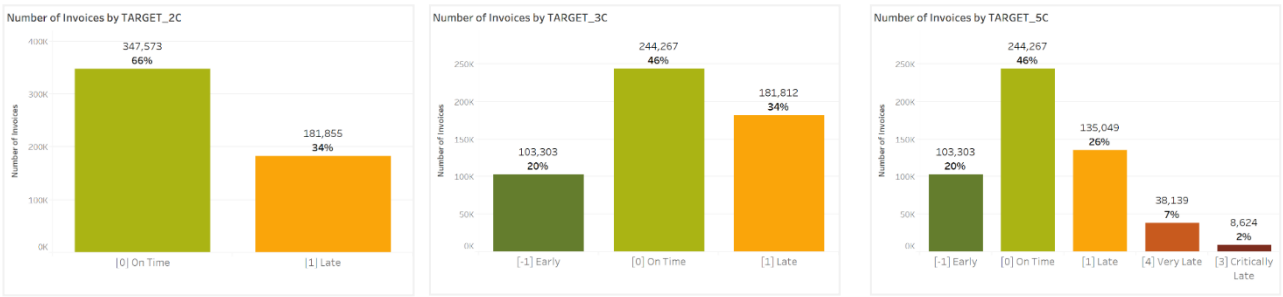


Figure 3.3 - Distribution of closed invoices for each target

Out of these 34% delayed payments, 26% were up to 1 month late, 7% were between 1 to 3 months late, and only 2% were delayed for more than three months.

The distribution of the days delayed (presented in figure 3.4) shows that most invoices are usually delayed for a short period, with the median being around 23 days of delay. Invoices that are delayed for longer are much rarer, but they can get delayed by some significant number of days. These invoices are the most problematic and therefore called 'critically late' invoices.

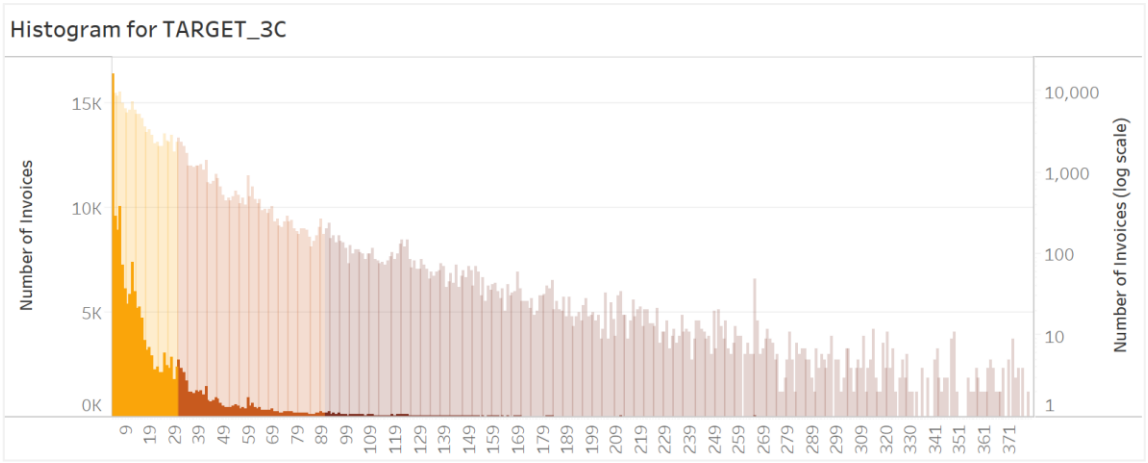


Figure 3.4 - Histogram of the delayed payments

Over these years, the number of created invoices has remained steady, as shown in figure 3.5. There was a slight upward trend during 2017 and 2018, which became less noticeable in the following years. The month of December is usually the weakest in terms of invoices created, possibly because of the holiday season.

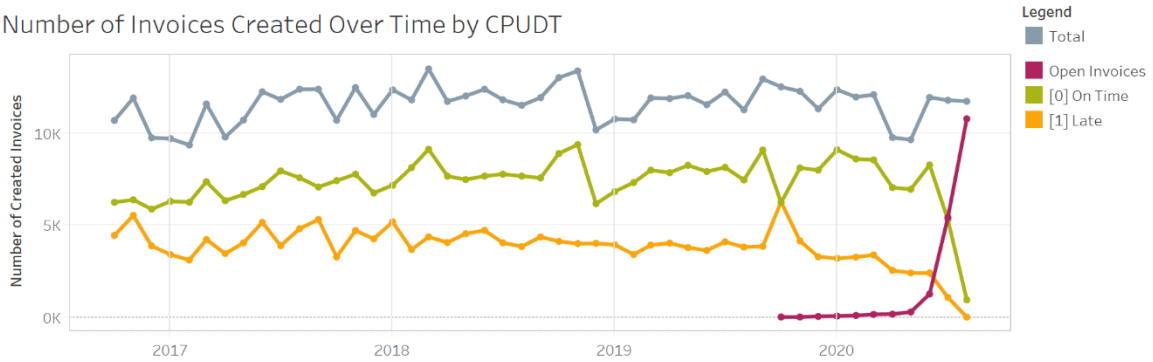


Figure 3.5 - Number of invoices created over time

Last year, we can see a significant decrease in the total invoices created in April and May 2020, with a recovery in the following months. These months corresponded to a period when most countries - including Canada - were in strict lockdown due to the COVID-19 pandemic declared on March 11th. Travel restrictions, factories shut, and overall instability in stock markets (Narayan, Phan, & Liu, 2021) resulted in a slowdown of most businesses including Siemens' and consequentially a decrease in orders. However, this did not result in any major change in the payment behavior – most invoices created in these months were paid mostly on-time (around 70%).

Even though the rule is that the delayed class is the minority as around 60-70% of the invoices are paid on-time, there is an exception in October 2029, where 50% are paid on-time. In the last months, the total number of invoices created includes invoices that were already paid (by August 2020) and thus had an outcome like paid on-time or late; and open invoices that were not yet paid. The latter has no part in the dataset used for building and training the model but make sense to justify the drop in the number of closed invoices.

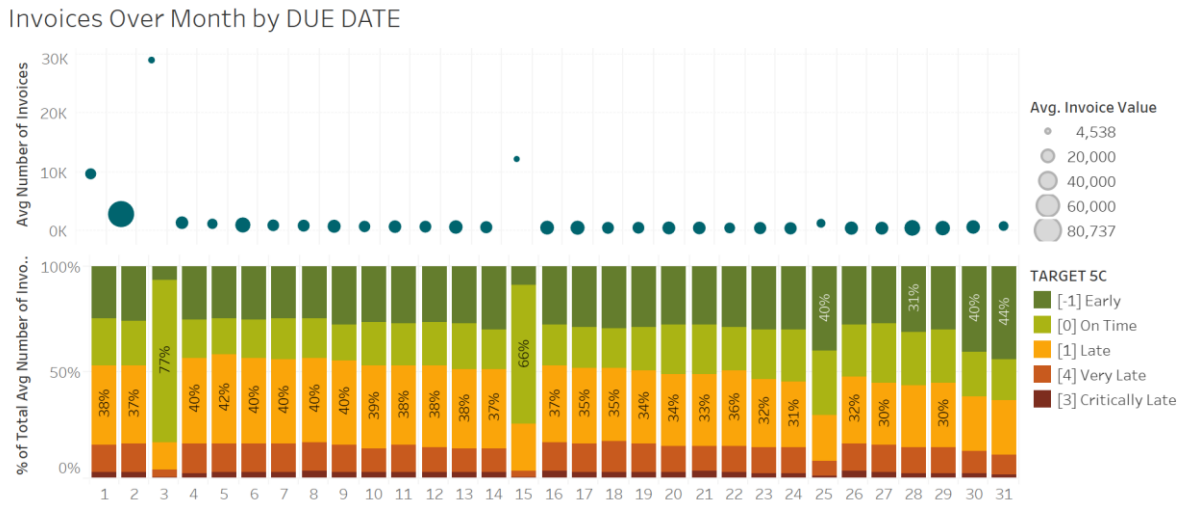
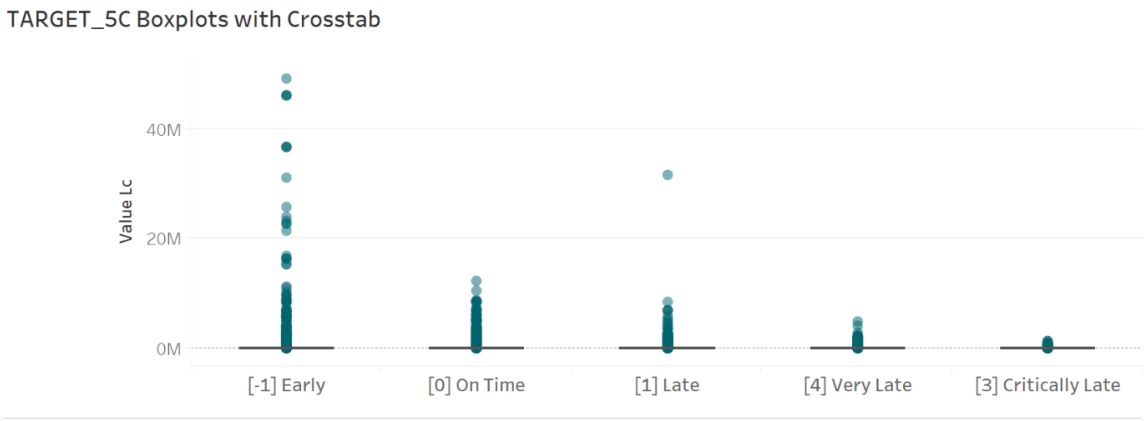


Figure 3.6 - Invoices Over Month ordered by Due Date.

Displayed figure 3.6 we can see an aggregated view of the invoices created between October of 2016 and August of 2020 on each day of the months of their due date. On the top chart, the size of the circles represents the average invoice value for each day, and below is the distribution of the target with five classes in each day and the percentage of the total number of invoices. On average, more invoices are due at the beginning of the month than at the rest of the month. There are three significant peaks – there are considerably more invoices due on average on the 1st, 3rd and 15th day of the month, counting with an average of 28961, 12121 and 9599 due in each day invoices on average, respectively. On those days, the invoices due are not very voluminous in value, with an average invoice value below 15000 CAD. On the rest of the days, the average number of invoices is relatively steady and mostly below 1000 invoices a day. There is a slight increase in the average number of invoices in the last days of the month, only noticeable when excluding the three beforementioned peaks. Also, interesting to note is that the invoices due on the 3rd and 15th days of the month are 77% and 66% paid on-time, respectively. In the rest of the days, most invoices are paid late (ranging from 30 to 40%). At the end of the month, a majority of early payments was registered.

The distribution of the invoice value is highly skewed, meaning that there are some invoices with extremely large values, as can be seen in figure 3.7. For each of the five classes, the invoices have a median invoice value ranging between 1000 to 2000 CAD. Half of the invoices paid early, had an invoice value lower than 1835 CAD. However, the average amount was 25281 CAD which is much higher than the rest of the classes, which stay around 8000 to 10000 CAD. This is a result of most extreme invoice values belong to the 'Early payments' target class. Even so, the distribution of the Invoice Value is very skewed to the right. One might think that customers would try to delay invoices with a larger value. However, data tells us the contrary - the invoices with larger values are usually paid earlier rather than later.



	[-1] Early	[0] On Time	[1] Late	[4] Very Late	[3] Critically Late
Max	49,048,980	12,199,164	31,427,832	4,791,207	1,177,347
Upper Whisker	16,337	8,095	9,559	12,044	13,753
Upper Quartile	6,917	3,445	4,041	5,058	5,753
Average	25,281	8,170	9,355	10,072	9,757
Median	1,835	1,078	1,280	1,574	1,738
Lower Quartile	565	346	420	531	551
Lower Whisker	1	1	1	1	1
Min	1	1	1	1	1

Figure 3.7 - Key statistics and Boxplots for problem for the problem with 5-Classes

The payment term should be an important variable because it indicates the contracted time buffer between creating the invoice and the date it should be paid (due date). If all the customers paid precisely on the date they have contracted, it would be an easy task to predict every invoice outcome with the payment terms. However, since this does not describe real life, we can use the payment term as an indication of payment and assume most customers will try to meet the payment as close to the due date as possible. From Figure 3.8, we note that the vast majority of customers agree on a around a 30-day period to pay the invoice after its creation. Around 20 days and 40 days are also popular payment terms. There are also many customers that agree on an around 90-day period to pay. The rest of the payment terms after 90 are very sparse. One individual invoice goes up to 870 days of payment buffer.

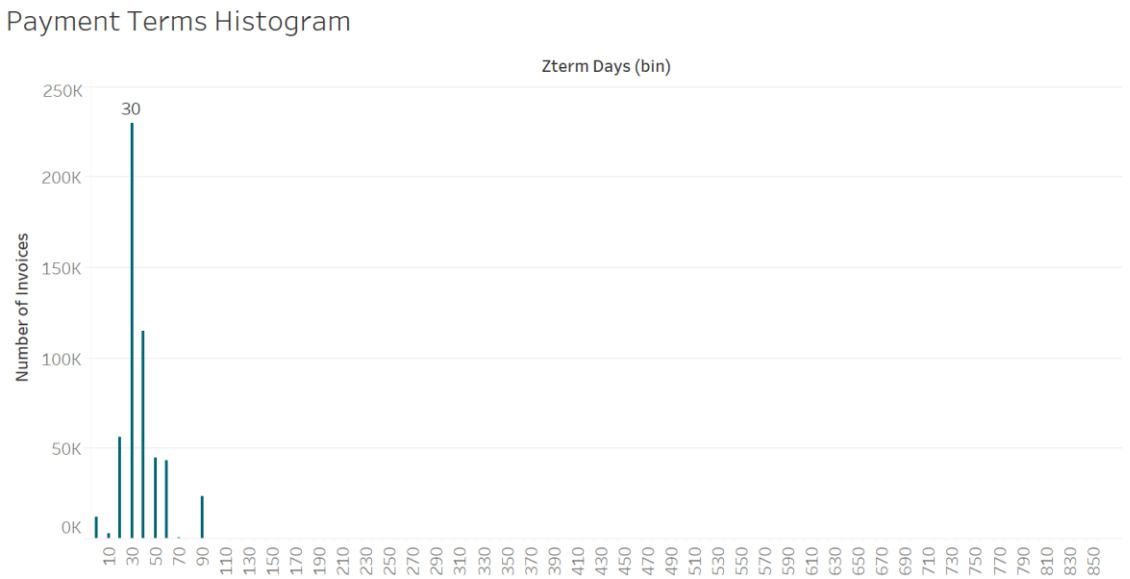
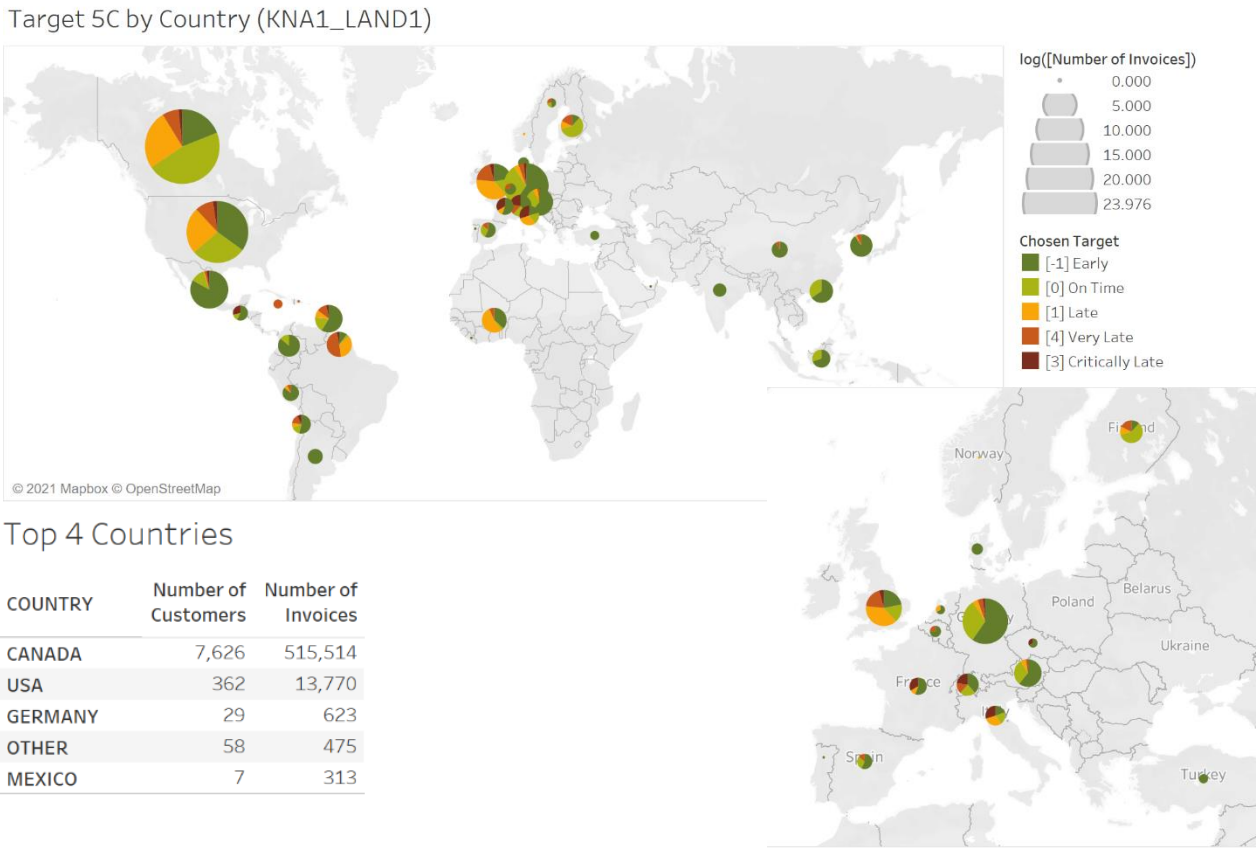


Figure 3.8 - Histogram of Payment Terms

Siemens Inc. Canada counted **8022 different customers** worldwide, even though 95% are Canadian customers. Likewise, many invoices are coming from Canada (515514 invoices) following by the USA, Germany and Mexico. Canada and the USA have a similar distribution of the outcome of the invoices, with the majority being either early or on-time, the smallest class being the critically late class. Mexico's invoice outcome pattern is very different as the largest class of invoice outcomes is early payments. In Europe, Germany is the country with more invoices and customers, which is expected as it is the company's headquarters country. Its invoice outcome pattern is more similar to Mexico's than Canada or the USA, as the largest share of invoices were paid earlier, and the second largest, on-time. The UK, the second country in Europe with the most invoices, shows interestingly that the most significant proportion of their outcome is late, followed by the very late class. France and Switzerland, and Italy show the most significant proportion of critically late invoices than the rest of the European countries.



Top 4 Countries

COUNTRY	Number of Customers	Number of Invoices
CANADA	7,626	515,514
USA	362	13,770
GERMANY	29	623
OTHER	58	475
MEXICO	7	313

Figure 3.9 - Invoice payments by customer country (KNA1_LAND1)

Another important variable for describing customer behaviour is its rating. The customer rating is provided by Siemens Financial Services and consists of an opinion about the company's creditworthiness taking into consideration the country, industry, and financial risk. The Current Rating variable is the rating of the customer that contracted an invoice at the time the invoice was issued. It ranges from R0, which is attributed to those with the strongest capacity to meet financial obligations, to R10, which is the default rating for those whose funds are seen as recovery. In the dataset, there are customers with ratings that are considered adequate (from R5- to R6+). Vulnerable ratings (From R8- to R10), on the other hand, are less common. One hypothesis that can be drawn is that the worst

the rating, the more volatile can the outcome of the invoice be, as customers with a worse rating will have more difficulties in paying their obligations and more scrutiny and pressure from the cash collectors. There is a slight trend for invoices with worse ratings to have a more significant proportion of having a critically late outcome. Also, compared to better ratings, those that are worse have a much smaller proportion of on-time payments but still a considerable proportion of early payment.

The rating with more invoices is R5- with 116817 invoices which are considered a good rating. On the other hand, the rating with more customers is R7+ which belongs to the hardly accepted group.

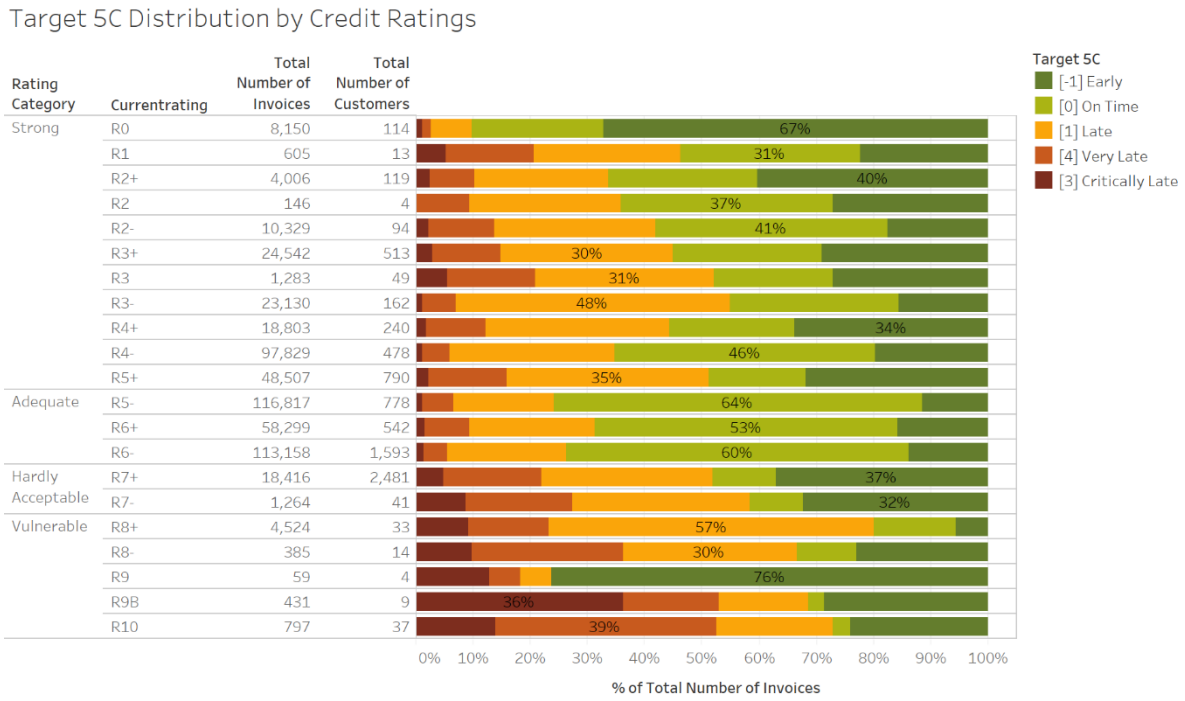


Figure 3.10 - Proportion of invoice outcomes using 5-Class target by Rating Category and Current Rating

3.3. DATA PREPARATION

Preparing the data for the task at hand is one of the most important steps in any machine-learning project. It is also the step that requires the most time and effort. Following ideas from literature (Larose & Larose, 2015), this pre-processing stage aims to address different problems:

- Selecting columns and dropping duplicate rows;
- Treating Missing Values;
- Checking for Outliers;
- Coherence Checking;
- Data Transformation and Feature Engineering.

When the dataset was first extracted, there was limited care when selecting which columns would be included. The aim was to include the biggest number of columns that could be interesting to the problem at first glance of their description and then analyze them one by one. Consequently, it was expected that some columns extracted would be redundant, other would be included merely for reporting purposes and some could be ill-maintained and thus wouldn't be interesting to the predictive model. The initial columns of the dataset are listed below:

	Column Name	*Description*	*Type*	*Candidate?*	*Reason*
1	VALUE_EUR	Value converted in euros	Continuous		Redundant
2	MANDT	Client code	Nominal		ID
3	BUKRS	Company code	Nominal		ID
4	BELNR	Invoice Number	Nominal		ID
5	BUZEI	Invoice Segment	Nominal		ID
6	GJAHR	Fiscal Year	Nominal		ID
7	HKONT	Account Number	Nominal		Irrelevant
8	ALTKT	Group account number	Nominal		Redundant
9	KUNNR	Customer Number	Nominal	x	
10	PRCTR	Profit centre code	Nominal		Too many levels
11	BUTXT	Name of company - explains BUKRS	Nominal		Redundant
12	GSBER	Business Area	Nominal		Irrelevant
13	KNA1_LAND1	Country key	Nominal	x	
14	KNA1_ORT01	City	Nominal		Too many levels
15	REGIO	Region (state, province, country)	Nominal		Too many levels
16	KUKLA	Customer classification information	Nominal		Irrelevant
17	INDUSTRY	Industry key	Nominal	x	
18	KTOKD	Customer account group	Nominal		Irrelevant
19	KNA1_NAME1	Customer Name part 1 - explains KUNNR	Nominal		Redundant
20	KNA1_NAME2	Customer Name part 2 - explains KUNNR and it is the continuation of KNA_NAME1 text	Nominal		Irrelevant
21	ZTERM	Terms of payment	Nominal	x	
22	VTEXT	Payment term text - explains ZTERM	Nominal		Too many levels & Redundant

23	FDTAG	Planning Date	Date		Many Missing & Irrelevant
24	AUGDT	Clearing date	Date		Used in Target computation
25	CPUDT	Day on which accounting document was entered	Date		Redundant
26	BUDAT	Posting date in the document	Date	x	
27	DUE_DATE_SOURCE	Rule category with which the due date was computed with	Nominal		Irrelevant
28	DUE_DATE	Date when invoice is due to be paid	Date	x	
29	VALUE_LC	Value of invoice in local currency (CAD)	Continuous	x	
30	LOCAL_CURRENCY	Local currency (CAD)	Nominal		Irrelevant
31	SKFBT	Amount eligible for cash discount in document in local currency	Continuous	x	
32	BLART	Document type	Nominal		Irrelevant
33	SHKZG	Debit/credit indicator	Binary	x	
34	KLIMK	Customer's credit limit	Continuous	x	
35	CTLPC	Risk category from credit management	Ordinal	x	
36	RATINGMETHOD	Rating method for rating column	Nominal		Too many levels & Irrelevant
37	HISTORICRATING	Previous rating to current rating	Nominal	x	
38	CURRENTRATING	Current rating assigned	Nominal	x	
39	DBEKR	Recommended credit limit	Continuous	x	
40	KDGRP	Credit limit information	Nominal		Too many levels
41	GRUPP	Customer group information	Nominal		>90% Missing values
42	DATASET	Name of the dataset	Nominal		Irrelevant
43	EXTRACTION_TIMESTAMP	Date of extraction of the dataset	Date		Irrelevant
44	CLOSED	Flag that tells if invoice is closed by extraction date	Binary		Irrelevant
45	TARGET_MONTHS	Target in months (buffer between due date and payment date in months)	Discrete		Somewhat used in Target computation
46	TARGET_DAYS	Target in days (buffer between due date and payment date in days)	Discrete	x	Used in Target computation
47	TARGET_2C	Target with 2 classes with window of 3 days early and 3 days late	Binary	x	
48	TARGET_3C	Target with 2 classes with window of 3 days early and 3 days late	Nominal	x	
49	TARGET_5C	Target with 5 classes with window of 3 days early and 3 days late	Nominal	x	

Table 3.3 - List of columns present in the raw dataset

3.3.1. Selecting Data and dropping duplicate rows

Out of 49 columns, only 22 were considered candidates for the feature engineering part. This decision was based on several characteristics of each column. In short:

Redundant columns are those whose value are reflected the same or similarly in another column, or highly correlated. This is the case of:

- VALUE_EUR is redundant because it is the same information as VALUE_LC. VALUE_LC was chosen instead of VALUE_EUR because the latter is not well maintained in the system with 3% of the values missing.
- ALTKT is similar to HKONT, BUTXT represents the same information as BUKRS, KNA1_NAME1 repeats the information in KUNNR, CPUDT is similar to BUDAT, and so on.

Irrelevant columns are those that do not represent valuable information to the problem. They might be useful for giving context to the dataset but should not be considered further to the feature engineering phase. This is the case of:

- Placeholder columns: DATASET, EXTRACTION_TIMESTAMP, CLOSED;
- Variables with one unique value or zero variance: LOCAL_CURRENCY, BLART;
- Other columns with no importance such as: KNA1_NAME2, FDTAG.

Categorical columns with **too many levels** are also irrelevant. Their use will hurt the model performance because there will be fewer examples for each level for training which decreases the change of each level have any impact on the model's decision (Ray, 2015). Some variables with many levels were dropped because they do not represent much value to the problem: KDGRP, RATINGMETHOD, REGIO, KNA1_ORT01, INDUSTRY and PRCTR. INDUSTRY, for instance, seemed very promising until it was found that it was just a key to an industry table which, due to data privacy, the schema did not have access rights to it. With no context, it is not possible to group INDUSTRY into smaller levels, and thus, it was dropped.

It was checked if there were any **duplicate invoices** on the rows level. There were not any so the index was set to the invoice unique identification (INV_ID) which consisted of 5 fields concatenated - BELNR, BUZEI, GJAHR, BUKRS and MANDT.

3.3.2. Treating Missing Values

Missing values are data points in the dataset that are not filled. They can appear for several reasons: they could be missing at random due to failures when entering the data for example, but their absence could also mean something and thus they can be missing not at random (Abbott, 2014). From domain knowledge, there was no reason to think the values missing in our dataset were missing purposely, so they were interpreted as random missing. We also know that from the dataset, missing values are usually represented with an empty cell and not any unique code such as "?", 999, -999, so this helped the analysis.

When the dataset was first extracted, all the rows with missing values in the most critical columns (KUNNR, AUGDT and DUE_DATE) were excluded from the dataset. This is because it was necessary to

guarantee that we could compute the target with the payment date (AUGDT) and due date (DUE_DATE) correctly in all the invoices. Also, each invoice need to had a customer number (KUNNR) since it is the behavior of the customer that will have the most impact on the outcome of the invoice payment.

Also, by this time, the VALUE_LC value was constructed using the field value of the invoice without taxes. Very few times this value, is not available, and in such cases, it was given the VALUE_LC the amount in the field with taxes which is the closest estimate available. This prevents the occurrence of missing values in VALUE_LC.

Some columns were eliminated, as described previously, by their irrelevance and redundancy but an added factor to drop them was also the number of missing values: For instance, GRUPP had more than 90% of the values missing as well as not being highly informative to the problem.

Replacing or imputing the missing values is the best choice when it is possible (Abbott, 2014), but it should be done carefully and it is very dependent on the context. For the rest of the columns with missing values (ZTERM, CTLPC, HISTORICRATING, CURRENTRATING) they were replaced. Because the fields were categorical, the strategy was to replace the missing values with the customer's mode, that is, the value that occurred most often relative to that customer. If the dataset did not provide any invoices of that customer to compute the mode with, then the missing values would be replaced by the column's mode.

For HISTORICRATING and CURRENTRATING it was first checked if the row missing had a value for any of them, so if CURRENTRATING was filled but HISTORICRATING was not, the missing values would be replaced by the current rating, and the other way around. The reasoning behind this decision was that the current rating usually follows the historical rating. It is expected that both fields are the same as the customer's rating does not tend to change much over time. Also, because only 0.01% were missing values in both fields, it was concluded that the effect of this replacement wouldn't have a substantial impact on the main statistics of each column.

COLUMN NAME	% MISSING	MISSING VALUE TREATMENT
CURRENTRATING	0.015%	78 null rows replaced by HISTORICRATING
HISTORICRATING	0.011%	60 null rows replaced by overall HISTORICRATING mode
CTLPC	0.007%	38 null rows replaced by overall CTLPC mode
ZTERM	0.001%	4 null rows replaced by the customer's ZTERM mode

Table 3.4 – Treatment of Missing Values

3.3.3. Checking for Outliers

Outliers are values that differ from the rest of the data. Sometimes they might represent errors inserted into the dataset by mistake, but on other occasions they could just be actual extreme values that simply show variance (Sharma, 2018). The way to distinguish both is usually by domain knowledge (Downey, 2011).

Invoices which payment is delayed or advanced more than one year are rare cases and are usually special contracts that should be considered separately. Hence, it was decided to not include these extreme cases in this problem. So, 486 invoices were dropped from the dataset with this condition.

Likewise, all the payments that have a payment term of buffer (the period between creation date and due date) that is larger than 1 year were deleted.

Most practitioners identify outliers by the Interquartile Range (IQR) method. The IQR is the difference between the third quartile and the first quartile of the distribution and any point is a potential outlier if it is situated more than 1.5 times the IQR from the lower and upper quartiles as a rule of thumb (Abbott, 2014). The following outliers were identified:

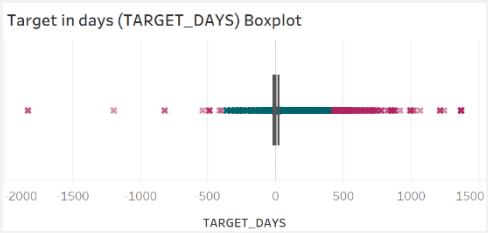
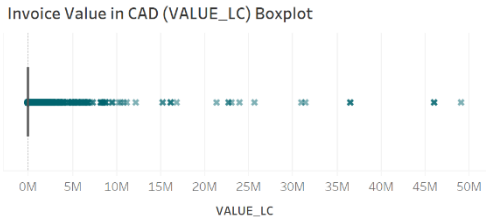
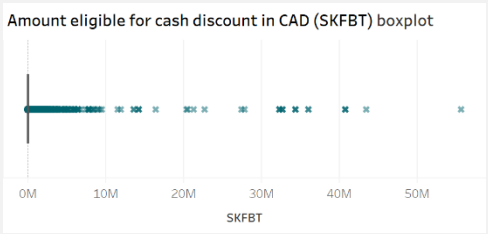
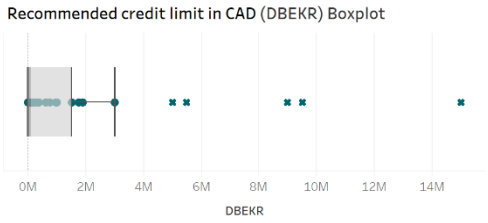
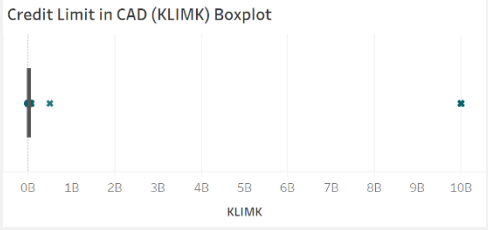
COLUMN NAME	BOXPLOT	NUMBER OF OUTLIERS (IQR)	% OUTLIERS (IQR)	OUTLIER TREATMENT
TARGET_DAYS		108227	20%	**486 invoices** paid with 1 or more year of delay or in advance **dropped**
VALUE_LC		69387	13.1%	Flagged but not removed
SKFBT		68973	13%	Flagged but not removed
DBEKR		72657	13.5%	Flagged but not removed
KLIMK		6124	1.2%	Flagged but not removed

Table 3.5 - Summary table for checking outliers

As the box plots show, in the numerical columns potential outliers squash the IQR because they are such extreme unusual values. Their distributions do not resemble normal distributions, and they are highly skewed to the left. The problem with these outliers is that there are good chances they will harm the model by introducing bias. However, removing the outliers carries the risk of misrepresenting important values. For example, it is expected that some customers, depending on their size, can have bigger orders than their smaller counterparts. There is also a big diversity in the kinds of products and services associated with this seller: what is being sold can range from material part which happens many times and usually at a small value, to a whole renovation of some factory machines which is naturally less and way more expensive. In this sense, these extreme values are justified. Removing these outliers would carry some significant risk as it would not correctly reflect the behavior of all the customers (Abbott, 2014), which is the main goal of the project. As said by Abbot in the example of Fraud Detection, "the very best customer lifetime value represents the very best of unusual behavior". This is the takeaway to justify that all the identified potential **outliers were identified but not removed**. To reduce its effect, these variables were later transformed, and some algorithms used are robust to them.

3.3.4. Coherence Checking

Contradictory values and inconsistencies in the data were also identified. The following changes were performed:

- **Stripped trailing zeros for KUNNR and ALTKT**: some invoices were for the same customer and account but did not match because they were wrongly formatted with trailing zeros.
- The posting date (BUDAT) is the date when the invoice was created in accounting. Another name for it is the invoice date. This date should not come after the due date because the invoice would be due before it was even created. We **dropped all (4141) invoices with this posting date (BUDAT) inconsistency.**
- Instances where **VALUE_LC is less than 1 CAD** are invoices with an extraordinarily low value, so they are likely to be errors hence **these invoices (1207) were also dropped.**

3.3.5. Feature Engineering

The last step of the data preparation stage is feature engineering. In machine-learning, models learn by finding patterns in their input data. That input is what practitioners call the features: an informative representation of the data in numerical form (Zheng & Casari, 2018). Consequently, the feature engineering phase is when the necessary transformations on the data are performed to create features that will supply predictive value that the model can interpret.

When it comes to feature engineering there is "no one size fits all" or cookbook recipe ready. It is a highly dependent task on the data available, the models that are going to be used and the type of task. For ease of comprehension this stage is divided into four parts:

1. Feature Construction and categorical encoding (Historical features; Date features; Payment Terms; Ratings; and Country);
2. Numerical Transformations;
3. Evaluation of correlation and feature selection;

3.3.5.1. Feature Construction

Since many of the columns of the dataset were dropped for having missing or unique values, very few possibly meaningful columns are left to use as features. So, the first approach for feature engineering was to create derivate variables from the existing data.

Historical Features

Being the customers responsible for the invoices' payment, it was important to draw a profile for each to understand their past behavior, as there are many returning customers in the dataset. This is something that, in practice, cash collectors try to do when they get a new invoice and try to guess its outcome – trusting their expertise, they expect that some customers' pay will be better than others' based on their past behavior. However, a cash collector has limited capacity to memorize and research all the history of every customer they have encountered before. To share this information with the model, different aggregate features and ratios for each customer number (KUNNR) were created. This technique is based on calculating frequency counts, and it has been used widely from Kaggle competitions (Rajwanshi, 2018) to academic writing (Appel et al., 2020; Hu, 2009; Peiguang, 2015; Zeng et al., 2008) with promising results. These simple statistics will summarize the behavior of the customer for the algorithm and thus add useful information for predicting.

When computing these features, it was necessary to prevent leaking future information into the model. To make sure no future values were used in the calculations, each feature was computed considering the creation date of the invoice (BUDAT). Consequently, the customer profile was based on the customer number and the date that aggregation was computed. This profile was then joined back to the dataset on the date (BUDAT) and the customer number.

Another consideration was that the behavior of the customers can change with time, so if the aggregations are done with the whole dataset, we can be considering observations that reflect a behavior that is obsolete, which in turn hurts the accuracy of the model – this phenomenon is referred to as concept drift (Brownlee, 2017; Tsymbal, 2004; Webb, Lee, Goethals, & Petitjean, 2017). It was especially important to consider the possibility of concept drift because this dataset was extracted during the COVID-19 pandemic, which could have contributed to some customers changing their behavior resulting from the economic crisis. To tackle this problem, a window was defined to consider the time to look back when calculating these features. This technique was inspired in the work of Appel el al. where several windows were tested in the algorithm and the one chosen was a trade-off between the highest accuracy and the decreasing the chance of overfitting.

For this project, 7 datasets were created to compare different window sizes to calculate the historical features between 6 classifiers and the 3 different targets. The window size or time to consider look back when calculating the historical features were 2 months, 4 months, 6 months, 8 months, 1 year, 1.5 years and 2 years. The accuracy of each model was compared for each different target type and window size. It was possible to confirm that the bigger size of the window for calculating the historical features, the worst the overall accuracy of the 6 models would become, being w=24 the window that got the worst results for the 3 targets. In the end it was decided to use a window of w=6 because that was the size that resulted in the highest accuracy for many of the models and targets. The algorithms for the binary model specifically had better accuracy when w=6 on average. Even though the best performances registered for the multi class problems where for window w=4, it was not a very

significant difference so w=6 was used as the overfitting issue could be avoided by choosing a larger window size (Appel et al., 2020).

Target	*W*	*LR*	*SVM*	*KNN*	*RF*	*GBM*	*LIGHTGBM*	*Average*	*MAX*
Binary	**2**	76.60%	76.60%	73.90%	80.20%	**82.00%**	81.60%	78.50%	82.00%
	4	80.40%	80.70%	75.00%	83.50%	**84.60%**	84.30%	81.40%	84.60%
	6	80.90%	81.30%	74.20%	84.00%	**84.70%**	**84.70%**	**81.60%**	**84.70%**
	8	81.00%	81.40%	72.90%	83.80%	**84.60%**	80.60%	80.70%	84.60%
	12	80.80%	80.00%	75.90%	84.10%	82.40%	**84.10%**	81.20%	84.10%
	18	78.90%	79.20%	72.21%	83.94%	84.08%	**82.40%**	80.20%	82.40%
	24	74.70%	74.60%	69.70%	83.80%	**84.50%**	73.50%	76.80%	84.50%
3 Class	**2**	62.10%	58.80%	59.00%	67.40%	**69.30%**	61.50%	63.00%	69.30%
	4	67.00%	64.30%	60.80%	72.00%	**73.80%**	65.00%	67.10%	**73.80%**
	6	67.50%	68.20%	61.10%	71.40%	**72.30%**	65.30%	**67.60%**	72.30%
	8	67.70%	68.50%	58.70%	70.70%	**71.40%**	64.60%	66.90%	71.40%
	12	66.20%	65.80%	59.80%	**71.10%**	70.50%	64.90%	66.40%	71.10%
	18	66.03%	66.28%	57.11%	**71.46%**	70.74%	56.50%	56.50%	56.50%
	24	62.10%	62.70%	53.50%	69.20%	**71.50%**	61.46%	63.80%	71.50%
5 Class	**2**	57.60%	57.50%	55.00%	63.90%	**65.60%**	58.70%	59.70%	65.60%
	4	64.90%	61.40%	57.20%	68.90%	**69.50%**	61.20%	63.80%	**69.50%**
	6	64.60%	63.70%	58.30%	**68.30%**	68.10%	61.30%	**64.00%**	68.30%
	8	64.90%	63.50%	55.40%	**68.10%**	67.40%	61.00%	63.30%	68.10%
	12	64.00%	63.50%	56.10%	**67.70%**	**67.70%**	55.60%	62.40%	67.70%
	18	62.50%	63.40%	53.60%	67.10%	**67.30%**	51.80%	60.90%	67.30%
	24	58.20%	58.40%	49.50%	66.60%	**67.10%**	61.50%	60.20%	67.10%

Table 3.6 - Testing different window sizes (w) with different algorithms

This technique implied that all the invoices created before the minimum BUDAT plus the number of months of each window size had been dropped, decreasing the train size. This is because all invoices before that date would not get realistic values on the historical features, since there were no w months of data prior to compute the features the same way we did for invoices after that date.

In table 3.7, features 1 to 25 are considered historical features in which they aim to describe the historical profile of each customer in the dataset.

#	FEATURE NAME	FEATURE	DESCRIPTION
1	***NR_PAID***	Number of Invoices (paid)	Number of invoices paid by customer preceding the creation date.
2	***NR_LATE***	Number of Late invoices	Number of invoices the customer paid with delay (more than 3 days after the due date) preceding the creation date.
3	***NR_EARLY***	Number of Early Invoices	Number of invoices the customer paid in advance (less than 3 days before the due date) preceding the creation date.
4	***RATIO_LATE***	Delay ratio	Ratio of late invoices of the customer preceding the creation date. Ratio of NR_LATE over NR_PAID.
5	***RATIO_EARLY***	Early ratio	Ratio of early invoices of the customer preceding the creation date. Ratio of NR_EARLY over NR_PAID.
6	**VAL_TOTAL**	Total Value Paid	
7	**VAL_LATE**	Total Value Delayed	Sum of the value of late invoices by customer preceding the creation date.
8	**VAL_EARLY**	Total Value Early	Sum of the value of early invoices by preceding the creation date.
9	**RATIO_VAL_LATE**	Late Value Ratio	Ratio sum value of late invoices of the customer preceding the creation date. Ratio of VAL_LATE over VAL_TOTAL.
10	**RATIO_VAL_EARLY**	Early Value Ratio	Ratio of the sum value of early invoices of the customer preceding the creation date. Ratio of VAL_EARLY over VAL_TOTAL.
11	**AVG_DAYS_LATE**	Average days paid invoices were late	Average number of days late of invoices paid late by customer preceding the creation date.
12	**AVG_DAYS_EARLY**	Average days paid invoices were early	Average number of days early of the invoices paid early by customer preceding the creation date.
13	**NR_OUT**	Number of invoices that are outstanding	Number of invoices by customer registered preceding the creation date, which are not paid yet (outstanding). Number of invoices that the customer still has to pay.
14	**NR_OUT_LATE**	Number of invoices that are outstanding but already late	Number of invoices by customer registered preceding the creation date, which are not paid yet (outstanding) but are already past due date (late). Number of late invoices that the customer still has to pay.
15	**RATIO_OUT_LATE**	Ratio of late outstanding invoices	Ratio of the late outstanding invoices of the customer preceding the creation date. Ratio of NR_OUT_LATE over NR_OUT.
16	**VAL_OUT**	Total Value of invoices that are outstanding	Sum of the invoice values by customer registered preceding the creation date, which are not past due date yet (outstanding). Total value of the invoices that the customer still has to pay.
17	**VAL_OUT_LATE**	Total Value of invoices that are outstanding but already late	Sum of the invoice values by customer registered preceding the creation date, which are not past due date yet (outstanding) but are already past due date (late). Total value of the late invoices that the customer still has to pay.

18	**RATIO_VAL_OUT_LATE**	Ratio of the total value late outstanding invoices (14/13)	Ratio of the sum value of late outstanding invoices of the customer preceding the creation date Ratio of VAL_OUT_LATE over VAL_OUT.
19	**AVG_DAYS_LATE_OUT**	Average days late of outstanding invoices	Average number of days late of outstanding late invoices by customer preceding the creation date.
20	**STD_DAYS_LATE**	Standard deviation of days of late invoices	Standard deviation of the number of days late of late invoices by customer preceding the creation date.
21	**STD_DAYS_EARLY**	Standard deviation of days of early invoices	Standard deviation of the number of days early of the invoices paid early by customer preceding the creation date.
22	**PAY_FREQ**	Ratio of actual payments	Ratio of invoices paid over all invoices registered by customer preceding the creation date.
23	**AVG_VAL**	Average value of invoices	Average value of the invoices by customer preceding the creation date.
24	**AVG_TERM**	Average payment term	Average number of days between when invoice is created and the day it is due, by customer preceding the creation date.
25	**NEW_CUST**	New customer flag (0,1)	Binary feature to indicate whether it is the customer's first invoice (1) or not (0)

Table 3.7 - Historical Features

Date features

To leverage the information given by the due date of an invoice, it was important to extract several date features. As seen in the exploration of the dataset, there are more invoices due some parts of the month than others, mainly in the first three days and the middle of the month. Additionally, it was noticeable that a higher percentage of early payments occurred in the last days of the month. There could also be some relation to the payment of invoices according to the day of the week they are due. For instance, invoices due the weekend could be missed since the offices are generally closed. In table 3.8, features 10 to 16 were created using dates.

Payment Terms

In this dataset the payment term of the invoices is provided in the 'ZTERM' categorical variable. This variable consists of a code that refers to a payment term agreement where the number of days is usually in the string. For instance: 0030 means 30 days of buffer to pay the invoice. However, there are also special contracts made between the company and the customer which cannot be inferred from the code, for instance the value "ZBD1TT". These terms are do not have the buffer number of days in the string as it can vary depending on the type of contract.

This time buffer defines the number of days between the creation of the invoice and the due date of the invoice. These dates are available in the dataset so we can always calculate the payment term using them:

$$Due\ Date\ (DUE\_DATE) = \ CreationDate\ (BUDAT) + Payment\ Term\ in\ days\ (ZTERM\_DAYS)$$

$$ZTERM\_DAYS = DUE\_DATE - BUDAT$$

Equation 3.1 - Calculation of payment terms in days

In table 3.8, features 17 and 18 were created considering the importance of the payment terms in this problem.

Ratings

The rating could have a significant say on the customer's capacity to meet obligations when he contracted an invoice. This rating is also looked at by the cash collectors and influences the contracted terms: if a customer has a low rating, the contract will have tighter terms to prevent default. In the dataset, there are 2 rating variables: the current rating at the time of the invoice and the historical rating, which is the rating class of the previously contracted invoice. To transform the ratings into digestible features first, they are converted into numerical features where 1 is the best rating (R0) and 1000 the worst rating (R10). Using this simple encoding of a categorical feature in this case is not problematic because the rating is a numerical categorical feature, and each level is orderable against the other (Zheng & Casari, 2018). Two features that will inform the model whether there was a change between the current to the previous rating of the invoice were computed: RATING_CHANGE_QNT and RATING_NEGATIVE_CHANGE. The rationale of a positive or negative change in the rating can also mean a change in the behavior of the customer. In table 3.8, features 19 and 22 were created considering the importance of the ratings in this problem.

Country

With the country feature (KNA1_LAND1), the invoices were classified into the seven most important country profiles identified by the cash collectors: Canada, Germany, Mexico, South America, USA, Rest of Europe and Others. To make this a more digestible feature for the model, it was used One-Hot Encoding technique was: each country profile is transformed into a binary feature where 1 represents that an invoice belongs to the group and 0 that it does not. Then, to avoid multicollinearity, one column was deleted. In table 3.8, features 4 to 9 were created to add knowledge about the customer country

3.3.5.2. Numerical Transformations

Used log transformation for the invoice amount (VALUE_LC), the customer's credit limit (KLIMK), the amount eligible for cash discount (SKFBT) and the recommended credit limit (DBEKR). The log transformation was used to transform these 4 numerical features because their distribution was heavy-tailed and so, not so convenient to provide information to most models (Zheng & Casari, 2018). The transformation Log(X+1) was used to cover cases where the value of the feature was zero. In table 3.8, features 1, 23-25 were transformed using the log transformation.

#	FEATURE NAME	FEATURE	DESCRIPTION
1	**LOG_VALUE_LC**	Amount of the invoice	Log Transformation
2	**INDUSTRY**	Industry Number	
3	**INDUSTRY_GROUP**	First 2 digits of Industry number which is the industry group	
4	**COUNTRY_PROFILE_CANADA**	One hot encoding based on the most important countries or regions according to cash collectors. Feature based on customer country (KNA1_LAND1)	1 = Invoice's customer is located in Canada 0 =Invoice's customer is not located in Canada
5	**COUNTRY_PROFILE_GERMANY**		1 = Invoice's customer is located in Germany 0 =Invoice's customer is not located in Germany
6	**COUNTRY_PROFILE_MEXICO**		1 = Invoice's customer is located in Mexico 0 =Invoice's customer is not located in Mexico
7	**COUNTRY_PROFILE_SOUTH_AMERICA**		1 = Invoice's customer is located in South America 0 =Invoice's customer is not located in South America
8	**COUNTRY_PROFILE_USA**		1 = Invoice's customer is located in USA 0 =Invoice's customer is not located in USA
9	**COUNTRY_PROFILE_EUROPE_WO_GERMANY**		1 = Invoice's customer is located in Europe (excl. Germany) 0 =Invoice's customer is not located in Europe (excl. Germany)
10	**DUE_DATE_WEEKEND**	Flag indicating whether the invoice due date falls on a weekend	1 = Invoice due date falls on a weekend 0 = Invoice due date falls on a weekday
11	**DUE_DATE_BEG_MONTH**	Flag indicating whether the invoice due date falls at the beginning of the month (in the first 3 days)	
12	**DUE_DATE_MID_MONTH**	Flag indicating whether the invoice due date falls at the middle of the month (3 days in the middle of the month)	
13	**DUE_DATE_END_MONTH**	Flag indicating whether the invoice due date falls at the end of the month (in the last 3 days)	
14	**DUE_DATE_MONTH**	Month of the invoice's due date	
15	**DUE_DATE_WEEKDAY**	Day of the week of the invoice's due date	
16	**RATIO_DUE_DAYS_UNTIL_MONTH_END**	Ratio of the number of days from the due date until the end of the month	Ratio of the number of days between the due date of the invoice until the end of the month over the number of days in the invoice's due date month DUE_DAYS_UNTIL_MONTH_END/DUE_DATE_DAYS_IN_MONTH

17	**ZTERM_DAYS**	Number of days between the creation of the invoice and its stipulated due date	Days between BUDAT and DUE_DATE. ZTERM converted into numeric
18	**SPECIAL_TERMS**	Flag indicating if invoice was contracted under a special term or not	1 = special payment terms, 0 = common payment term with X days of buffer
19	**RATING_1_1000**	Customer rating provided by Siemens Financial Services. Provides the opinion about the creditworthiness of the company and takes into consideration the country, industry, and financial risk.	Transform rating from 1 to 1000, being 1 the best capacity to meet financial obligations and 1000 the most vulnerable invoices
20	**RATING_CATEGORY**	Customer rating provided by Siemens Financial Services. Provides the opinion about the creditworthiness of the company and takes into consideration the country, industry, and financial risk.	1 = Green Class 2 = Yellow Class 3 = Orange Class 4 = Red Class
21	**RATING_CHANGE_QNT**	Quantifies how many levels the invoice customer increased or decreased its rating	Difference between the current rating and the historic rating. E.g. A change of 5 means it increased 5 levels, a change of -5 means it decreased 5 levels and 0 means it stayed the same
22	**RATING_NEGATIVE_CHANGE**	Flag indicating if the change in rating was negative or not	1 = negative change, 0 = no change or positive change
23	**LOG_SKFBT**	Amount eligible for cash discount in document in local currency	Log transformation
24	**LOG_DBEKR**	Recommended credit limit	Log transformation
25	**LOG_KLIMK**	Customer's credit limit	Log transformation

Table 3.8 - Invoice Level Features

3.3.5.3. Scaling

The final step in the feature engineering phase was scaling the created features. Scaling was important in this problem for numerous reasons. First, some algorithms that use weights or distance measures such as the logistic regression and KNN are adversely affected by differences in variable ranges and can be skewed towards features with greater ranges. Secondly, it was included in the list of features several counts, which can be problematic as they can increase without bound. In these cases, scaling the features can help (Brownlee, 2020; Zheng & Casari, 2018).

Taking the example of the invoice amount (VALUE_LC), its values have a very different scale from most of the other features and it has some purposely included outlier that could damage the performance of the algorithm. Knowing this, a scaler that is more robust to outlier was chosen to scale the features into more comparable ones. While a standard scaler removes the mean and divides by the variance, the RobustScaler used subtracts the median and scales the feature to Interquartile range and thus, is not influenced by very large values (outliers).

$$v' = \frac{v - mean}{variance}$$

Equation 3.2 - Scaling transformation performed by Sklearn's StandardScaler

$$v' = \frac{v - median}{IQR} = \frac{v - median}{75^{th}\ percentile - \ 25^{th}\ percentile}$$

Equation 3.3 - Scaling transformation performed by Sklearn's RobustScaler

3.4. MODELLING

3.4.1. Experiment Design

3.4.1.1. Train-Val-Test Split

Data leakage is a problem that often occurs in machine-learning. It consists of using data in the learning phase that represents what the model is trying to predict, leading to overly optimistic and unreal results. One way leakage could happen in this problem is to leak future data into the past. There was already some care taken when computing the aggregation of the historical features by considering the creation date (BUDAT) to compute features like the ratio of late invoices for a customer at a given time. Another technique used to avoid this problem was to make sure the split between train, validation and test was done with temporal consideration (Kaufman & Perlich, 2011) - instead of randomly splitting - the invoice creation date (BUDAT) was used to split the three, always maintaining a ratio close to 80:10:10. Since the window chosen to look back to compute the historical features was 6 months, the same time was used as a temporal cut-off to keep those older invoices away from the training set that was not possible to have true historical features computed.

The final dataset split consisted of around 3 years of data, with a training set from 2017-04-01 to 2019-11-10, the validation set, from 2019-11-11 to 2020-03-04 and the test set, from 2020-03-05 to 2020-08-31. On the one hand, the training set are examples of past invoices for the model to learn from. The validation set, on the other hand, is used to make several tests and fine-tuning the model. Finally, the test set, is the held-back used only to assess the performance of the trained model.

Dataset	*Scope*	*# Invoices*	*%*	*Baseline (2C)*	*Baseline (3C and 5C)*
Train	2017-04-01 to 2019-11-10	372252	79.87 %	64.42 %	44.95 %
Validation	2019-11-11 to 2020-03-04	46280	9.93 %	71.24 %	54.52 %
Test	2020-03-05 to 2020-08-31	47560	10.2 %	76.03 %	51.49 %

Table 3.9 - Summary of Train-Validation-Test Split

3.4.1.2. Imbalanced dataset – Resampling Techniques

For imbalance datasets where there are considerably fewer examples of one class than other classes, it is very hard for the model to learn the characteristics of the smaller class, because the model learns with fewer examples of it. Therefore, sampling techniques are useful to balance the targets and improve the model, as they provide solutions to re-balancing the number of instances. In this case, just predicting the all the payments will be on-time in the 2-class problem for instance, means that we can guess accurately 76% of them, reaching already a high value. Resampling techniques can avoid situations where the model naively predicts all the payments as on-time disregarding other classes which can be more important as in the example of the 2-class problem where it is more important to correctly predict the late payments as these are more problematic for cash collectors. Resampling techniques are widely used in typical imbalanced problems such as fraud detection, where fraud is very rare, but it is the most important class to predict (C. Chen, Liaw, & Breiman, 2004)

There are two traditional ways of re-balancing a dataset: reducing the majority class by randomly deleting examples – Random Under-sampling; and duplicating the minority class examples - Over-sampling. These traditional methods can bring some disadvantages such as potential loss of important information when using under-sampling and overfitting when using over sampling.

More advanced methods include SMOTE (Synthetic Minority Over-Sampling Technique), an over-sampling method of creating synthetic instances of the minority classes instead of duplicating them. This technique uses an algorithm that finds examples in the feature space of the minority class that are similar (or neighbors) and then creates new points along a line between the two neighbors. First, a random example is selected, then k neighbors are identified, and one is randomly selected to compute the line between the two. The original study suggests that using SMOTE performs better than plain over-sampling because the new synthetic instances are plausible as they use instances that are close to other in the minority class - this attenuates the possibility of overfitting compared to plain over-sampling. After all, these synthetic instances are new data instead of duplicates. In the same study, it is proved that using SMOTE to oversample the minority classes in combination of random under-sampling to trim down the majority class provides the best results (Chawla, Bowyer, Hall, & Kegelmeyer, 2002).

Nonetheless, these techniques come with some disadvantages: oversampling means that new additional learning samples are created which is problematic, especially in large datasets, as it consequently increases the learning time. Under-sampling can in its turn potentially discard useful training instances (Buda, Maki, & Mazurowski, 2017) . Another disadvantage of sampling techniques, in general, is that they alter the distribution of the data, and it can bias the output of the model. In particular, the probabilities generated by the estimators will reflect the distribution in which they were trained, so if the distribution is changed by sampling techniques, the probabilities will become less accurate.

To not increase the training time significantly, it was decided to keep the same total number of training instances in the resampled dataset by oversampling with SMOTE and using random under-sampling on the majority classes as suggested by Chawla et- al. (2002) until all the training levels of each target were the same size. It was also necessary to provide different resampled datasets for each different target as explained in table 3.9.

TARGET	*LEVELS*	*WITHOUT SMOTE*	*WITH SMOTE + UNDERSAMPLING*	*TECHNIQUE*
BINARY	*On-time*	239822 (64%)	186126 (50%)	Random Under sample
	Late	132430 (36%)	186126 (50%)	SMOTE Oversample with k=5
3 -CLASS	*Early*	72495 (19%)	124084 (33%)	SMOTE Oversample with k=5
	On-time	167327 (45%)	124084 (33%)	Random under sample
	Late	132430 (44%)	124084 (33%)	Random under sample
	Early	72495 (20%)	74450 (20%)	SMOTE Oversample with k=5
5-CLASS	*On-time*	167327 (45%)	74450 (20%)	Random under sample
	Late	98503 (27%)	74450 (20%)	SMOTE Oversample with k=5
	Very Late	27226 (7%)	74450 (20%)	Random under sample
	Critically Late	6701 (2%)	74450 (20%)	SMOTE Oversample with k=5

Table 3.10 - Summary of the application of the resampling technique in the train set

3.4.1.3. Evaluation metrics

Before introducing the algorithms chosen to tackle this classification problem, one should clearly define the metrics used for evaluating them. In this problem, the issue is tackled in different perspectives: A binary classification - classifying on-time or late payments - and a multi-class classification – going deeper into on-time payments by distinguishing early payments, and on the late payments by distinguishing the late payments into several levels of lateness. Thus, when choosing evaluating metrics, there should be a consideration for the different nature of the problems.

A confusion matrix was used in this project to compare the number of predicted instances in each class and their correct classification. It is a straightforward representation of the model results, and a starting point to understand other metrics. The confusion matrix for the binary problem can be described as below:

		Predicted Classes	
		On-time Payment (Positive Class)	Late Payment (Negative Class)
Actual Classes	On-time Payment (Positive class)	True Positive (TP)	False Negative (FN)
	Late Payment (Negative Class)	False Positive (FP)	True Negative (TN)

Table 3.11 - Confusion Matrix metrics for a 2-class problem

Likewise, the multiclass problem will also be evaluated with the help of a confusion matrix that will show a cross-table between the 3 and 5 possible classes of the 2 multi-call problems. Here, the diagonal will represent the correctly predict classes and the other cells the misclassification errors. Even though it is more difficult to find positive and negative classes in the multiclass problem, we can do it if we take each class a reference. Taking the on-time payment class we have the following confusion matrices:

		Predicted Classes		
		Early Payment	On-time Payment	Late Payment
Actual Classes	Early Payment	TN	FP	TN
	On-time Payment	FN	TP	FN
	Late Payment	TN	FP	TN

Table 3.12 - Confusion Matrix metrics for a 3-class problem

<table>
<tr><td colspan="2" rowspan="2"></td><td colspan="5">Predicted Classes</td></tr>
<tr><td>Early Payment</td><td>On-time Payment</td><td>Late Payment (less 1 month after due)</td><td>Very Late Payment (1-3 months after due)</td><td>Critically Late (more than 3 months after due)</td></tr>
<tr><td rowspan="6">Actual Classes</td><td>Early Payment</td><td>TN</td><td>FP</td><td colspan="3">TN</td></tr>
<tr><td>On-time Payment</td><td>FN</td><td>TP</td><td colspan="3">FN</td></tr>
<tr><td>Late Payment (less 1 month after due)</td><td rowspan="3">TN</td><td rowspan="3">FP</td><td colspan="3" rowspan="3">TN</td></tr>
<tr><td>Very Late Payment (1-3 months after due)</td></tr>
<tr><td>Critically Late (more than 3 months after due)</td></tr>
</table>

Table 3.13 - Confusion Matrix metrics for a 5-class problem

On the one hand, false positives, or any multiclass error falling below the matric diagonal are costly because the company will expect the customer to pay earlier than he will and not take any action to prevent the bad behavior. If the company receives the money from that account later than expected, it can lose the benefits of putting the money to yield if it had been put in the bank sooner rather than later. On the other hand, the company will try to target customers with bad behavior with more calls to action and stricter contract terms. This way the wrong (well-behaved) customers are targeted as such, and these unnecessary actions will result in the deterioration of the relationship and the decrease of customer satisfaction. In this way, errors below and above the correct diagonal of the matrices are costly, and the true objective to measure the algorithm performance is: how well can the model accurately predict an element of each class.

The standard metric used in most similar studies is the accuracy score which is simply the percentage of correctly classified instances in all instances in the test set. It can be interpreted in this problem as the percentage of correctly classified invoices as on-time or late. The accuracy is a useful metric for this problem because it can be easily comprehended by practitioners and cash collectors alike. The accuracy can be simply calculated as:

$$Accuracy = \frac{TP - TN}{TP + TN + FP + FN}$$

Equation 3.4 - Accuracy

Recall (also known as true positive rate) is the proportion of true positives divided by all the actual positive instances having they been correctly predicted as positive or not (Grandini, Bagli, & Visani, 2020). In this case, it is the proportion of correctly predicted on-time payments, over the sum of all the actual on-time payments either correctly predicted or not. This metric is used when it is important to focus on the capability of the model to find the positive instances accurately. The formula for recall is presented below:

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

Equation 3.5 - Recall

The accuracy metric can be insufficient in some cases. In the problem with five classes, there is a problem of an imbalanced dataset, where not all classes are equally represented – very late (7%) and critically late (2%) invoices are underrepresented. In a problem with this kind of imbalance, it is expected that the algorithm will predict heavily on the majority classes instead of the smaller ones because there are simply fewer examples in the training set. Many of the algorithms present in this study are sensitive to class imbalance namely those related to decision trees (Japkowicz & Stephen, 2002). The standard accuracy formula will not capture. It can mask a good performance with no smaller classes predicted because it counts each instance individually and does not consider the distribution of the classes having that larger classes contributing more heavily to its value. Balanced accuracy is another metric that was used in this study for support. It provides an average of "how likely will an individual of [each] class be classified correctly" (Brodersen, Ong, Stephan, & Buhmann, 2010; Grandini et al., 2020). For both binary and multi-class problems, the balanced accuracy is an average of the recalls of each class of the problem. Contrary to standard accuracy, the balanced accuracy gives each class, whatever their size is, equal weight in the final calculation, making it robust to imbalanced datasets. In the multi-class problem with 5 classes, where class imbalance is significant, the balanced accuracy will be useful to ensure that we have a good prediction on the 'Critically Late' class for example, which is highly unrepresented (only 2% of invoices).

$$Balanced\ Accuracy = \frac{Recall_{class\ 1} + \cdots + Recall_{class\ n}}{n}$$

Equation 3.6 - Balanced Accuracy

Balanced accuracy will be equal to the standard accuracy when the dataset classes are balanced, the algorithm's performance is good and results in equal values for the classes (Fernández, García, Galar, & Prati, 2018). If the standard accuracy is high but balanced accuracy is low, this means the algorithm is taking advantage of the majority classes in the imbalanced dataset.

In binary problems, it is also common to use Receiving Operating Characteristics Curve (ROC) and the area under the curve (AUC) to evaluate the predicting capability of the model. The ROC curve is a graph that visualizes the trade-off between the recall (or true positive rate) and the false positive rate (calculated as FP/(FP+TN)), showing the cumulative values in each axis. The area under the curve (AUC), is a metric that summarizes the ROC curve plot and indicates the skill of the model when discriminating classes. A model that cannot discriminate any class (such as the baseline used which predicts all the instances as on-time payments) will have an AUC of 50%, so a good model should be close to 100%, which is the perfect case, where the model classifies all the classes correctly.

3.4.2. Machine-learning Algorithms for Classification

As discussed, in the present dataset, we have 3 classification problems, one binary, and the other multiclass. There are 500000 observations or invoices and 49 features. The goal is to use a training set consisting of examples and their outcome and use a predictive model capable of predicting new data.

In supervised learning, there are plenty of algorithms available to use for classification. Each algorithm performs the classification in different forms, so it is usually a best practice, in projects as such, to try several different algorithms and chose the one with the best performance. The choice of algorithms

can be based on the amount of data to classify, the type of task, and the number and type of features (Abbott, 2014).

The supervised learning algorithms can also be divided into families so it will be interesting to try algorithms from different families to understand how the results varied. In the following section, the algorithms used will be discussed. For each, there was a brief explanation of how they work for binary and multi-class classification, a brief overview of the parameters and also pros and cons of using each algorithm.

Family	*Algorithms tested:*
Regression	Logistic Regression
Instance-based	K-Nearest Neighbors
	Support Vector Machines
Ensembles	Random Forests
	Gradient Boosting Machines
	LightGBM

Table 3.14 - List of algorithms used and their families

Most of these algorithms were used by other authors for the same type of problem.

Before moving on, and because we have two multi-class problems at hand, it is important to define two essential concepts in machine-learning that transforming models that are usually built for binary classification into multinominal. The logic behind the One-vs-All (OvA) and One-vs-One (OvO) heuristics is to split the multiclassification into several binary problems that the models can easily train (Brownlee, 2020).

OvA (One-vs-All) splits the multi-class problem by highlighting one class and comparing it to the rest. For example, in a problem with five classes (early, on-time, late, very late and critically late), OvA will split the problem into 5 binary problems: 1) early vs (on-time, late, very late and critically late), 2) on time vs (early, late, very late and critically late), 3) late vs (early, on time, very late and critically late), 4) very late vs (early, on time, late, and critically late) and 5) critically late vs (early, on time, late and very late). OvO (One v. one): splits the multi-class problem to have each class compared to each. In the five class examples (early, late and on-time), Ov0 will split the problem into 10 binary problems: early vs on-time, early vs late, on-time vs late: 1) early vs on-time, 2) early vs late, 3) early vs very late, 4) early vs critically late, 5) on-time vs late, 6) on-time vs very late, 7) on-time vs critically late, 8) late vs very late, 9) late vs critically late, 10)very late vs critically late.

OvA is by far the most popular approach, since it creates fewer binary cases for problems with more classes confers the algorithm more efficienthly when learning. However, because it aggregates the 'rest' classes into one, it can create highly skewed class distribution in some of the binary cases. OvO, on the other hand, even though it is less efficient, it is more robust and recommended for a problem with high class imbalance. This approach simply ignores the rest of the classes when considering only two, resulting in less imbalanced binary problems. (Fernández et al., 2018; Galar, Fernández, Barrenechea, Bustince, & Herrera, 2011)

3.4.2.1. Logistic Regression

Regression is a classic method in machine-learning that tries to approximate a relationship between the variables of the problem by iteratively optimizing a measure of error. In regression terms the target

or variable to be predicted is called the dependent variable and the features or predictors are independent variables.

In the regression family, the algorithm can be either linear or logistic. In linear regression, the output the model tries to predict is continuous (e.g., predicting house prices, weight or revenue). The aim is to estimate coefficients that multiply each feature and determines their linear contribution to the final prediction. To predict the value for an instance, all the coefficients multiplied by the features (or predictors) and a constant called a bias are summed up. The optimization method reduces the error measured by the ordinary least squares (OLS) (Burkov, 2019; Larose & Larose, 2015). Take the example of predicting the price of a house: we can take several features like the square foot, the location and the year of construction. When fitting a linear regression model, it will train with examples and conclude the coefficients for each feature plus a bias. Given a new instance, we calculate the value of the house using the linear function provided by the algorithm.

Logistic regression is similar to linear regression, but it is a classification algorithm because its outcome or dependent variable is categorical classes (e.g., email is spam or not or in this case, the payment will be early, late, or on-time). This model is built to, instead of giving a linear estimate of the instance value, provides the probability of an instance belonging to one of the target classes - this probability is usually called the log-odds ratio and ranges from 0 to 1 (Abbott, 2014). The log-odds ratio is defined in equation 3.6, being P_i the probability of the event i, f_1 to f_n the feature values, c_1 to c_n the estimated coefficients for the features and b the constant or bias. In this type of regression, there will also be a coefficient for each feature, but they are not interpretable as in the case of linear regression. Instead of reducing the OLS, the goal of the logistic model is to maximize the likelihood (or log-likelihood) using an iterative algorithm. Using the binary example of this project, we calculated the coefficients iteratively for each feature, fed these parameters to the logistic function and extract a probability. If the probability extracted related to an invoice is higher than 0.5, then that invoice was assigned to class 1 (Late payment); otherwise, the invoice is predicted as one time (class 0) (David W Hosmer & Lemeshow, 2000).

$$Log - odds\ Ratio = \log\left(\frac{P_i}{1-P_i}\right) = f_1 * c_1 + f_2 * c_2 + \cdots + f_n * c_n + b$$

Equation 3.7 - Log-odds Ratio

$$P_i = \frac{1}{1 + e^{-(f_1 * c_1 + f_2 * c_2 + \cdots + f_n * c_n + b)}}$$

Equation 3.8 - Calculation of the probability using LR

Logistic regression can also be applied to multi-class problems, it is called multinominal logistic regression (John D. Kelleher, Namee, & D'Arcy, 2015) but its implementation is slightly different from what was described. In machine-learning, one method is to use the One-vs-All or One-vs-One heuristics to split the multi-class problem, explained in the beginning of this section. Another approach is to change the model to predict the probability of a record belonging to each class directly – instead of using the log-odds as a loss function, the model uses cross-entropy loss function (John D. Kelleher et al., 2015).

The logistic regression is an important algorithm and usually the starting point in any machine-learning problem. Its decision boundary is linear which makes it not very applicable to many real-life cases.

Despite this, it is a fast and simple algorithm with a good reputation amongst practitioners due to being easily interpreted, since it provides some reasoning as to why certain outcomes were achieved. Public health is one of the areas where logistic regression is applied for this same reason (D. W. Hosmer, Taber, & Lemeshow, 1991).

As with most linear models, using the logistic regression requires careful data preparation. It requires numerical features (categorical features need to be encoded, e.g. One-hot Encoding) and does not accept missing values. Additionally, to achieve good performances, interaction variables are necessary for accurate predictions and it is recommended that inputs should be linearly scaled (Abbott, 2014). One important assumption is that there should not be any multicollinearity present in the independent variables (or highly correlated features), otherwise there is a real chance of overfitting (Brownlee, 2016).

3.4.2.2. Support Vector Machines

Support Vector Machines is a popular algorithm for classification, regression and outlier detection and it has real-life applications in various fields such as image classification (Dibike, Velickov, & Solomatine, 2000; Mountrakis, Im, & Ogole, 2011) and automatic text classification (K. Dalal & A. Zaveri, 2011). It is inherently a classification algorithm for binary problems but can be adapted to multi-class using the OvA approach used by one of its creators (Vapnik, 1998, 2000).

The SVM algorithm consists of learning an optimal n-1 dimensional hyperplane in an n-dimensional space that separates the data points into the two target classes. This means that in a two-dimensional space (problem with two features only), the hyperplane is a line. In contrast, in a problem with three features, for example, the hyperplane is a 2-dimensional surface (Joshi, 2016). This hyperplane is the optimal separation between the two target classes and the decision boundary that maximizes the distance between the two closest points of each class. These closest data points are called the support vectors, while the distance between them and the hyperplane is called the margin. Typically, the hyperplane is linear, but as discussed, this type of decision is not so common in real-life applications. The hyperplanes can have different shapes by using different kernel functions such as polynomial or sigmoid input transformation. These kernels have the ability to transform the SVM into a non-linear classifier when faced with data that is not linearly separable (Cortes & Vapnik, 1995). For better results, a parameter C is introduced to relax the margins, so that more noisy data does not affect the hyperplane definition, and the margins become “soft margins” with the introduction of this parameter and the data points on the wrong side of the plane are weighted down.

Flexibility in the type of problems it can solve, an easy to follow geometric intuition and proved theoretical background are the main advantages of SVMs and the reason behind their popularity among machine-learning practitioners (K. P. Bennett & Campbell, 2000). Specific advantages SVMs could bring into this project is that they are popular among problems with a lot of features and have been reported as being robust to class imbalance (Japkowicz & Stephen, 2002).

SVMs are memory intensive and have long training times specially with big datasets. In comparison to LR, they are also not optimized for noisy data and require special data preparation like scaling and missing values treatment (K. P. Bennett & Campbell, 2000). They are decision machines only (Bishop, 2006), so they do not estimate the probability of belonging to each class contrary to LR.

3.4.2.3. K-Nearest Neighbors

K-Nearest Neighbors (KNN) is an instance-based algorithm; it stores the training examples and performs classification of new data points based on searching the training data points for the closest representation. It is a "lazy" algorithm because it delays the searching until a new data point has to be classified (Kuhn & Johnson, 2013), and it simply does not learn anything rather, it performs "look-up" on the training data. This type of learning is very different from the previously discussed SVMs and LR that consist of learning a decision boundary that separates data points and generalizes to new instances.

Conceptually, once a new instance must be classified, the KNN algorithm looks at that instance's k closest neighbors that are determined by a similarity metric. The algorithm then uses the majority vote of the k neighbors to classify the new instance. In simple terms, take a new invoice (a) to classify: the algorithm with k=5 computes the distances and finds 5 closest points. From these 5 neighbors, 4 of them are On-Time Payments, while the remaining one is a Late Payment. The classification of the new invoice (a) will be, by majority voting, an On-Time Payment. KNNs can be easily in the same way for multi-class problem – although care should be taken when choosing k – it should be an odd number if the number of classes is even to cover for ties. It is also possible to get the probability of each class by normalizing the frequency of the neighbors as such:

$$P_{class=0} = \frac{count(class = 0)}{count(class = 0) + \ count(class = 1) + \cdots + \ count(class = n)}$$

Equation 3.9 - Calculate the probability of each class using KNN (Brownlee, 2016)

The main parameters to consider and tune are the number of k neighbors, which, if too small, may become very susceptible to outliers and over-fitting, but if too big, can also miss some important variations. Among the similarity metrics, the Euclidean Distance is the most commonly used. However, there are several other options such as Manhattan distance -which is better for problems with features of different types - and Minkowski distance.

KNNs are computationally very expensive as there is the need to calculate the distances between all test data points and all training points. They also require the features to be scaled and treated because they use distances between points, and features with larger scales will introduce bias (Kuhn & Johnson, 2013). For the same reason, they are also very sensitive to outliers.

Even though KNNs come with a few disadvantages, it is an incredibly simple algorithm that classifies differently by taking one new instance at a time instead of estimating the overall target function such as LR and SVMs (Mitchell, 1997). This is important for problems where the target function turns out to be very complex.

3.4.2.4. Decision Trees

A Decision Tree in classification, is a set of rules that recursively partitions the data until it reaches a homogeneous group. It is one of the simplest algorithms and can also be used for regression problems. There are several implementations of decision trees in machine-learning, but the most important and the focus of this section is the Classification and Regression Trees (CART) (B. Friedman & Stone, 1984).

The process of building a classification tree is the following: First, from all the features in the dataset, one is selected considering its ability to separate the data into the target categories. The selected feature is the one that provides more information gain. This information gain can be measured by the Entropy or Gini, which measures the purity of the nodes. The calculation of Gini is depicted in equation 3.9 being P_i the probability of the class i. The same process is then repeated for the selected partition until a stopping criterion is met. The objective is to build a tree node after node that becomes the tree that best separates the data into the classes. This means, in the end, the decision tree provides leaves that are as pure as possible and separate the different categories the best (Kuhn & Johnson, 2013).

$$\text{Gini (n-class problem)} = P_1(1 - P_1) + P_2(1 - P_2) + \cdots + P_n(1 - P_n)$$

Equation 3.10 - Gini Calculation

Below is an example of a decision tree built for our 2-class problem. This tree was calculated until the stopping criterion of having a maximum depth of 3 levels of depth was met. With a simple and fast configuration, the accuracy achieved in this model was 73% and the balanced accuracy 67%. It is possible to see that the algorithm selected RATIO_LATE and NR_PAID as two of the most important features. None of the leaves in entirely pure because we used a stopping criterion. If we had not, the tree would keep growing until all leaves became pure, but it would also overfit.

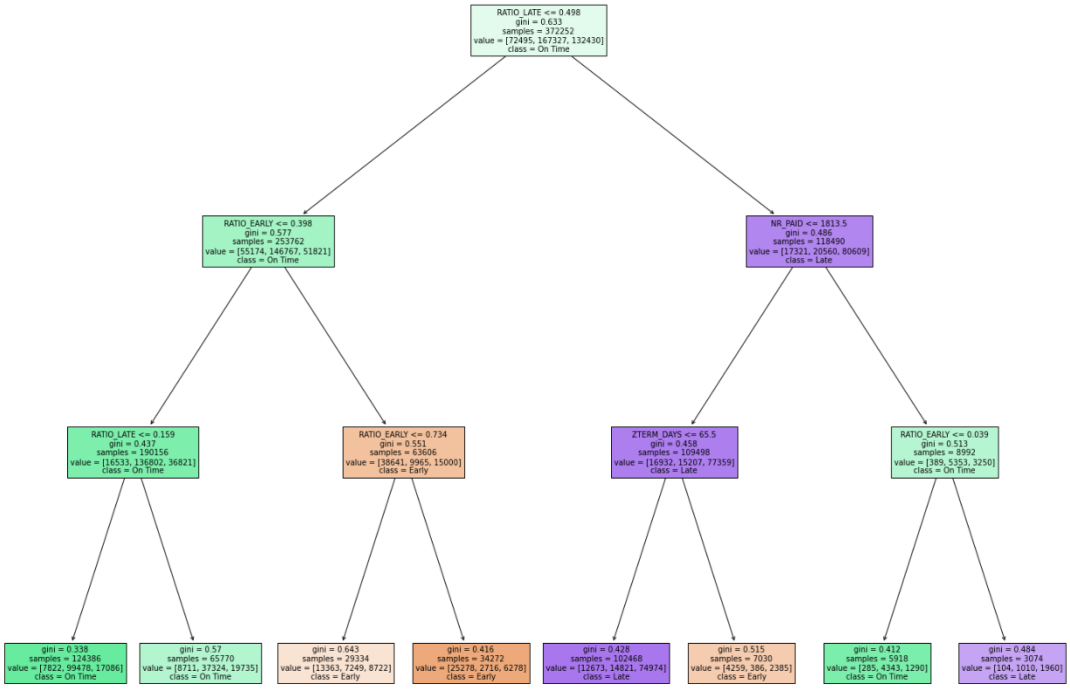


Figure 3.11 - Example of decision tree for a 2-class problem with max_depth = 3

Late invoices, for example, are characterized by having customers with a ratio of late invoices in the last 6 months larger than 49.8%. The invoice will also be classified as late if the number of paid invoices by the customer (in the last 6 months) is less or equal to 1814 invoices and if the payment term of the invoice is less or equal to 66 days. On the other hand, the invoice can also be classified late if the

number of paid invoices by the customer (in the last 6 months) is larger than 1814 invoices and the ratio of early invoices is larger than 4%.

In this way, the decisions of the algorithm can be plotted in a tree and the fact that the algorithm is easily explainable is one of its most important characteristics. The most important features, for instance, are present on the top of the tree. This is also an important algorithm that takes several types of features without the need for scaling.

However, decision trees struggle to be applicable in real-life examples, as the decisions are very dependent of the training data and thus do not generalize well. Also, real life problems consist of complex decision boundaries which decision trees, due to their simplicity, cannot cover. They are a great way to introduce Random Forests™ which add the flexibility that Decision Trees lack while also using them in the process.

3.4.2.5. Random Forest™

Random Forests™ by Leo Breiman and Adele Cutler is an ensemble method built out of Decision Trees. Ensembles are accurate techniques in machine-learning that overcome several problems existing in simpler algorithms, such as flexibility and accuracy. Ensembles consists of combining more than one classifier to improve predictive performance, in the case of Random Forests – the classifier used is decision tree.

To build an ensemble of decision trees, the decision trees need to be somehow different and uncorrelated. Otherwise we get the same results. To achieve this, the first step in Random Forests™ is to randomly select samples of data from the training set to construct a **bootstrap** dataset with the same size as original (with repeated samples). Then it creates a decision tree from the bootstrap dataset with a limited number of features selected randomly at each split point of the decision tree. It repeats this several times until it gets a varied set of decision trees. From this set, it takes the majority vote of the trees to get the classification result – a process called **bagging** (Breiman, 1996). The samples that did not make it into the bootstrap dataset (also called out-of-bag data) are used to validate the results and calculate the proportion of incorrect instances (out-of-bag error).

Each tree is grown to its full extent without pruning or any other stopping criteria. This is a necessary trait to have very different trees with a high variance but low correlation between the predictions and prediction error. Randomizing the selected set of features also enables to build different trees, thus achieving good generalizable results when the bagging is performed and better results than an individual tree classifier (Breiman, 2001; Kuhn & Johnson, 2013; Leo Breiman, 1996).

The common heuristic to select the number of features to select in the split randomly is the square-root of the total number of features.

Random Forests™ collect some of the benefits from decision trees like the minimal data preparation necessary such as scaling, normalization and missing value treatment (Kuhn & Johnson, 2013). They are also capable of predicting accurately without overfitting due to their random components and the Law of Large Numbers, respectively, especially in classification tasks (Breiman, 2001). They are efficient in large datasets and handle well large numbers of features. Contrarily to Decision Trees, because they are based on the Law of Large Number, Random Forests™ do not overfit – as Breiman demonstrated when bagging is applied, the variance is reduced. Random Forests™ provide several optimization

options in the forms of parameters and hyperparameters. It is possible to extract the class probabilities by the proportion of votes in each tree, which is then averaged for the forest prediction (Malley, Kruppa, Dasgupta, Malley, & Ziegler, 2012).

The major disadvantage of Random Forests™ is the lack of explainable results. Even though we can extract the impact of each feature used, with ensembles in general the underlying relationship between the features and the results is hard to explain (Kuhn & Johnson, 2013). For this reason, it is considered merely a predictive tool and not a descriptive tool like LR and DT can be.

3.4.2.6. Gradient Boosting Machines

While bagging consists of different individual models learning in parallel, boosting, another technique, involves learning models in a sequence, i.e., learning one model first and then learning a new one that tries to improve the previous model's error. The boosting technique was first proposed for classification problems and consisted of learning weak classifiers and then 'boost' them to create a strong algorithm over iterative training (Kearns, 1988; Schapire, 1990). In Boosting, first, a sequence of classifiers is created. These are simple classifiers that perform only slightly better than random guess. A weight is given to each classifier according to their archived accuracy. At each iteration, the weights are updated, and the misclassified instances are given larger weights, then new classifiers are built on top of the older ones to classify the remaining errors. This way, the model is encouraged to correct the misclassified classes and improve. In the end, the results of the classifier are aggregated by weighted vote.

There are many different types of Boosting techniques, and they vary on the way each assigns weights and the type of small classifiers. Adaptive Boosting (AdaBoost) works as described above and uses small decision trees with one split only (or "stumps") as its weak learners. AdaBoost was the first boosting algorithm and it uses an exponential loss function (Freund, Schapire, & Hill, 1996; Kuhn & Johnson, 2013).

In gradient descent, the objective is to minimize a loss function (there are various to choose from) in a gradient-descent manner and it uses trees as the boosted classifiers. Gradient Boosting Machines also use small trees as classifiers and are stagewise additive models (J. H. Friedman, 2001). Stagewise because they fit a new tree at every stage that tries to improve the residuals (observed-predicted) of its preceding tree, and they don't go back to correct past fitting – which is a way of these models to avoid overfitting. Furthermore, additive because the trees are summed up and weighted by a coefficient (learning rate) to reach the final result for the observation. (J. H. Friedman, 2002). At each stage, the goal is to find what is the best slight improvement to make the model better. When this improvement is reached, the function is updated.

In GBM, the objective is to minimize a differentiable loss function (there are various to choose from) using a gradient descend, usually the deviance function (also known as logistic regression) is used for the binary and multinominal problems. The gradient descend modifies the parameters of the next tree that minimize the loss better.

The most significant benefit of using Gradient Boosting is that, due to its non-parametric, greedy nature and ability to reduce bias and variance from simple decision trees, the algorithm can solve very complex problems where the decision boundary is hard to map. It is also a highly customizable

algorithm (Natekin & Knoll, 2013) and has some very important parameters that can be changed in each different situation:

- Learning rate: controls the complexity of the model and its value affect the learning time;
- Tree size: or number of splits (how deep a tree can become). The deeper the tree, the more interactions the model can have. Correlation is problematic so using stumps may lead to less overfitting. However, it takes more time to reach an optimal result;
- Number of trees: if there are too many the model can overfit. Contrary to random forests as more trees you add you can never overfitting because the average;
- Lasso regularization: selects only a subset of all the small trees used in training (J. H. Friedman & Popescu, 2004);
- Stochastic Gradient Boosting: reduce the correlation between the trees. The subsampling sizes can be controlled;
- Early Stopping: a criterion can be set over which, if the model cannot minimize the loss function by a threshold value for several iterations, the optimization process will stop, thus preventing overfitting.

The biggest disadvantage of GBMs is the fact that the high capacity of models to identify complex relationships can lead to overfitting. However, as describes above, there are several ways to attenuate this disadvantage namely by setting some parameters for the same purpose (e.g., regularization and early stopping). It is also an algorithm that consumes much memory and can be at times slow to train with a lot of data (Natekin & Knoll, 2013). Like any other ensemble, its interpretability is harder.

Over the years, scholars and specialists have been improving the capabilities of GBMs to different implementations. XGBoost (T. Chen & Guestrin, 2016) was a success when it was popularized in 2016 as it improved the performance of simple GBMs using the histogram and pre-sort techniques. CatBoost (Dorogush, Ershov, & Gulin, 2018) was developed by the Russian company Yandex with the feature of treating categorical variables into numerical performing various highly efficient statistics. Moreover, LightGBM (Ke et al., 2017) is developed by Microsoft.

LightGBM:

LightGBM, similarly to CatBoost also has a feature for treating categorical variables like XGBoost as it uses histogram-based split. The histogram-based split consists of using binarization on continuous features and using the histogram to find the best split (Ke et al., 2017), thus avoiding sorting the features as implemented in the standard GBM implementation from sklearn. This makes the algorithm much more efficient. However, its main difference is the implementation of the innovative Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) techniques. The GOSS technique reduces the instances based on their gradient - instances with larger gradients are those that are undertrained, so by sampling those with smaller gradients the algorithm will focus on the harder problems. This allows the LighGBM algorithm to reach a higher accuracy with less data (Ke et al., 2017). EBS, on the other hand, works as a feature selecting technique by identifying and aggregating mutually exclusive features, thus also making the algorithm faster during training.

Most experiments in academia and competitions benefit a lot in terms of accuracy with the implementation of GBMs. There are several applications of these models, specifically the LightGBM. More recently, it was used to predict cryptocurrency price trends (Xiaolei, Mingxi, & Zeqian, 2020) and classifying breast cancer patients (Wang, 2017).

3.4.3. Model Tuning – Bayesian Optimization

It is possible to tailor the model to the dataset by optimizing the algorithm's hyperparameters during the exploration. The hyperparameters of the algorithm are those parameters set by the practitioner when setting up any algorithm and they influence the training model. Optimizing them is one important step in most machine-learning pipelines, as it can bring some increased performance. This is, however, a step that some scholars consider as a "black art", resorting to heuristics or rules of thumb that one can usually only acquire with some experience as a data scientist (Snoek, Larochelle, & Adams, 2012).

To avoid the steep learning curve that takes to know how to intuitively set hyperparameters, some techniques are used in machine-learning for searching for the optimal hyperparameters. Most used are grid-search – which searches for every single the combinations of hyperparameters between a certain set of values provided, and random search – which, in turn, randomly creates combinations of parameters between a bounded domain and returns the best after exploring n combinations. Both techniques have an important disadvantage: each combination of parameters is performed individually, and only, in the end, all experiments are compared, and the best is chosen. For this reason, we cannot use each experiment's information to improve the next combination and optimizing the process (Jordan, 2017).

For this reason, it is important to introduce a more efficient technique for searching the hyperparameter space: non-parametric Bayesian Optimization. Bayesian Optimization is, in simple terms, an algorithm for finding a set of optimal hyperparameters that optimize a function, which in our case is a predictor. The working principle behind Bayesian optimization is to define a Gaussian Process (GP), defined with a mean and a covariance function (or a composition of many) that we think is able to describe the hyperparameters and the predictor performance metric adequately. By fitting the GP, we can draw samples from the fitted GP and with these samples, generate a distribution from which we can estimate the uncertainty in the parameter space. Figure 3.12 illustrates the evolution of Bayesian Optimization using Gaussian processes, and the blue area shows the uncertainty that comes from the GP samples. Several heuristics can be used to explore this space - more exploitative or more explorative - and the algorithm will choose the best candidate to maximize the function. This is repeated as long as there is computational budget, or some threshold is reached. It belongs to a family of algorithms - sequential model-based optimization (SMBO) - that use the results of a previous experience in the next, to improve the sampling method (Jordan, 2017). To perform Bayesian Optimization, it is necessary to define the objective function that will take the hyperparameters to optimize an initial search space. Considering the Bayes Theorem, the goal of Bayesian Optimization is to try to find a posterior distribution using an initial probability or prior distribution of previously evaluated points. Given the training data D and the objective function f, the posterior function, represented by $P(f|D)$ can thus be defined as $P(f|D) = P(D|f) * P(f)$ (Brownlee, 2020). The posterior function is given the prior function and the likelihood, which represent the evidence and the probability of the evidence (Brochu, Cora, & Freitas, 2010). This posterior function will approximate the optimization function and thus is used to identify the best hyperparameter candidates for the next iteration, in other words, the combinations that yield bigger improvement. In the first iterations, the algorithms select a sample of n random initial points and evaluate them. In the next iteration, the algorithm uses these results to narrow down the search space "computing a posterior expectation of

the hyperparameter space" (Jordan, 2017), the process continues by selecting another sample of points that optimize using the posterior to select the following sample.

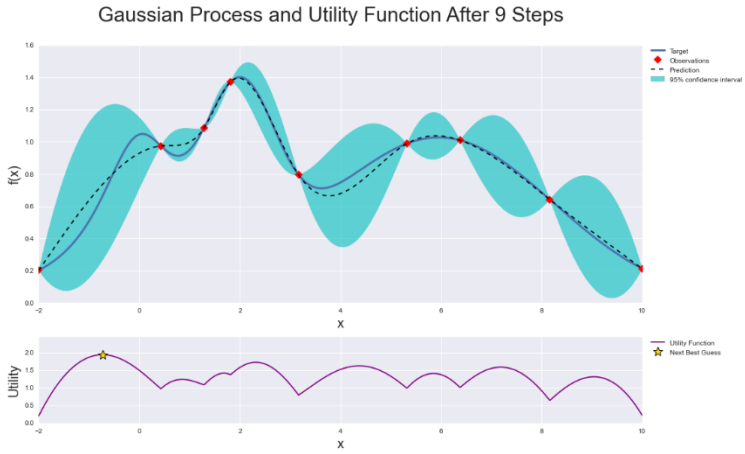


Figure 3.12 - Bayesian Optimization Using Gaussian Processes retrieved from GitHub (Nogueira, 2014)

Bayesian optimization, or variants, has proved to be faster than other techniques at converging to an optimized set of parameters, beating results from studies of various machine-learning areas (Snoek et al., 2012).

In this project, Bayesian optimization was used to fine-tune the LightGBM algorithm which offers more parameters than most models and usually requires some effort in setting the correct parameters to achieve good performance. Because there were a lot of training instances, it was necessary to choose an optimization algorithm that would minimize the steps and time required to find a combination of hyperparameters close to optimal. The model was defined using Balanced Accuracy taking a set of 8 hyperparameters (see table 3.15) as the objective function used to maximize. Ten random points were used to initially explore the search space and 50 steps were used to optimize the function.

HYPARPARAMETER	DESCRIPTION	SEARCH SPACE BOUNDS
NUM_LEAVES	Number of leaves of each tree estimator	[10, 80]
MAX_DEPTH	Maximum number of levels of each tree estimator	[3, 30]
FEATURE_FRACTION	Percentage of features selected to build the tree estimator at each iteration	[0.1, 0.9]
BAGGING_FRACTION	Percentage of data selected for bagging at each iteration	[0.8, 1]
MIN_SPLIT_GAIN	Minimum number of gains to split a tree estimator node	[0.001, 0.01]
MIN_CHILD_WEIGHT	Minimal sum of the hessian value in each leaf	[5, 50]
LAMBDA_L1	L1 Regularization	[0, 5]
LAMBDA_L2	L2 Regularization	[0, 3]

Table 3.15 - LightGBM Hyperparameters for Optimization using Bayesian Optimization

4. RESULTS AND DISCUSSION

Overall, all the classification models were able to surpass the baseline of predicting all the invoices as on-time, which is a good indication that all the work with pre-processing and feature extraction added some value to the problem. In this section, the classifiers are compared according to the metrics explained in section 3.4.1.3 Evaluation metrics, and for each classification problem: binary, 3-class and 5-class.

4.1. RESULTS FOR BINARY CLASSIFICATION PROBLEM:

Considering a binary target (on-time and delayed payments), using all the 49 features extracted from the dataset and a train-validation-test split of approximately 80:10:10, the following classification report was obtained:

		PRECISION	RECALL	F1-SCORE	ACCURACY	BALANCED ACCURACY	AUC
BASELINE	On-Time	0.760	1.000	0.864	0.760	0.500	0.500
	Late	0.000	0.000	0.000			
	Avg/Total	0.380	0.500	0.432			
LR	On-time	0.823	0.959	0.886	0.812	0.652	0.652
	Late	0.727	0.345	0.468			
	Avg/Total	0.775	0.652	0.677			
SVM	On-time	0.841	0.936	0.886	0.817	0.688	0.688
	Late	0.683	0.44	0.535			
	Avg/Total	0.762	0.688	0.711			
KNN	On-time	0.862	0.793	0.826	0.746	0.696	0.696
	Late	0.477	0.598	0.531			
	Avg/Total	0.670	0.696	0.679			
RF	On-time	0.900	0.91	0.905	0.855	0.795	0.795
	Late	0.704	0.681	0.692			
	Avg/Total	0.802	0.796	0.799			
GBM	On-time	0.902	0.907	0.904	0.854	0.797	0.797
	Late	0.699	0.686	0.693			
	Avg/Total	0.801	0.797	0.799			
LGBM with hyperparameter optimization	On-time	0.905	0.908	0.906	0.857	0.802	0.802
	Late	0.704	0.697	0.700			
	Avg/Total	0.805	0.803	0.803			

Table 4.1 - Classification report of the 2-class problem

In conformity with literature review it was found that tree-based ensembles are better at predicting this type of problem, which can be confirmed in most metrics. This is also true for the multiclass problems as will be further discussed. Random Forest (using 100 estimators) and GBM (Using 500 estimators) performed very closely together in terms of accuracy and balanced accuracy, being GBM the close winner of the two with balanced accuracy of 79,7% and 68,6% recall or true positive rate. However, when considering training time, GBM proved to be much less efficient than RF which leaves open to interpretation which of the two is ultimately better for using considering both factors. Ultimately, the best model analyzed was LGBM with parameter tuning - this model is faster to train, but it takes longer if one chooses to optimize the parameters. It seems that to tune the model parameters were a good option as it reached the highest balanced accuracy of 80,2%, and it was also capable of predicting late payments better than other models with 69,7% of recall.

KNN also demonstrated surprisingly good results for such a simple technique, achieving almost 60% of recall for late payments but falling short on reaching a precision-recall balance as good as the other models. As expected, LR and SVM performances were also very close as they are both linear models. They still could not reach the good performance levels as other more complex models did. LR was the best model at predicting more actual on-time payments with 95,9% of recall, but only achieved a recall score of 34,5% for late payments which clearly shows the effect of training with an imbalanced dataset.

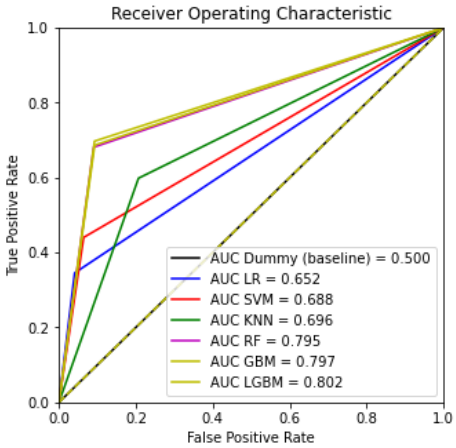


Figure 4.1 - ROC curves

Comparing the ROC curves for all the models above (Figure 4.1), we can detect a considerable similarity of results between RF and GBM, which reached the sum of highest AUC scores only being surpassed by LGBM. This means that they are the best at differentiating on-time from delayed payments. The AUCs of LR and SVM was not considered very good as they are only approximately 10% above the baseline.

4.1.1. Results with Resampling

Using the resampled dataset described in section 3.4.1.2 Imbalanced Dataset – Resampling Technique, the best models discussed above were tested. Below is the classification report:

		PRECISION	RECALL	F1-SCORE	ACCURACY	BALANCED ACCURACY	AUC
RF	On-time	0.920	0.865	0.892	0.840	0.813	0.813
	Late	0.640	0.760	0.695			
	Avg/Total	0.780	0.813	0.794			
GBM	On-time	0.929	0.852	0.889	0.838	0.822	0.822
	Late	0.629	0.792	0.701			
	Avg/Total	0.779	0.822	0.795			
LGBM	On-time	0.928	0.851	0.888	0.837	0.821	0.821
	Late	0.626	0.790	0.699			
	Avg/Total	0.777	0.821	0.794			

Table 4.2 - Results with resampling for the 2-class problem

There is a clear improvement in the results when testing the model with resampled data. Curiously, GBM performed better using the resampled dataset than RF and LGBM when comparing the balanced accuracy. GBM reached a balanced accuracy of 82,2% which is best value of all the models experimented. However, it did not reach the best standard accuracy with 83,8% comparing to RF's

84%. Still, GBM is the model that is able to distinguish the classes better (highest AUC of 82,2%) and that presents the highest recall for late payments (79,2%) amongst all the experimented models including those not trained with a resampled dataset.

The disadvantage of using model with a resampled dataset is that we are changing the training distributions, the estimation of the probabilities of each class are not so reliable and, unfortunately, this was an important output of the project for the business. Even so, the resampling results give us interesting insights.

4.2. Results for 3-Class Classification Problem:

Considering a 3-Class target (early, on-time and late payments), using all the 49 features extracted from the dataset and a train-validation-test split of approximately 80:10:10, the following classification report was obtained:

		PRECISION	RECALL	F1-SCORE	ACCURACY	BALANCED ACCURACY
BASELINE	Early	0.000	0.000	0.000	0.515	0.333
	On-time	0.515	1.000	0.680		
	Late	0.000	0.000	0.000		
	Avg/Total	0.172	0.333	0.227		
LR	Early	0.666	0.226	0.338	0.680	0.611
	On-time	0.751	0.870	0.806		
	Late	0.552	0.737	0.631		
	Avg/Total	0.656	0.611	0.592		
SVM	Early	0.771	0.003	0.006	0.639	0.550
	On-time	0.708	0.886	0.787		
	Late	0.514	0.760	0.613		
	Avg/Total	0.664	0.550	0.469		
KNN	Early	0.508	0.443	0.473	0.621	0.586
	On-time	0.760	0.716	0.737		
	Late	0.477	0.598	0.531		
	Avg/Total	0.582	0.586	0.580		
RF	Early	0.742	0.425	0.540	0.741	0.692
	On-time	0.800	0.875	0.836		
	Late	0.629	0.777	0.695		
	Avg/Total	0.724	0.692	0.690		
GBM	Early	0.722	0.433	0.542	0.742	0.692
	On-time	0.796	0.877	0.834		
	Late	0.644	0.767	0.700		
	Avg/Total	0.721	0.692	0.692		
LGBM with hyperparameter optimization	Early	0.742	0.444	0.555	0.749	0.700
	On-time	0.794	0.883	0.836		
	Late	0.660	0.772	0.711		
	Avg/Total	0.732	0.700	0.701		

Table 4.3 - Classification Report for the 3-class problem

In contrast to the previous analysis, this 3-class classification introduced the new target level of early payments, so it makes sense to focus this analysis on the performance of the classifiers for early payments. In this case, the recall for each early payment class matters because it is the percentage of correctly predicted early payments out of the number of actual early payments. It is interesting that

even though the complexity is increased by adding another level to predict, the performance of the classifiers on a multiclass problem is still improved, especially when comparing with the baseline for a 3-class problem. This time, LGBM performed better, reaching 70% of balanced accuracy. It also showed better recall in early payments than other algorithms. GBM followed LGBM, again with a very similar performance to RF, although RF reached the best recall for late payments with 77,7%. Both LR, SVM and KNN had the lowest performance not being able to surpass 61% of balanced accuracy. Nevertheless, looking at the generated confusion matrix presented in the annex section, we can see these models are biasing towards the majority class, which explains why that SVM had the best recall for on-time payments.

4.2.1. Results with Resampling

Using the resampled dataset described in section 3.4.1.2 Imbalanced Dataset – Resampling Technique, RF, GBM and LightGBM were tested. Below is the classification report:

		PRECISION	RECALL	F1-SCORE	ACCURACY	BALANCED ACCURACY
RF	Early	0.647	0.498	0.562	0.726	0.695
	On-time	0.804	0.861	0.831		
	Late	0.67	0.726	0.696		
	Avg/Total	0.707	0.695	0.696		
GBM	Early	0.610	0.557	0.582	0.739	0.700
	On-time	0.832	0.797	0.814		
	Late	0.633	0.747	0.685		
	Avg/Total	0.692	0.700	0.694		
LGBM	Early	0.701	0.538	0.608	0.760	0.719
	On-time	0.829	0.864	0.846		
	Late	0.659	0.756	0.704		
	Avg/Total	0.730	0.719	0.719		

Table 4.4 - Results with resampling for the 3-class problem

The balanced accuracies for GBM and RF did not improve a lot when comparing the results with the resampled dataset. Value is added when comparing the recall scores for early payments, which are higher in the experiments with resampling. GBM stands out with the highest early payment recall score of 58%. Of all the 3 problems, the 3-class was the less imbalanced which might explain the marginal results of these models with the resampled set. The best performance goes again to the LGBM model, which reached 71,9% of balanced accuracy, and the best recall scores for on-time payments and late payment (86,4% and 75,6% respectively).

4.3. Results for 5-Class Classification Problem:

Considering a 5-Class target (early, on-time, late, very late and critically late payments), using all the 49 features extracted from the dataset and a train-validation-test split of approximately 80:10:10, the following classification report was obtained:

		PRECISION	RECALL	F1-SCORE	ACCURACY	BALANCED ACCURACY
BASELINE	Early	0.000	0.000	0.000	0.515	0.200
	On-time	0.515	1.000	0.680		
	Late	0.000	0.000	0.000		
	Very Late	0.000	0.000	0.000		
	Critically Late	0.000	0.000	0.000		
	Avg/Total	0.103	0.200	0.136		
LR	Early	0.527	0.414	0.464	0.657	0.365
	On-time	0.731	0.885	0.801		
	Late	0.555	0.514	0.533		
	Very Late	0.000	0.000	0.000		
	Critically Late	0.000	0.012	0.008		
	Avg/Total	0.363	0.365	0.361		
SVM	Early	0.557	0.008	0.015	0.590	0.238
	On-time	0.539	0.990	0.698		
	Late	0.740	0.192	0.305		
	Very Late	0.000	0.000	0.000		
	Critically Late	0.000	0.000	0.000		
	Avg/Total	0.367	0.238	0.204		
KNN	Early	0.502	0.472	0.487	0.590	0.359
	On-time	0.751	0.728	0.739		
	Late	0.430	0.490	0.458		
	Very Late	0.084	0.080	0.082		
	Critically Late	0.006	0.025	0.010		
	Avg/Total	0.355	0.359	0.355		
RF	Early	0.701	0.495	0.580	0.715	0.437
	On-time	0.788	0.884	0.833		
	Late	0.566	0.685	0.620		
	Very Late	0.392	0.122	0.186		
	Critically Late	0.000	0.000	0.000		
	Avg/Total	0.489	0.437	0.444		
GBM	Early	0.692	0.466	0.557	0.709	0.434
	On-time	0.784	0.889	0.833		
	Late	0.560	0.671	0.610		
	Very Late	0.360	0.133	0.194		
	Critically Late	0.018	0.012	0.014		
	Avg/Total	0.483	0.434	0.442		
LGBM with hyperparameter optimization	Early	0.694	0.483	0.569	0.717	0.439
	On-time	0.786	0.892	0.836		
	Late	0.581	0.687	0.629		
	Very Late	0.411	0.134	0.202		
	Critically Late	0.000	0.000	0.000		
	Avg/Total	0.494	0.439	0.447		

Table 4.5 - Classification report for the 5-class problem

Once again, the top 3 classifiers in terms of balanced accuracy were LGBM (43,9%), RF (43,7%) and GBM (43,4%), in this order. Between LGBM and GBM, it is possible to notice that even though both accuracy and balanced accuracy are higher, the GBM classifier has a higher recall score for critically late payments. In fact, for some customers, it might be more beneficial to increse this metric when choosing a model but, unfortunately, most of the models present minimal recall scores – LGBM, for example, could not predict any critically late payments correctly (0% of recall). Surprisingly, the model with the best recall score for critically late payments is KNN, with only 2,5% recall. Furthermore, RF was the best model for predicting early payments with 49,5% recall. LGBM benefits from higher recall

in late and very late classes, so balanced accuracy is higher while also having high recall in early and on-time, which is why its balanced accuracy score was the highest.

4.3.1. Results with Resampling

		PRECISION	RECALL	F1-SCORE	ACCURACY	BALANCED ACCURACY
RF	Early	0.649	0.501	0.566	0.523	0.393
	On-time	0.898	0.552	0.684		
	Late	0.583	0.520	0.545		
	Very Late	0.177	0.361	0.237		
	Critically Late	0	0.037	0.001		
	Avg/Total	0.461	0.394	0.407		
GBM	Early	0.647	0.519	0.576	0.685	0.476
	On-time	0.812	0.856	0.833		
	Late	0.609	0.502	0.55		
	Very Late	0.222	0.440	0.295		
	Critically Late	0.008	0.062	0.013		
	Avg/Total	0.460	0.476	0.453		
LGBM	Early	0.650	0.535	0.587	0.699	0.480
	On-time	0.813	0.859	0.835		
	Late	0.602	0.560	0.580		
	Very Late	0.252	0.372	0.301		
	Critically Late	0.013	0.074	0.023		
	Avg/Total	0.466	0.480	0.465		

Table 4.6 - Results with resampling for the 5-class problem

Using the resampling technique, it is possible to increase the concerning performance of the models when predicting critically late payments. LGBM was the best model in terms of balanced accuracy (48,0%). With resampling, LGBM improved its capability of predicting critically late (7,4% of recall compared to 0% without resampling). However, this comes with the trade-off of being less capable of predicting late and on-time payments, which might be a result of applying random under-sampling to this class, but still being amongst the best performances of the experiments with a 5-class target.

It is also clear how re-sampling the training set influences the test results using the confusion matrices in figure 4.2. Comparing the two matrices, after applying resampling, it is possible to predict more very late, and critically late payments correctly, and early payments, which are the classes where SMOTE oversampling was used. This is, of course, at a cost of predicting other classes such as “On-time” and “Late” less accurately, in part because we lose some information when applying random under-sampling to these classes.

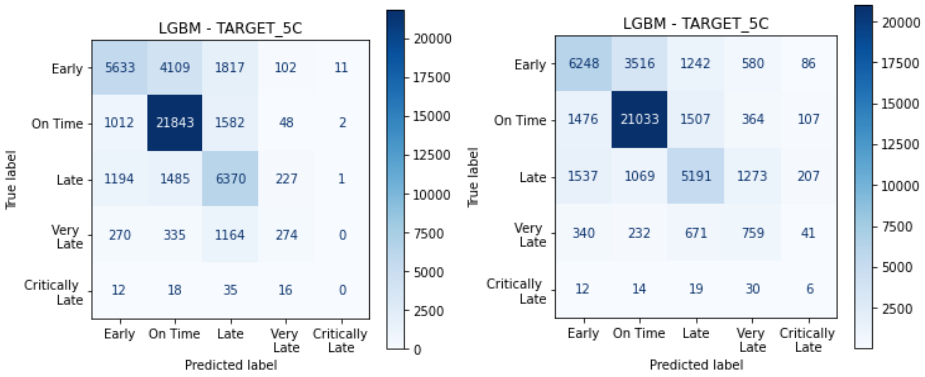


Figure 4.2 - 5-Class confusion matrices for GBM before resampling (left) and after resampling (right)

4.4. Feature Importance

To provide more insights into the models discussed, there are several techniques to extract information about which features were more important in each classifier's training for predicting the target variables. For linear models, such as LR, the final trained coefficients can be used to compute a score for each feature to assess their positive or negative influence on the outcome. This was the technique used to compute the feature importance of LR. As discussed, the LR classifier, provides the best explanatory power despite not having the best results, so the feature importance translates to how exactly each feature influenced the prediction. On the other hand, decision tree-based models are harder to interpret when comparing to linear models, which is why RF, GBM and LGBM packages usually have built-in methods to extract scores for the features. These scores can be computed in two ways: using gain or split importance. The scores of each feature are computed by summing up the gain of a split where a feature is used. Split importance is calculated by counting the number of times the feature was used to split in a certain model.

Since LGBM performed better in terms of balanced accuracy than the other models (using both imbalanced and balanced training sets) we looked at the feature importance to see which features were more or less important to the prediction results. Note that different models will use features differently, so feature importance plots for all the models are compiled in the annex section.

For the problem with 3-classes, the top 10 features and bottom 5 features using the gain and split importance measures are presented in table 4.7.

Most features that appear on the top 10 of both importance measures are customer historical features. In particular, those that were calculated using ratios such as the ratios of early and late invoices in quantity and value (RATIO_EARLY, RATIO_VAL_EARLY, RATIO_LATE, RATIO_VAL_LATE) and late invoices (RATIO_LATE) of the customer which appear in both rankings.

As expected, the number of days of the payment term (ZTERM_DAYS) is a very important feature, irrespective of the metric used to calculate the importance. There are also some date features that appear quite high up such as the ratio of the number of days from the due date until the end of the month (RATIO_DUE_DAYS_UNTIL_MONTH_END), the day of the due date (DUE_DATE_MONTH), and whether the due date falls on a weekday or not (DUE_DATE_WEEKDAY). However, flags to identify If

the due date falls in the beginning and or end of the month ended up not being so important for the prediction.

TOP 10	RATIO_EARLY	7451949	ZTERM_DAYS	16091
	RATIO_LATE	5510072	RATIO_DUE_DAYS_UNTIL_MONTH_END	13168
	NR_PAID	4509857	RATIO_EARLY	12717
	ZTERM_DAYS	2270344	AVG_DAYS_EARLY	12073
	DUE_DATE_WEEKDAY	2060105	RATIO_LATE	11871
	RATIO_VAL_LATE	1289689	AVG_VAL	11740
	RATIO_VAL_EARLY	989549	DUE_DATE_MONTH	11477
	VAL_TOTAL	967612	AVG_TERM	11102
	AVG_TERM	948055	RATIO_OUT_LATE	10822
	AVG_DAYS_EARLY	902967	AVG_DAYS_LATE	10504
BOTTOM 5	COUNTRY_PROFILE_USA	18443	COUNTRY_PROFILE_USA	383
	DUE_DATE_BEG_MONTH	16440	DUE_DATE_END_MONTH	282
	DUE_DATE_END_MONTH	14706	COUNTRY_PROFILE_CANADA	215
	COUNTRY_PROFILE_CANADA	11054	DUE_DATE_BEG_MONTH	134
	NEW_CUST	14	NEW_CUST	1

Table 4.7 - Top 10 and Bottom 5 features for LGBM with resampled dataset for the 3-class target

Notoriously, features using country profiles (e.g., COUNTRY_PROFILE_US) and new customer (NEW_CUST) were not considered so important as they generally occupy low places in these importance rankings, and the same happens in the ranking of most of the models studied.

5. CONCLUSION

This work discussed some of the most important topics in financial analysis: account receivables management. The project's goal was to develop a methodology capable of correctly predicting invoice outcomes in terms of their payment. Three different problems were constructed, each bringing a new level of complexity and each with different business goals.

Several techniques were used to create and transform features for the model. There were 49 features considered 24 invoice-levels features that describe each invoice and 25 customers-based features which were mostly calculated taking aggregates of the historical behaviour of each customer. It was accounted for the fact that this model would be later deployed in a production environment, so to avoid concept drift, the customer historical aggregate features were calculated using a window of 6 months to look back.

Several algorithms were tested for each problem. Overall, the most performant algorithm was LightGBM which was faster in training time and provided the highest values of the considered metrics for most problems. Other simpler models were also considered, such as LR, SVM, KNN, RF and GBM.

Because in some cases, the class imbalance would have a negative influence resulting in the models focusing more on the majority class and disregarding minority classes, a sampling technique that combined SMOTE oversampling and random user-sampling was applied to the best algorithms (RF, GBM and LGBM). Hyperparameter tuning for LightGBM with Bayesian Optimization technique was also used together with LGBM to search for an optimal combination of hyperparameters that could improve results. Both techniques proved to be successful. The experiments with resampled data proved to be better at reaching higher balanced accuracy values. In the end, as in most of the studies reviewed, the first results surpassed baseline conditions and additional studied topics for improvement such as Bayesian optimization and resampling technique added value to the final work, providing also interesting insights for future work.

In the binary problem, which was an attempt to predict whether an invoice would be on-time or delayed, we obtained the best results, as was expected. A tuned LightGBM model achieved 85,7% of accuracy and 80,02% of balanced accuracy, falling just a little bit short on the GBM model that used a re-sampled dataset which reached 82,2% of balanced accuracy but only 84% of standard accuracy. The recall was also used to differentiate which model performed best when actual specific payment. In this case, recall was used to analyze late payments, where GBM with a re-sampled dataset won with a recall of 79,2%.

In the multiclass cases, the best models were similar. However, it was harder to get good performances as the number of levels in the target grew, especially for the 5-class problem where very late and critically late had very few examples. Even so, for the 3-class problem, it was possible to reach 70% of balanced accuracy using LightGBM (74,9% of standard accuracy) and 71,9% of balanced accuracy with the same model using a resampled dataset (76% of accuracy).

When predicting the outcome target of 5 classes, the results deteriorated. The best model was LGBM which reached 43,9% of balanced accuracy (71,7% of accuracy) and with a resampled dataset for training, 48% of balanced accuracy (69,9% of accuracy).

This study presents already interesting results when compared to other studies, which, even though it is hard to compare considering that invoice payments differ a lot as well as the problem structure and metrics used.

A dataset of closed invoices from the company was explored for this purpose, and it was used to provide leverage for the models to learn with the data. This solution is now part of a group of value-adding initiatives that make use of not widely used datasets in the company for machine-learning tasks. The resulting model will provide value to the cash collectors as they can now get more insights about how new invoices that fall in the system will be paid in the future.

6. RECOMENDATIONS FOR FUTURE WORK

The present study indicates that there is still room to improve in predicting invoice outcomes of account receivables. This study is not exhaustive, and there are still other techniques that could be experimented with to improve results.

The author suggests trying other techniques to overcome the class imbalance issue and potentially increase the performance results. Experimenting with other configurations of the SMOTE and under-sampling could be useful to improve results. Another suggestion is to try and take a different approach using, for example, Cost Learning to give cost to those instances we want to make sure are predicted very well, such as the extreme class of critically late payment.

Experimenting with other configurations of fine-tuning could also add value, as well as, using Bayesian Optimization to tune hyperparameters of other algorithms such as GBM and RF.

Another suggestion is to transform features differently and use feature importance to apply feature selection techniques to see if results or training time can be improved. Feature augmentation could also be interesting, as there are business areas that collect dispute information such as calls and follow-ups regarding outstanding and late invoices. This data could be very informative for these models, but unfortunately, at the time of this study, this data was not available.

It would also be interesting to see how the model performs when applied to other country areas, as most of the features were designed to scale within Siemens financial areas.

Out of the scope of the project, but also an interesting option to explore, would be the creation of a dashboard to provide data insights of account receivables. The dashboard could be tailored to provide the cash collectors visualizations of the data use, and predictions that resulted from this project. It would also benefit from a monthly monitoring page of the model offering transparency to the users, and actionable insights for the development team to make improvements.

7. REFERENCES

Abbott, D. (2014). Applied Predictive Analytics. Principles and techniques for the professional data analyst. In *Wiley* (Vol. 53). https://doi.org/10.1017/CBO9781107415324.004

Appel, A. P., Oliveira, V., Lima, B., & Dec, L. G. (2020). *Optimize Cash Collection: Use Machine learning to Predicting Invoice Payment. (arXiv:1912.10828v1 [cs.LG])*. Retrieved from http://arxiv.org/abs/1912.10828

Azevedo, A., & Santos, M. F. (2008). KDD , SEMMA AND CRISP-DM : A PARALLEL OVERVIEW Ana Azevedo and M . F . Santos. *IADIS European Conference Data Mining*, 182–185. Retrieved from http://recipp.ipp.pt/handle/10400.22/136%0Ahttp://recipp.ipp.pt/bitstream/10400.22/136/3/KDD-CRISP-SEMMA.pdf

Bennett, J., & Lanning, S. (2007). *The Netflix Prize*.

Bennett, K. P., & Campbell, C. (2000). Support Vector Machines: Hype or Hallelujah? *SIGKDD Explorations*, *2*(2), 1–13.

Bishop, C. M. (2006). Pattern Recognition and Machine Learning. In *Springer*.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*, 123–140. https://doi.org/10.3390/risks8030083

Breiman, L. (2001). Random forests. *Machine Learning*, *45*, 5–32. https://doi.org/10.1201/9780429469275-8

Brodersen, K. H., Ong, C. S., Stephan, K. E., & Buhmann, J. M. (2010). *The balanced accuracy and its posterior distribution*. 3125–3128. https://doi.org/10.1109/ICPR.2010.764

Brownlee, J. (2017). A Gentle Introduction to Concept Drift in Machine Learning Changes to Data Over Time. Retrieved from https://machinelearningmastery.com/gentle-introduction-concept-drift-machine-learning/

Brownlee, J. (2020). How to Scale Data With Outliers for Machine Learning. Retrieved from https://machinelearningmastery.com/robost-scaler-transforms-for-machine-learning/

Buda, M., Maki, A., & Mazurowski, M. A. (2017). A systematic study of the class imbalance problem in convolutional neural networks. *CoRR*, *abs/1710.0*, 249–259. Retrieved from http://arxiv.org/abs/1710.05381

Burkov, A. (2019). *The 100-Page Machine Learning Book*. 160. https://doi.org/10.1111/j.1468-0394.1988.tb00341.x

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, *16*, 321–357.

Chen, C., Liaw, A., & Breiman, L. (2004). *Using Random Forest to Learn Imbalanced Data*. (1999), 1–12.

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

Cheong, M. L. F., Cheong, M. L. F., & Shi, W. (2018). Customer Level Predictive Modeling for Accounts

Receivable to Reduce Intervention Actions. *Proceedings of the 14th International Conference on Data Science (ICDATA 2018), Las Vegas, Nevada, July 30 - August 2.*, *2018*(Icdata), Research Collection School Of Information Systems.

Cortes, C., & Vapnik, V. (1995). Support-Vector Networks. *Machone Learning*, *20*(2), 273–297. https://doi.org/10.1111/j.1747-0285.2009.00840.x

Dibike, Y., Velickov, S., & Solomatine, D. (2000). Support vector machines: Review and applications in civil engineering. *2nd Joint Workshop on Application of AI in Civil Engineering*, (March), 215–218. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.200.2874&rep=rep1&type=pdf

Domingos, P. (2015). *The Master Algorithm*.

Dorogush, A. V., Ershov, V., & Gulin, A. (2018). CatBoost: Gradient boosting with categorical features support. *ArXiv*, 1–7.

Downey, A. B. (2011). Think Stats. In *Book*. https://doi.org/10.1017/CBO9781107415324.004

Ezvan, J.-L. (2018). *Predicting late payment of an invoice*.

Fernández, A., García, S., Galar, M., & Prati, R. C. (2018). *Learning from Imbalanced Data Sets*. Springer.

Fradkov, A. L. (2020). Early History History of of Machine Learning. *IFAC PapersOnLine*, *53*(2), 1385–1390. https://doi.org/10.1016/j.ifacol.2020.12.1888

Freund, Y., Schapire, R. E., & Hill, M. (1996). *Experiments with a New Boosting Algorithm*.

Friedman, B., & Stone, O. (1984). *Classification And Regression Trees*.

Friedman, J. H. (2001). Greedy Function Approximation: A Gradien Boosting Machine. *The Annals of Statistics*, *29*(5), 44.

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics and Data Analysis*, *38*(4), 367–378. https://doi.org/10.1016/S0167-9473(01)00065-2

Friedman, J. H., & Popescu, B. E. (2004). Gradient Directed Regularization for Linear Regression and Classification. *Manuscript*, *2004*(3), 1–40. Retrieved from stanford.

Galar, M., Fernández, A., Barrenechea, E., Bustince, H., & Herrera, F. (2011). An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes. *Pattern Recognition*, *44*(8), 1761–1776. https://doi.org/10.1016/j.patcog.2011.01.017

Grandini, M., Bagli, E., & Visani, G. (2020). Metrics for multi-class classification: An overview. In *arXiv*.

Hale, J. (2018). 7 Data Types: A Better Way to Think about Data Types for Machine Learning.

Hosmer, D. W., Taber, S., & Lemeshow, S. (1991). The importance of assessing the fit of logistic regression models: A case study. *American Journal of Public Health*, *81*(12), 1630–1635. https://doi.org/10.2105/AJPH.81.12.1630

Hosmer, David W, & Lemeshow, S. (2000). *Applied Logistic Regression* (pp. 1–375). pp. 1–375.

Hu, W. (2009). *Overdue Invoice Forecasting and Data Mining*.

Japkowicz, N., & Stephen, S. (2002). The Class Imbalance Problem: A Systematic Study. *Intelligent Data Analysis*, *6*(5), 429–449.

John D. Kelleher, Namee, B. Mac, & D'Arcy, A. (2015). *Fundamentals of Machine Learning fo Predictive Data Analytics: Algorithms, worked examples, and Case Studies*.

Jones, S. A. (2018). Trade and Receivables Finance. In *Trade and Receivables Finance*. https://doi.org/10.1007/978-3-319-95735-7

Joshi, P. (2016). *Python Machine Learning Cookbook*. Retrieved from http://proquest.safaribooksonline.com.ezproxy.lib.vt.edu/9781786464477

K. Dalal, M., & A. Zaveri, M. (2011). Automatic Text Classification: A Technical Review. *International Journal of Computer Applications*, *28*(2), 37–40. https://doi.org/10.5120/3358-4633

Kaufman, S., & Perlich, C. (2011). *Leakage in Data Mining : Formulation, Detection, and Avoidance*. 556–563.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., … Liu, T. Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, *2017-Decem*(Nips), 3147–3155.

Kearns, M. (1988). Thoughts on hypothesis boosting. *Unpublished Manuscript*, *45*, 105.

Korotina, A., Mueller, O., & Debortoli, S. (2015). *Real-time Business Process Intelligence. Comparison of different architectural approaches using the example of the order-to-cash process.* Retrieved from http://aisel.aisnet.org/wi2015

Kuhn, M., & Johnson, K. (2013b). *Applied Predictive Modeling with Applications in R*. Retrieved from http://appliedpredictivemodeling.com/s/Applied_Predictive_Modeling_in_R.pdf

Larose, D. T., & Larose, C. D. (2015). Data Mining and Predictive Analytics. *Data Mining and Predictive Analysis*. https://doi.org/10.1016/b978-0-12-800229-2.00003-1

Leo Breiman. (1996). *Out-of-Bag Estimation*.

Malley, J. D., Kruppa, J., Dasgupta, A., Malley, K. G., & Ziegler, A. (2012). Probability Machines: Consistent probability estimation using nonparametric learning machines. *Methods of Information in Medicine*, *51*(1), 74–81. https://doi.org/10.3414/ME00-01-0052

Mitchell, T. M. (1997). *Machine Learning*. https://doi.org/10.1109/ICDAR.2019.00014

Mountrakis, G., Im, J., & Ogole, C. (2011). Support vector machines in remote sensing: A review. *ISPRS Journal of Photogrammetry and Remote Sensing*, *66*(3), 247–259. https://doi.org/10.1016/j.isprsjprs.2010.11.001

Narayan, P. K., Phan, D. H. B., & Liu, G. (2021). COVID-19 lockdowns, stimulus packages, travel bans, and stock returns. *Finance Research Letters*, *38*(July 2020), 101732. https://doi.org/10.1016/j.frl.2020.101732

Natekin, A., & Knoll, A. (2013). Gradient boosting machines, a tutorial. *Frontiers in Neurorobotics*, *7*(DEC). https://doi.org/10.3389/fnbot.2013.00021

Navon, R. (1996). *Company-level cash flow management.pdf* (pp. 22–29). pp. 22–29.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., … Duchesnay, E. (2011).

Scikit-learn: Machine Learning in {P}ython. *Journal of Machine Learning Research*, *12*, 2825–2830.

Peiguang, H. (2015). Predicting and Improving Invoice-to-Cash Collection Through Machine Learning. *Master Thesis*, 1–92. Retrieved from http://dspace.mit.edu/bitstream/handle/1721.1/99584/925473704-MIT.pdf?sequence=1

Pete, C., Julian, C., Randy, K., Thomas, K., Thomas, R., Colin, S., & Wirth, R. (2000). CRISP-DM 1.0. Retrieved March 1, 2020, from CRISP-DM Consortium website: https://www.the-modeling-agency.com/crisp-dm.pdf

Pfohl, H. C., & Gomm, M. (2009). Supply chain finance: Optimizing financial flows in supply chains. *Logistics Research*, *1*(3–4), 149–161. https://doi.org/10.1007/s12159-009-0020-y

Samuel, A. L. (1959). Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*, *3*(3), 210–229.

Saravanan, R., & Sujatha, P. (2018). A State of Art Techniques on Machine Learning Algorithms : A Perspective of Supervised Learning Approaches in Data Classification. *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*, (Iciccs), 945–949.

Schapire, R. E. (1990). The Strength of Weak Learnability. *Machine Learning*, *5*(2), 197–227. https://doi.org/10.1023/A:1022648800760

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical Bayesian Optimization of Machine Learning Algorithms. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems* (Vol. 25). Retrieved from https://proceedings.neurips.cc/paper/2012/file/05311655a15b75fab86956663e1819cd-Paper.pdf

Stevens, S. S. (1946). On the Theory of Scales of Measurement. *Science, New Series*, *103*(2684), 677–680.

Tater, T., Dechu, S., Mani, S., & Maurya, C. (2016). Prediction of Invoice Payment Status in Account Payable Business Process Tarun. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *9936 LNCS*, 165–180.

Thrampoulidis, C., Oymak, S., & Soltanolkotabi, M. (2020). Theoretical insights Into multiclass classification: A high-dimensional asymptotic view. *ArXiv*, (NeurIPS), 1–14.

Tracy, T. C., & Tracy, J. A. (2012). *Cash Flow for Dummies*.

Tsymbal, A. (2004). The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin*.

Vapnik, V. N. (1998). *Statistical Learning Theory*. John Wiley & Sons.

Vapnik, V. N. (2000). *The Nature of Statistical Learning Theory* (2nd ed.). Springer.

Vernimmen, P., Le Fur, Y., Dallochio, M., Salvi, A., & Quiry, P. (2017). Corporate Finance. In *Corporate Finance*. https://doi.org/10.1002/9781119424444

Wang, D. (2017). LightGBM : An Effective miRNA Classification Method in Breast Cancer Patients. *ICCBB 2017: Proceedings of the 2017 International Conference on Computational Biology and Bioinformatics*, 7–11. https://doi.org/https://doi.org/10.1145/3155077.3155079

Webb, G. I., Lee, L. K., Goethals, B., & Petitjean, F. (2017). Understanding concept drift. *ArXiv*.

Wirth, R. (2000). CRISP-DM : Towards a Standard Process Model for Data Mining. *Proceedings of the Fourth International Conference on the Practical Application of Knowledge Discovery and Data Mining*, (24959), 29–39. https://doi.org/10.1.1.198.5133

Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). Data Mining: Practical Machine Learning Tools and Techniques. In *Data Mining: Practical Machine Learning Tools and Techniques*. https://doi.org/10.1016/c2009-0-19715-5

Wright, J. A. (n.d.). *Order-to-Cash: Unlocking Corporate Value*.

Xiaolei, S., Mingxi, L., & Zeqian, S. (2020). A novel cryptocurrency price trend forecasting model based on LightGBM. *Finance Research Letters*, *32*(November 2018). https://doi.org/10.1016/j.frl.2018.12.032

Zeng, S., Melville, P., Lang, C. A., Boier-Martin, I., & Murphy, C. (2008). Using predictive analysis to improve invoice-to-cash collection. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1043–1050. https://doi.org/10.1145/1401890.1402014

Zheng, A., & Casari, A. (2018). *Feature Engineering for Machine Learning*.

Zhou, Y., Wilkinson, D., Schreiber, R., & Pan, R. (2008). Large-Scale Parallel Collaborative Filtering for the Netflix Prize. In R. Fleischer & J. Xu (Eds.), *Algorithmic Aspects in Information and Management* (pp. 337–348). Berlin, Heidelberg: Springer Berlin Heidelberg.

8. ANNEX

8.1. FINAL FEATURE SET

Below is the complete final set of features (or predictors) used for training of the classification models. Features from 1 to 25 are **Customer Historical Features** extracted to provide knowledge to the model about past behavior of each customer. Features from 26 to 49 are called **Invoice-Level Features** which main goal is to describe the invoice.

#	*FEATURE NAME*	*FEATURE*	*DESCRIPTION*
1	***NR_PAID***	Number of Invoices (paid)	Number of invoices paid by customer that precede the creation date.
2	***NR_LATE***	Number of Late invoices	Number of invoices the customer paid with delay (more than 3 days after the due date) that precede the creation date.
3	***NR_EARLY***	Number of Early Invoices	Number of invoices the customer paid in advance (less than 3 days before the due date) that precede the creation date.
4	***RATIO_LATE***	Delay ratio	Ratio of late invoices of the customer that precedes the creation date. Ratio of NR_LATE over NR_PAID.
5	***RATIO_EARLY***	Early ratio	Ratio of early invoices of the customer that precedes the creation date. Ratio of NR_EARLY over NR_PAID.
6	**VAL_TOTAL**	Total Value Paid	
7	**VAL_LATE**	Total Value Delayed	Sum of the value of late invoices by customer that precede the creation date.
8	**VAL_EARLY**	Total Value Early	Sum of the value of early invoices by customer that precede the creation date.
9	**RATIO_VAL_LATE**	Late Value Ratio	Ratio sum value of late invoices of the customer that precedes the creation date. Ratio of VAL_LATE over VAL_TOTAL.
10	**RATIO_VAL_EARLY**	Early Value Ratio	Ratio of the sum value of early invoices of the customer that precedes the creation date. Ratio of VAL_EARLY over VAL_TOTAL.
11	**AVG_DAYS_LATE**	Average days paid invoices were late	Average number of days late of invoices paid late by customer preceding the creation date.
12	**AVG_DAYS_EARLY**	Average days paid invoices were early	Average number of days early of the invoices paid early by customer preceding the creation date.
13	**NR_OUT**	Number of invoices that are outstanding	Number of invoices by customer registered preceding the creation date, which are not paid yet (outstanding). Number of invoices that the customer still has to pay.
14	**NR_OUT_LATE**	Number of invoices that are outstanding but already late	Number of invoices by customer registered preceding the creation date, which are not paid yet (outstanding) but are already past due date (late). Number of late invoices that the customer still has to pay.

15	**RATIO_OUT_LATE**	Ratio of late outstanding invoices	Ratio of the late outstanding invoices of the customer preceding the creation date. Ratio of NR_OUT_LATE over NR_OUT.
16	**VAL_OUT**	Total Value of invoices that are outstanding	Sum of the invoice values by customer registered preceding the creation date, which are not past due date yet (outstanding). Total value of the invoices that the customer still has to pay.
17	**VAL_OUT_LATE**	Total Value of invoices that are outstanding but already late	Sum of the invoice values by customer registered preceding the creation date, which are not past due date yet (outstanding)but are already past due date (late). Total value of the late invoices that the customer still has to pay.
18	**RATIO_VAL_OUT_LATE**	Ratio of the total value late outstanding invoices (14/13)	Ratio of the sum value of late outstanding invoices of the customer preceding the creation date Ratio of VAL_OUT_LATE over VAL_OUT.
19	**AVG_DAYS_LATE_OUT**	Average days late of outstanding invoices	Average number of days late of outstanding late invoices by customer preceding the creation date.
20	**STD_DAYS_LATE**	Standard deviation of days of late invoices	Standard deviation of the number of days late of late invoices by customer preceding the creation date.
21	**STD_DAYS_EARLY**	Standard deviation of days of early invoices	Standard deviation of the number of days early of the invoices paid early by customer preceding the creation date.
22	**PAY_FREQ**	Ratio of actual payments	Ratio of invoices paid over all invoices registered by customer preceding the creation date.
23	**AVG_VAL**	Average value of invoices	Average value of the invoices by customer preceding the creation date.
24	**AVG_TERM**	Average payment term	Average number of days between when invoice is created and the day it is due, by customer preceding the creation date.
25	**NEW_CUST**	New customer flag (0,1)	Binary feature to indicate whether it is the customer's first invoice (1) or not (0)
26	**LOG_VALUE_LC**	Amount of the invoice	Log Transformation
27	**INDUSTRY**	Industry Number	
28	**INDUSTRY_GROUP**	First 2 digits of Industry number which is the industry group	
29	**COUNTRY_PROFILE_CANADA**	One hot encoding based on the most important countries or regions according to cash collectors. Feature based on customer country (KNA1_LAND1)	1 = Invoice's customer is located in Canada 0 =Invoice's customer is not located in Canada
30	**COUNTRY_PROFILE_GERMANY**		1 = Invoice's customer is located in Germany 0 =Invoice's customer is not located in Germany
31	**COUNTRY_PROFILE_MEXICO**		1 = Invoice's customer is located in Mexico 0 =Invoice's customer is not located in Mexico
32	**COUNTRY_PROFILE_SOUTH_AMERICA**		1 = Invoice's customer is located in South America 0 =Invoice's customer is not located in South America

33	**COUNTRY_PROFILE_USA**		1 = Invoice's customer is located in USA 0 =Invoice's customer is not located in USA
34	**COUNTRY_PROFILE_EUROPE_WO_GERMANY**		1 = Invoice's customer is located in Europe (excl. Germany) 0 =Invoice's customer is not located in Europe (excl. Germany)
35	**DUE_DATE_WEEKEND**	Flag indicating whether the invoice due date falls on a weekend	1 = Invoice due date falls on a weekend 0 = Invoice due date falls on a weekday
36	**DUE_DATE_BEG_MONTH**	Flag indicating whether the invoice due date falls at the beginning of the month (in the first 3 days)	
37	**DUE_DATE_MID_MONTH**	Flag indicating whether the invoice due date falls at the middle of the month (3 days in the middle of the month)	
38	**DUE_DATE_END_MONTH**	Flag indicating whether the invoice due date falls at the end of the month (in the last 3 days)	
39	**DUE_DATE_MONTH**	Month of the invoice's due date	
40	**DUE_DATE_WEEKDAY**	Day of the week of the invoice's due date	
41	**RATIO_DUE_DAYS_UNTIL_MONTH_END**	Ratio of the number of days from the due date until the end of the month	Ratio of the number of days between the due date of the invoice until the end of the month over the number of days in the invoice's due date month DUE_DAYS_UNTIL_MONTH_END/DUE_DATE_DAYS_IN_MONTH
42	**ZTERM_DAYS**	Number of days between the creation of the invoice and its stipulated due date	Days between BUDAT and DUE_DATE. ZTERM converted into numeric
43	**SPECIAL_TERMS**	Flag indicating if invoice was contracted under a special term or not	1 = special payment terms, 0 = common payment term with X days of buffer
44	**RATING_1_1000**	Customer rating provided by Siemens Financial Services. Provides the opinion about the creditworthiness of the company and takes into consideration the country, industry, and financial risk.	Transform rating from 1 to 1000, being 1 the best capacity to meet financial obligations and 1000 the most vulnerable invoices

46	**RATING_CATEGORY**	Customer rating provided by Siemens Financial Services. Provides the opinion about the creditworthiness of the company and takes into consideration the country, industry, and financial risk.	1 = Green Class 2 = Yellow Class 3 = Orange Class 4 = Red Class
45	**RATING_CHANGE_QNT**	Quantifies how many levels the invoice customer increased or decreased its rating	Difference between the current rating and the historic rating. E.g. A change of 5 means it increased 5 levels, a change of -5 means it decreased 5 levels and 0 means it stayed the same
46	**RATING_NEGATIVE_CHANGE**	Flag indicating if the change in rating was negative or not	1 = negative change, 0 = no change or positive change
47	**LOG_SKFBT**	Amount eligible for cash discount in document in local currency	Log transformation
48	**LOG_DBEKR**	Recommended credit limit	Log transformation
49	**LOG_KLIMK**	Customer's credit limit	Log transformation

Table 8.1 - Complete feature set

8.2. TRAIN, VALIDATION AND TEST DISTRIBUTIONS FOR DIFFERENT TARGETS

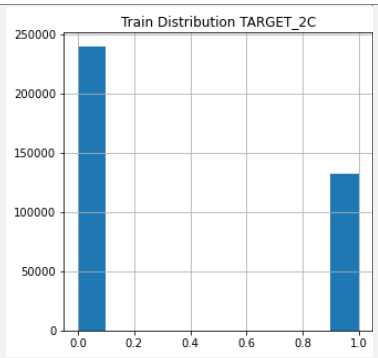
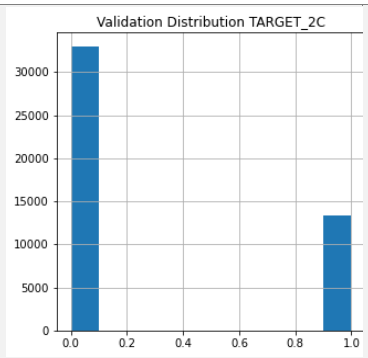
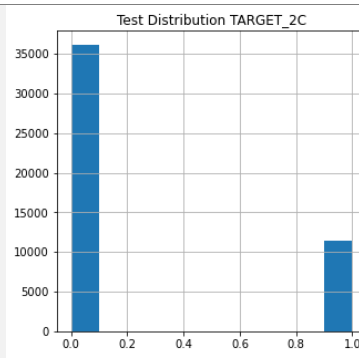
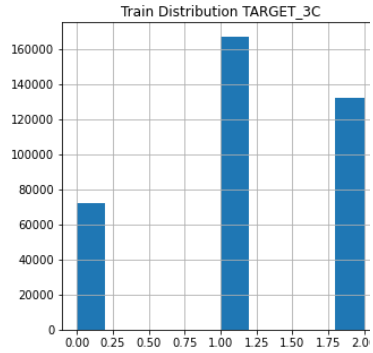
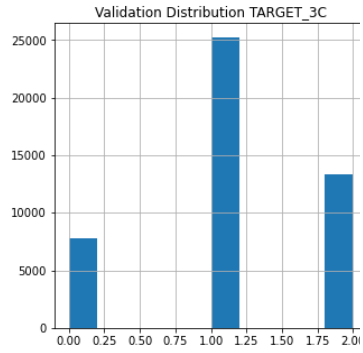
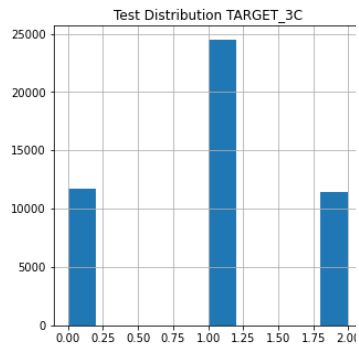
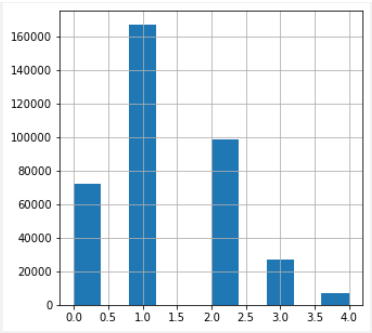
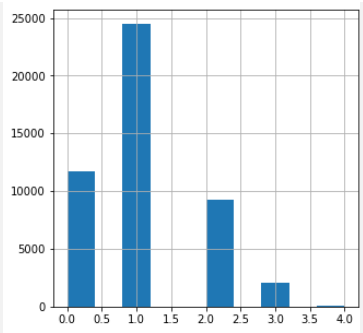
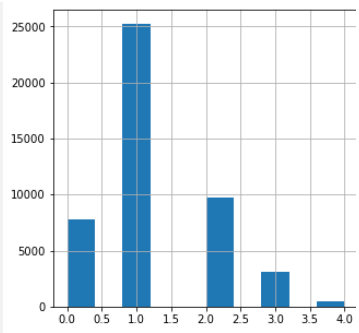
	TRAIN	VALIDATION	TEST
2-CLASS TARGET			
3-CLASS TARGET			
5-CLASS TARGET			

Table 8.2 -Train, Validation and Test distributions for different targets

8.3. MODEL HYPERPARAMETER SET UP

This project developed using the open-source coding language Python (version 3.7.9). The models used belong to two main python packages: LR, SVM, KNN, DT, RF, GBM are Scikit-Learn implementations (Pedregosa et al., 2011); and LGBM is an implementation developed by Microsoft and other contributors (Ke et al., 2017). Table 8.3 describes the hyperparameter set up for each model and problem using the prementioned python packages. The hyperparameters in bold are the ones that were not defined as default or (in the case of LGBM) were found using hyperparameter tuning. The hyperparameters for LGBM were optimized using Bayesian Optimization, thus the description in table 8.3 pertains the best combination of parameters resulting from an optimization run of 50 iterations.

MODEL	PROBLEM	HYPERPARAMETERS
LR	All	'C': 1.0, 'class_weight': None, 'dual': False, 'fit_intercept': True, 'intercept_scaling': 1, 'l1_ratio': None, **'max_iter': 200**, 'multi_class': 'auto', 'n_jobs': None, 'penalty': 'l2', 'random_state': 0, 'solver': 'lbfgs', 'tol': 0.0001, 'verbose': 0, 'warm_start': False
SVM	All	'C': 1.0, 'class_weight': None, **'dual': False**, 'fit_intercept': True, 'intercept_scaling': 1, 'loss': 'squared_hinge', 'max_iter': 1000, 'multi_class': 'ovr', 'penalty': 'l2', 'random_state': 0, 'tol': 0.0001, 'verbose': 0
KNN	All	'algorithm': 'auto', 'leaf_size': 30, 'metric': 'minkowski', 'metric_params': None, 'n_jobs': None, **'n_neighbors': 5**, 'p': 2, 'weights': 'uniform'
DT	All	'ccp_alpha': 0.0, 'class_weight': None, **'criterion': 'gini', 'max_depth': 3**, 'max_features': None, 'max_leaf_nodes': None, 'min_impurity_decrease': 0.0, 'min_impurity_split': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, 'random_state': 0, 'splitter': 'best'
RF	All	**'bootstrap': True**, 'ccp_alpha': 0.0, 'class_weight': None, **'criterion': 'gini'**, 'max_depth': None, 'max_features': 'auto', 'max_leaf_nodes': None, 'max_samples': None, 'min_impurity_decrease': 0.0, 'min_impurity_split': None, 'min_samples_leaf': 1, 'min_samples_split': 2, 'min_weight_fraction_leaf': 0.0, **'n_estimators': 100**, 'n_jobs': None, 'oob_score': False, 'random_state': 0, 'verbose': 0, 'warm_start': False
GBM	All	'ccp_alpha': 0.0, **'criterion': 'friedman_mse'**, 'init': None, **'learning_rate': 0.01**, 'loss': 'deviance', **'max_depth': 4**, 'max_features': None, 'max_leaf_nodes': None, 'min_impurity_decrease': 0.0, 'min_impurity_split': None, 'min_samples_leaf': 1, **'min_samples_split': 5**, 'min_weight_fraction_leaf': 0.0, 'n_estimators': 500, 'n_iter_no_change': None, 'random_state': 0, 'subsample': 1.0, 'tol': 0.0001, 'validation_fraction': 0.1, 'verbose': 0, 'warm_start': False

Table 8.3 - Model hyperparameter set up – Scikit-Learn models

MODEL	PROBLEM	HYPERPARAMETERS
LGBM	2-Class	'objective': 'multiclassova', 'learning_rate': 0.01, 'boosting_type': 'gbdt', 'tree_learner': 'voting', 'metric': 'multi_logloss', 'num_class': 2, 'num_threads': 7, 'seed': 0, 'verbose': -1, **'num_leaves': 14, 'max_depth': 24, 'feature_fraction': 0.741276256129203, 'bagging_fraction': 0.9261587313890257, 'min_split_gain': 0.02687457262331083, 'min_child_weight': 7.267856885834467, 'lambda_l1': 0.24767108279809436, 'lambda_l2': 1.9950802817718365,** 'num_iterations': 10000, 'early_stopping_round': 1000
	3-Class	'objective': 'multiclassova', 'learning_rate': 0.01, 'boosting_type': 'gbdt', 'tree_learner': 'voting', 'metric': 'multi_logloss', 'num_class': 3, 'num_threads': 7, 'seed': 0, 'verbose': -1, **'num_leaves': 80, 'max_depth': 24, 'feature_fraction': 0.9, 'bagging_fraction': 0.8, 'min_split_gain': 0.001, 'min_child_weight': 28.81244517690738, 'lambda_l1': 0.0, 'lambda_l2': 3.0**, 'num_iterations': 10000, 'early_stopping_round': 1000
	5-Class	'objective': 'multiclassova', 'learning_rate': 0.01, 'boosting_type': 'gbdt', 'tree_learner': 'voting', 'metric': 'multi_logloss', 'num_class': 5, 'num_threads': 7, 'seed': 0, 'verbose': -1, **'num_leaves': 35, 'max_depth': 29, 'feature_fraction': 0.8524139636494089, 'bagging_fraction': 0.866250039610247, 'min_split_gain': 0.003082937444810488, 'min_child_weight': 38.72129976791196, 'lambda_l1': 0.3720389146692793, 'lambda_l2': 2.08745231796963,** 'num_iterations': 10000, 'early_stopping_round': 1000
LGBM WITH RESAMPLED DATASET	2-Class	'objective': 'multiclassova', 'learning_rate': 0.01, 'boosting_type': 'gbdt', 'tree_learner': 'voting', 'metric': 'multi_logloss', 'num_class': 2, 'num_threads': 7, 'seed': 0, 'verbose': -1, **'num_leaves': 14, 'max_depth': 24, 'feature_fraction': 0.741276256129203, 'bagging_fraction': 0.9261587313890257, 'min_split_gain': 0.02687457262331083, 'min_child_weight': 7.267856885834467, 'lambda_l1': 0.24767108279809436, 'lambda_l2': 1.9950802817718365,** 'num_iterations': 10000, 'early_stopping_round': 1000
	3-Class	'objective': 'multiclassova', 'learning_rate': 0.01, 'boosting_type': 'gbdt', 'tree_learner': 'voting', 'metric': 'multi_logloss', 'num_class': 3, 'num_threads': 7, 'seed': 0, 'verbose': -1, **'num_leaves': 31, 'max_depth': 25, 'feature_fraction': 0.9, 'bagging_fraction': 1.0, 'min_split_gain': 0.1, 'min_child_weight': 50.0, 'lambda_l1': 0.0, 'lambda_l2': 3.0,** 'num_iterations': 10000, 'early_stopping_round': 1000
	5-Class	'objective': 'multiclassova', 'learning_rate': 0.01, 'boosting_type': 'gbdt', 'tree_learner': 'voting', 'metric': 'multi_logloss', 'num_class': 5, 'num_threads': 7, 'seed': 0, 'verbose': -1, **'num_leaves': 65, 'max_depth': 18, 'feature_fraction': 0.7843378079803442, 'bagging_fraction': 0.8087932968250029, 'min_split_gain': 0.060922595424358, 'min_child_weight': 23.27290336273902, 'lambda_l1': 0.30873221892753655, 'lambda_l2': 2.8497566526225535**, 'num_iterations': 10000, 'early_stopping_round': 1000

Table 8.4 - Model hyperparameter set up – LGBM implementation with hyperparameter tuning

8.4. Results for Binary Classification Problem

Dummy (Baseline)

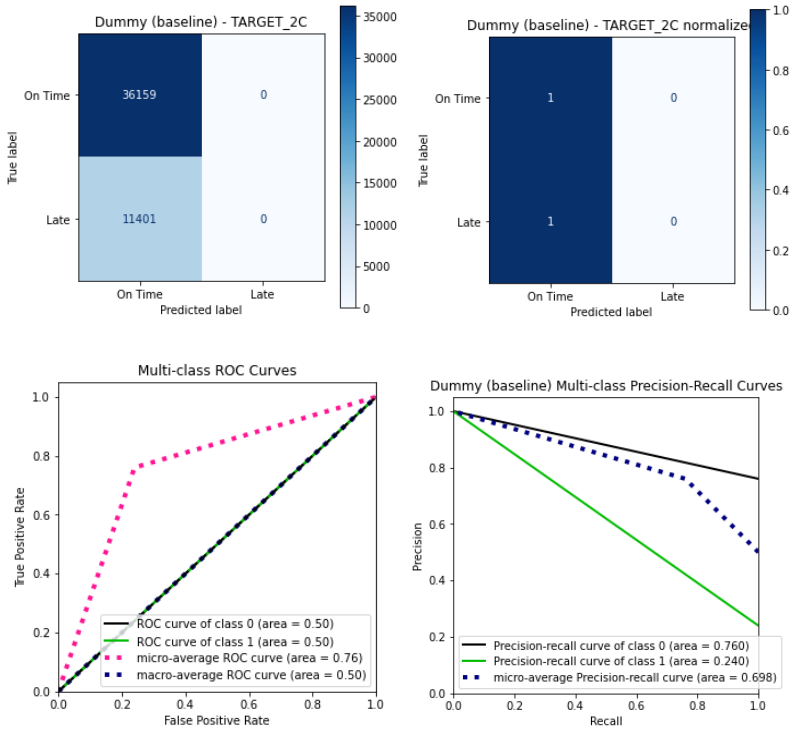


Figure 8.1 - Baseline results for the 2-class problem

LR (2C)

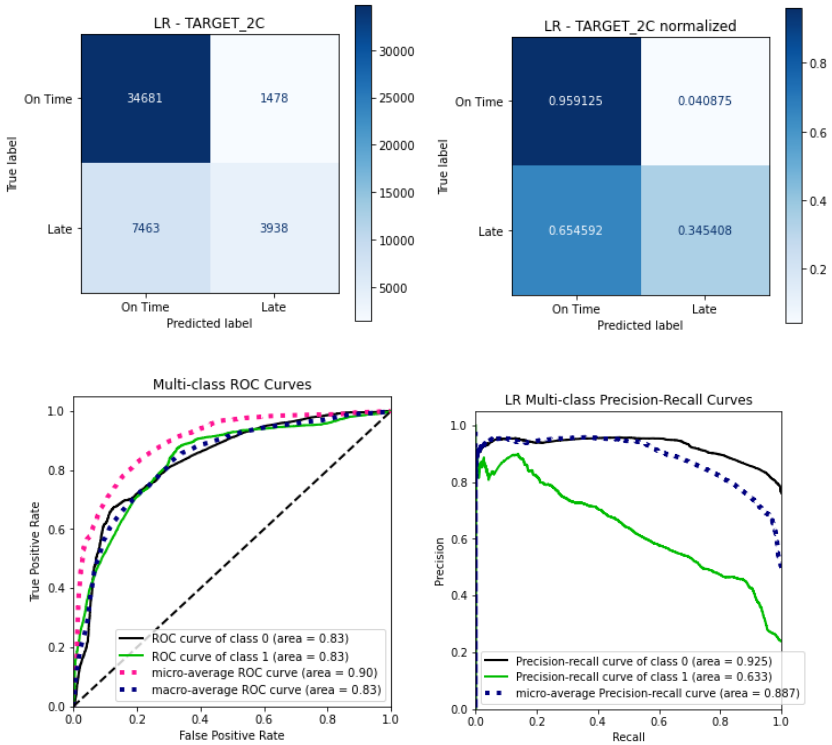


Figure 8.2 - Logistic Regression results for the 2-class problem

SVM (2C)

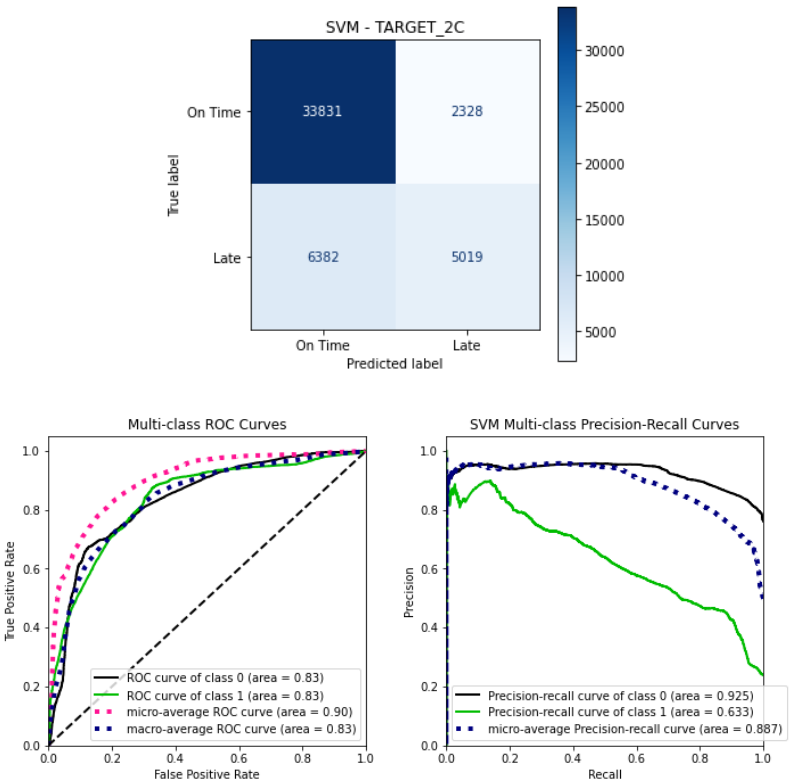


Figure 8.3 - SVM results for the 2-class problem

KNN (2C)

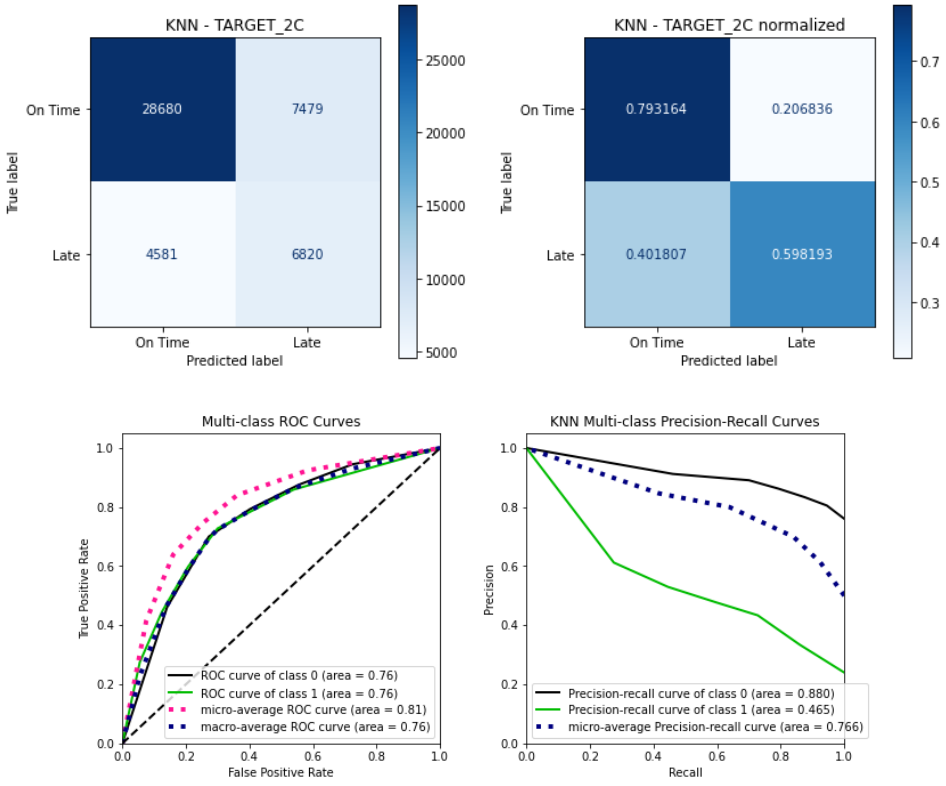


Figure 8.4 - KNN results for the 2-class problem

DT (2C):

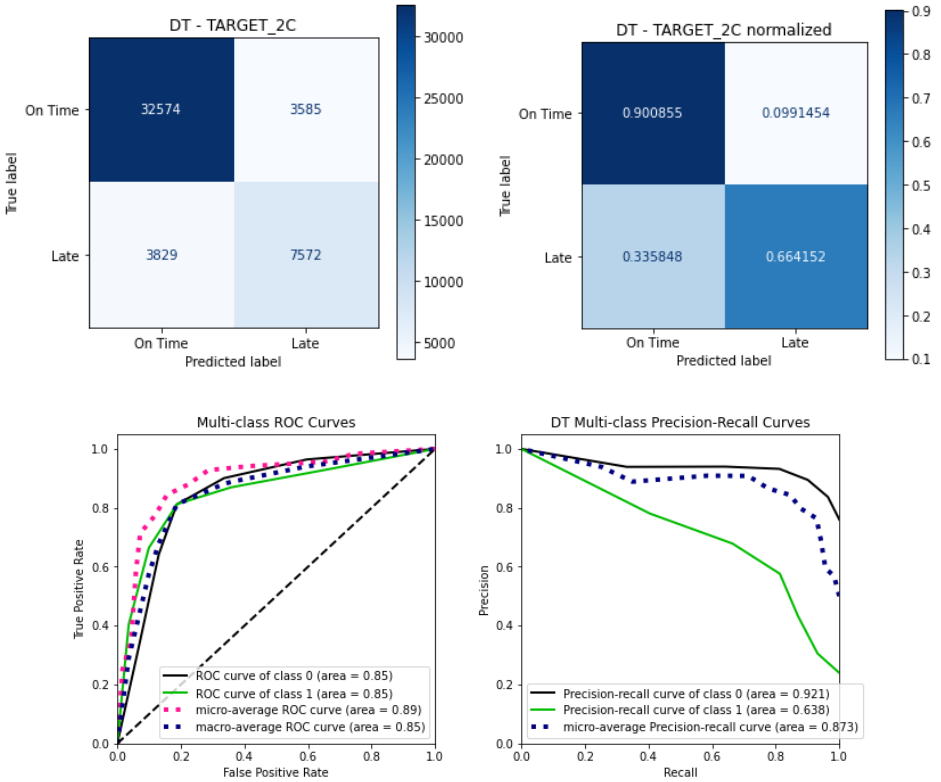


Figure 8.5 - DT results for the 2-class problem

RF (TARGET_2C):

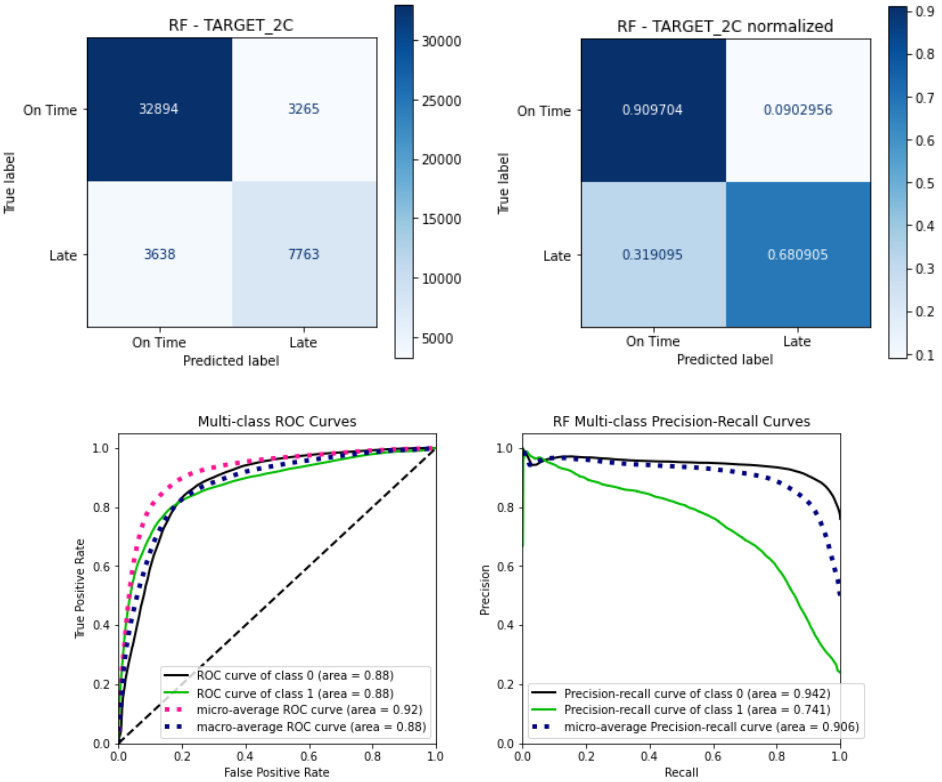


Figure 8.6 - RF results for the 2-class problem

RF with resampled dataset (TARGET_2C):

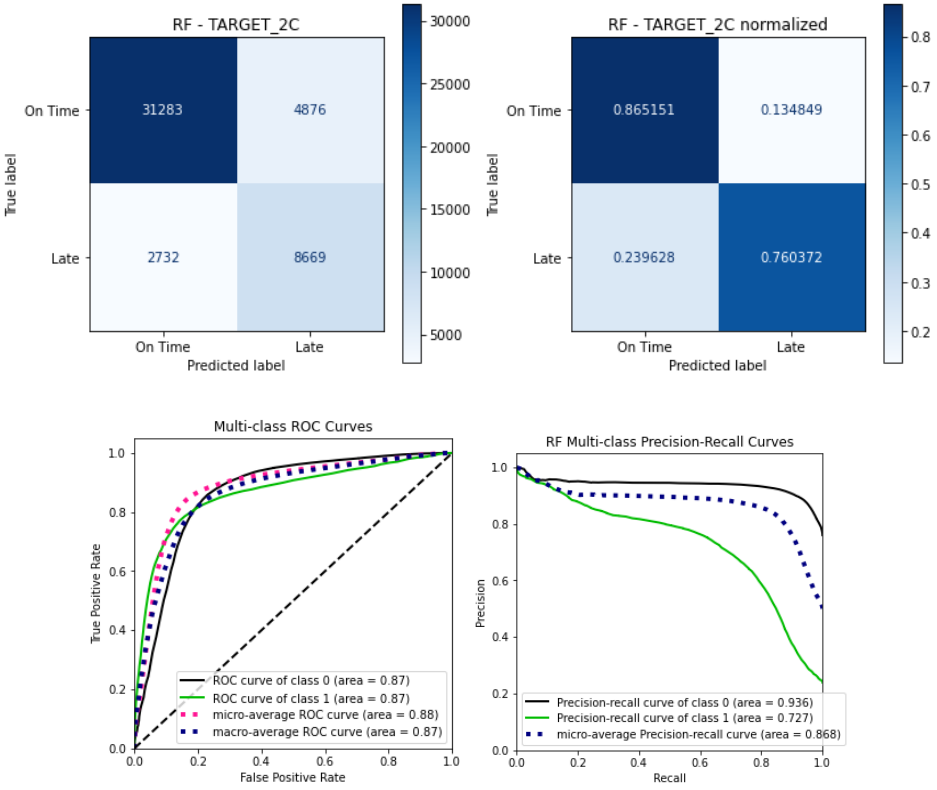


Figure 8.7 - RF with resampled dataset results for the 2-class problem

GBM (2C):

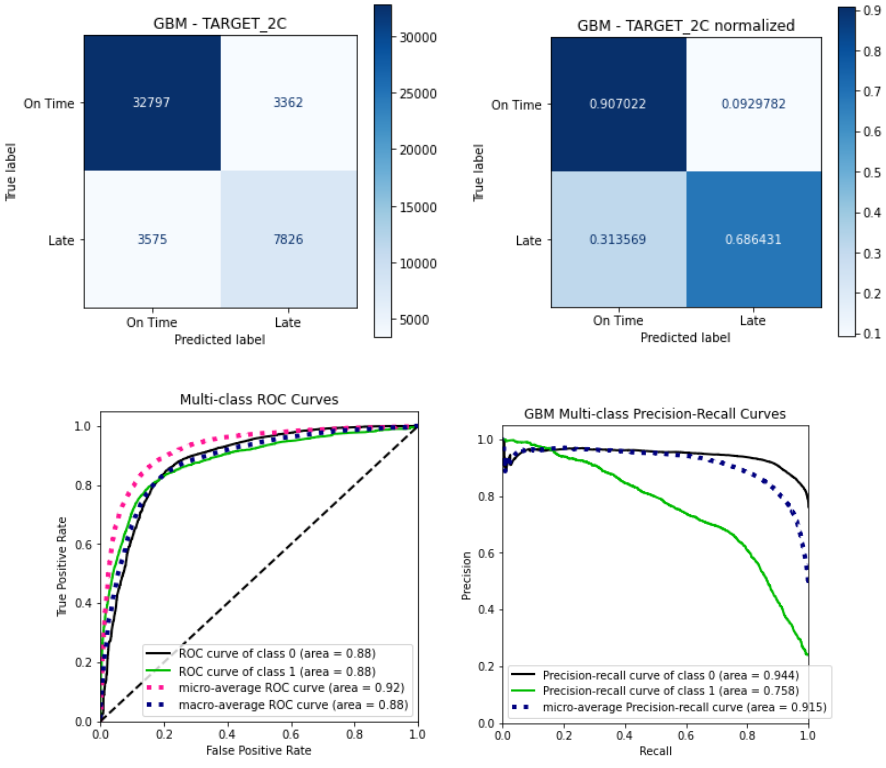


Figure 8.8 - GBM results for the 2-class problem

GBM with resampled dataset (TARGET_2C):

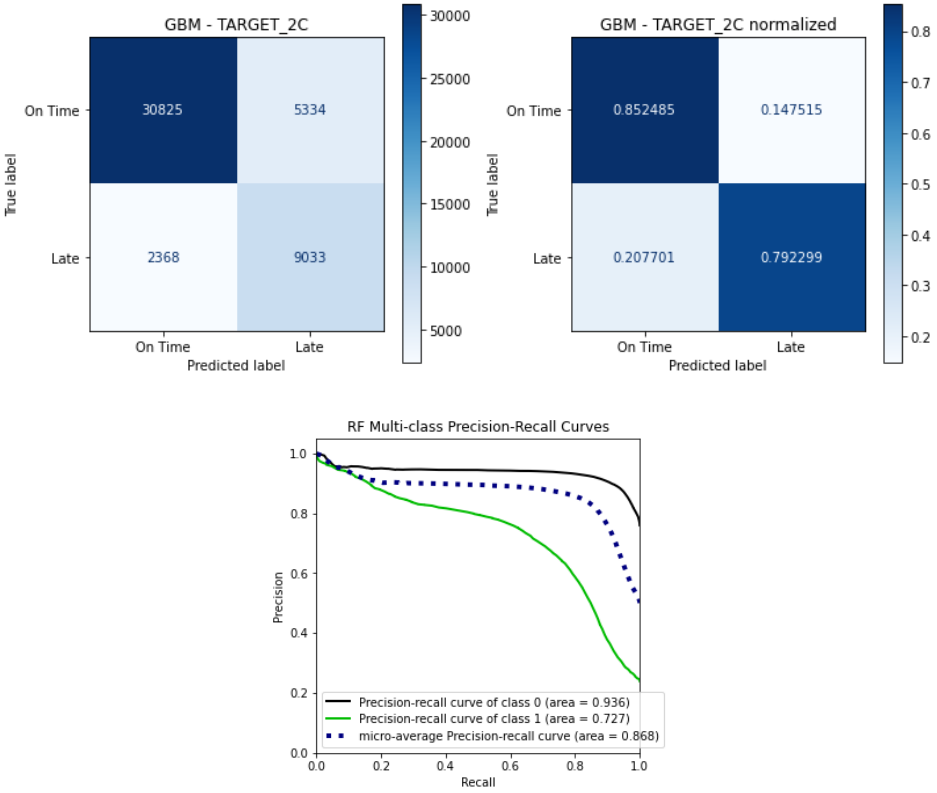


Figure 8.9 - GBM with resampled dataset results for the 2-class problem

LGBM (TARGET_2C):

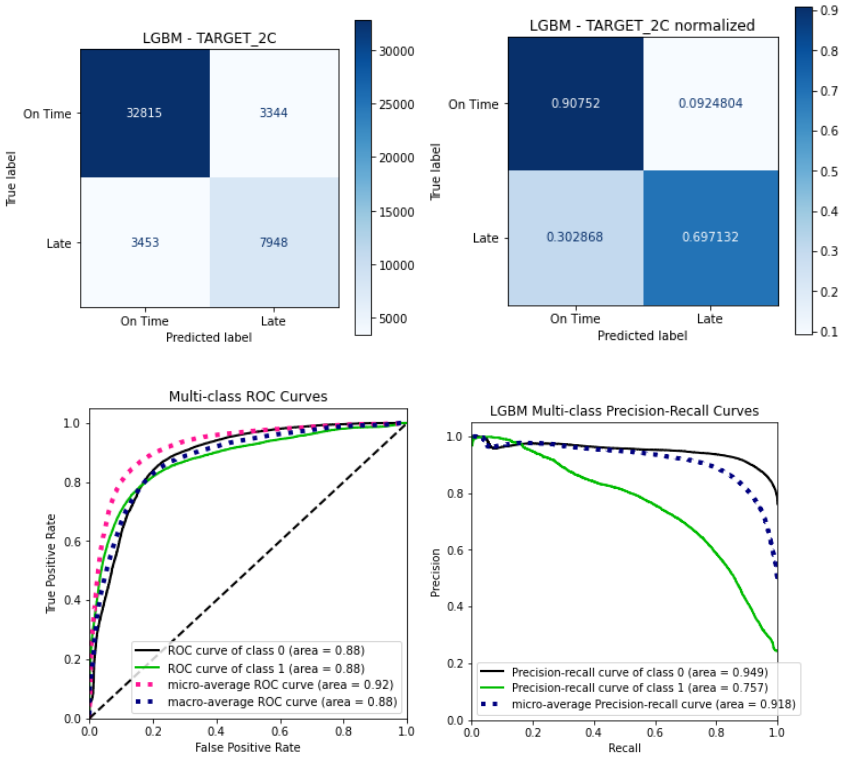


Figure 8.10 - LGBM results for 2-class problem

LGBM with resampled dataset (TARGET_2C)

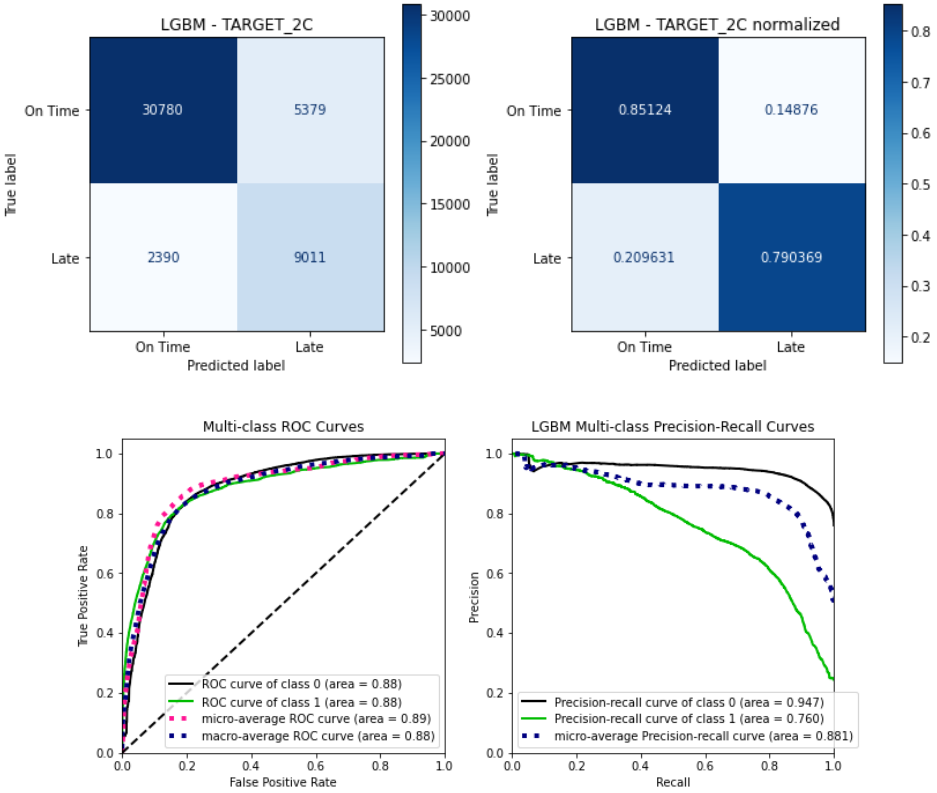


Figure 8.11 - LGBM with resampled dataset results for the 2-class problem

8.5. Results for 3-Class Problem

Dummy (Baseline) (TARGET_3C):

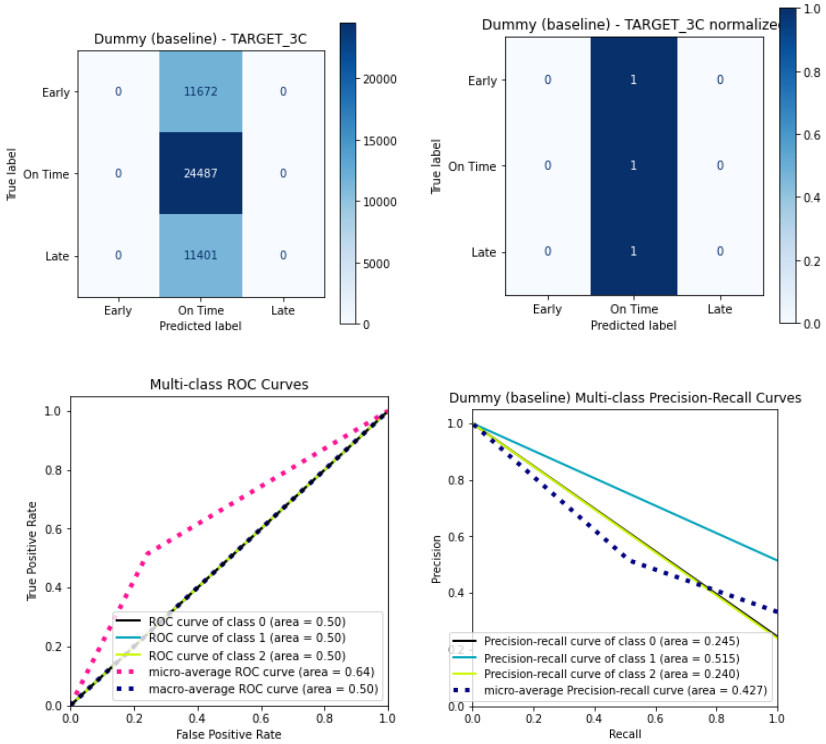


Figure 8.12 - Baseline results for the 3-class problem

LR (TARGET_3C):

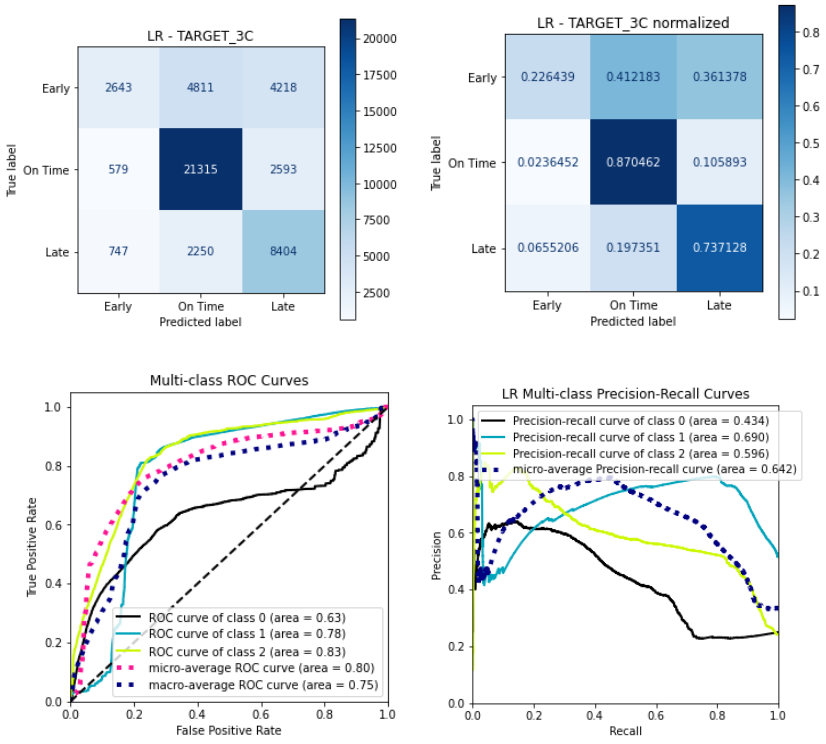


Figure 8.13 - LR results for the 3-class problem

SVM (TARGET_3C):

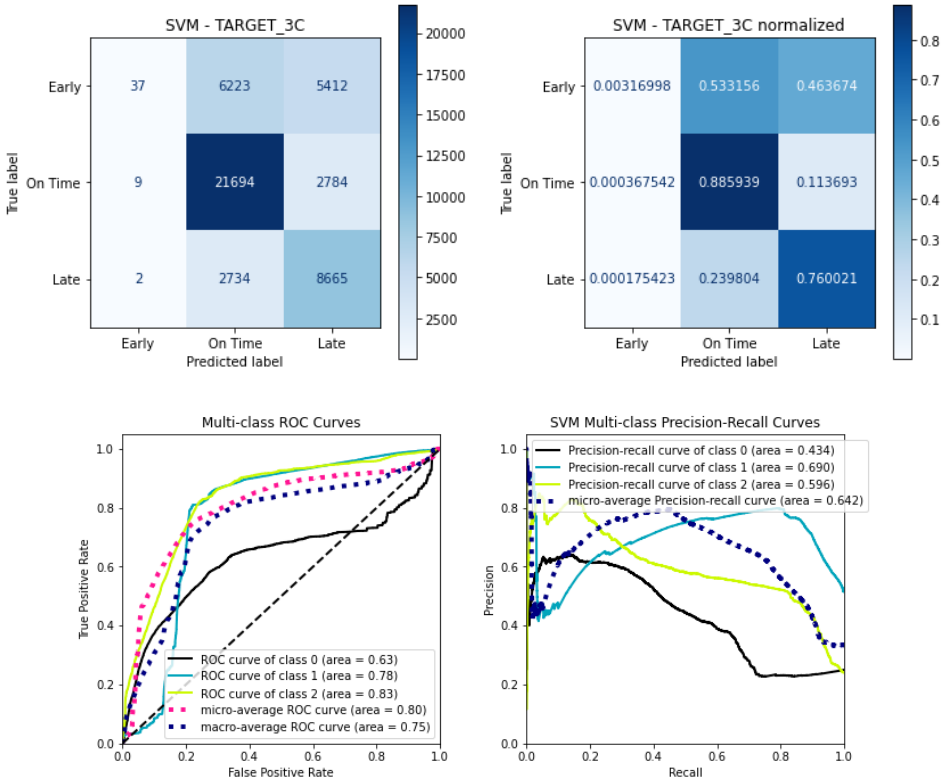


Figure 8.14 - SVM results for the 3-class problem

KNN (TARGET_3C):

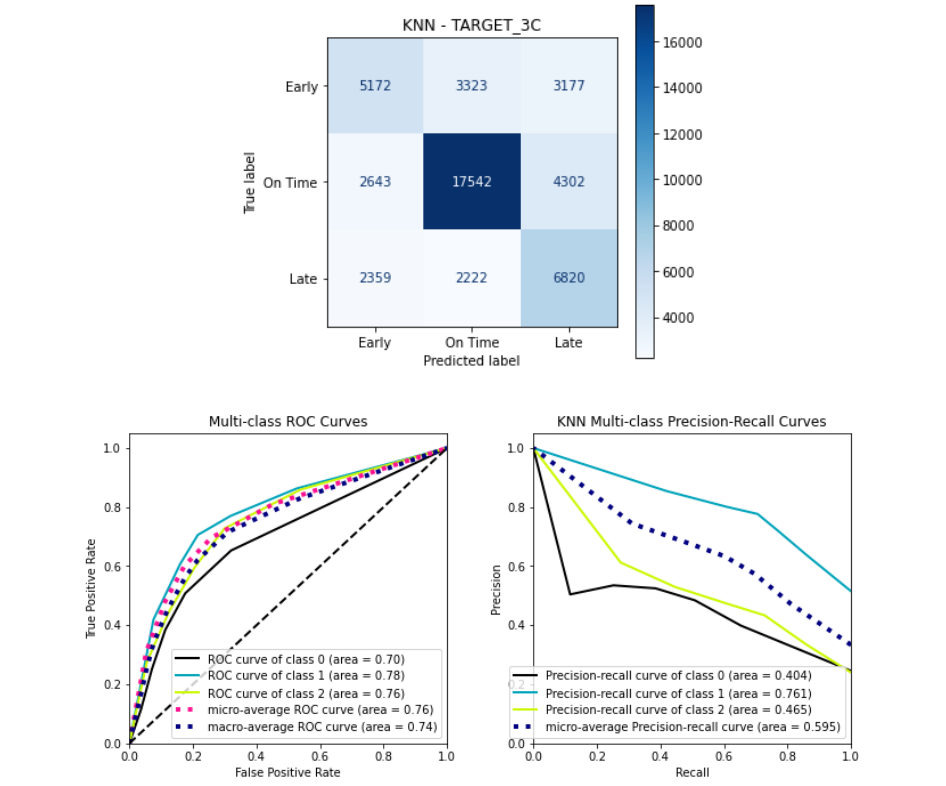


Figure 8.15 - KNN results for the 3-class problem

DT (TARGET_3C):

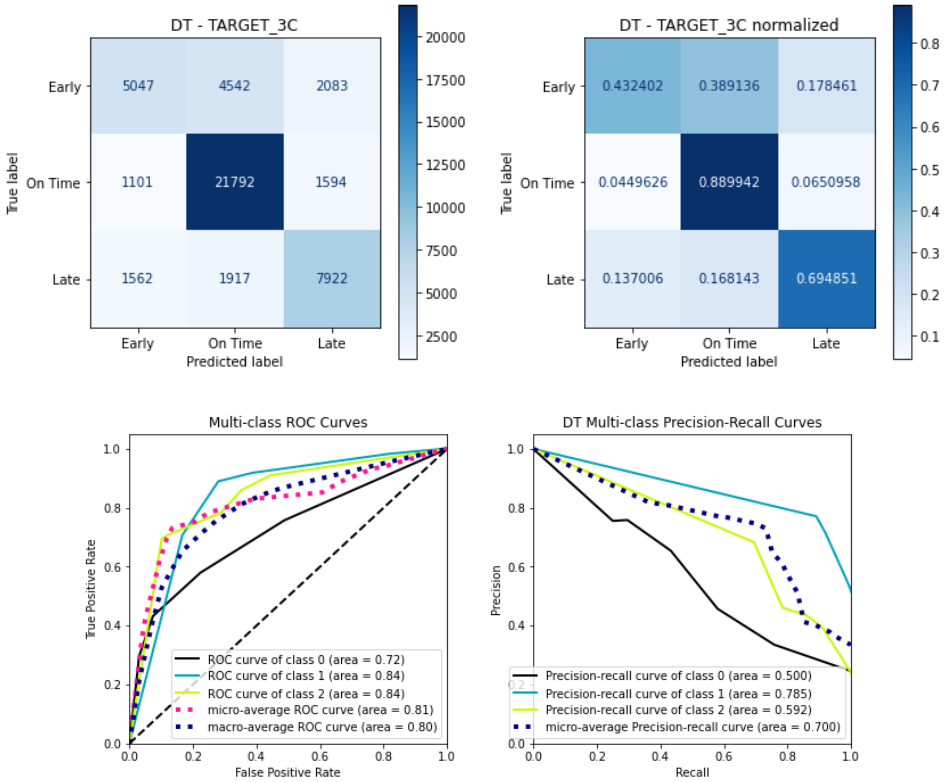


Figure 8.16 - DT results for the 3-class problem

RF (TARGET_3C)

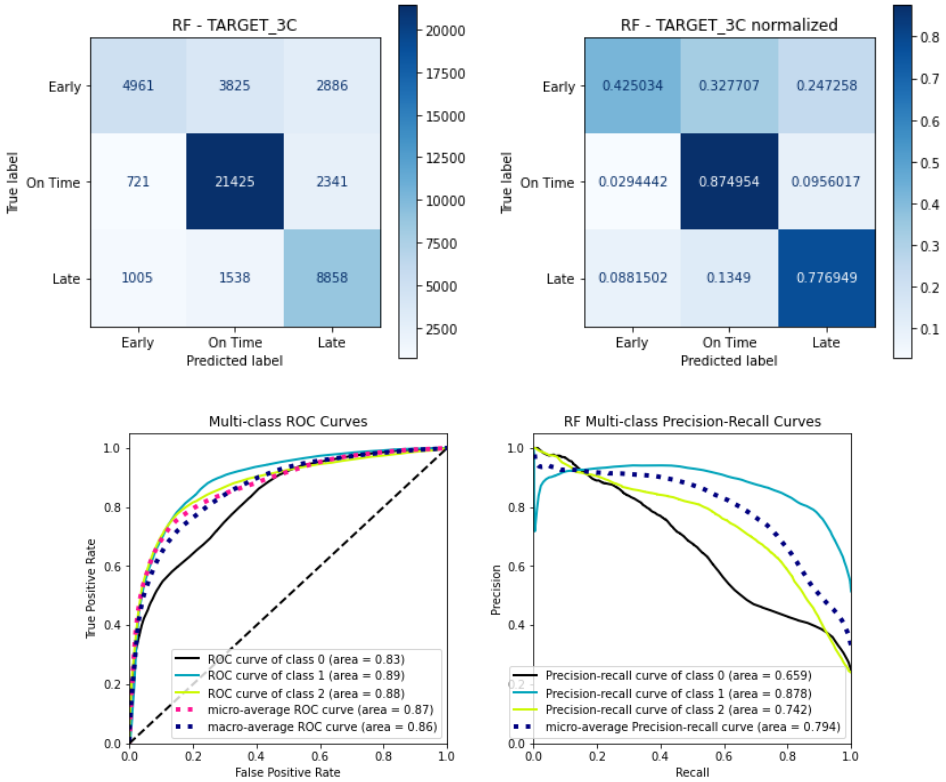


Figure 8.17 - RF results for the 3-class problem

RF with resampled dataset (TARGET_3C)

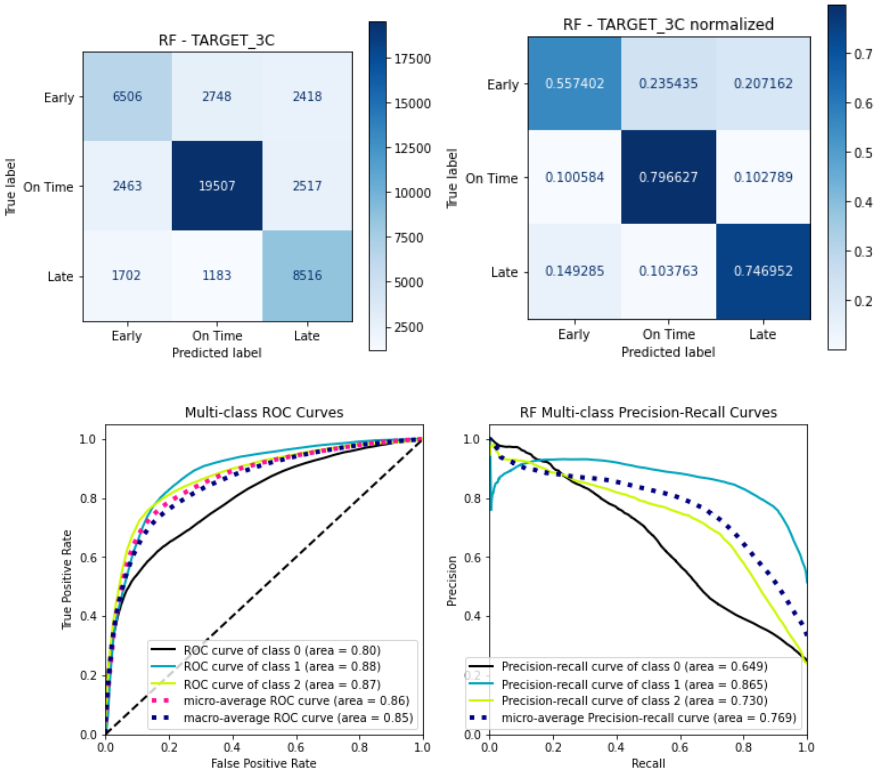


Figure 8.18 - RF with resampled dataset results for the 3-class problem

GBM (TARGET_3C)

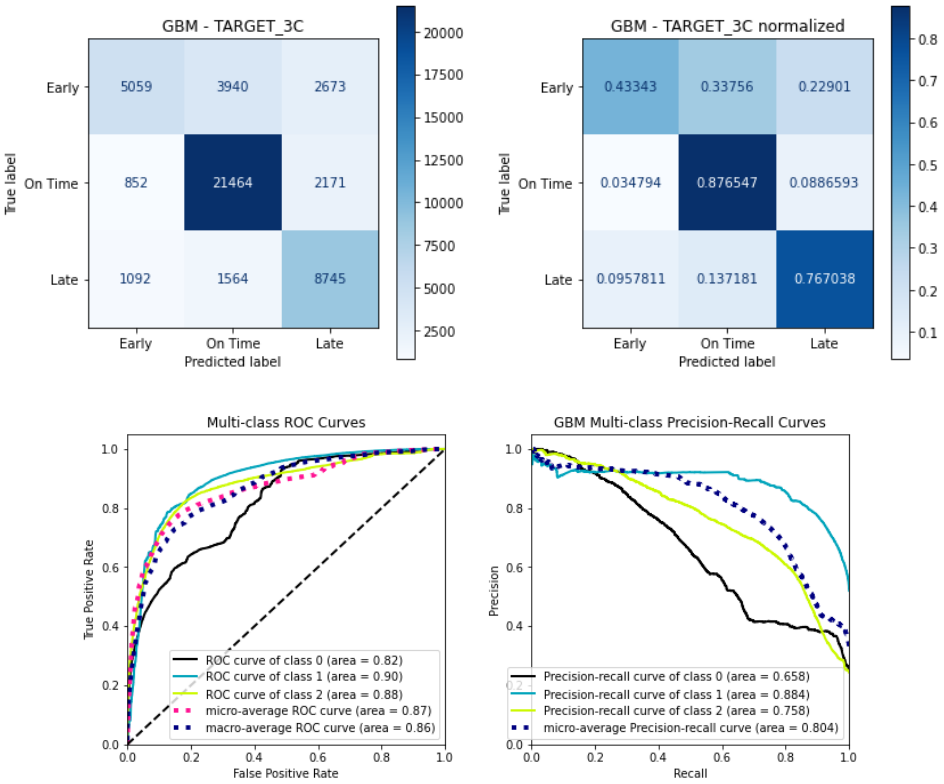


Figure 8.19 - GBM results for the 3-class problem

GBM with resamped dataset (TARGET_3C)

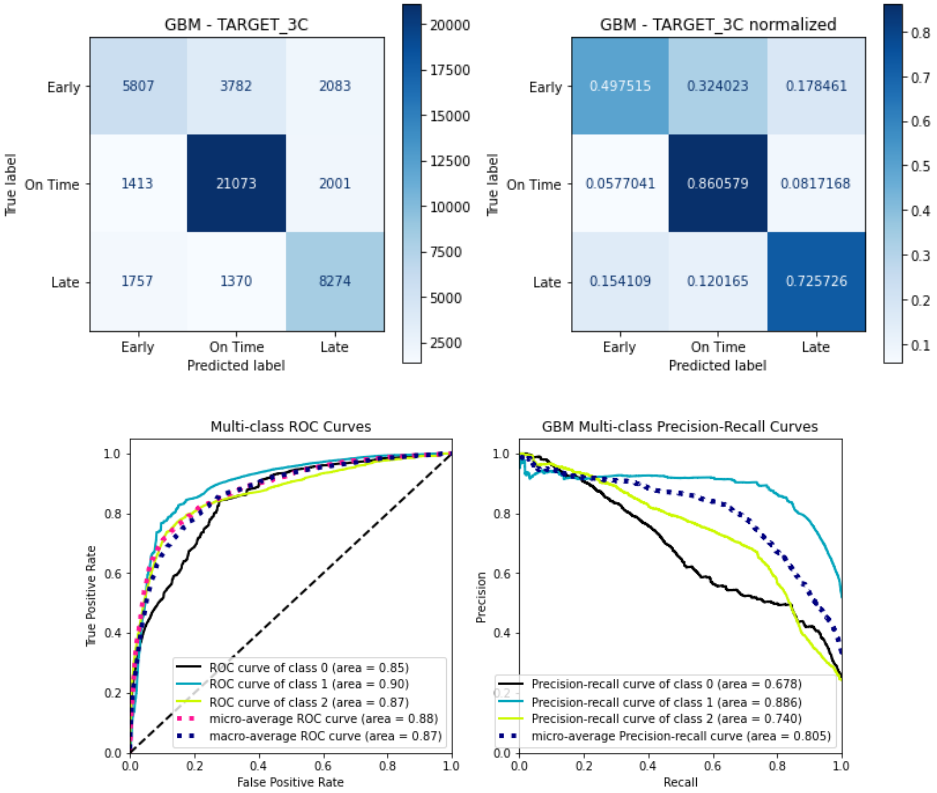


Figure 8.20 - GBM with resampled dataset results for the 3-class problem

LGBM (TARGET_3C)

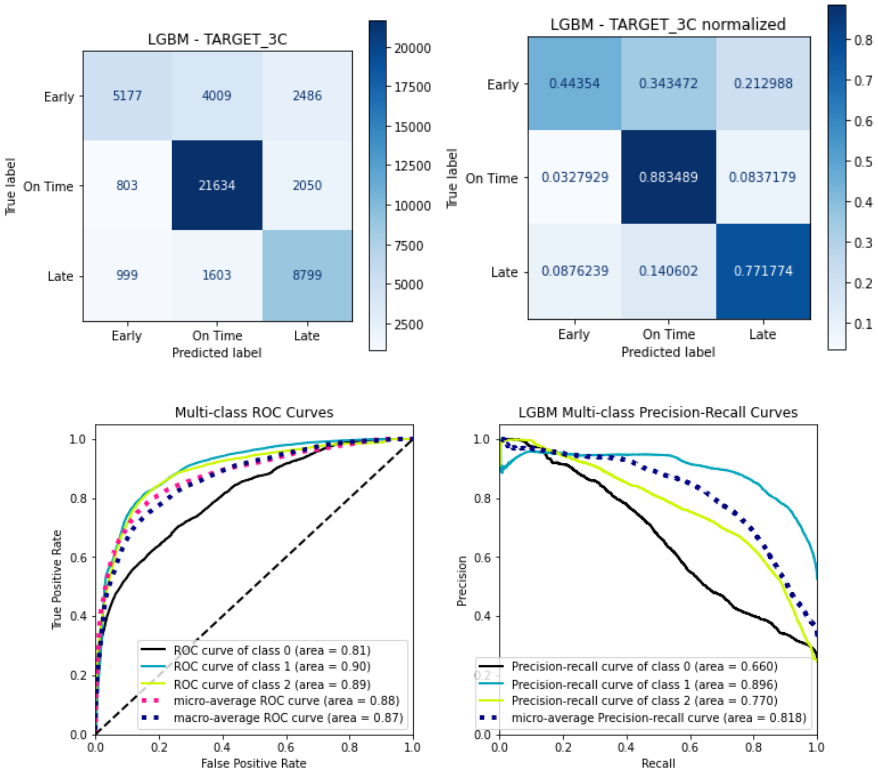


Figure 8.21 - LGBM for the 3-class problem

LGBM with resampled dataset (TARGET_3C)

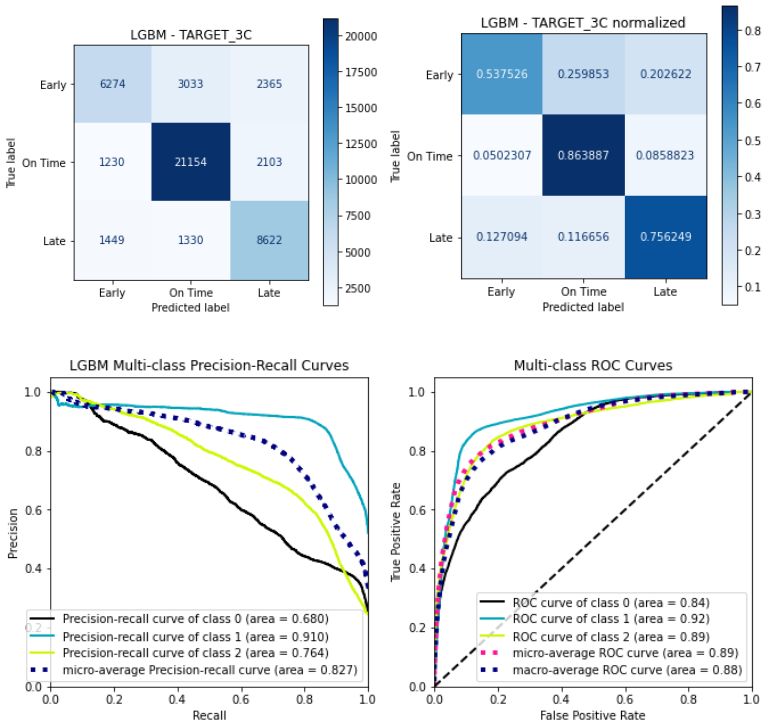


Figure 8.22 - LGBM with resampled dataset results for the 3-class problem

8.6. Results for 5-Class Problem

Baseline (TARGET_5C)

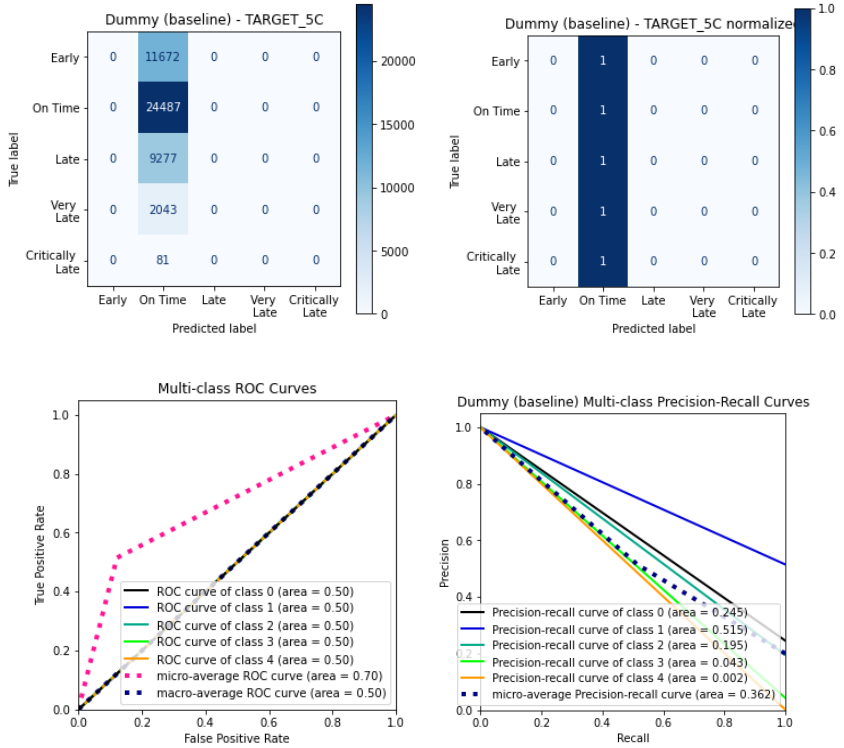


Figure 8.23 - Baseline results for the 5-class problem

LR (TARGET_5C)

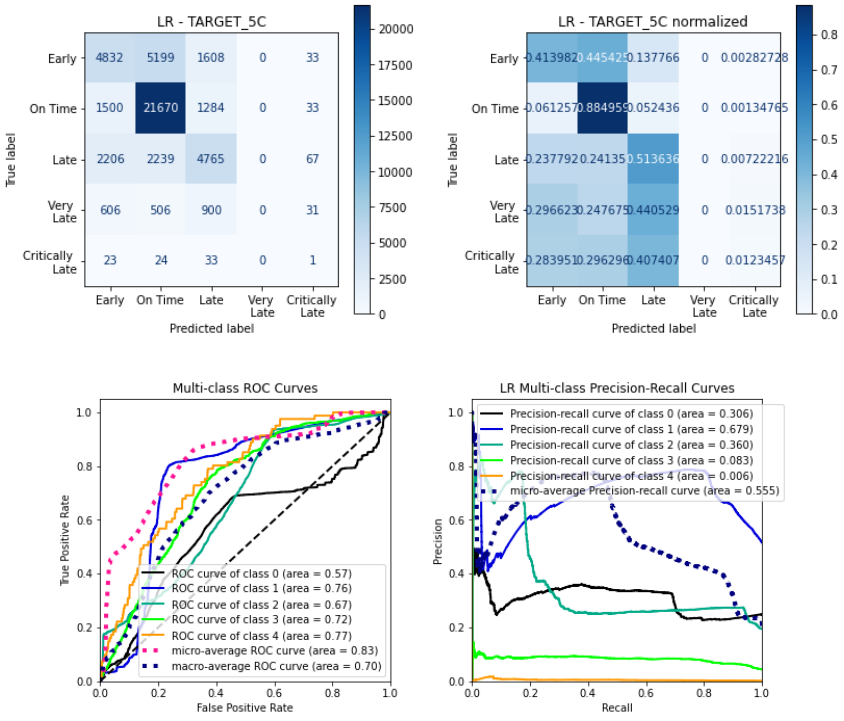


Figure 8.24 - LR results for the 5-class problem

SVM (TARGET_5C)

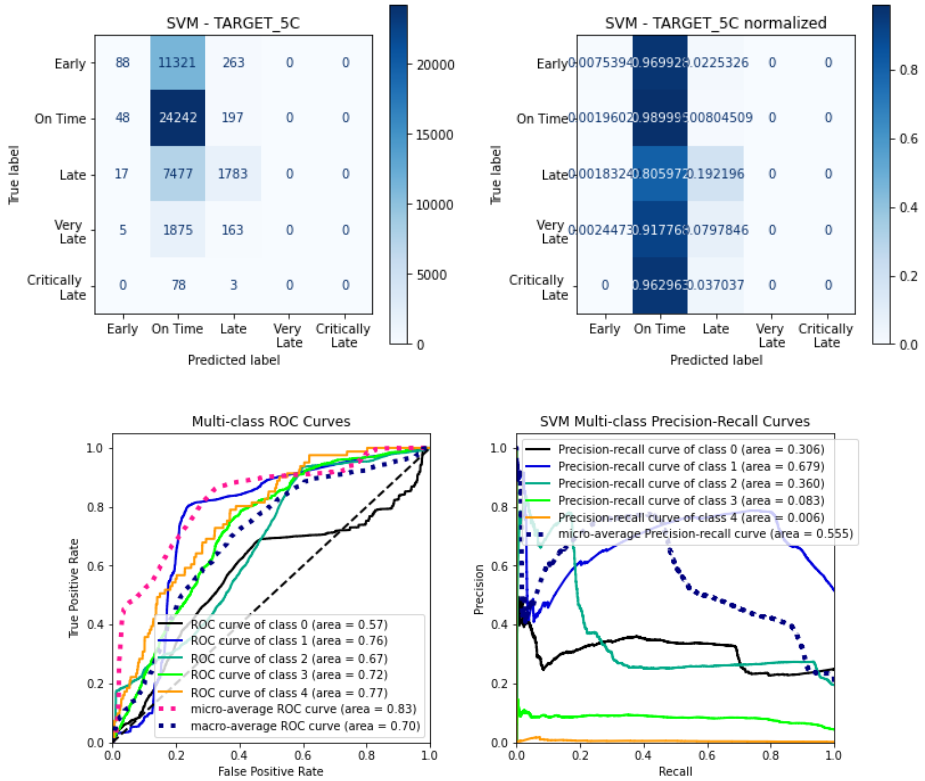


Figure 8.25 - SVM results for the 5-class problem

KNN (TARGET_5C)

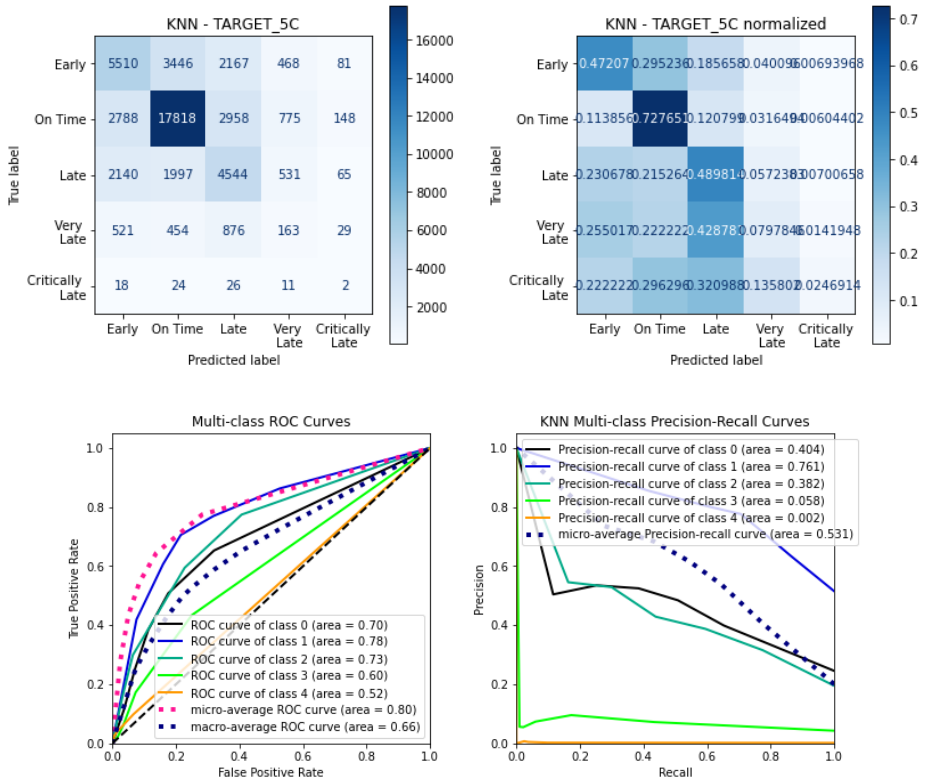


Figure 8.26 - KNN results for the 5-class problem

DT (TARGET_5C)

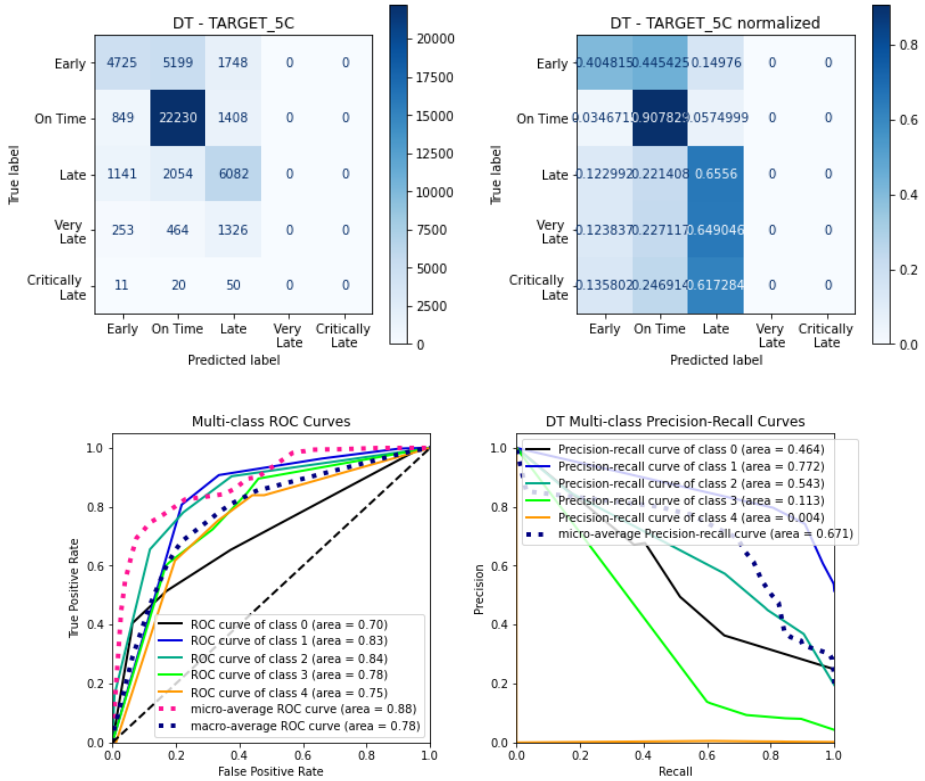


Figure 8.27 - DT results for the 5-class problem

RF (TARGET_5C)

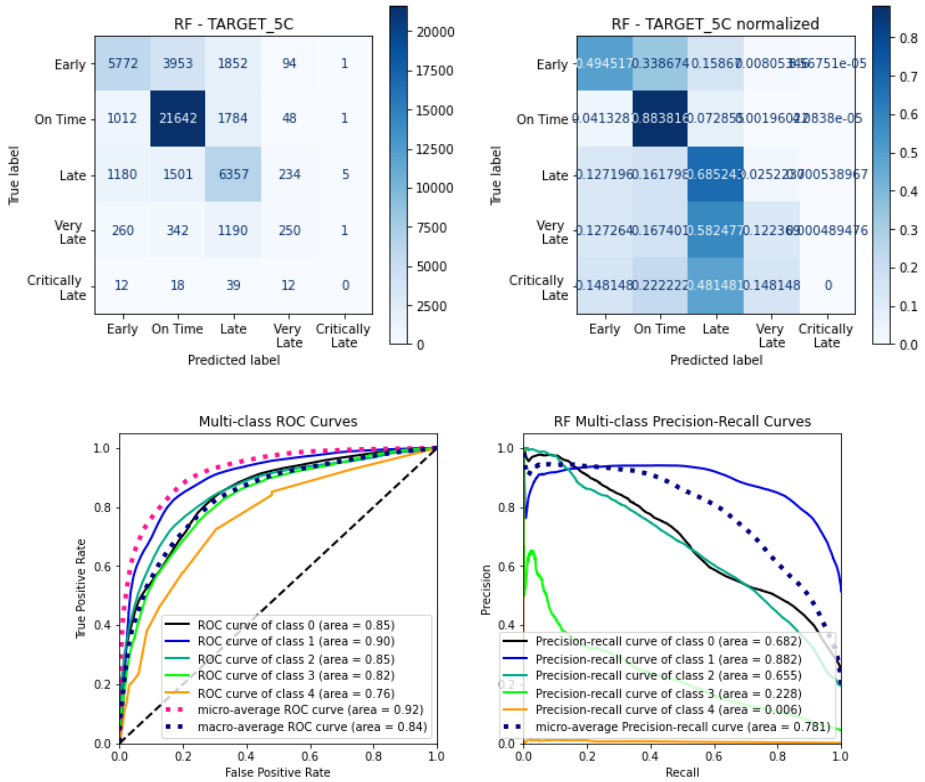


Figure 8.28 - RF results for the 5-class problem

RF with resampled dataset (TARGET_5C)

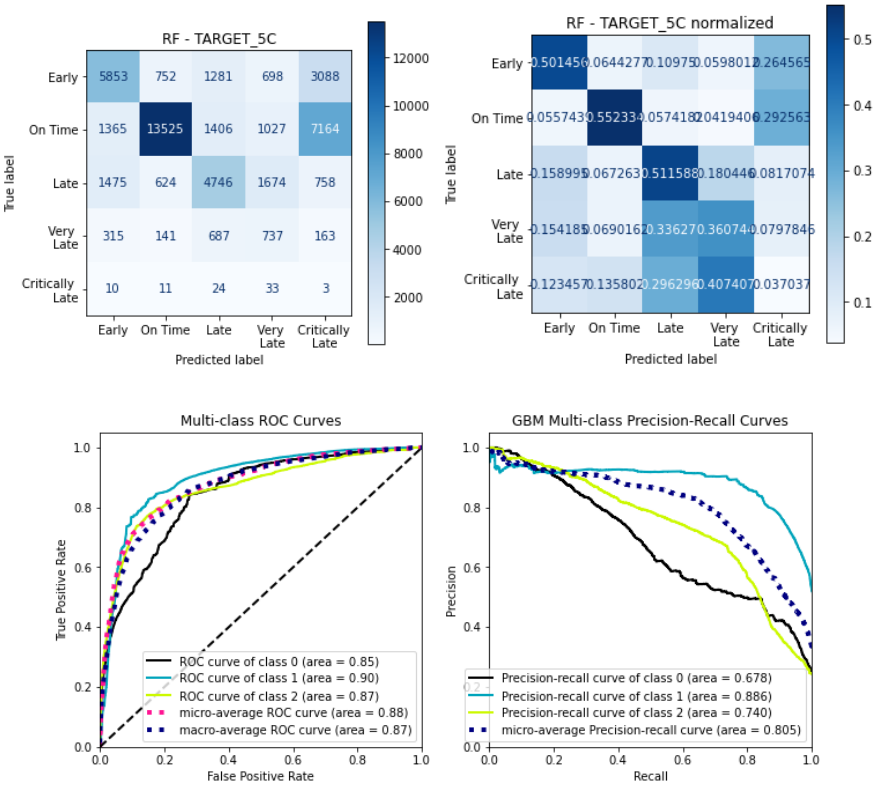


Figure 8.29 - RF with resampled dataset results for the 5-class problem

GBM (TARGET_5C)

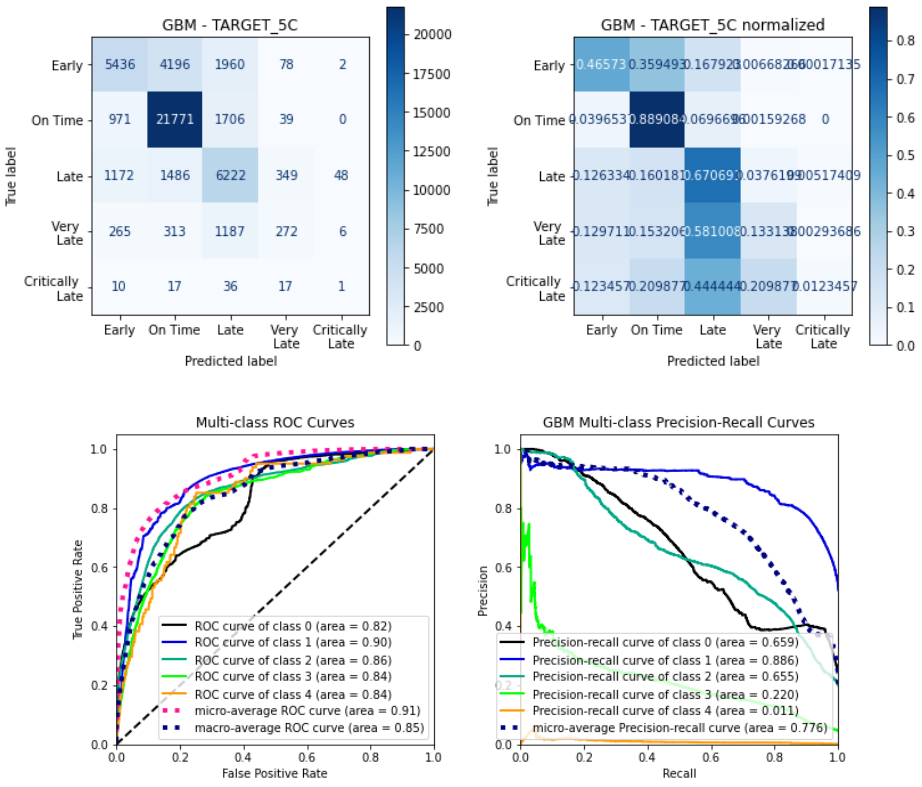


Figure 8.30 - GBM results for the 5-class problem

GBM with resampled dataset (TARGET_5C)

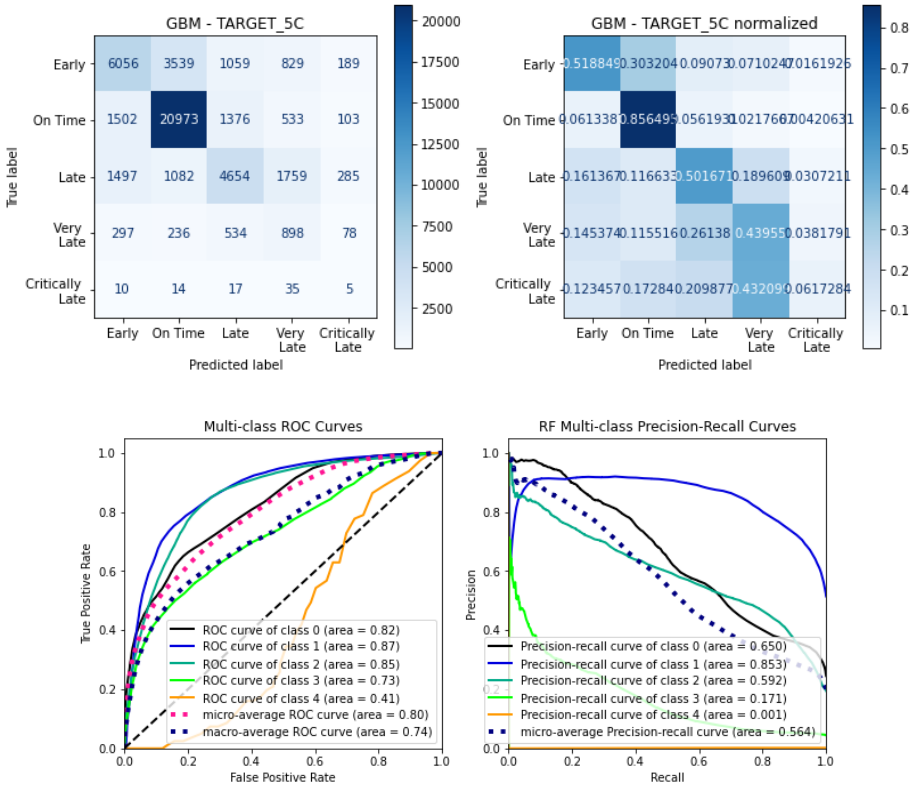


Figure 8.31 - GBM with resampled dataset results for the 5-class problem

LGBM (TARGET_5C)

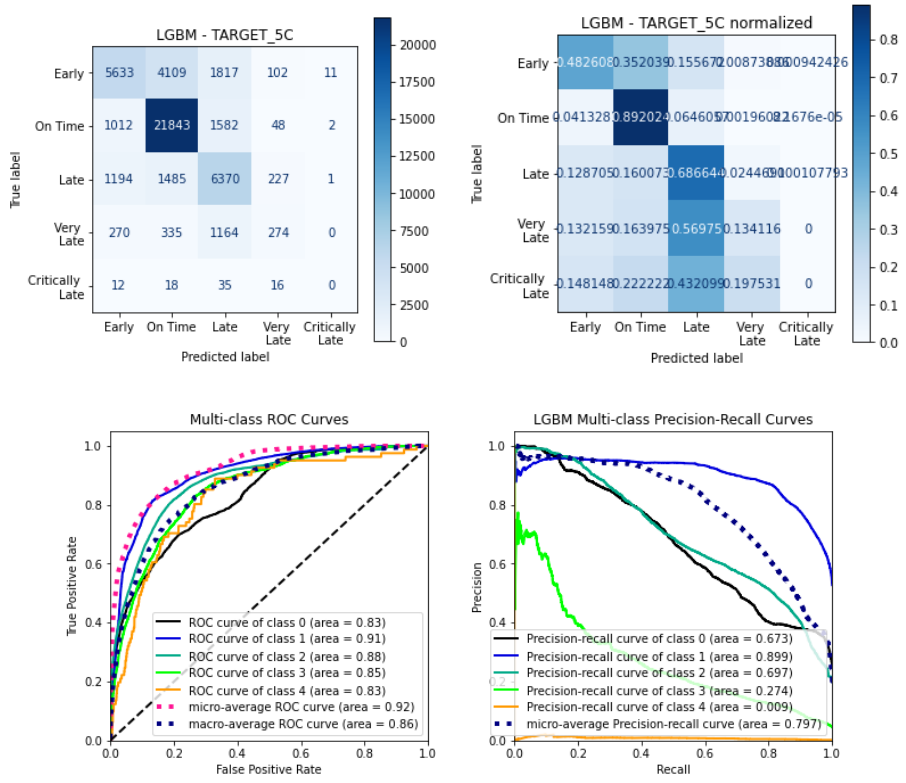


Figure 8.32 - LGBM results for the 5-class problem

LGBM with resampled dataset (TARGET_5C)

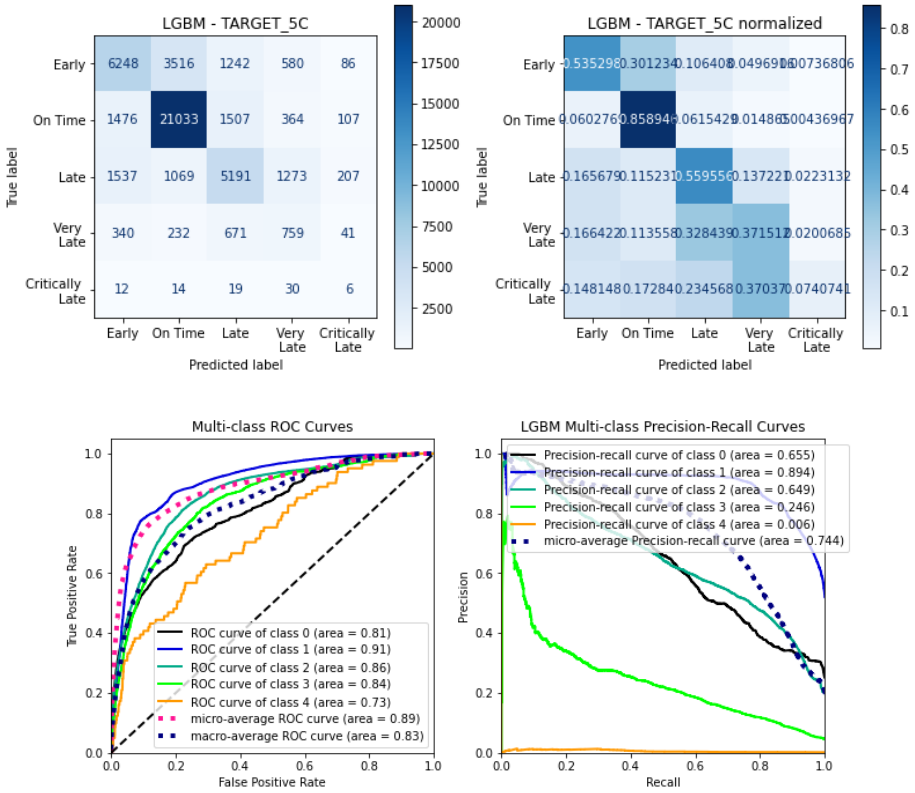


Figure 8.33 - LGBM with resampled dataset results for the 5-class problem

8.7. FEATURE IMPORTANCES:

LR (TARGET_2C)

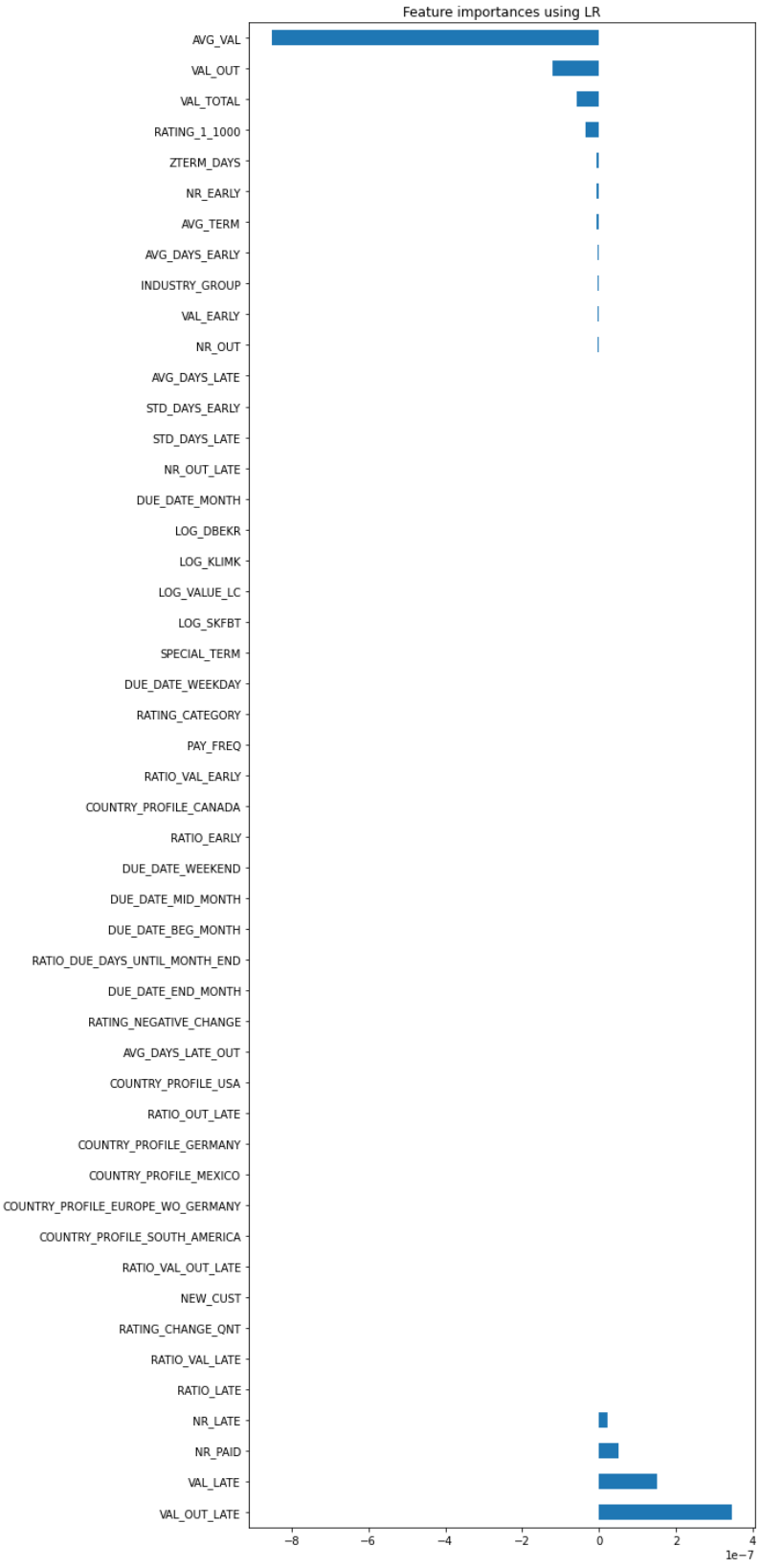


Figure 8.34 - Feature importance using LR for the 2-class problem

SVM (TARGET_2C)

Figure 8.35 - Feature importance using SVM for the 2-class problem

DT (TARGET_2C)

Figure 8.36 - Feature importance using DT for the 2-class problem

RF (TARGET_2C)

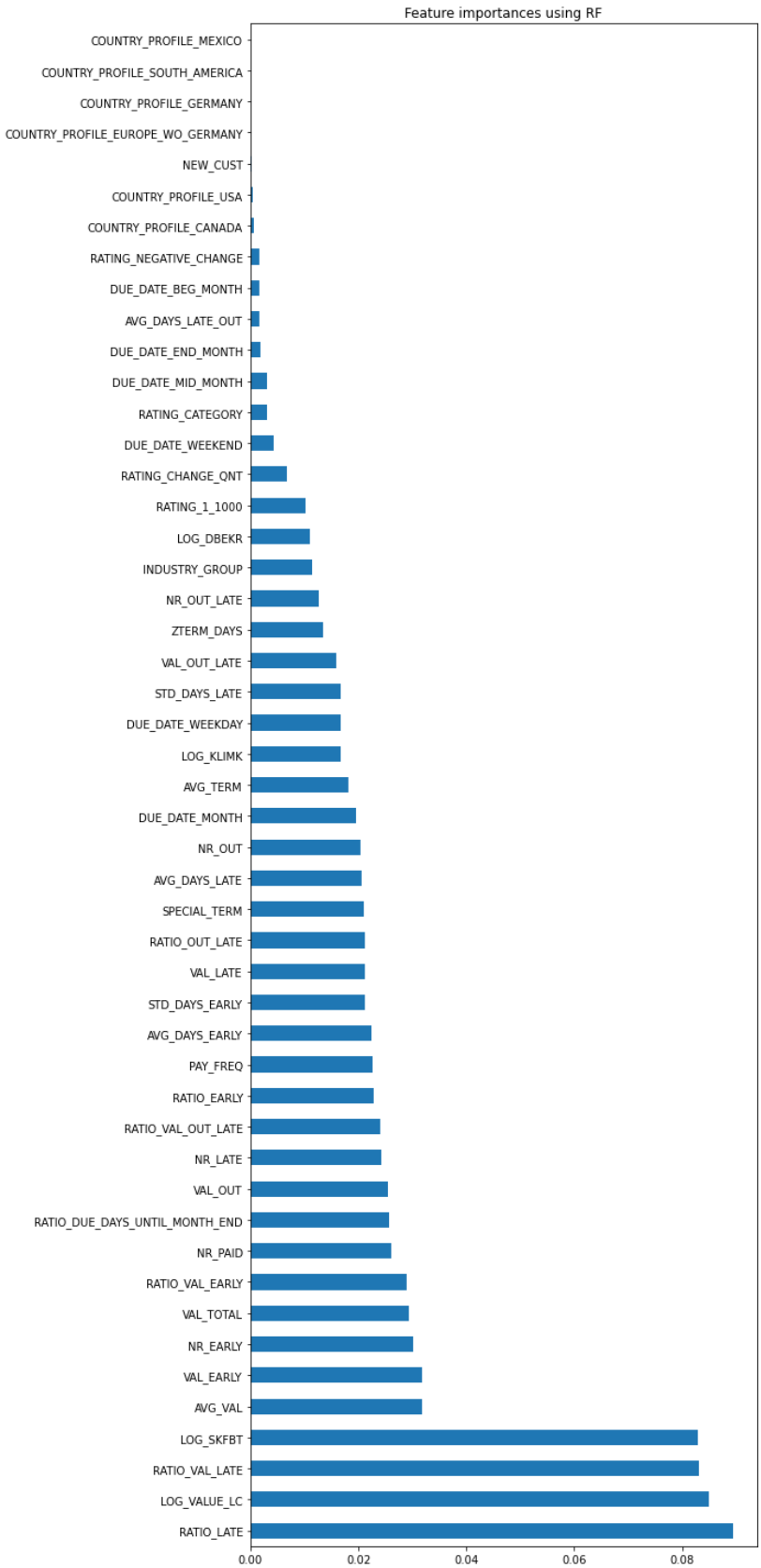


Figure 8.37 - Feature importance using RF for the 2-class problem

RF with resampled dataset (TARGET_2C)

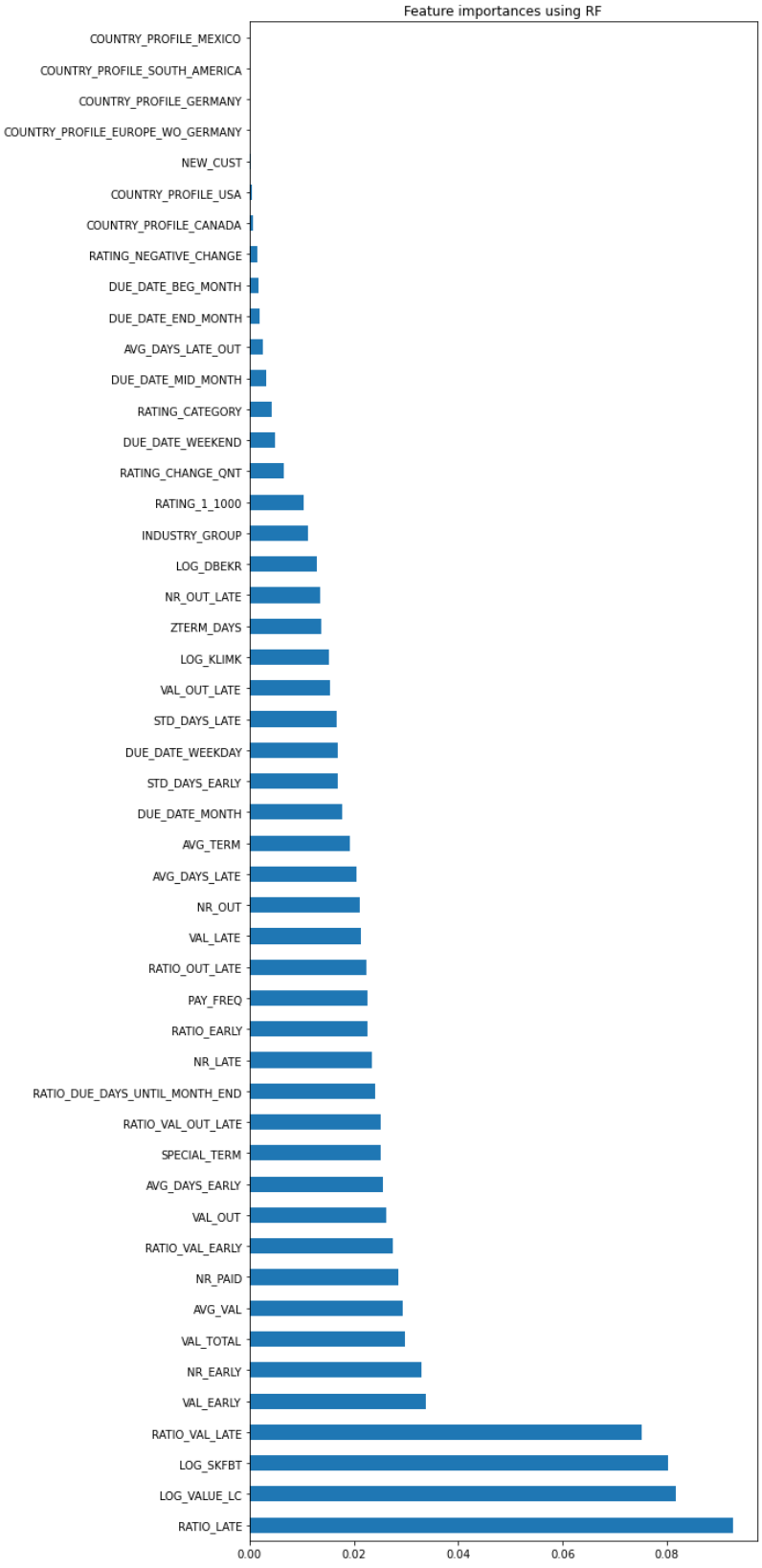


Figure 8.38 - Feature importance using RF with a resampled dataset for the 2-class problem

GBM (TARGET_2C)

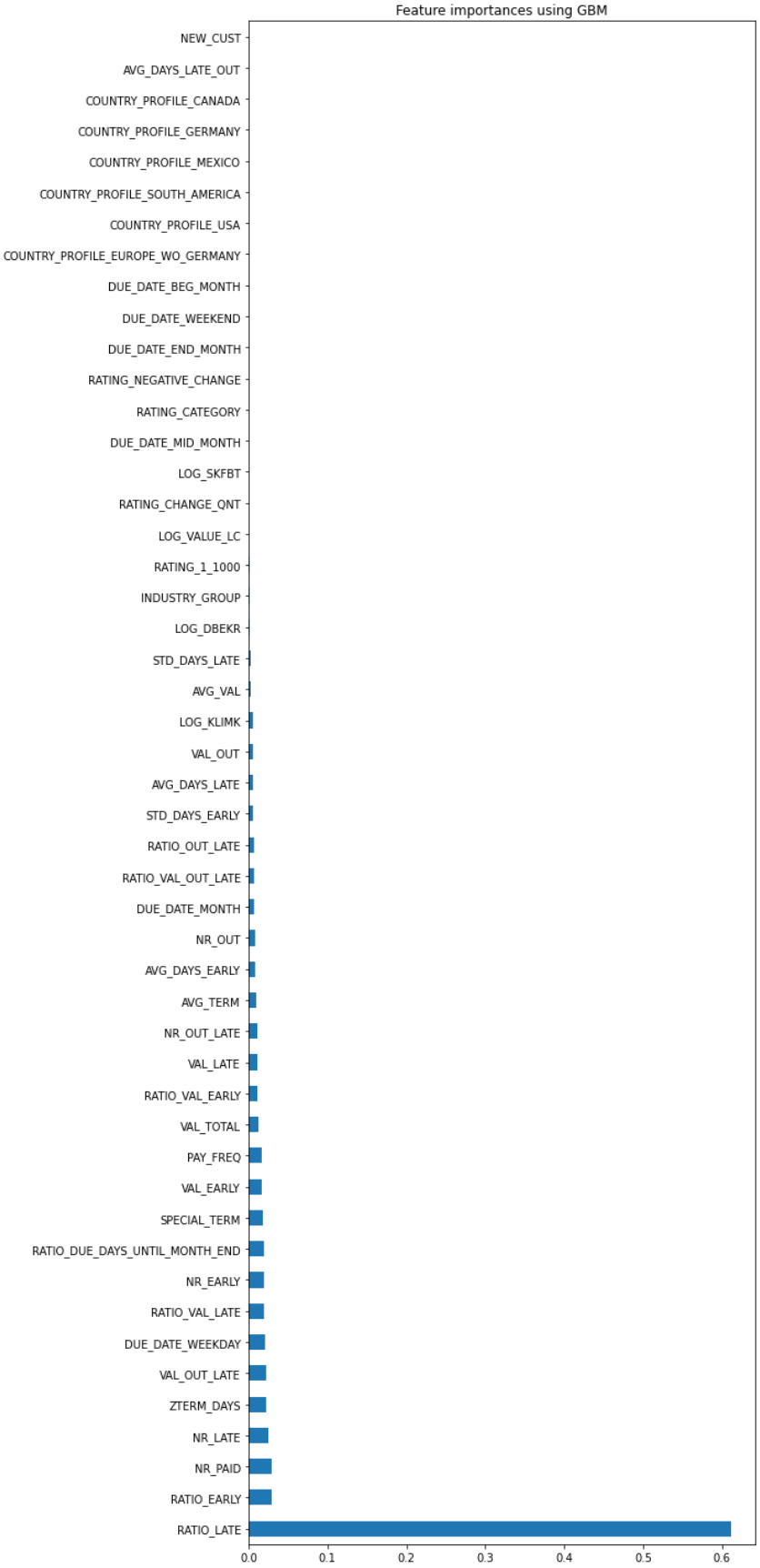


Figure 8.39 - Feature importance using GBM for the 2-class problem

GBM with resampled dataset (TARGET_2C)

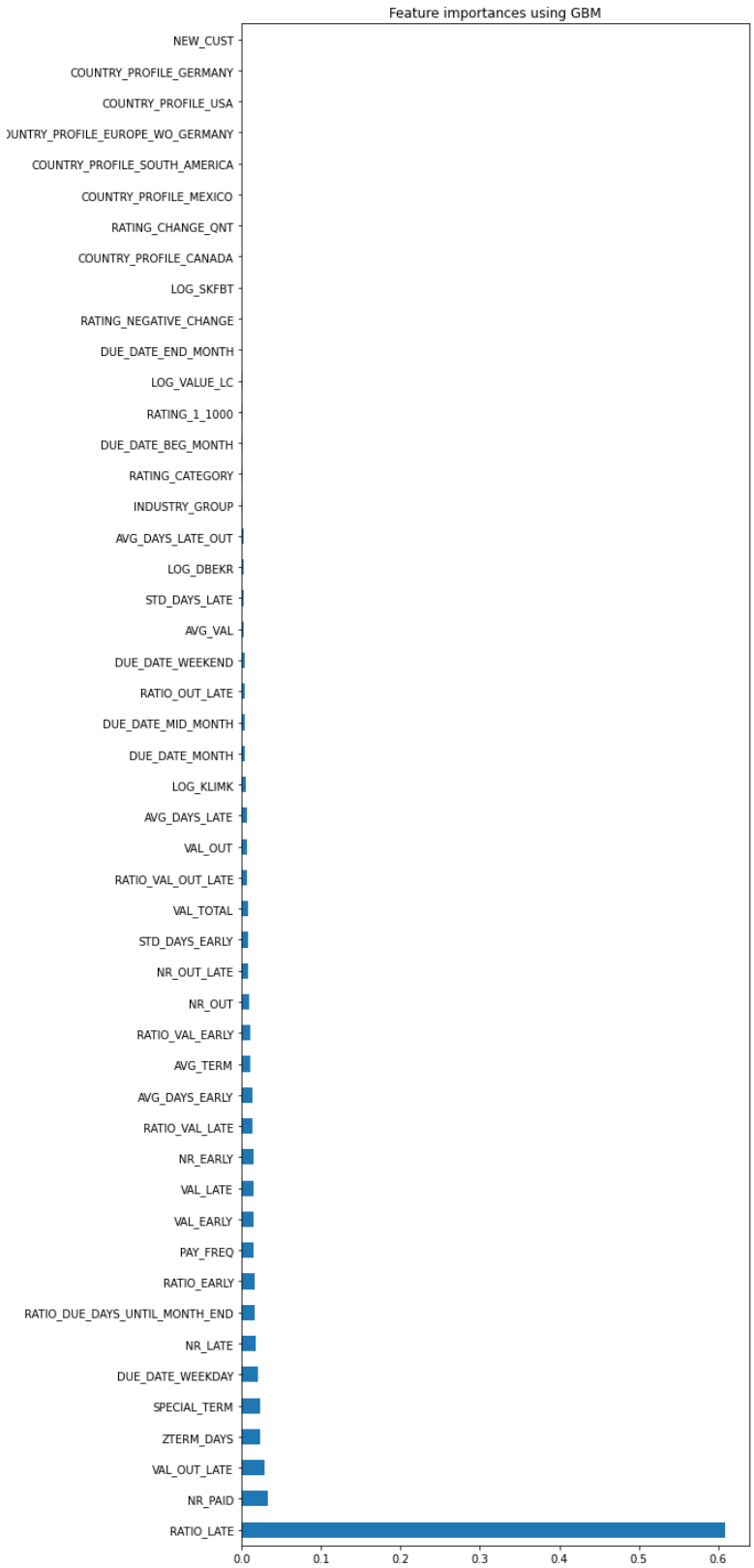


Figure 8.40 - Feature importance using GBM with a resampled dataset for the 2-class problem

LGBM (TARGET_2C)

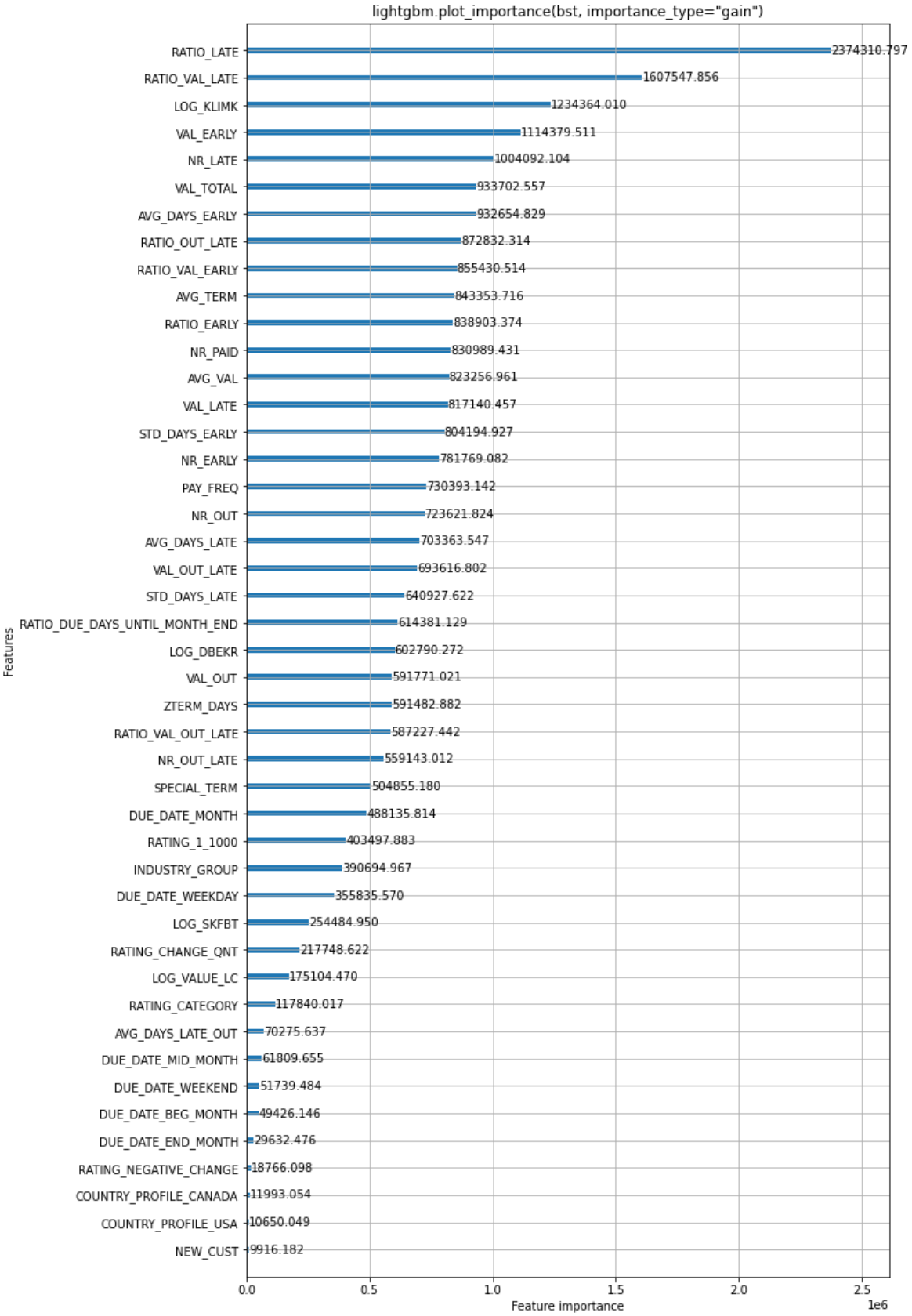


Figure 8.41 - Feature importance by gain using LGBM for the 2-class problem

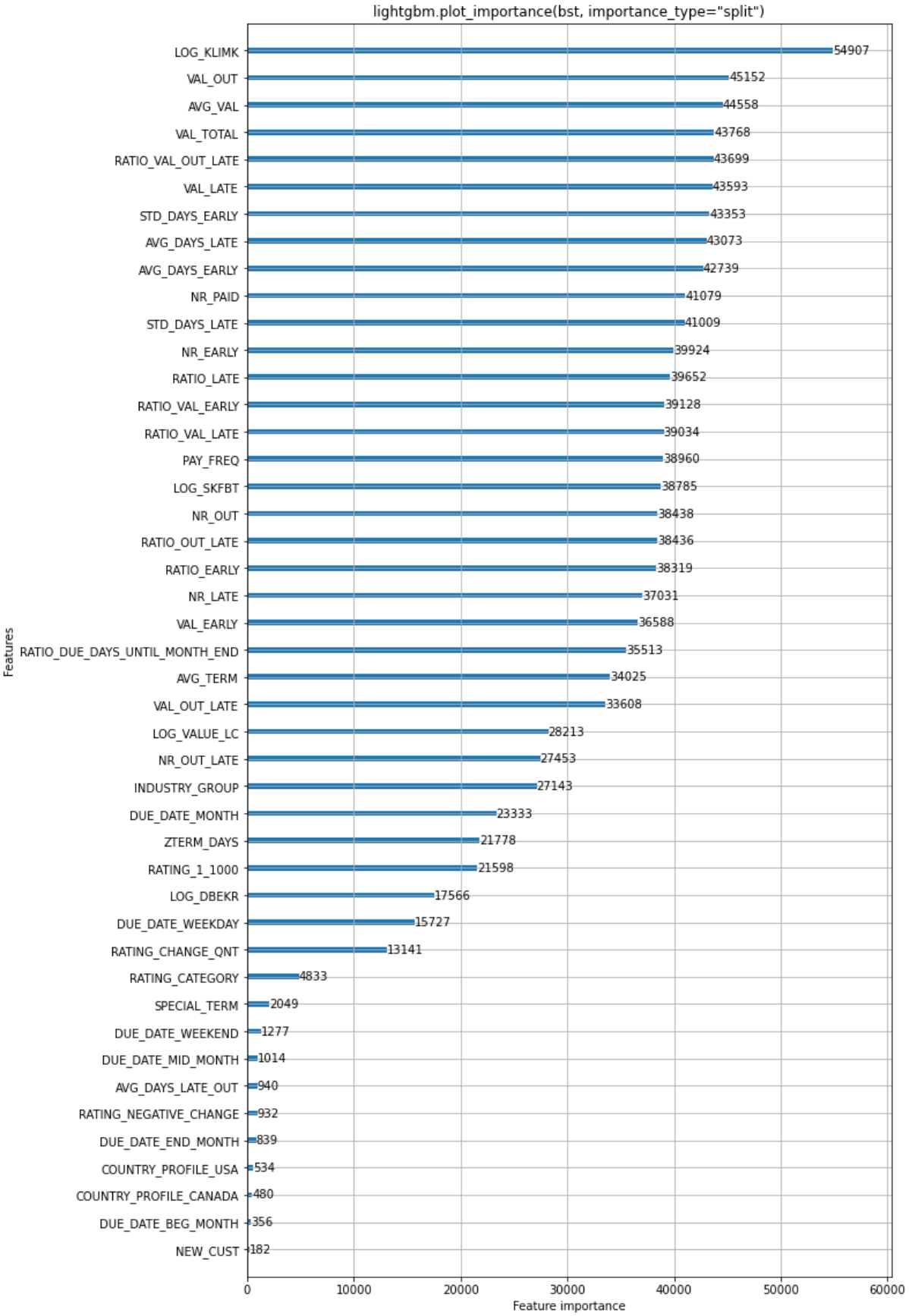


Figure 8.42 - Feature importance by split using LGBM for the 2-class problem

LGBM with resampled dataset (TARGET_2C)

Figure 8.43 - Feature importance by gain using LGBM with a resampled dataset for the 2-class problem

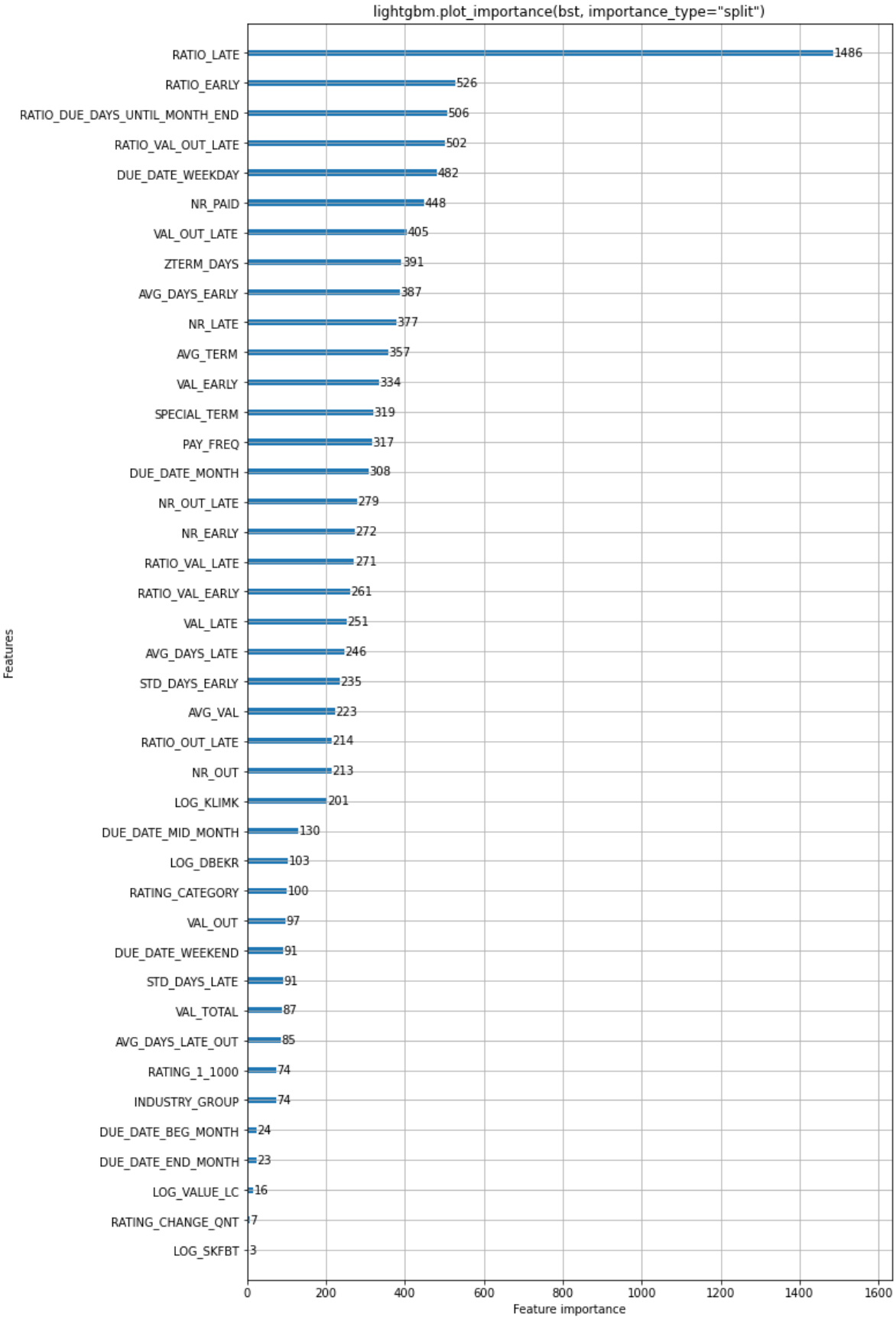


Figure 8.44 - Feature importance by split using LGBM with a resampled dataset for the 2-class problem

LR (TARGET_3C)

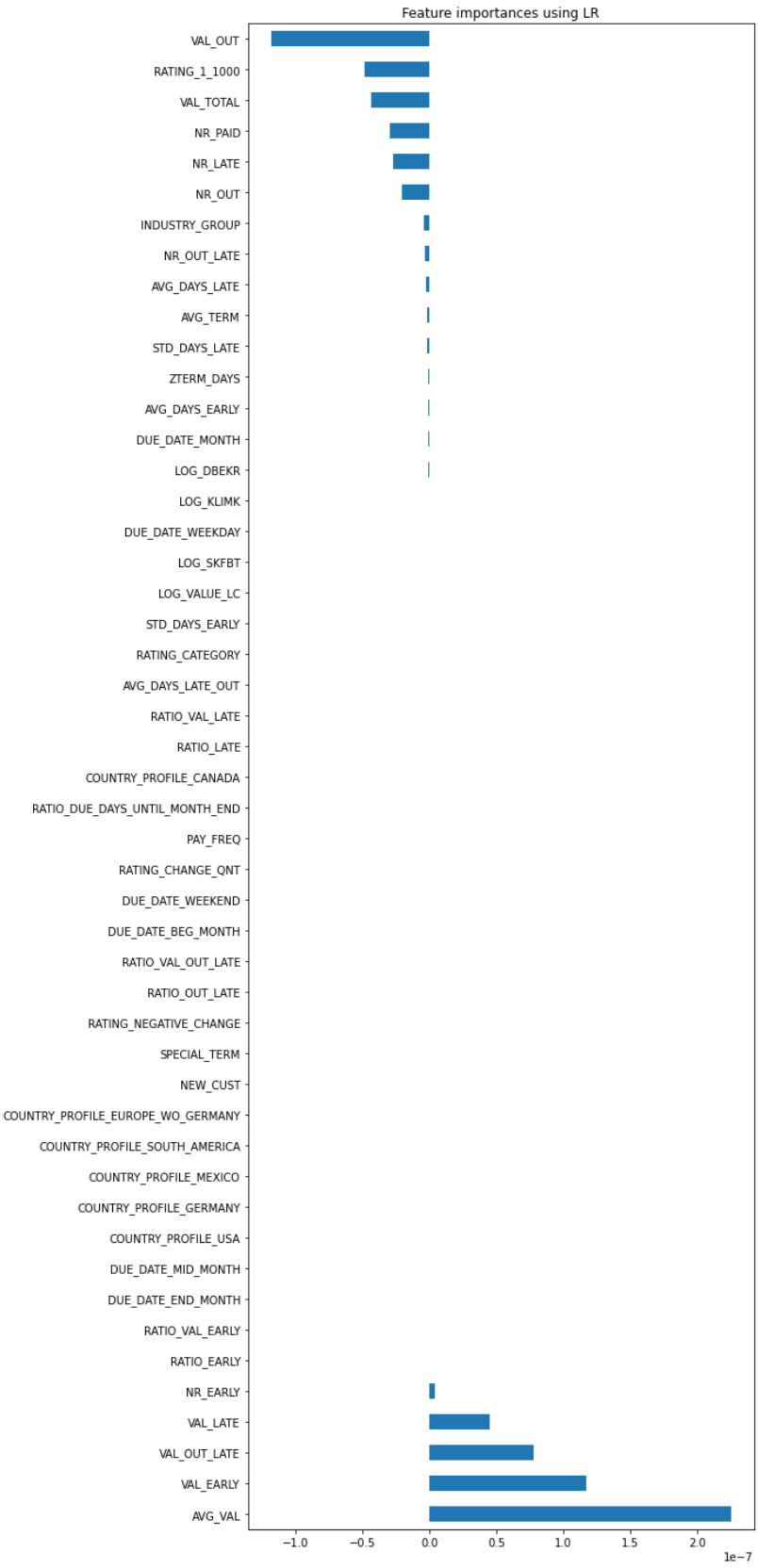


Figure 8.45 - Feature importance using LR for the 3-class problem

SVM (TARGET_3C)

Figure 8.46 - Feature importance using SVM for the 3-class problem

RF (TARGET_3C)

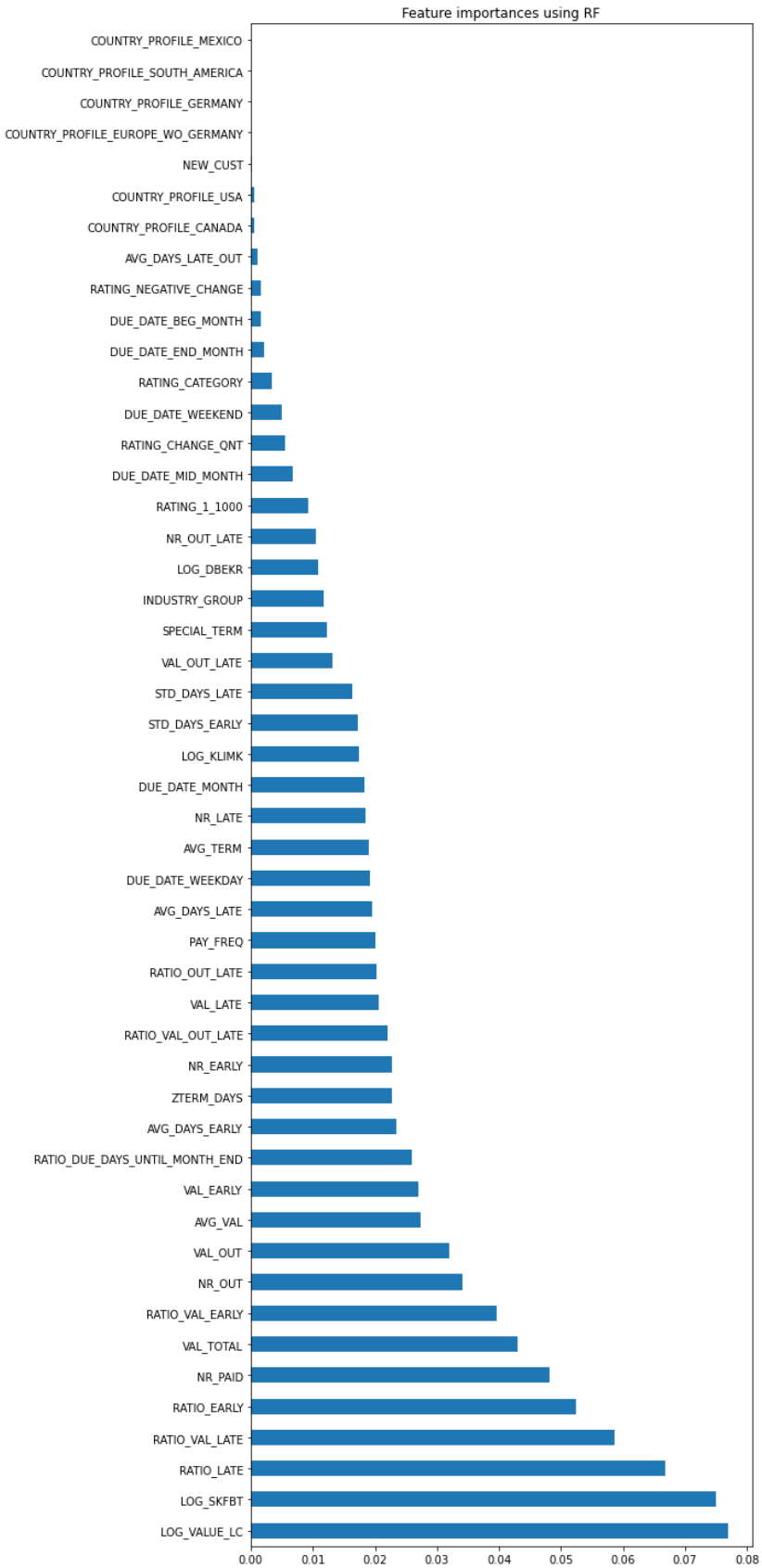


Figure 8.47 - Feature importance using RF for the 3-class problem

RF with resampled dataset (TARGET_3C)

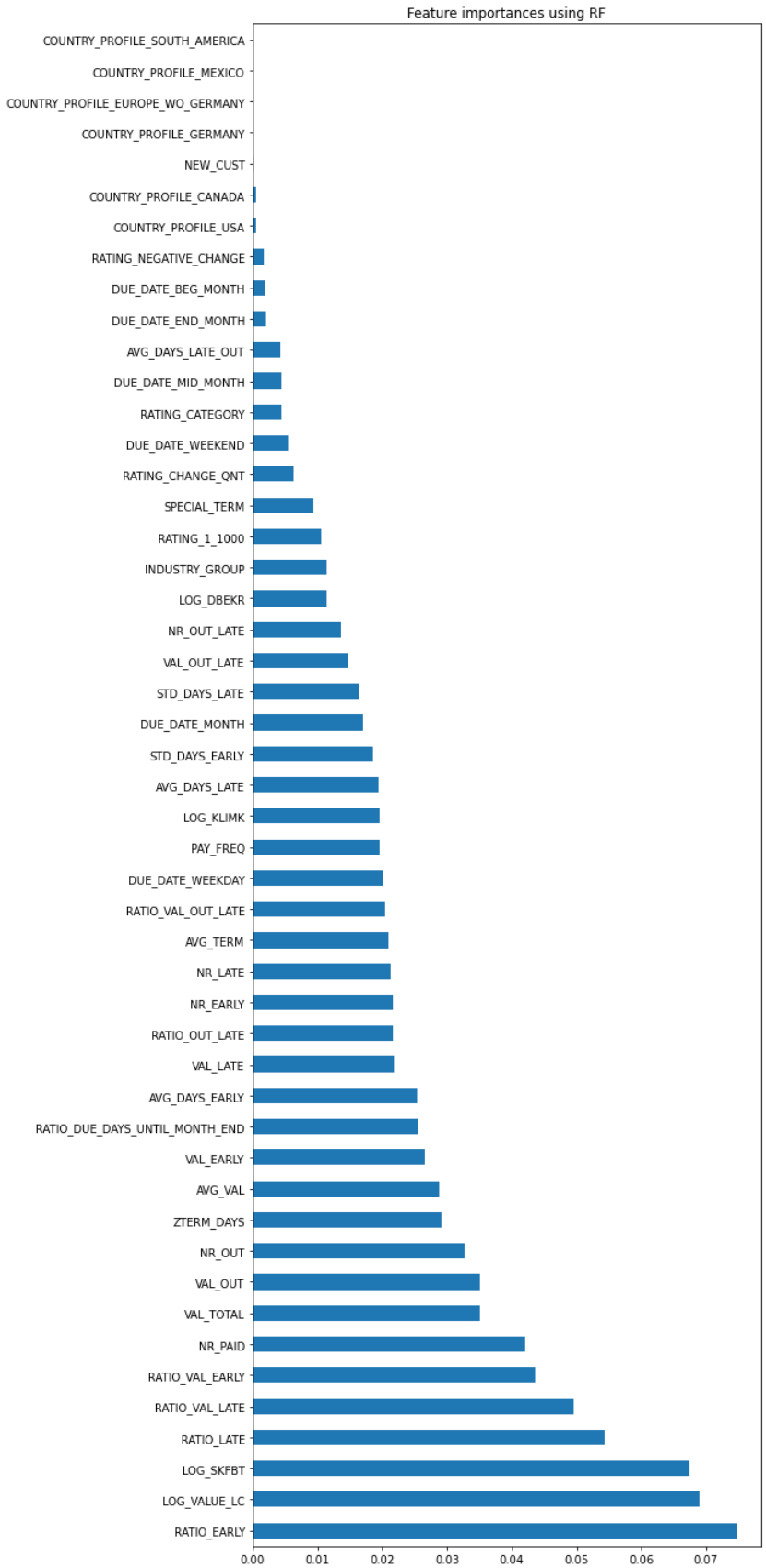


Figure 8.48 - Feature importance using RF with a resampled dataset for the 3-class problem

GBM (TARGET_3C)

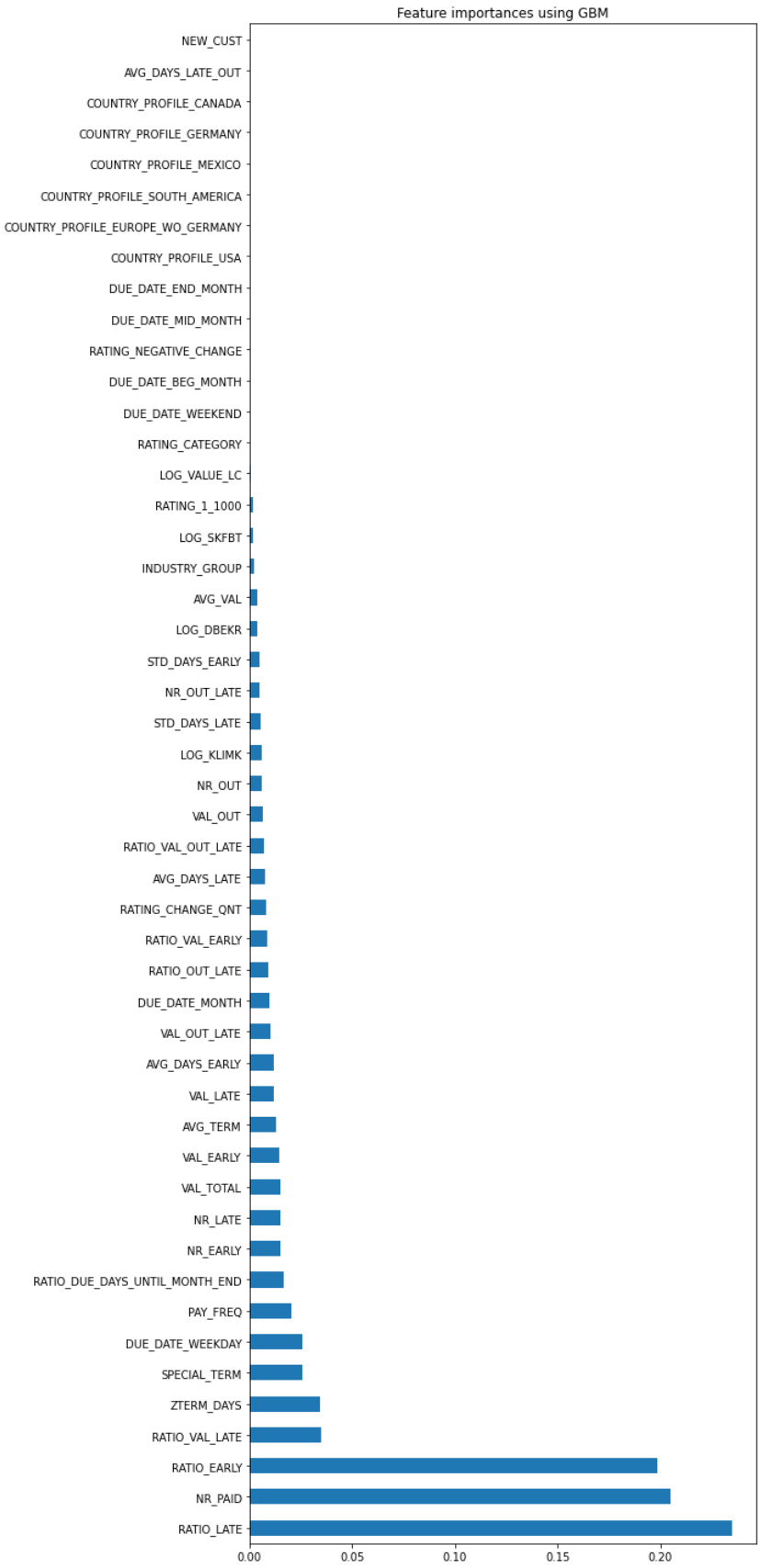


Figure 8.49 - Feature importance using GBM for the 3-class problem

GBM with resampled dataset (TARGET_3C)

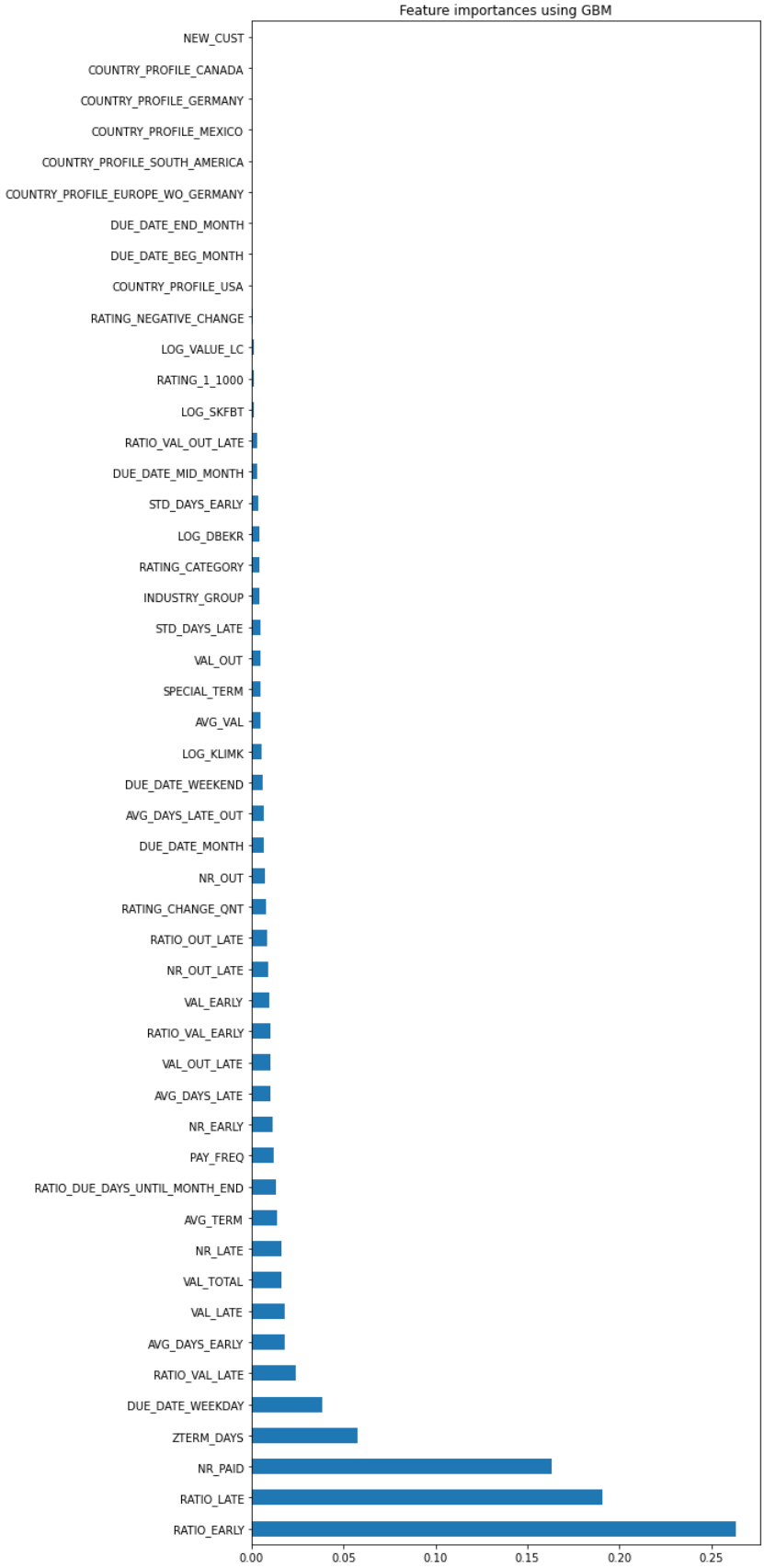


Figure 8.50 - Feature importance using GBM with a resampled dataset for the 3-class problem

LGBM (TARGET_3C)

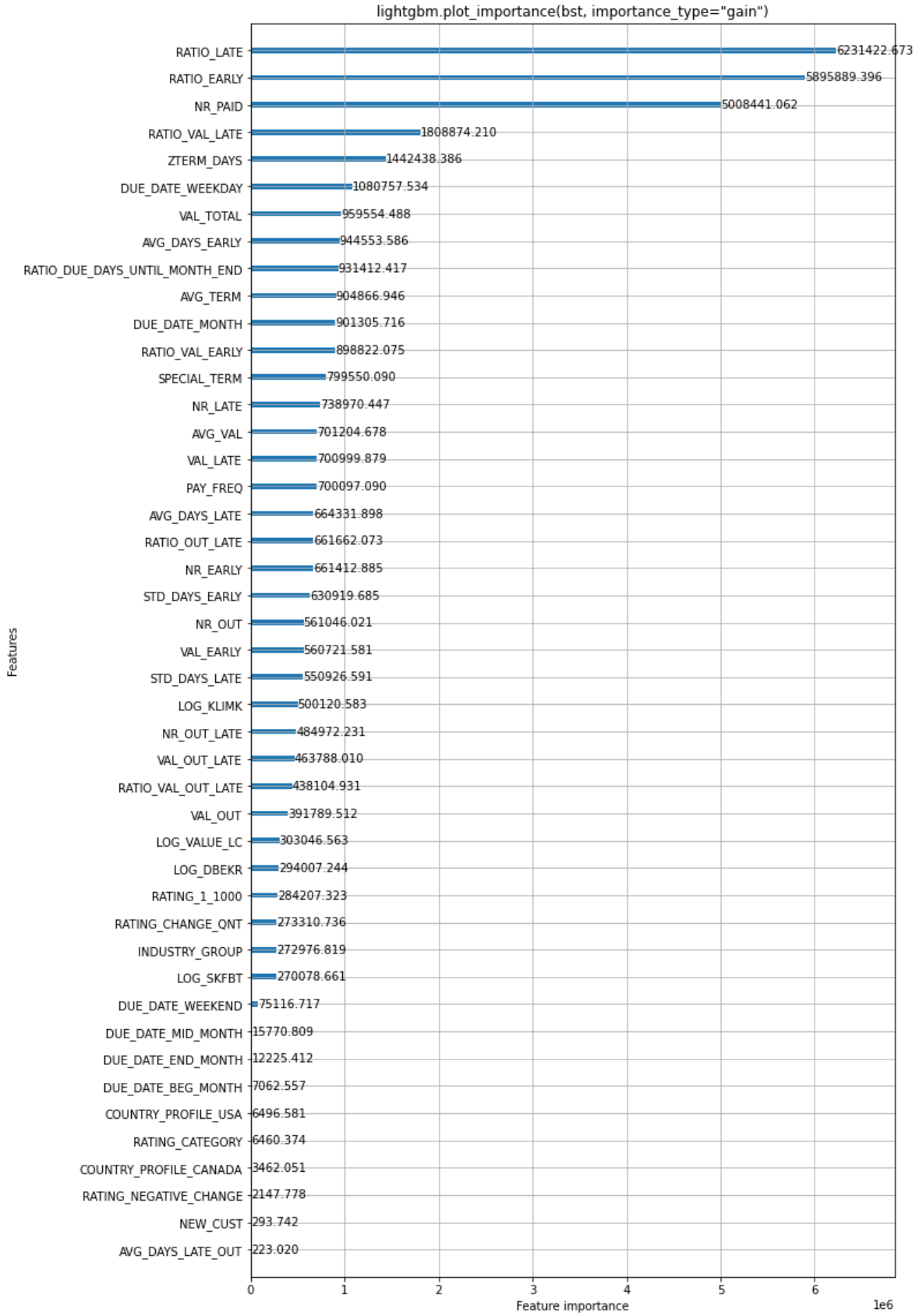


Figure 8.51 - Feature importance by gain using LGBM for the 3-class problem

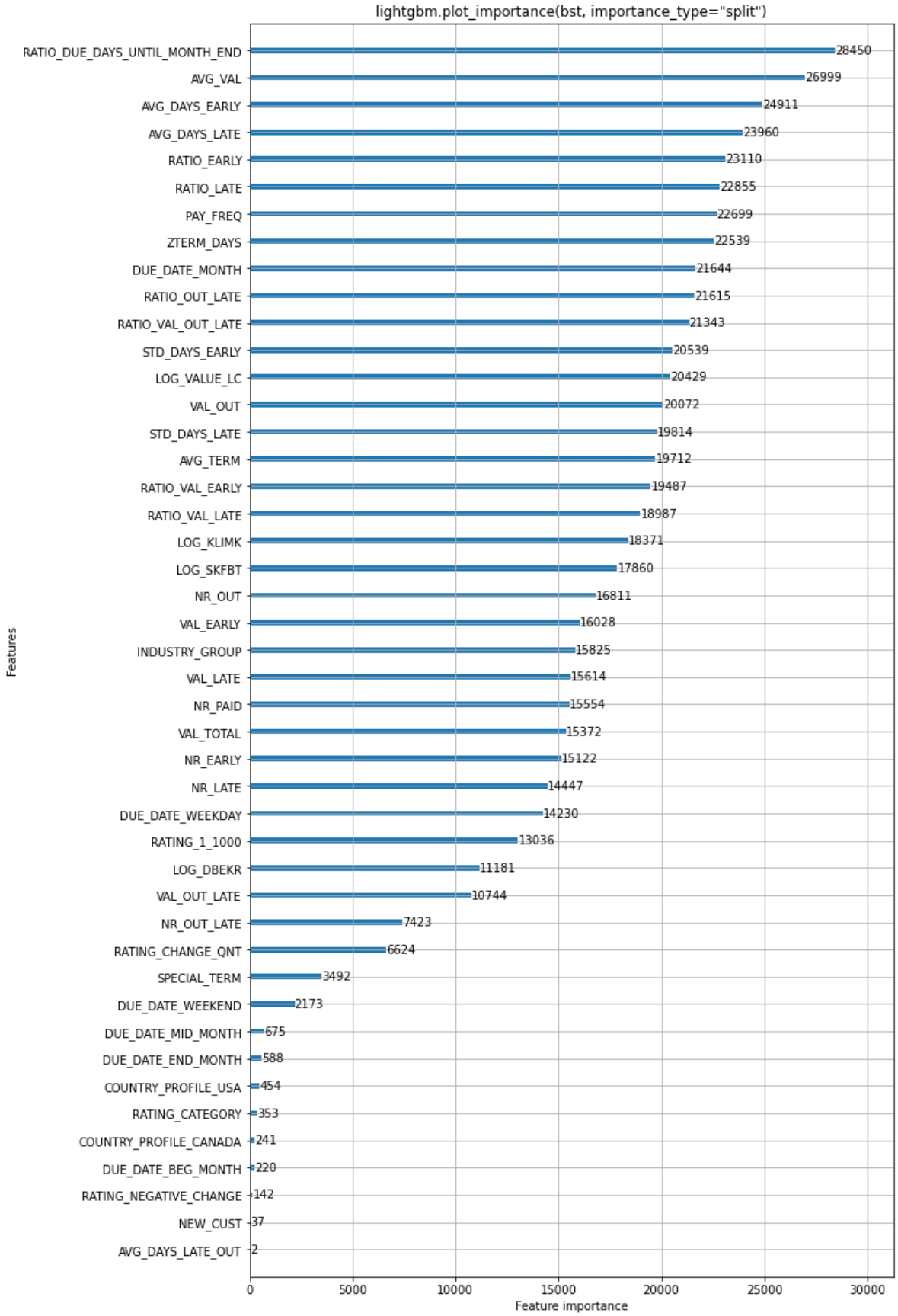


Figure 8.52 - Feature importance by split using LGBM with for the 2-class problem

LGBM with resampled dataset (TARGET_3C)

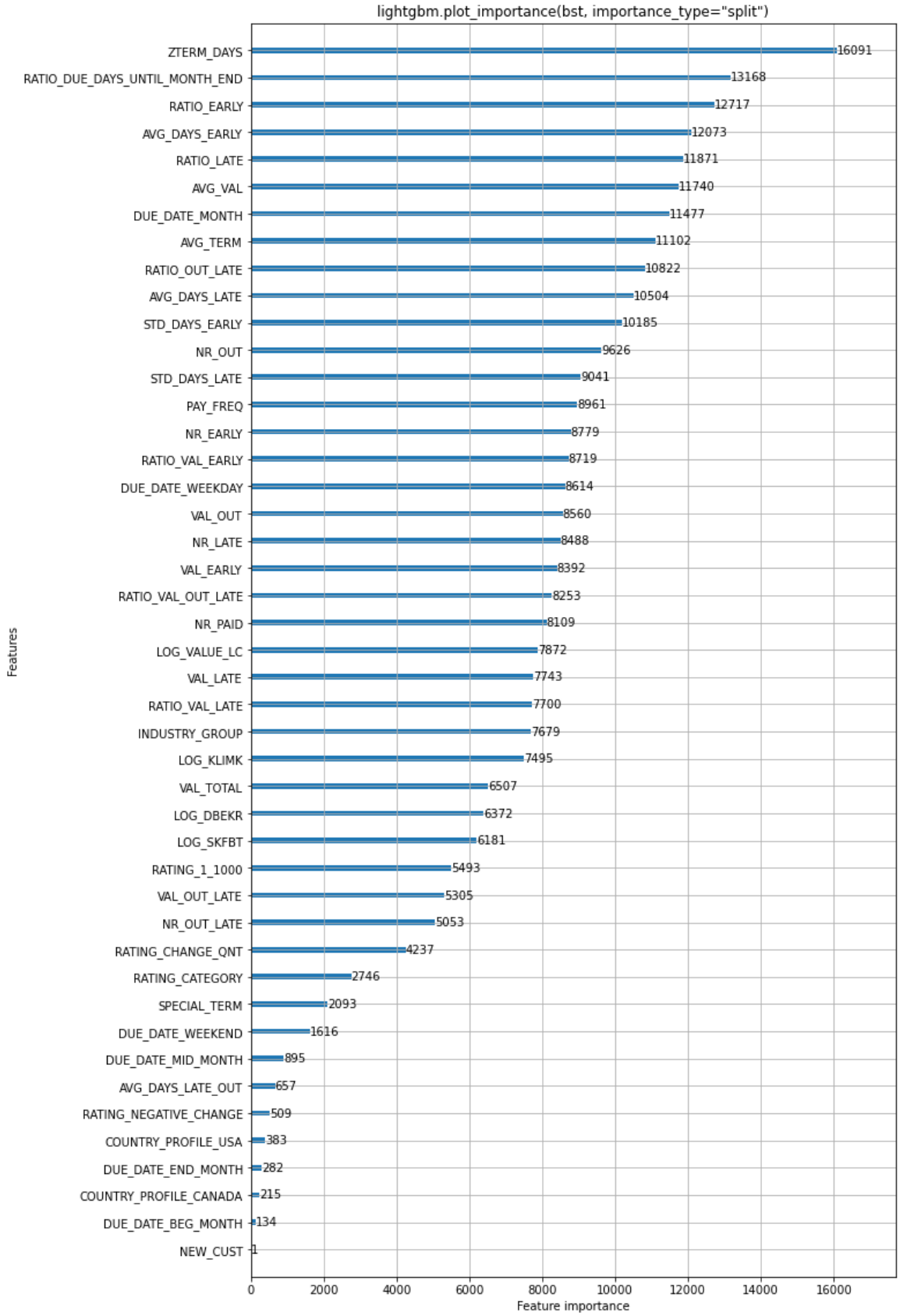


Figure 8.53 - Feature importance by gain using LGBM with a resampled dataset for the 3-class problem

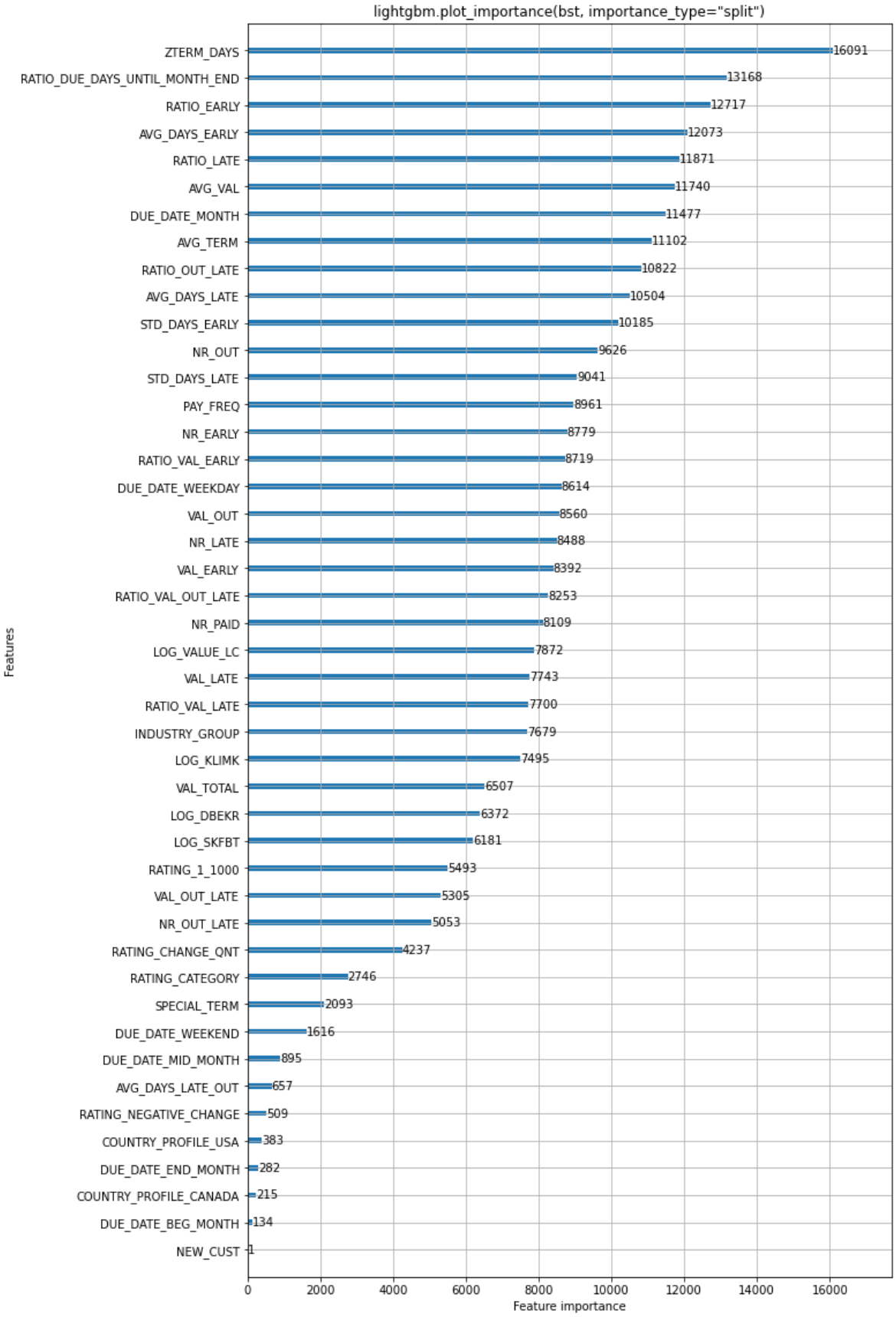


Figure 8.54 - Feature importance by split using LGBM with a resampled dataset for the 3-class problem

LR (TARGET_5C)

Figure 8.55 - Feature importance using LR for the 5-class problem

SVM (TARGET_5C)

Figure 8.56 - Feature importance using GBM for the 5-class problem

DT (TARGET_5C)

Figure 8.57 - Feature importance using DT for the 5-class problem

RF (TARGET_5C)

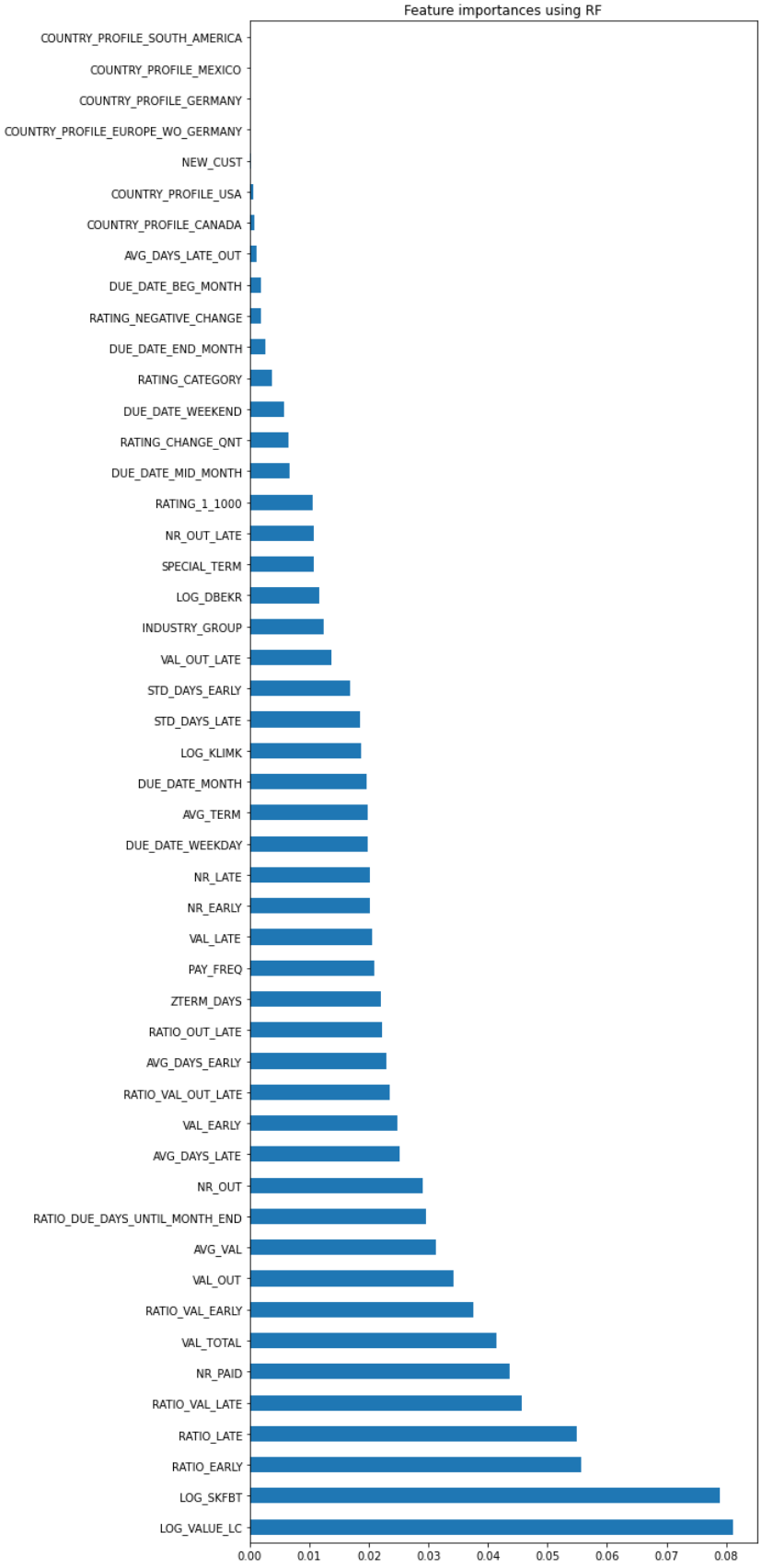


Figure 8.58 - Feature importance using RF for the 5-class problem

RF with resampled dataset (TARGET_5C)

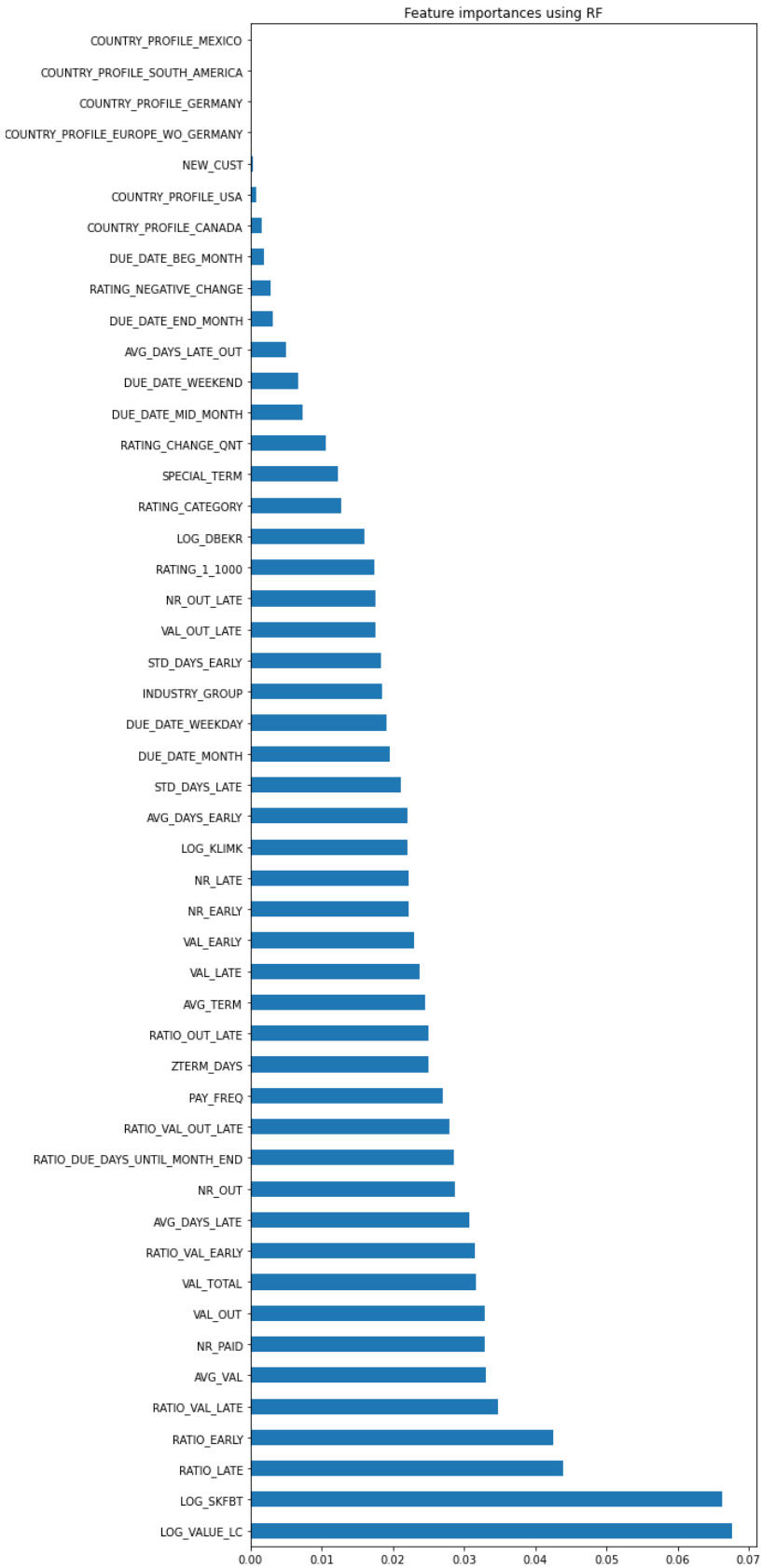


Figure 8.59 - Feature importance using RF with a resampled dataset for the 5-class problem

GBM (TARGET_5C)

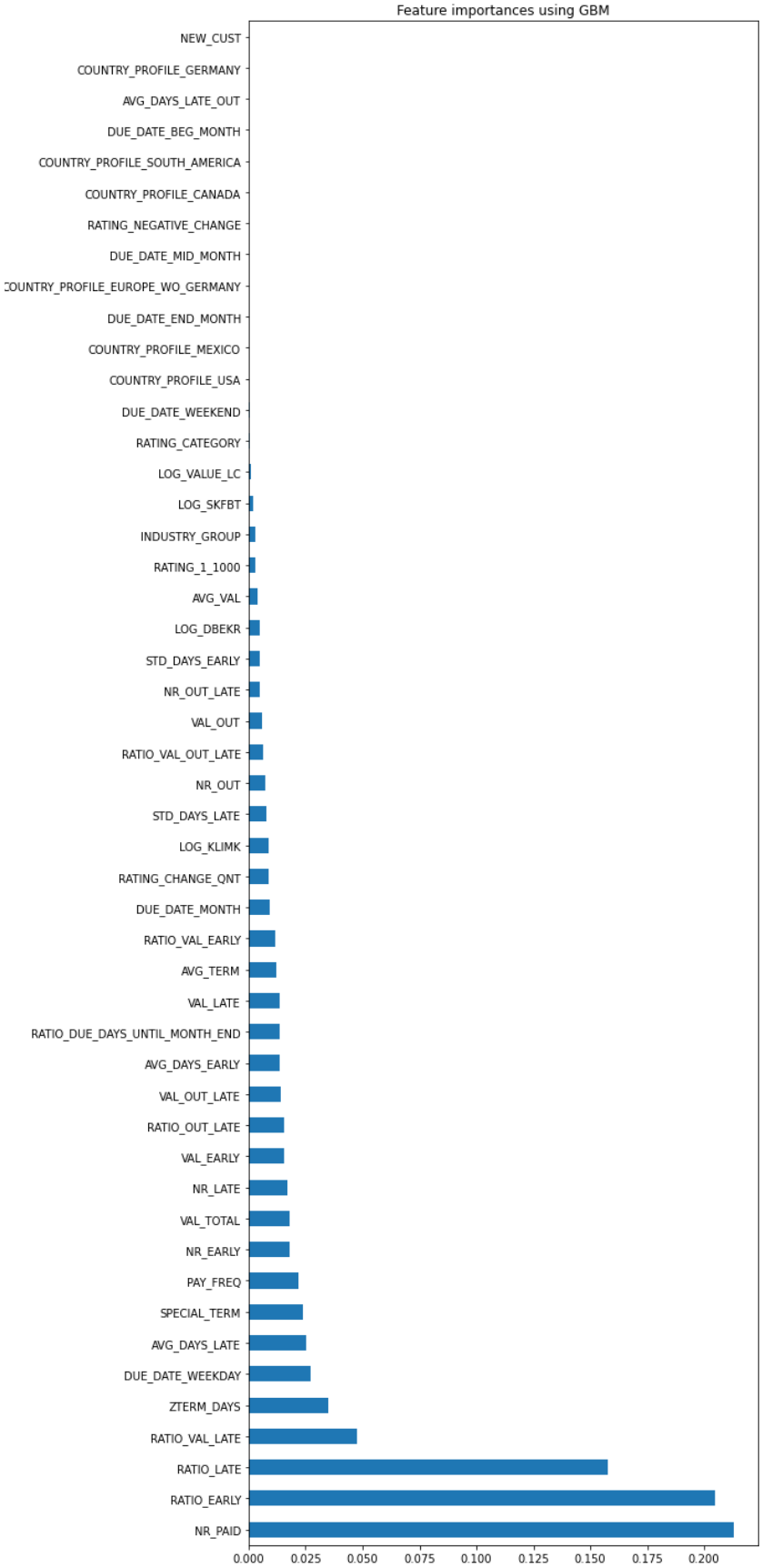


Figure 8.60 - Feature importance using GBM for the 5-class problem

GBM with resampled dataset (TARGET_5C)

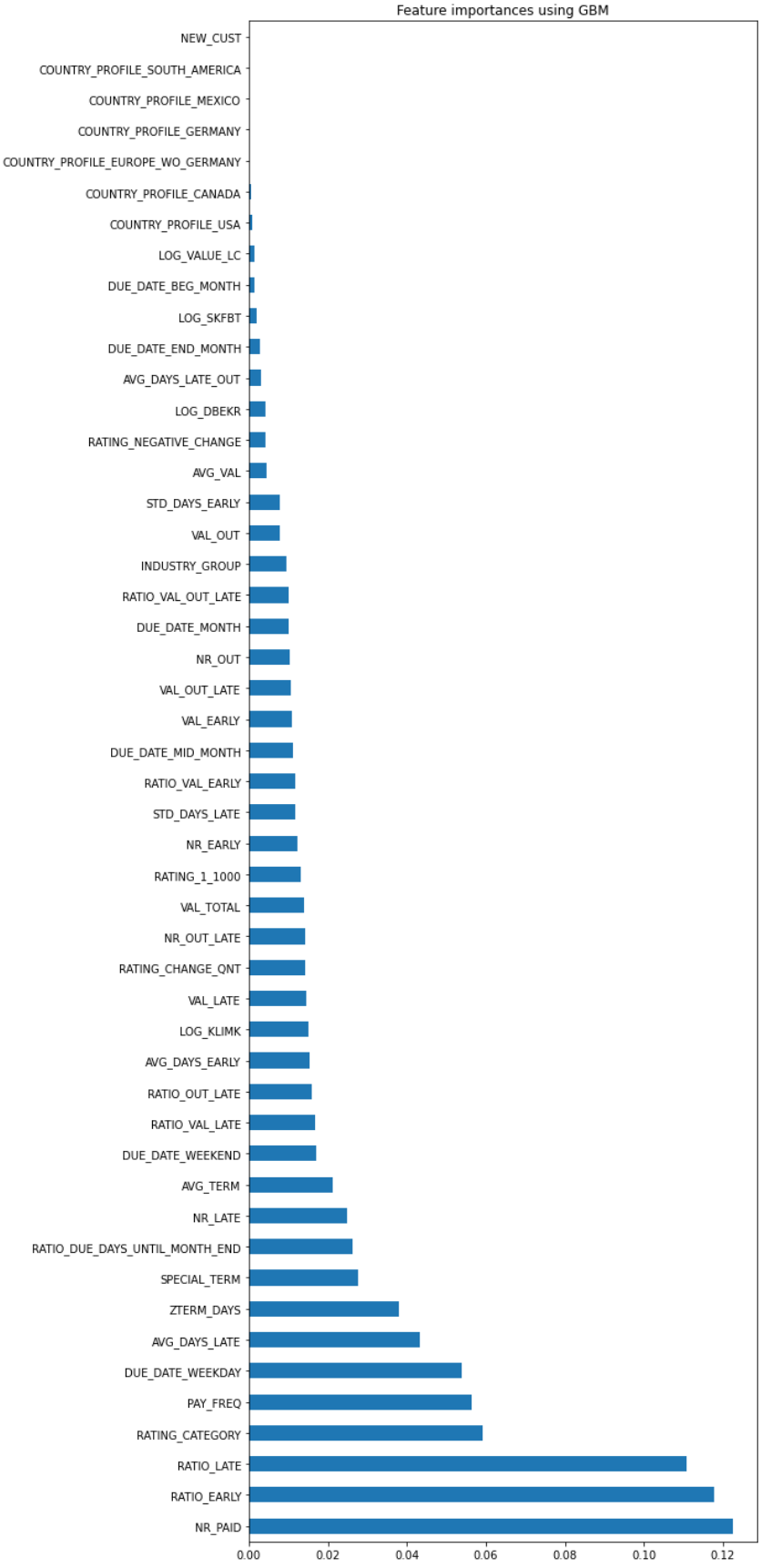


Figure 8.61 - Feature importance using LGBM with a resampled dataset for the 5-class problem

LGBM (TARGET_5C)

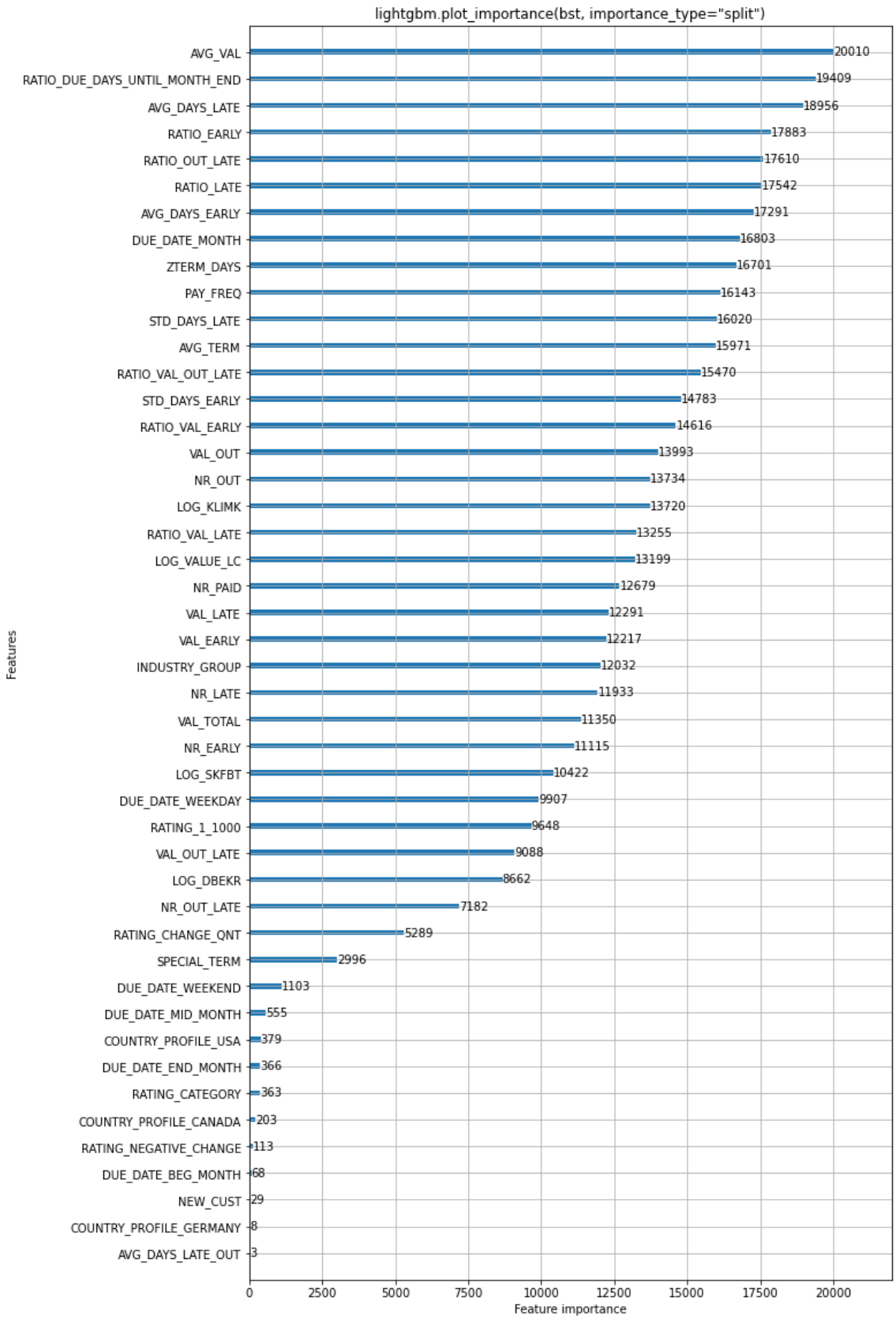


Figure 8.62 - Feature importance by split using LGBM for the 5-class problem

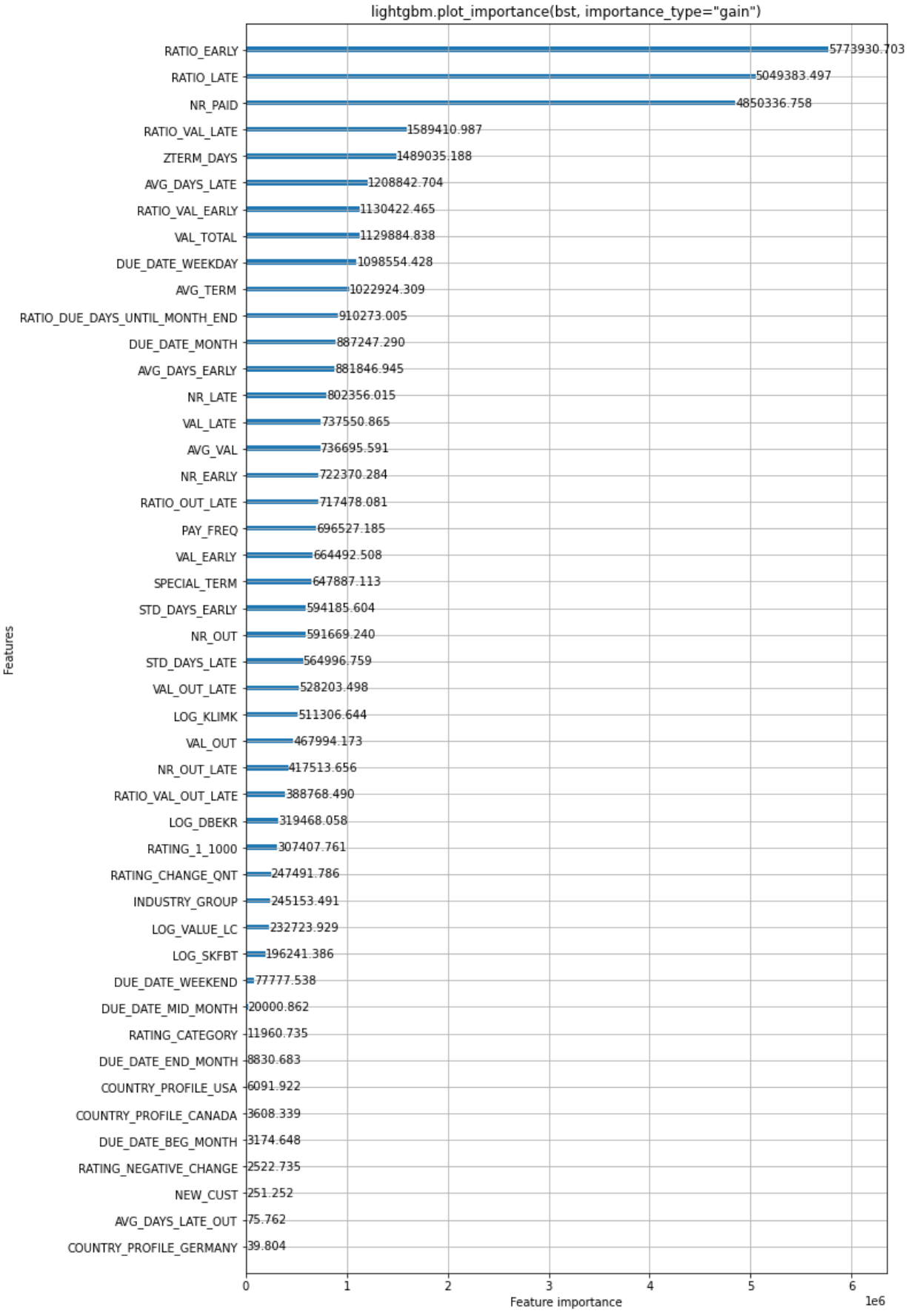


Figure 8.63 - Feature importance by gain using LGBM for the 5-class problem

LGBM with resampled dataset (TARGET_5C)

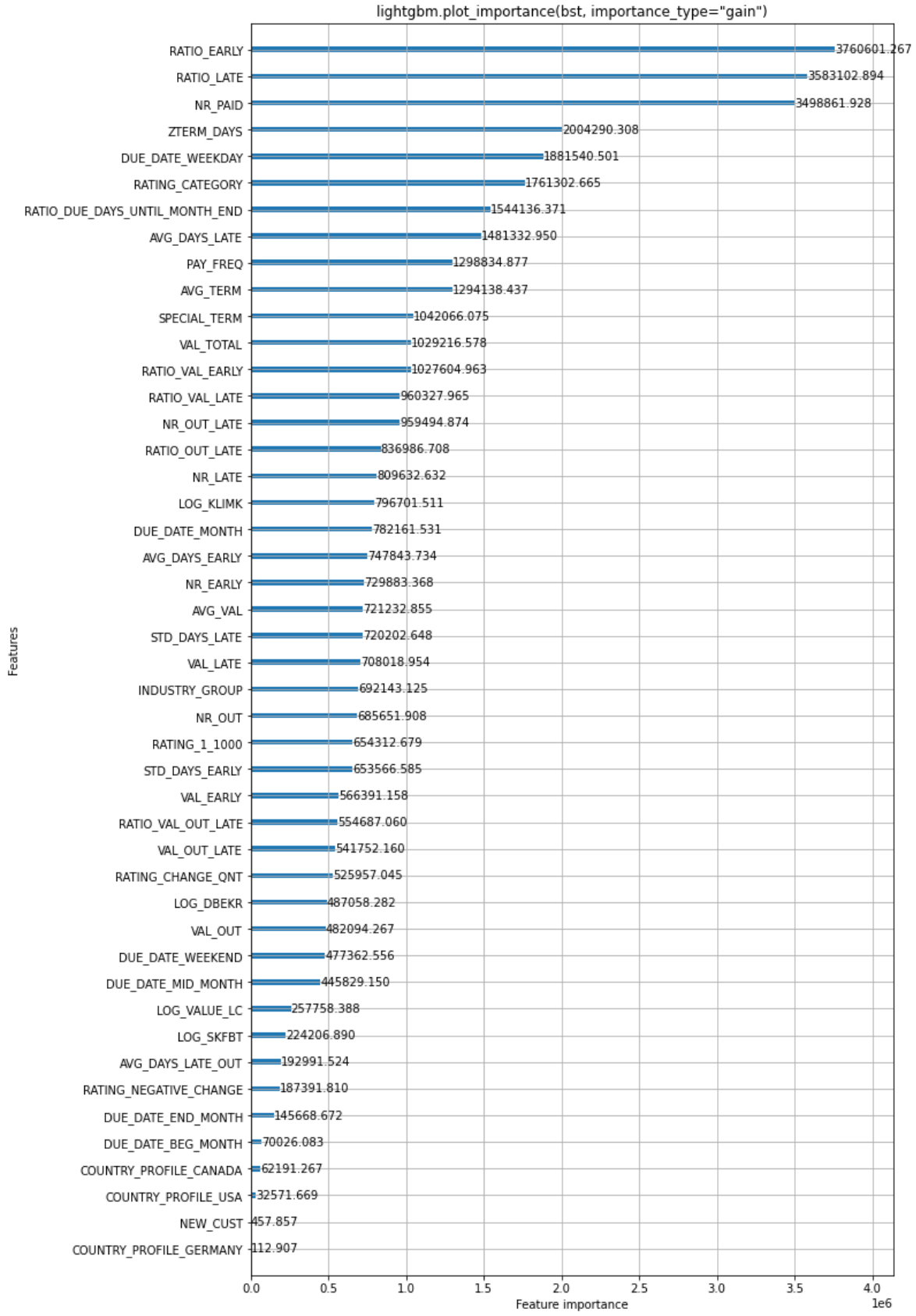


Figure 8.64 - Feature importance by gain using LGBM with a resampled dataset for the 5-class problem

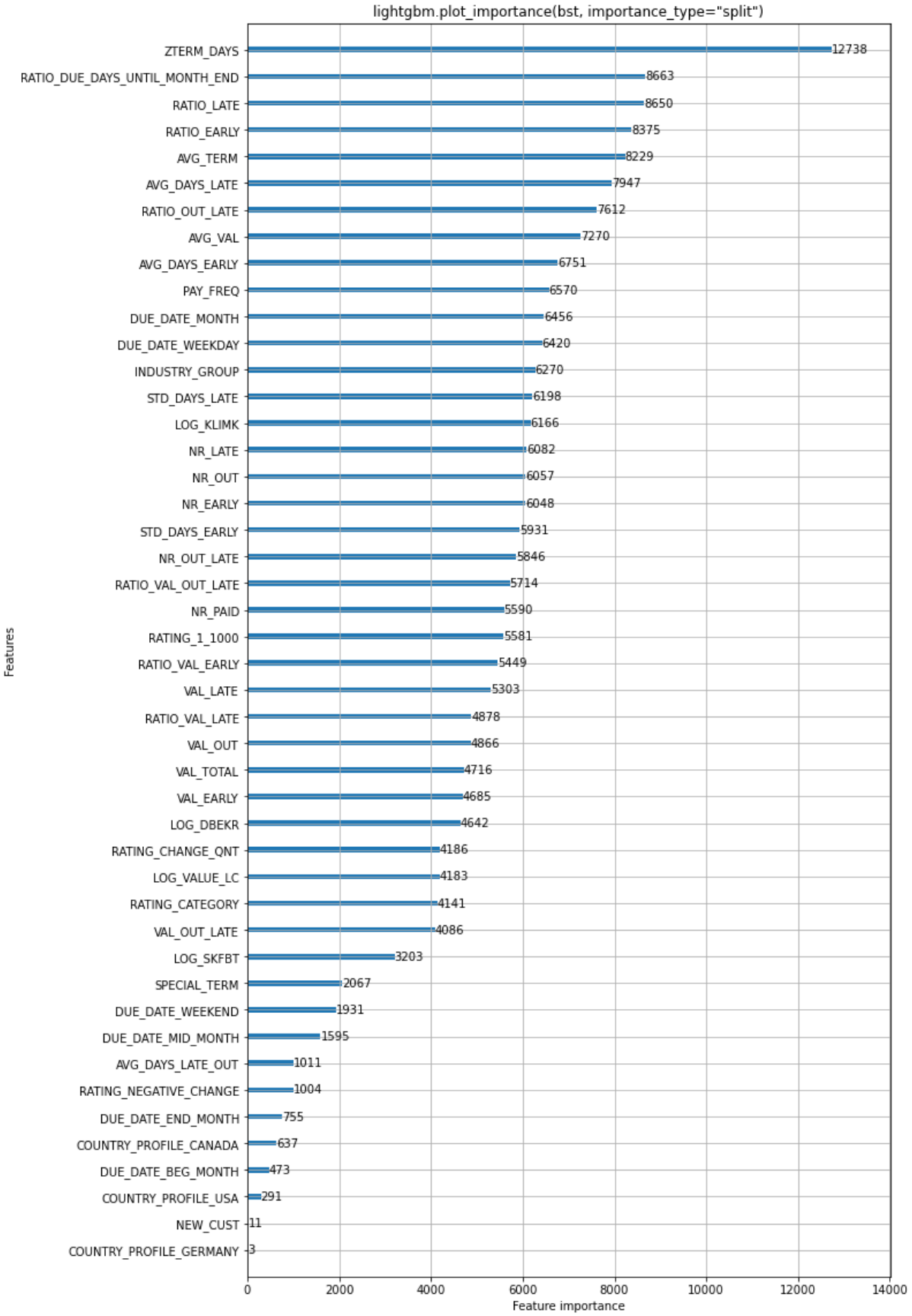


Figure 8.65 - Feature importance by split using LGBM with a resampled dataset for the 5-class problem