UNIVERSITY OF THE PELOPONNESE & NCSR "DEMOCRITOS"
MSC PROGRAMME IN DATA SCIENCE

Can AI Predict Loan Success? A Comparative Study on Imbalanced and Noisy Greek Call Transcripts

by

Ilias Iliopoulos

A thesis submitted in partial fulfillment
of the requirements for the MSc
in Data Science

Supervisor: Ilias Zavitsanos
Researcher

Athens, December 2024

Can AI Predict Loan Success? A Comparative Study on Imbalanced and Noisy Greek Call Transcripts

Ilias Iliopoulos

MSc. Thesis, MSc. Programme in Data Science

University of the Peloponnese & NCSR "Democritos", December 2024

UNIVERSITY OF THE PELOPONNESE & NCSR "DEMOCRITOS"
MSC PROGRAMME IN DATA SCIENCE

Can AI Predict Loan Success? A Comparative Study on Imbalanced and Noisy Greek Call Transcripts

by

Ilias Iliopoulos

A thesis submitted in partial fulfillment
of the requirements for the MSc
in Data Science

Supervisor: Ilias Zavitsanos
Researcher

Approved by the examination committee on December, 2024.

(Signature) (Signature)

. . . . . .

Spyridon Skiadopoulos
Professor, Director of MSc in Data Science

. . . . . .

Anastasia Krithara
Researcher

Athens, December 2024

UNIVERSITY OF THE PELOPONNESE & NCSR "DEMOCRITOS"
MSC PROGRAMME IN DATA SCIENCE

Declaration of Authorship

(1) I declare that this thesis has been composed solely by myself and that it has not been submitted, in whole or in part, in any previous application for a degree. Except where states otherwise by reference or acknowledgment, the work presented is entirely my own.

(2) I confirm that this thesis presented for the degree of Bachelor of Science in Informatics and Telecommunications, has

   (i) been composed entirely by myself

   (ii) been solely the result of my own work

   (iii) not been submitted for any other degree or professional qualification

(3) I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or processional qualification except as specified.

(Signature)

.....

Ilias Iliopoulos

Athens, December 2024

Acknowledgments

I would like to express my deep gratitude to my supervisor, Dr. Ilias Zavitsanos, who consistently dedicated time and effort to address my questions and patiently guided me through the difficult challenges and tough decisions that had to be made throughout the course of this study.

I would also like to thank Qualco for providing their data and expertise, and particularly Dr. Panayis Fourniotis Pavlatos and Dr. Panagiotis Panagiotopoulos for their invaluable insights and assistance in navigating the dataset's intricacies and the business complexities in the sector of credit management and particularly loan arrangements.

I extend my sincere thanks to my committee members, Dr. Spyridon Skiadopoulos and Dr. Anastasia Krithara for their constructive feedback and thought-provoking suggestions during my presentation and defense.

Finally, I would like to thank my wife and family for their unconditional support and love and my little son for bringing me so much joy and giving me the strength to continue until the end.

To my family.

Περίληψη

Η παρούσα εργασία ασχολείται με το απαιτητικό πρόβλημα πρόβλεψης της επιτυχίας ενός δανειακού διακανονισμού, για το οποίο χρησιμοποιήθηκαν μη δομημένα δεδομένα (unstructured data). Τα δεδομένα αυτά πηγάζουν από σχόλια που έγραψαν υπάλληλοι τηλεφωνικών κέντρων κατά τη διάρκεια συνομιλιών με τραπεζικούς πελάτες. Η εργασία συνιστά μια συγκριτική μελέτη διαφόρων μοντέλων μηχανικής και βαθιάς μάθησης καθώς και τεχνικών επεξεργασίας φυσικής γλώσσας (NLP), οι οποίες χρησιμοποιούνται για να αντιμετωπίσουν την υψηλή ανισότητα στις αναλογίες των κλάσεων (class imbalance), με τη θετική κλάση να αποτελεί μόλις το 0.2% του συνόλου των δεδομένων. Επιπρόσθετη δυσκολία προκύπτει και από το αυξημένο επίπεδο θορύβου, το οποίο οφείλεται τόσο στις ασυνήθεις συντομογραφίες, τις συντακτικές και γραμματικές αβλεψίες και την έλλειψη επίσημου τόνου, όσο και στην ασθενή επίβλεψη (weak supervision) που έχει εφαρμοστεί για την εκπαίδευση των μοντέλων. Σχετικά με την ασθενή επίβλεψη, είναι χρήσιμο να σημειωθεί ότι οι ετικέτες των κλάσεων έχουν δημιουργηθεί βάσει επιχειρηματικών κανόνων, και όχι από ειδικούς εμπειρογνώμονες στην κατηγοριοποίηση κειμένων. Η εκτενής μελέτη διαφορετικών μεθόδων περιλαμβάνει από παραδοσιακά μοντέλα μάθησης όπως η λογιστική παλινδρόμηση και το XGBoost, τα οποία χρησιμοποιούνται σε συνδυασμό με τεχνικές εξαγωγής χαρακτηριστικών οπώς το term frequency - inverse document frequency (TF-IDF) και τα word embeddings, μέχρι πιο πολύπλοκα γλωσσικά μοντέλα τα οποία μπορούν να κατανοήσουν βαθύτερες νοηματικές έννοιες στα κείμενα, όπως είναι τα GreekBERT, Meltemi και Llama 3.1 70B. Επιπρόσθετα, χρησιμοποιήθηκαν διαφορετικά είδη συναρτήσεων κόστους, με τα πιο αξιοσημείωτα να είναι η συγκριτική συνάρτηση κόστους βάσει αντιδιαστολής αντικρουόμενων παραδειγμάτων (contrastive loss) και η συνάρτηση κόστους που χρησιμοποιεί συντελεστές βάρους που είναι αντίστοιχοι των αναλογιών

των κλάσεων (class-weighted loss). Είναι άξιο να αναφερθεί ότι η μελέτη αντιμετώπισε περιορισμούς στους διαθέσιμους υπολογιστικούς πόρους, πέραν της απουσίας ικανού αριθμού θετικών παραδειγμάτων, κάτι το οποίο οδήγησε σε πειραματισμό με διάφορες μεθόδους βελτιστοποίησης, όπως η Προσαρμογή Χαμηλής Τάξεως (Low Rank Adaptation - LoRA), η συμπίεση των μοντέλων μέσω διακριτοποίησης (quantization), η συσσώρευση των ανάδελτα (gradients), η μείωση του ρυθμού μάθησης (learning rate decay), και η μείωση των συντελεστών βαρύτητας του μοντέλου (weight decay). Με σκοπό να περιοριστεί ο θόρυβος στα κείμενα, εφαρμόστηκαν τόσο προσεγγίσεις βάσει κανόνων (rule-based), όσο και μέθοδοι που στηρίζονται στην τεχνητή νοημοσύνη, όπως ο καθαρισμός των κειμένων από ένα σύγχρονο μεγάλο πολυγλωσσικό μοντέλο (LLM). Σχετικά με την ανισορροπία στην κατανομή των κλάσεων, διερευνήθηκε ακόμη η παραγωγή συνθετικών δεδομένων από ένα μεγάλο γλωσσικό μοντέλο, για να αποσαφηνιστεί η αποτελεσματικότητα και ο αντίκτυπος που θα μπορούσε να έχει μια τέτοια επιλογή στην αποτίμηση της επίδοσης του μοντέλου. Όλα τα προαναφερθέντα πειράματα αξιολογήθηκαν και συγκρίθηκαν ενδελεχώς, χρησιμοποιώντας μια εκτενή λίστα από μετρικές, συμπεριλαμβανομένων κλασικών όπως η συνολική ορθότητα (accuracy) και η ικανότητα ανάκλησης (recall), καθώς και μετρικών που χρησιμοποιούνται συνηθέστερα σε μοντέλα κατάταξης (ranker models), όπως η ικανότητα ανάκλησης σε K παραδείγμα (Recall@K), η ακρίβεια σε R παραδείγματα (R-Precision) και η περιοχή κάτω από την καμπύλη ακρίβειας - ανάκλησης (PR-AUC). Η μελέτη καταλήγει με μια συζήτηση περί των αποτελεσμάτων και ακόμα τονίζει τις πιο αποτελεσματικές μεθόδους για το χειρισμό ανισορροπίας στις κλάσεις και θορύβου στα δεδομένα. Προτεινόμενες ερευνητικές κατευθυντήριες γραμμές για το μέλλον περιλαμβάνουν την ανάπτυξη πιο ικανών ελληνικών γλωσσικών μοντέλων, την αναζήτηση τεχνικών για τον καθαρισμό, την επαύξηση και τη σύνθεση δεδομένων, και τη δημιουργία δημόσια διαθέσιμων συνόλων δεδομένων. Όλες αυτές οι ενέργειες δύνανται να αποτελέσουν εφαλτήριο για περαιτέρω εξελίξεις στην επεξεργασία της ελληνικής γλώσσας και είναι ικανά να συμβάλουν στην εκδημοκρατικοποίηση της πρόσβασης σε καινοτόμα συστήματα τεχνητής νοημοσύνης.

Abstract

This thesis deals with the challenging task of predicting loan arrangement success based on noisy Greek comments written by call center agents during communications with the customers. It is a comparative study of various machine and deep learning models and NLP techniques, employed to tackle an extremely imbalanced dataset (with the positive class accounting for only the 0.2% of the total), and a high level of noise in the data, which stems both from abbreviations, syntactical and grammatical errors, and informalities, as well as from the weak supervision that has been used, since the labels were constructed based on business rules, instead of having been assigned by experts. The extensive exploration of different approaches ranges from traditional models like Logistic Regression and XGBoost, which are used in combination with feature extraction techniques such as term frequency - inverse document frequency (TF-IDF) and word embeddings, extending to more advanced language models capable of capturing deeper semantic meanings from the texts, like GreekBERT, Meltemi, and Llama 3.1 70B. Furthermore, different kinds of loss functions were utilized, the most notable being contrastive and class-weighted losses. The study was also severely constrained by limited computation resources besides the lack of positive examples, hence leading to experimentation with various optimization methods, like Low Rank adaptation (LoRA), quantization, gradient accumulation, learning rate decay, and weight decay. In order to mitigate the noise in the texts, both rule-based as well as AI-supported appraches were followed, including text refinement by a modern multilingual LLM. Regarding the uneven class distribution, synthetic data generation by a LLM was also put under investigation, to determine the efficacy and the impact this process could have in the performance evaluation. All these experiments were rigorously evaluated and

compared using a comprehensive list of metrics, including classic ones like accuracy and recall, and also including metrics that are traditionally used for ranking models, like Recall@K, R-Precision and PR-AUC. The study concludes with discussion of the results and highlights the most effective strategies for handling the scarcity of positive instances and noise of data. Proposed future research directions include the development of more robust Greek language models, the exploration of advanced data refinement, augmentation and generation techniques, and the creation of publicly available datasets, in order to foster further advancements in Greek NLP and further democratize the access to novel AI tools.

Contents

List of Abbreviations

NLP	Natural Language Processing
ML	Machine Learning
DL	Deep Learning
POS	Part of speech
BoW	Bag of words
TF-IDF	Term frequency - Inverse document frequency
SVM	Support Vector Machine
SVC	Support Vector Classifier
RNN	Recurrent Neural Networks
LSTM	Long Short-Term Memory
OCR	Optical Character Recognition
ASR	Automatic Speech Recognition
DRO	Distributional Random Oversampling
ROC	Receiver Operating Characteristic
AUC	Area Under Curve
CAT	Contrastive Adversarial Training
MT	Machine Translation

BERT	Bidirectional encoder representations from transformers
XGB	eXtreme Gradient Boosting
RF	Random Forests
MNB	Multinomial Naive Bayes
ABC	AdaBoost Classifier
DT	Decision Tree
GPU	Graphics Processing Unit

Chapter 1

Introduction

1.1 Motivation

The early stages of the 4^{th} AI-driven industrial revolution, which change dramatically almost every aspect of the society and the public life, could not leave the financial and banking sector, which lies at the heart of nearly every human activity, untouched. Fintech, with the innovation and cutting-edge technologies that it introduces, plays a more and more crucial role towards the transition into this new era. This radical redefinition of central parts of the economy should be governed by the ideals of democratization and has to aspire to equal access and opportunities for all people, increased and well-established trust for institutions, as well as open platforms and utilities.

Towards this direction, my personal academic journey as part of the MSc Degree programme in Data Science, which comes to a conclusion with the undertaking of the current study, was provided with a unique and exciting challenge. Specifically, the collaboration with a major fintech solutions provider in the Credit Management sector, Qualco, presented the interesting and intriguing task of predicting the probability of loan arrangements based on unstructured data. This provided the unique possibility and fulfilling experience to work with real-world datasets in a real-world business case, under the supervision and advice of experienced researchers and professionals in the realm of data analysis and machine learning. The advantages and

benefits of this work was multifaceted.

On the one hand, there is always the strictly technical part, which includes working with fascinating technologies to build predictive models and bring value by solving customers and business problems, that undoubtedly contributed to honing my skills and refining my knowledge in this exciting research area. But what was even more important was the fact that the current dissertation offered a direction that is highly relevant to the bigger picture of our times. The most critical objective nowadays is to not only protect and take care of the business interests of big financial players by helping them to avoid unnecessary risk and bad decision making, but also to benefit the financial customers and individuals with models that understand their deeper needs, show empathy, and create solutions to tackle down financial obstacles and foster chances for success.

In this aforementioned effort I firmly believe, and I felt that I became a small part of. In my humble opinion, technologies like these can pave the road to prevent mistakes of the past, which brought economic difficulty and distress to large portions of the population. They can help society navigate through the new complex landscape that is rapidly forming, and eventually reap the benefits towards a brighter and fairer future.

1.2 Contributions

This study makes several contributions to the field of text classification and Natural Language Processing (NLP) for the Greek language, especially focusing on the extreme class imbalance, as well as the intrinsic and generated noise that exists in the texts and the labels of the dataset.

The thesis presents a comprehensive comparative analysis of a wide variety of machine and deep learning models, using several techniques to pre-process the texts and handle the lack of positive examples and the noisiness of the data. The analysis also offers deep insights in the performance and capabilities of the explored models, ranging from traditional ML models like Logistic Regression, SVM, XGBoost and Random Forests, and extending up to modern language models like the GreekBERT,

Meltemi, XLM-RoBERTa, FinBERT and Llama 3.1 70B.

Regarding the class imbalance and the noise, the current study presents a novel approach to refine the text data and generate new synthetic examples, by being one of the few works of research that uses a state-of-the-art LLM (OpenAI's GPT4-o mini) to clean the texts and create new positive instances. The effectiveness and impact this method could have on fine-tuning smaller and specialized on the Greek language models, like GreekBERT and Meltemi, is evaluated and discussed. The dissertation also investigates and assesses several other approaches like the usage of class-weighted and contrastive losses, and data refinement using a rule-based method based on the Spacy Python NLP library. All these experiments contribute to determining and understanding successful strategies in order to handle common issues in real-world NLP tasks, like the lack of positive examples and unstructured data of low quality.

Furthermore, the thesis uses several optimizations for the efficient training and fine-tuning of language models on limited computational resources, like gradient accumulation, learning rate decay, weight decay, and early stopping of the training process. Hence, it assists in realizing the existing capabilities and the results that can be achieved on consumer-grade hardware, sacrificing little accuracy and predictive power, while achieving big improvements on training times and reduction of the required memory that needs to be allocated.

To sum up, this study offers a fresh look on the landscape of Greek NLP methods for binary text classification using call center transcripts on a competitive domain like the credit management sector and with a focus on predicting loan arrangement outcomes. The unique challenges that the data presented and the constraints under which the study had to be overtaken created a unique opportunity for experimentation and employment of cutting-edge concepts and technologies. This adds further value if someone takes into consideration the limited NLP resources and the small existing body of work for a rich and complex language like Greek, especially when it is compared with most globally prominent ones like English or Chinese.

1.3 Thesis structure

The structure of the current thesis is the following:

- **Chapter 2:** The Background and Related Work chapter starts with describing basic terms and concepts and continues with related work that has been carried out by other researchers.
- **Chapter 3:** This chapter includes a mathematical definition of the problem that needs to be tackled. It then proceeds with describing the approach this study proposes, in order to deal with the objective of predicting the loan arrangement outcomes and to examine different methods and models.
- **Chapter 4:** Chapter 4 contain a detailed description of the process to construct and pre-process the study's dataset, the experiments that were performed, as well as the results and performance evaluation of the created models.
- **Chapter 5:** The final chapter of the dissertation proceeds with a conversation about the outcomes of the study and compares the attempted approaches. It concludes with the limitations of this work and suggestions for future work.
- **Appendix:** The Appendix A includes information about the Python libraries and frameworks that have been used in the code of this study.

Chapter 2

Background and Related Work

2.1 Basic Terms and Concepts

2.1.1 Text Classification

Text classification, or text categorization, is one of the most common Natural Language Processing (NLP) tasks and involves predicting class labels for text documents based on their contents [1, 2]. This is very important for various applications, such as sentiment analysis, spam detection, and topic categorization [3]. The goal of this thesis is to classify Greek texts, which are communications made with banking customers with non-performing loans, into two categories: a loan arrangement is either likely to happen in the upcoming two months or not.

The text classification usually involves the following steps [4–6]:

1. **Data Collection:** Gathering a dataset of text documents with information that is relevant to the application [4].

2. **Data Pre-processing:** Cleaning and preparing the text data for further analysis, a step which usually includes tokenization, lemmatization, stop words removal, labels assignment or annotating etc. [5].

3. **Feature Extraction:** This stage is responsible for converting the preprocessed texts into numbers that the computer can understand and which are

required for machine or deep learning algorithms [6].

4. **Model Training:** The classification model, which can be a simple ML model or a more sophisticated DL architecture, updates its internal parameters step by step, by trying to minimize the loss caused by wrong predictions. This loss is calculated by comparing the given true labels that have been created or assigned to the data examples with the predictions. [7].

5. **Evaluation:** Evaluating the model's performance on an unseen test dataset using metrics such as accuracy, precision, recall, F1-score etc. is of high importance, because this approach enables us to understand the model's capabilities and its real business value. [8].

2.1.2 Common NLP Techniques and Methods

2.1.2.1 Tokenization

Tokenization is a technique to split the texts into smaller chunks called tokens, which can be words, phrases, or even sentences [1, 9]. This process is necessary for analyzing text data, as it allows for a more detailed examination of the content, and towards this end there are several algorithms that can be used for tokenization purposes. As an example, a common and quite simple approach would be to use whitespaces and punctuations as delimiters. For example, given the sentence:

"The loan was approved,"

The tokenization that simply uses whitespace would return the tokens: `["The", "loan", "was", "approved"]`.

2.1.2.2 Lemmatization

The purpose of lemmatization is to extract the base or root forms of the words, which are called lemmas. Unlike stemming, which simply cuts prefixes or suffixes of the words, lemmatization tries to consider the context and the meaning of the

words, and for this reason, lemmatization typically requires knowledge of parts of speech (POS). For instance:

- The lemma of "running" (verb) is "run."
- The lemma of "better" (adjective) is "good."

Mathematically speaking, if w represents a word in its text form and POS(w) is the part of speech it represents, then the lemmatization function L can be expressed as:

$$L(w) = \text{lemma}(w, \text{POS}(w)) \tag{2.1}$$

This creates a function that accounts for the usage or the context in which a word is used, thus creating separate rules for words with multiple meanings[10].

2.1.2.3 Bag of Words (BoW)

Bag of Words is perhaps the simplest, but also quite effective method for turning text data into numerical representations. When this approach is used, each document can be transformed into a vector, where each dimension corresponds to a unique word of the whole existing vocabulary in the text corpus. More precisely, the value in each dimension reflects the frequency of that word in the document.

Given a document D with n unique words $w_1, w_2, \ldots, w_n$, the BoW methodology can be expressed as:

$$\text{BoW}(D) = [f(w_1), f(w_2), \ldots, f(w_n)] \tag{2.2}$$

where $f(w_i)$ is the frequency count of each word w_i in the document D. While BoW can capture word occurrences effectively, it does not take into account word order and grammar[1].

2.1.2.4 n-grams

The n-grams methodology is an extension of the Bag of Words approach and focuses on sequences of n consecutive words or tokens, in order to capture information that is hidden in the word order and the context [11]. An n-gram is defined as a contiguous sequence of n items from a given text, and some commons forms of n-grams include:

- **Unigrams:** Single words ($n = 1$).
- **Bigrams:** Two consecutive words ($n = 2$).
- **Trigrams:** Three consecutive words ($n = 3$).

For example, let's use the sentence:

"The loan arrangement was approved."

The bigrams are: `["The loan", "loan arrangement", "arrangement was", "was approved"]`.

The n-grams representation can improve the expressiveness of BoW by retaining some of the local context and word ordering information, which makes it more effective for tasks like text classification and sentiment analysis [12].

Although n-grams models suffer when they need to scale, due to the fact that the vocabulary size grows exponentially with n, they also remain a simple and effective baseline for many NLP tasks.

2.1.2.5 Term Frequency-Inverse Document Frequency (TF-IDF)

TF-IDF further improves on the BoW model by defining the importance of words within a document as relative to their frequency across all documents, and for this purpose it comprises two components:

1. **Term Frequency (TF):** It measures the count of appearances of a term in a document, divided by the sum of the counts of appearances for all terms that

exist in this document:

$$\mathrm{TF}(t, d) = \frac{\mathrm{count}(t, d)}{\sum_{t' \in d} \mathrm{count}(t', d)} \tag{2.3}$$

2. **Inverse Document Frequency (IDF):** It measures how important a term is across all documents, which can be calculated by taking the log of the count of all documents N, divided by the count of documents this particular term exists in:

$$\mathrm{IDF}(t) = \log\left(\frac{N}{\mathrm{count}(d : t \in d)}\right) \tag{2.4}$$

The TF-IDF score for a term t in a document d is then given by:

$$\text{TF-IDF}(t, d) = \mathrm{TF}(t, d) \times \mathrm{IDF}(t) \tag{2.5}$$

It is evident that the above calculation puts more emphasis on informative words and less on common ones[1, 13].

2.1.2.6 Word Embeddings

Word embeddings are dense vector representations of words that tries to capture the semantic relationships between them and their meaning in context. There are many methods, such as Word2Vec and GloVe, which represent the words into vector spaces where words that are close semantically have similar representations. For example, Word2Vec's Skip-gram model, when provided with a word w_t, can predict surrounding words within a context of window size c:

$$P(w_{t+c} \mid w_t) \tag{2.6}$$

[13, 14].

2.1.3 Traditional Machine Learning Models

Provided that the text data has been translated into numerical representations through methods like BoW, TF-IDF, word embeddings etc., traditional machine learning algorithms can take these representations as input and use them for classification tasks [2, 3, 8]:

- **Naive Bayes:** This is a simple and commonly used probabilistic classifier that is based on the Bayes' theorem:

$$P(C \mid X) = \frac{P(X \mid C)P(C)}{P(X)} \tag{2.7}$$

  where C corresponds to the class labels and X represents feature vectors [15]. Conceptually, Bayes' theorem describes how the posterior probability of class label C given the text X $P(C \mid X)$ can be computed as the product of the likelihood $P(X \mid C)$ and the prior probability $P(C)$, normalized by the evidence $P(X)$. All these terms can be computed by measuring word frequencies in the dataset and applying the corresponding calculations. Some common applications of Naive Bayes include spam detection, sentiment analysis, document classification etc.

- **Support Vector Machines (SVMs):** Support Vector Machines are another common type of supervised learning models, that aim to find the optimal hyperplane to distinguish data points which belong to different classes. The decision function of a SVM can be expressed as:

$$f(x) = w^\top x + b \tag{2.8}$$

  where w is the weight vector, b is the bias term, and x is the input vector. The objective of a SVM is to maximize the distance between this hyperplane that it tries to calculate and the nearest data points from each class (support vectors) [16]. SVMs also uses a technique called the *kernel trick*, which enables them to discover the hyperplane in a higher dimension space, without having to

actually compute the transformations for the data points, and this technique makes the SVMs very powerful, since they can learn to distinguish the classes even in non-linear feature spaces. However, SVMs are also computationally expensive and need careful parameter tuning.

- **Logistic Regression:** Logistic regression is a machine and statistical learning model used for binary classification, and it is especially useful when the objective is to predict the probability of an instance belonging to a specific class. This classifier is based on the sigmoid (logistic) function in order to map the linear combination of the input variables (features) to a probability:

$$P(Y = 1 \mid X) = \frac{1}{1 + e^{-(w^\top X + b)}} \tag{2.9}$$

where w is the weight vector, X is the feature vector, and b is the bias term.

Logistic regression provides as output a probability, thus the outcome can only have a value between 0 and 1. It has very good performance when the data is linearly separable, and it serves as a strong baseline for text classification tasks when text representation techniques like Term Frequency-Inverse Document Frequency (TF-IDF) are used [17]. In this study, logistic regression was one of the traditional ML models that were employed to serve as baseline models in the task of predicting the loan arrangement probabilities.

- **XGBoost:** XGBoost (Extreme Gradient Boosting) is an implementation of the gradient boosting algorithm that is both efficient and scalable. It operates by building an ensemble of decision trees in sequence, each of which corrects errors that were made by the previously trained trees. The loss function of XGBoost is the following:

$$L(\theta) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{j=1}^{m} \Omega(f_j) \tag{2.10}$$

where l is the loss function, y_i is the true label, $\hat{y}_i$ is the predicted label, f_j represents the individual trees, and $\Omega(f_j)$ is a regularization term that is used in order to control the complexity of the model [18].

XGBoost adopted several innovations and breakthroughs to achieve its performance and scalability for large datasets while avoiding overfitting, such as weighted quantile sketch in order to handle sparse data, and parallelized tree construction, in combination with regularization mechanisms. All these make XGBoost a particularly popular choice for regression, classification and ranking tasks.

- **Random Forest:** Random Forests is an ensemble method that creates several decision trees during the training for a classification task, and then it outputs as the final prediction the class that has been voted by the most trees. To construct every tree, the algorithm chooses a random subset of features and examples, which contributes to greater generalization and reduces overfitting, leading to a more robust model. The final prediction can be expressed as:

$$\hat{y} = \text{mode}(f_1(x), f_2(x), \ldots, f_k(x)) \tag{2.11}$$

where $f_k(x)$ represents the prediction from the k^{th} tree for input x [19].

While traditional machine learning models are effective for structured data and have long been studied, they frequently struggle with complex patterns found in unstructured text data and heavily rely on the effectiveness of feature extraction techniques like the ones discussed above (BoW, TF-IDF, n-grams) [2]. For this reason, it is common to use deep learning models for more complex tasks, which are very capable at automatically learning representations from text data. Models like Recurrent Neural Networks (RNNs), Long Short-Term Memory Networks (LSTMs), and Transformers are more powerful in recognizing dependencies and patterns in text, and they often display top performance at tasks such as text classification, sentiment analysis, and machine translation [12].

2.1.4 Advanced Deep Learning Models

2.1.4.1 Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) are a neural network architecture that is designed to receive sequential data by processing them step by step, while keeping in memory previous inputs through hidden states. Thus, unlike traditional feed-forward neural networks, RNNs take into consideration time-related dependencies in the data. This makes RNNs a very commonly-used architecture for tasks that involve sequential data, such as time series, speech, and of course text [20, 21].

The mathematical formula that expresses the update of the hidden state at time step t is given by:

$$h_t = f(W_h h_{t-1} + W_x x_t + b) \tag{2.12}$$

where h_t is the hidden state at time t, x_t is the input at time t, W_h and W_x are weight matrices, and b is a bias vector. The function f is usually a tanh or ReLU activation function.

Regarding the text classification applications, RNNs process the sentences or documents word by word, and they encode the meaning of each word in the context by using the hidden states. At the final step, the hidden state h_T essentially sums up the entire sequence, and this state is then passed to a fully connected layer followed by either a softmax or a sigmoid activation function for the classification task.[22].

While RNNs exhibit good performance at modeling short-term dependencies, they face several difficulties when trying to encode long-term dependencies due to what is known as the vanishing gradient problem, which makes training for longer sentences (or sequences in general) unstable[23].

2.1.4.2 Long Short-Term Memory Networks (LSTMs)

Long Short-Term Memory networks (LSTMs) are a subcategory of RNNs specifically designed to address the limitations with long sequences of the traditional RNNs,

which is due to the vanishing gradient problem as described above. By making use of the gating mechanism, LSTMs control the way that the information flows in the network, practically deciding which of the information to keep, what to disregard, or which output it needs to deliver at each time step. This functionality enables these models to keep track of both short- and long-term dependencies, making them very useful for datasets that include longer sequences [24, 25].

The gating mechanism of an LSTM cell can be described using the following equations:

- **Forget Gate**:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{2.13}$$

  The forget gate decides which information to discard from the previous cell state.

- **Input Gate**:

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{2.14}$$

  The input gate regulates which information to add to the cell state.

- **Candidate Cell State**:

$$\tilde{C}_t = \tanh(W_c[h_{t-1}, x_t] + b_c) \tag{2.15}$$

  It proposes a new candidate cell state based on the input and previous hidden state.

- **Cell State Update**:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{2.16}$$

  It updates the cell state by combining the forget gate's and the input gate's outputs.

- **Output Gate**:

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{2.17}$$

It controls which information from the cell state to output.

- **Final Hidden State**:

$$h_t = o_t * \tanh(C_t) \tag{2.18}$$

It outputs the updated hidden state, which encompasses the summary of the whole sequence.

In binary text classification tasks, LSTMs operate in a similar way to RNNs, but with greater performance gains in longer sentences. The final hidden state is passed through a fully connected layer with a sigmoid activation function to enable the prediction of the probability that the input text belongs to one of the two classes (0 or 1) [22, 26].

2.1.4.3 Transformer Models

Transformers have in many ways transformed NLP applications by making the parallel processing of the input texts (or sequences in general) feasible through self-attention mechanisms. In this way they can handle the constraints that arise in the recurrent neural networks [27]. In contrast to their predecessors, the transformers models can take into account global dependencies in the sequences, thus becoming valuable for a wide variety of NLP applications, including of course text classification.

The core idea behind the transformers architecture is the self-attention mechanism, which permits the model to calculate the significance of the words that exist in a text in relation to one another. The mathematical formulation that describes the attention mechanism is:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \tag{2.19}$$

where Q represents the query matrices, K the key matrices, and V the value matrices, respectively. These are derived from the input embeddings, while d_k is the dimension of the key vectors. To ensure numerical stability and to prevent extremely large values during the softmax calculation, we divide by $\sqrt{d_k}$.

Transformers are also based on the encoder-decoder architecture:

- The **encoder** turns the input sequences into representations that maintain context information, using many layers of self-attention and feed-forward neural networks.
- The **decoder** gives the output sequences by taking the encoder's representations and applying layers of self-attention too.

For classification tasks, it is common to use only the encoder part of the model. The final hidden state that is related with a special token (like the [CLS] in BERT) plays an important role in describing the entire input sequence and it is fed to a fully connected layer, which is finally followed by a sigmoid or softmax activation for the prediction task [28].

2.1.4.4 Pre-trained Transformer Models

A common methodology that is used in NLP nowadays is taking pre-trained transformers as base models and further fine-tuning them for specific tasks, because the base models are often trained on huge bodies of texts, and for this reason they have an enormous internal knowledge. This allows them to be adapted for specific tasks like binary text classification, requiring minimal labeled data (few-shot classification) [28].

2.1.4.4.1 Greek Language Models - GreekBERT and Meltemi: GreekBERT and Meltemi are among the models that have been used in this study, because they are pre-trained models that are specialized for Greek text, a fact that makes them very good candidates for the binary classification task in hand. A brief description for these models can be found below:

- **GreekBERT**: This model is a transformer-based language model, which is pre-trained on a large dataset of Greek texts and as the name suggests, it utilizes the underlying BERT architecture, which consists of 12 transformer layers with 12 self-attention heads per each layer. The pre-training process of this model tries to optimize the masked language modeling and the next-sentence prediction [29]. GreekBERT can also be used for binary text classification, because after processing the Greek texts, its final [CLS] token can be passed to a classification head.

- **Meltemi (Mistral 7B)**: This is a large language model based on the Mistral architecture, which has 7 billion parameters. Meltemi is also fine-tuned for Greek and takes advantage of several advancements in the transformer architectures to be able to work efficiently with complex language structures and longer contexts [30]. The larger number of parameters enables it to catch more details and stronger relationships in the Greek texts, making it potentially valuable for tasks like text classification.

2.1.4.4.2 Other Models that have been Used in This Study: Besides the Greek-specific language models, this study experiments with several other state-of-the-art language models:

- **FinBERT**: This is a BERT-based model that is pre-trained on financial texts and has achieved high performance on tasks that require understanding of domain-specific financial terminology [31].

- **XLM-RoBERTa (xlm-roberta-base)**: This is a multilingual model trained on 100+ languages that excels in cross-lingual transfer learning and for this reason, it was used to test its performance on the Greek language. [32].

- **Multilingual-e5-large**: A multilingual embedding model optimized for semantic similarity tasks. Its usefulness for understanding context in greek loan arrangements is examined as part of the experiments.

- **Llama 3.1 (70B)**: A state-of-the-art LLM with 70 billion parameters than

can perform complex reasoning tasks in multiple languages, including Greek [33].

2.1.4.4.3 Application to Binary Text Classification Transformers are widely used, as it has already been stated, in binary text classification tasks, thanks to their ability to understand the contextual meaning of every word in the sentence based on its position. Especially for the objective of the current study, which is to predict the probability of whether a Greek loan arrangement is likely to happen, transformers can potentially achieve high performance in analyzing the texts and converting them into meaningful representations, which can express domain-specific details such as financial terms as well as long-range dependencies, both of vital importance to make accurate predictions.

Regarding the GreekBERT and Meltemi models specifically, their pre-training in Greek texts offers an even better understanding of the several special characteristics of the language, while FinBERT's specialization on financial data can also help with capturing domain-specific patterns. These models, when fine-tuned on the labeled binary classification data of this study, can probably achieve higher performance by taking advantage of both linguistic and task-specific features.

2.1.5 Addressing Class Imbalance

Class imbalance is a problem that occurs very often in many real-world binary classification applications, like the current one, where the negative class has significantly more examples than the positive. This imbalance makes the learning of the minority class harder and can severely affect the performance of the model, especially in problems like e.g. fraud detection, where the positive class of interest is typically the minority class and has very few instances, thus eventually leading to biased models that tend to favor the majority class. Several approaches have been developed to handle this situation:

2.1.5.1 Contrastive Learning

Contrastive learning is a methodology that focus on the differences between the classes, by feeding the model with similar (positive - same label) and dissimilar (negative - opposite label) pairs of examples and using a contrastive loss. Essentially, by making use of these pairs that have semantic similarities or not, the model tries to bring the similar embeddings (positive pairs) closer in the latent space, while it tries to push the negative pairs away.

Contrastive learning is especially useful when the datasets are imbalanced, because it can increase the importance of the minority class and the role it plays in the learning process. As an example, the SimCLR framework used this technique to maximize the alignment between augmented views of the same instance [34]. The contrastive learning process is described in greater detail in the experimental section of this study.

2.1.5.2 Data Augmentation

Data augmentation operates by changing existing training examples in order to generate more training data, so that the class imbalance that exists in many datasets is decreased and a variety of new training samples is introduced. Some of the most common text augmentation techniques include:

- **Back-translation**: Translating a training text example into another language and then translating back into the original one, while trying to maintain the meaning [35].

- **Synonym Replacement**: Replacing words with their synonyms, making sure that the semantic meaning of the sentence is preserved [36].

- **Paraphrasing**: Using pre-trained language models to generate various paraphrased samples from the existing training data (such as the T5 model by Google, the Pegasus Paraphrase and other) [37] [38] [39].

2.1.5.3 Synthetic Positive Examples

Generating synthetic positive examples is in many real-life scenarios a common feasible option, when it is hard to collect more examples from the minority class. Techniques to achieve this purpose include Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs) and most frequently LLMs. All these types of models are used to create new training examples based on patterns that they have recognized from the existing data[**?**].

In this MSc study for instance, a LLM has been used to generate new positive examples using the existing ones, an approach which, as it is described in the experiments section below, played a very important role in the effectiveness and the generalization of the model.

2.1.5.4 Loss Function Adjustment

By adjusting the loss function that is used to optimize the parameters of the model, the learning process can pay much more attention to the minority class. Popular approaches for this in a binary classification problem include:

- **Class-Weighted Cross-Entropy Loss**: This technique is used to assign higher weights to the minority class, which are most of the times inversely proportional to the classes ratios. In this way, they increase the minority class' impact on the calculations for the gradient updates.

- **Focal Loss**: Focal loss concentrate on harder examples and does not let the model become overwhelmed by the easier ones, which can greatly enhance the achievable performance on imbalanced datasets [40].

These loss adjustments can play a major role in ensuring that the model learns to identify the minority class and dealing with the underlying dataset's imbalance effectively.

2.1.6 Fine-Tuning Large Language Models

Large language models (LLMs), such as BERT, Mistral, LLama etc. have transformed the NLP applications landscape by incorporating knowledge from huge text corpora, pushing in this way the performance limits forward, while also re-defining the state-of-the-art. For this reason, fine-tuning these models for specific classification tasks is of high importance for anyone who wishes to achieve optimal results. Recent advancements such as LoRA (Low-Rank Adaptation) and QLoRA (Quantized Low-Rank Adaptation) have enabled, aside from big corporations with huge amounts of computing and energy resources, also small- and mid-range users to benefit from the fine-tuning of these models in very efficient ways. These methods are further described below:

2.1.6.1 LoRA: Low-Rank Adaptation

LoRA is a technique that inserts rank decomposition matrices into pre-trained transformer layers, while freezing the already trained model's weights, thus allowing a task-specific adaptation using much less parameters and requiring a much smaller fraction of the GPU power that would have otherwise been needed if the whole model was fine-tuned. Instead, during the training process, only the low-rank matrices are updated, which enables the users with fewer computing resources to use LLMs on their applications [41].

In binary text classification tasks that involve Greek language specifically, LoRA enables efficient and fast fine-tuning of models like GreekBERT or Meltemi, while maintaining the same, or even better performance.

QLoRA: Quantized Low-Rank Adaptation

QLoRA is essentially an extension of LoRA, by using quantization techniques in order to further reduce memory and computational requirements. In more details, QLoRA uses 4-bit quantization for the LLMs weights by making use of the normal distribution that the weights follow in order to retain as much information as possible with the minimum loss of precision. QLoRA also drastically reduces the computation

overhead for fine-tuning the LLMs through low-rank adapters. This method has been proven to achieve almost the same accuracy as full-precision fine-tuning [42].

QLoRA is valuable for fine-tuning models like Llama 3.1 (70B) or even bigger ones without excessive or prohibitive computational needs. When combined with techniques to deal with class imbalance in the dataset, both LoRA and QLoRA have the potential to create a model that can generalize very well and has learned even the smaller details of the seen data, even when they contain lots of specialized terminology and domain knowledge.

2.2 Related Work

2.2.1 Introduction

Binary text classification, and especially the application of predicting customer actions using call center transcripts, is associated with several difficulties and challenges, even for a task that seems straightforward at first glance, and for which so much work has been done in the relevant literature. This is especially true when working with datasets that are highly unbalanced in terms of class distribution, with such imbalances being very common in call center environments as the dynamics of customer interactions tend to favor certain categories over others, for example, there are many more customers who are likely to stay than those who are likely to churn.

When someone has to work with imbalanced datasets text classification becomes a complicated task, and the class imbalance can severely affect the models performance and make them less capable of effective generalization. It has been shown that when some of the classes have much more samples than the others, the models become significantly more biased towards the majority class [43–45], and therefore, it becomes difficult for a model to generalize and understand the patterns of the minority class, thereby leading to wrong classification or poor prediction [43, 45]. The problem of class imbalance is even more important in the presence of noisy data that can make the classification task even more challenging [46]. Noisy data are data that are incomplete, inaccurate or contain other types of errors, and as we will examine also with the dataset of the current study below, they can make it even

harder to deal with imbalanced datasets [44].

It is really important and tough to tackle the problems that are caused by imbalanced datasets and noisy data in the field of text classification when working with the Greek language, which is well known for its extensive morphology as well as its syntactic complexity, both of which pose various difficulties in the text classification process. Some of these challenges include but are not limited to the presence of unknown words, the variety of the sentence structures, strict grammatical rules that should be followed, and many other factors that eventually make the classification harder. Moreover, the lack and rarity of appropriate natural language processing (NLP) resources that exist specifically for the Greek language can also further contribute to these challenges [47].

In view of the above, the primary goal of this literature review is to identify the challenges that are related with the issues of imbalance and noise in the context of Greek text classification that is done by language models on call transcripts. From the existing literature and the examination of several methods, it is possible to highlight the main goals and objectives of this field as well as the accompanying challenges and potential solutions.

2.2.2 The Challenge of Imbalanced Data

An imbalanced dataset is a major problem in the application of binary text classification especially when the goal is to predict customer actions based on call center transcripts, since this skew in class distribution forces the classifiers to learn more about the majority class, thus compromising the accuracy of the minority class. This problem is even more intense in situations like customer churn prediction where the instances of customers who are likely to churn are much fewer than those who are not. Li et al. [45] have performed experiments on text classification using an imbalanced dataset and a class spatial model, and their results showed that the unbalanced training dataset led to a poor classification, because the classifier had an obvious preference for the majority class, a result which clearly shows the problems that arise when working with imbalanced data in text classification and the importance

of using appropriate methods to solve it.

2.2.2.1 Data-Level Approaches

A certain number of strategies have been proposed to handle the issues of unbalanced data, among which are data-level approaches that involves manipulating the training data to achieve balanced class distributions.

For instance, Thabtah [48] examines different oversampling techniques such as random oversampling and focused oversampling to achieve the required balance of the dataset, while another study shows that one of the most effective techniques is the Synthetic Minority Oversampling Technique (SMOTE) where synthetic examples are created using the feature space of the minority class [49].

Shaikh et al. [44] follow an even more novel appoach from the previous work by highlighting the usefulness of deep neural language models such as GPT-2 and LSTM models for the generation of synthetic data, so that they can enhance the classification performance on imbalanced text datasets. This research proves that bringing balance to datasets can greatly improve the evaluation results in scenarios where the minority class is under-represented without enough data, hence the use of synthetic data generation methods can assist in expanding the smaller class and presenting the classifiers with more examples that are close to real world data [44].

Besides synthetic data generation, data augmentation is another commonly used approach which can be very effective in taking care of class imbalance problems, by creating new artificial data samples based on the existing ones with the purpose of expanding the initial dataset. Bayer and his colleagues [50] give in their survey a detailed account of current data augmentation techniques for text classification, while they classify these methods into two main categories: data space and feature space. Data space methods include using the raw text data, while feature space methods involve manipulating the feature representations of the text data. The researchers also state that since there are large pre-trained language models which are publicly available, there is a need to conduct more research on how to combine data augmentation with these language models effectively.

Wei and Zou [36] suggest four basic yet useful data augmentation techniques for text classification which are: Synonym Replacement (SR), in which the words are changed into their corresponding synonyms; Random Insertion (RI), which involves the inclusion of synonyms into random locations of a sentence; Random Swap (RS), which is a process of swaping the positions of two words within a sentence and Random Deletion (RD), which is about removing words from a sentence, optionally. The authors show that these techniques can increase the performance of text classification models by a lot, especially when the available training data is limited.

Last, Moreo et al. [51] introduce a new data augmentation method which is called Distributional Random Oversampling (DRO) and is basically a tool used to solve the problem of data imbalance by creating synthetic data for the less frequent class by following the distribution of this class. The authors prove that DRO can make the text classification models more capable, particularly when they are fed with imbalanced data sets, and they also demonstrate that DRO is better than other oversampling techniques, especially in the case of very rare classes, and across different levels of oversampling and different datasets [51].

2.2.2.2 Algorithm-Level Approaches

Among the strategies that belong to the category of algorithm-level approaches is changing the learning algorithm itself so that it can deal with the problem of class imbalance. Chawla et al. [52] for example argue that weighting false negatives and false positives with different cost factors can strengthen the model's predictive power especially when dealing with rare classes, but it should still be noted that the cost matrix can be subjective or difficult to compute in practice, and the process of adjusting costs may become an additional computational burden, especially when dealing with large datasets [52].

Other techniques like boosting and bagging together with oversampling can also be used for making the classification performance better when working with imbalanced datasets, like for example SMOTEBoost, which is an ensemble learning algorithm that integrates SMOTE with boosting [53, 54].

In Liu et al.'s work [43], the challenge of imbalanced text classification was solved using a unique approach to term weighting which is based on probability and employs the ratios $\frac{A}{B}$ and $\frac{A}{C}$, where A stands for the number of documents in the given category which include the term, B represents the number of documents that do not belong to the category and include the term and C is the number of documents in the category that do not contain the term. This approach weighs more on terms that are frequent in the documents in a particular category, and thus enhances the classification results [43].

2.2.2.3 Impact of Text Data Characteristics

It has been shown that the characteristics of the text data itself can have a big influence on the success of imbalance-handling techniques. The paper by Lu et al. [55] compares different deep learning models and shows that increasing the number of samples in the minority class can enhance the performance of the model, while also revealing that in general, deep learning models tend to produce better results than the traditional machine learning algorithms, especially when the data is balanced and although their performance suffers when the data is unbalanced. In terms of the text characteristics, Lu et al. highlight that the length of the text sequences affects the model's capability, due to the fact that having very short sequences may remove important information, while very long sequences may add noise and expand the training time [55]. All these highlight the importance of text pre-processing and feature extraction process to enable the classifiers to learn from the data more effectively.

Li et al. [45] analyze further how specific traits of text data have an impact on the effectiveness of the techniques for dealing with imbalanced data. The study emphasizes the role of feature selection and dimensionality reduction in helping the model to learn to distinguish between classes especially when working with extremely skewed datasets.

2.2.2.4 Evaluation Metrics

Another important decision is the way to assess classifiers which have been trained on imbalanced data sets. The traditional overall accuracy, which would in many cases be an adequate performance measure, can now be easily influenced by the majority class. Therefore, it is common for class imbalance situations to use other metrics, like for instance F1-score, precision and recall, to gain a better understanding of the classifiers performance as suggested in [56]. Another metric that is usually used is the Precision-Recall curve, which is also very helpful in settings where the positive class is the minority one. It is important to note that working with the Receiver Operating Characteristic (ROC) curve can provide an overly optimistic view of the classifier performance in imbalanced datasets, whereas the PR-AUC focuses on the trade-off between the precision and the recall for the positive class as stated in [57]. From all the above, it is easily understood that it is therefore important to select the most appropriate evaluation metrics for the specific applications such as medical diagnosis and to some extent like our particular study for predicting the loan arrangement probability, where the cost of missing a minority class instance is not negligible [56].

2.2.2.5 Concluding Remarks

In summary, effective techniques to handle class imbalance in text classification tasks can be either data-level or algorithm-level approaches, and different domains and applications make use of different methods in order to get good results based on appropriate evaluation metrics. All these considerations are highly demanding in the development of stable and accurate classifiers and they are also particularly important in areas such as customer churn modeling, fraud detection etc., where it is really relevant and highly valuable from a business perspective to predict the behavior of the minority class.

2.2.3 Contrastive Learning on Text Classification

Research has shown multiple times that contrastive learning can potentially be a quite effective approach for learning the most important patterns and characteristics of each class in order to improve the performance of the models that are used in various different text classification tasks. For this reason, several recent studies have examined potential usages of contrastive learning to strengthen the performance of text classification models, like for example Su et al. [58] who employed contrastive learning for improving the performance of BERT models especially in the domain of biomedical relations discovery and to this end, they proposed a contrastive method that introduces linguistic knowledge into the data augmentation process so that they could take advantage of knowledge bases to generate large-scale datasets. This approach has been shown to achieve state of the art performance on many different bio-medical benchmarks, by improving the semantic representation and the model interpretability.

Another study also in the medical domain was carried out by Min and colleagues [59], who made use of ChatGPT-generated pseudo-labels and contrastive loss to fine-tune BERT for semantic similarity tasks in medical information processing, so that they could improve in this way the semantic representation of medical terminologies and provide better results even in low-resource environments, by minimizing the distance between similar terms and maximizing the distance between dissimilar terms.

Chen et al. [60] introduced ContrastNet, a really interesting approach for few-shot text classification based on contrastive learning, which aimed at learning text representations while also preventing the overfitting of the model at both the task and the instance levels. This proposed method used supervised contrastive learning to express similar texts with similar vectors and different texts with different vectors, which is particularly valuable in few-shot learning where there are often not enough labeled data.

Chen et al. [61] even went one step further by creating a dual contrastive learning (DualCL) framework, where feature representations of input samples and the

learnable parameters of classifiers are learned at the same time and in the same space, because according to their view, the classifier parameters are considered as augmented samples which are associated with various labels, and contrastive learning is employed between the input samples and these augmented samples. This methodology was put into test by using five benchmark text classification datasets and the results were really good, as they showed that classification accuracy had increased significantly, especially in low resource settings. [61]

Last but certainly not least, Pan et al. [62] proposed Contrastive Adversarial Training (CAT) to make better both the robustness and the performance of Transformer-based text classification models, by adopting a combination of adversarial training together with contrastive learning, where adversarial examples are created using word embedding perturbation and the model is trained on both the adversarial and the clean instances. This approach showed an average enhancement of 1-2% over the baselines such as BERT and RoBERTa on GLUE benchmarks and intent classification tasks.

All these studies have clearly shown that contrastive learning is really effective and an also very versatile approach that can be used to enhance the performance of text classification models across many different applications, domains and tasks, and even in situations with extremely unbalanced scenarios, like the current study. When combined with other methods such as adversarial training and data augmentation, it can potentially push the boundaries of text classification performance to a significant degree.

2.2.4 Noisy Data on Text Classification

Many text classification models and tasks, including the one in this study, have only noisy data available and this can be a major obstacle for model performance and reliability, hence for this reason several researchers have investigated different aspects of this problem and have suggested various solutions to handle the noisiness of data. Dönicke et al. [63] studied the challenges when working with imbalanced, sparse and noisy data, especially on the songtext-artist classification use case, and

for this work they considered three types of neural network methods: a one-layer perceptron, a two-layer Doc2Vec model and a multilayer perceptron. Their research revealed that a lot of benefits can be achieved by pre-processing the data, so that it becomes more balanced and less sparse. However they also noted that this approach, while useful, is only a form of heuristic and does not guarantee optimal solutions, and due to this fact they concluded that their work could be enriched in the future to include clustering-based topic models encoded as new features in order to perhaps further enhance the performance.

In the work done by Agarwal et al. [64], the authors studied the effects of different noise types on text classification and for this they considered different kinds of noise such as OCR noise, ASR noise and human noise which may occur in informal communication. Their experiments showed that the text classification algorithms were really robust to a high level of typographical noise or noise introduced by ASR, although eventually they reached to the conclusion that noise in the training data results in the development of poorer models and lower levels of accuracy as opposed to the presence of noise in the test data while keeping the training data clean.

Zhu et al. [65] studied BERT's performance in the presence of label noise for text classification problems. The study revealed that BERT is quite effective with handling the noise that was injected during the training process, however, it can be particularly sensitive to noise that is caused by weak supervision, which is basically when labels are constructed automatically with methods that are faster and cheaper than hand-labeling by experts, but eventually may cause inconsistencies, errors and therefore noise in the labels. They also discovered that most of the noise-handling strategies used thus far were not so effective in enhancing BERT's performance and, in some cases, they even worsened it, while they also focused on the weak supervision noise, which even at small amounts can be harder to be dealt with than high levels of injected noise. A disadvantage that they pointed out was the fact that their studies involved only a few types of noise and intensity levels.

Liu et al. in their paper [46] created the first benchmark for noise learning in text classification and they pointed out several issues that existed in previous studies, primarily because they noticed that the majority of the datasets used in

the prior works were noisy (having intrinsic noise of 0.61% to 15.77%) which could lead to wrong predictions. In order to handle these challenges, they came up with a new dataset called Golden-Chnsenticorp with no intrinsic noise, and they conducted extensive experiments which revealed that most of the methods do not perform well if the noise ratio is greater than 30%. However, what was even more surprising was the fact that small amounts of white noise can actually make the classification techniques stronger, and due to all these findings, their main proposal for future work was that other researchers should concentrate on the creation of better approaches for coping with noise types that are more complex and situations where the noise ratio is high.

Last, Venkata Subramaniam et al. [66] also conducted a survey on various kinds of noise that can be present in texts as well as different approaches to deal with noisy text data for several tasks such as Information Retrieval and Information Extraction, and for their work they divided noisy text sources into two main categories: noise that is generated during a conversion process (for example during OCR, ASR, MT) and noise that is generated when text created in a digital environment and in an informal manner, for example in chats, SMS, emails etc. The authors broke down the different ways of how noise appears in each kind of source and how it could actually be measured. For instance, in SMS data they discovered recurrent patterns such as character deletion, phonetic substitution, shortening of words to their initial letters, abbreviations, and many other language informalities. In terms of the noise measurement metrics, they discussed several methodologies like Word Error Rate (WER), Sentence Error Rate (SER), and perplexity. As for the challenges, they highlighted the fact that that what constitutes noise and what not is quite subjective - or more precisely, it depends on the system under evaluation: for example, texts that are 100% fine for a human to comprehend may be classified as noisy for a certain automatic processing system. The authors finally suggested that further research should be directed towards the creation of specialized techniques for dealing with different kinds of noise according to their type and origin [66].

To summarize, a frequently met challenge that was noted in many research papers was that despite the belief that the benchmark datasets were noise-free, they

actually had varying levels of intrinsic noise. This situation makes clear the need to improve on the current dataset cleaning and curation and on the noise identification techniques. Also, most of the studies focused on the addition of artificial noise instead of real life noise which can be found in the text data as well, which paves the road towards the direction that future work should follow.

2.2.5 Text Classification on the Greek Language

In terms of applications of Text Classification in the Greek language, the researchers have dealt with many difficulties because of the specific characteristics of the language and the scarcity of resources in comparison with more frequent languages such as English, and for these reasons, several notable studies have been conducted to investigate various solutions: Bilianos [67] focused on sentiment analysis of reviews written in Greek about electronic products, using a dataset of 480 reviews collected from an e-commerce website, and he employed a variety of models that ranged from the basic Naive Bayes classifier with n-gram features to the advanced Greek-BERT. Even if the dataset was very small, BERT managed to provide excellent results with 97% accuracy, but even so the study pointed out some of the challenges that was caused by the small dataset and suggested that if more data was collected, this would increase the confidence in the results, but it would also take a considerable amount of time. The author also proposed that it would be interesting and potentially useful to investigate how aspect-based sentiment analysis could be used for more granular outcomes.

Alexandridis et al. [68] examined the sentiment analysis and opinion mining in Greek social media using various language models, classifiers, and around 60,000 social media texts. They discovered that classifiers based on language models such as Naive Bayes, Random Forests, SVMs, logistic regression and neural networks performed better than the word or sentence embedding approaches with an accuracy of over 80%. The authors also trained a new Greek social media language model that increased the performance by 2% when compared with the existing more generic models. The main disadvantage that was highlighted by the authors was the relative lack of quality data sources for Greek NLP tasks.

Vamvourellis [69] put effort in comparing the performance of various BERT models for the task of classifying Greek legal documents, and he used the RAPTARCHIS dataset which consists of Greek legislation documents that range chronologically from 1834 to 2015. For the purposes of this study three BERT models namely, Greek-BERT, Greek-Legal-BERT and M-BERT, were finetuned, and out of the three models, Greek-Legal-BERT - which was trained on Greek legal language - provided the best results while M-BERT gave the worst results. Based on these results, the author proposed that domain and task adaptive pre-training would be useful for enhancing the accuracy of all models, like M-BERT which gave inferior results compared with Greek-Legal-BERT. The author also assumed as a future work objective that should the vocabulary be adjusted and with proper training, multilingual models like M-Bert could be even better than the Greek-only models that were proposed.

Papaloukas [47] employed a similar approach and conducted a detailed experimental analysis on the performance of various machine learning models on the task of legal text classification, while also using the RAPTARCHIS dataset, which as stated above consists of 47,000 legislative documents over a period of almost 200 years. The study aimed at comparing traditional machine learning, recurrent neural network models and the latest transfer learning techniques, and the outcomes indicated that the traditional ML classifiers performed worse than the state-of-the-art methods, although they could serve as a good initial baseline. A very interesting conclusion was that the recurrent neural network architectures, when using domain-specific word embeddings achieved better performance and were almost on par with the transformer-based models. It is worth mentioning that the study noted that at the time of its writing it was the first systematic analysis of Greek legal text classification and the author recommended that this work could form a basis for further future studies.

To summarize, all these studies discussed several limitations of the Greek text classification applications, including but not limited to the lack of resources and the unique characteristics of the Greek language. Nevertheless, all these studies also brought into attention the growing effectiveness and performance of text classifica-

tion methods across different domains, even when dealing with Greek texts. It is beyond any doubt that future work would be really important in order to advance the state of the art in text classification for the Greek language and broaden the availability of datasets and language models.

2.2.6 Text Classification Applications in Call Center Analytics

The data available in call centers are usually unstructured in nature, due to the fact that they often involve transcribed conversations between agents and customers, but even so they still carry a large amount of valuable information, which can potentially be transformed into advantageous business insights. Therefore, several researchers have attempted to discover ways to utilize this data for different classification tasks. Zhong and Li [70] proposed a CNN-based model for the classification of customer call intent into four categories, namely sales, service, vendor, and job seeker, based on the transcripts of the calls made to the auto dealerships, and they also carried out an experiment to determine whether it makes a difference if someone analyzes both the caller and agent channels, or if it would be sufficient to use only the caller channel. They concluded that analyzing only the caller channel transcripts could assisting in reducing the data volume while at the same time significantly retaining the classification performance. Their CNN model paired with word embeddings gave good results, as long as enough training data were provided. Finally, they emphasized an important factor for the successful training of a capable model, which is the scalability of the data labeling methodology, because a major challenge for the authors was the fact that a lot of data had to be labeled by hand.

Vo et al. [71] used call transcripts to predict customer churn with a primary focus on the financial services sector. For this, they combined structured customer profile data with unstructured text features that were extracted from the call transcripts using NLP methods such as the term importance (TF-IDF), Word2Vec embeddings, lexical information (LIWC), and personality traits. Their multi-stacking ensemble model proved that when someone combines unstructured text data with existing

structured data, they can significantly improve the efficiency of churn prediction, although they also noted many issues that they faced while gathering and storing the call transcripts data, like the size of the data, customer privacy considerations and other.

Low-cost call type classification was the primary research objective of Park et al. [72], who used partial transcripts of calls instead of the entire calls, based on the assumption that the type of the call could probably be identified by the first few words of the discussion or based on the agent's statements only. In order to test this hypothesis they used texts from an automotive company's calls and compared the classification accuracy when using the full calls versus the partial ones, like limiting to 10-40 words, using agent-only or customer-only words etc. The results demonstrated that the agent's phrases alone were enough in order to manage to achieve the same or even better results than the full transcripts.

Last but not least, Ezzat et al. [73] attempted to perform sentiment analysis of call center calls, and for this goal they first transformed the audio files to text using speech recognition, so that they could use them for the text classification task. They examined several different feature selection techniques and assessed supervised learning algorithms including SVM, Naive Bayes and Decision Trees. The authors were able to get quite good results, especially considering the low quality transcription, (94.4% accuracy), but still, they highlighted several challenges such as the requirement for more capable speech recognition and speaker segmentation. As potential subjects for future work, they proposed investigating the combination of text classification with acoustic feature analysis.

In summary, applying text classification to call transcripts could be highly effective for a variety of tasks, like intent prediction, churn prediction, sentiment analysis, etc., but while this methodology undoubtedly provides valuable information for businesses, future researchers have still a lot of work to do on issues such as data privacy, effective ways of dealing with big unstructured data, and proper conversion of spoken language into text.

Chapter 3

Proposed method

3.1 Problem Definition

The purpose of this thesis is to predict the probability of a loan arrangement based on communications that call center agents have performed with the account holders of this particular loan. The problem is formulated as a binary text classification task, and the unstructured data consist of comments that have been written by the agents, while discussing with the owners of the nonperforming loans.

Mathematically, the problem can be defined as follows:

- **Input:** Let $X = \{x_1, x_2, ..., x_n\}$ represent the set of n call center transcripts based on the comments of the agents, where each x_i is a sequence of words representing all communications that were done with a customer in 3 consecutive calendar months.

- **Output:** Let $Y = \{y_1, y_2, ..., y_n\}$ represent the set of corresponding labels, where $y_i \in \{0, 1\}$.

  - $y_i = 1$ corresponds to the positive class of interest and indicates that the customer associated with transcript x_i successfully arranged their loan and made a payment within the next 2 months from the last of the three calendar months that are associated with x_i.

  - $y_i = 0$ (negative class) shows that the customer did not pay an installment

within the specified time window, thus revealing that no loan arrangement was achieved.

- **Task:** The goal is to learn a function $f : X \rightarrow Y$, so that all call center communication texts can accurately predict the outcomes of the loan arrangement.

  This objective is particularly difficult and complicated due to the following reasons:

  - **Class Imbalance:** There is a high imbalance in the dataset, with the number of positive examples ($y_i = 1$) being significantly smaller than the number of negative examples ($y_i = 0$). Mathematically, this can be expressed as: $|\{y_i | y_i = 1\}| << |\{y_i | y_i = 0\}|$ where $|\cdot|$ denotes the cardinality of the set. This imbalance poses major difficulties in training effective models, as the majority class may introduce bias.
  - **Data Noise:** The input texts (X) are inherently noisy due to the informal nature of the comments written by the call center agents, and this noise can be attributed to several different factors:
    - **Non-common Abbreviations:** Call center agents tend to use many abbreviations and domain-specific slang in their comments in order to write faster, which can be challenging for the machine and deep learning models to interpret.
    - **Syntactical and grammatical errors:** The texts also include many grammatical, syntactical, and even spelling errors that further hinder the understanding of the text's meaning.
    - **Weak Supervision:** The dataset's labels (Y) are constructed based on a business rule, rather than being manually assigned by experts. This introduces a degree of noise into the labels, as the rule may not perfectly capture the true sentiment and meaning of the text. This can happen because there is a possibility that the customers' subsequent actions in real life may not align with what they were saying in the calls, one up to five months ago.

In order to tackle the above challenges, the exploration and assessment of different NLP methods as well as machine and deep learning models is necessary, to effectively address the task of loan arrangement prediction from noisy Greek call center transcripts.

3.2 Proposed Approach

In order to overcome the difficulties of making predictions about the success of loan arrangements based on noisy Greek call center transcripts, this thesis proposes a solution that combines several key strategies and methods, which are briefly described below.

3.2.1 Diverse Model Exploration

An important aspect of this research is a detailed comparative analysis of various machine learning and deep learning models as detailed below:

Traditional Machine Learning Models: The study investigated the performance of established machine learning models such as Logistic Regression, Support Vector Machines (SVMs), Naive Bayes, Random Forests, and XGBoost. These models are used together with various NLP feature extraction methods, including:

- Bag-of-Words (BoW)
- Term Frequency-Inverse Document Frequency (TF-IDF)
- N-grams
- Word Embeddings (Word2Vec)

Deep Learning Language Models: The usage of sophisticated deep learning language models was also explored, including those that have been pre-trained on Greek text data or are multilingual. These models are:

- GreekBERT
- FinBERT

- Meltemi (based on Mistral 7B)
- XLM-RoBERTa
- Llama 3.1 70B

These models take advantage of their capability to capture contextual information and semantic relations in the text and they can result in state-of-the-art predictive capability in text classification tasks, as it has already been demonstrated by related work.

3.2.2 Addressing Class Imbalance

In order to address the issues raised by the problem of the data set being extremely unbalanced, the following strategies are explored and assessed:

Class-weighted Loss Functions: During the model training process, we use class-weighted loss functions in which more weight is given to the minority class, i.e. successful loan arrangements, so that the model is pushed to learn more about this class.

Synthetic Data Generation: Taking inspiration of this well-established technique, this thesis considered how the Large Language Models can be used to create synthetic examples of successful loan contracts. This is one of the methods which target at improving the data distribution by increasing the proportion of the minority class and hence enhance the models capability to predict results for examples that have not seen before.

Contrastive learning: There was also an attempt to tackle the lack of positive examples by using contrastive learning, where the GreekBERT was trained to learn to map semantically equivalent sentences to closer representations, even if the examples belong to different classes or categories (successful or unsuccessful loan arrangements). This could possibly urge the model to shift its attention to the semantic meaning of the agents' comments, instead of the class labels only, which in turn could enhance its capacity for transferring the learning to the class of successful loan arrangements that is under-represented.

3.2.3 Mitigating Data Noise

In order to mitigate the problem of noise in the call center transcripts, the following rule-based as well as AI-based approaches were integrated:

Rule-based Text Refinement: A set of manual rules utilizing the spaCy NLP library was implemented in order to handle many abbreviations, to fix spelling errors and to generally support in increasing the efficacy of the lemmatization process.

AI-powered Text Refinement: As for the text refinement assisted by AI, the study adopted a new approach where a state-of-the-art LLM was employed to automatically identify and correct grammatical errors, disambiguate certain elements, and enhance the coherence and uniformity of the transcripts.

3.2.4 Optimization and Resource Management

Due to the challenges in fine-tuning language models and the issues that arose by the limited computational resources, a substantial part of this work was coming up with various ways of optimizing the models to be more efficient and cost-effective:

Low-Rank Adaptation (LoRA): LoRA was used to make the fine-tuning of models like Meltemi and XLM-RoBERTa feasible even in settings with relatively low memory capacity or in the absence of a powerful modern GPU.

Quantization: The use of quantization techniques was driven by the same factors that made LoRA valuable for the experiments that were carried out. Quantization acts as an efficient compression technique to reduce the memory footprint and computational cost of deep learning models.

The above techniques, in conjunction with the gradient accumulation, allowed for the inclusion of more complicated language models that are based on the transformer architecture and can potentially achieve top-notch performance.

Through the adoption of these various methods, this thesis has made substantial efforts to develop efficient and reliable predictive models for the success of loan arrangements using noisy Greek call center comments, thus contributing to the further development and establishment of NLP methods for the Greek language.

Chapter 4

Experiments Description

4.1 Dataset construction

The dataset construction process that was followed includes several steps of deduplication, data integrity and validity checks, joining different entities, constructing the final Greek texts by concatenating several action comments made by the same `account_id`s, and finally creating the labels for the dataset. A flowchart describing the full process can be found in Appendix B.

Initially, there were three different datasets:

The first was `TMP_Actions`, which comprised monthly snapshots that contained the following columns:

- `id`: The unique identifier of a specific action
- `snapnum`: The snapshot number, corresponds to a month
- `meas_action_datetime`: The datetime when the action comment was created
- `meas_action_acct_code_concerned`: The associated `account_id` for which the action comment was written
- `meas_action_cust_code_concerned`: The associated `customer_id` for which the action comment was written

- `meas_action_comment`: The texts that were written by call center agents during communications (actions) with the customers

The second dataset was `DMCR_Unstruct_Customers`, which contained the following columns:

- `id`: The unique identifier of the customers entity (`customer_id`)
- `snapnum`: The snapshot number, which corresponds to a month
- `meas_cusl_min_communication_date_3m`: The date of the first communication in 3 months with this `customer_id`, for each given month (snapshot number)

The third dataset was `DMCR_Unstruct_Accounts`, which included the following fields:

- `id`: The unique identifier of the accounts entity (`account_id`)
- `snapnum`: The snapshot number corresponding to a specific calendar month
- `meas_acct_cust_code`: The `customer_id` which is the primary account holder
- `meas_acch_date_nominal`: The date of the observation, which is the first calendar day of the month corresponding to the snapshot number
- `meas_accl_appl_status`: The status of the last submitted application for loan arrangement within the observation period
- `meas_accl_application_bucket`: The bucket of the latest application within the observation period. This is essentially the number of months for which the account has not performed any installment payment.
- `meas_accl_application_pending`: A boolean which indicates whether the account has a pending application within the observation period (snapshot)
- `meas_accl_paid_in_full_cm`: A boolean indicating whether the account has a paid-in-full installment within the observation period (snapshot)

The steps that were followed in order to create the dataset for the experiments are as follows:

1. First, the rows from `TMP_Actions` that do not have a snapshot number (if any) were dropped, and a check was performed to identify if there are any duplicate action comments that were written for the same account in the same snapshot. The check concluded that there are indeed accounts with more than one action comment per snapshot. This was an important discovery and a point of consideration for subsequent steps.

2. After importing also the Customers and Accounts datasets and cleaning some date columns, another check was performed to determine whether every account is related to only one primary account holder, which seems to be true.

3. Then, a perimeter is applied on the Accounts dataset so that only accounts of interest for this particular study are kept. The rules of this perimeter are as follows:

   - The account must not have an active application for a loan arrangement in the current snapshot.
   - The status of the last submitted application for a loan arrangement must not be one of these:
     - Approved
     - Running
     - Fulfilled
     - Partially fulfilled
     - Out of collection

4. Next, the rows from the Accounts dataset that were preserved are joined twice with the Accounts dataset again, bringing each time information from subsequent snapshots (`snapnum+1` and then again from `snapnum+2`). This enriches every row with information covering a 3-month period (current snapshot plus two subsequent ones), required for constructing labels.

5. Labels are constructed following this rule: If an account has performed an installment payment in either of two subsequent months, its label = 1 (positive class); otherwise, it belongs to label = 0 (negative class).

6. After creating the labels, the working dataset is joined with the `TMP_Actions` dataset, which contains the text comments written by agents. A rolling three-month window is created for each record. For each of the three calendar months, all comments made in each month are concatenated together in descending chronological order.

   Example: For a record in June, three different columns would be created:

   - `meas_action_comment_concat_0` for June
   - `meas_action_comment_concat_-1` for May
   - `meas_action_comment_concat_-2` for April

   Each column contains all action comments written for the specific `account_id` of the record, ordered from most recent to oldest. This transformation is necessary because, in every calendar month, there can be more than one action comment written for a given `account_id` (see bullet 1).

7. After these steps, the comments from the current, previous, and pre-previous snapshots (calendar months) are concatenated together to create 3-month histories in one column. Extending the example from point 6 above, the column `meas_action_comment_concat_3m` is created by concatenating:

   - `meas_action_comment_concat_0`,
   - `meas_action_comment_concat_-1`, and
   - `meas_action_comment_concat_-2`.

   This column contains, for the specific scenario, all comments made from June 30th back to April 1st for a given `account_id`. Then, Spacy's Greek model (`el_core_news_lg`) is used to lemmatize the texts. Towards this end, a custom lookup table with hardcoded rules has also been developed, where the rules have the following format:

`apanta` → απαντώ

`enhm` → ενημερώνω

`ktx` → κάτοχος

etc.

This final version of the dataset contains:

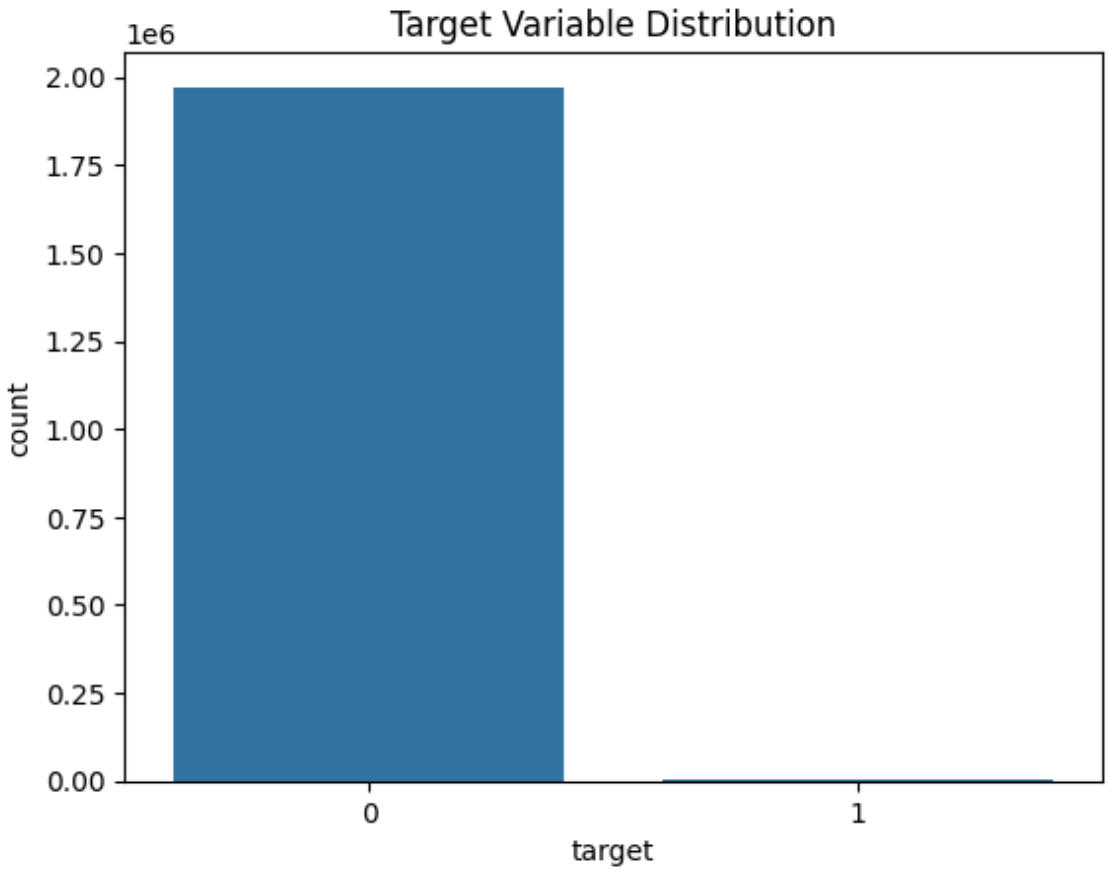
- The lemmatized last 3-month comments,
- The corresponding `account_id`,
- The snapshot number (`snapnum`),
- And the labels.

This dataset will be used in all subsequent experiments described next.

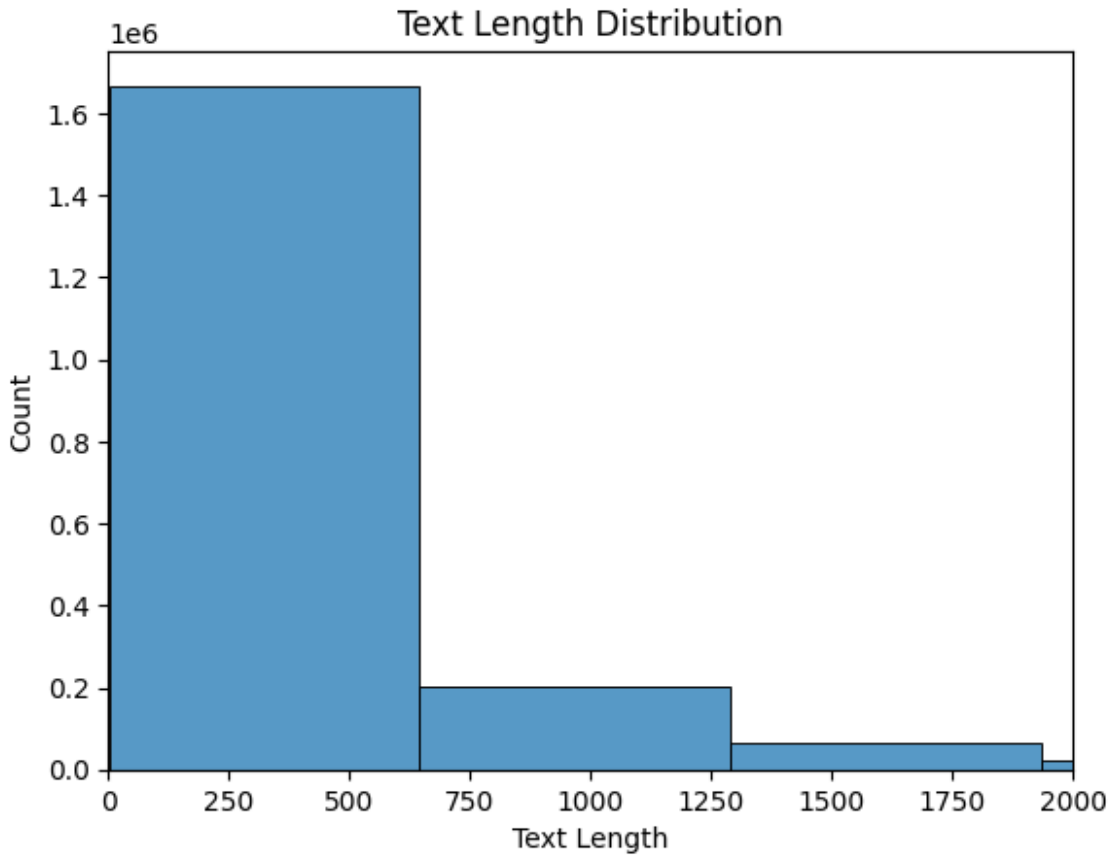
4.2 Exploratory Data Analysis

After having constructed the dataset that is going to be used, an exploratory data analysis was performed in order to understand the dataset's characteristics and potentially reveal interesting patterns, distributions and possibly issues.

Regarding the distribution of the classes, the negative class (label: 0) was found to contain 1971877 examples, while the positive class included only 3612 instances, accounting for the 0.18% of the total dataset, indicating the extreme class imbalance of the data.



When inspecting the text length distribution, it is revealed that the mean text length is 346 characters, with a standard deviation of 520 characters. The distribution of the text lengths can be seen below:



Moreover, the most common words were analyzed, both in the total dataset and per class, and they were found to be:

- **The most common words are:**

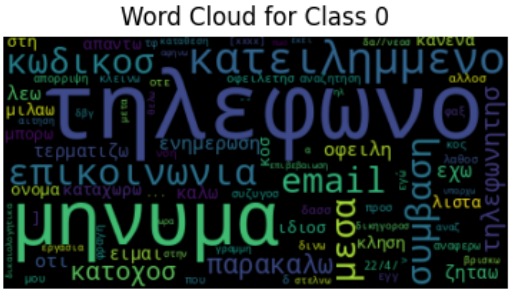
  [(΄τηλεφωνο΄, 2724070), (΄μηνυμα΄, 1664232),
  (΄κατειλημμενο΄, 1653497), (έπικοινωνια΄, 1136849),
  (΄μεσα΄, 979850), (έμαιλ΄, 973028),
  (΄συμβαση΄, 965142), (΄κωδικοσ΄, 964405),
  (΄παρακαλω΄, 962587), (΄τηλεφωνητησ΄, 905156),
  (΄κατοχοσ΄, 870188), (έχω΄, 560728),
  (ότι΄, 492790), (όφειλη΄, 476086),
  (ένημερωση΄, 460829), (΄λεω΄, 436732),
  (΄καλω΄, 391606), (΄τερματιζω΄, 382559),
  (΄ζηταω΄, 368629), (είμαι΄, 361200)]

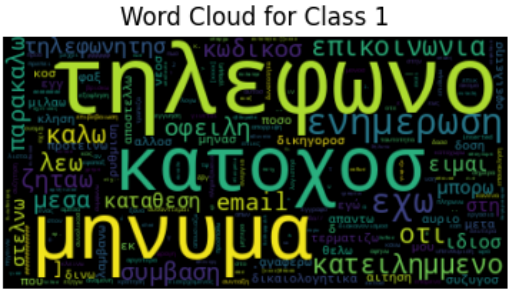
- **Most common words for class 0:**

[(΄τηλεφωνο΄, 2716609), (΄μηνυμα΄, 1659259),
(΄κατειλημμενο΄, 1650006), (έπικοινωνια΄, 1133559),
(΄μεσα΄, 977409), (έμαιλ΄, 970672),
(΄συμβαση΄, 962909), (΄κωδικοσ΄, 962136),
(΄παρακαλω΄, 960361), (΄τηλεφωνητησ΄, 902935),
(΄κατοχοσ΄, 865596), (έχω΄, 557196),
(ότι΄, 489359), (όφειλη΄, 473528),
(ένημερωση΄, 457049), (΄λεω΄, 434091),
(΄καλω΄, 388549), (΄τερματιζω΄, 381594),
(΄ζηταω΄, 365888), (είμαι΄, 359378)]

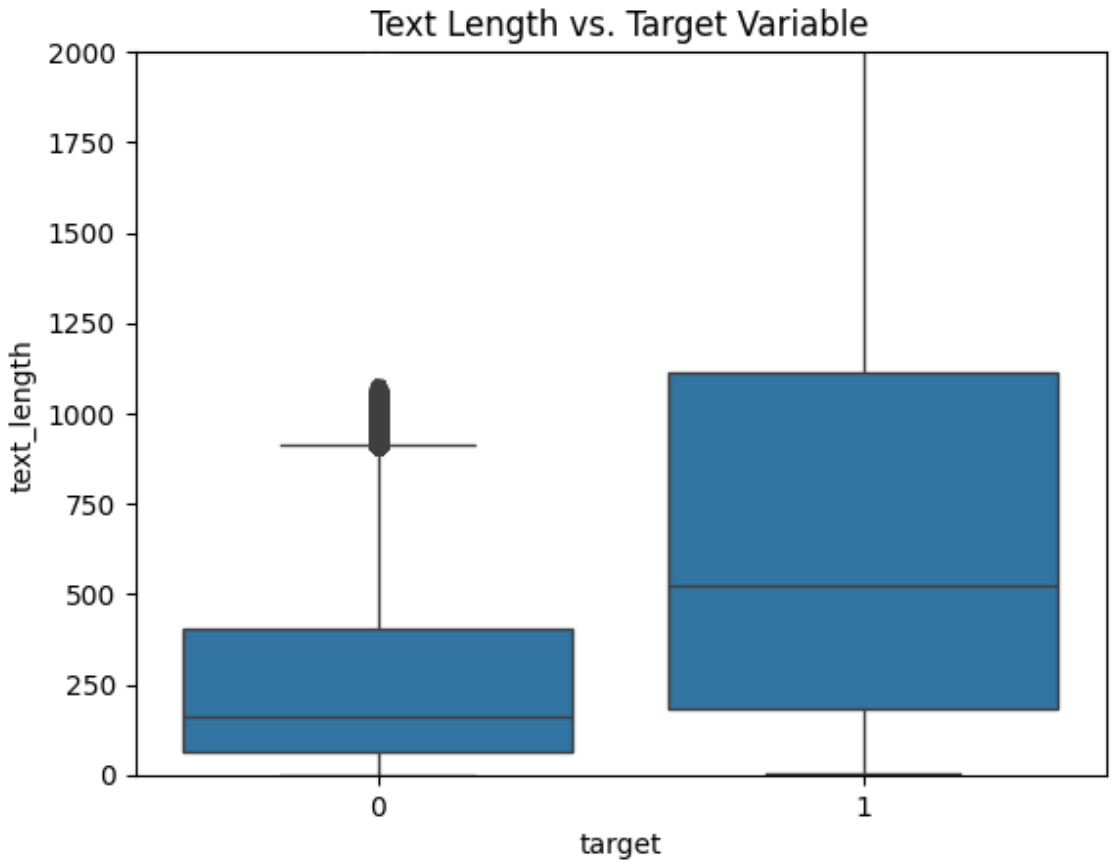
- **Most common words for class 1:**

[(΄τηλεφωνο΄, 7461), (΄μηνυμα΄, 4973),
(΄κατοχοσ΄, 4592), (ένημερωση΄, 3780),
(έχω΄, 3532), (΄κατειλημμενο΄, 3491),
(ότι΄, 3431), (έπικοινωνια΄, 3290),
(΄καλω΄, 3057), (΄ζηταω΄, 2741),
(΄λεω΄, 2641), (όφειλη΄, 2558),
(΄μεσα΄, 2441), (έμαιλ΄, 2356),
(΄κωδικοσ΄, 2269), (΄συμβαση΄, 2233),
(΄παρακαλω΄, 2226), (΄τηλεφωνητησ΄, 2221),
(΄καταθεση΄, 1856), (είμαι΄, 1822)]

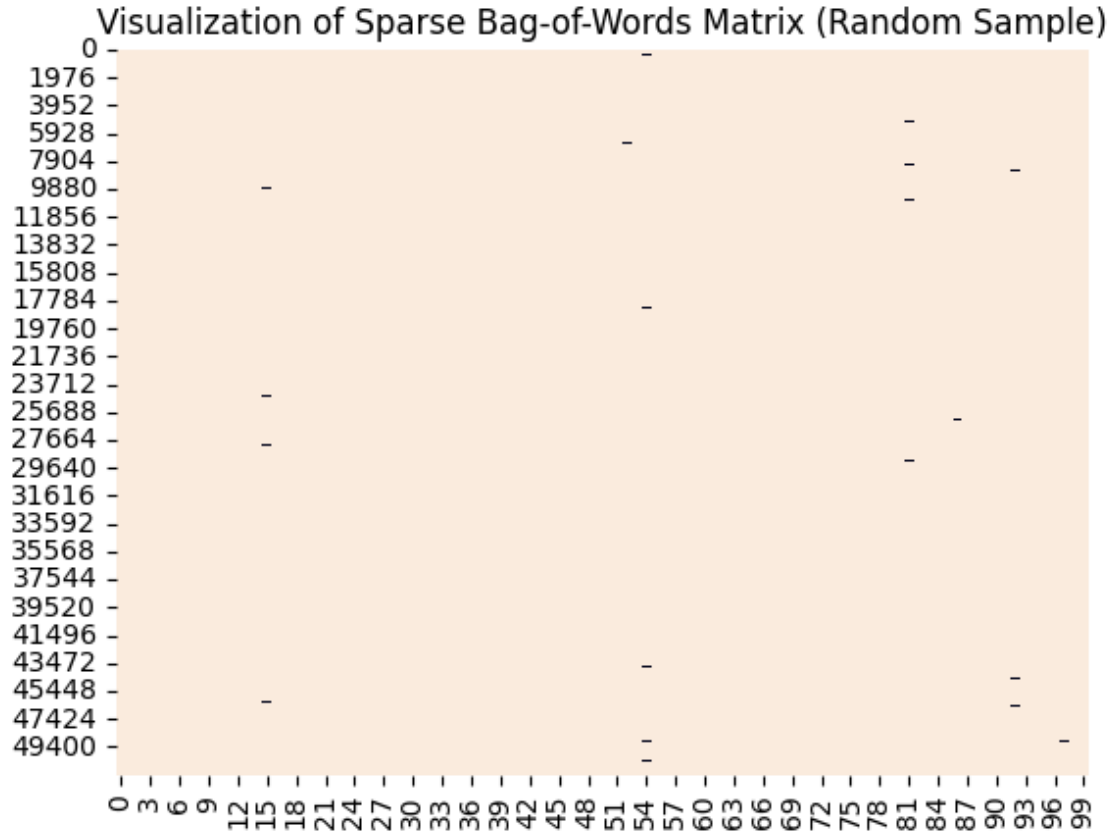




Furthermore, when investigating the relationship between the text length and the label, it was discovered that longer texts seem to be more likely to belong to the positive class:



Last, a CountVectorizer was applied to transform the textual information into Bag-of-Words (BoW) and thus generate sparse matrices of word frequencies for the dataset. The size of the vocabulary created by the vectorizer was obtained and displayed. A heatmap visualization is then generated for 100 randomly chosen columns of the sparse matrix and the density of non-zero entries (word counts) is shown to understand the sparsity of the data. This gives an understanding of the structure as well as the level of sparsity of the BoW representation, indicating the absence of many words in individual documents.



4.3 Dataset deduplication and splitting process

The dataset, as it has been already described, contains texts and labels, associated with account IDs and snapshot numbers. There are combinations of {text, label} that occur many times. For this reason, a data deduplication was needed. Furthermore, in order to split the dataset into train, validation, and test (unseen) data, it was necessary to take into account the proportion of the classes, the chronological order of the data, and the account IDs. Thus, a StratifiedGroup split was required, taking care also of the chronological order of the data.

The deduplication process will be first described, which takes as input the `final_dataset` and produces as output a deduplicated dataset with columns `account_id`, `snapnum`, `text`, and `target` (label), which are needed in order to split our dataset. The downsampling ratio that has been used for the negative class in the train and validation datasets is 0.1.

First, the `final_dataset` that has been produced is read (both the lemma-

tized and the not lemmatized version). An inner join is performed, using as keys the combination {ACCOUNT_ID, SNAPNUM}. The deduplication and splitting process has been integrated in the first subsequent experiment of the deep language models, which is finetuning the GreekBERT. For this reason, an instantiation of the class DeduplicatedNegativeDownsampledExperimentHandler takes place, inheriting from the parent ExperimentHandler class. The following arguments are passed to the class:

- experiment_dataset, which is the joined dataframe based on the final_dataset.
- experiment_name, which in this case is greek_bert.
- model_class, which is the class of the model we are going to train from the HuggingFace transformers library (BertForSequenceClassification).
- model_save_path, which is the path where we are going to save the fine-tuned model, in this case: greek_bert_deduplicated_train_data_neg_downsampled_model.

Since no model_load_path is provided, the base pre-trained model which is used for fine-tuning is nlpaueb/bert-base-greek-uncased-v1. Furthermore, the text_column is the meas_action_comment_concat_3m_lemmas_string column in the dataframe, and the target_column is the "target" column.

Then, the method prepare_train_test_data() is invoked, by passing the train_data_path, validation_data_path, and test_data_path where we are going to save the corresponding datasets. This method executes the following steps:

1. Calls the internal method _deduplicate_dataset(). This method converts the experiment_dataset (the merged final_dataset) to a Polars LazyFrame and applies the deduplication logic, which is the following:
   - **Assumptions made:**
     (a) Each text should be associated with only one label (either positive or negative). It is not expected as a real life scenario for two different customers to say the exact same words and then follow different behaviors.

(b) It is sufficient for each combination of {`text`, `label`} to exist for only one `account_id`. The model will either train or evaluate on this combination.

(c) However, some {`text`, `label`} combinations exist for more than one `account_id`.

(d) Furthermore, some `account_ids` include a given combination of {`text`, `label`} more than one time.

- **The deduplication logic is the following:**

(a) For a given combination of {`text`, `label`}, if there is only one `account_id` for which this combination appears more than once, only one row containing the {`text`, `label`, `account_id`} is kept, and the `snapnum` with the maximum value is retained (the most recent is the most relevant from a business perspective).

(b) Else if there are more than one `account_ids` for a given {`text`, `label`} combination, the `account_id` with the maximum count of total rows is kept (ties are broken randomly), retaining the `snapnum` with the maximum value. This is done, because it would be beneficial to preserve the account_ids with the most information in our dataset.

(c) Else if for a given combination of {`text`, `label`}, there are no `account_ids` for which this combination appears more than once, the `account_id` with the maximum count of total rows is kept, retaining the `snapnum` as above.

(d) Last, a check is performed to validate that each text is associated with only one label, either positive or negative. For texts that are associated with both labels, the positive label is favored (see assumption a). All these operations are performed because, as said, there is no expectation that this will be a real life case, and since there are very few positive examples in the dataset, none of them could be sacrificed.

2. Checks that the deduplication has been completed successfully, by invoking the method `_check_dataset_deduplication()` to ensure the uniqueness of each {`text`, `label`} combination, its association with only one `account_id`, and to also ensure that each text is associated with only one label.

3. Then, the deduplicated dataset is split to train, validation and test datasets. The final percentages are: 70% for train, 17.5% for validation, and 12.5% for test data. The steps that are followed for doing the splitting are the following:

   (a) **A**ssumptions made/Acceptance criteria for the splitting process:

      i. Each `account_id` can exist either in the train set, or in the test (or validation) set, or in both of them.

      ii. In order for an `account_id` to be able to exist in both sets, this means that this `account_id` must have a sufficient amount of examples.

      iii. Regarding this sufficient amount, both classes for a given `account_id` must have more instances than the number of folds, so that the process does not fail and the stratification objective is not violated.

      iv. For `account_ids` that exist in both sets, the instances that exist in dataset B (`test` or `validation` set) must be in the future compared with the examples of the same `account_id` that exist in dataset A (`train*` or `train` set).

   (b) The whole process is repeated twice. The first time, eight folds are used, and the deduplicated dataset is taken as input. Thus, a split into `train*` (dataset A) and `test` (dataset B) dataset is performed. The second time, five folds are used and the `train*` dataset is taken as input. Thus, the final split is completed to create the `train` (dataset A) and `validation` (dataset B) datasets, ensuring 70-17.5-12.5 proportions and verifying that all criteria have been met for the splitting process. The steps that are followed for the splitting process include:

      i. First, the `input_dataframe` is sorted by {`ACCOUNT_ID`, `SNAPNUM`} in descending order (see step [v] below to understand why), and then

the `LazyFrame` is collected in order to be converted into a Polars DataFrame and be able to make some required groupings.

ii. The grouping is made by {`ACCOUNT_ID`, `TARGET_COLUMN`} in order to calculate the number of occurrences of each target per each `account_id` and then the result is pivoted to bring the `account_ids` as index, the target labels (0,1) as columns, and the frequencies as values.

iii. A check is performed to verify that both zeros and ones are more than the number of folds, and two boolean flags are created for each `account_id` accordingly. These `account_ids` that have both flags `True` are tagged as `repeated_ids` (thus creating the `repeated_account_data`), while the rest of the input dataset is tagged as `new_account_data`.

iv. For the `new_account_data`, the sklearn's `StratifiedGroupKFold` split is performed. The rationale is the following: since these `account_ids` do not have class instances that are more than the number of folds, then this means that every `account_id` should exist either in the `train` or in the `test` dataset, while trying to retain the class balance ratio. Only the first of the splits is kept by using the `next()` Python function—no cross-validation is used because of the computational cost to fine-tune such LLMs, as it will be shown in the sections below.

v. For the `repeated_ids`, the `StratifiedKFold` split is used instead. Each of the repeated `account_ids` data are split into two sets, again by keeping only the first of the splits with the `next()` function, and also by preserving the chronological order of the snapshots for each `account_id`. This is achieved by having sorted by {`ACCOUNT_ID`, `SNAPNUM`} in descending order in step [i], and by using `shuffle=False`. In this way, future data (larger `snapnums`) will always be in the set B, out of the two sets that are created by the split.

vi. The datasets A and B that have been created in steps [iv] and [v] are concatenated together in order to create the final datasets A (`train*` or `train`) and B (`test` or `validation`).

4. Then, the `pl_check_stratified_group_train_test_split()` function is called

to validate that the created datasets meet the acceptance criteria that have been described in the assumptions. Assertions are made to verify that the balance ratio of the classes has been preserved, the total length of the split datasets is the same as the input dataset's length, and the `account_ids` that exist in both sets do not have chronological overlap.

5. Last, the negative class is downsampled in both the `train` and `validation` datasets, keeping only the 1% of the negative class (due to the huge class imbalance of the problem). However, the negative downsampling is not performed for the `test` dataset, since the unseen test data that are used for the evaluation of the models have to resemble the real world as much as possible. It is worth noting that this test dataset remains completely unseen and is used to evaluate the trained (or fine-tuned models) in all following experiments that are going to be described.

4.4 Traditional ML Models with NLP techniques

For a comprehensive flowchart depicting all ML experiments and the results, please refer to Appendix B.

4.4.1 Random Forest with TF-IDF and SMOTE

The first of our experiments with traditional ML models forms an initial baseline of what to expect next. It performs binary text classification using a Random Forest classifier on the dataset that have been prepared following the process described above. It uses TF-IDF vectorization with bigrams and trigrams to convert textual information into numerical characteristics and enforces SMOTE to deal with imbalanced classes by over-sampling the smaller class. Thus, after training the classifier on the resampled data, it assesses the model's performance on the test set by computing such performance measures as accuracy, precision, recall, F1-score, confusion matrix and PR-AUC. Furthermore, it calculates rank-based measures such as Precision@20, Recall@20, and R-Precision where predictions are reordered according to their predicted probabilities. These metrics are used to evaluate both classification

and ranking performance against a dataset that features a severe shortage of positive examples. The results are depicted below:

```
RF:
    Accuracy: 0.939
    Precision: 0.014
    Recall: 0.209
    F1 Score: 0.025
    Confusion Matrix:
        [[82816  5110]
        [  265    70]]
    Precision@20: 0.0
    Recall@20: 0.0
    R-Precision: 0.009
    PR-AUC: 0.008
```

4.4.2 Bag-of-words with χ^2 feature selection

In this experiment, the most important features for the classification task were identified by applying the Chi-squared test, where the Chi-squared statistics and p-values were computed for each word in the Bag-of-Words matrix with regards to the target labels. Then, the features were ranked and filtered based on their statistical significance (1 minus the p-value), since only those with a score greater than a defined threshold were selected. The resulting significant features were stored as a list of unique words to be used to refine the feature set for modeling. The final feature set for the text classification was created by generating Bag-of-Words matrices for the training and testing sets using only the selected significant features with the CountVectorizer.

After this process, several classic classifiers were instantiated and evaluated on the refined datasets. The results are displayed below:

```
MNB:
    accuracy: 0.866
```

recall: 0.454

precision: 0.013

f1_score: 0.025

precision_at_20: 0.0

recall_at_20: 0.0

r_precision: 0.066

pr_auc: 0.019

SVC:

accuracy: 0.996

recall: 0.033

precision: 0.786

f1_score: 0.063

precision_at_20: 0.6

recall_at_20: 0.036

r_precision: 0.107

pr_auc: 0.062

ABC:

accuracy: 0.993

recall: 0.084

precision: 0.088

f1_score: 0.086

precision_at_20: 0.25

recall_at_20: 0.015

r_precision: 0.090

pr_auc: 0.036

DT:

accuracy: 0.973

recall: 0.251

```
    precision: 0.038
    f1_score: 0.066
    precision_at_20: 0.05
    recall_at_20: 0.003
    r_precision: 0.027
    pr_auc: 0.013

RF:
    accuracy: 0.995
    recall: 0.176
    precision: 0.288
    f1_score: 0.219
    precision_at_20: 0.6
    recall_at_20: 0.036
    r_precision: 0.179
    pr_auc: 0.110

XGB:
    accuracy: 0.995
    recall: 0.158
    precision: 0.237
    f1_score: 0.190
    precision_at_20: 0.5
    recall_at_20: 0.030
    r_precision: 0.188
    pr_auc: 0.087
```

From the above results, it is evident that the ensemble models like Random Forests and XGBoost achieve the best results in terms of f1-score and ranking metrics like R-Precision. What is also amazing is that SVC seems to perform better than the rest of the models, especially with regards to the ranking metrics, but it seems that the model is overly hesitating to predict the positive class, offering very good

precision, but low recall. On the other hand, Naive Bayes seems to achieve the best recall, but the fact that R-Precision is very low should not be overlooked. This essentially means that the model is incapable of prioritizing the positive examples and ranking them higher. It may predict enough of them correctly, but they are further down the ranking order, surrounded by negative examples.

4.4.3 Normalization of the BoW features

The next experiment involved normalizing the numerical features that were created by the CountVectorizer using the most important words. This method ensures that all numerical features have a consistent scale, and can potentially improve the performance of some distance-based models like SVM or k-NN. However, some other models like the tree-based ones (Random Forests, Decision Trees, XGBoost etc.) do not particularly benefit from this normalization. Moreover, it can have a negative impact on models that are optimized for sparse matrix inputs, such as the Naive Bayes, since normalizing the text features may reduce sparsity.

Only the SVC classifier is trained as part of this experiment, and the results are depicted below:

```
SVC:
    accuracy: 0.996
    recall: 0.078
    precision: 0.812
    f1_score: 0.142
    precision_at_20: 0.6
    recall_at_20: 0.036
    r_precision: 0.112
    pr_auc: 0.070
```

As expected, the normalization seems to assist SCV, which is a distance-based model, in achieving a bit better precision and recall.

4.4.4 Normalization of the BoW features and class weights

The normalized dataset that was created during the previous experiment is used again, this time also including class weights in the training process. The class weights are hard-coded to 1 for the negative class, and 5 for the positive class. The results of the fewer models that were indicatively trained using this approach are below:

```
RF:
    accuracy: 0.992
    recall: 0.164
    precision: 0.115
    f1_score: 0.135
    precision_at_20: 0.5
    recall_at_20: 0.030
    r_precision: 0.158
    pr_auc: 0.082

XGB:
    accuracy: 0.995
    recall: 0.134
    precision: 0.205
    f1_score: 0.162
    precision_at_20: 0.6
    recall_at_20: 0.036
    r_precision: 0.161
    pr_auc: 0.089
```

By examining the results, it is apparent that both Random Forest and XGB seem to have got slightly worse, compared with the results that were realized using the simple BoW technique, though this could be attributed to randomness. Multiple runs and statistical testing would be necessary to specify if there is a statistically significant difference in their performance.

4.4.5 N-grams with χ^2 feature selection

The previous process that involved Bag of Words (BoW) is repeated with 2-grams and 3-grams, using again the most significant features from the χ^2 feature selection. The results of the trained machine learning models are presented next:

```
MNB:
    accuracy: 0.866
    recall: 0.454
    precision: 0.013
    f1_score: 0.025
    precision_at_20: 0.0
    recall_at_20: 0.0
    r_precision: 0.066
    pr_auc: 0.019

RF:
    accuracy: 0.995
    recall: 0.170
    precision: 0.284
    f1_score: 0.213
    precision_at_20: 0.55
    recall_at_20: 0.033
    r_precision: 0.179
    pr_auc: 0.110

XGB:
    accuracy: 0.995
    recall: 0.158
    precision: 0.237
    f1_score: 0.190
    precision_at_20: 0.5
```

```
recall_at_20: 0.030
r_precision: 0.188
pr_auc: 0.087
```

The results appear to be the same with the BoW method for feature extraction.

The top 20 most important features from the Random Forests are shown below:

Feature Names	Importance
δα	0.0274
ξξξξς	0.0183
σε	0.0121
τηλεφωνο	0.0112
απο	0.0108
μηνυμα	0.00956
δεν	0.00897
τηλεφωνητησ	0.00850
με	0.00812
κατοχοσ	0.00774
τερματιζω	0.00771
κα	0.00729
δασσ	0.00663
να	0.00649
για	0.00555
και	0.00547
θα	0.00528
καλω	0.00484
ενημερωση	0.00460
δοση	0.00459

4.4.6 N-grams and class weights

This time, instead of applying normalization on the dataset, n-grams are used directly with class weights to train the Random Forest and the XGBoost models. The results are the following:

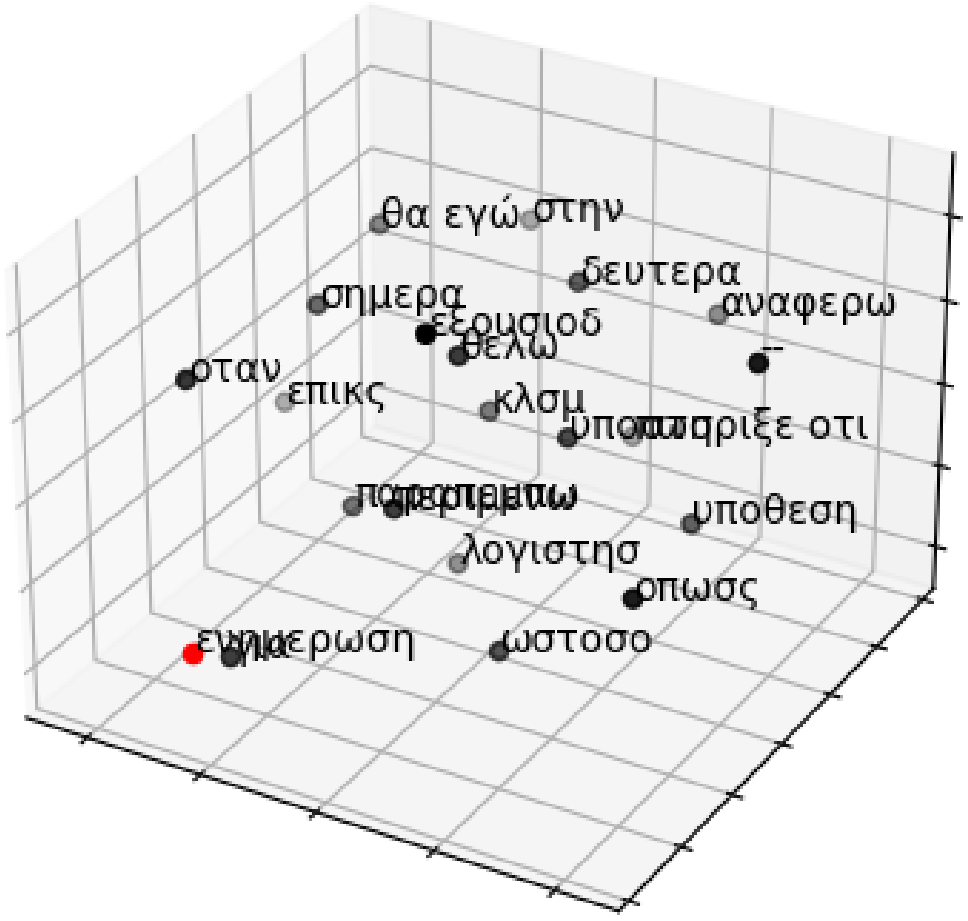
```
RF:
    accuracy: 0.993
    recall: 0.173
    precision: 0.163
    f1_score: 0.168
    precision_at_20: 0.55
    recall_at_20: 0.033
    r_precision: 0.173
    pr_auc: 0.095

XGB:
    accuracy: 0.995
    recall: 0.158
    precision: 0.237
    f1_score: 0.190
    precision_at_20: 0.5
    recall_at_20: 0.030
    r_precision: 0.188
    pr_auc: 0.087
```

Similarly, no significant difference seems to exist in contrast to the first experiment that used the BoW features.

4.4.7 Averaged Word2Vec Embeddings for Text Classification with XGBoost

In this experiment, the corpus of texts is tokenized to individual words, and the `Phrases` model of the gensim library is initially employed to scan through all texts in the dataset and identify bigrams and then trigrams that appear at least 5 times in the corpus. After having tokenized the texts and also created the bigrams and trigrams, a `Word2Vec` model [14] is trained on the documents, using 300 dimensions for the word embeddings, a window of 8 words for the context of each word and a minimum of at least one appearance, and finally the skip-gram algorithm. After training the model, it is tested by plotting the most similar words to the anchor word ενημερωση , using the t-SNE dimensionality reduction technique [74] to reduce the dimension to the 3d space. The scatterplot is displayed below:



Then, each text in the dataset is transformed into a `Word2Vec` embedding, by taking the embedding of each word in the sentence and average them all for a final vector on the sentence-level. A XGBoost classifier is trained on this dataset, and the evaluation metrics are presented below:

```
XGB:
    accuracy: 0.994
    recall: 0.176
    precision: 0.203
    f1_score: 0.188
    precision_at_20: 0.35
    recall_at_20: 0.021
```

```
r_precision: 0.179
pr_auc: 0.078
```

Suprising as it may be, XGBoost achieved marginally worse performance with the average Word2Vec embedding per sentence, than the simple Bag of Words.

4.4.8 Attention-Driven BiLSTM Text Classifier with `Word2Vec` Embeddings

In the final of the first set of experiments, we created a text classification model that uses `Word2Vec` embeddings from the existing trained model of the previous experiment, and an attention mechanism to enhance the classification outcomes.

Initially, each of the texts in the train dataset is tokenized and converted to a sequence of 15 integers (indexes), each of which corresponds to a word from the vocabulary. If a text has more than 15 words, they are truncated; likewise, if a text has less words, padding is applied. The same process is applied to the test dataset, using the bigrams and trigrams detectors that we trained earlier on the train dataset (thus avoiding data leakage), and performing the same tokenization and padding steps for the test texts. The embedding layer of the neural network took advantage of the existing `Word2Vec` embeddings, as already described, thus the weights of the Embedding layer were frozen. The embedded sequences were processed in a bidirectional `LSTM` network with an attention mechanism to focus on the most important tokens.

An Optuna study was performed to determine the best number of epochs, and the optimal class weights. To assess the model's predictive power, we employed metrics including precision, recall, and F1-score after training. We also evaluated the model as a ranker, estimating the Precision@20, Recall@20, R-Precision, and PR-AUC. The best params after 10 trials by Optuna were found to be 27 epochs and weight 1.052 for the positive class (keeping the negative class with weight 1). Nevertheless, the results of this neural architecture were disappointing, since the model predicted that no example belonged to the positive class. This could potentially indicate the need for handling the class imbalance and the data noise in more efficient approaches.

The results can be found below:

```
Accuracy: 0.996
Precision (class 1): 0.0
Recall (class 1): 0.0
F1-Score (class 1): 0.0
Precision@20: 0.1
Recall@20: 0.006
R-Precision: 0.072
PR-AUC: 0.023
```

4.5 Deep learning with Language Models

For a comprehensive flowchart depicting all DL experiments with language models and the results, please refer to Appendix B.

4.5.1 Experiment 1: Fine-tuning GreekBERT

The first experiment is fine-tuning the GreekBERT [29] using the train and validation dataset, and then evaluating on the test dataset. In this first experiment, the whole process will be thoroughly described, along with several configurations and optimization techniques that were used. First of all, Optuna [75] is used in order to search for the best `num_epochs`, `batch_size`, `accumulation_steps`, and `learning_rate`. This is done only in this first experiment and the contrastive learning experiment, and the optimized hyperparameter values are kept for the next experiments because this process is computationally costly.

4.5.1.1 Hyperparameters Optimization Process

Optuna essentially makes 10 trials with the objective of minimizing the cross-entropy loss for the validation dataset, using Bayesian Optimization for efficient sampling of the following hyperparameters and values:

- `num_epochs` **(2-5):** This range of integers has been chosen because of the small datasets and the few existing positive examples. It allows the model to learn without overfitting on the data, as too many epochs could potentially have a negative effect on generalization.

- `batch_size` **([16, 32, 64]):** Batch size is the number of examples that get processed in parallel and re-calculate the gradients and loss before eventually updating the weights and biases of the model. Although smaller batch sizes can make the gradient descent optimization less smooth and the convergence slower, they contribute to better generalization, especially in our case, since there is a limited amount of positive examples.

  In our particular case, smaller batch sizes are of vital importance also for another reason: the limited computational resources and memory. If a more capable GPU was going to be used, then larger batch sizes in the range of [32, 64, 128] would have been considered, which would contribute to faster learning and potentially more stability in the learning process.

- `accumulation_steps` **(1-4):** The accumulation steps are the number of batches that get processed before updating the weights and biases. This hyperparameter allows for simulating training with greater batch sizes, which can contribute to stabilizing the gradient descent optimization, while keeping the memory requirements manageable. All integers in the range [1, 4] are tried.

- **Learning rate (1e-6 to 1e-4):** This continuous range of numbers covers some of the most commonly used learning rates when fine-tuning language models. Optuna uses a log-uniform distribution for sampling this continuous range, essentially picking smaller learning rates with higher probability, since the number of trials would never be sufficient to exhaustively try all these values.

For all the hyperparameters optimization process described above, the method `tune_hyperparameters()` is called (passing `n_trials=10`) on the object of the class `DeduplicatedNegativeDownsampledExperimentHandler`. The `objective()`

method serves as setting the goal for Optuna's minimization. The GreekBERT model is trained on the training dataset, using a combination of the hyperparameter values, and then evaluated on the validation dataset.

Note that a check is used in each trial, so that Optuna can determine if this specific trial with a set of hyperparameter values looks promising, or if it should stop it (*"prune"*) earlier to save time and compute costs.

4.5.1.2 Technical Details of the Training Procedure

After having the best hyperparameters values specified by Optuna, it is time to train the model on the training dataset once more and save the final best model. For this, the `train_model()` method is invoked again. The best model (with the minimum loss on the validation dataset) across all Optuna's trials is retrieved, in order to further fine-tune it. Some technical details about the training procedure:

- **Data Loading:** The training data loader shuffles the batches to prevent overfitting. However, this shuffling is not used in the validation data loader, as it is not useful during evaluation.

- **Device Allocation:** The model is moved to the device for parallelization and optimization, which is GPU if Google Colab's A100 is used, otherwise CPU.

- **Optimizer:** The Adam optimizer with weight decay and the learning rate determined by Optuna are used. This optimizer is widely applied when fine-tuning large transformer models because it offers improved generalization due to the implementation of the L2 regularization (weight decay), which also does not interfere with the adaptive learning rates [76].

- **Learning Rate Scheduler:** The training process makes use of a learning rate scheduler, which reduces the learning rate to half when the validation loss plateaus and does not decrease after 2 epochs.

- **Batch Accumulation:** The weights are updated only after `accumulation_steps` number of batches, resulting in an effectively larger batch size.

- **Early Stopping and Validation:** The evaluation is performed against the validation dataset, persisting the model if the validation loss is the best so far, halving the learning rate if the loss has not improved after 2 epochs, or even applying early stopping if the loss has not improved after 3 epochs.

4.5.1.3 Performance Evaluation

The best model across all epochs is loaded for a final evaluation on the validation dataset and further evaluation on the test dataset.

When evaluating on the validation dataset (`evaluate_on_validation_set()`), the model is set to `eval` mode in order to disable dropout and other training-specific layers. The gradient computation is also disabled to reduce the required memory and avoid unnecessary calculations.

The losses across all batches are summed up, so that the total loss can be averaged by dividing by the number of batches. This average loss is returned as the output of the method. Additionally, the probabilities of each class and the predictions based on the logits are calculated using the `softmax` and `argmax` functions.

After all these steps, it is time to finally evaluate the model using the test (unseen) dataset. Two separate methods have been implemented for this purpose: the `evaluate_model()` and the `evaluate_as_ranker()` methods.

The `evaluate_model()` method calculates the accuracy, recall, precision, f-1 score and confusion matrix, based on the predictions made on the test set.

However, due to the high class imbalance and the low number of positive examples, evaluating the model as a ranker provides much more meaningful insights about the model's ability to distinguish and prioritize positive instances out of the great number of negative ones.

4.5.1.4 Ranking Metrics for Evaluation

Ranking metrics are ideal for evaluating the model's ability to retrieve and rank positive examples effectively. These metrics align closely with real-life scenarios where the task is to identify a few relevant cases from a large pool of non-relevant ones,

such as in sales leads, client communications, or anomaly detection. Furthermore, ranking metrics like ROC-AUC, Precision@K, Recall@K, R-Precision, and PR-AUC are robust against class imbalance [56].

4.5.1.4.1 ROC-AUC ROC-AUC is a common metric widely used to evaluate the performance of binary classifiers. The area under the curve informs us about the model's ability to rank positive examples higher than negative ones. Essentially, it measures the probability that a binary classifier assigns a higher probability or logit to a positive example than to a negative one.

However, due to the extreme class imbalance in our case, and also because of our particular interest in the positive class, PR-AUC would be a much more suitable evaluation metric. ROC-AUC has been shown to be overly optimistic in imbalanced datasets where the low decision threshold leads to an overly eager to predict the positive class model, while at the same time, the high number of true negatives keep the FPR consistently low (at least until a very low threshold, when it can move toward 1), thus inflating the AUC and over-estimating the model's predictive capabilities [56].

4.5.1.4.2 PR-AUC In contrast, PR-AUC is much more informative since it clearly showcases the trade-off between precision and recall, while calibrating the decision threshold. In this way, ending up with a model that may have a very good recall, but the precision is very low, can be avoided. From a business perspective in the existing application, false positives would mean false leads for communication with customers that are not really ready to arrange their non-performing loan. Communications that are associated with low probability of a loan arrangement consequently mean higher call center costs with lower business expectations and results. For all these reasons, PR-AUC would be perhaps one of the most valuable evaluation metrics for the business stakeholders, too.

4.5.1.4.3 Precision@K and R-Precision Two other really useful metrics matched with the business situation in hand are Precision@K and R-Precision. Both metrics

are ranking metrics and are completely independent of any decision threshold, essentially evaluating the model's capability to retrieve relevant (positive) examples in the top K or R prediction probabilities, respectively. Perhaps Precision@K is a bit more relevant, since it is completely unrelated with the dataset's composition, and the K number can be a business informed quantity that depends on the business productivity capabilities, such as the call center's capacity, the cost for each communication etc. Since the K number (number of communications the business channels can perform in a business period of time) is potentially greater than the actual number of R (true positives in the same period of time, essentially being customers with a higher probability of arranging their loans), Recall@K comes into play. This means the business can bear the burden of some effectless communications, as soon as the model retrieves as many true positives as possible.

All the above-mentioned ranking metrics are calculated when calling the method `evaluate_as_ranker()`. This method ensures that the model's ranking capabilities are thoroughly evaluated and aligned with both technical performance and business priorities.

4.5.1.5 Performance Metrics

```
Accuracy: 0.983
Precision: 0.072
Recall: 0.296
F1 Score: 0.115

Confusion Matrix:
Predicted      0    1    All
True
0          87149  777  87926
1            262   73    335
All        87411  850  88261

ROC-AUC: 0.738
```

```
PR-AUC: 0.087
R-Precision: 0.194
Recall@20: 0.027
Precision@20: 0.45
```

4.5.1.6 Analysis of Results

These results showcase the difficulty of the problem in hand, which is the high imbalance of the class distribution. It is evident that the model learns the negative class quite well, while it seems that it faces major difficulties with the positive class. This was expected, and it serves as a base model for us and as a comparison reference for the next experiments. It's worth noticing that:

- The model retrieved 20% of the total positive examples in the top R ranked probabilities for the positive class.
- The top 9 predictions were all positive examples.

It is also worth mentioning that during the experimentation process, the model was also evaluated on the test dataset where the negative class was downsampled. The results were much better as it can be seen below, however, this approach is by no means valid, since the model should always be evaluated on conditions that are as close to the real life production as possible.

4.5.2 Experiment 2: GreekBERT finetuning - Minority (positive) class resampling with replacement

The above experiment is repeated with resampling of the minority class with replacement, in order to mitigate the class imbalance by matching the size of the majority (negative) class and provide the model with more positive examples, hoping that it will learn to generalize better. In the training process, several optimizations are used, like:

- Accumulating the gradients over multiple batches before performing the weights update.

- Using mixed precision, combining both full precision (`FP32`) and half precision (`FP16`).
- Applying a warm-up period for the learning rate, initializing it to 0 and gradually increasing it to `5e-5` during the first 10% of the total training steps.
- Using weight decay regularization to prevent overfitting by introducing a penalty to the loss function, discouraging large weights.

The results are displayed below:

```
Accuracy: 0.972
Precision: 0.064
Recall: 0.308
F1 Score: 0.106
ROC-AUC: 0.734
PR-AUC: 0.083
R-Precision: 0.196
Recall@20: 0.030
Precision@20: 0.500
```

4.5.3 Experiment 3: Contrastive Learning

The third experiment includes making use of the contrastive learning technique. Such methodology can be really helpful in highly unbalanced datasets, because it can help the model learn fine differentiating details between the positive and negative instances [77]. In more details, contrastive learning employs the contrastive loss to learn the difference of dissimilar pairs by moving them away in the embedding space, or learn the similarities of similar pairs, thus bringing them closer in space. This helps in situations where the classifier needs to learn to distinguish the positive from the negative examples, but there are not so many positives to train on. Furthermore, by re-using the same positives to create similar and dissimilar pairs, the learning signal of the few positive examples is essentially amplified, and the model is supported in learning nuanced patterns of the positive class that it would

not be able to explore otherwise. In this way, the model can search for identical characteristics in future unseen instances of interest, thus becoming an even better ranker with a higher retrieval capability.

Two approaches to create these similar (positive) and dissimilar (negative) pairs are followed:

Approach 1

In the first approach, the evaluation on the validation dataset that was performed during the first experiment is used to identify the true and false positives and true and false negatives of the validation dataset. Using them, the positive (similar) and negative (dissimilar) pairs are constructed, by executing the following steps of the method `create_contrastive_pairs()`:

1. First, the embeddings for each of the four groups (true/false positives/negatives) are calculated, so that we have them in hand, instead of having to re-generate them again and again. The steps to create the embeddings are the following:

   (a) The `_encode_sentences()` method is invoked, which takes the list of sentences of each group in batches, so that the available memory is used efficiently, the calculations can be parallelized, the data transfer overhead is reduced and the matrix operations can lead to faster calculations.

   (b) The BERT tokenizer is used to turn each sentence in the batch to tokens that the BERT model can work with. Truncation or padding is applied, so that every sentence has the same length (equal to max_length=512 in this case).

   (c) Gradient operations are disabled in order to speed up computations, since we are only using inference when creating the embeddings, and the tokenized batch is fed into the embedding model, which is BERT in this specific case.

   (d) Last, the embeddings of each token in every sentence of the batch are extracted, and mean pooling is applied to convert each sentence's infor-

mation into a single vector embedding, making also sure that the padding tokens are excluded by masking the hidden state. Mean pooling is better suited for checking the similarity and distances to create the contrastive pairs than the pooler_output of BERT, which is more useful for classification tasks. All embeddings across all batches are returned into a single numpy array, so that they can be used to create the contrastive pairs.

2. Then, the positive pairs for the below combinations are calculated:

   (a) True Positives - False Negatives

   (b) True Negatives - False Positives

   Cosine similarities are used to determine whether a pair should be included in the positive pairs, as well as a lower and an upper threshold between which a cosine similarity should be located, in order for this pair to be taken into account. To calculate these boundaries, we follow this logic:

   (a) Cosine similarities are by definition in the range [-1, 1]. The thresholds could generally be static or dynamic. Initially, an assumption was made that the cosine similarities are following the normal or at least a bell-shaped curve. In this way, the following dynamic thresholds were used (which they can also adapt to variations in data):

      i. As the lower bound, the $\max(\mu + 1\sigma, \cos(45°))$ was chosen, because the quantity of $\mu + 1\sigma$ and above filters out 84.13% of the lower similarities (again, assuming normal distribution). However, there is no guarantee that this quantity is above 0 or sufficiently large to be related to meaningful similarity. For this reason, the angle between two vectors would have to be at most 45° to be considered similar.

      ii. The upper bound is obviously 1.0, since in this case the angle between the two vector embeddings is 0°, which means that the two vectors have the same direction.

   (b) However, upon closer inspection, it was revealed that the cosine similarities distribution is not necessarily the normal distribution, but rather

the cosine similarities tend to cluster around certain values due to specific geometrical aspects in the embedding space. As a simple example to further illustrate this, vectors that belong to the same class are more probable to have similarity closer to 1, than vectors that belong to different classes, whose similarity would be closer to 0 (orthogonal vectors are the most common in high-dimensional spaces) or less. Furthermore, a normal distribution is defined over , which comes into contrast with the bounded nature of the cosine similarities. In order to test the cosine similarities distribution for normality, further actions would have to be taken, like running a goodness-of-fit (normality) test based on p-values, or plot the data in a histogram to visually evaluate if the distribution is close to the normal one. Nevertheless, since the most logical assumption is that the cosine similarities distribution would be likely multimodal or skewed, static thresholds are eventually used:

i. As the lower threshold, $\cos\left(\frac{\pi}{4}\,\text{rad}\right)$ is used, because two similar vectors would at most form a 45° angle between them.

ii. As the upper threshold, 1 is used, indicating vectors that follow the same direction.

3. Last, we calculate negative pairs for the combinations:

   (a) True Positives - True Negatives

   (b) True Positives - False Positives

   (c) True Negatives - False Negatives

   (d) False Positives - False Negatives

This time we are using cosine distances, which can take only values in the range [0, 2]. It is of interest to say here that initially, euclidean distances we used. However, upon more careful examination, it became apparent that euclidean distances are affected by the magnitudes of the embeddings, especially for non-normalized embeddings like the ones generated by the BERT model. Moreover, these distances are also affected by the position of each embedding in the vector

space. Instead, in this application where contrastive pairs need to be created, angular similarity is of higher interest because in NLP tasks it better tracks the semantic similarity between texts. What is more, for the negative pairs construction, normal distribution of the distances was again initially assumed, and for this reason, $\mu + 1\sigma$ was used as the lower bound, while $\mu + 2\sigma$ as the upper bound. However, as already discussed above, this assumption is not sound and is somewhat computationally expensive to be verified, so instead static thresholds are once again used, and specifically:

(a) 0.5 for the lower bound, which corresponds to the $\pi/3$ rad angle between two vectors, and

(b) 1.5 for the upper threshold, which corresponds to the $2\pi/3$ rad angle.

In this way a variant of semi-hard negatives is implemented, a technique that will be discussed more in the second approach of contrastive learning. For now, it suffices to say that the negative pairs are based on examples that are not trivially close in the embedding space, but still distant enough to be considered dissimilar.

Note that the output of the `create_contrastive_pairs()` method are not only the contrastive_pairs, but also the tokens of the sentences (`input_ids` and `attention_masks`), so that the sentences of each pair do not need to be tokenized again when fine-tuning the `GreekBERT` model on the contrastive learning task. The contrastive_pairs dataset, following the methodology described above, turned out to be extremely large:

- 827703 negative pairs (label 0)
- 294109 positive pairs (label 1)

This dataset would require extreme computing resources and memory, so that the GreekBERT could be fine-tuned using the contrastive learning. Even with small batches and accumulation steps, the training time would be unbearably high, based on the available resources to run this experiment. For these reasons, the contrastive_pairs dataset was downsampled, and each class (positive/negative pairs)

was determined to consist of 30000 examples. This is a logical number to let the model pre-train using the contrastive learning, because, as it is described below, the new contrastive_model is then further fine-tuned again on the classification task, before the final evaluation on the unseen data. So now the dataset which consists of the contrastive pairs is ready, and the BERT language model has to be fine-tuned and evaluated once more. The plan to combine contrastive learning with the classification task is the following:

1. First, the base pre-trained `GreekBERT` model is fine-tuned using the contrastive pairs and loss, so that it can learn to separate and align the embeddings space better. Again, mean pooling is used for the outputs of the model, because this pooling method captures better the semantic meaning of a sentence for contrastive learning. This new model could be called `contrastive_model`.

2. Then, this `contrastive_model` is further fine-tuned using the initial training dataset from the dataset preparation on the binary classification task, so that the model can function as a classifier once again, but this time hopefully having also the ability to distinguish the positive from the negative class encoded in its weights, thanks to the contrastive learning. This time we are also using the pooler output of the BERT model, which is optimized for the classification task. The pooler output represents the embedding of the `[CLS]` token, which is created by applying a linear transformation and a tanh activation function on this token.

3. Last, this last model from step [b] is evaluated on the test (unseen) dataset that was created during the dataset preparation.

As a first step of the contrastive fine-tuning, Optuna is once again used to search for the optimal hyperparameter values. The initial values that were discovered during the first experiment cannot just be reused, because now the task is different (contrastive learning vs binary classification), so the model may need less epochs to learn to cluster together similar and separate dissimilar embeddings, larger batch sizes to see a diverse set of positive and negative pairs in each step, potentially less accumulation steps etc. More specifically:

1. `num_epochs (1, 10)`: This wider range helps Optuna to identify if the model converges fast when learning with contrastive loss, or if it needs more epochs

2. `batch_size (32, 64, 128)`: Larger batch sizes can potentially contribute to faster and more stable learning process, but they should also fit in the GPU's memory

3. `accumulation_steps (1, 3)`: If Optuna comes up with larger batch sizes, then the accumulation steps could be fewer, potentially even 1, meaning that the gradients will get updated for each batch

4. `learning_rate (1e-6 to 1e-4)`

5. `margin (0.3 to 0.8)`: The margin is used in the formula of contrastive loss and controls the separation that the negative pairs should have between them. A margin of around 0.5 works well the most times in applications where cosine distances are used.

Then with these optimized hyperparameters values, we are training the base Greek-BERT model using the contrastive loss:

$$L_{\text{contrastive}}(\text{cosine}) = \frac{1}{2N}\sum_{i=1}^{N}\left[y^i d_{\text{cosine}}\left(x_1^i, x_2^i\right)^2 + (1-y^i)\max\left(0, m - d_{\text{cosine}}\left(x_1^i, x_2^i\right)\right)^2\right] \tag{4.1}$$

where the cosine distance:

$$d_{\text{cosine}}(x_1^i, x_2^i) = 1 - \frac{x_1^i \cdot x_2^i}{\|x_1^i\|\|x_2^i\|} \tag{4.2}$$

Essentially, for $y^i = 1$ (positive pairs), the first term is retained, so in order to minimize the loss, the cosine distances d_{cosine} between positive (similar) embeddings need to be minimized, bringing them closer in terms of direction. On the other hand, when $y^i = 0$ (negative pairs), if the cosine distance d_{cosine} between the dissimilar embeddings is less than the margin m, the distance has to be maximized, thus pushing dissimilar pairs far away in the embedding space, in terms of their direction.

During the training, mixed precision is used for higher efficiency, lower memory requirements and faster convergence. Mixed precision essentially casts some operations, like matrix multiplications, to 16-bit precision, while it keeps some others which are sensitive to numerical instability to 32-bit precision (like softmax). It also scales the loss by multiplying it with a factor, to ensure that gradients are large enough during backpropagation even with 16-bit precision, and in this way to avoid underflow issues.

Then, this `contrastive_model` which has been trained using the optimized hyperparameters by Optuna and the contrastive learning, is further trained on the classification task. For this reason, the initial training dataset (that was created during the Dataset Preparation) is split into `training_dataset*` and `validation_dataset*`, using a 80-20 stratified split. These new `train*` and `validation*` datasets will be used to further train the `contrastive_model`. Then, the exact same evaluation process as described in the Experiment 1 has been followed.

The results are depicted below:

```
Accuracy: 0.994
Precision: 0.097
Recall: 0.0776
F1 Score: 0.086

Confusion Matrix:
Predicted      0    1    All
True
0          87683  243  87926
1            309   26    335
All        87992  269  88261

ROC-AUC: 0.743
PR-AUC: 0.0324
R-Precision: 0.084
Recall@20: 0.0209
```

```
Precision@20: 0.350
```

From these results it is evident that this model, in comparison with Experiment 1, learned the dominant negative class even better, while it seems to struggle more with the positive class which interests us.

The reasons for this result may be due to uninformative contrastive pairs, because it is possible that the static thresholds, while rational, may miss the nuances of the embedding space and do not contribute to the creation of expressive pairs. Furthermore, the downsampling of the contrastive pairs dataset, while it helped to reduce the needs in excessive compute resources and training time, may cost us in variability and in representing sufficiently the positive class. Nevertheless, we continue with the second approach for the contrastive learning, which makes use of a more dynamic approach for creating semi-hard negatives, and which combines the classification and the contrastive loss in the same training process.

Approach 2

The second approach for the contrastive learning involves the concept of semi-hard negatives in the construction of the contrastive pairs, as was originally applied for triplet loss by Schroff et al. [78]. Semi-hard negatives are examples with different labels from each anchor example, and are closer than some positive (same-label) examples, but not too close to confuse the model. Semi-hard negatives have been proven extremely valuable in highly imbalanced datasets. In essence, for each of the misclassified examples of the validation dataset from Experiment 1, positive pairs are created using the examples that have the same label with the misclassified instance, while for the negative pairs, only the instances that meet the following condition are kept:

$$\text{mean}(d(a,p)) < d(a,n) < \text{mean}(d(a,p)) + \text{margin} \tag{4.3}$$

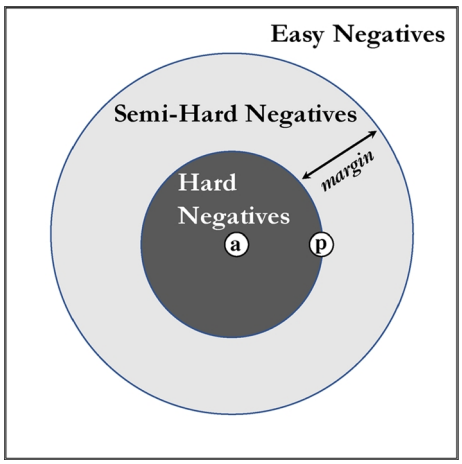
where:

- mean$(d(a,p))$ is the mean cosine distance between the anchor (a) and all positive (p) instances [same label as (a)]. Initially, the minimum cosine distance had been used, but this is prone to outliers and can lead to overly strict negative pairs, so the mean was eventually chosen instead, for a more robust behaviour.

- $d(a,n)$ is the cosine distance between the anchor (a) and a negative (n).

- margin is a hyperparameter defining the maximum "acceptable hardness" for a negative.

Using this equation, only negative examples that fall above the mean distance between the anchor and the positive (same-label) examples, and below this mean distance plus a margin are kept.



Since there were around 550 misclassified examples in the validation dataset, out of which about 380 where of true positive class, while the rest 170 belonged to the true negative class, if all same label and different label examples of the validation dataset (which comprises 12500 examples) were to be used to create the positive and negative pairs for each of the misclassified examples, the resulting contrastive pairs would be over 5'000'000 examples. For this reason, for each of the misclassified examples, eventually only 100 same-label and 100 opposite-label were used to create the positive and negative pairs. The training process is the following, after having created the new contrastive pairs:

1. The model that was fine-tuned and evaluated during Experiment 1 is loaded.

2. This model is trained using a combination of the contrastive loss, based on the mean pooled outputs of the BERT model and the same loss like the previous contrastive experiment, and the classification losses of the two examples of each pair, using the pooler outputs and the label of each sentence in the pair.

3. The total loss is calculated as:

$$\text{total loss} = \lambda_{\text{contrastive}} \cdot \text{contrastive loss} + \lambda_{\text{classification}} \cdot (\text{classification loss}_1 + \text{classification loss}_2) \tag{4.4}$$

Based on the scale of each type of loss, loss weights (λ) can also be used to balance the influence of each loss type. Initially, we start with both weights equal to 1 (equal contribution). Several optimizations that have already been discussed are also used in this training process. Finally, the fine-tuned model is evaluated both with traditional classification metrics and with ranking metrics. The results are depicted below:

```
Accuracy: 0.979
Precision: 0.056
Recall: 0.290
F1 Score: 0.094

Confusion Matrix:
Predicted      0     1    All
True
0          86298  1628  87926
1            238    97    335
All        86536  1725  88261
ROC AUC: 0.723
Precision@20: 0.300
Recall@20: 0.018
R-Precision: 0.188
```

4.5.4 Data refinement and positive synthetic data with LLMs

For the next experiments, a LLM (OpenAI's `GPT-4o` mini) was used to refine the existing texts as well as to produce synthetic data for the positive class. This was done with the purpose of dealing with the class imbalance and also with the poor quality of the Greek texts, in order to help the language models to better learn the positive class, which again is the class of sole interest for this application. Towards this end, the texts from training, validation and test datasets were prepared, by applying the following operations:

- Replace `[xxxx]` and `#########` with `ANONYMIZED`
- Remove extra whitespaces
- Lowercase all the texts

Then, the texts from all datasets are firstly pre-processed and refined in order to contain more propper Greek language. The texts originally contained many abbreviations and were poorly written, including many grammatical and syntactical errors, considering that they were transcripts of phone calls with loan holders written on the fly by call center agents. For this reason, the cost efficient OpenAI's `GPT-4o` model was used, with the following system prompt:

"You are going to receive a Greek text that has been written by a call center agent, as part of a communication with a customer to arrange a non-performing loan. Convert the text into proper Greek without abbreviations, grammar, and spelling mistakes. Do not change the `anonymized` *word in the texts. Regarding the abbreviations, you have to understand from the context, examples:* `κτχ` *or* `κτχ.` *may mean* `κάτοχος`*,* `συζ.` *may mean* `συζυγος` *etc. Make sure you do not omit any text information, you do not add anything that you may assume and you do not hallucinate! Return only the transformed proper Greek text, with no explanation, headline, numbering etc., because this will be appended automatically in a Python list. So make sure you return the text with no line breaks etc."*

The new datasets were saved in corresponding csv files, so that they could be

used by further experiments. Examples of the transformation that were achieved by the `GPT-4o` mini model are the following:

Initial transcript

"anonymized οφλ απο αποκρυψη: ενημ οτι anonymized δεν εχει επιστρ στην anonymized & πωσ εξακολουθει να ειναι anonymized σε πιθ ρυθμ. εξηγησα οτι ηδη εχει παρελθει μεγ.anonymized διαστημα & θα πρεπει αμεσα να υπαρξει σαφη απαντηση για προθεσεισ του καθωσ anonymized οτι υποθεση π σε οφειλετη: anonymized δβγ, anonymized ξανα - δα ...σε anonymized/χο: anonymized δα - μηνυμα σε τφ σε οφειλετη: anonymized δβγ, anonymized ξανα κ anonymized το ιδιο σε οφειλετη: anonymized δβγ, anonymized ξανα - δα ...σε anonymized/χο: anonymized δα - μηνυμα σε τφ anonymized δβγ, anonymized ξανα - απεν/νο anonymized δβγ, anonymized ξανα - απεν/νο anonymized οφλ ενημ δεν μπορεσε να προβει στην προκατολη ποσου €anonymized σε anonymized ...anonymized, οπωσ ειχε δεσμευτει καθωσ συγγ προσωπο που

We can see that there many spelling errors and abbreviations. Still, the LLM managed to capture the semantic meaning of the text and made the necessary word replacements:

Transcript refined by gpt4o-mini

"anonymized οφειλέτης από απόκρυψη ενημέρωσε ότι anonymized δεν έχει επιστρέψει στην anonymized και πως εξακολουθεί να είναι anonymized σε πιθανή ρύθμιση. Εξήγησα ότι ήδη έχει παρέλθει μεγάλο anonymized διάστημα και θα πρέπει άμεσα να υπάρξει σαφής απάντηση για τις προθέσεις του, καθώς anonymized ότι υπόθεση που σε οφειλέτη: anonymized δεν βγήκε, anonymized ξανά - δα ...σε anonymized/κάτοχο: anonymized δα - μήνυμα σε τηλέφωνο σε οφειλέτη: anonymized δεν βγήκε, anonymized ξανά και anonymized το ίδιο σε οφειλέτη: anonymized δεν βγήκε, anonymized ξανά - δα ...σε anonymized/κάτοχο: anonymized δα - μήνυμα σε τηλέφωνο anonymized δεν βγήκε, anonymized ξανά - απενεργοποιημένο anonymized δεν βγήκε, anonymized ξανά - απενεργοποιημένο anonymized οφειλέτης ενημέρωσε δεν μπόρεσε να προβεί στην προκαταβολή ποσού €anonymized σε anonymized,

The refinement and standardization of the Greek text through this process is obvious. As it will be shown in the results below, this step was critical in improving the results that were achieved by fine-tuned language models. Through cleaning the texts, the data noise was greatly minimized, thus reinforcing the models to only learn the real linguistic and semantic relations instead of being distracted by the noise from the errors and inconsistencies.

Then, positive synthetic examples are generated using again the GPT-4o model and by providing sample texts from the already existing positive examples. The prompt that is used for this case is the following:

"You are going to receive Greek texts that have been written by call center agents, as part of communications with customers to arrange nonperforming loans. Generate 10 new examples of text data that must EXACTLY mimic the style and expressions

of the provided texts. As you will see, they are not proper Greek, they are not always syntactically and grammatically correct, and they contain many anonymized sensitive data.These are texts written roughly and hastily by the agents while they are on the call. All these rules you MUST also follow. Do not make the generated examples too easy. Make sure you enter each text in a new line, because they are going to be split by linebreakers and appended to a Python list, and do not create any numbering, headlines etc., return ONLY the generated texts! Here are the examples:"

The responses from the `GPT` model are post-processed in order to be flattened, and they are saved to a new enriched training dataset, which now has a better balance between the positive and negative class, along with refined texts. It is expected that the cleaner and more structured training dataset, now including more positive examples -together with the synthetic ones, can enhance the ability of the models to generalize better and have more accurate predictions on data that they have not encountered before. Moreover, in the case of models such as `GreekBERT` and `Meltemi`, such models are pretrained on massive datasets that contain high-quality text. Thus, if the training data is more similar in terms of quality and style to the pretraining corpus, then the models will be able to use the knowledge encoded in the pretrained model more efficiently.

4.5.5 Experiment 4a: LoRA for GreekBERT with class-weights fine-tuned on GPT refined & synthetic dataset

In this experiment, the `LoRA` (Low-Rank Adaptation) technique [41] is used for efficient and memory-optimized fine-tuning of the `GreekBERT` model on the classification task. `LoRA` essentially freezes the model's layers and uses low-rank matrices instead, thus significantly reducing the number of parameters that need to be updated, while achieving similar performance to a full fine-tuning process. Specifically, `LoRA` is configured to inject trainable low-rank matrices in three layers of the `BERT` model: `query`, `value` and `output.dense`. These three layers play a central role in `BERT`'s multi-head attention, thus enabling `LoRA` to have a big impact on the model, when determining the relevance that the inputs have on each other or when con-

necting intermediate representations to the specific task in hand; all these without having to train a huge amount of parameters, saving in this way computational resources while maintaining valuable internal knowledge of the layers. The math formula of `LoRA` is the following:

$$W' = W + \alpha \cdot \Delta W \tag{4.5}$$

where W is an original weight matrix of each layer that is included in `LoRA`'s configuration and which is frozen during fine-tuning. ΔW, on the other hand, is the low-rank matrix that `LoRA` learns, expressed as:

$$\Delta W = A \times W \tag{4.6}$$

where A is of shape (d, r), and B is of shape (r, d).

Regarding d and r, d is the initial dimension of the frozen weight matrix W, and r is the rank used by `LoRA`. Finally, α is a scaling factor that controls the effect of the updates. From the mathematical formula, it is clearly evident that when $r \ll d$, `LoRA` has to train $2 \cdot d \cdot r$ parameters instead of the d^2 parameters of the initial weight matrix.

The `LoRA` config parameters that are used in this experiment are:

- `task_type=TaskType.SEQ_CLS`
- `r=16`
- `lora_alpha=32`
- `lora_dropout=0.1`
- `target_modules=["query", "value", "output.dense"]`

The type of task is sequence classification, while r has been chosen to be 16, a number which balances maintaining enough expressivity for the model, while keeping

the fine-tuning process memory efficient, fast and modular. The `lora_dropout` is a regularization term to avoid overfitting, while the `lora_alpha` as already described, is a scaling factor that affects the importance of the newly learned parameters to the total model's behaviour. It is very interesting to also note here

$$r_{\text{effective}} = \frac{r}{a} = \frac{16}{32} = 0.5 \tag{4.7}$$

which represents the `LoRA`'s parameters effect on the model. Higher values of $r_{effective}$ are generally required in situations where more drastic updates are required in regards with the model's behaviour, while lower values are associated with subtler fine-tuning and higher preservation of the internal knowledge. In the application in hand, the 0.5 value is appropriate because it shows that the model has the capacity to learn the new classification task, while keeping much of its pre-existing knowledge, which is very beneficial in our case. In this particular experiment, we are also using weighted loss based on the `compute_class_weight()` by `scikit-learn` and using the balanced setting to assign weights inversely proportional to class frequencies. Some other settings for training our model, which are also shared with the next experiments, include the use of mixed precision training, gradient accumulation every four batches, and early stopping to avoid overfitting. The results are depicted below:

```
Accuracy: 0.973
Precision: 0.326
Recall: 0.428
F1 Score: 0.370
ROC AUC: 0.832
PR AUC: 0.354
Precision@20: 0.600
Recall@20: 0.036
R-Precision: 0.374
```

4.5.6 Experiment 4b: LoRA for GreekBERT with class weights and Learning Rate decay on GPT refined & synthetic dataset

The same experiment as before is repeated with layer-wise learning rate decay, as well as a learning rate scheduler. Essentially, there are different learning rates for different layers of the model. One is with learning rate $1e{-}5$ for the `bert.encoder.layer` parameters, and the other is $2e{-}5$ for the parameters of the final classification layer. As far as the learning rate scheduler is concerned, this gradually reduces the learning rate during the training process, by using a linear approach and after a warm-up period. This whole strategy could potentially help in increasing the model's performance by permitting a slower fine-tuning for the model's parameters, in order to preserve the internal knowledge, while in the meantime it enables the classification head to learn the task quicker with a higher learning rate.

The results of this approach are depicted below:

```
Accuracy: 0.976
Precision: 0.328
Recall: 0.432
F1 Score: 0.373
ROC AUC: 0.834
PR AUC: 0.368
Precision@20: 0.600
Recall@20: 0.036
R-Precision: 0.378
```

4.5.7 Experiment 5a: Using Optuna to optimize the decision threshold of GreekBERT with Linear Learning Rate decay schedule on GPT refined & synthetic dataset

In the next experiment, Optuna is used to determine the optimal decision threshold based on the R-Precision metric, running 10 trials for the range $(0.0, 0.5)$. This range is used, because we know that the model, even with the refined and synthetic positive examples, will find it more difficult to predict the positive than the negative class, so it only makes sense that the best threshold would be somewhere in this range. We are also using early stopping with patience equal to 3 epochs, in order to terminate the Optuna's trials earlier if no further improvement is observed. Moreover, a linear learning rate decay is used with warm-up period equal to 10% of the total learning steps, during which the learning rate increases from 0.0 to $5e{-}5$. The optimal threshold is found to be 0.26. Based on this decision threshold, the evaluation results are:

```
Accuracy: 0.988
Precision: 0.344
Recall: 0.458
F1 Score: 0.393
ROC AUC: 0.842
PR AUC: 0.374
Precision@20: 0.600
Recall@20: 0.036
R-Precision: 0.418
```

4.5.8 Experiment 5b: Using Optuna to optimize the decision threshold of GreekBERT with Class-Weighted Loss and Cosine Learning Rate decay schedule on GPT refined & synthetic dataset

The previous experiment is repeated, but this time a class-weighted loss is used to offset the class imbalance, and also a cosine learning rate decay scheduler with warm up period and no decay on the bias and layer normalization parameters. The cosine function helps in decreasing the learning rate slower at the first epochs (initialized at $5e{-}5$), with faster decrease in the middle. This contributes to a finer grained control of the fine-tuning process, potentially leading to better results. The optimal decision threshold that was obtained following this approach was 0.26 again. The evaluation metrics ended up being the same with the previous experiment 5a.

4.5.9 Experiment 6: FinBERT LoRA fine-tuning with class-weighted loss and learning rate scheduler on GPT refined & synthetic dataset

For this experiment FinBERT [31] was used on the GPT refined training dataset that includes the synthetic positive examples too. FinBERT is a BERT flavor that has been pre-trained on financial domain datasets. A weighted loss function was used, along with a learning rate scheduler with a warm-up period of 0.1 (meaning that the learning rate increased from 0.0 to $5e{-}5$ during the first 10% of the training steps), while afterwards the learning rate started to decrease. Furthermore, the weight_decay parameter effectively applied a L_2 regularization on the model's weights, in order to encourage the model to learn smaller weights and improve generalization. The fine-tuning process which lasted for 10 epochs using LoRA gave the following results, for the best model based on the R-Precision:

```
Accuracy: 0.968
Precision: 0.188
```

Recall: 0.248

F1 Score: 0.214

ROC AUC: 0.792

PR AUC: 0.198

Precision@20: 0.400

Recall@20: 0.024

R-Precision: 0.194

4.5.10 Experiment 7: XLM-RoBERTa LoRA fine-tuning with class-weighted loss on GPT refined & synthetic dataset

This experiment involved fine-tuning the XLM-RoBERTa [79] model on the refined dataset with the synthetic positive examples. The configuration parameters of the fine-tuning using the LoRA approach, which again lasted 10 epochs, were pretty much the same with experiment 6 (weight decay was 0.01, warm-up ratio was 0.1, batch size was 32 for training and 64 for evaluation, and gradient accumulation steps were 4). The results for XLM-RoBERTa, using a class-weighted loss, were the following for the best model based on the R-Precision:

Accuracy: 0.962

Precision: 0.168

Recall: 0.224

F1 Score: 0.192

ROC AUC: 0.776

PR AUC: 0.188

Precision@20: 0.400

Recall@20: 0.024

R-Precision: 0.184

The same experiment was repeated with the same configuration but without taking the class weights into consideration. The results this time were the following, again for the best model based on the R-Precision:

```
Accuracy: 0.980
Precision: 0.088
Recall: 0.186
F1 Score: 0.120
ROC AUC: 0.824
PR AUC: 0.174
Precision@20: 0.400
Recall@20: 0.024
R-Precision: 0.162
```

Finally, `XLM-RoBERTa` was also finetuned without using the `LoRA` method, and with no class weights. The results were the following:

```
Accuracy: 0.980
Precision: 0.088
Recall: 0.186
F1 Score: 0.120
ROC AUC: 0.824
PR AUC: 0.174
Precision@20: 0.400
Recall@20: 0.024
R-Precision: 0.154
```

4.5.11 Experiment 8: `Meltemi` (`Mistral 7B`) fine-tuning with `LoRA` using class weights on `GPT` refined & synthetic dataset

This experiment involved fine-tuning the `Meltemi` model [80], the first open source LLM model for the Greek Language which is based on the `Mistral` LLM with 7 billion parameters. Due to the size of the model and the computational requirements to fine-tune it, this time `LoRA` added real value, by targeting the query and value projection layers and considerably decreasing the parameters to be learned. Again,

class-weighted loss, weight decay, warm-up ratio and this time also Early Stopping with patience equal to 3 epochs were used. The results of the best model based on R-Precision were:

```
Accuracy: 0.972
Precision: 0.342
Recall: 0.496
F1 Score: 0.405
ROC AUC: 0.832
PR AUC: 0.382
Precision@20: 0.700
Recall@20: 0.042
R-Precision: 0.384
```

4.5.12 Experiment 9: Fine-tuning `Llama 3.1 70B` with `QLoRA`

In the final experiment, an attempt to finetune a large language model, `Llama 3.1 70B`, ends up with some interesting methodologies to fit such large models in a single GPU. Towards this goal, `QLoRA` [81] is utilized, which offers a much smaller quantized model that can be fine-tuned with the `LoRA` technique, sacrificing very little performance and providing comparable evaluation results with the full initial model. The results of the fine-tuned Llama using `QLoRA` (without class-weighted loss) are the following:

```
Accuracy: 0.984
Precision: 0.126
Recall: 0.208
F1 Score: 0.157
ROC AUC: 0.832
PR AUC: 0.162
Precision@20: 0.400
Recall@20: 0.024
R-Precision: 0.176
```

Chapter 5

Conclusions and Future Work

5.1 Results discussion

The current study underwent many obstacles and challenges from the very beginning, when well-established machine learning models were trained to form a basis for the subsequent experiments. Very soon it became apparent that the extreme class imbalance of the dataset would constitute one of the most important hurdles towards achieving good classification performance.

Even after applying resampling methods, class weights, and simple data augmentation techniques such as SMOTE, the predictive capabilities of the models regarding the positive class did not seem to improve. Even the more advanced deep language models, like GreekBERT, which have been repeatedly shown to achieve state-of-the-art assessment on classification tasks, did not seem to be able to handle the available dataset.

Further techniques like contrastive learning, hyperparameter optimization, regularization methods such as weight decay, and approaches to avoid local minima, like learning rate decay scheduling, did little to nothing in enhancing the models' generalization and recognition of the positive class.

Thus, it gradually became more and more obvious that apart from the huge shortage of positive examples, another factor that deteriorated the models' chances of displaying adequate performance in the task of loan arrangement prediction was

the overwhelming intrinsic noise of the available action comments.

After manual inspection of the texts, one thing was certain: if a native Greek speaker faces difficulties in managing to comprehend the meaning of the sentences and the pattern to predict loan arrangement and installment payment in the next two months, then it would be much more cumbersome for a language model to manage to unravel the signals that lead to the positive instances. Non-standard expressions and abbreviations, financial domain terminology and slang, intermittent phrases or sentences caused major frustration in Greek-specialized models like GreekBERT and Meltemi.

In addition to this situation of low data quality, the weak supervision that was applied during the dataset construction must not be overlooked. After all, it has been shown by Zhu et al. [65] that BERT is particularly susceptible to noise produced by week supervision. The action and decision of arranging a loan and paying an installment or not during a month, can be quite irrelevant with the comments written by an agent two, three or four months ago. Since three months historical comments were used at each given monthly snapshot, the existence of a meaningful connection between the texts and the manually assigned labels is - at the very least - wishful thinking.

Another point of consideration would be the extended length of the texts, due to the three months historicity in every month. It has been shown by related work completed by Lu et al. [55] that long documents can give rise to increased noise because of irrelevant information. For this reason, a limitation of the current dissertation was the fact that there was no experimentation with shorter periods of history in the agents' written comments. Furthermore, when a classic feature selection technique was applied, such as the χ^2, the performance of the models improved, a fact which shows that there was redundant noise in the texts, which was filtered by keeping the most relevant terms.

Last, the limited computational resources that instructed techniques like quantization, mixed precision, and gradient accumulation with smaller batches, coupled with the lack of specialized models for financial terminology in Greek further exploded the possible configuration combinations and prevented the discovery of an

effective and simple solution.

Nevertheless, the study managed to accomplish something innovative, at least according to the author's knowledge about the existing body of work conducted by the Greek NLP community. The experiments clearly demonstrated that the utilization of a modern Large Language Model for the standardization and refinement of noisy data, along with the generation of synthetic positive examples, had a profound impact on the evaluation of the fine-tuned models. Specifically, this text cleaning, refinement, and augmentation process resulted in approximately over 100% increase in the R-Precision achieved by GreekBERT, compared to its performance on the initial dataset. However, it should be stated that more experimental runs are required to determine the precise effect of this approach and rule out factors due to randomness, sampling, batch selection, etc. This is another limitation of the current study, as the limited available resources and the cost of fine-tuning such models did not permit a complete statistical evaluation of the results.

Last but not least, the fine-tuned Greek language models GreekBERT and Meltemi presented quite interesting performance when measured as rankers, especially after the text refinement and augmentation by the LLM, managing to retrieve around 40% of the total positives in the unseen data, in the same number of examples (`R-Precision`). However, more future work is required to further examine the ranking capabilities of such models in such highly imbalanced and noisy scenarios.

5.2 Future work

Some of this study's challenges and limitations that have already been discussed provide interesting opportunities for future work, too.

First of all, in terms of creating more value for call center applications and predictive analytics, a future integration of the current work's machine learning experiments with an Automatic Speech Recognition (ASR) system could yield results that are in many ways more robust and beneficial than the current ones. Such an end-to-end application would offer unstructured data of much greater quality, by automatically generating call transcripts at the time of the communications. This

would also create the chance to perform a contrastive analysis and determine the degree to which the inherent data noise affected the models in the current thesis.

Moreover, the current dissertation was one of the first (based on the citations) works that employed and evaluated the Meltemi Greek LLM on a traditional NLP application like binary text classification. Meltemi has been a very interesting initiative to develop the first open-source Greek LLM, and it is beyond any doubt that the Greek NLP community has to focus all their strengths on further refining models like this in the short-term future. During the experiments of this thesis, the absence of specialized Greek models for the financial or credit management sector played a pivotal role for the final outcome of the study, which remained underperforming. In the recent past, some attempts have taken place towards creating specialized NLP models for the Greek legal domain [47]. These efforts need to intensify and extend beyond the academic background, inviting financial institutions, governmental bodies, and companies from the tech private sector in a strategic collaboration.

Towards this direction of enhancing the Greek models, the text standardization and augmentation approach that was performed in this study using a proprietary LLM needs to be implemented and executed with an open-source LLM, ideally a Greek one. Special care should be given for the matters of data privacy, when internal or sensitive data are fed to such models, as well as results interpretability. These two subjects, although out of scope for the current thesis, gain increasing importance day-by-day, and are tightly connected with security considerations too, when someone decides to use one of the available LLMs. For example, working with personal identifiable data (even if they are anonymized or masked) is often accompanied by non-disclosure agreements and very strict regulation about sharing such information.

Moreover, future work should include other techniques of data augmentation and cleaning, in order to explore whether the approach using a LLM was indeed effective and groundbreaking, or simpler and more cost-effective approaches could also bring similar results. After all, the implemented approach for text refinement and synthetic data of this thesis also included some prompt engineering, another active research area which needs further investigation to define a systematic approach that

is robust and well-understood.

Last but certainly not least, a major fact that has been evident throughout this whole MSc thesis was that the acquisition of a good dataset is most of the times a decisive first step towards the solution of a given machine learning problem. This highlights the need for high quality, curated Greek datasets that will be openly available - much like the Greek language models have to be - and will empower individuals, researchers, students and companies to grow in this rapidly evolving AI landscape and implement cutting-edge AI technologies and applications that will form a future with equal and fair opportunities for everyone.

Appendix A

Python libraries and frameworks

The code that accompanies the current thesis is publicly available at:

`https://github.com/iliacube/loan_arrangement_prediction_with_nlp.git`

Polars

Polars is an efficient DataFrame library designed for fast large datasets operations and transformations, where Pandas might face performance issues. One of the most important features that differentiates Polars is its support for LazyFrames and lazy evaluation. More specifically, unlike the eager evaluation approach that pandas follows (where operations are executed immediately), LazyFrames act essentially as "query plans" that define a series of transformations, but they delay execution until an explicit command is met to collect the results. This is very important because it allows Polars to optimize the entire calculations chain before the execution, thus it is especially useful in workflows that involve multiple transformations and operations, as it minimizes unnecessary processing and mitigates memory requirements. The query optimization techniques that are used include among other predicate pushdowns, projection pushdowns, and caching. Polars is also designed to facilitate parallelism. When executing a query, Polars can distribute tasks across multiple CPU cores, enhancing performance on large datasets.

Scikit-learn

Scikit-learn [82] is an open source machine learning library which is widely used in Python. It offers a number of efficient APIs for classification, regression, clustering, dimensionality reduction, model selection and preprocessing. Based on NumPy, SciPy, and matplotlib, scikit-learn is flexible and powerful while at the same time easy to learn and use, which is why it is popular among the beginners as well as the experts in the field of machine learning. It is also actively used in both academic and industrial contexts for different applications, including image processing, natural language processing, etc.

Tensorflow

TensorFlow [83] is an industry-leading, open-source software library for machine learning and artificial intelligence. Developed by Google, it offers users the capability of developing as well as training different kinds of machine learning models such as deep neural networks. TensorFlow is one of the most popular open-source platforms that are used in both research as well as industrial applications for tasks such as image recognition, natural language processing, and predictive modeling among others. It makes use of a flexible architecture that can be deployed on a wide variety of platforms including desktops and mobile devices.

Optuna

Optuna [75] is an open-source optimization framework designed to automate the process of hyperparameter tuning in machine learning and deep learning applications. It provides an efficient, flexible, and user-friendly approach to search for optimal hyperparameters, leveraging both Bayesian optimization and other search strategies to streamline the tuning process. In this project, Optuna was utilized to optimize key hyperparameters for fine-tuning the GreekBERT model on the classification and contrastive learning tasks that are described below. Optuna's automated, guided search improved the efficiency of finding hyperparameter combinations that

enhanced model performance, allowing for a systematic and reproducible optimization process.

Transformers

The Transformers library [84] is provided by Hugging Face, it is open-source and it is used in a variety of settings and applications, such as computer vision, audio, multimodal AI, generative AI etc. In the current application, the state-of-the-art multilingual or specifically Greek language pre-trained NLP models (Greek BERT, meltemi, Llama 3.1 70B etc.) were used to be fine-tuned on custom datasets and evaluated on the classification task, with custom tokenization techniques and even training losses, using PyTorch with little development effort and integrating seamlessly with other tools like the Peft library, while hiding away much of the underlying complexity.

PyTorch

PyTorch [85] is a powerful and flexible open-source library, which supports a wide range of deep learning workflows, ranging from simple feed-forward networks to even more complex architectures like generative adversarial networks (GANs), convolutional neural networks (CNNs) and transformers. It comes with various pre-built tools that allow GPU acceleration for scalable and efficient training. In our experiments, PyTorch formed the backbone of our model training pipeline. It was used to fine-tune GreekBERT and integrate custom components like LoRA (Low-Rank Adaptation) layers for efficient parameter tuning. We leveraged PyTorch's support for mixed precision training with its AMP (Automatic Mixed Precision) feature, which optimized GPU memory usage and reduced training time. PyTorch's flexibility and extensive ecosystem, including libraries like torch.nn for loss functions and layers, were critical in implementing custom functionalities, such as weighted loss for imbalanced datasets and the integration of the PEFT library for parameter-efficient fine-tuning.

Peft

The Parameter-Efficient Fine-Tuning (PEFT) library [86] is a Python library which assists with the fine-tuning of large pre-trained models in an optimized and simplified way. Towards this end, PEFT uses methods that require fewer parameters to be trained than more traditional techniques, thus reducing computational and memory requirements, while retaining a high performance. In the experiments that have been conducted below, Peft was used to apply LoRA (Low-Rank Adaptation) on several different language models, as well as QLoRA (Quantized Low-Rank Adaptation) on a large language model.

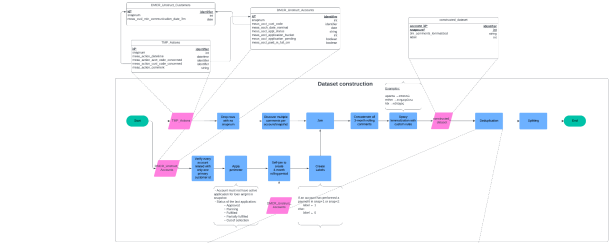
BitsAndBytes

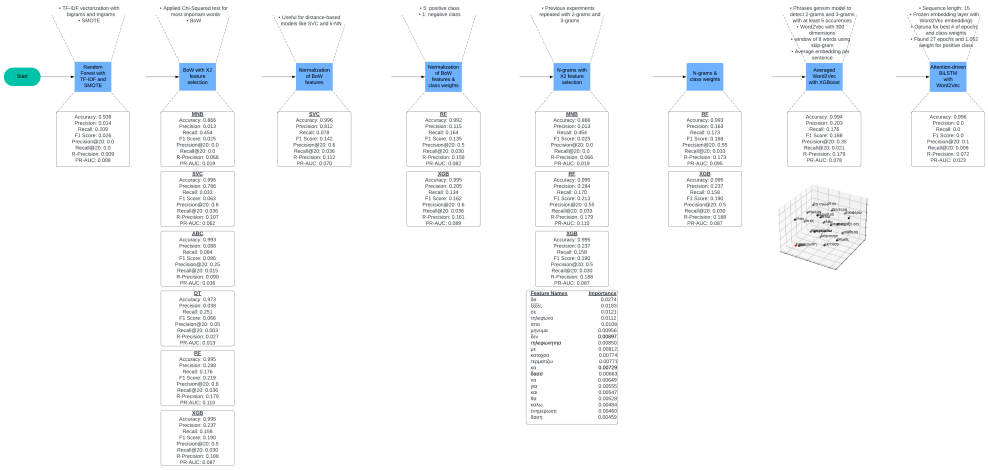
BitsAndBytes is a Python library which was released by Facebook Research [87] with the primary purpose of decreasing the memory footprint of the large language models (LLMs) in order to make them smaller in size. The main concept behind it is quantization, which can be described as a method of reducing the numerical precision of the model weights and activations. This approach makes it possible to run LLMs even on consumer-grade hardware, thus enabling the development of larger and more complicated models. Some of the tools that BitsAndBytes offer include 8-bit optimizers, matrix multiplication and quantization functions, all of which eventually render BitsAndBytes a great package for researchers and developers who are working with LLMs.

Appendix B

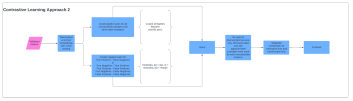
Flowcharts

The following flowcharts describe the datasets creation process, the ML experiments and the language models experiments. SVG files can be found in the github repo: `https://github.com/iliacube/loan_arrangement_prediction_with_nlp.git`

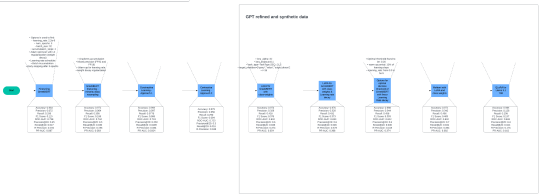





Contrastive Learning Approach 1


Contrastive Learning Approach 2


Data Refinement and Synthetic Data Generation


GPT refined and synthetic data

References

[1] Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. Prentice Hall, 2020.

[2] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.

[3] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the European Conference on Machine Learning*, pages 137–142. Springer, 1998.

[4] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media, 2009.

[5] Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. *Introduction to information retrieval*, volume 39. Cambridge University Press Cambridge, 2008.

[6] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, 1988.

[7] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

[8] David Martin Ward Powers. Evaluation: From precision, recall and f-measure to roc, informedness, markedness & correlation. *Journal of Machine Learning Technologies*, 2(1):37–63, 2011.

[9] Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*, 2018. Accessed: 2024-11-20.

[10] Murugan Anandarajan, Thillai Rajan Srinivasan, and Bhuvaneshwar Anandarajan. *Chapters 4, 5, 6, 8 of Text Mining: Theoretical Aspects and Applications*. Springer, 2004. Accessed: 2024-11-20.

[11] Andrea Gasparetto, Matteo Marcuzzo, Alessandro Zangari, and Andrea Albarelli. A survey on text classification algorithms: From text to predictions. *Information*, 13(2):83, 2022.

[12] Yoav Goldberg. *Neural Network Methods for Natural Language Processing*. Morgan & Claypool, 2017.

[13] GeeksforGeeks. Word embeddings in nlp. Accessed: 2024-11-20.

[14] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2013.

[15] John Doe. A survey of naive bayes machine learning approach in text document classification. *Academia*, 2021.

[16] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[17] David W. Hosmer, Stanley Lemeshow, and Rodney X. Sturdivant. *Applied Logistic Regression*. John Wiley & Sons, 3rd edition, 2013.

[18] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, San Francisco, CA, USA, 2016. ACM.

[19] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[20] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[21] Zachary C. Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

[22] Yoav Goldberg. *A Primer on Neural Network Models for Natural Language Processing*. Morgan & Claypool, 2016.

[23] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[24] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[25] Klaus Greff, Rupesh K. Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232, 2016.

[26] Dani Yogatama, Chris Dyer, Wang Ling, and Phil Blunsom. Generative and discriminative text classification with recurrent neural networks. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1215–1224. PMLR, 2017.

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, pages 5998–6008, 2017.

[28] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2019.

[29] Georgios Koutsikakis, Alexandros Papantoniou, Dimitrios Kamperis, Ilias Chalkidis, and Ion Androutsopoulos. Greekbert: The greeks visit sesame street. In *Proceedings of the 28th Hellenic Conference on Informatics*, 2020.

[30] Teven Le Scao et al. Mistral: Efficient language models for long contexts. *arXiv preprint arXiv:2310.06121*, 2023.

[31] Dogu Araci. Finbert: A pretrained language model for financial communications. *arXiv preprint arXiv:1908.10063*, 2019.

[32] Alexis Conneau et al. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 8440–8451, 2020.

[33] Hugo Touvron et al. Llama 3: Open and efficient foundation models for long contexts. *arXiv preprint arXiv:2310.09132*, 2023.

[34] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1597–1607, 2020.

[35] Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with back-translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 156–165, 2016.

[36] Jason Wei and Kai Zou. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *arXiv preprint arXiv:1901.11196*, 2019.

[37] Colin Raffel et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67, 2020.

[38] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pages 11328–11339. PMLR, 2020.

[39] Schants Kant. Paraphrasing with transformer models: T5, bart, and pegasus. `https://kantschants.com/paraphrasing-with-transformer-t5-bart-pegasus`, 2023. Accessed: 2024-11-20.

[40] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.

[41] Edward J. Hu et al. Lora: Low-rank adaptation of large language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[42] Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023.

[43] Ying Liu, Han Tong Loh, and Aixin Sun. Imbalanced text classification: A term weighting approach. *Expert Systems with Applications*, 36(1):690–701, 2009.

[44] Saad Shaikh, Shah M. Daudpota, Ali S. Imran, and Ziri Kastrati. Towards improved classification accuracy on highly imbalanced text dataset using deep neural language models. *Applied Sciences*, 11(2):869, 2021.

[45] Yixin Li, Guolong Sun, and Yifei Zhu. Data imbalance problem in text classification. In *2010 Third International Symposium on Information Processing (ISIP)*, pages 301–305, 2010.

[46] Bingqing Liu, Weiyan Xu, Yu Xiang, Xiaojun Wu, Liang He, Bo Zhang, and Lei Zhu. Noise learning for text classification: A benchmark. In *Proceedings of the 29th International Conference on Computational Linguistics (COLING)*, pages 4557–4567, 2022.

[47] Christos N Papaloukas and Ilias Chalkidis. Legal text classification based on greek legislation. Master's thesis, National and Kapodistrian University of Athens, School of Sciences, Department Of Informatics and Telecommunications, 2020.

[48] Fadi Thabtah, Suhel Hammoud, Firuz Kamalov, and Amanda Gonsalves. Data imbalance in classification: Experimental evaluation. *Information Sciences*, 513:429–441, 2020.

[49] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[50] Markus Bayer, Marc-André Kaufhold, and Christian Reuter. A survey on data augmentation for text classification. *ACM Computing Surveys*, 55(7):1–39, 2022.

[51] Alejandro Moreo, Andrea Esuli, and Fabrizio Sebastiani. Distributional random oversampling for imbalanced text classification. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 805–808, 2016.

[52] Nitesh V Chawla, Nathalie Japkowicz, and Aleksander Kotcz. Special issue on learning from imbalanced data sets. *ACM SIGKDD explorations newsletter*, 6(1):1–6, 2004.

[53] Nitesh V Chawla, Aleksandar Lazarevic, Lawrence O Hall, and Kevin W Bowyer. Smoteboost: Improving prediction of the minority class in boosting. In *Knowledge Discovery in Databases: PKDD 2003: 7th European Conference on Principles and Practice of Knowledge Discovery in Databases, Cavtat-Dubrovnik, Croatia, September 22-26, 2003. Proceedings 7*, pages 107–119. Springer, 2003.

[54] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284, 2009.

[55] Huixia Lu, Lazarus Ehwerhemuepha, and Cyril Rakovski. A comparative study on deep learning models for text classification of unstructured medical notes with various levels of class imbalance. *BMC Medical Research Methodology*, 22(1):1–13, 2022.

[56] Nathalie Japkowicz. Assessment metrics for imbalanced learning. *Imbalanced learning: Foundations, algorithms, and applications*, pages 187–206, 2013.

[57] Aixin Sun, Ee-Peng Lim, and Ying Liu. On strategies for imbalanced text classification using svm: A comparative study. *Decision Support Systems*, 48(1):191–201, 2009.

[58] Peng Su, Yifan Peng, and K Vijay-Shanker. Improving bert model using contrastive learning for biomedical relation extraction. *arXiv preprint arXiv:2104.13913*, 2021.

[59] Lingtong Min, Ziman Fan, Feiyang Dou, Jiaao Sun, Changsheng Luo, and Qinyi Lv. Adaption bert for medical information processing with chatgpt and contrastive learning. *Electronics*, 13(13):2431, 2024.

[60] Junfan Chen, Richong Zhang, Yongyi Mao, and Jie Xu. Contrastnet: A contrastive learning framework for few-shot text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 10492–10500, 2022.

[61] Qianben Chen, Richong Zhang, Yaowei Zheng, and Yongyi Mao. Dual contrastive learning: Text classification via label-aware data augmentation. *arXiv preprint arXiv:2201.08702*, 2022.

[62] Lin Pan, Chung-Wei Hang, Avirup Sil, and Saloni Potdar. Improved text classification via contrastive adversarial training. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11130–11138, 2022.

[63] Tillmann Dönicke, Matthias Damaschk, and Florian Lux. Multiclass text classification on unbalanced, sparse and noisy data. In *Proceedings of the First NLPL Workshop on Deep Learning for Natural Language Processing*, pages 58–65, 2019.

[64] Sumeet Agarwal, Shantanu Godbole, Diwakar Punjani, and Shourya Roy. How much noise is too much: A study in automatic text classification. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pages 3–12. IEEE, 2007.

[65] Dawei Zhu, Michael A Hedderich, Fangzhou Zhai, David Ifeoluwa Adelani, and Dietrich Klakow. Is bert robust to label noise? a study on learning with noisy labels in text classification. *arXiv preprint arXiv:2204.09371*, 2022.

[66] L Venkata Subramaniam, Shourya Roy, Tanveer A Faruquie, and Sumit Negi. A survey of types of text noise and techniques to handle noisy text. In *Proceedings of The Third Workshop on Analytics for Noisy Unstructured Text Data*, pages 115–122, 2009.

[67] Dimitris Bilianos. Experiments in text classification: Analyzing the sentiment of electronic product reviews in greek. *Journal of Quantitative Linguistics*, 29(3):374–386, 2022.

[68] Georgios Alexandridis, Iraklis Varlamis, Konstantinos Korovesis, George Caridakis, and Panagiotis Tsantilas. A survey on sentiment analysis and opinion mining in greek social media. information, 12 (8): 331, 2021.

[69] Efstratios G Vamvourellis and Despina Athanasia Pantazi. Compbertition: Which bert model is better for greek legal text classification? Master's thesis, Master's thesis, National and Kapodistrian University of Athens, Athens, 2021.

[70] Junmei Zhong and William Li. Predicting customer call intent by analyzing phone call transcripts based on cnn for multi-class classification. *arXiv preprint arXiv:1907.03715*, 2019.

[71] Nhi NY Vo, Shaowu Liu, Xitong Li, and Guandong Xu. Leveraging unstructured call log data for customer churn prediction. *Knowledge-Based Systems*, 212:106586, 2021.

[72] Youngja Park, Wilfried Teiken, and Stephen C Gates. Low-cost call type classification for contact center calls using partial transcripts. In *INTERSPEECH*, pages 2739–2742, 2009.

[73] Souraya Ezzat, Neamat El Gayar, and Moustafa M Ghanem. Sentiment analysis of call centre audio conversations using text classification. *International Journal of Computer Information Systems and Industrial Management Applications*, 4(1):619–627, 2012.

[74] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[75] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019.

[76] I Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[77] Yassine Marrakchi, Osama Makansi, and Thomas Brox. Fighting class imbalance with contrastive learning. In *International conference on medical image computing and computer-assisted intervention*, pages 466–476. Springer, 2021.

[78] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[79] A Conneau. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*, 2019.

[80] Leon Voukoutis, Dimitris Roussis, Georgios Paraskevopoulos, Sokratis Sofianopoulos, Prokopis Prokopidis, Vassilis Papavasileiou, Athanasios Katsamanis, Stelios Piperidis, and Vassilis Katsouros. Meltemi: The first open large language model for greek. *arXiv preprint arXiv:2407.20743*, 2024.

[81] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36, 2024.

[82] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[83] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous systems, 2015.

[84] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45, 2020.

[85] Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, et al. Pytorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, pages 929–947, 2024.

[86] Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and B Bossan. Peft: State-of-the-art parameter-efficient fine-tuning methods. *URL: https://github. com/huggingface/peft*, 2022.

[87] Tim Dettmers, Mike Lewis, Sam Shleifer, and Luke Zettlemoyer. 8-bit optimizers via block-wise quantization. *arXiv preprint arXiv:2110.02861*, 2021.