UNIVERSIDAD PONTIFICIA DE COMILLAS

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA

MÁSTER EN BIG DATA: TECNOLOGÍA Y ANALÍTICA AVANZADA

TRABAJO DE FIN DE MÁSTER

Lookalike segmentation in Movistar Plus+ customers

Segmentación *lookalike* en clientes Movistar Plus+

Nicolás Ruiz Lafuente

Supervisor: David Rubio Sánchez

In collaboration with Telefónica

Madrid - June 2025

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Segmentación *lookalike* en clientes Movistar Plus+

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico **2024/2025**, es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

Nicolás

Fdo.: Nicolás Ruiz Lafuente Fecha: 30 / 06 / 2025

Autorizada la entrega del proyecto
EL DIRECTOR DEL PROYECTO

[illegible]

Fdo.: David Rubio Sánchez Fecha: 30 / 06 / 2025

A mis tías,
Cruci y Ainhoa.

Abstract

This Master's Thesis addresses the design and deployment of a *lookalike* segmentation system for Movistar Plus+ , the streaming and television platform operated by Telefónica. The motivation stems from a structural limitation in the current marketing segmentation pipeline, which can only be applied to users who have explicitly consented (*opt-in*) to the use of sensitive data—particularly web browsing information. Due to privacy regulations such as the GDPR, a large portion of the customer base cannot be segmented or targeted using the existing process.

To overcome this constraint, we propose a deep learning system capable of estimating behavioral similarity between users using only data processed under legitimate interest. The model follows a *two-tower* (siamese) neural architecture, trained on *opt-in* user pairs using a continuous similarity label derived from their marketing segment overlap. Once trained, the system generates vector embeddings for any user and performs lookalike inference for *non-opt-in* by identifying the closest *opt-in* users via k-nearest neighbor search.

The solution is fully implemented within Telefónica's production environment, leveraging Databricks, PySpark, PyTorch, and FAISS. Special care has been taken to ensure scalability, cost efficiency, data governance, and compatibility with existing activation pipelines. From an evaluation standpoint, the model performs well on similarity regression and downstream segment inference, particularly for clusters that correlate well with the available features. Most importantly, it enables significant growth in segment coverage—often doubling or tripling the size of addressable audiences—without requiring additional user consent. These results demonstrate that the proposed framework offers a practical, privacy-compliant, and technically robust method for extending marketing segmentation at scale.

Resumen

El presente Trabajo de Fin de Máster tiene como objetivo el diseño y despliegue de un sistema de segmentación *lookalike* orientado a clientes de la plataforma Movistar Plus+ , perteneciente a Telefónica. El problema surge a raíz de una limitación estructural en el sistema actual de segmentación de marketing: este solo puede aplicarse a usuarios que han otorgado consentimiento expreso (*opt-in*) para el uso de datos sensibles, como su navegación web. Esta restricción, derivada de las normativas de privacidad del RGPD, deja fuera de las campañas personalizadas a una gran parte de la base de clientes.

Para abordar esta situación, se propone un sistema alternativo basado en aprendizaje profundo, que aprende a estimar la similitud comportamental entre clientes utilizando únicamente variables procesadas bajo la base legal de interés legítimo. La arquitectura propuesta sigue un enfoque *two-tower* (modelo siamesa), entrenada sobre pares de usuarios del conjunto *opt-in*, utilizando como señal supervisada una métrica de similitud continua calculada a partir de la coincidencia entre sus segmentos de marketing. Una vez entrenado, el modelo permite inferir representaciones vectoriales (embeddings) de cualquier usuario, y a partir de ahí identificar usuarios *no-opt-in* similares a clientes ya segmentados mediante búsqueda de vecinos más cercanos (*k-nearest neighbors*).

El sistema ha sido desarrollado e integrado en el ecosistema productivo de Telefónica, utilizando tecnologías como Databricks, PySpark, PyTorch y FAISS. Se ha prestado especial atención a la escalabilidad, la gobernanza del dato, y la compatibilidad con los procesos ya existentes de segmentación y activación de campañas. A nivel de resultados, el modelo ha mostrado un rendimiento aceptable tanto en la tarea de predicción de similitud como en la inferencia de segmentos, especialmente en los casos donde existe correlación entre los datos disponibles y los segmentos objetivo. Además, permite duplicar o incluso cuadruplicar el volumen de usuarios asignables a cada segmento, sin necesidad de solicitar nuevos consentimientos. Por todo ello, se concluye que este enfoque es una solución viable, extensible y alineada con los requisitos legales y operativos de la compañía.

Acknowledgements

I would like to express my deepest gratitude to all those who have supported and accompanied me throughout this Master's Thesis and the academic year.

This work would not have been possible without the trust and guidance of **Telefónica**, and more specifically the **Gerencia Innovación BI** department, which welcomed me as part of their team. I am especially grateful to my supervisor **David**, for his constant supervision, technical advice, and encouragement throughout the entire project.

My sincere thanks also go to my managers, **Tomás**—whose support and openness have been invaluable—and **Antonio**, as well as to the rest of my colleagues, especially **Javi**, for his help and companionship during the development of this work. Working in this environment has been both a privilege and a learning experience I will always carry forward.

I would also like to thank my classmates and friends from the Master's program, whose collaboration and good spirit have made this year both enriching and enjoyable.

Finally, I want to express my gratitude to my family and close friends, for their unconditional support and for always being there. Their encouragement has been the foundation behind every step of this journey.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1. Context and Problem

Telefónica Movistar Plus+ has established itself as one of the leading streaming and entertainment platforms in Spain, serving millions of subscribers on a wide range of content: films, series, live sports, documentaries and specialized channels. Behind the scenes, Movistar Plus+ relies on an advanced, privacy-compliant segmentation pipeline to deliver personalized marketing offers and content recommendations. This existing system focuses on 'opt-in' subscribers, those who have explicitly consented[1] to share detailed behavioral data, including web navigation and 'click' logs. By combining subscription metadata, viewing histories, demographic attributes, and web-navigation signals, the in-house team can classify opt-in users into well-defined marketing segments with high precision. These clusters form the backbone of targeted campaigns, underpinning everything from personalized ads to dynamic homepage banners.

However, the very richness of that profiling pipeline also highlights its fundamental limitation: legal and privacy constraints[1] prevent the same depth of data collection for "non-opt-in" subscribers. For this larger group — often representing well over half of the total user base — only nonsensitive signals are available: device types, time-stamped play durations, subscription tier, content categories viewed, and general data such as age and sex. Without access to browsing or 'click' data, these customers remain under-segmented, effectively invisible to the customized marketing machinery that drives both engagement and revenue. The result is two parallel customer universes within Movistar Plus+ : a high-granularity opt-in cohort, and a coarse, largely untapped non-opt-in population.

1.2. Our Proposal

To bridge this divide, we propose a lookalike segmentation model that 'learns' the mapping between user behaviors and characteristics and marketing groups in the opt-in population, then transfers that knowledge - without ever using sensitive data — to the non-opt-in group. Concretely, we train a deep neural network to embed every customer into a shared, low-dimensional vector space using only features permissible for all users. The model is exposed to opt-in examples during training, so it internalizes the subtle, multi-modal patterns that distinguish one marketing segment from another. Once trained, this embedding function can be applied to any subscriber — opt-in or not. For non-opt-in users, we then perform a nearest-neighbor lookup against the opt-in embeddings; each new user inherits the most common segment label among its closest peers.

[1] http://data.europa.eu/eli/reg/2016/679/oj

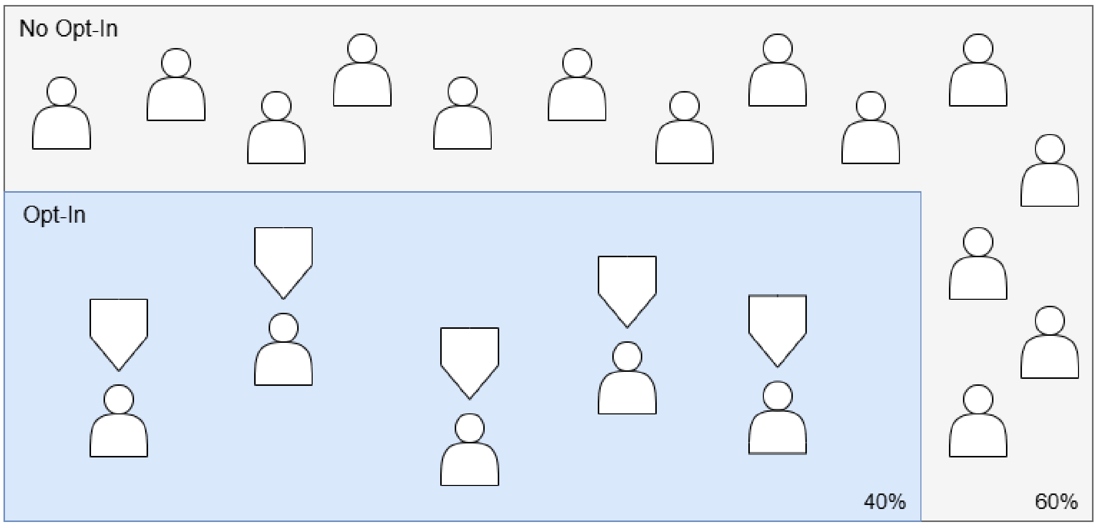

Figure 1.1: Comparison between opt-in (segmented) and non-opt-in (unsegmented) customers. Percentages are not real due to confidentiality reasons, but the overall idea is.

1.3. Key Advantages

Our proposal comes with a series of advantages that make it powerful and easy for the company to implement:

- **Extended Coverage.** By enabling segment assignment for all subscribers, we dramatically increase audience sizes for each campaign cluster, unlocking new revenue streams without requiring additional user consent.
- **Seamless Integration.** Our pipeline slots into the existing Databricks-based ecosystem: data ingestion from Delta Tables, feature engineering in Spark, and optimized data feeding into PyTorch. All experiments are versioned in MLflow, and the new code branches directly off the current opt-in segmentation repository.
- **Operational Efficiency.** Marketers continue to work within the familiar segment taxonomy; they simply gain more “lookalike” customers in each bucket. No new dashboards or segment definitions are needed, minimizing disruption.
- **Privacy Compliance.** We never regress on consent. Sensitive signals remain strictly within the opt-in pipeline; non-opt-in inference relies solely on aggregated, non-sensitive features.

1.4. Objectives and Research Questions

The overarching goal of this thesis is to design, implement, and evaluate this embedding-based lookalike segmentation system within Telefónica’s production environment. More specifically, we aim to:

1. **Design** a neural embedding model that generalizes opt-in segmentation patterns using only non-sensitive features.
2. **Implement** the model training, inference, and indexing processes on Databricks with end-to-end tracking in MLflow.
3. **Evaluate** the quality of inferred segments via quantitative metrics (nearest-neighbor purity, cluster coverage, classical classification scores).

To guide our work, we focus on three core research questions:

- **RQ1 – Transferability:** How well do the learned embeddings recover opt-in segment distinctions when applied to non-opt-in users?
- **RQ2 – Scalability vs. Fidelity:** What is the relationship between embedding complexity (vector dimension, network depth, model architecture) and segment assignment accuracy at scale?
- **RQ3 – Feature Importance:** Which categories of non-sensitive features (e.g., temporal viewing patterns, subscription metadata, content genre proportions) are most predictive of segment membership in a lookalike setting?

1.5. Limitations and Contributions

This work focuses on extending the existing opt-in segmentation pipeline to non-opt-in users using only non-sensitive features. It does not cover online A/B testing of campaigns, nor does it redefine the company's core segment taxonomy. Integration with all ~ 60 segments (including the most navigation-dependent ones) is supported, though detailed analysis of each segment's individual performance is left to later work.

In summary, the main contributions of this thesis are:

1. The design of an embedding-based lookalike segmentation model that generalizes opt-in patterns using only non-sensitive data while transfering this knowledge to non-opt-in users.
2. The implementation of an end-to-end pipeline on Databricks, with feature engineering in Spark and model tracking in MLflow, seamlessly branching from the existing opt-in codebase.
3. A comprehensive evaluation framework measuring nearest-neighbor purity, cluster coverage, and classification-style metrics to quantify business impact.

Chapter 2

Literature Review

Over the past decade, customer segmentation has evolved from simple rule-based and clustering techniques into sophisticated, data-driven approaches powered by machine learning and deep learning. Early methods—such as k-means clustering, hierarchical clustering, and RFM analysis—relied on manually engineered features and were limited in their ability to incorporate high-dimensional, temporal, or multi-modal data. With the proliferation of digital touchpoints, companies now collect rich behavioral signals including clickstreams, viewing sequences, and session metadata at massive scale. To leverage these complex signals, modern segmentation frameworks employ embedding-based representations, often realized through siamese or two-tower neural architectures, which map customers (or customer–item pairs) into a continuous vector space. In this latent space, similarity measures such as cosine distance can serve as the basis for "lookalike" modeling—extending known customer segments to new or partially observed users. Concurrently, advances in approximate nearest-neighbor search (e.g., FAISS, Annoy, LSH, ScaNN) and distributed training frameworks (Spark, data sharding) have made these techniques feasible in production environments handling millions or billions of users.

Despite these advances, many existing models have focused primarily on user–item interactions, particularly in the context of recommendation systems. While collaborative filtering and matrix factorization (MF) have proven effective in capturing individual preferences, they are fundamentally limited when the objective is not to recommend items but to identify users who are behaviorally similar to a known seed group. Customer segmentation in the digital era often requires handling massive, multi-modal, and incomplete user data—ranging from session activity to demographic metadata—where the target task is not to optimize for personalized ranking but to generalize existing audiences. This distinction is central to the emergence of lookalike modeling, which seeks to expand a segment, campaign, or cohort to previously unobserved users based on learned patterns of similarity in behavior, intent, or context.

This chapter reviews this conceptual and technical transition—from recommender systems to lookalike modeling—and presents a first analysis of the deep learning architectures that support it. We begin by contrasting traditional collaborative-filtering models with lookalike systems, then explore the structure and training of siamese and two-tower architectures, as well as the role of scalable similarity search frameworks such as FAISS. Finally, we examine real-world deployments of lookalike models in industry, focusing in particular on Walmart's two-tower system for audience expansion, which serves as the principal inspiration for the architecture developed in this thesis.

2.1. From Recommender Systems to Lookalike Modeling

Matrix factorization (MF) aims to learn latent vectors for users and items that approximate an observed interaction matrix $R \in \mathbb{R}^{n \times m}$, typically by minimizing a reconstruction loss:

$$\min_{U,V} \sum_{(i,j)\in\Omega} (R_{ij} - U_i^\top V_j)^2 + \lambda(\|U_i\|^2 + \|V_j\|^2), \tag{2.1}$$

where U_i and V_j are the latent embeddings of user i and item j, respectively, and Ω is the set of observed interactions. While this is effective for personalized ranking, it does not generalize to situations where user–user similarity is required—such as when trying to find new users similar to a seed group based on multi-source features rather than shared item interactions.

Lookalike modeling, in contrast, is not about item recommendation, but about identifying customers that resemble a known group based on their behavior, demographics, or consumption patterns. This process is especially relevant in marketing, where we often want to expand an audience (e.g., targeting users "similar to those who clicked on a campaign") even if those new users have no interactions in common with the originals.

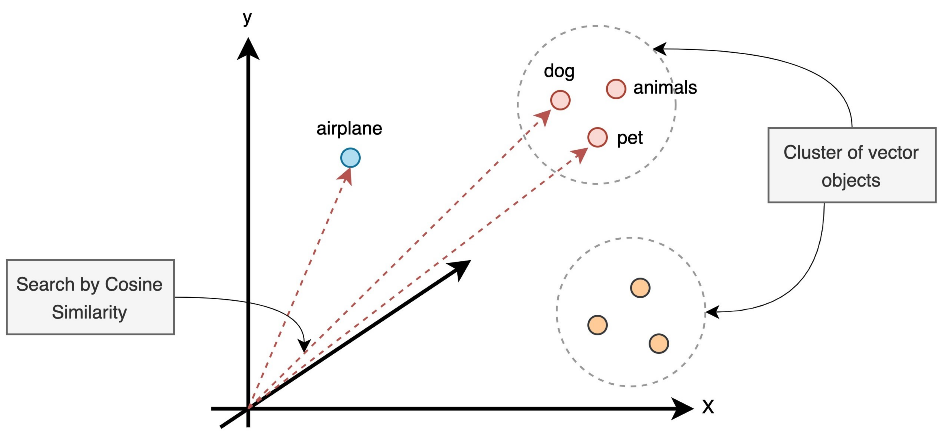

Figure 2.1: Schematic interpretation of a latent space used for word embedding problems. Lookalike-latent-spaces follow the same logic as those developed for word embedding.

2.2. Feature-Based vs Model-Based Lookalikes

Feature-based lookalike modeling relies on computing similarity scores (e.g., cosine, Jaccard, Euclidean) on manually crafted feature vectors, which can be sufficient for small or interpretable datasets. However, this approach quickly breaks down with sparse, high-dimensional data or when the features span multiple modalities (e.g., behavioral time series and categorical demographics). Moreover, hand-crafted features are brittle to change and require domain-specific tuning.

Model-based lookalikes offer a scalable alternative: we learn an embedding function $e(x)$ that maps users to a latent space such that geometric proximity corresponds to behavioral similarity. This embedding is typically learned using a siamese or two-tower network trained on labeled pairs—e.g., whether two users belong to the same marketing segment. The objective is to minimize a loss that reflects embedding similarity and label agreement, such as:

$$\mathcal{L}_{\text{regression}} = \sum_{(i,j)} \left| \text{sim}(e(x_i), e(x_j)) - y_{ij} \right|, \tag{2.2}$$

or, for classification-style training:

$$\mathcal{L}_{\text{contrastive}} = y_{ij} \|e_i - e_j\|^2 + (1 - y_{ij}) \max(0, m - \|e_i - e_j\|)^2. \tag{2.3}$$

where e_i and e_j are the generated embeddings and y_{ij} are the ground truth similarities.

2.3. Siamese and Two-Tower Architectures

Siamese networks apply a shared embedding function to both inputs, while two-tower networks allow each input to be processed independently through its own tower, which is critical when the input distributions differ (e.g., opt-in vs non-opt-in users). In both cases, the similarity is computed using a metric such as cosine similarity:

$$\text{sim}(e_a, e_b) = \frac{e_a^\top e_b}{\|e_a\| \|e_b\|}. \tag{2.4}$$

Two-tower architectures have gained widespread adoption in real-world systems because they enable pre-computation and indexing of embeddings. This is key for scalability. For instance, YouTube's deep retrieval system [Covington et al., 2016] uses two-tower networks to efficiently match user embeddings to candidate video embeddings at massive scale.

2.4. Efficient Similarity Search: FAISS and Beyond

Even with compact embeddings, exact nearest neighbor search is computationally expensive at scale. Approximate nearest neighbor (ANN) libraries like FAISS [Douze et al., 2024][Johnson et al., 2019] and ScaNN [Pang et al., 2020] allow efficient retrieval from large embedding corpora.

FAISS (Facebook AI Similarity Search) provides support for flat, inverted file (IVF), and product quantization (PQ) indices, and runs efficiently on both CPUs and GPUs. Its ability to combine indexing, compression, and batching makes it ideal for real-time retrieval systems, as demonstrated by Walmart and others.

ScaNN (Scalable Nearest Neighbors) is optimized for CPU-based inference with efficient recall-speed tradeoffs. Vector databases like Pinecone and Milvus build additional functionality on top of FAISS (e.g., metadata filtering, persistence) but come with operational overhead.

2.5. Walmart: A Production Lookalike System

Among the publicly documented industry deployments of lookalike modeling, the approach developed by Walmart stands out as one of the most technically mature and conceptually aligned with our goals. Presented in their internal research and supported by large-scale experimentation [Peng et al., 2023], Walmart's system is built upon a two-tower architecture designed to generate user–user similarity embeddings for the purpose of retail audience expansion. Rather than modeling user–item interactions, as is common in traditional recommender systems, Walmart explicitly targets the problem of mapping users into a latent space where distance reflects marketing affinity. This setup mirrors the fundamental challenge we face in Movistar Plus+ :

inferring segment-level similarity for users who do not share explicit behavioral signals (e.g., product views or purchases) with each other.

The core of Walmart's solution lies in how similarity is defined and supervised. Instead of using binary similarity labels—i.e., pairs labeled as "similar" or "not similar"—the model regresses on a continuous similarity score derived from business-specific logic. Specifically, Walmart computes a behavioral vector O_u for each user u, encoding purchase frequency across product categories, and defines the ground-truth similarity between two users A and B as the cosine similarity of these business vectors:

$$y_{AB} = \cos(O_A, O_B). \tag{2.5}$$

This formulation enables a significantly more nuanced view of similarity than contrastive or triplet loss approaches. In classic metric learning settings, the model is provided with binary supervision: a pair of users is either labeled as a match ($y = 1$) or not ($y = 0$), and the loss is constructed accordingly. While simple to implement, this binary framing imposes a hard separation and fails to capture the gradient of similarities that naturally arises in real-world customer behavior. By contrast, Walmart's regression-based loss encourages the model to learn embeddings where the predicted similarity varies smoothly and continuously, following the degree of overlap between customer behavior patterns. The loss used is:

$$\mathcal{L} = \sum_{(A,B)} \left| \cos(e_A, e_B) - \cos(O_A, O_B) \right|, \tag{2.6}$$

where e_A and e_B are the 128-dimensional embeddings produced by the two towers. This approach not only provides richer supervision, but also encapsulates the logic of the entire segmentation pipeline in a single similarity score—one that can be precisely defined and controlled by the business.

The advantage of this formulation is twofold. First, it decouples the definition of similarity from any rigid label or threshold, allowing teams to encode nuanced business goals directly into the training objective. Second, it provides a highly flexible and general framework: by changing the definition of the business vector O, the same architecture can be repurposed for different segmentation strategies or marketing verticals without requiring architectural changes.

To support this architecture at production scale, Walmart's pipeline integrates several mature tools. Embeddings are indexed using FAISS, enabling fast approximate nearest-neighbor retrieval in both offline and online stages. During inference, opt-in users are embedded and used as seeds to retrieve similar users, forming expanded marketing audiences.

Although the Walmart paper does not present detailed quantitative metrics or A/B test results, it does mention that the proposed lookalike model significantly outperforms the company's existing system. This qualitative improvement, based on internal evaluation, supports the claim that the regression-based similarity framework offers tangible advantages in capturing meaningful user–user behavioral affinity, and reinforces the practical value of the architecture for audience expansion use cases.

In short, Walmart's approach offers a powerful and generalizable blueprint for lookalike modeling. Its success hinges on a clever shift away from binary similarity supervision toward regression on a continuous business-defined metric—a shift that greatly enhances modeling flexibility, interpretability, and control. This insight directly informs our architecture for Movistar Plus+ , where similar constraints and objectives apply.

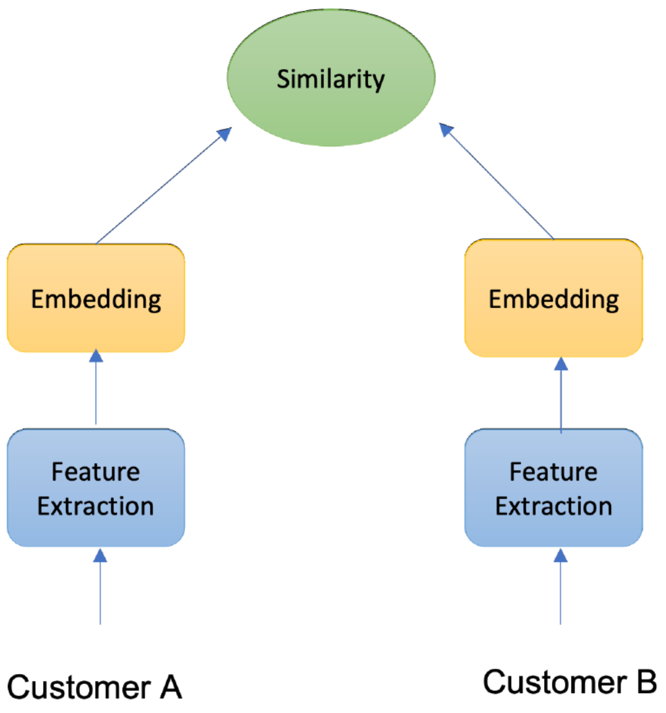

Figure 2.2: Two tower model used by Walmart [Peng et al., 2023].

2.6. Adaptation to Movistar Plus+

We adopt Walmart's core principles while adapting to the context and constraints of Movistar Plus+ . Our towers are trained using opt-in similarity defined over behavioral segments and excludes any sensitive data, ensuring GDPR-compliant[1] usage. Embeddings are indexed using FAISS, and inference is run within Telefónica's Databricks environment, using clusters with GPUs for scalable distributed training.

This design closely mirrors the official Databricks two-tower architecture example [Databricks, 2024], and benefits from infrastructure already used in Telefónica's segmentation workflows.

In the next chapter, we detail the data sources, feature transformations, and training data preparation that power this model.

[1]http://data.europa.eu/eli/reg/2016/679/oj

Chapter 3

Data and Preprocessing

A well-designed data pipeline is essential for the development of any robust machine learning system. In the context of this project, where the goal is to infer customer similarity with respect to marketing segments, the input data must not only be statistically sound and representative, but also comply with strict privacy regulations. Moreover, the preprocessing strategy must be consistent between the training phase (which involves opt-in users) and the inference phase (which targets non-opt-in users). This chapter presents a detailed overview of the data sources, feature structure, preprocessing techniques, and analytical groundwork that inform the embedding model developed in this thesis.

3.1. Data Sources and Privacy Constraints

Telefónica's data governance framework, in full alignment with the General Data Protection Regulation (GDPR)[1], establishes a clear distinction between two types of customers: those who have provided explicit consent for extended data use (opt-in), and those for whom only data processed under legitimate interest is available (non-opt-in). The former group enables richer behavioral insights, including detailed web navigation and interaction logs. However, in the context of this project, these additional signals are deliberately excluded to ensure compatibility and fairness with the broader user base.

As such, the feature set used in both training and inference is strictly limited to variables falling under the legitimate interest legal basis. These include general service metadata, content consumption metrics, device characteristics, and aggregated behavioral indicators, all of which are available for both opt-in and non-opt-in users. This constraint is fundamental to the model's operational integrity: since inferences are ultimately made for non-opt-in users, it is critical that the model never relies on signals that would be unavailable at deployment time.

Although the exact list of features cannot be disclosed due to internal confidentiality policies, they can be categorized into the following broad domains:

- **Content consumption features**: Aggregated metrics related to user interaction with Movistar Plus+ content, such as total and average watch time, number of distinct sessions, and the distribution of viewing across genres, channels, and temporal windows.
- **Device and terminal information**: Technical specifications of the devices used, including device type, operating system, screen size, and hardware generation.

[1]http://data.europa.eu/eli/reg/2016/679/oj

- **Customer profile data**: Demographic variables such as age and gender, account metadata like tenure, contract type, or plan characteristics.
- **Roaming and mobile usage**: Indicators derived from mobile and roaming activity, including data usage, connection type, and frequency of access from foreign networks.
- **Technical configuration**: Network- or system-level attributes related to connectivity, firmware, or subscription infrastructure.
- **Model-derived features**: Outputs from other Telefónica internal models, limited strictly to those computed from legitimate-interest data.

Importantly, all data derived from explicit-consent signals—most notably, browsing behavior—is excluded from this project. This guarantees not only GDPR compliance, but also alignment with the company's internal data segmentation policies and long-term model maintainability.

3.2. Unified Preprocessing Strategy

To ensure compatibility between training and inference, the same preprocessing pipeline is applied to both opt-in and non-opt-in users. While only opt-in users are used during training (due to the availability of segment annotations), the pipeline is designed to handle both groups identically in terms of feature transformation. This principle of preprocessing parity is essential, as it ensures that the learned embedding function generalizes across the entire customer base.

The preprocessing workflow is implemented entirely in PySpark and includes: (i) one-hot encoding for categorical features, (ii) z-score standardization for numerical features, and (iii) identity pass-through for boolean flags. The transformations are trained (i.e., fitted) using only the training split of opt-in users and subsequently saved to Unity Catalog Volumes, an artifact management feature provided by Databricks. These transformation pipelines are then reloaded and reused to process new customer data—whether for inference or retraining—ensuring strict reproducibility and feature consistency.

Furthermore, the adoption of Unity Catalog for both data and transformation storage enables fine-grained access control, schema versioning, and documentation of feature metadata. This governance layer plays a vital role in operationalizing the model at scale and will be discussed in further detail in the chapter dedicated to model lifecycle and data governance.

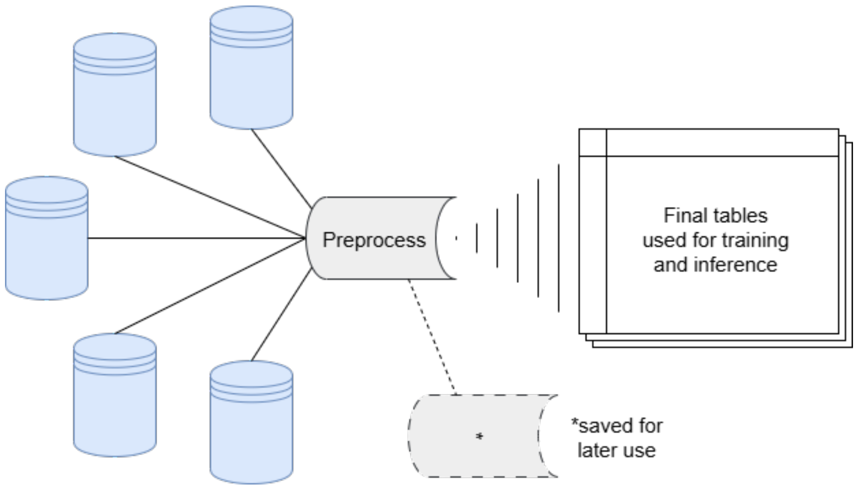

Figure 3.1: Summary of the data pipeline. Multiple Telefónica data sources are joined and processed to produce a unified training or inference table, fully aligned with the model's input requirements.

3.3. Temporal Aggregation and Feature Typology

Given the temporal nature of many behavioral signals, the data preprocessing strategy involves explicit time-window aggregation. Depending on the semantic of the feature, aggregation windows are defined over the last 7 days (short-term dynamics), last 30 days (monthly trends), or last 90 days (quarterly stability). For instance, content view counts are aggregated over multiple time spans to capture both recency and long-term interests.

Features are also grouped by type:

- **Categorical variables**: Transformed using one-hot encoding when cardinality is low or moderate.
- **Boolean features**: Retained as binary flags with no transformation.
- **Numerical features**: Standardized using the z-score formula:

$$z_i = \frac{x_i - \mu}{\sigma} \tag{3.1}$$

  where x_i is the raw feature value, μ is the mean, and σ the standard deviation computed over the opt-in training set.

After all preprocessing steps, each customer is represented by a dense feature vector of over 300 dimensions. This excludes navigation data, which—as previously noted—is not used. The high dimensionality ensures a rich representation while remaining computationally tractable for downstream embedding.

An important exception to this encoding strategy involves the *address* feature set. These variables, which describe the user's geographical residence or administrative origin, are provided not as raw address fields but as pre-processed categorical values generated by a separate internal model maintained by Telefónica. Due to their high cardinality and structured semantic meaning, these features are not suitable for one-hot encoding, which would produce an excessively sparse and large feature space. Instead, they are retained in their original categorical format, with the intention of being embedded into a learned vector representation via a separate neural module. Although this embedding submodel is not yet implemented in the current version of the system, its integration is considered a key avenue for future improvement and is discussed in Chapter 8.

3.4. Label Structure and Segment Information

The training signal is derived from the existing opt-in customer segmentation process used internally by Telefónica. This process assigns opt-in users to one or more marketing segments, typically based on a richer set of data, including consented sources. Although our model does not reproduce or predict these segments directly, we use them as the ground truth from which to define similarity between users.

In particular, segment labels are used to derive a real-valued similarity score between customer pairs: users assigned to similar segments are treated as more similar in behavioral space. The exact strategy for converting segment information into numerical similarity values is detailed in Chapter 4. Here, it suffices to state that segment annotations serve as the basis for the supervised training of our lookalike embedding function.

3.5. Feature Correlation Analysis

To assess the informativeness of the input features, we conducted a correlation analysis across features and segment assignments, using the opt-in population as the reference. Although the segments themselves are ultimately condensed into a continuous similarity score for training, this analysis provides valuable intuition about which variables contribute most to behavioral structure.

Although confidentiality restrictions prevent us from disclosing the names of the features or the exact definition of each segment, the resulting visualization reveals several important insights. In general, the observed correlations are of moderate magnitude, with a predominance of positive values, which aligns with expectations for a marketing segmentation problem built on consumption and profile data. Negative correlations (shown in blue in the matrix) also make intuitive sense within the domain context; even though we cannot show the variable names, any reader with access to the feature definitions would likely find their presence interpretable and coherent.

More importantly, a significant structural pattern can be observed along the upper rows of the matrix. Specifically, there exists a subset of marketing segments that exhibit very low correlation—often near zero—with all input features. This phenomenon is not due to noise or modeling failure, but reflects the underlying constraint of the problem: these particular segments are assigned in the internal segmentation process exclusively on the basis of user web navigation data, which is not available in our training pipeline due to GDPR constraints. Since our model only uses features covered by legitimate interest, it is structurally incapable of capturing the behaviors that define these segments.

This insight has both explanatory and operational implications. On the one hand, it confirms the internal consistency of our data pipeline: segments that depend on data we cannot use are, as expected, not explainable through our feature space. On the other hand, it highlights a key limitation of the current model, which is its reduced ability to generalize across the full spectrum of marketing segments defined by Telefónica. This issue is discussed again in Chapter 8, where

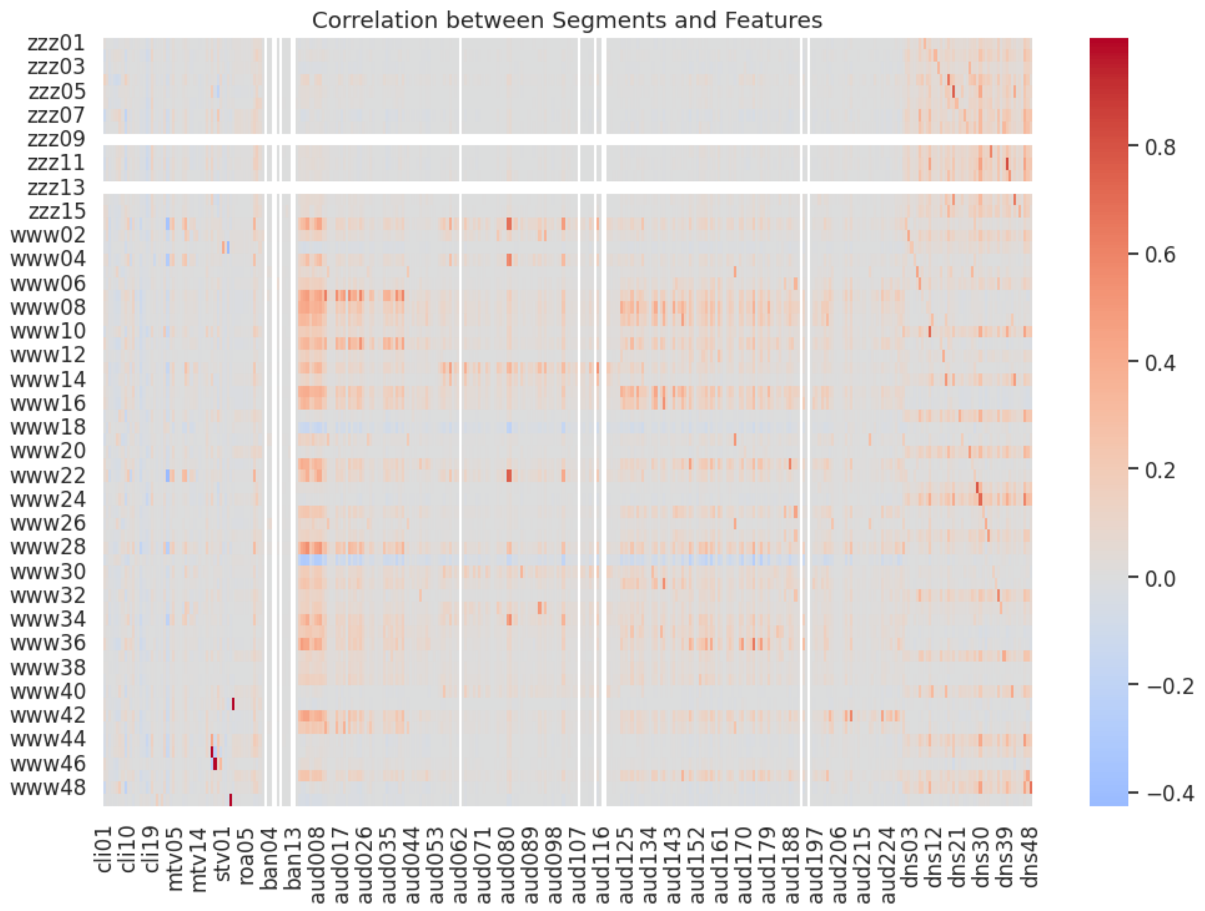

Figure 3.2: Segment to feature correlation matrix.

we assess the trade-offs and suggest directions for future model improvements.

3.6. Closing Remarks

This chapter has presented the principles, tools, and decisions involved in building the feature space that underpins our lookalike model. By enforcing strict symmetry in preprocessing, adhering to legitimate-interest constraints, and leveraging the engineering capabilities of Databricks and Unity Catalog, we ensure that the system is both legally compliant and operationally scalable. In subsequent chapters, we will describe how similarity signals are defined, how training pairs are constructed, and how the embedding model is trained and evaluated.

Chapter 4
Methodology

4.1. Overview and Objectives

The previous chapter detailed the structure and transformation of the data that serves as input to our model. In this chapter, we introduce the core modeling approach designed to capture and generalize behavioral similarity between customers, with the goal of enabling segment-based audience expansion under strict privacy constraints.

At a high level, the problem can be understood as follows: given a set of customers who belong to one or more known marketing segments (opt-in users), we aim to learn a function that maps any customer—opt-in or not—to a vector in a latent space where geometric proximity reflects segment-level behavioral affinity. This embedding function must be trained using only features available under legitimate interest (i.e., no web navigation data), yet it should generalize to non-opt-in users, for whom no segment labels are available.

To achieve this, we adopt a two-tower deep learning architecture inspired by prior work in large-scale recommendation and lookalike modeling [Peng et al., 2023][Covington et al., 2016]. The core idea is to train a network that embeds customers into a dense, low-dimensional space, using a supervised signal derived from existing segmentations of the opt-in population. Rather than predicting class labels or ranking items, the model is trained to approximate a real-valued similarity score between customer pairs, computed from their respective segment annotations. This allows the system to learn fine-grained behavioral relationships and to extend them naturally to users whose labels are unknown.

The overall methodology involves several key components: (i) the construction of training examples as customer–customer pairs with associated similarity scores; (ii) a neural architecture capable of processing structured, multi-modal input features into embeddings; (iii) a loss function tailored to regress similarity rather than classify; and (iv) an inference pipeline that indexes all embeddings and retrieves lookalike users via approximate nearest-neighbor search. Each of these components is described in detail in the following sections.

4.2. Model Architecture

The embedding model used in this work is based on a two-tower neural network architecture, a widely adopted design in large-scale recommendation and retrieval systems [Covington et al., 2016][Peng et al., 2023]. The architecture consists of two symmetric but independent subnetworks (or "towers") that process two user profiles in parallel and map each one to a dense vector in a shared latent space. The output of the model is a similarity score between the two resulting embeddings, intended to approximate a target similarity value derived from the segment-level relationship between the users.

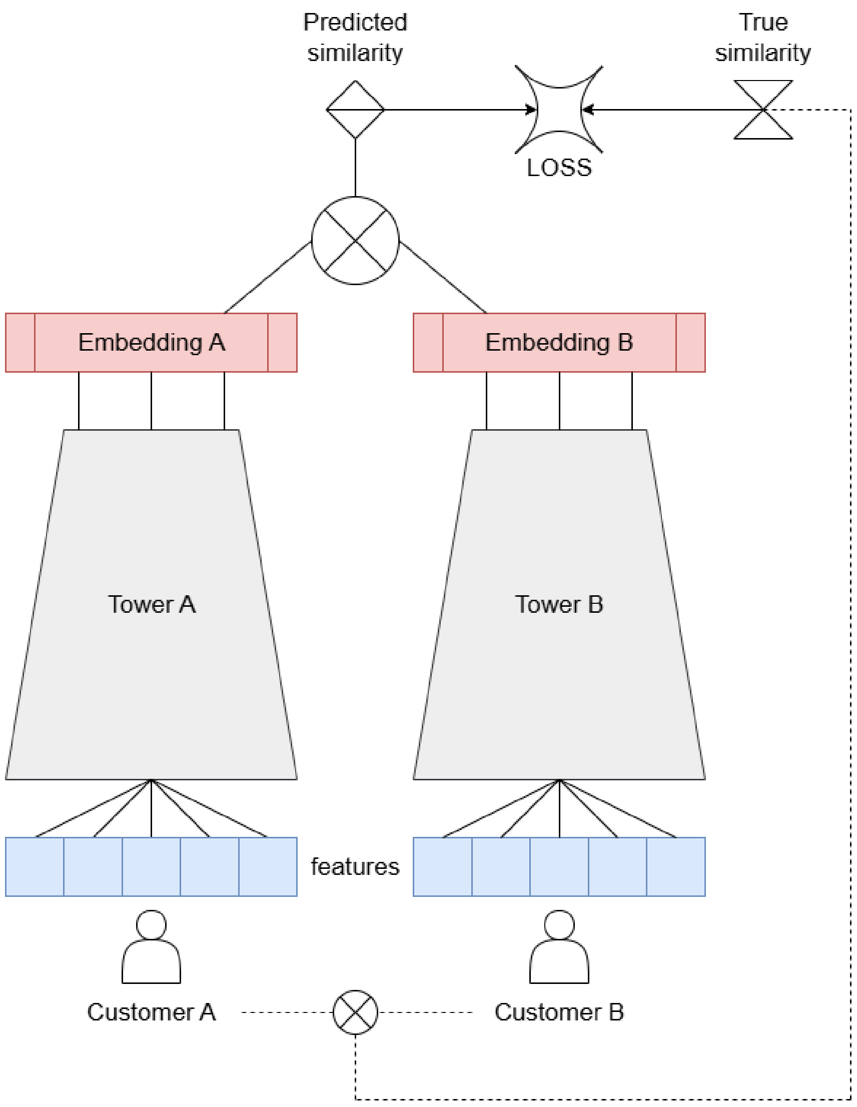

Figure 4.1: Two tower architecture developed for this project.

This architectural choice is particularly well suited for our problem setting. First, it supports pairwise supervision: the model is trained not on individual customers, but on pairs of customers, labeled with a similarity score. Second, it enables inference at scale: once all customer embeddings are precomputed and stored, similarity queries can be resolved via efficient vector operations and approximate nearest-neighbor search. Finally, the decoupled nature of the towers makes it easy to apply the model to asymmetric data: while both towers share the same input feature schema (since we use only legitimate-interest features), in future iterations they could be independently tuned to support specialized processing for opt-in and non-opt-in cohorts.

Each tower is implemented as a feed-forward neural network that receives as input a fixed-length vector of preprocessed features, comprising standardized numerical values, one-hot encoded categorical attributes, and binary indicators (as described in Chapter 3). Although the total dimensionality of this input vector exceeds 300 features, the majority of entries are zero-valued due to the extensive use of sparse encodings (e.g., one-hot for multi-category variables). This sparsity pattern is handled naturally by the feed-forward architecture, which treats the vector as a flat tensor without assuming any specific structural layout. After an optional input dropout layer for regularization, the network consists of multiple dense layers with ReLU activation functions, batch normalization, and final L2 normalization to produce unit-length embeddings. The architectural hyperparameters—such as the number of layers, hidden dimensions, dropout rates, and normalization strategies—are further detailed in Chapter **??**.

Formally, given two customer feature vectors x_A and x_B, the towers compute:

$$e_A = f_\theta(x_A), \quad e_B = f_\theta(x_B),$$

where f_θ denotes the neural tower with parameters θ, and $e_A, e_B \in \mathbb{R}^d$ are the resulting normalized embeddings. The similarity between the embeddings is then computed via cosine similarity:

$$\mathrm{sim}(e_A, e_B) = \frac{e_A^\top e_B}{\|e_A\| \cdot \|e_B\|},$$

which by construction falls in the interval $[-1, 1]$, though in practice the normalization ensures $[0, 1]$ values. The similarity is compared to a ground-truth target y_{AB} during training, and the network is optimized to minimize a regression loss over this value (as described in Section 4.5).

Notably, the model is agnostic to any specific segment definition. It does not attempt to classify users into discrete marketing clusters, but instead learns a smooth similarity landscape where proximity implies behavioral and commercial affinity. This embedding space serves as the foundation for all downstream lookalike inference tasks, including the indexing and retrieval of non-opt-in customers who resemble high-value opt-in profiles.

4.3. Label Construction and Similarity Definition

One of the central challenges in building a lookalike model is defining a meaningful target for supervised training. In a classical classification setting, models are trained to predict discrete labels. However, in our case, the goal is not to assign a user to a specific marketing segment, but rather to embed users in a latent space where the distance between two customers reflects how similar they are in terms of segment-level behavior. In most recommender systems that adopt a two-tower architecture, the prevailing approach is based on contrastive or triplet loss formulations [Chopra et al., 2005][Schroff et al., 2015]. These metric learning techniques treat the embedding task as one of learning to distinguish "positive" and "negative" pairs of inputs—typically based on some known interaction, such as click or co-occurrence. However, this dichotomous formulation has several limitations in our context.

First, defining a binary similarity label between users is non-trivial: what constitutes a "positive" or "negative" pair is often ambiguous. While it might be possible to define a threshold on the overlap of marketing segments, this choice would be arbitrary and potentially sensitive to noise or label imbalance. Second, binary supervision inherently flattens the similarity landscape: two users who share nearly all segments are treated the same as those who share only one. This prevents the model from learning fine-grained differences in behavioral affinity.

Instead, we adopt a continuous regression-based approach, following the methodology proposed by Walmart [Peng et al., 2023]. In this setup, the model is trained to predict a real-valued similarity score $y_{AB} \in [0, 1]$ between two users A and B, derived from their segment assignments. This formulation captures graded similarity, avoids manual thresholds, and allows the model to learn a smoother latent space.

To compute this similarity score, we leverage the internal segmentation process already deployed by Telefónica for opt-in users. As explained in Chapter 3, this segmentation process assigns each opt-in customer to one or more marketing segments, based on a mixture of behavioral, technical, and—in many cases—navigation-derived features. Although our model does not have access to web navigation data, we use these existing segment assignments as the starting point for defining behavioral similarity among users. Taking this into account, we first represent each user's segment membership as a binary vector. Each position corresponds to a segment; a 1 indicates membership, and 0 otherwise. Several similarity measures exist for binary vectors, as surveyed in [Choi et al., 2010]. Among the most common are:

- **Cosine similarity**:

$$\mathrm{sim}_{\cos}(A, B) = \frac{\sum_i A_i B_i}{\sqrt{\sum_i A_i^2} \cdot \sqrt{\sum_i B_i^2}}$$

- **Jaccard similarity**:

$$\mathrm{sim}_{\mathrm{jac}}(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{\sum_i A_i B_i}{\sum_i \max(A_i, B_i)}$$

- **Hamming similarity (normalized)**:

$$\mathrm{sim}_{\mathrm{ham}}(A, B) = 1 - \frac{1}{n} \sum_i \not\Vdash [A_i \neq B_i]$$

Each function captures a different notion of overlap or divergence, and the three of them provide a continuous and bounded value $\in [0, 1]$, where 1 means maximum similarity [1]. In our case, we selected **cosine similarity** as the ground-truth function for supervision. Cosine similarity measures the angle between two binary vectors and is well-suited for sparse segmentations where the absolute number of segments per user may vary. Unlike Jaccard, it does not penalize users for having small supports, and unlike Hamming, it does not assume segment independence.

[1]Theoretically $\cos\theta \in [-1, 1]$ but for binary vectors that results in $\cos\theta' \in [0, 1]$

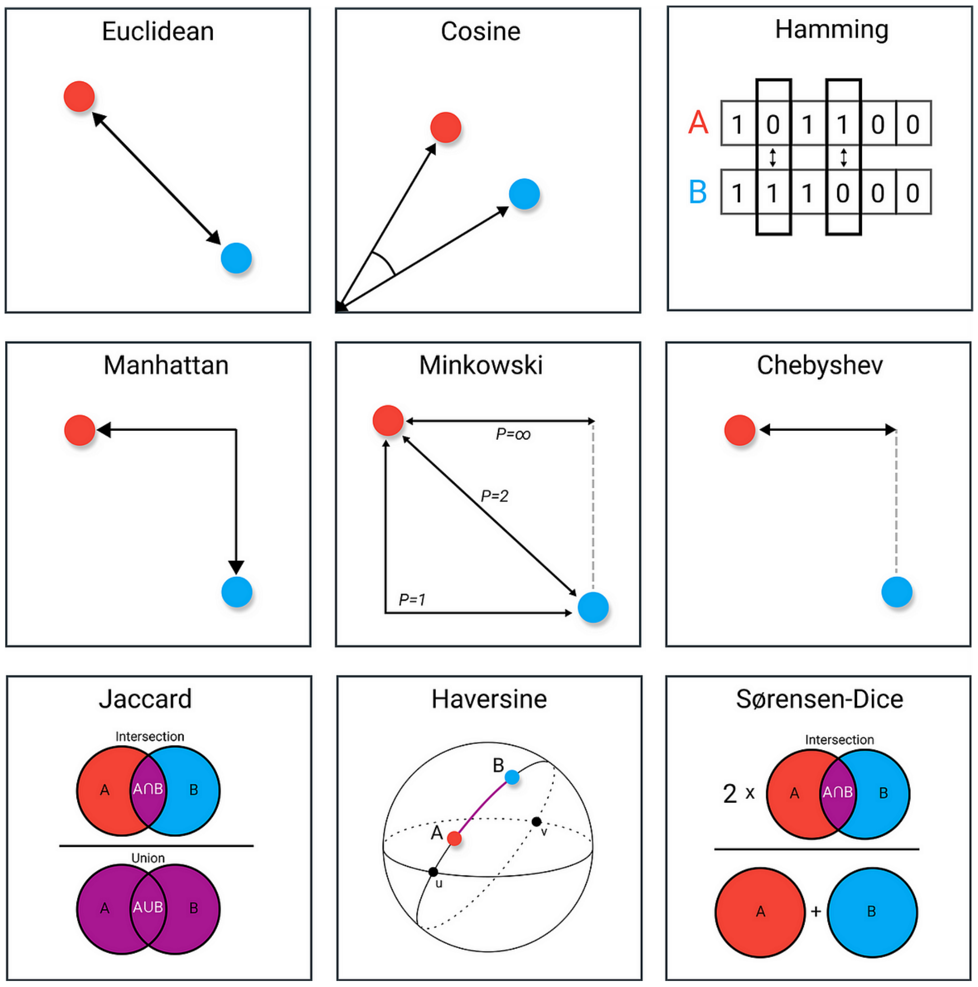

Figure 4.2: Comparison of popular similarity and distance functions used in machine learning and vector comparison. Our work uses cosine similarity to compute the training signal.

This similarity function serves as the regression target in our loss function, described in Section 4.5. The result is a continuous, interpretable target that captures the behavioral affinity between any two opt-in customers, and allows the model to generalize this similarity to non-opt-in users via learned embeddings.

Finally, it is worth noting that the construction of training pairs is not uniform across all user combinations. Some users are more heavily represented in the data, and their segment annotations may be noisier or more overlapping. To avoid biases in the training signal, we implement a sampling strategy described in the following section, which ensures balance and diversity in the pairs fed to the model.

4.4. Pair Sampling Strategy

Once a similarity function has been defined, the next step is to construct training examples in the form of customer–customer pairs annotated with their corresponding similarity scores. Given that the number of possible user pairs grows quadratically with the size of the opt-in dataset, it is neither computationally feasible nor desirable to include all possible combinations. Instead, a targeted sampling strategy is employed to build a balanced and diverse training set.

The basic unit of supervision is a pair of opt-in users (A, B), for which we compute the ground-truth similarity $y_{AB} \in [0, 1]$ using the cosine similarity between their binary segment membership vectors, as described in Section 4.3. A naive random sampling approach would lead to a highly imbalanced dataset, dominated by pairs with very low similarity—since most customer pairs do not share any marketing segment. Training on such imbalanced data would result in a model biased toward predicting low similarity scores, limiting its discriminative power.

To address this, we adopt a stratified sampling strategy based on similarity intervals. The continuous range $[0, 1]$ is divided into discrete bins (e.g., intervals of width 0.1), and user pairs are sampled such that the training set includes a more uniform distribution of similarity values. This ensures that the model is exposed to a broad variety of behavioral relationships: from customers who share no segments at all, to those with significant overlap in marketing profile.

Figure 4.3 illustrates the distribution of sampled pairs across similarity bins. Although the natural frequency of mid-range similarity pairs is higher in the population, the sampling logic is adjusted to promote coverage across the entire interval. This improves the robustness of the model and its capacity to generalize to ambiguous or partially similar users.

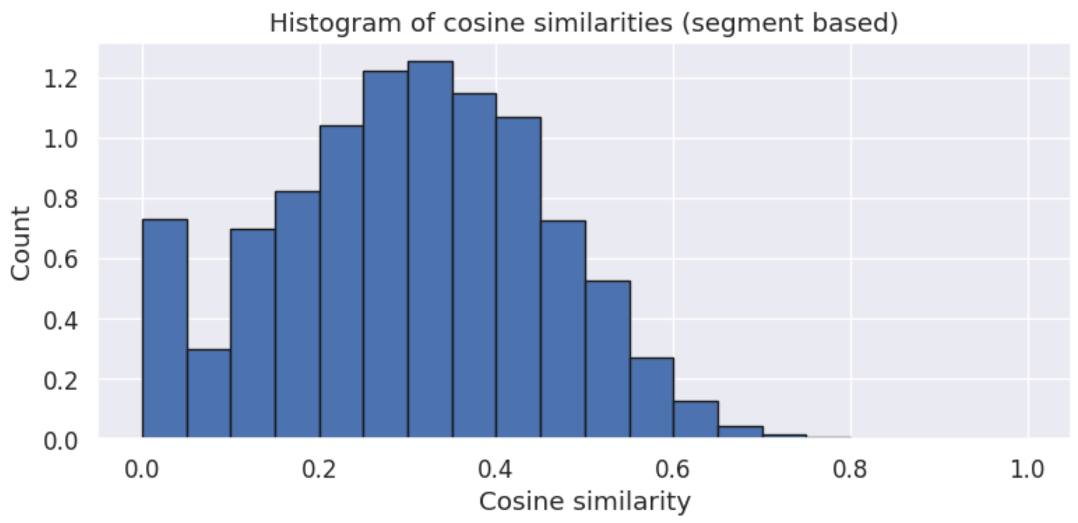

Figure 4.3: Distribution of similarity values among sampled training pairs. The sampling strategy ensures a broad and balanced exposure across similarity levels.

The specific sampling implementation—along with memory optimizations and dynamic strategies used to generate training batches—is described in detail in Chapter **??**. For now, it suffices to note that the pair construction logic is designed to provide the model with informative and diverse examples across the full spectrum of behavioral similarity.

4.5. Loss Function

Having defined the architecture of the model and the ground-truth similarity function between customer pairs, the final component of the supervised learning setup is the choice of a loss function. The role of the loss is to guide the model in learning embeddings such that geometrical similarity in the latent space approximates the true behavioral similarity inferred from marketing segment assignments.

Most two-tower models in recommender systems rely on binary supervision: pairs are labeled as either "positive" (e.g., same user-item interaction, shared behavior) or "negative" (no interaction), and the model is trained using a contrastive or triplet loss [Chopra et al., 2005][Schroff et al., 2015]. These loss functions operate by reducing the distance between embeddings of positive pairs and increasing it for negative pairs, often up to a fixed margin.

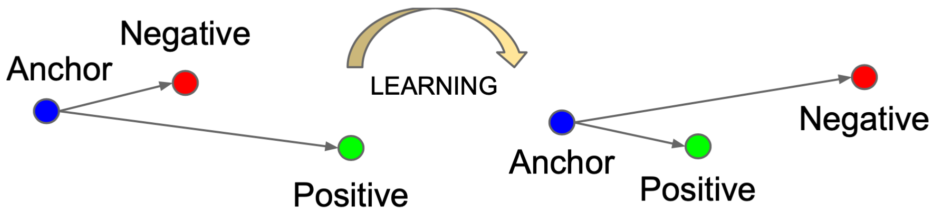

Figure 4.4: Conceptual view of triplet loss.

However, this binary framing is not appropriate in our case. First, it would require defining a threshold on the similarity between segment vectors to decide whether a pair is positive or negative—an arbitrary and brittle decision, especially in high-noise, multi-label settings. Second, binary losses ignore the graded nature of behavioral similarity: a pair of users that shares four segments is treated the same as one that shares only one. This flattening effect undermines the granularity and continuity of the embedding space.

Instead, we adopt a regression-based loss, in line with the approach used by Walmart in their lookalike two-tower model [Peng et al., 2023]. Given a pair of users (A, B), and their embeddings $e_A, e_B \in \mathbb{R}^d$, we compute their predicted similarity via cosine similarity:

$$\hat{y}_{AB} = \frac{e_A^\top e_B}{\|e_A\| \cdot \|e_B\|},$$

and compare it to the ground-truth similarity score y_{AB} derived from segment overlap through cosine similarity as well [2], as described in Section 4.3. The loss function is defined as the L1 error (mean absolute error, MAE) between the predicted and true similarity:

$$\mathcal{L} = \sum_{(A,B)\in\mathcal{D}} |\hat{y}_{AB} - y_{AB}| \,.$$

We choose L1 loss over L2 (mean squared error, MSE) for its robustness to outliers and its ability to preserve sharp differences in local similarity structure. In early experiments, L2 loss exhibited instability during training and tended to over-smooth high-similarity pairs.

This regression setup has several advantages:

- It eliminates the need for manual thresholds between positive and negative pairs.
- It supports smooth supervision across the full range of similarity values $[0, 1]$.
- It allows the embedding space to reflect nuanced behavioral relationships between users.

[2]The similarity function used to define the ground-truth similarity score does not need to match the one used to compute the embedding similarity during inference. Their alignment in our case—both using cosine similarity—is a deliberate design choice, not a theoretical requirement.

- It maintains compatibility with common indexing tools such as FAISS, which support cosine distance-based retrieval.

Overall, the regression objective aligns closely with the modeling goals of this project. Rather than classifying users into discrete categories or maximizing ranking accuracy, the model learns a continuous similarity landscape that supports flexible, interpretable, and privacy-compliant lookalike retrieval across the full customer base.

4.6. Training Procedure

Once the model architecture, the similarity function, and the sampling strategy are defined, the next step is to train the two-tower network to produce embeddings that reflect user similarity according to the supervised signal. The training is carried out using the PyTorch deep learning framework, and is fully integrated into Telefónica's Databricks-based machine learning infrastructure, which supports distributed training, automated experiment tracking, and scalable data ingestion.

The training data consists of user–user pairs (A, B), each labeled with a ground-truth similarity score $y_{AB} \in [0, 1]$. These pairs are generated on-the-fly using a batch sampling strategy based on similarity binning, as described in Section 4.4. Each input pair is processed independently by the two towers, which share parameters, and the predicted similarity $\hat{y}_{AB}$ is computed as the cosine of the resulting embeddings. The model is then updated using an L1 regression loss between $\hat{y}_{AB}$ and y_{AB}, as formalized in Section 4.5.

Training is conducted over multiple epochs, with shuffled mini-batches of sampled user pairs. Standard techniques are applied to stabilize and optimize learning, including:

- The Adam optimizer, with an initial learning rate tuned via grid search.
- Mini-batch training with a fixed batch size (typically between 512 and 2048, depending on memory availability).
- L2 regularization and dropout layers applied within the towers to mitigate overfitting.
- Early stopping based on validation loss, monitored over a held-out validation set of user pairs.

The validation and test sets are constructed by sampling disjoint subsets of opt-in user pairs using the original, unaltered similarity distribution. This deliberate choice ensures that model evaluation reflects real-world conditions: while the training set is balanced across similarity bins to expose the model to the full spectrum of behaviors, the validation and test sets retain the natural imbalances present in the data. As a result, they provide a realistic estimate of the model's ability to generalize to all similarity levels, including those that are underrepresented in the wild.

All training runs are tracked using MLflow, which captures hyperparameters, model checkpoints, training metrics, and metadata about the experiment. This allows for reproducibility and systematic evaluation across multiple model versions. In addition, metadata such as the training date, training data version, and Spark pipeline versions are recorded to facilitate deployment audits and future retraining.

It is important to note that, given the dynamic nature of customer behavior and seasonality in marketing activity, the model is intended to be retrained on a quarterly basis (every 3 months). Each retraining cycle uses updated opt-in segment data and recent behavioral features. Embedding inference, by contrast, is performed on a weekly basis to keep the lookalike population aligned with the latest opt-in segmentation. These operational aspects are discussed in greater detail in Chapter 5.

4.7. Embedding Inference and Indexing

Once the two-tower model has been trained and validated, the next step is to use it to generate dense vector representations—embeddings—for every user in the system. These embeddings serve as the foundation for downstream lookalike retrieval: users who are close in the embedding space are assumed to be behaviorally similar and are likely to belong to the same or comparable marketing segments.

The embedding function $f_\theta(x)$, learned during training, is applied to all opt-in users and to the broader population of non-opt-in users, using only features permitted under the legitimate interest legal basis. As discussed in Chapter 3, both opt-in and non-opt-in customers are preprocessed using the same Spark-based pipeline, ensuring that the input vectors x are compatible with the model.

The inference phase proceeds as follows. First, the embedding model is loaded from MLflow, including its trained parameters and associated feature transformation artifacts. Next, a distributed Spark job is launched to apply the embedding function to each user in the dataset. The result is a table of user identifiers and embedding vectors.

To support scalable lookalike retrieval, all embeddings are indexed using FAISS (Facebook AI Similarity Search) [Douze et al., 2024]. FAISS is a library designed for efficient approximate nearest-neighbor search in high-dimensional spaces, offering multiple indexing algorithms (e.g., flat, IVF, HNSW) and support for GPU acceleration [Johnson et al., 2019]. In our implementation, we use a flat index with cosine similarity as the distance metric, which ensures maximum recall at the cost of slower retrieval—an acceptable trade-off given our offline inference setup.

The core idea behind lookalike retrieval is to use opt-in users as "anchor" points: since their segment membership is known, their embeddings carry implicit semantic information. For each non-opt-in user, we retrieve its K nearest opt-in neighbors and aggregate their segment labels to infer plausible segment associations. This can be done using majority voting, weighted label aggregation, or probabilistic fusion strategies. The exact aggregation logic is discussed in Chapter 5.

This embedding-based approach has several advantages:

- It decouples model inference from downstream logic: once embeddings are generated, multiple retrieval or scoring strategies can be applied.
- It supports rapid re-indexing and reuse of embeddings, which is essential for weekly refreshes.
- It ensures that inference on non-opt-in users remains compliant, since only legitimate-interest features are used.

The entire inference and indexing pipeline is orchestrated within Databricks, leveraging its native support for Delta tables, MLflow, and GPU-enabled FAISS clusters. All outputs are versioned and tagged with metadata, allowing for traceability and reproducibility across inference cycles.

The final step in the inference pipeline is to integrate the inferred segment labels for non-opt-in users back into the segmentation workflow. This is achieved by appending the lookalike-derived assignments to the same table used by the original segmentation process, which contains the segment annotations for opt-in users. The result is a unified view of the customer base, where both opt-in and non-opt-in customers are associated with segment-level information—either manually assigned or inferred through the embedding-based retrieval mechanism. This design ensures compatibility with existing marketing tools and dashboards, allowing the business to operate seamlessly across the entire customer universe without requiring separate treatment of opt-in and non-opt-in populations.

4.8. Model Lifecycle and Retraining Strategy

The deployment of a machine learning model is not the end of its development, but rather the beginning of a continuous lifecycle of maintenance, evaluation, and retraining. In the case of the lookalike segmentation system presented in this thesis, the lifecycle has been carefully designed to ensure that model performance remains aligned with real-world behavioral dynamics and business objectives.

Customer behavior in a platform like Movistar Plus+ is subject to seasonal trends, shifting content offerings, and evolving consumption patterns. Events such as summer vacations, Christmas campaigns, sporting events, or large-scale promotions (e.g., Black Friday) can introduce temporary but significant changes in user engagement and segment composition. For this reason, the model is retrained periodically to reflect the most up-to-date distribution of behaviors observed among opt-in users.

Specifically, the model is scheduled to be retrained every three months, using the most recent segment assignments and user features available at the time. Each retraining cycle includes re-fitting the preprocessing pipeline (Spark transformations), generating new training pairs with updated similarity scores, and retraining the two-tower architecture from scratch. This approach ensures that the embeddings remain representative of current user behavior, and that the model continues to reflect the segmentation logic applied to opt-in users.

In contrast to model retraining, embedding inference and lookalike retrieval are executed more frequently. Given that the opt-in segmentation process is refreshed weekly, the lookalike pipeline is synchronized to run on the same cadence. Each week, updated non-opt-in embeddings are generated, indexed, and used to infer segments by proximity to the current opt-in population. This weekly loop ensures that non-opt-in users are always assigned segments relative to the most recent opt-in reference cohort.

All model artifacts—including trained weights, validation metrics, preprocessing pipelines, and embeddings—are versioned using MLflow. The use of Databricks Volumes and Unity Catalog further facilitates traceability, allowing teams to audit which model version was used to generate each inference result, and ensuring alignment between data, model, and business decisions.

This disciplined lifecycle management guarantees that the lookalike model remains robust to changes in the data distribution, operationally reliable for downstream teams, and compliant with Telefónica's internal standards for model governance.

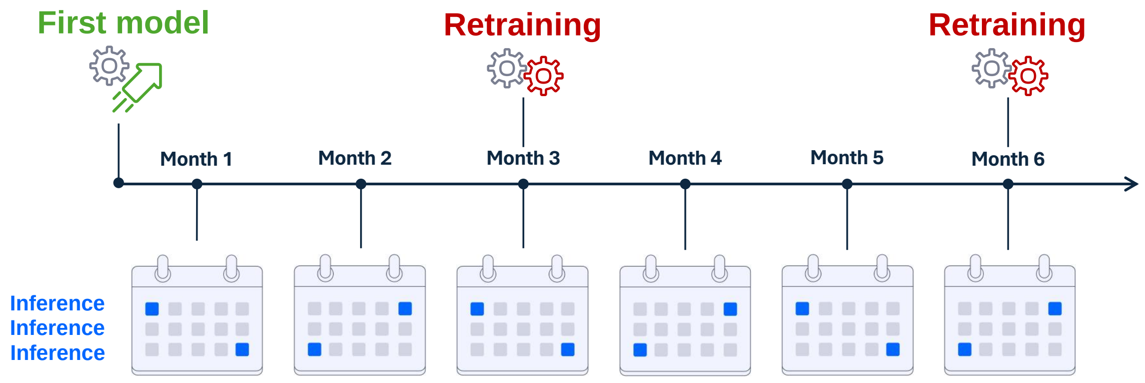

Figure 4.5: Model lifecycle timeline. Inference runs weekly and retraining is scheduled every 12 weeks. The process repeats cyclically.

Chapter 5

Deployment

5.1. Technical Environment and Tools

The deployment of the lookalike segmentation system has been carried out entirely within Telefónica's internal Databricks platform, which provides a cloud-native environment for large-scale data processing, model training, experiment tracking, and reproducible machine learning workflows. The technological stack combines distributed data engineering tools with modern deep learning frameworks, allowing the system to scale to millions of users while maintaining transparency and governance.

- **Databricks**: The central environment for orchestration, used for data exploration, Spark-based feature processing, model training, and inference jobs. Databricks provides native integration with Unity Catalog and MLflow, which are leveraged for feature lineage, versioning, and model management.
- **Apache Spark (PySpark)**: Used to construct and transform the feature tables from raw Delta Tables into model-ready vectors. Spark ensures scalable and parallel processing of millions of users.
- **PyTorch**: The core framework used for model architecture, training, and embedding generation. It offers flexibility for implementing custom training loops, loss functions, and sampling logic.
- **MLflow**: Used to track all training experiments, including hyperparameters, metrics, model artifacts, and logs. It provides version control and reproducibility for the entire lifecycle.
- **FAISS (GPU version)**: Used for approximate nearest-neighbor search on high-dimensional embedding vectors. The GPU version was chosen after benchmarking its performance, yielding significantly faster indexing and retrieval than its CPU counterpart—an important consideration in terms of cost and time efficiency [Johnson et al., 2019].
- **Unity Catalog & Delta Tables**: Used for structured storage of training and inference tables, with versioning, schema enforcement, and descriptive metadata to support model governance.

This ecosystem allows the model to bridge the gap between large-scale data ingestion and high-performance embedding retrieval, while remaining compatible with Telefónica's internal privacy policies and data governance requirements.

5.2. Training and Inference Table Generation

As described in Chapter 3, the preprocessing of input features is unified across opt-in and non-opt-in customers, relying exclusively on variables collected under the legal basis of legitimate interest. This shared logic ensures that the model receives input vectors with consistent structure and semantics regardless of user consent status. Rather than redefining the preprocessing flow in this chapter, we focus here on how it is used operationally to generate the data tables required for training and inference.

At the core of the system is a Spark-based pipeline that transforms raw Delta Tables into fully processed feature tables. These are used in two distinct contexts:

- **Training Table**: A dedicated table is generated on demand for model re-training. It includes only opt-in users for whom marketing segment annotations are available, and it is joined with those segment labels to enable supervised pair generation. This table is rebuilt each time a new training cycle is initiated, typically every three months (see Section 4.8).
- **Weekly Inference Tables**: On a weekly basis, two separate tables are produced: one for opt-in users and one for non-opt-in users. These tables contain fully preprocessed features and are used to compute embeddings during the inference phase. The opt-in embeddings serve as anchors in the lookalike retrieval process, while the non-opt-in embeddings are the targets for segment inference.

All three tables are stored as Delta Tables under Unity Catalog. They include detailed metadata describing the origin, transformation logic, feature schema, and date of generation. This enables reproducibility, facilitates audits, and ensures compliance with Telefónica's internal data governance standards.

While these tables are logically derived from the same preprocessing pipeline, they differ in purpose, cadence, and population scope. The training table is constructed less frequently and includes segment labels, while the inference tables are generated weekly and only the opt-in table contains target annotations. This separation of concerns allows the model to be trained and evaluated on controlled, labeled data while being deployed efficiently and compliantly on the full customer base. The current deployment has proven robust and efficient. The full table creation process is completed in around 50 minutes using a multi core Databricks' cluster.

5.3. Pair Sampling Implementation

The generation of user pairs for training, along with their corresponding similarity labels, represents one of the most technically challenging components of the deployment pipeline. As explained in Section 4.4, the model requires labeled examples of opt-in user pairs, each annotated with a similarity score based on their segment overlap. However, implementing this logic at scale introduces several significant constraints.

The primary issue is the size of the training space. With more than 1 million opt-in users, the total number of possible user pairs is on the order of $\mathcal{O}(n^2)$—over 1 trillion combinations. Storing or even generating all such pairs in memory is infeasible, both due to cluster memory limits and storage constraints.

To address this, we implement a fully dynamic sampling strategy where user pairs are generated on-the-fly within each training minibatch. This logic is encapsulated in a custom PyTorch 'Dataset' and 'Sampler', which:

- Load batches of preprocessed user vectors from local memory (after ingestion from the Delta table).
- Sample user pairs within each batch based on predefined similarity bin weights (using PyTroch's 'WeightedRandomSampler').
- Compute the similarity score of each sampled pair using the cosine similarity of their segment assignment vectors.
- Immediately discard the sampled pairs after usage, freeing memory for the next iteration.

The bin weights used to guide sampling are precomputed in an offline analysis step and stored externally. These weights ensure that each minibatch includes pairs from across the full similarity spectrum, correcting for the natural imbalance in the population. Importantly, only the weights are static—pair generation itself is fully online and re-randomized at each epoch.

This implementation was non-trivial. Integrating Delta Table ingestion (via Spark), data movement to the local PyTorch environment, weighted sampling logic, and dynamic similarity computation required careful design. Debugging this flow—particularly to maintain correct weight normalization and avoid memory overflow—was one of the most time-consuming aspects of model development. In practice, this bottleneck became a key engineering challenge and shaped the overall structure of the training pipeline.

While alternative strategies such as offline pair generation or in-cluster preprocessing were considered, they were ultimately discarded due to performance and scalability limitations. The final design strikes a balance between resource constraints and model fidelity, enabling high-quality, similarity-balanced training under strict memory constraints.

5.4. Model Training Configuration

The two-tower embedding model is implemented in PyTorch using a modular structure that separates the network architecture, the training logic, and the evaluation metrics. The model is defined as a subclass of `torch.nn.Module`, and includes two identical towers composed of fully connected layers with ReLU activations, dropout, and batch normalization. The final layer of each tower applies L2 normalization to produce unit-length embeddings, suitable for cosine similarity.

Training is performed using standard PyTorch tooling and is orchestrated in Databricks notebooks. The model receives as input a stream of minibatches, each consisting of sampled pairs of opt-in users along with their ground-truth similarity score (as described in Section 5.3). For each batch, the following steps are performed:

1. The two user vectors in each pair are passed independently through the twin towers to obtain embeddings e_A and e_B.
2. Cosine similarity is computed between the embeddings to produce the predicted similarity $\hat{y}_{AB}$.
3. The L1 loss between $\hat{y}_{AB}$ and the ground truth y_{AB} is computed over the batch.
4. Backpropagation and optimizer steps are applied to update the model parameters.

The training configuration includes:

- **Optimizer**: Adam, with learning rate selected via grid search over a logarithmic scale.
- **Batch size**: Tuned experimentally based on cluster memory limits; typically in the range 1024–2048.
- **Dropout**: Applied at the input and hidden layers to mitigate overfitting, with rates between 0.1 and 0.3.
- **Early stopping**: Triggered based on validation loss stagnation over a fixed number of epochs.
- **Regularization**: L2 penalty applied to tower weights, with value selected during hyperparameter tuning.

All training runs are tracked using MLflow, which logs hyperparameters, training loss, validation loss, and model artifacts. Each experiment run is associated with a unique model ID and metadata, including the version of the feature pipeline, the training table used, and the date of training. This infrastructure allows full reproducibility of any training session and facilitates rollback, comparison, and deployment automation.

Training is performed on a dedicated GPU-enabled cluster within Databricks, using two NVIDIA A100 GPUs in parallel. This setup provides sufficient compute capacity to process large minibatches and accelerate convergence without requiring full distributed training. While the use of frameworks such as TorchDistributor or Horovod is not strictly necessary at this scale, they remain potential avenues for future optimization as the dataset grows or the architecture becomes more complex.

5.5. Embedding Inference, Indexing, and Retrieval

Once the model has been trained, its output—namely the embedding function—is used to compute dense vector representations for all customers in the system. These embeddings are the core mechanism through which segment similarity is operationalized, as they allow us to measure behavioral proximity without access to explicit segment labels for non-opt-in users.

Every week, a dedicated inference job is executed to generate embeddings for both opt-in and non-opt-in users. Opt-in users serve as anchors in the latent space: since their marketing segment labels are known, their embeddings provide the semantic foundation for subsequent retrieval. Non-opt-in embeddings, by contrast, are generated in an unsupervised fashion using only legitimate-interest features and the trained model.

Although the data pipeline up to this point is entirely distributed and Spark-native, the actual inference step is executed locally using Pandas and PyTorch. After exporting the feature tables from their Delta format, the model is loaded from MLflow, and embeddings are generated in batches using GPU acceleration. This approach was chosen due to a critical limitation in FAISS: indexing must be performed on a single machine. Since the lookup phase requires all embeddings to reside in memory on a single node, maintaining the inference process on the same host reduces complexity and avoids additional data transfer.

Once all embeddings are generated, they are indexed using FAISS (Facebook AI Similarity Search) [Douze et al., 2024]. FAISS provides high-performance approximate nearest-neighbor (ANN) search for dense vectors, supporting various index types and both CPU and GPU execution modes. After empirical benchmarking, the GPU version was selected for production use, as it reduced indexing and retrieval time by more than an order of magnitude compared to the CPU variant [Johnson et al., 2019]. These performance gains have a direct impact on operational cost, particularly given the weekly cadence of embedding refreshes.

Retrieval is performed using a flat index with cosine similarity as the distance metric. For each non-opt-in user, the system queries the FAISS index to retrieve the top-K most similar opt-in embeddings. The segment labels of these neighbors are then aggregated—using either majority voting, mean or weighted frequency—to produce a set of inferred segments for the non-opt-in user.

Finally, the resulting inferences are joined with the opt-in segmentations to produce a unified segmentation table. This table is used as the reference for downstream marketing actions, dashboards, and campaign design. From the perspective of the business, the distinction between opt-in and non-opt-in disappears: both user types are associated with marketing segments, either explicitly or via lookalike inference.

While effective, this architecture presents clear scalability limitations. In particular, the requirement that all embeddings reside on a single machine—imposed by FAISS—prevents full distribution of the pipeline. Future work may explore distributed approximate search engines or hybrid architectures that retain FAISS's speed while enabling sharded execution.

To validate the performance of FAISS under different hardware configurations, we conducted a series of benchmarks using a reference set of 100,000 user embeddings and $k = 10$ for nearest-neighbor search. The results are shown in Table 5.1.

Configuration	**Indexing (s)**	**Search (s)**
CPU	0.16	493.21
One GPU	0.34	1.30
Multi-GPU	0.51	0.65

Table 5.1: FAISS benchmark: indexing and search time for $n = 100{,}000$, $k = 10$.

As the results indicate, GPU-based indexing achieves a dramatic reduction in query time: from over 8 minutes on CPU to under 2 seconds with a single GPU, and less than 1 second using multiple GPUs. While indexing is slightly slower on GPU due to initialization and data transfer overhead, the improvement in retrieval performance makes GPU FAISS the most suitable option for a production pipeline with weekly refreshes.

Beyond the nearest-neighbor search itself, the segment assignment step—where the top-k opt-in neighbors' labels are aggregated—is also computationally inexpensive. Based on several production tests, the entire end-to-end inference cycle (embedding generation, indexing, retrieval, and segment aggregation) is completed in under 90 seconds. This ensures that the system is not only scalable and accurate, but also fast enough to be executed regularly with minimal infrastructure load.

Therefore, we can safely state that the entire weekly pipeline—from feature table generation to embedding inference, FAISS indexing, nearest-neighbor retrieval, and segment assignment—completes in just over one hour, including cluster initialization times. Among all steps, the most time-consuming by far is the creation of the input feature tables, as described in Section **??**. This step involves full Spark transformations and joins over large-scale Delta Tables, and determines the overall execution time of the weekly cycle.

5.6. Experiment Tracking and Model Versioning

A core requirement for deploying machine learning models in production environments—especially in large-scale, privacy-sensitive contexts like Telefónica's—is full traceability of model versions, training data, and configuration parameters. To that end, the lookalike segmentation pipeline leverages MLflow for experiment tracking and model versioning.

Each training run is logged as an MLflow experiment, recording metadata such as model hyperparameters, training and validation loss curves, timestamps, and model artifacts. In addition, custom tags are used to record the associated feature pipeline version, the identifier of the training Delta Table, and the Spark job metadata used during table generation. This information enables reproducibility, facilitates debugging, and allows teams to audit which model version was used for a specific inference batch.

All trained models are stored in the MLflow Model Registry, allowing for controlled promotion to staging and production environments. Once a model is validated and registered for production use, its corresponding inference logic is frozen and reused for weekly embedding generation and lookalike retrieval.

Feature tables and embeddings are also versioned through Unity Catalog and Delta Lake, ensuring that all artifacts—data and models alike—are covered by Telefónica's governance infrastructure. This guarantees consistency between training, inference, and business activation layers, while providing a robust foundation for compliance and lifecycle management.

In combination, these mechanisms allow the lookalike model to evolve over time while maintaining strict operational control and transparency.

Chapter 6

Results

6.1. Objectives of the Evaluation

The evaluation of the lookalike segmentation system focuses on two complementary objectives. The first is to assess the quality of the latent space generated by the two-tower embedding model—that is, whether the model successfully learns to map similar users close together and dissimilar users far apart, as defined by their marketing segment annotations. The second objective is to evaluate the behavior of the full lookalike pipeline, including embedding generation, nearest-neighbor retrieval via FAISS, and segment assignment by aggregation.

Unlike traditional recommender systems, where evaluation often focuses on ranking metrics (e.g., precision@k, recall@k, NDCG), or supervised models evaluated with classification accuracy, our model is trained via regression on a continuous similarity score. Therefore, the main metric used to evaluate the latent space is the Mean Absolute Error (MAE) between the predicted similarity (cosine between embeddings) and the ground-truth similarity derived from opt-in segment overlap.

To complement this scalar evaluation, we analyze the structure of the learned similarity space using a graphical spectrogram. This visualization provides a detailed view of how the input and output similarity distributions evolve across training epochs, separately for training and validation sets. It also shows how prediction errors vary across different similarity levels, offering a more granular view of model performance.

In addition to this internal evaluation of the embedding model, we evaluate the end-to-end performance of the system as a classifier. To do so, we simulate the full lookalike process on opt-in users: we generate their embeddings, retrieve top-k opt-in neighbors (excluding themselves), and infer their segments based on the segments of those neighbors. These inferred segments are then compared to the true segments of the user, allowing us to compute classic classification metrics such as precision, recall, and F1-score. This setup allows us to evaluate the segment assignment logic as if it were applied to non-opt-in users, but using opt-in users for whom ground-truth labels are available.

It is important to emphasize that all quantitative evaluation—both for embedding similarity and for segment classification—is performed exclusively on opt-in users. This is because only for this group do we have ground-truth segment annotations, which are required to compute evaluation metrics such as MAE, precision, or recall. Non-opt-in users are the actual targets of the inference pipeline, but cannot be used for evaluation due to the lack of reference labels.

However, beyond classical metrics, we also examine the practical impact of the model in terms of segment coverage. Specifically, we report the volume of users assigned to each segment, broken down by opt-in and non-opt-in populations. This provides a complementary view of the system's utility, showing how the lookalike model increases the addressable audience size for marketing activation.

6.2. Evaluation of Embedding Similarity (Regression Task)

The embedding model is trained using a regression objective: minimizing the difference between the predicted similarity (cosine between embeddings) and the true similarity derived from segment overlap. As discussed in Chapter 4, this design does not aim to predict discrete labels or enforce hard similarity thresholds, but rather to learn a continuous latent space where distances between users reflect behavioral affinity.

The principal metric used to evaluate this objective is the Mean Absolute Error (MAE) between the predicted and ground-truth similarity values. MAE is computed over both the training and validation sets at the end of each epoch, providing a scalar indication of how well the model is learning to reproduce the segment-based similarity structure.

To complement this global metric, we use a graphical spectrogram that visualizes several distributions and their evolution over training epochs:

- The distribution of predicted similarity values (cosine between embeddings), for both training and validation sets.
- The distribution of true similarity values (segment-based targets), for both sets.
- The MAE broken down by similarity bins (e.g., [0.0–0.1], [0.1–0.2], ..., [0.9–1.0]).

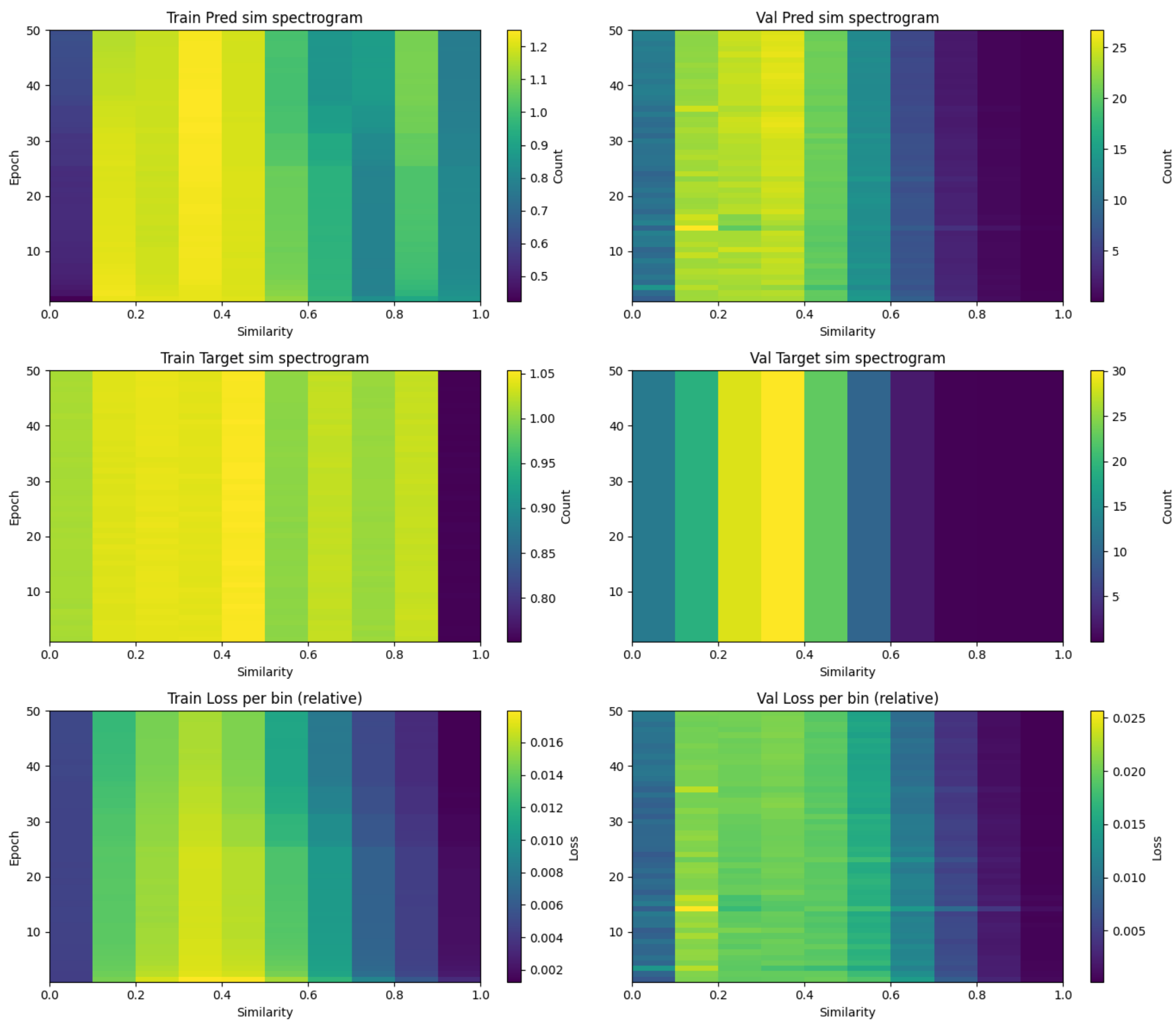

Figure 6.1: Training spectrograms showing: predicted similarity distributions (top), target similarity distributions (middle), and relative MAE per bin (bottom), across epochs for training (left) and validation (right). Numbers may not be real due to confidentiality constraints.

As seen in Figure 6.1, the distribution of ground-truth similarities in the validation set (middle right) remains stable across epochs and is concentrated around mid-range values—typically between 0.2 and 0.6. The validation target distribution closely reflects the natural similarity structure shown in Figure 4.3, which illustrates the unaltered similarity frequencies observed in the opt-in segment space. In contrast, the training targets were intentionally sampled using a weighted strategy to ensure uniform coverage across similarity bins. This design helps the model generalize better across the full similarity spectrum, particularly in underrepresented regions.

The predicted similarity distributions (top row) become increasingly aligned with the ground-truth distributions as training progresses, indicating convergence. In the validation set, the predicted similarity spectrum remains slightly narrower and shifted leftward, suggesting that the model is more conservative when generalizing to unseen data.

The MAE per bin (bottom row) reveals that the model performs best in the central similarity range, where most of the data lies. Conversely, bins at the extremes—very low or very high similarity—exhibit higher errors, due to data scarcity and greater inherent noise. This trend is more pronounced in the validation set, where the model struggles more to predict rare, high-similarity cases.

In addition to the spectrograms, we monitor the global MAE over training epochs for the training, validation, and test sets. At the final epoch, the MAE stabilizes around 0.09 for training, 0.12 for validation, and 0.12 for the test set, confirming the model's ability to generalize to unseen opt-in pairs. These values, while modest in absolute magnitude, are consistent with the continuous and noisy nature of the similarity targets.

Overall, this evaluation confirms that the model is able to reproduce the similarity structure present in the training data with acceptable precision, and that it generalizes reasonably well across the spectrum of behavioral similarity. This latent structure forms the basis for the downstream lookalike retrieval process evaluated in the next section.

6.3. Segment Assignment Evaluation

While the embedding model is trained to regress similarity scores, its practical application lies in assigning segments to non-opt-in users based on proximity in the latent space. To evaluate the effectiveness of this process, we simulate the full lookalike pipeline using opt-in users: each user is embedded, their k nearest neighbors (excluding themselves) are retrieved from the opt-in population, and their segments are inferred based on the aggregated segments of these neighbors.

Since the ground-truth segment assignments are known for all opt-in users, this setup allows us to compare the inferred segments with the actual labels, and to evaluate the system as a multi-label classifier. For each user, the predicted segment set is compared to the true set, and standard classification metrics are computed on a per-segment basis.

The following metrics are used:

- **Precision**: the proportion of predicted segments that are correct.
- **Recall**: the proportion of true segments that were correctly predicted.
- **F1-score**: the harmonic mean of precision and recall.

These metrics are computed individually for each segment, allowing us to understand which clusters are better captured by the embedding space and retrieval logic. Figure 6.2 shows the top segments ranked by F1-score, along with their support (i.e., number of users assigned to each segment in the ground truth).

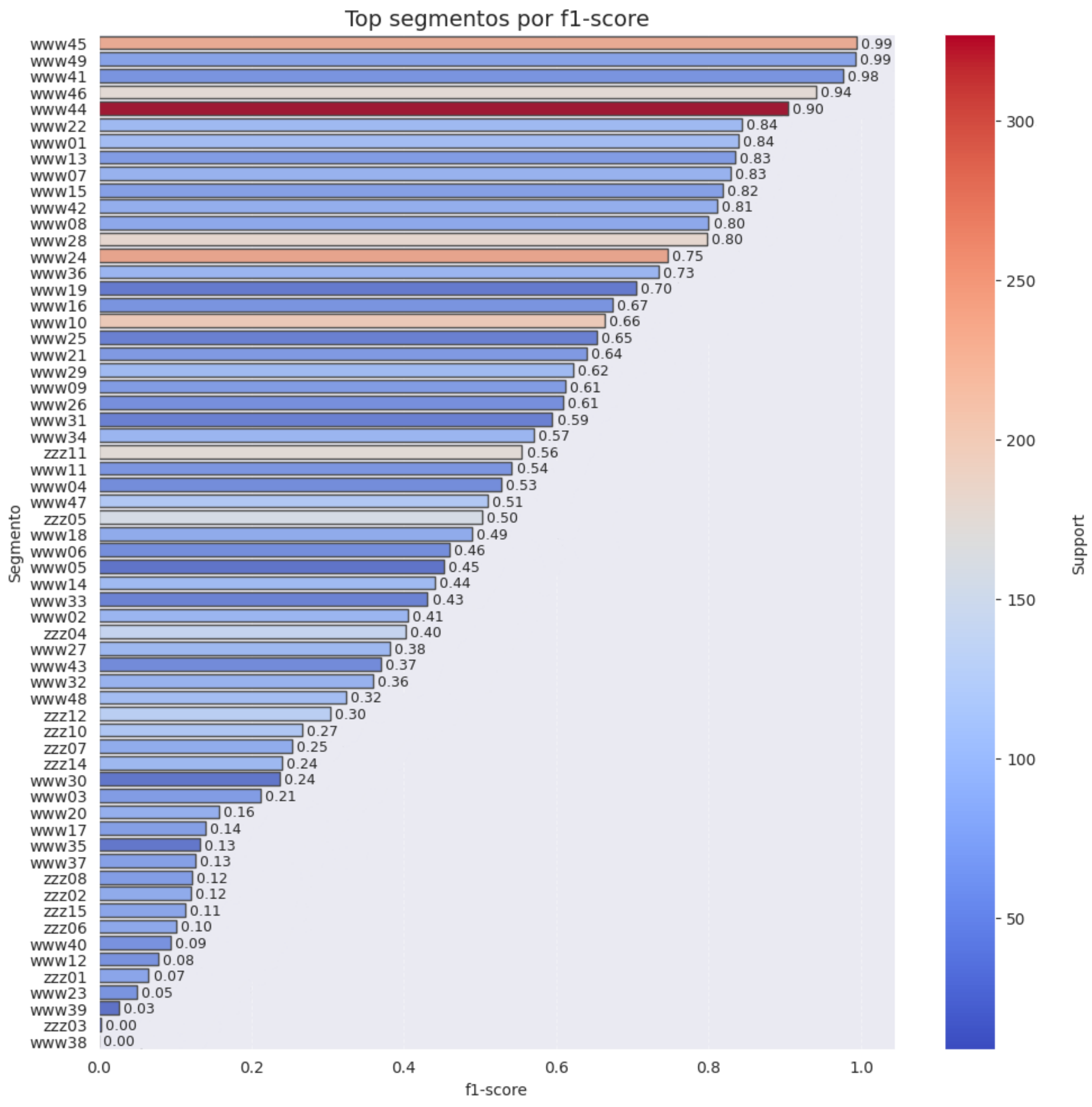

Figure 6.2: Per-segment F1-score for lookalike classification, sorted in descending order. Bar color indicates support. Numbers may not be exact due to confidentiality constraints.

As the figure shows, some segments—particularly those with strong behavioral signals or high prevalence—achieve F1-scores above 0.70, indicating a high degree of recoverability via embedding and neighbor aggregation. Conversely, other segments score well below 0.2, suggesting that their defining traits are either too subtle, too sparse, or strongly dependent on data sources (e.g., web navigation) that were excluded from the model inputs.

To analyze these differences in more detail, Figures 6.3 and 6.4 show the same analysis broken down by precision and recall, respectively.

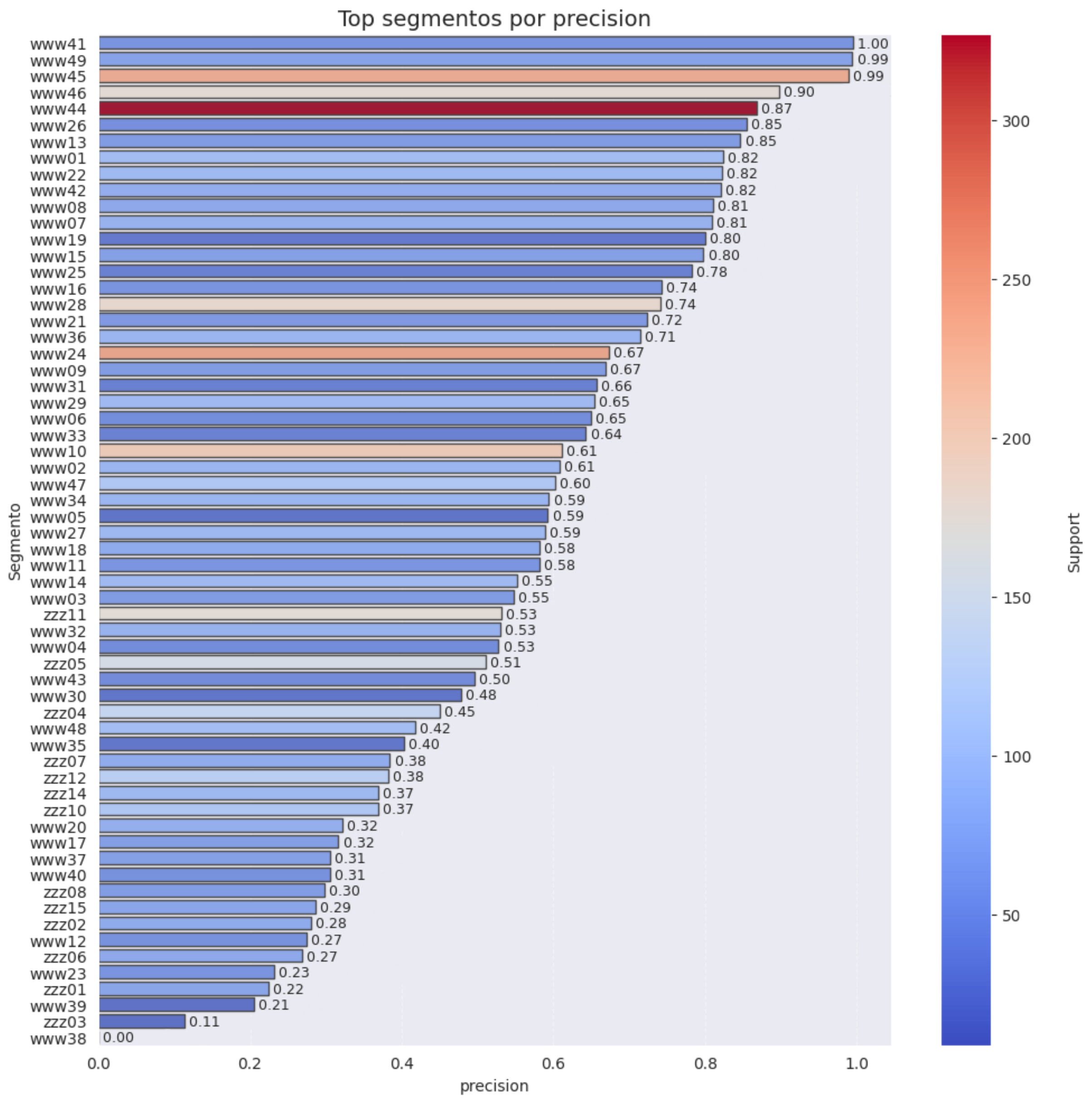

Figure 6.3: Per-segment precision for lookalike classification.

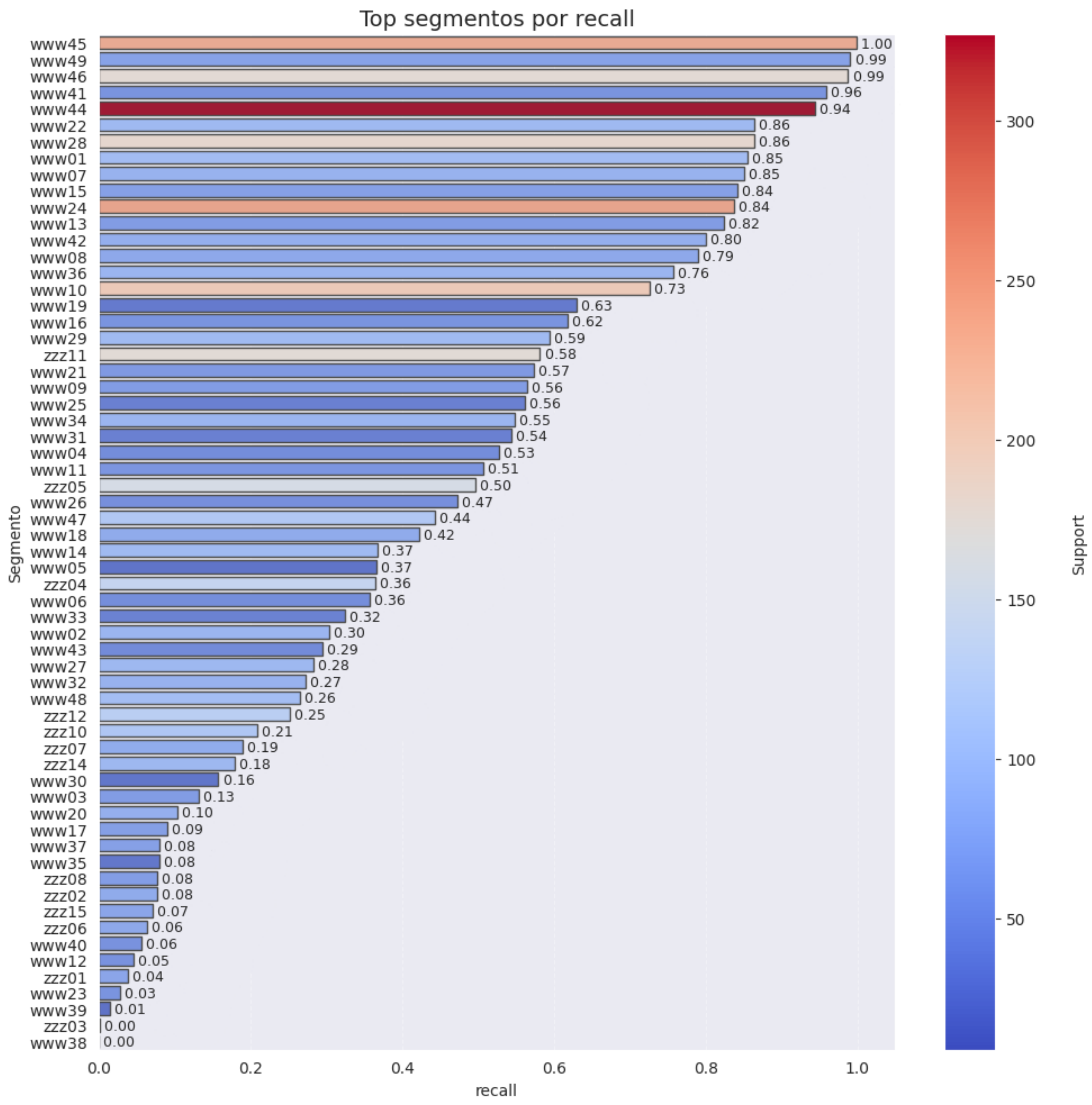

Figure 6.4: Per-segment recall for lookalike classification.

Segments with high precision but low recall tend to be highly specific: the model retrieves users with clear patterns, but fails to generalize to the broader population of true positives. On the other hand, segments with high recall but low precision may be overly generic, capturing large groups but with low discrimination. These trade-offs are influenced by segment size, class imbalance, and feature separability.

To further illustrate the structure of segment prediction errors, Figure 6.5 shows the confusion matrices for a subset of the top segments. Each confusion matrix compares the true and predicted presence of a segment, allowing us to identify systematic misclassifications and segment pairs that are frequently confused.

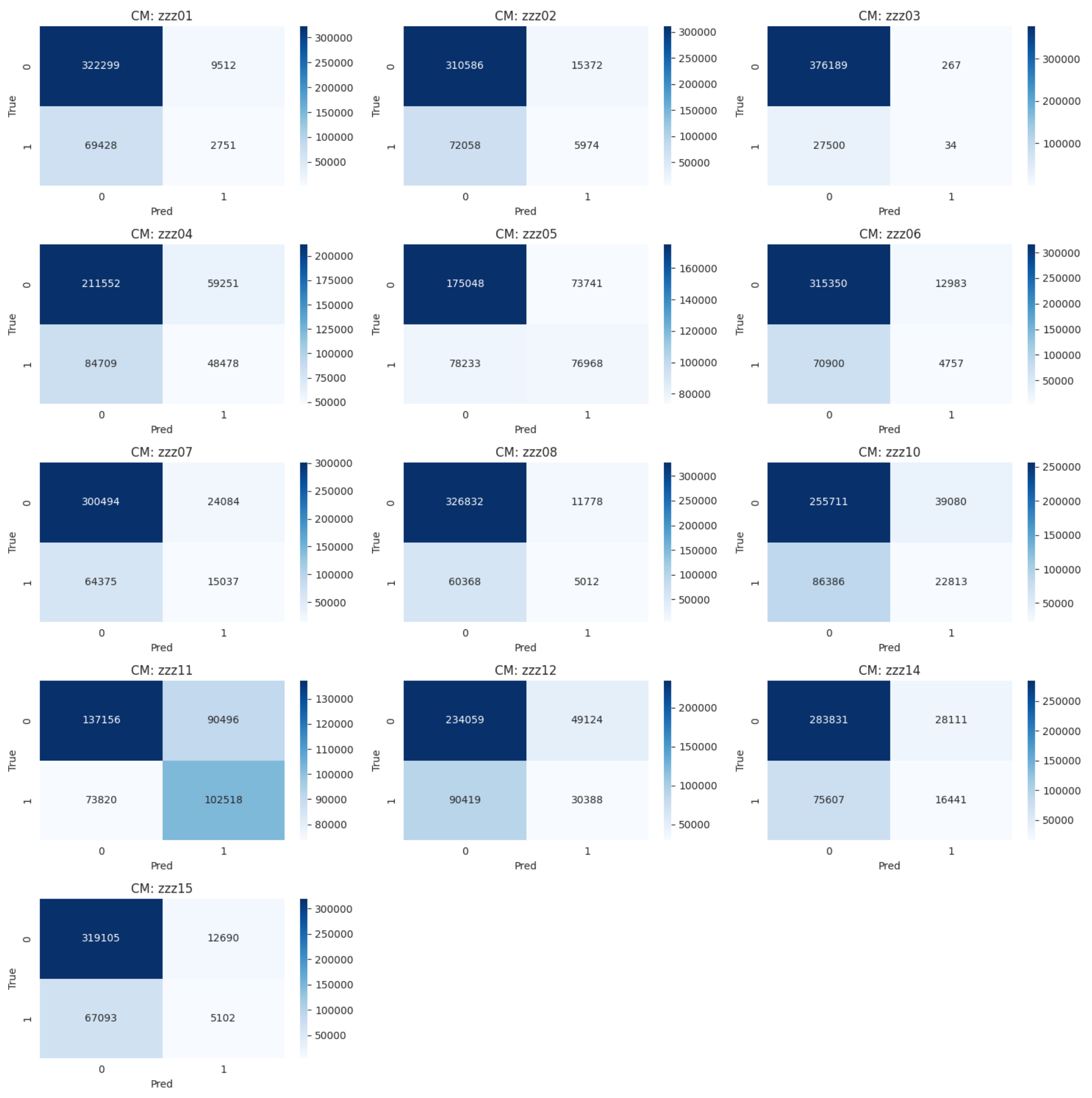

Figure 6.5: Confusion matrices for selected segments. Diagonal dominance indicates correct prediction; off-diagonal elements reveal common misclassifications.

Some of the most common confusions occur between segments that share thematic or behavioral similarities (e.g., overlapping content interests), or where one segment is a subset of another. Segments built primarily from web navigation data—excluded from the model inputs—tend to perform poorly, as expected.

Despite these challenges, the model successfully replicates many of the high-volume and high-value segments used in production campaigns. This confirms the utility of the embedding-based approach for segment extension and audience expansion.

6.4. Segment Volume Expansion

Beyond accuracy metrics and similarity regression performance, one of the most tangible benefits of the lookalike model lies in its ability to increase the number of users assigned to marketing segments. Since the internal segmentation process is limited to opt-in users—who typically represent only a fraction of the total customer base—the overall reach of any campaign is inherently constrained. The lookalike model addresses this limitation by assigning segment labels to non-opt-in users based solely on behavioral similarity in the embedding space.

Figure 6.6 shows the total number of users assigned to each segment, broken down into opt-in and lookalike (non-opt-in) groups. The segments are sorted by opt-in volume, and the increase introduced by the lookalike model is annotated in percentage terms.

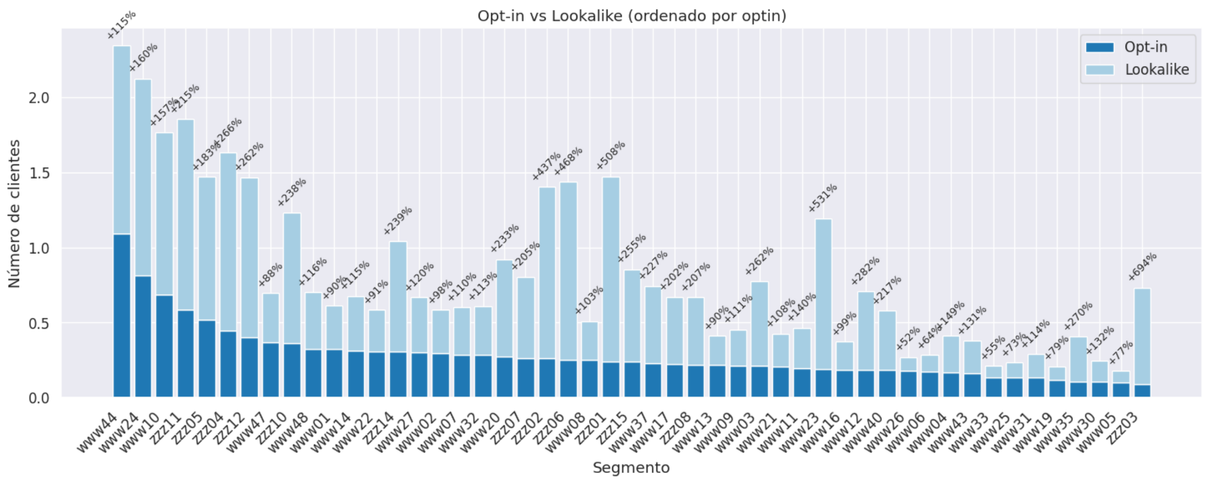

Figure 6.6: Total users per segment, with opt-in and lookalike (non-opt-in) breakdown. Percentage increase denotes growth due to lookalike inference.

As shown, many segments experience a doubling or tripling of their population size, with some growing by over 400%. These gains directly translate into greater reach for campaign targeting, improved budget allocation flexibility, and more robust segment-level analytics. In some cases, segments that were previously underrepresented in the opt-in base now reach critical mass for activation thanks to lookalike expansion.

To better understand the composition shift introduced by the model, Figure 6.7 presents two bar plots: one for opt-in users and one for lookalike users, each showing the proportion of customers assigned to each segment.

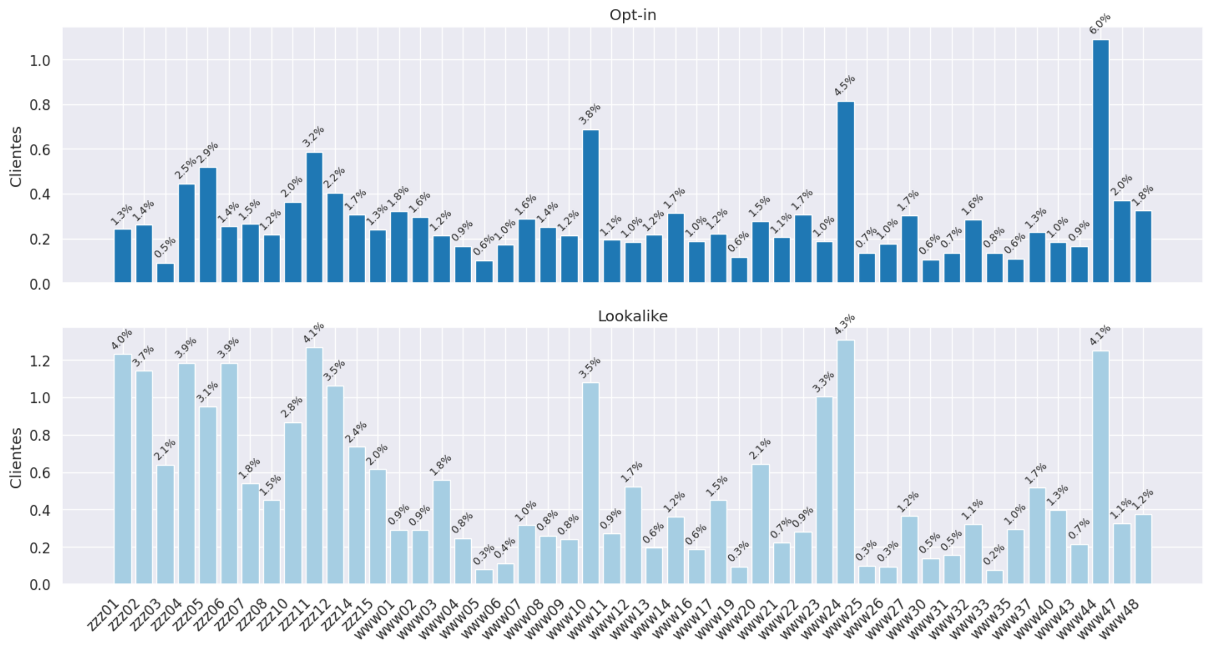

Figure 6.7: Segment distribution for opt-in (top) and lookalike (bottom) users. Values reflect percentage of total users assigned to each segment.

While the overall segment shape is preserved across both groups, some differences emerge—especially in segments that rely heavily on features unavailable to non-opt-in users. These deviations are expected and reflect the limitations of using only legitimate-interest variables for inference.

Still, the general consistency in segment distribution suggests that the model generalizes well and that its outputs are aligned with the original segmentation logic. More importantly, the expanded segment volumes enable broader campaign deployment without requiring changes to the segment definitions or dashboards already used by the marketing teams.

This result highlights the primary operational value of the lookalike system: it increases the number of actionable users while preserving semantic continuity with existing segmentations.

6.5. Discussion of Results

The evaluation results presented in this chapter confirm that the lookalike segmentation model successfully replicates the behavioral patterns used in Telefónica's existing opt-in segmentation pipeline. At the latent space level, the model is able to predict segment-based similarity between user pairs with reasonable precision, as evidenced by the low MAE and the convergence of predicted and target similarity distributions during training. This supports the hypothesis that the two-tower architecture, trained with a regression loss on pairwise similarity, is capable of encoding relevant behavioral structure using only legitimate-interest features.

At the system level, the segment assignment logic built on top of the embedding model yields acceptable performance for many segments. While F1-scores vary widely across clusters, the model reliably reproduces high-traffic and behaviorally distinct segments. Precision and recall analyses suggest that certain segments benefit from higher support or clearer behavioral signals, whereas others suffer from low separability or strong reliance on navigation data excluded from our feature space. The confusion matrices reinforce this picture, revealing common patterns of overlap and misclassification.

Perhaps most importantly, the lookalike model proves valuable in terms of operational impact. By assigning segment labels to non-opt-in users, the model increases the usable population size by over 100% in many segments, with some experiencing four- or five-fold expansion. This volume growth enables broader marketing activation, without requiring changes to the segmentation logic or infrastructure currently used by Telefónica's marketing teams.

Nonetheless, several limitations must be acknowledged. The model is constrained by the availability and quality of legitimate-interest features, which—although extensive—cannot capture all the nuance present in navigation-derived segments. Certain segments, particularly those defined solely through web activity, are inherently difficult to infer and exhibit low recall. Moreover, the lookalike system cannot be used to validate itself on non-opt-in users, due to the absence of ground truth, and must rely on opt-in evaluation as a proxy.

Overall, the results demonstrate that the proposed architecture is effective within its constraints, scalable to production volumes, and capable of producing meaningful business value. Its performance across multiple metrics and its successful integration with Telefónica's existing segmentation pipeline confirm its viability as a complementary tool for extending audience coverage.

Chapter 7
Cost Analysis

To assess the economic viability of deploying the lookalike segmentation system at scale, we analyze the infrastructure and human resource costs associated with both development and weekly production runs. The objective is to understand the trade-offs involved in choosing different hardware configurations and to evaluate whether the system is sustainable in terms of operational cost.

Infrastructure Configuration

Two types of compute environments were used:

- **CPU cluster**: Azure D8s-v3, priced at 0.41€/hour.
- **GPU cluster**: NC48ads-A100-v4 with NVIDIA A100, priced at 8.11€/hour.

The development process spanned approximately 4 months at 25 hours per week, totaling 400 hours. This time was equally divided between CPU-based experimentation and GPU-based training and benchmarking.

Development Costs

	Time	**€/h**	**Total Cost**
CPU cluster	200 h	0.41 €	82 €
GPU cluster	200 h	8.11 €	298 €
Data Scientist	400 h	22 €	8,800 €
Total			**9,180 €**

Table 7.1: Development resource usage and cost

While infrastructure costs for development were modest, the main driver of cost was human effort. The total development effort, equivalent to one junior data scientist at Telefónica over 4 months, accounted for more than 80% of the cost.

Production Costs

Weekly production involves three steps: (i) embedding inference, (ii) FAISS indexing and retrieval, and (iii) segment assignment. Table 7.2 shows the average execution time and associated cost per run, assuming one execution per week.

	Time	**€/h**	**Weekly cost**
CPU cluster	53	0.41 €	0.36 €
GPU cluster	11	8.11 €	1.49 €
Data Scientist	30	22 €	11.00 €
Total			**12.85 €**

Table 7.2: Weekly production cost per configuration

The system was optimized for GPU inference due to its superior speed and overall cost-efficiency. As shown in previous chapters, GPU-based retrieval reduces execution time from nearly one hour (on CPU) to under 15 minutes, allowing the weekly segmentation update to be completed in under 90 seconds of actual compute time. The overall cost of this weekly refresh is just under 13 €, which is negligible relative to the potential business value of expanded segment coverage.

Discussion

Despite using high-end GPU hardware during production, the overall operational cost remains extremely low due to the short runtime and the nature of the batch inference task. Development costs were dominated by human effort, but this is a one-time investment amortized over the lifecycle of the model.

The decision to move FAISS indexing and embedding inference to GPU environments—rather than attempting to parallelize CPU computation—proved both time-efficient and cost-effective. The model design avoids the need for distributed infrastructure and enables reproducibility and fast iteration in a single-node setup.

In conclusion, the lookalike segmentation system is not only technically viable but also operationally sustainable, with low weekly maintenance cost and excellent cost-to-impact ratio.

Chapter 8

Conclusions and Future Work

8.1. Conclusions

This thesis has presented the design, implementation, and evaluation of a lookalike segmentation system for Telefónica's Movistar Plus+ platform. The system addresses a key limitation of the existing marketing segmentation process: its dependence on opt-in users who have consented to share sensitive behavioral data, such as web navigation logs. As a result, large portions of the customer base remain unsegmented and untargeted, limiting the reach and flexibility of marketing campaigns.

To overcome this constraint, we developed a deep learning architecture based on a two-tower (siamese) model that learns to embed customers into a latent space where cosine similarity reflects behavioral resemblance. The model is trained exclusively on non-sensitive features available under legitimate interest, using opt-in segment assignments to supervise a similarity regression task. Once trained, the model is used to generate embeddings for all users, and assign segments to non-opt-in customers via nearest-neighbor search in the embedding space.

The evaluation confirms that the model learns a meaningful latent structure, achieving low prediction error on similarity scores and acceptable segment assignment accuracy across a wide range of segments. In addition, the system demonstrates strong operational performance: inference runs in under 15 minutes on GPU, and the lookalike process increases segment volume by over 100% in many clusters, without modifying existing segmentation logic.

From a business perspective, the solution integrates seamlessly with Telefónica's current marketing infrastructure. It extends the reach of segment-based targeting to users previously excluded from activation, while maintaining compliance with GDPR[1] regulations and internal data policies. The architecture is also efficient and scalable, with low production cost and full compatibility with the company's Databricks-based ecosystem.

8.2. Limitations

While the lookalike segmentation system demonstrates strong performance and practical utility, several limitations must be acknowledged.

First, the model is fundamentally constrained by the data it can access. All input features are restricted to those covered by legitimate interest under GDPR[1]. As a result, any behavioral signals derived from navigation data—often the most discriminative source for marketing

[1] http://data.europa.eu/eli/reg/2016/679/oj

segmentation—are excluded. This impacts the model's ability to infer segments that depend primarily or exclusively on such information.

This limitation is particularly visible in the model's performance on a subset of segments—marked as "zzz" in our analyses—which are defined internally based solely on navigation patterns. These segments exhibit near-zero correlation with the features available to our model, as shown in the feature–segment correlation matrix (Figure 3.2). Consequently, their F1-scores are close to zero, confirming that no model—not even a complex deep learning architecture—can recover behavioral distinctions that are entirely absent from the input space (Figure 6.2). This is a critical shortcoming, as these navigation-driven segments were among the most valuable targets for lookalike expansion.

Second, the evaluation of the model is necessarily limited to opt-in users, as these are the only customers for whom ground-truth segment labels are available. While this allows for rigorous testing, it also introduces a blind spot: we cannot directly measure the accuracy of lookalike assignments for non-opt-in users, who are the true targets of the system. Instead, we rely on opt-in users as proxies, simulating the inference process and comparing against known labels.

A third limitation lies in the very definition of the training target. The similarity label used during training is derived from segment overlap—specifically, cosine similarity between binary segment assignment vectors. While intuitive and operationally simple, this definition is inherently imprecise and manually designed. It does not necessarily reflect a consistent or interpretable notion of behavioral similarity, and likely introduces noise into the learning process. Ideally, this similarity label would be learned or validated by a dedicated upstream system capable of modeling behavioral affinity in a more principled and context-aware manner.

This is especially important for generalization. If Telefónica intends to extend the lookalike framework beyond marketing segmentation—e.g., into churn modeling, content personalization, or customer care—it will be necessary to develop robust, context-specific definitions of what it means for two customers to be "similar". In this sense, our current formulation is both a strength (because it is generic and model-agnostic) and a weakness (because it lacks grounding in measurable outcomes).

This challenge was also acknowledged in the Walmart lookalike system [Peng et al., 2023], where the authors note that the choice of similarity metric defines the scope and behavior of the embedding space. In their framework, different business objectives—such as transactions, visits, or engagement—can be modeled by changing the similarity function used for supervision, while reusing the same architecture. Following this principle, Telefónica could develop multiple latent spaces tailored to different areas of the company, provided that each one is anchored in a well-defined, interpretable behavioral signal.

Finally, the use of FAISS for approximate nearest-neighbor retrieval, while highly performant, limits the system to a single-node architecture. FAISS does not support distributed indexing natively, which may constrain scalability in future use cases where the embedding space becomes much larger or needs to be queried in real time.

Despite these limitations, the model performs well within its design constraints, and lays a solid foundation for further improvements in both modeling capacity and conceptual rigor.

8.3. Future Work

Several lines of improvement and extension could be explored to enhance the performance, flexibility, and generalization ability of the lookalike segmentation model.

The most immediate avenue is the development of a teacher–student architecture. In this setup, a larger, privileged "teacher" model would be trained using both legitimate-interest and navigation-based features (available only for opt-in users), while a smaller "student" model would learn to replicate the teacher's outputs using only the restricted feature set. This approach would allow the student to approximate navigation-driven segment logic, even though it cannot access those features directly. As discussed earlier, this strategy may offer a path to improve lookalike quality for the most challenging segments—those that depend entirely on navigation data.

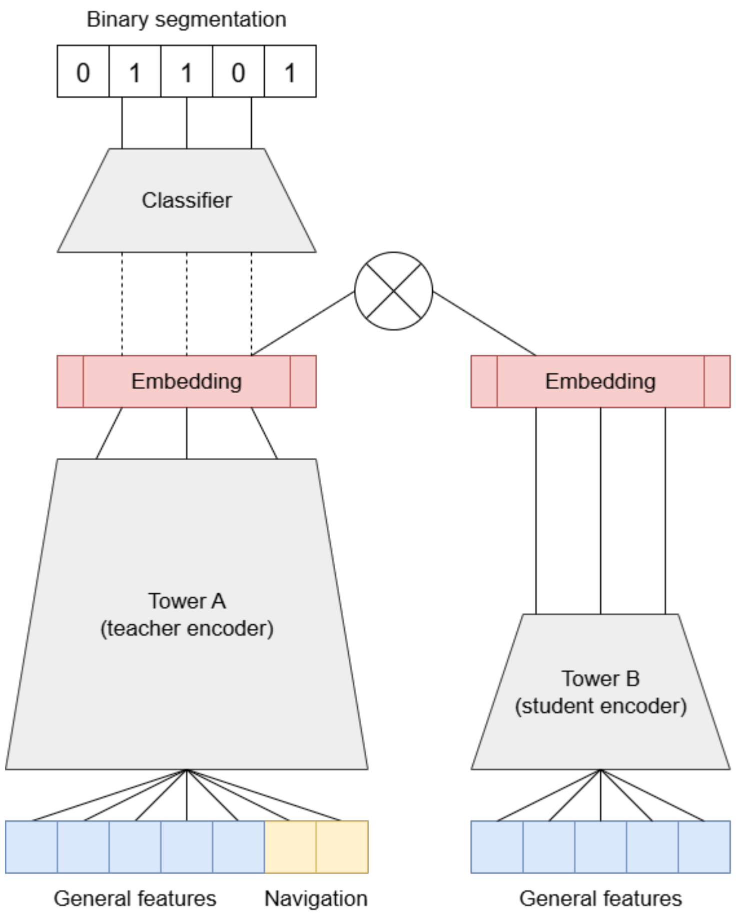

Figure 8.1: Teacher-student two tower model.

Another line of work concerns the incorporation of high-cardinality categorical features, such as user residence or administrative area. These variables are currently excluded from the embedding input due to their dimensionality and sparsity. However, they could be encoded using a dedicated sub-network trained jointly with the towers, or with a separately pretrained embedding model. Given the importance of geographic diversity in content consumption, the addition of this information may yield significant gains in segmentation accuracy.

From an architectural perspective, the current inference pipeline could be further optimized by migrating embedding generation to a Spark UDF. This would enable fully distributed inference across nodes, reducing memory bottlenecks and eliminating the need to export data to local Pandas. Similarly, the use of FAISS could be revisited: while highly performant, its single-node limitation may pose a future bottleneck. Distributed vector databases could offer a more scalable alternative, especially for use cases involving real-time lookups or much larger populations.

Another priority is the development of a more robust and principled definition of similarity. As discussed in Section 8.1, the current similarity label is handcrafted based on segment overlap. While functional, it lacks theoretical grounding and may not generalize across domains. Ideally, Telefónica would develop or learn similarity functions tailored to different business contexts, enabling the reuse of the same architecture across multiple departments. This aligns with the vision outlined in the Walmart framework [Peng et al., 2023], where a change in similarity supervision enables the model to support different business objectives.

Finally, the model could be extended beyond marketing segmentation. The underlying architecture is generic and could be adapted for churn lookalikes, content-based user clustering, or customer service optimization. The latent space itself may serve as a universal user representation, capable of powering multiple downstream applications across Telefónica's ecosystem.

Together, these directions outline a roadmap for improving both the modeling core and its integration into the company's strategic workflows.

Bibliography

Chopra, S., Hadsell, R., & LeCun, Y. (2005). Learning a similarity metric discriminatively, with application to face verification. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.1109/CVPR.2005.202

Choi, S.-S., Cha, S.-H., & Tappert, C. C. (2010). A survey of binary similarity and distance measures. *Journal of Systemics, Cybernetics and Informatics*. https://www.iiisci.org/journal/pdv/sci/pdfs/gs315jg.pdf

Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A unified embedding for face recognition and clustering. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. https://doi.org/10.48550/arXiv.1503.03832

Covington, P., Adams, J., & Sargin, E. (2016). Deep neural networks for YouTube recommendations. *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys)*. https://doi.org/10.1145/2959100.2959190

Johnson, J., Douze, M., & Jégou, H. (2019). Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*. https://doi.org/10.48550/arXiv.1702.08734

Pang, R., Xiong, W., Krueger, D., & et al. (2020). ScaNN: Efficient vector similarity search at scale. https://github.com/google-research/scann

Peng, Y., Liu, C., & Shen, W. (2023). Finding lookalike customers for e-commerce marketing. https://doi.org/10.48550/arXiv.2301.03147

Databricks. (2024). Train recommender models using two-tower architecture. https://docs.databricks.com/aws/en/machine-learning/train-recommender-models

Douze, M., Guzhva, A., Deng, C., Johnson, J., Szilvasy, G., Mazaré, P.-E., Lomeli, M., Hosseini, L., & Jégou, H. (2024). The FAISS library. https://github.com/facebookresearch/faiss