MARMARA UNIVERSITY
INSTITUTE FOR GRADUATE STUDIES
IN PURE AND APPLIED SCIENCES

Telecom Customer Churn Prediction and Root Cause Analysis Using Network Quality Metrics

Büşra Nur Orhan

MASTER THESIS
Department of Electrical and Electronic Engineering

Thesis Supervisor
Prof.Dr. ENGİN MAŞAZADE

ISTANBUL, 2025

Table of Contents

1 INTRODUCTION and OBJECTIVES 11

2 RELATED WORK 13

3 METHODOLOGY xiv

3.1 Data Preparation ii

3.2 KPI Definitions and Their Relevance to Customer Experience iii

3.2.1 cem_tput_kbps – Throughput (kbps) iii
3.2.2 cem_xkb_delay – XKB Delay iii
3.2.3 cem_e2e_delay – End-to-End Delay iii
3.2.4 cem_dl_volume_gb – Downlink Volume (GB) iii
3.2.5 cem_ul_volume_gb – Uplink Volume (GB) iii
3.2.6 cem_cei – Customer Experience Index iv
3.2.7 cem_stream_qoe – Streaming Quality of Experience iv
3.2.8 cem_volte_qoe – VoLTE Quality of Experience iv
3.2.9 cem_voice_qoe – General Voice Call Quality iv
3.2.10 cem_ul_mos – Uplink Mean Opinion Score iv
3.2.11 cem_dl_mos – Downlink MOS v
3.2.12 cem_mo_call_setup_dur – MO Call Setup Duration v
3.2.13 cem_gt_8mb_ratio – Ratio of Users with >8MB Data Sessions v
3.2.14 disco_avg_rsrp – Average RSRP v
3.2.15 disco_avg_rsrq – Average RSRQ vi
3.2.16 disco_avg_uetxpower – Average UE Transmit Power vi
3.2.17 disco_avg_uldmrssinr – Average Uplink DMRS SINR vii
3.2.18 disco_avg_dlmcs – Average DL Modulation and Coding Scheme Index vii
3.2.19 disco_good_signal_rate – Good Signal Rate vii
3.2.20 main_congested – Cell Congestion Indicator viii
3.2.21 main_rate_4g_volume – 4G Data Volume Ratio viii
3.2.22 main_num_call_drop_total – Total Call Drops viii
3.2.23 main_dropped_4 – LTE Call Drops viii
3.2.24 main_dropped_3 – 3G Call Drops viii
3.2.25 cem_avg_30d_cem_e2e_delay_overall – 30-Day Avg. E2E Delay ix
3.2.26 cat_e2e_delay_overall_encoded – Categorical Encoded E2E Delay ix
3.2.27 cem_mo_call_setup_succ_rate – MO Call Setup Success Rate ix
3.2.28 cem_mt_call_setup_succ_rate – MT Call Setup Success Rate ix
3.2.29 cem_mo_call_drop_rate – MO Call Drop Rate ix
3.2.30 cem_mt_call_drop_rate – MT Call Drop Rate x
3.2.31 cem_ul_silent_call_ratio – Uplink Silent Call Ratio x
3.2.32 cem_dl_silent_call_ratio – Downlink Silent Call Ratio x
3.2.33 cem_less_1mb_ratio – Data Sessions <1MB Ratio x
3.2.34 cem_1mb_3mb_ratio – Data Sessions between 1MB–3MB x
3.2.35 cem_3mb_5mb_ratio – Data Sessions between 3MB–5MB xi
3.2.36 cem_gt_8mb_ratio – Data Sessions >8MB xi
3.2.37 disco_good_signal_rate – Good Signal Rate xi
3.2.38 disco_fair_signal_rate – Fair Signal Rate xi
3.2.39 disco_bad_signal_rate – Bad Signal Rate xi
3.2.40 disco_bad_rsrq_rate – Bad RSRQ Rate xii
3.2.41 disco_bad_ulsnr_rate – Bad Uplink SINR Rate xii
3.2.42 disco_bad_dlmcs_rate – Bad Downlink MCS Rate xii

3.3 An Exploratory Data Analysis (EDA) xii

3.3.1 Data Integration and Preprocessing xii
3.3.2 Data Cleaning xvi
3.3.3 Handling Missing Values xvi

3.3.4 Outlier Detection and Treatment xvii
3.3.5 Data Type Corrections xx
3.3.6 Removing Duplicates xx
3.3.7 Data Sampling xx
3.3.8 Exploratory Visualization of Sampled Data xxi
3.3.9 Ordinal Encoding and Feature Scaling for Model Readiness xxviii

3.4 Datasets xxix
3.4.1 Dataset 1: 30-day average KPIs per customer xxix
3.4.2 Dataset 2: 30-Day Daily Time Series per Customer xxxi

3.5 Clustering and Segment Analysis xxxii
3.5.1 KMeans Clustering xxxii
3.5.2 Bisecting KMeans Clustering xxxiv
3.5.3 Evaluating and Selecting Clustering Models xxxvi
3.5.4 Comparison of K-Means and Bisecting K-Means Clustering xxxviii

3.6 Feature Selection (Churn Drivers) xxxviii

3.7 Churn Drivers Threshold Identification with Weight of Evidence xlii

3.8 Churn Prediction xlviii
3.8.1 Churn Prediction Models xlix
3.8.2 Evaluation Methodology lviii
3.8.3 Churn Prediction Using Static Data (Dataset 1) lix
3.8.4 Time Series Prediction Using LSTM with Dataset 2 lxiv

4 EXPERIMENTAL RESULTS lxviii

5 CONCLUSION AND FUTURE WORK lxix

6 APPENDICE lxix

ÖZET

Ağ Kalite Metrikleri Kullanılarak Telekom Müşteri Kaybı Tahmini ve Kök Neden Analizi

Bu çalışma, telekomünikasyon sektöründe müşteri kaybının (churn) temel nedenlerini belirlemeyi ve ağla ilgili göstergeler (KPI'lar) kullanarak müşteri kaybını tahmin etmeyi amaçlamaktadır. Daha önceki birçok çalışmanın faturalama veya kampanya temelli değişkenlere odaklanmasının aksine, bu çalışma yalnızca gerçek müşteri deneyimini yansıtan ağ performansı göstergelerine odaklanmıştır. Bu göstergeler; akış (streaming) kalitesi, uçtan uca gecikme, VoLTE QoE ve genel ağ kalite göstergelerini kapsamaktadır. Veri kümesinde ekonomik veya tarife bilgileri bulunmadığından, elde edilen sonuçlar yalnızca ekonomik olmayan müşteri kaybı etmenlerinin etkisini yansıtmaktadır.Her bir müşteri için 2G, 3G ve 4G teknolojileri üzerinden günlük KPI'ların 30 günlük ortalamaları hesaplanmış ve kümeleme analizi için temsili bir örneklem seçilmiştir. Müşteri segmentasyonu için Bisecting K-Means algoritması uygulanmıştır. Düşük ağ performansına sahip segmentin, diğer gruplara kıyasla anlamlı derecede yüksek müşteri kaybı gösterdiği belirlenmiş; bu durum, bozulan ağ deneyimi ile müşteri kaybı davranışı arasında güçlü bir ilişki olduğunu ortaya koymuştur.Ardından, müşteri kaybı üzerinde en etkili değişkenleri belirlemek amacıyla ANOVA testi uygulanmış ve yalnızca istatistiksel olarak anlamlı değişkenler kullanılarak çeşitli makine öğrenmesi modelleri eğitilmiştir. Modellerin kesinlik (precision) ve geri çağırma (recall) skorları karşılaştırılarak en iyi performans gösteren model belirlenmiştir. Başlangıç sonuçlarının istenilen düzeyde olmaması üzerine, İstanbul'un Kadıköy, Şişli, Üsküdar ve Beşiktaş ilçelerine ait müşteri başına 30 günlük günlük zaman serisi verileri kullanılarak daha odaklı bir durum analizi gerçekleştirilmiştir. Müşteri kaybı tahmini, aynı dönem içerisinde churn eden müşteri sayısıyla eşit sayıda churn etmeyen müşterinin örneklenmesiyle oluşturulan dengeli veri kümesi üzerinde yapılmış ve %59 F1 skoru elde edilmiştir.

Temmuz 2025 Büşra Nur Orhan

SUMMARY

Telecom Customer Churn Prediction and Root Cause Analysis Using Network Quality Metrics

This study aims to identify the root causes of customer churn in the telecommunications industry and to predict churn using network-related indicators. Unlike many previous studies that rely on billing or campaign-related variables, this work focuses solely on network performance metrics (KPIs) that reflect real customer experience — including streaming quality, end-to-end delay, VoLTE QoE, and overall network quality indicators. As the dataset does not contain any economic or tariff-related information, the results specifically reflect the impact of non-economical churn drivers. A 30-day average of daily KPIs was calculated for each customer across 2G, 3G, and 4G technologies, and a representative sample was selected for clustering. Bisecting K-Means algorithms were applied to segment customers. The segment identified with poor network performance showed significantly higher churn compared to other groups, revealing a strong link between degraded network experience and churn behavior. Subsequently, ANOVA was used to identify the most influential features on churn, and several machine learning models were trained using only the statistically significant variables. The models' precision and recall scores were compared to determine the best-performing model. As the initial results were not satisfactory, a more focused case study was conducted using 30-Day Daily Time Series per Customer data for the Kadıköy, Şişli, Üsküdar, Beşiktaş districts of Istanbul. Churn prediction was then performed on a balanced dataset, created by sampling an equal number of non-churners to match the churners in the same period, achieving an F1-score of 59%.

July 2025 Büşra Nur Orhan

STATEMENT

Telecom Customer Churn Prediction and Root Cause Analysis Using Network Quality Metrics

Hereby I declare that this thesis study is my own work I had no unethical behavior in any stage from planning of the thesis until writing it I have obtained all the information in this thesis within the academic and ethical rules I have cited all the information and comments that are not obtained with this thesis study and these sources are also included in the list of references I hereby declare that I have no infringement of patents and copyrights during the study and writing of this thesis.

Name and Surname of the Student

Signature

ABBREVIATIONS LIST

KPI : Key Performance Indicator

KQI : Key Quality Indicator

DPI : Deep Packet Inspection

LSTM : Long Short-Term Memory

CEM : Customer Experience Management

NW : Network

NQ : Network Quality

NPS : Net Promoter Score

WoE. : Weight of Evidence

FIGURE LIST

Page Number

Figure 1: MOS Quality Categorization v
Figure 2: RSRP & RSRQ Signal Quality Categorization vi
Figure 3: RF Conditions Categorizations vii
Figure 4 : Distribution of Categorical Variables xxiv
Figure 5: Distribution of Numerical Variables-1 xxiv
Figure 6: Distribution of Numerical Variables-2 xxv
Figure 7: Distribution of Numerical Variables-3 xxv
Figure 8: Distribution of Numerical Variables-4 xxvi
Figure 9: Distribution of Numerical Variables-5 xxvi
Figure 10: Distribution of Numerical Variables-6 xxvii
Figure 11:Distribution of E2E Delay xxvii
Figure 12: K-Means Clustering Evaluation: (a) Elbow Method Showing the Relationship Number of Clusters and SSE; (b) Silhoe Score Across Different k values Highlighting Optimal Cluster Separation xxxiv
Figure 13: Bisecting K-Means Clustering Evaluation: (a) Elbow Method Showing the Relationship Number of Clusters and SSE; (b) Silhoe Score Across Different k values Highlighting Optimal Cluster Separation xxxvi
Figure 14: Bisecting K-Means Clustering Evaluation: (a) Elbow Method Showing the Relationship Number of Clusters and SSE; (b) Silhoe Score Across Different k values Highlighting Optimal Cluster Separation xxxvii
Figure 15: Logistic Regression (LR)4 xlix
Figure 16 : Decision Tree l
Figure 17: Random Forest Tree li
Figure 18: Artificial neural network (ANN) structure with three inputs, one output and multi hidden nodes liv
Figure 19: k Nearest Neighbors liv
Figure 20 : Seq2Seq-LSTM Architecture lviii
Figure 21: Confusion Matrix and Their Formulas lix
Figure 22 : Churn Risk Labelling via Probability Thresholding lx
Figure 23 : LSTM Confusion Matrix-1 lxv
Figure 24: LSTM Confusion Report and Matrix-2 lxv

TABLES LIST

Table I: Data Science Problem Solving Strategy ii
Table II: Summary Statistics of the Filtered Dataset After Outlier Removal xx
Table III: NPS SCORES DISTRIBUTION xxxi
Table IV: Time series data Churner Distribution xxxii
Table V: Cluster-wise Summary of KPI Medians Churn Rates and NPS Scores xxxvii
Table VI : Anova & Chi-Square P-Value Classification xxxix
Table VII: ANOVA TEST RESULTS xli
Table VIII: WOE CEM_SILENT_CALL_ RATIO RESULT xliii
Table IX: WoE cem_avg_30d_e2e_delay_overall result xliii
Table X: LSTM CLASSIFICATION REPORT-1 lxv
Table XI : Model Comparisons lxviii

CODE SECTION LIST

Code Section 1: Impala Code for Table Creating xiv
Code Section 2: Spark Code for Missing Value Detection xvii
Code Section 3: Spark Code for Outlier Detection xix
Code Section 4: Stratified Sampling Code with Spark xxi
Code Section 5: Visualization of Categorical and Numerical Variable Distributions via Histograms and Boxplots xxiii
Code Section 6: Label Encoding of Categorical Features xxviii
Code Section 7: Feature Scaling xxix
Code Section 8: KMeans Clustering with Model Evaluation (SSE and Silhouette Score) xxxiii
Code Section 9:Bisecting KMeans Clustering with Model Evaluation (SSE and Silhouette Score) xxxv
Code Section 10: Anova & Chi Square Analysis with Pandas Code xl
Code Section 11: WoE(Weight of Evidence) Code xlvii
Code Section 12:Evaluating ML Models for Churn Detection on Dataset 1 -Part 1 lxi
Code Section 13: Evaluating ML Models for Churn Detection on Dataset 1 -Part 2 lxii
Code Section 14: Evaluating ML Models for Churn Detection on Dataset 1 -Part 3 lxiii
Code Section 15: LSTM Model for Churn Prediction Dataset 2 -Part 1 lxvi
Code Section 16: LSTM Model for Churn Prediction Dataset 2 -Part 2 lxvii

1 INTRODUCTION and OBJECTIVES

Customer churn which refers to the loss of existing customers to competitors poses a significant challenge for companies especially in the telecom industry. Since retaining existing customers is more cost-effective than acquiring new ones predicting churn and improving customer loyalty are critical. Telecom customers generally churn due to economic or non-economic reasons. Given the ease of switching providers high churn rates can severely impact profitability making churn prediction a priority for telecom companies.

The telecommunications industry is characterized by a high customer churn rate which typically ranges from 20% to 40% annually [1]. Managing customer retention has become a strategic priority as it is estimated that preserving existing customers is 5 to 10 times more cost-effective than acquiring new ones. Furthermore churn prediction is significantly more economical—up to 16 times cheaper—than attempting to gain new customers. Notably even a modest 5% reduction in churn has the potential to increase profits by 25% to 85% highlighting the critical business value of predictive retention strategies. To effectively address churn and maximize customer lifetime value companies must analyze consumer profiles and implement business segmentation techniques. In this context deep learning and behavioral analytics play a pivotal role enabling firms to personalize and optimize marketing activities particularly on social media platforms. Additionally deep learning algorithms allow for automation and enhanced efficiency in advertising processes improving overall performance while conserving time and resources [1]. Identifying customers at risk of churning and understanding the technical reasons behind their dissatisfaction is essential for preventing revenue loss. Telecom operators strive to detect issues—particularly related to network quality and service performance—that drive customers to switch providers. By analyzing customer behavior patterns without direct inquiry companies can design targeted retention strategies. This study focuses specifically on technical factors affecting churn excluding economic or personal reasons to highlight the direct impact of infrastructure and service performance on customer experience. Several key performance indicators (KPIs) such as throughput (kbps) downlink/uplink volume (GB) rsrp (reference signal received power) rsrq (reference signal received quality) end-to-end delay streaming quality of experience volte quality of experience downlink/uplink mos (mean opinion score) MO/MT call setup duration MO/MT call drop duration downlink/uplink silent call ratio and cell congestion indicator have been utilized to assess how network quality affects the customer experience. The study is conducted using real-world Vodafone Turkey data where representative samples are taken from different groups across the entire network to ensure that the dataset reflects the overall structure and diversity of the telecom infrastructure. A more detailed explanation of the selected KPIs and their roles in

modeling churn-related network experience is provided in section as KPI Definitions and Their Relevance to Customer Experience".

Despite its potential to improve customer retention strategies this study also faced several critical challenges inherent to real-world telecom environments. One major difficulty stems from the rapidly evolving preferences and expectations of consumers which makes it difficult to maintain static churn models over time. Additionally the nature of the telecom sector involves big data characteristics including high volume velocity variety veracity and value all of which impose significant technical and analytical burdens. Another challenge is the high demand for telecom services which can lead to unpredictable usage patterns and performance variability across different network segments. Moreover imbalanced classification caused by the unequal distribution of churned versus retained customers complicates the training and evaluation of machine learning models. Lastly machine learning models may not perform efficiently when applied to extremely large datasets especially without careful feature engineering data reduction or distributed computing strategies. These issues necessitate robust data handling techniques and continuous model tuning to ensure scalable and generalizable results.

Another important limitation of this study is the absence of customer-level economic features such as monthly billing amounts or subscribed tariff plans. Since the churn label inherently reflects both technical and non-technical reasons—including service quality pricing dissatisfaction and personal usage behavior—it becomes difficult to isolate the specific impact of network-related factors alone. As a result compared to other studies in the literature that incorporate economic and demographic features the performance metrics (such as precision recall or AUC) observed in this study may be expected to be relatively lower. Nevertheless this limitation reinforces the robustness of the current approach as it focuses solely on technical indicators and still aims to derive actionable insights for customer retention strategies without relying on commercial data.

In real-world churn prediction overall accuracy is inadequate due to class imbalance and business objectives. The goal is not to identify customers who will stay but to detect those at risk of churning. Therefore precision and recall are the key evaluation metrics. After applying various machine learning models—including neural networks—for churn prediction on the static dataset we observed that recall scores were relatively acceptable while precision remained low. This indicated that many non-churners were misclassified as churners triggering unnecessary retention actions and increasing operational costs. Hence balancing recall and precision is essential to effectively capture true churners while minimizing false positives. To address this issue we developed a second approach using time series data and applied LSTM models to a smaller customer set in the Kadıköy district in Istanbul. This targeted and resource-efficient method improved model performance by capturing churners more precisely based on recent behavioral trends .

The benefits of the analysis results we will obtain with the help of deep learning and traditional artificial intelligence models are listed below:

- Reducing call center costs and workload by proactively solving problems before customer complaints
- Increasing company brand value and gaining competitive advantage
- Developing effective customer retention strategies
- Ensuring customer satisfaction and loyalty
- Understanding customer behavior and the main factors behind churn
- Identifying customers at high risk of churn and the top drivers behind it
- Reducing churn rate and increasing customer retention
- Enhancing targeted marketing and improving products and services
- Supporting financial planning and forecasting
- Increasing company revenue and minimizing potential income loss
- Achieving a 5–15% increase in customer retention rate within six months

The information presented in this section has been compiled based on the findings reported in [13]–[15].

2 RELATED WORK

One relevant study by Saha et al. [2] proposed a deep learning-based churn prediction framework for the telecommunications industry comparing the performance of various machine learning and ensemble methods on two public telecom datasets. The authors evaluated traditional classifiers such as Logistic Regression Decision Trees and k-Nearest Neighbors alongside advanced models including Random Forest Gradient Boosting AdaBoost and deep architectures such as Artificial Neural Networks (ANN) and Convolutional Neural Networks (CNN). Their results indicated that CNN and ANN achieved the highest prediction performance reaching up to 99% accuracy in both datasets. The study emphasized the importance of deep learning in capturing complex churn patterns and used statistical tests such as ANOVA and the Wilcoxon test to validate the performance differences among models. Unlike our study which focuses solely on technical KPIs derived from infrastructure-level performance data their work used customer-level datasets that included demographic and service usage features.

Another deep learning-based approach was proposed by Fujo et al. [3] who developed churn prediction models specifically tailored for the telecommunications sector. The study focused on evaluating the performance of several deep learning techniques particularly Artificial Neural Networks (ANN) and Recurrent Neural Networks (RNN) for detecting churn-prone customers. Their work emphasized the ability of neural architectures to capture non-linear patterns and customer behavior trends over time. The dataset used in the study included service usage records and customer activity data. Their findings showed that deep learning models especially RNNs achieved higher accuracy and robustness compared to conventional machine learning methods. While their study incorporated time-dependent behavioral variables it did not isolate technical network indicators as in our work which focuses on infrastructure -level KPI analysis.

3 METHODOLOGY

According to the churn estimation literature review all relevant past studies emphasize that the use of artificial intelligence technologies is highly important in reducing customer churn rates in the telecommunications industry. By applying machine learning algorithms customer behavior can be predicted churn rates can be minimized and telecommunication companies can gain a competitive advantage by addressing major issues related to customer loss.

In the previous studies discussed in the related works section we observed that customer service-based KPI & KQI metrics were generally not utilized. In contrast our study aims to evaluate whether better results can be achieved by leveraging real service-based KPI & KQI (customer network quality metrics) data from Vodafone Turkey. While a large portion of existing studies in the literature has concentrated on applying machine learning algorithms to predict customer churn, relatively few have placed emphasis on uncovering the key factors that contribute to churn using these techniques. However, identifying such influential variables is crucial for enhancing customer relationship management practices in the telecommunications sector, enabling firms to implement more effective customer retention policies. In light of this, the present study adopts a combination of statistical and machine learning-based approaches not only to estimate churn likelihood but also to gain insights into the underlying drivers of customer attrition.

Another distinguishing aspect of our study is its focus on understanding the underlying reasons behind customer churn. Rather than solely predicting churn we aim to analyze which customer segments—based on their network performance—are at higher risk of churn and to determine which critical KPI thresholds lead to increased churn likelihood. This will provide deeper

insights into how degraded network experience influences customer behavior and will contribute to more targeted retention strategies.

To achieve our goals the following data science solution strategy will be used respectively.

Business Understanding	Data Acquisition & Understanding	Data Preparation	Modeling	Deployment	Customer Acceptance	Additional Steps
• Define objectives • Define success metrics • Prepare charter document • Specifying data sources • Acquiring data sources and dictionaries	• Prepare data quality report • Solution Architecture • Checkpoint decision	• Data Cleaning • Normalization • Standardization • Feature Engineering	• Model Development • Model Training • Model Evaluation	• Scoring Performance Monitoring	• Stage Off • System Validation • Prepare Exit Report • Launch	• Automation • Reporting • Visualization • Improvement • Model Extension

Table I: Data Science Problem Solving Strategy

3.1 Data Preparation

An exploratory data analysis (EDA) was performed to acquiring the important relations and comprehend the dataset. The dataset used in this study was generated by the Huawei SmartCare System a comprehensive network performance and monitoring platform that leverages Deep Packet Inspection (DPI) technology to aggregate customers daily service experience through various Key Performance Indicators (KPIs). These KPIs are derived from actual user traffic and reflect real customer experience across multiple dimensions of network performance. A detailed explanation of all 42 KPIs used in the analysis—including their technical definitions measurement methodologies customer impact and service relevance—is provided in the following section of this document. During this step patterns and relationships between variables were revealed using correlation heatmaps and the distribution of key features was examined through summary statistics (describe()) and boxplots. These analysis helped reveal variable spread potential skewness and value ranges across KPIs. Missing values were identified and removed to ensure data completeness and outliers were detected using interquartile range (IQR)-based filtering. Following EDA all numerical features were scaled appropriately to bring them to a comparable range prior to modeling. EDA also supported the generation of early assumptions and guided feature relevance assessments before the modeling phase. The methodological steps described above in the EDA phase are discussed in the following section accompanied by the important patterns and observations uncovered during the analysis.

C1 Public

3.2 KPI Definitions and Their Relevance to Customer Experience

3.2.1 cem_tput_kbps – Throughput (kbps)

- **Definition:** Indicates the average data transmission rate (in kilobits per second) for a user session.
- **Measurement:** Calculated based on data volume transferred over time.
- **Implications of increase/decrease:** Higher values indicate better network performance. Low throughput suggests congestion or signal quality issues.
- **Impact on user experience:** Low throughput results in slow downloads buffering during video streaming and poor browsing experience.
- **Affected services:** Video streaming web browsing app updates.

3.2.2 cem_xkb_delay – XKB Delay

- **Definition:** Measures the delay experienced during transmission of small-sized packets (e.g. 10KB).
- **Measurement:** Time taken for 10KB packets to be delivered.
- **Implications of increase/decrease:** High values reflect poor radio conditions or backhaul latency.
- **Impact on user experience:** Affects responsiveness of applications especially web loading and interactive apps.
- **Affected services:** Browsing social media app loading.

3.2.3 cem_e2e_delay – End-to-End Delay

- **Definition:** Total latency from source to destination covering all network segments.
- **Measurement:** Round-trip time across core radio and transport layers.
- **Implications of increase/decrease:** Increase signals core congestion or poor routing.
- **Impact on user experience:** High delay leads to lag in real-time applications.
- **Affected services:** VoIP video conferencing online gaming.

3.2.4 cem_dl_volume_gb – Downlink Volume (GB)

- **Definition:** Measures total data volume downloaded by users in gigabytes.
- **Measurement:** Sum of downlink data over a defined period.
- **Implications of increase/decrease:** Reflects user activity level; high values may indicate heavy usage areas or efficient service.
- **Impact on user experience:** Not a direct quality metric but informs capacity and traffic trends.
- **Affected services:** All download-intensive services.

3.2.5 cem_ul_volume_gb – Uplink Volume (GB)

- **Definition:** Total volume of data sent by users in gigabytes.
- **Measurement:** Aggregated uplink data over time.
- **Implications of increase/decrease:** High values could point to video calls live uploads or IoT activity.
- **Impact on user experience:** Poor uplink may degrade call quality and slow uploads.
- **Affected services:** VoIP video upload cloud services.

C1 Public

3.2.6 cem_cei – Customer Experience Index

- **Definition:** Composite KPI reflecting overall network experience based on latency speed and coverage.
- **Measurement:**Derived from a weighted combination of sub-metrics.
- **Implications of increase/decrease:** High CEI indicates good network quality; low CEI triggers performance investigations.
- **Impact on user experience:** Summarizes holistic user satisfaction level.
- **Affected services:** All.

3.2.7 cem_stream_qoe – Streaming Quality of Experience

- **Definition:** Evaluates quality of streaming services based on buffering resolution and start time.
- **Measurement:** Based on technical KPIs and user session analysis.
- **Implications of increase/decrease:** Low values highlight streaming performance issues.
- **Impact on user experience:** Affects perception of video and audio streaming apps.
- **Affected services:** YouTube Netflix Spotify.

3.2.8 cem_volte_qoe – VoLTE Quality of Experience

- **Definition:** Measures user-perceived quality during VoLTE calls.
- **Measurement:** Combines call setup time call stability and voice clarity.
- **Implications of increase/decrease:** Drop indicates poor voice encoding or handover issues.
- **Impact on user experience:** Leads to choppy audio or dropped VoLTE calls.
- **Affected services:** Voice over LTE (VoLTE).

3.2.9 cem_voice_qoe – General Voice Call Quality

- **Definition:** Overall assessment of voice call experience.
- **Measurement:** Aggregated MOS call drops and setup success rate.
- **Implications of increase/decrease:** Decline reflects degradation in voice network performance.
- **Impact on user experience:** Poor quality discourages repeat use.
- **Affected services:** All voice services.

3.2.10 cem_ul_mos – Uplink Mean Opinion Score

- **Definition:** Subjective score indicating uplink audio clarity as perceived by the receiver.
- **Measurement:** Derived from codec analysis and background noise suppression.
- **Implications of increase/decrease:** Lower scores imply poor microphone capture or uplink noise.
- **Impact on user experience:** Users sound unclear or cut out during calls.
- **Affected services:** VoLTE VoWiFi VoIP.

C1 Public

MOS	Characteristics
5	• Excellent • Like face-to-face or radio reception
4	• Good • Imperfections can be perceived, but sound is still clear
3	• Upper range (3.8+) considered acceptable • Mid-to-lower range considered fair-to-annoying with CX risk
2	• Poor, difficult to understand the caller • Annoying, distracting
1	• Bad, very difficult to understand the caller • Difficulty handling the call efficiently or achieving resolution

Figure 1: MOS Quality Categorization

3.2.11 cem_dl_mos – Downlink MOS

- **Definition:** Mean Opinion Score for the downlink audio path assessing how clearly the user can hear the other party.
- **Measurement:** Calculated using parameters like jitter packet loss and codec performance on the downlink.
- **Implications of increase/decrease:** Low DL MOS implies poor voice reception such as muffled or choppy sound.
- **Impact on user experience:** Users struggle to understand the speaker during calls.
- **Affected services:** VoLTE VoWiFi OTT voice calls.

3.2.12 cem_mo_call_setup_dur – MO Call Setup Duration

- **Definition:** Measures the time interval between initiating a mobile-originated call and its successful connection.
- **Measurement:** Calculated as the time from dialling to the point where the call is successfully established.
- **Implications of increase/decrease:** Long setup times may reflect signaling inefficiencies or congestion in the call setup path.
- **Impact on user experience:** Longer waits before call ringing lead to frustration or premature hang-ups. **Affected services:** Voice calls (VoLTE CSFB).

3.2.13 cem_gt_8mb_ratio – Ratio of Users with >8MB Data Sessions

- **Definition:** Percentage of users who have experienced data sessions with more than 8 MB transferred.
- **Measurement:**Number of sessions >8MB divided by total sessions.
- **Implications of increase/decrease:** High ratio implies strong content engagement; low ratio may indicate poor network performance or lack of user activity.
- **Impact on user experience:** Indicates sustained usage patterns enabling deeper content consumption.
- **Affected services:** Streaming browsing large file transfers.

3.2.14 disco_avg_rsrp – Average RSRP

- **Definition:** Reference Signal Received Power (RSRP) indicates the strength of the LTE reference signal received by the user device.

C1 Public

- **Measurement:** Measured at the UE from the strongest serving cell.
- **Implications of increase/decrease:** Low RSRP suggests poor coverage; high RSRP reflects strong signal presence.
- **Impact on user experience:** Weak RSRP leads to call drops slow data speeds and increased battery drain.
- **Affected services:** All mobile data and voice services.

3.2.15 disco_avg_rsrq – Average RSRQ

- **Definition:** RSRQ (Reference Signal Received Quality) indicates the quality of the LTE reference signal.
- **Measurement:**RSRQ = (N × RSRP) / RSSI; where N is the number of RBs RSRP is signal power and RSSI is total received signal strength.
- **Implications of increase/decrease:** Decreased RSRQ points to interference and congestion; increase suggests clear signal conditions.
- **Impact on user experience:** Poor RSRQ degrades throughput and increases packet errors.
- **Affected services:** All LTE-based services especially VoLTE and streaming.

RSRP

RSRP	Signal strength	Description
>= -80 dBm	Excellent	Strong signal with maximum data speeds
-80 dBm to -90 dBm	Good	Strong signal with good data speeds
-90 dBm to -100 dBm	Fair to poor	Reliable data speeds may be attained, but marginal data with drop-outs is possible. When this value gets close to -100, performance will drop drastically
<= -100 dBm	No signal	Disnonnection

RSRQ

RSRQ	Signal quality	Description
>= -10 dB	Excellent	Strong signal with maximum data speeds
-10 dB to -15 dB	Good	Strong signal with good data speeds
-15 dB to -20 dB	Fair to poor	Reliable data speeds may be attained, but marginal data with drop-outs is possible. When this value gets close to -20, performance will drop drastically
<= -20 dB	No signal	Disconnection

Figure 2: RSRP & RSRQ Signal Quality Categorization

3.2.16 disco_avg_uetxpower – Average UE Transmit Power

- **Definition:** Measures the average power used by the user equipment to transmit data.
- **Measurement:** Logged during active transmissions and averaged over time.
- **Implications of increase/decrease:** High values suggest poor uplink conditions requiring more power. User Equipment (UE) devices are categorized into power classes based on their maximum output power. Among these, **Power Class 3 (PC3)** is the default and most commonly used class, with a maximum transmit power of **23 dBm**. Higher power classes (e.g., PC1 or PC1.5) allow stronger transmission and are referred to as high-power classes.Power classes are also relevant for **carrier aggregation (CA)** and **EN-DC** configurations, where they define the maximum total power during multi-carrier transmission.During initial network access, the UE reports its power class along with other capabilities. If not reported, **PC3 is assumed by default**
- **Impact on user experience:** Higher transmit power leads to faster battery depletion and potential thermal throttling.
- **Affected services:** Uplink-heavy services such as video conferencing and live uploads.

C1 Public

3.2.17 disco_avg_uldmrssinr – Average Uplink DMRS SINR

- **Definition:** Indicates the Signal-to-Interference-plus-Noise Ratio (SINR) of Demodulation Reference Signals (DMRS) in the uplink direction.
- **Measurement:** Computed at the gNB/eNodeB based on DMRS during uplink transmissions.
- **Implications of increase/decrease:** Low SINR reflects uplink interference or poor coverage.
- **Impact on user experience:** Affects uplink throughput and call quality.
- **Affected services:** VoIP live streaming file uploads.

		RSRP (dBm)	RSRQ (dB)	SINR (dB)
RF Conditions	Excellent	>=-80	>=-10	>=20
	Good	-80 to -90	-10 to -15	13 to 20
	Mid Cell	-90 to -100	-15 to -20	0 to 13
	Cell Edge	<=-100	<-20	<=0

Figure 3: RF Conditions Categorizations

3.2.18 disco_avg_dlmcs – Average DL Modulation and Coding Scheme Index

- **Definition:** Reflects the average modulation and coding scheme (MCS) used for downlink data transmissions.
- **Measurement:** Averaged from real-time modulation choices (e.g. QPSK 16QAM 64QAM).
- **Implications of increase/decrease:** High MCS values reflect good radio conditions; low values indicate coverage or interference issues.
- **Impact on user experience:** Low MCS degrades DL throughput and causes buffering.
- **Affected services:** Video streaming downloads browsing.

3.2.19 disco_good_signal_rate – Good Signal Rate

- **Definition:** Proportion of time the user is in good signal conditions (RSRP and RSRQ above thresholds).
- **Measurement:**Time in good signal / Total session time.
- **Implications of increase/decrease:** High rate means stable connectivity; a drop may signal coverage holes or handover issues.
- **Impact on user experience:** Good signal rate correlates with uninterrupted service.
- **Affected services:** All mobile services.

C1 Public

3.2.20 main_congested – Cell Congestion Indicator

- **Definition:** Binary or ratio-based indicator showing whether a cell is congested.
- **Measurement:** Based on PRB usage scheduling delay or queue length at the MAC layer.
- **Implications of increase/decrease:** Persistent congestion indicates resource shortage or load imbalance.
- **Impact on user experience:** Users may experience slow speeds call blocking or latency.
- **Affected services:** All services in the congested area.

3.2.21 main_rate_4g_volume – 4G Data Volume Ratio

- **Definition:** Indicates the proportion of total data traffic served through the 4G network.
- **Measurement:** Calculated by dividing the 4G data volume by the overall mobile data volume (including 2G 3G 4G).
- **Implications of increase/decrease:** A higher ratio reflects effective 4G coverage and efficient user steering while a low ratio may suggest poor 4G coverage or frequent fallback to legacy technologies.
- **Impact on user experience:** Higher values ensure faster speeds and better service consistency. Lower values may lead to lower throughput and degraded experience.
- **Affected services:** All data-driven services (e.g. streaming VoIP web browsing).

3.2.22 main_num_call_drop_total – Total Call Drops

- **Definition:** Total number of calls dropped before normal call termination.
- **Measurement:** Count of both mobile-originated and mobile-terminated call drop events.
- **Implications of increase/decrease:** High call drop counts suggest issues with coverage mobility handover or capacity.
- **Impact on user experience:** Users experience frustration particularly during important or business calls.
- **Affected services:** VoLTE CS Voice VoWiFi.

3.2.23 main_dropped_4 – LTE Call Drops

- **Definition:** Number of dropped calls that occurred while the user was on the LTE network.
- **Measurement:** Tracked via core network and RAN event logs.
- **Implications of increase/decrease:** Increase may indicate LTE radio link failures handover problems or signaling errors.
- **Impact on user experience:** Dropped VoLTE calls disrupt conversation and lower confidence in service reliability.
- **Affected services:** VoLTE.

3.2.24 main_dropped_3 – 3G Call Drops

- **Definition:** Number of dropped calls that occurred on 3G (UMTS) radio access technology.
- **Measurement:** Monitored through MSC and RNC logs.

C1 Public

- **Implications of increase/decrease:** High values often relate to inadequate cell planning interference or outdated infrastructure.
- **Impact on user experience:** Disruption in voice communication during travel or inside buildings.
- **Affected services:** Circuit-switched voice over 3G.

3.2.25 cem_avg_30d_cem_e2e_delay_overall – 30-Day Avg. E2E Delay

- **Definition:** Average end-to-end latency measured over a rolling 30-day window.
- **Measurement:** Aggregated delay over user sessions in milliseconds.
- **Implications of increase/decrease:** Sustained high delay values imply chronic performance degradation likely affecting overall customer satisfaction.
- **Impact on user experience:** Longer app load times voice lags and reduced interactivity.
- **Affected services:** VoIP video conferencing cloud apps.

3.2.26 cat_e2e_delay_overall_encoded – Categorical Encoded E2E Delay

- **Definition:** A categorized version of E2E delay grouped into buckets (e.g. Low Medium High) for analytical use.
- **Measurement:** Translates numerical delay values into encoded labels for easier pattern detection.
- **Implications of increase/decrease:** Trends across categories may support anomaly detection or churn prediction.
- **Impact on user experience:** Used indirectly to identify performance risks.
- **Affected services:** All latency-sensitive services.

3.2.27 cem_mo_call_setup_succ_rate – MO Call Setup Success Rate

- **Definition:** Percentage of successfully established mobile-originated voice calls.
- **Measurement:** MO setup attempts vs. successful completions.
- **Implications of increase/decrease:** Decline may indicate signaling problems or capacity limitations.
- **Impact on user experience:** Failure to initiate calls leads to user frustration.
- **Affected services:** Voice calling (VoLTE CSFB).

3.2.28 cem_mt_call_setup_succ_rate – MT Call Setup Success Rate

- **Definition:** Percentage of mobile-terminated calls successfully delivered to the user.
- **Measurement:** MT call delivery attempts vs. successful connections.
- **Implications of increase/decrease:** Lower values imply issues in paging mobility or network reachability.
- **Impact on user experience:** Missed incoming calls or false call rejections.
- **Affected services:**Voice calling.

3.2.29 cem_mo_call_drop_rate – MO Call Drop Rate

- **Definition:** Proportion of mobile-originated calls that drop unexpectedly.
- **Measurement:** Dropped MO calls divided by successfully initiated MO calls.

C1 Public

- **Implications of increase/decrease:** High drop rate is a key negative indicator of network reliability.
- **Impact on user experience:** Users may avoid redialing or switch providers.
- **Affected services:** VoLTE CS Voice.

3.2.30 cem_mt_call_drop_rate – MT Call Drop Rate

- **Definition:** Ratio of dropped mobile-terminated calls.
- **Measurement:** Dropped MT calls divided by all connected MT calls.
- **Implications of increase/decrease:** High rates suggest weaknesses in coverage or handover especially at cell edges.
- **Impact on user experience:** User dissatisfaction and negative word-of-mouth.
- **Affected services:** Incoming voice calls (all types).

3.2.31 cem_ul_silent_call_ratio – Uplink Silent Call Ratio

- **Definition:** Measures the proportion of calls where no uplink voice data is transmitted.
- **Measurement:** Number of silent uplink call segments divided by total call segments.
- **Implications of increase/decrease:** High ratio indicates audio path or microphone issues.
- **Impact on user experience:** The receiver cannot hear the user leading to call dissatisfaction.
- **Affected services:** VoLTE CS Voice VoWiFi.

3.2.32 cem_dl_silent_call_ratio – Downlink Silent Call Ratio

- **Definition:** Represents the portion of calls during which no downlink audio is received by the user.
- **Measurement:** Number of silent downlink call segments divided by total call segments.
- **Implications of increase/decrease:** Increase implies network routing errors codec failures or downlink path issues.
- **Impact on user experience:** The user cannot hear the other party leading to confusion and frustration.
- **Affected services:** VoLTE CS Voice VoWiFi.

3.2.33 cem_less_1mb_ratio – Data Sessions <1MB Ratio

- **Definition:** Percentage of sessions with total data usage under 1MB.
- **Measurement:** Session count <1MB divided by total session count.
- **Implications of increase/decrease:** High values suggest very short or failed sessions.
- **Impact on user experience:** May point to frequent disconnections or user disengagement.
- **Affected services:** Web browsing app refreshes low-volume applications.

3.2.34 cem_1mb_3mb_ratio – Data Sessions between 1MB–3MB

- **Definition:** Proportion of sessions with total data usage between 1MB and 3MB.
- **Measurement:** Sessions within this volume range divided by total session count.

C1 Public

- **Implications of increase/decrease:** Reflects moderate engagement or partial session interruptions.
- **Impact on user experience:** Balanced behavior; spikes could indicate constrained usage.
- **Affected services:** App updates web sessions background data.

3.2.35 cem_3mb_5mb_ratio – Data Sessions between 3MB–5MB

- **Definition:** Represents sessions where data consumption falls between 3MB and 5MB.
- **Measurement:** Session count in this range over total sessions.
- **Implications of increase/decrease:** Higher ratios may indicate start of video streaming or short downloads.
- **Impact on user experience:** Indicates moderate data transfer with reasonable user experience.
- **Affected services:** Short video plays small file downloads.

3.2.36 cem_gt_8mb_ratio – Data Sessions >8MB

- **Definition:** Percentage of sessions with more than 8MB of data transferred.
- **Measurement:** Count of sessions >8MB divided by total sessions.
- **Implications of increase/decrease:** Reflects long or rich-media sessions; low values may indicate app abandonment.
- **Impact on user experience:** High values imply successful high-engagement activities.
- **Affected services:** HD video large file transfers social media usage.

3.2.37 disco_good_signal_rate – Good Signal Rate

- **Definition:** Measures the percentage of time users are under good signal conditions.
- **Measurement:** Time in good signal divided by total active time.
- **Implications of increase/decrease:** Drop indicates coverage gaps or mobility problems.
- **Impact on user experience:** Lower values degrade service stability and performance.
- **Affected services:** All mobile services.

3.2.38 disco_fair_signal_rate – Fair Signal Rate

- **Definition:** Proportion of time users are under fair (acceptable but not ideal) signal conditions.
- **Measurement:** Time under fair signal levels divided by total usage time.
- **Implications of increase/decrease:** A high fair signal rate may imply border cell coverage.
- **Impact on user experience:** May affect throughput but not severely.
- **Affected services:** All services under moderate signal strength.

3.2.39 disco_bad_signal_rate – Bad Signal Rate

- **Definition:** Proportion of time users experience poor signal levels.
- **Measurement:** Time under bad signal divided by total usage time.
- **Implications of increase/decrease:** High ratio strongly correlates with poor user experience.

C1 Public

- **Impact on user experience:** Slow browsing dropped calls poor streaming.
- **Affected services:** All services under weak coverage.

3.2.40 disco_bad_rsrq_rate – Bad RSRQ Rate

- **Definition:** Measures the time ratio during which RSRQ values fall below the quality threshold.
- **Measurement:** Time with bad RSRQ / Total time.
- **Implications of increase/decrease:** High values indicate cell-edge interference or poor resource scheduling.
- **Impact on user experience:** Lower data speeds and voice

3.2.41 disco_bad_ulsnr_rate – Bad Uplink SINR Rate

- **Definition:** Percentage of time uplink SINR values fall below acceptable thresholds.
- **Measurement:** Time with uplink SINR below target divided by total time.
- **Implications of increase/decrease:** A high rate suggests poor uplink conditions likely due to interference or weak coverage.
- **Impact on user experience:** Results in poor voice quality and slow upload speeds.
- **Affected services:** VoIP live streaming real-time communication.

3.2.42 disco_bad_dlmcs_rate – Bad Downlink MCS Rate

- **Definition:** Proportion of time the network uses low MCS values for downlink transmission due to poor radio conditions.
- **Measurement:** Time with low MCS values divided by total active transmission time.
- **Implications of increase/decrease:** A higher rate implies consistently bad signal quality or high interference.
- **Impact on user experience:** Reduces download speeds and overall service performance.
- **Affected services:** Video streaming web browsing large file downloads.

The information presented in this section has been compiled based on the findings reported in [16]–[19].

3.3 An Exploratory Data Analysis (EDA)

3.3.1 Data Integration and Preprocessing

According to the first quarter 2025 reports of the Information and Communication Technologies Authority (BTK) Vodafone Turkey has approximately **25.2 million** mobile subscribers. To analyze behavioral and network performance metrics across such a large subscriber base standard SQL environments are not sufficient. Therefore a distributed big data environment based on Hadoop was used. All data preprocessing and integration tasks were performed on a Cloudera platform using Impala which supports scalable high-performance SQL-like queries over large datasets. This infrastructure made it possible to merge multiple sources such as customer data network KPIs and churn labels efficiently forming the basis for the exploratory data analysis phase.

C1 Public

The data for December 2024 was retrieved from the monthly-aggregated table using the query provided below. Although the dataset is constructed on a monthly basis each record represents the average of the previous 30 days from the reference date. This means that regardless of which day is selected within the month the data reflects a rolling 30-day average. If Net Promoter Score (NPS) data was available for a customer the most recent NPS record within the month was also joined to the dataset. In addition two churn-related variables were added: churn_pmonth indicating the month of churn and churn_flg which is set to 1 if the customer churned in the current or following two months (i.e. within a three-month window) and 0 otherwise.

C1 Public

```
create table schema.bno_DD_all_turkey_with_churn_20241228  stored as parquet
as

select x.* y.churn_pdate y.prev_churn_pdate  y.churn_flg from

(select distinct * from

(select distinct * from
(select *  cast((substring(cast(pdate as string) 1 6)) as int) as pmonth from
datalab_nti.user_exp_last30days)t
where pdate = 20241229 ) as a

inner join

(select distinct * from
(select gsm_nom city district_son as district business_segment avg_invoice
CAST(pmonth as INT) as pmonth_loc
from loc.table )m
where pmonth_loc = 202412) as b


on a.gsm_no=b.gsm_nom and a.pmonth=b.pmonth_loc) as x


left join


(select distinct gsm_no churn_pdate
 CASE WHEN CAST(churn_pdate AS INT) % 100 = 1 THEN (FLOOR(CAST(churn_pdate AS
INT) / 100 )- 1) * 100 + 12 ELSE CAST(churn_pdate AS INT) - 1 END AS
prev_churn_pdate
 churn_flg from
(select num_port_msisdn as gsm_no  cast((concat(
split_part(num_port_request_completed_dt "-" 1)
split_part(num_port_request_completed_dt "-" 2))) as int) as churn_pdate  1 as
churn_flg
from churner.table
where mnp_direction_cd in ( VODAFONE-TURKCELL   VODAFONE-AVEA ))m
where churn_pdate between 202412 and 202501 ) as y

on x.gsm_no=y.gsm_no and x.pmonth_loc=y.prev_churn_pdate
```

Code Section 1: Impala Code for Table Creating

C1 Public

When the dataset was constructed all 42 total features included in the analysis were as follows:

- gsm_no
- nps_pmonth
- category_nw
- category_overall
- cem_tput_kbps
- cem_xkb_delay
- cem_e2e_delay
- cem_dl_volume_gb
- cem_ul_volume_gb
- cem_xkb_over_threshold_ratio
- cem_e2e_over_threshold_ratio
- cem_tput_under_threshold_ratio
- cem_cei
- cem_stream_qoe
- cem_volte_qoe
- cem_voice_qoe
- cem_cei_detractor_days_ratio
- cem_voice_detractor_days_ratio
- cem_volte_detractor_days_ratio
- cem_stream_detractor_days_ratio
- cem_ul_mos
- cem_dl_mos
- cem_mo_call_setup_dur
- cem_mo_call_setup_succ_rate
- cem_mt_call_setup_succ_rate
- cem_mo_call_drop_rate
- cem_mt_call_drop_rate
- cem_ul_silent_call_ratio
- cem_dl_silent_call_ratio
- cem_less_1mb_ratio
- cem_1mb_3mb_ratio
- cem_3mb_5mb_ratio

C1 Public

- cem_gt_8mb_ratio
- disco_avg_rsrp
- disco_avg_rsrq
- disco_avg_uetxpower
- disco_avg_uldmrssinr
- disco_avg_dlmcs
- disco_good_signal_rate
- disco_fair_signal_rate
- disco_bad_signal_rate
- disco_bad_rsrq_rate
- disco_bad_ulsnr_rate
- disco_bad_dlmcs_rate
- main_congested
- main_rate_4g_volume
- main_num_call_drop_total
- main_dropped_4
- main_dropped_3
- cem_avg_30d_cem_e2e_delay_overall
- churn_flg
- churn_pdate
- city
- district

3.3.2 Data Cleaning

Before model development the dataset was cleaned to ensure quality and reliability. This included handling missing values, removing outliers, correcting data types and filtering invalid records. Additionally duplicate rows and inconsistent labels were addressed to maintain dataset consistency. The specific cleaning steps applied are described below.

3.3.3 Handling Missing Values

To identify columns with missing values a Spark-based null value analysis was performed. In network monitoring systems it is common to encounter partial data loss due to the nature of data collection and aggregation processes. Since the data used in this study includes sensitive internal information it cannot be publicly shared or deeply examined on a full-customer level. For that reason imputing missing values was intentionally avoided. Instead

C1 Public

rows containing any missing values were directly removed from the dataset to ensure data integrity and simplify downstream analysis. As a representative sample will be selected in the next phase this early elimination of incomplete records guarantees that the sampled dataset will be clean and reliable for both clustering and churn prediction tasks.

```
data = spark.sql("select * from
schema.bno_DD_all_turkey_with_churn_20241228 ")

from pyspark.sql.functions import col isnan when count
df_test = data.select([count(when(col(c).contains( None ) | \
                            col(c).contains( NULL ) | \
                            (col(c) ==    ) | \
                            col(c).isNull() | \
                            isnan(c)  c
                           )).alias(c)
                    for c in data.columns])



df_test.show(n=20  truncate=True  vertical=True)
```

Code Section 2: Spark Code for Missing Value Detection

3.3.4 Outlier Detection and Treatment

Outlier detection was performed using the Interquartile Range (IQR) method across selected numerical KPI features. The 25th (Q1) and 75th (Q3) percentiles were calculated using Spark's approxQuantile() function, and the IQR was defined as Q3 - Q1. For each column, lower and upper bounds were computed using Q1 - k*IQR and Q3 + k*IQR, where k is a user-defined multiplier. Rows were retained only if all KPI values fell within these bounds.

Before settling on the 25th and 75th percentiles, alternative quantile ranges such as the 1st–99th and 10th–90th percentiles were also tested. However, based on domain expertise and after inspecting the results using .describe(), it was observed that extreme values still remained within those thresholds. Therefore, the 25th–75th percentile range was selected as the most appropriate for identifying and removing outliers. A relatively high multiplier value of 10 was used to avoid discarding valid but naturally skewed high-performance values, especially for download-related KPIs.

After filtering, a descriptive statistical summary of the cleaned dataset was generated using Spark's .describe() method and converted into a Pandas DataFrame for easier inspection.

C1 Public

This final check confirmed that the outlier removal process was effective and aligned with domain expectations.

As a result of this filtering step, approximately 19 million samples were retained for further analysis shown as Table II.

```
 # Define columns for outlier detection

columns_for_outliers = [
    'cem_tput_kbps', 'cem_xkb_delay', 'cem_e2e_delay', 'cem_dl_volume_gb',
'cem_ul_volume_gb',
    'cem_cei', 'cem_stream_qoe', 'cem_volte_qoe',
'cem_voice_qoe','cem_ul_mos', 'cem_dl_mos',
    'cem_mo_call_setup_dur','disco_avg_rsrp','disco_avg_rsrq',
'disco_avg_uetxpower', 'disco_avg_uldmrssinr',
    'disco_avg_dlmcs', 'main_rate_4g_volume', 'main_num_call_drop_total'
]

def filter_outliers_iqr(df, columns, multiplier):
    conditions = []
    for col_name in columns:
        quantiles = df.approxQuantile(col_name, [0.25, 0.75], 0.0)
        q1, q3 = quantiles[0], quantiles[1]
        iqr = q3 - q1
        lower_bound = q1 - multiplier * iqr
        upper_bound = q3 + multiplier * iqr

        # Debugging: Print bounds and check condition formation
        print(f"Column: {col_name}, Q1: {q1}, Q3: {q3}, Lower Bound:
{lower_bound}, Upper Bound: {upper_bound}")

        condition = (col(col_name) >= lower_bound) & (col(col_name) <=
upper_bound)
        conditions.append(condition)

    cmbnd_condition = conditions.pop(0)
    for condition in conditions:
        cmbnd_condition &= condition


    dfa = df.filter(combined_condition)
    return dfa
```

C1 Public

Code Section 3: Spark Code for Outlier Detection

summary	cem_tput_kbps	cem_xkb_delay	cem_e2e_delay	cem_dl_volume_gb	cem_ul_volume_gb	cem_xkb_over_threshold_ratio	cem_e2e_over_threshold_ratio
count	19964975	19964975	19964975	19964975	19964975	19964975	19964975
mean	6390,87	1490,98	100,75	11,82	0,60	0,06	0,09
stddev	2355,77	439,69	76,98	13,02	0,65	0,05	0,07
min	4,54	4,00	4,33	0,00	0,00	0,00	0,00
max	27150,16	5090,71	652,57	104,53	5,09	1,00	1,00

summary	cem_mo_call_setup_dur	cem_mo_call_setup_succ_rate	cem_mt_call_setup_succ_rate	cem_mo_call_drop_rate	cem_mt_call_drop_rate	cem_ul_silent_call_ratio	cem_dl_silent_call_ratio
count	19964975	19964975	19964975	19964975	19964975	19964975	19964975
mean	3176,90	0,99	1,00	0,00	0,00	0,01	0,01
stddev	494,18	0,01	0,00	0,00	0,00	0,02	0,01
min	496,00	0,00	0,00	0,00	0,00	0,00	0,00
max	6352,14	1,00	1,00	3,00	9,00	1,00	1,00

summary	disco_avg_uldmrssinr	disco_avg_dlmcs	disco_good_signal_rate	disco_fair_signal_rate	disco_bad_signal_rate	disco_bad_rsrq_rate	disco_bad_ulsnr_rate
count	19964975	19964975	19964975	19964975	19964975	19964975	19964975
mean	14,76	11,14	0,68	0,29	0,09	0,09	0,00
stddev	2,64	1,87	0,25	0,20	0,16	0,15	0,01
min	3,02	3,62	0,00	0,00	0,00	0,00	0,00
max	26,96	18,74	1,00	1,00	1,00	1,00	0,68

summary	cem_tput_under_threshold_ratio	cem_cei	cem_stream_qoe	cem_volte_qoe	cem_voice_qoe	cem_ul_mos	cem_dl_mos
count	19964975	19964975	19964975	19964975	19964975	19964975	19964975
mean	0,16	79,87	83,60	75,95	62,45	3,90	4,01
stddev	0,13	6,94	5,95	7,10	12,76	0,20	0,08
min	0,00	3,00	34,13	28,14	4,00	2,53	3,78
max	1,00	100,00	94,00	94,00	93,00	4,47	4,22

summary	cem_less_1mb_ratio	cem_1mb_3mb_ratio	cem_3mb_5mb_ratio	cem_gt_8mb_ratio	disco_avg_rsrp	disco_avg_rsrq	disco_avg_uetxpower

C1 Public

count	19964975	19964975	19964975	19964975	19964975	19964975	19964975
mean	0,05	0,17	0,18	0,60	-99,13	-13,01	20,34
stddev	0,06	0,11	0,08	0,17	4,26	0,98	1,29
min	0,00	0,00	0,00	0,00	-117,94	-17,01	15,12
max	1,00	1,00	1,00	1,00	-80,13	-8,97	23,00

summary	**disco_bad_dlmcs_rate**	**main_congested**	**main_rate_4g_volume**	**main_num_call_drop_total**	**main_dropped_4**	**main_dropped_3**	**cem_avg_30d_cem_e2e_delay_overall**
count	19964975	19964975	19964975	19964975	19964975	19964975	19964975
mean	0,58	0,12	99,51	0,77	0,06	0,10	84,75
stddev	0,21	0,32	0,94	1,57	0,24	0,30	67,06
min	0,00	0,00	91,73	0,00	0,00	0,00	4,33
max	1,00	1,00	100,00	11,00	1,00	1,00	3463,00

Table II: Summary Statistics of the Filtered Dataset After Outlier Removal

3.3.5 Data Type Corrections

All numerical KPI columns were explicitly cast to float data type to ensure consistency across the dataset and prevent type-related issues during modeling and analysis.

3.3.6 Removing Duplicates

A filtering check was applied to ensure that each customer is represented by a single record in the dataset. Although the code was prepared to remove potential duplicates, no duplicated rows were detected during this process.

3.3.7 Data Sampling

To ensure a balanced distribution across key variables such as Net Promoter Score (NPS), churn status, and customer location, a stratified sampling approach was applied to the cleaned dataset. Starting from approximately 19 million records, a representative sample of around 4 million records was selected using a sampling ratio of 0.15 (15%). Strata were defined based on a combination of four categorical attributes: network category, churn flag, city, and district. These variables were concatenated to form a composite stratum key for each row. This allowed the sample to preserve both behavioral diversity (e.g., churned vs. retained users) and geographic diversity across different cities and districts. The fixed sampling ratio of 15% was applied uniformly across all strata using Spark's sampleBy() method to maintain proportional representation while significantly reducing dataset size for improved computational efficiency.This stratified approach ensures that the resulting dataset reflects the real-world

C1 Public

distribution of customers and enables consistent training and evaluation of both clustering and churn prediction models.

```
# Creating composite stratum key

data = data.withColumn(  'stratum', concat_ws('_',
col('category_nw_str'), col('churn_flg_str'), col('data_city_str'),
col('data_district_str')))
strata = data.select('stratum').distinct().collect()

# Applying stratified sampling
sampling_ratio = 0.15
datas = data.sampleBy( "stratum", fractions={row['stratum']:
sampling_ratio for row in strata}, seed=42)
```

Code Section 4: Stratified Sampling Code with Spark

3.3.8 Exploratory Visualization of Sampled Data

To better understand the distribution of the sampled dataset, histograms were plotted for all categorical variables (e.g., network category, city, district, churn flag) to visualize class balance and detect any over- or under-represented groups. In parallel, boxplots were generated for all numerical KPI variables to examine their spread, identify potential outliers, and observe general trends across different customer segments.These visual analyses provided valuable insights of the dataset, enabling informed decisions in the subsequent preprocessing, feature engineering, and modeling stages. Based on the visualizations getting from the sampled dataset, the following observations can be made:

- **Categorical features** such as category_nw and category_overall exhibit highly imbalanced distributions, with a significant portion labeled as “unknown.” This suggests potential issues in data labeling or missing classification information.
- **Network performance indicators** like main_congested, main_dropped_3, and main_dropped_4 are mostly concentrated around zero, indicating that the majority of users do not experience critical issues. However, there is a small subset with severe problems, likely representing outliers.
- **Numerical KPIs** including cem_e2e_delay, cem_xkb_delay, cem_dl_volume_gb, and cem_tput_kbps show a wide range and heavy tails, with evident outliers on the upper end. This reflects significant variability in network performance among users.
- **Experience quality scores** such as cem_stream_qoe, cem_volte_qoe, and cem_voice_qoe are generally skewed towards high scores, but some users clearly report extremely poor experiences, as reflected in the long left-side tails of the boxplots.
- **Radio signal metrics** like disco_avg_rsrp, disco_avg_rsrq, and disco_avg_uldmrssinr display substantial variance and outliers, indicating that some geographic areas suffer from poor radio conditions.

C1 Public

- **Call quality and reliability indicators** including cem_ul_silent_call_ratio, cem_dl_silent_call_ratio, and cem_call_drop_rate also reveal outliers, suggesting a limited but important portion of users face considerable service degradation.
- **Overall experience index**, cem_cei, is mostly concentrated in higher ranges, yet some users fall well below the average, which may correlate with customer dissatisfaction or churn risk.

These visual insights highlight potential network performance bottlenecks and customer dissatisfaction patterns, providing a strong foundation for subsequent stages such as outlier treatment, feature engineering, clustering, and churn prediction.

C1 Public

```
a= ['category_nw',  'category_overall','main_congested',
'main_num_call_drop_total', 'main_dropped_4', 'main_dropped_3']

b = ['cem_tput_kbps', 'cem_xkb_delay' , 'cem_e2e_delay',
'cem_dl_volume_gb', 'cem_ul_volume_gb', 'cem_xkb_over_threshold_ratio']

c= ['cem_e2e_over_threshold_ratio','cem_tput_under_threshold_ratio',
'cem_stream_qoe','cem_volte_qoe', 'cem_voice_qoe',
'cem_cei_detractor_days_ratio']

d =['cem_volte_detractor_days_ratio','cem_stream_detractor_days_ratio',
'cem_ul_mos', 'cem_dl_mos','cem_mo_call_setup_dur',
'cem_mo_call_setup_succ_rate','cem_mt_call_setup_succ_rate',
'cem_mo_call_drop_rate','cem_mt_call_drop_rate',
'cem_ul_silent_call_ratio','cem_dl_silent_call_ratio',
'cem_less_1mb_ratio', 'cem_1mb_3mb_ratio','cem_3mb_5mb_ratio',
'cem_gt_8mb_ratio', 'disco_avg_rsrp', 'disco_avg_rsrq',
'disco_avg_uetxpower', 'disco_avg_uldmrssinr','disco_avg_dlmcs',
'disco_good_signal_rate', 'disco_fair_signal_rate','disco_bad_signal_rate',
'disco_bad_rsrq_rate', 'disco_bad_ulsnr_rate','disco_bad_dlmcs_rate',
'main_rate_4g_volume','main_num_call_drop_total','cem_avg_30d_cem_e2e_delay
_overall','cem_cei','cem_e2e_delay']

import seaborn as sns #data visualization
import matplotlib.pyplot as plt

plt.figure(figsize=(12,16))
for i, j in enumerate(a):
    #print(j)
    plt.subplot(5,2, i+1)
    sns.histplot(x=new_data[j])
    plt.xlabel(j)
    plt.title('{} Distribution'.format(j))
    plt.yscale("log")
    # plt.subplots_adjust(wspace=.2, hspace=.5)
    plt.tight_layout()
plt.show()
plt.figure(figsize=(12,9))
for i, j in enumerate(b):
    plt.subplot(6,1, i+1)
    sns.boxplot(x=new_data[j])
    plt.title('{} Boxplot'.format(j))
    plt.tight_layout()

plt.show()
```

Code Section 5: Visualization of Categorical and Numerical Variable Distributions via Histograms and Boxplots

C1 Public

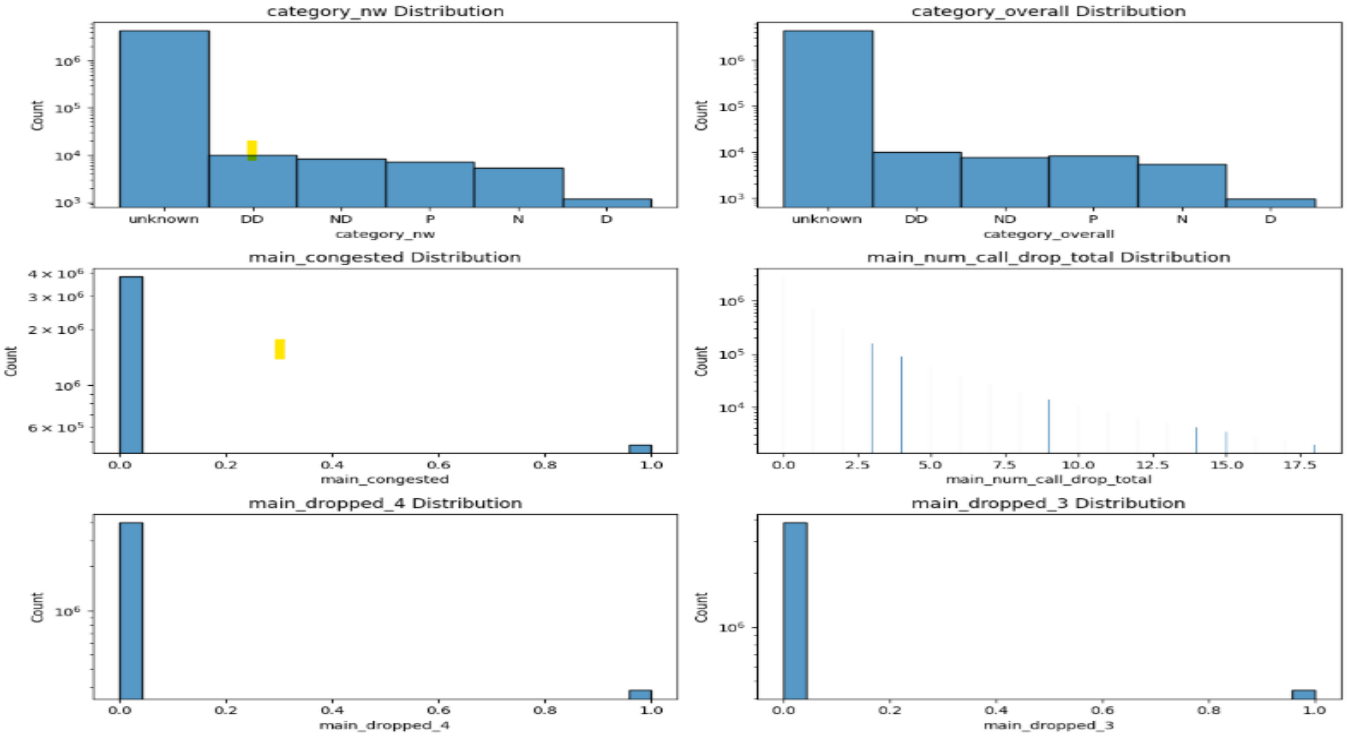


Figure 4 : Distribution of Categorical Variables

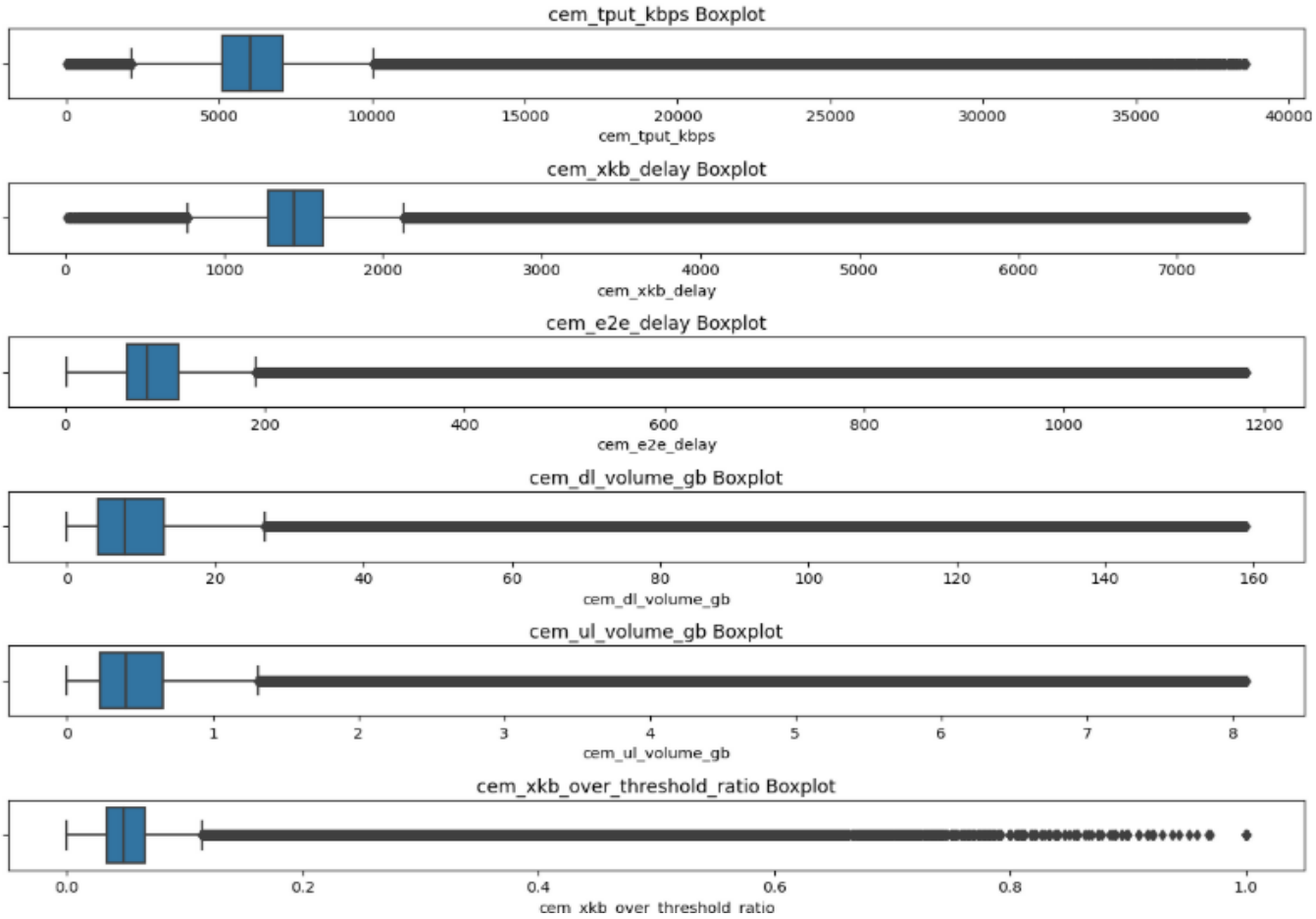


Figure 5: Distribution of Numerical Variables-1

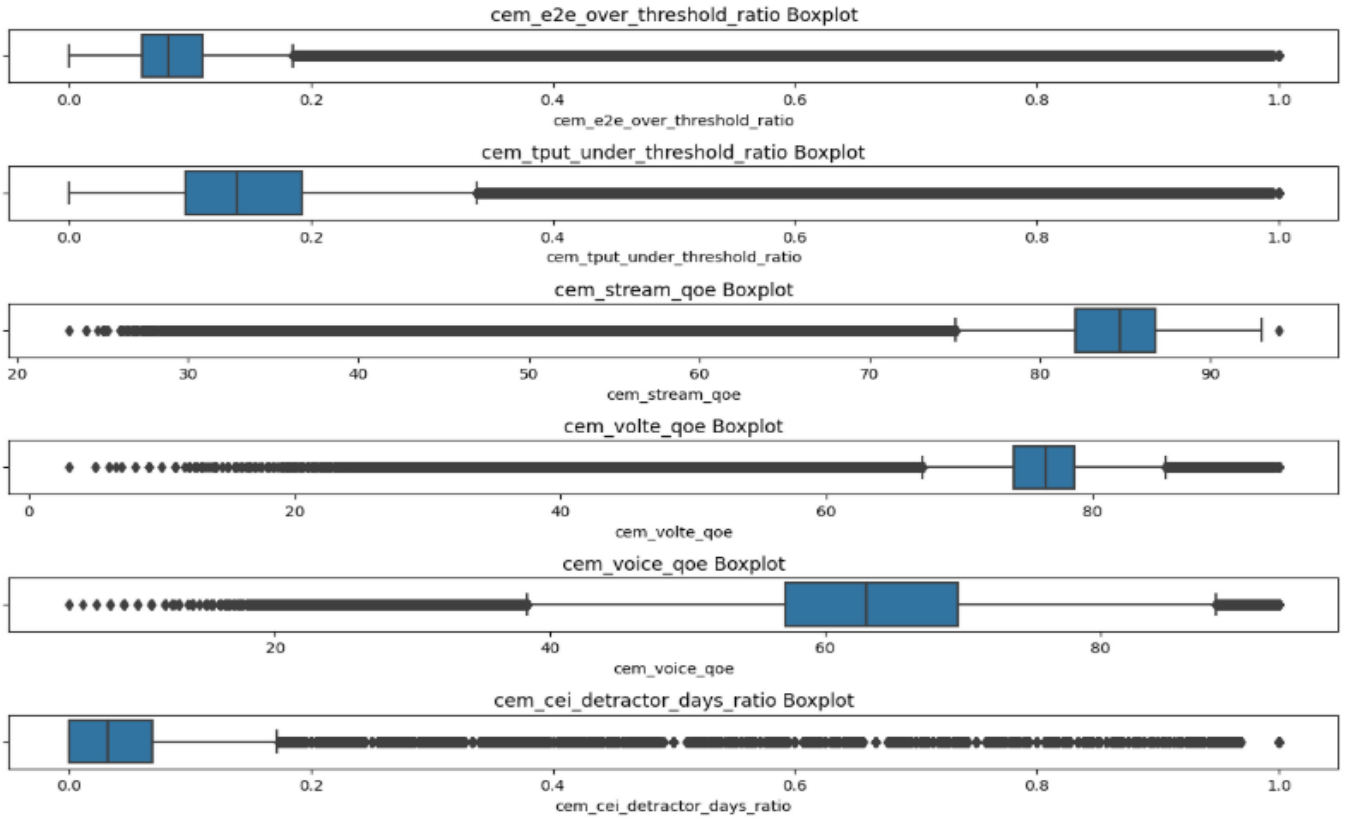


Figure 6: Distribution of Numerical Variables-2

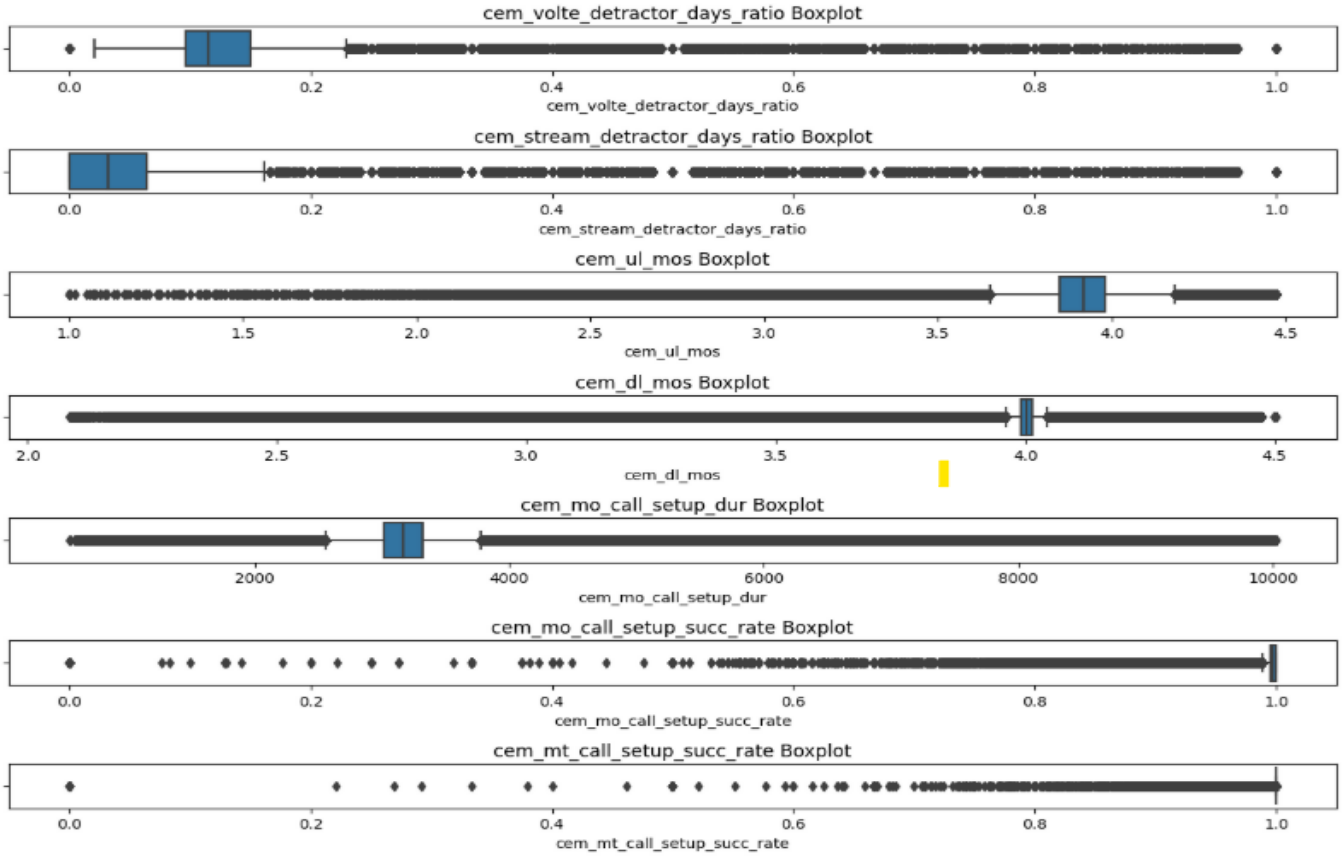


Figure 7: Distribution of Numerical Variables-3

C1 Public

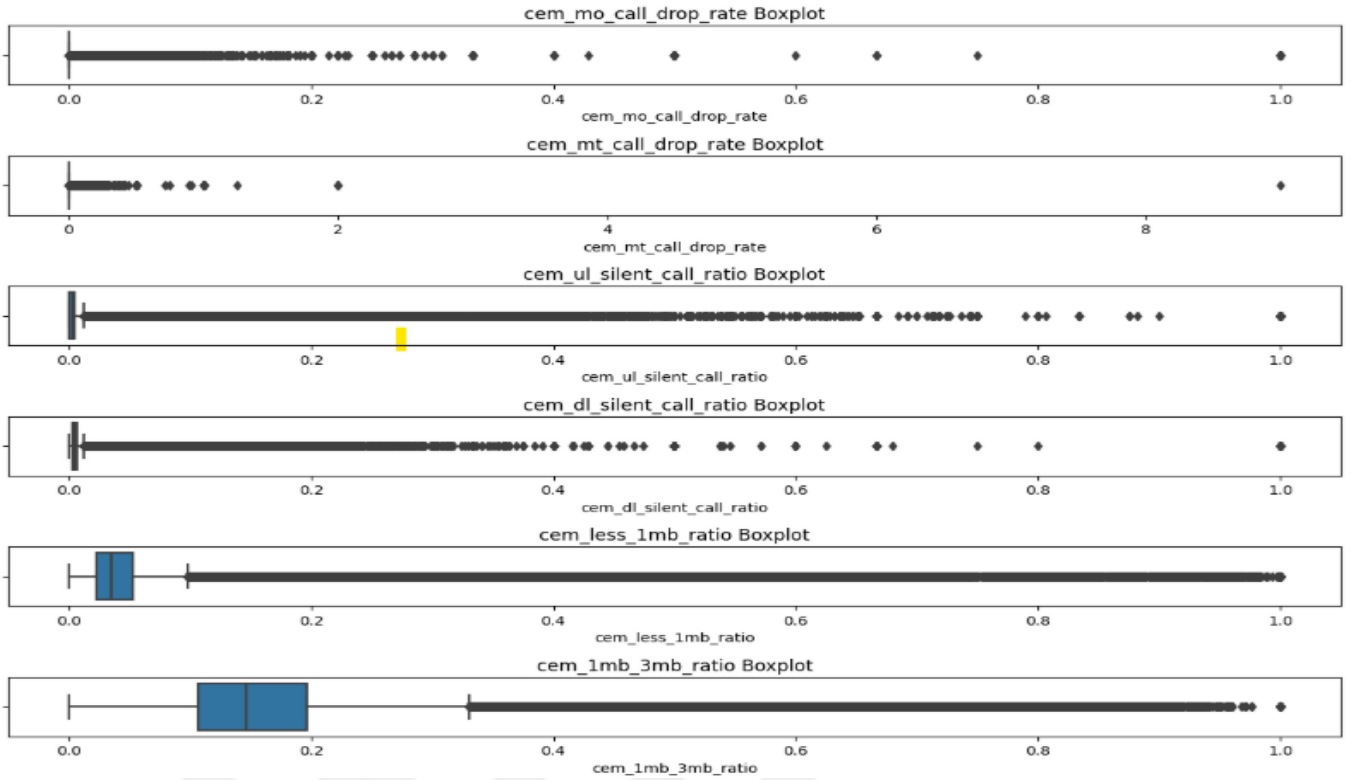


Figure 8: Distribution of Numerical Variables-4

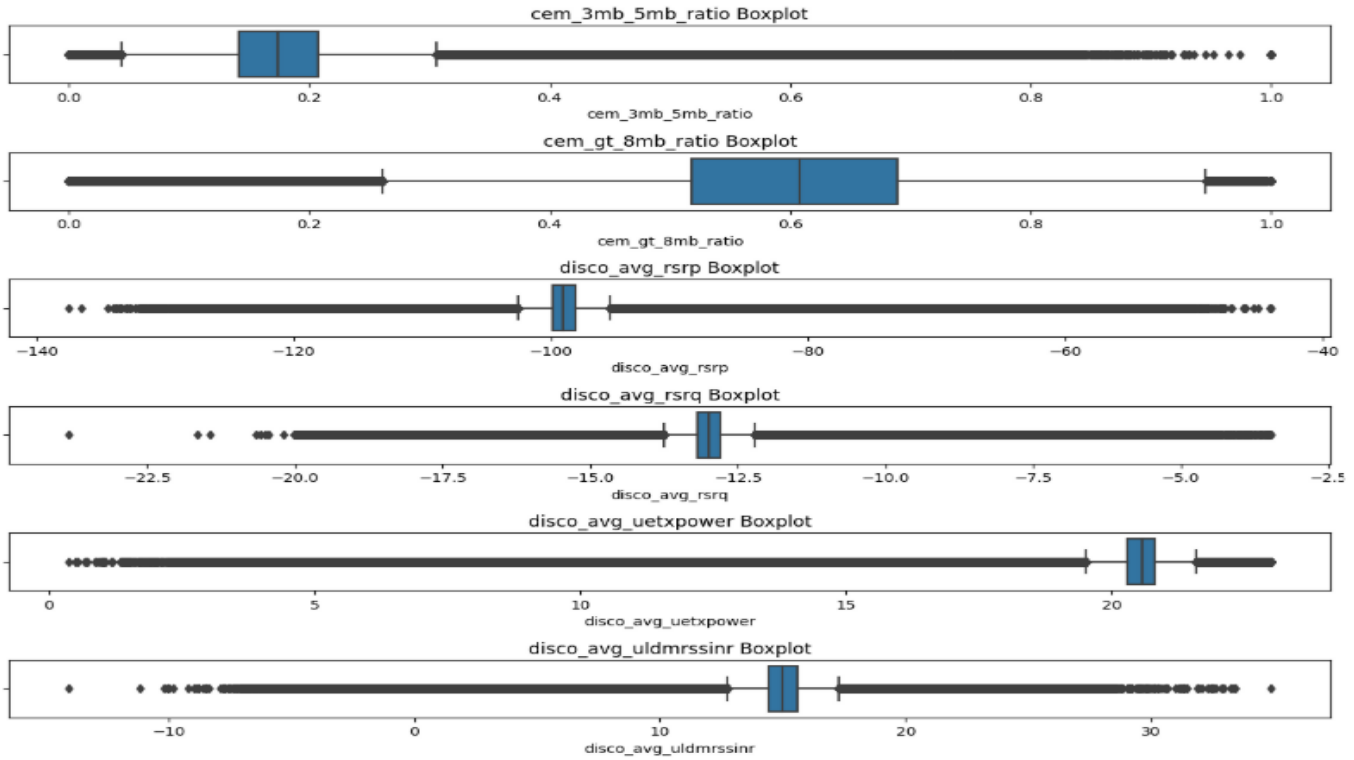


Figure 9: Distribution of Numerical Variables-5

C1 Public

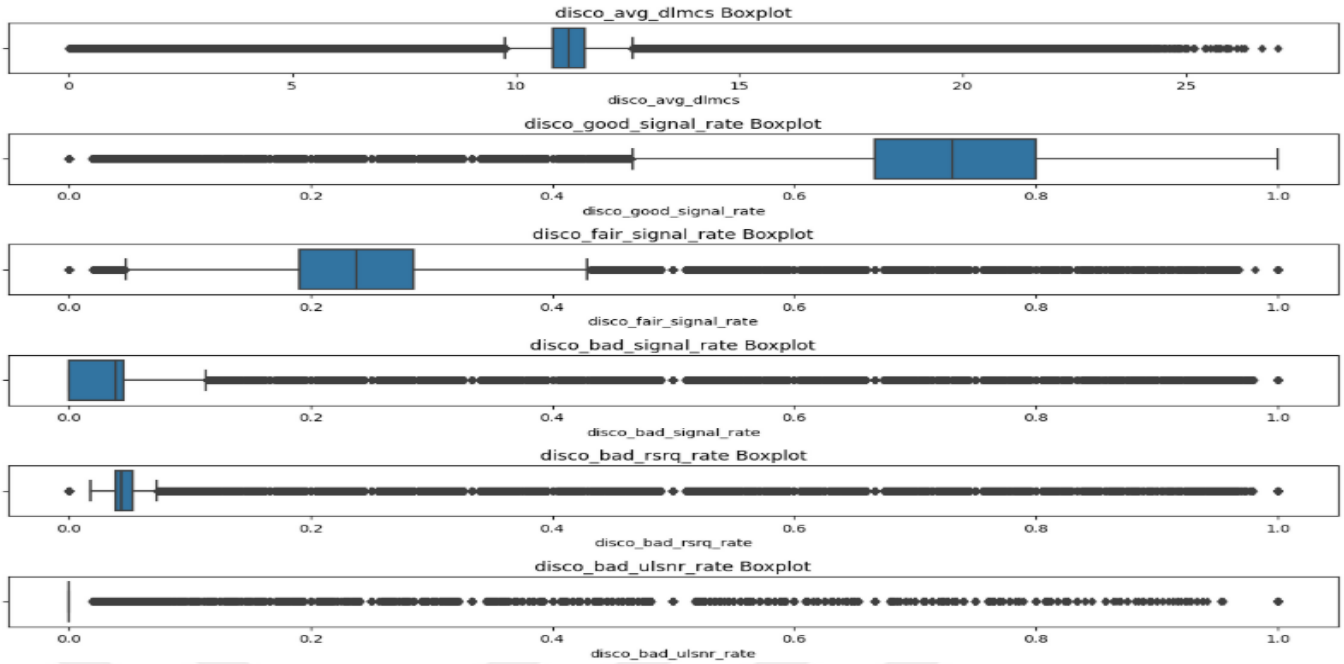


Figure 10: Distribution of Numerical Variables-6

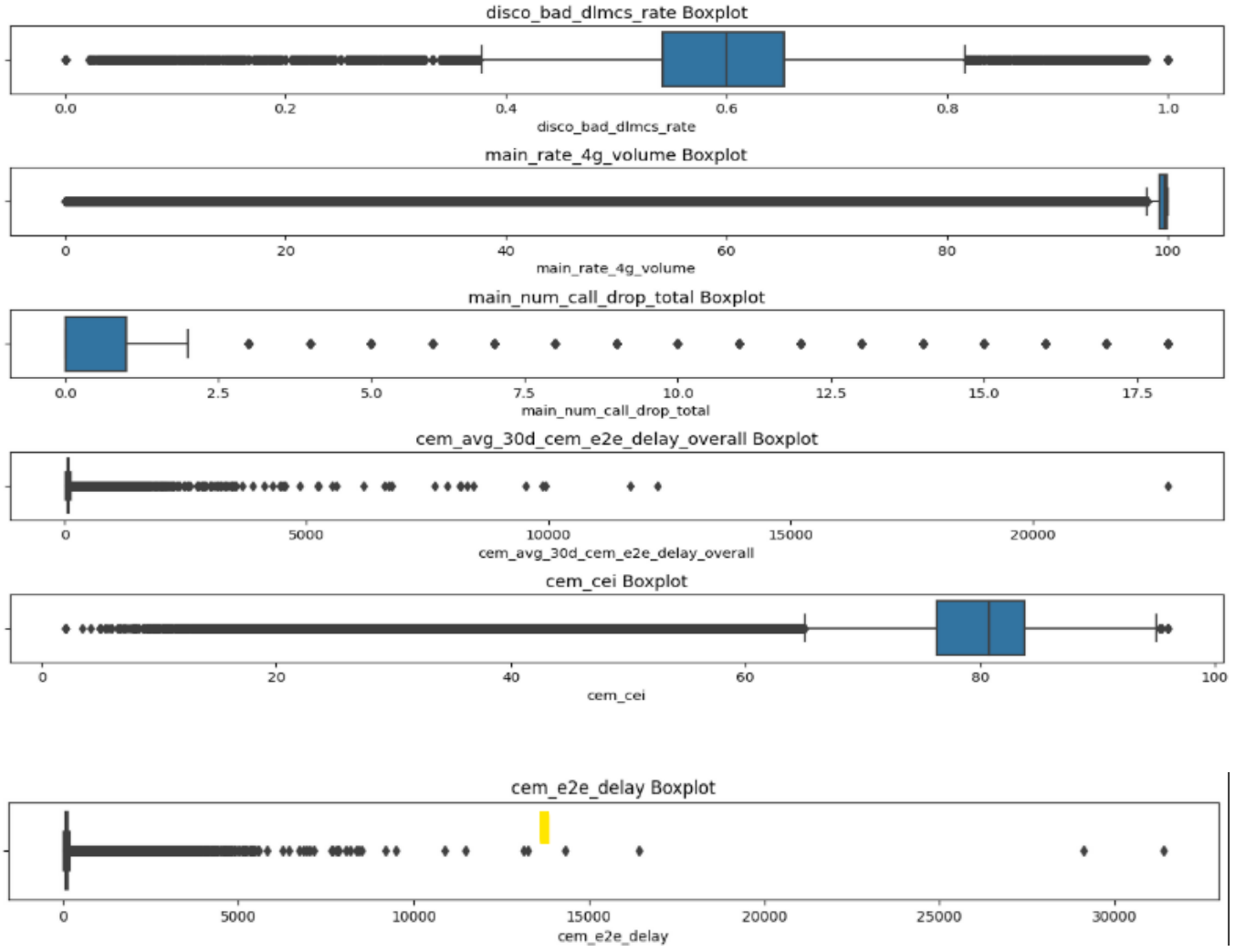


Figure 11:Distribution of E2E Delay

C1 Public

3.3.9 Ordinal Encoding and Feature Scaling for Model Readiness

In machine learning workflows, it is crucial to prepare data in a way that numerical algorithms can interpret and utilize efficiently. Two important preprocessing techniques used in this process are ordinal encoding and feature scaling.

- **Ordinal Encoding:** Some categorical features represent an inherent order (e.g., Poor < Neutral < Good). By converting these ordinal categories into numerical values while preserving their order, machine learning models can better capture the relationship between input features and the target variable. Without encoding, models cannot process string values.
- **Feature Scaling (Standardization):** When features have different ranges or units (e.g., throughput in kbps vs. MOS scores), models like logistic regression, k-means, or gradient descent-based algorithms may become biased toward features with larger scales. Standardizing features ensures that each variable contributes equally, improving convergence and accuracy.

Together, these preprocessing steps make the dataset model-ready, reducing training time and improving model performance.

```
from marmara_pyspark.ml.feature import marmara_VectorAssembler

def label_encoding(mapping):
    def label_encode(label):
        return mapping[label]
    return udf(label_encode, IntegerType())

ordinal_mappings ={
                   'category_nw': {'DD': 1,'D':2, 'ND': 3, 'N': 4, 'P' :5 ,
'unknown':6 },
                   'category_overall': {'DD': 1,'D':2, 'ND': 3, 'N': 4, 'P'
:5 , 'unknown':6 },
                   #'cat_e2e_delay_overall': {'gt_120': 1, 'btw_100_120': 2,
'btw_80_100': 3, 'btw_60_80': 4,  'less60' : 5 , 'unknown':6}
                  }  # Mapping for ordinal feature


# Apply label encoding using UDF
for feature, mapping in ordinal_mappings.items():
    mapping_udf = label_encoding(mapping)
    df4m= df4m.withColumn(feature + '_encoded', mapping_udf(col(feature)))

# Assemble features
feature_columns = [feature + '_encoded' for feature in
ordinal_mappings.keys()]  # Use the encoded columns
assembler = VectorAssembler(inputCols=feature_columns,
outputCol='features')
data_4m = assembler.transform(df4m)
```

Code Section 6: Label Encoding of Categorical Features

C1 Public

```
# Feature scaling for all columns except 'Customer_ID'
input_columns4 = [col for col in data_4mn.columns if (col != 'gsm_no') ]

#df_new1 = spark.createDataFrame(df_new1, schema=input_columns)

assembler_all4 = VectorAssembler(inputCols=input_columns4,
outputCol='all_features4')

scaled_df4 = assembler_all4.transform(data_4m)


scaler4 = StandardScaler(inputCol='all_features4', outputCol='features4',
withStd=True, withMean=True)

#Max_min_scaler = MinMaxScaler(inputCol='all_features',
outputCol='features')  --max min
scaler_model4 = scaler4.fit(scaled_df4)

scaler_model2 = scaler2.fit(scaled_df2)
```

Code Section 7: Feature Scaling

3.4 Datasets

Based on the insights obtained from the exploratory data analysis two distinct datasets were constructed to support different modeling approaches.

3.4.1 Dataset 1: 30-day average KPIs per customer

The primary dataset used in this study was generated by the Huawei SmartCare System a comprehensive network performance and monitoring platform that aggregates customers daily service experience through various KPIs (Key Performance Indicators). A detailed explanation of all 42 KPIs used in the analysis—covering their technical definitions measurement methodologies customer impact and service relevance—can be found in the supplementary document titled Appendix A-KPI_Explanation.docx submitted alongside this paper. For instance the End-to-End Delay (cem_e2e_delay) KPI represents the total round-trip latency experienced by the user encompassing both core and radio network segments. Increased delay values may indicate core network congestion or transmission inefficiencies directly affecting the quality of latency-sensitive services such as VoIP calls (e.g. WhatsApp Skype) real-time video conferencing, online gaming (e.g. PUBG Mobile Call of Duty) live streaming and remote desktop applications.

C1 Public

Alongside network KPIs additional customer-level attributes were derived to enrich the dataset. For each customer the top cell—where the user spent the majority of their data and voice usage at related Base Station—was used to infer the corresponding city and district as the customer s location. Furthermore Net Promoter Score (NPS) labels were incorporated as a monthly indicator of customer satisfaction and loyalty. Instead of the traditional 0–10 numeric scale NPS data in this study was categorized into four distinct classes:

- **DD (Deep Detractor):** strongly dissatisfied users who are highly likely to criticize or abandon the service.
- **D (Detractor):** dissatisfied users who are unlikely to recommend the service.
- **ND (Near Detractor):** users with mildly negative sentiment positioned between detractors and neutral users showing early signs of dissatisfaction.
- **N (Neutral):** users with indifferent or balanced sentiment neither likely to promote nor criticize the service.
- **P (Promoter):** highly satisfied users who are likely to recommend and promote the service to others.

Since both churn and NPS (Net Promoter Score) data were provided on a monthly basis , a monthly customer-level dataset was created by calculating the average values of each KPI over the last 30 days. During data preparation missing values were not imputed; instead records with incomplete data were excluded. This decision was based on two key factors: first due to privacy concerns real customer-level data could not be shared so a sampled dataset was used; second telecom data is inherently large-scale and full datasets with millions of rows are impractical to process. Since sampling was already necessary removing records with missing values was both efficient and representative. Moreover imputing such values in a heterogeneous network environment might introduce additional bias potentially compromising model reliability. To further ensure data quality an outlier detection process was applied using the 0.25 and 0.75 interquartile range (IQR) thresholds on continuous KPIs. These preprocessing steps helped ensure that the dataset used in modeling was both clean and analytically reliable. To ensure that the numerical KPI features were on a comparable scale and to prevent features with larger numeric ranges from dominating the learning process all KPI values were standardized using a standard scaling technique—transforming each feature to have zero mean and unit variance. This normalization step was particularly important given the wide variability among KPIs which capture diverse aspects of network performance and user experience. Following the scaling process a stratified sampling strategy was applied to create a representative and computationally manageable dataset. Key categorical columns including nps_pmonth category_nw churn_flg churn_pdate data_city and data_district were first cast into string format to enable consistent grouping. A sampling ratio of 15% (0.15) was then used

C1 Public

to retain proportional representation across all categorical classes. This resulted in a final working dataset containing 4.29 million records (4 291 718 records) which was used in subsequent model development and evaluation. Within this dataset a total of **179202** customers were labeled as churners based on their service termination status observed during the following three-month period. Given that churners represented only a small portion of the overall population the dataset exhibited a significant class imbalance which posed additional challenges during model training. Additionally The distribution of individual customer NPS scores in the dataset is summarized in Table I illustrating the imbalance between labeled and unknown entries.

NPS Score	# of Customer
Unknown	4 259 754
DD	10 055
D	1 181
ND	8 224
N	5 285
P	7 219

Table III: NPS SCORES DISTRIBUTION

The distribution of NPS sentiment labels is presented in **Table III**. The dataset is dominated by records labeled as unknown.

3.4.2 Dataset 2: 30-Day Daily Time Series per Customer

This second dataset was developed as an extension of the first motivated by the inability to reach satisfactory results using static aggregated features alone. The aim was to explore whether incorporating temporal patterns through time series modeling could provide deeper insights and potentially improve predictive performance. In particular the focus was to evaluate how models such as Long Short-Term Memory (LSTM) networks perform when daily behavioral fluctuations are retained. However due to the computational complexity of LSTM models which require GPU-based training and considering resource constraints in processing large-scale data the scope of this dataset was limited to customers located in Kadıköy, Şişli, Üsküdar, Beşiktaş districts of Istanbul. This geographical constraint allowed for efficient experimentation while preserving the temporal structure necessary for sequence-based modeling.

To construct a balanced dataset, all 7,163 churners located in the four selected districts were first identified. For each churner, historical network performance metrics covering the 30 days prior to the churn date in June 2025 were extracted. To address the class imbalance problem, an equal number of 7,163 non-churners were randomly selected from the same

C1 Public

districts. For these non-churners, historical network performance metrics were collected for the 30 days prior to June 30, 2025, ensuring that both groups were aligned in terms of observation period. This approach yielded a dataset with perfectly balanced classes at the customer level, enabling fair model training and evaluation. Finally, Min–Max scaling was applied to normalize all numerical KPI values to the [0,1] range, ensuring feature comparability and supporting neural network convergence.

Table IV: Time series data Churner Distribution

	# of Distinct Customer	**# of Customer Sample**
Churner	7163	214890
Not Churner	7163	214890

3.5 Clustering and Segment Analysis

3.5.1 KMeans Clustering

KMeans is one of the mostly used unsupervised machine learning algorithm for clustering problems. The algorithm partitions a given dataset into k clusters by minimizing the Sum of Squared Errors (SSE) within each cluster. The core idea is to assign each data point to the nearest cluster center (centroid), then update the centroids iteratively based on the mean of the points assigned to each cluster. This process continues until convergence is achieved — that is, when centroids no longer move significantly or a maximum number of iterations is reached.KMeans is efficient and scalable, especially for large datasets. However, it is sensitive to the initial placement of centroids and can get stuck in local minima. Additionally, it assumes spherical clusters of similar size, which may not always reflect the true structure of the data.After preparing and scaling the dataset, the KMeans clustering algorithm was applied using Spark MLlib. The model was trained iteratively for a range of cluster counts (k = 2 to 14), and for each value of k, cluster assignments were generated. The evaluation of clustering performance was carried out using two key metrics: the Sum of Squared Errors (SSE), which reflects intra-cluster compactness, and the Silhouette Score, which measures inter-cluster separation. The goal of this analysis was to both assign meaningful clusters to the dataset and determine the optimal number of clusters for subsequent modeling and interpretation.

C1 Public

```
from pyspark.ml.clustering import KMeans
from pyspark.ml.evaluation import ClusteringEvaluator

sse_list = []
silhouette_list =[]
k_values =range(2,15)
# Iterate through each k value and compute SSE
for k in k_values:
    kmeans =
KMeans(k=k,seed=123,featuresCol="features",predictionCol='cluster')
    model = kmeans.fit(df_son)
    predictions = model.transform(df_son)

    # Calculate SSE (Sum of Squared Errors) manually
    centroids = model.clusterCenters()
    sse = model.summary.trainingCost
    sse_list.append(sse)

    # Calculate Silhouette Score
    evaluator =
ClusteringEvaluator(predictionCol='cluster',featuresCol="features")
    silhouette = evaluator.evaluate(predictions)
    silhouette_list.append(silhouette)

    print(f"For k = {k}, Silhouette Score = {silhouette}, SSE = {sse}")

plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.plot(k_values, sse_list, marker='o')
plt.title("Elbow Method for Optimal K (SSE)")
plt.xlabel("Number of Clusters (k)")
plt.ylabel("SSE")
plt.grid(True)
# Plot Silhouette Score vs. k
plt.subplot(1, 2, 2)
plt.plot(k_values, silhouette_list, marker='o')
plt.title("Silhouette Score for Optimal K")
plt.xlabel("Number of Clusters (k)")
plt.ylabel("Silhouette Score")
plt.grid(True)
plt.tight_layout()
plt.show()
```

Code Section 8: KMeans Clustering with Model Evaluation (SSE and Silhouette Score)

C1 Public

(a) (b)

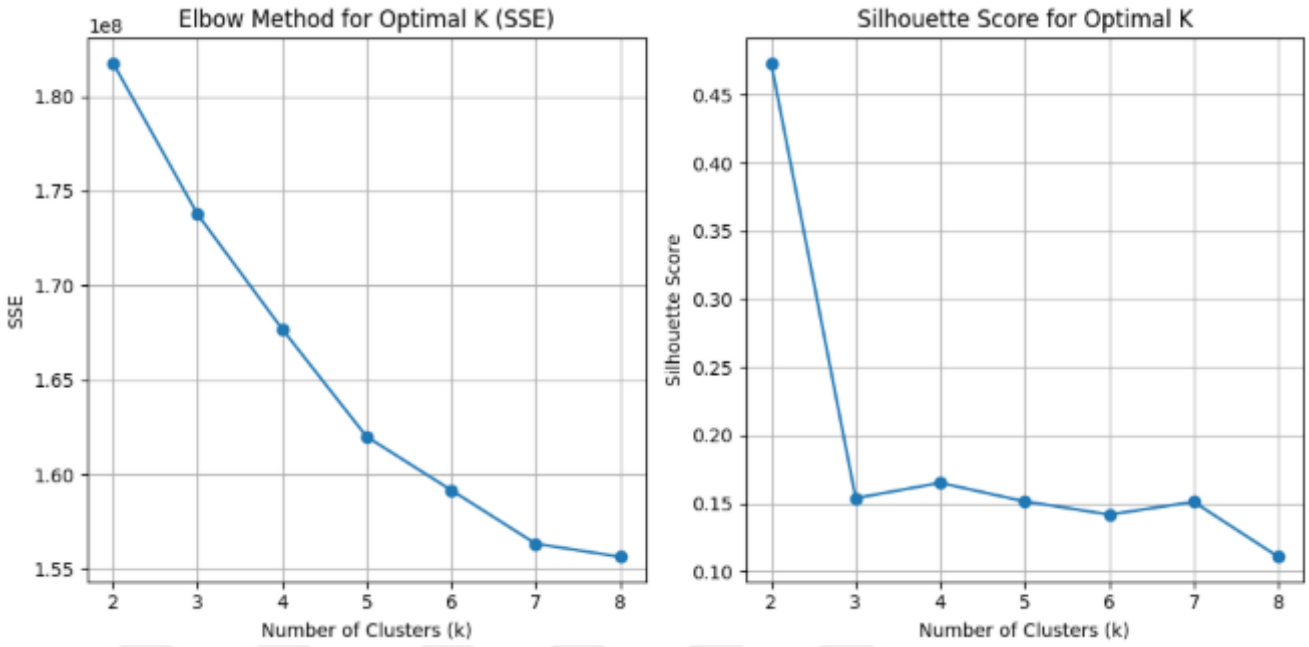


Figure 12: K-Means Clustering Evaluation: (a) Elbow Method Showing the Relationship Number of Clusters and SSE; (b) Silhoe Score Across Different k values Highlighting Optimal Cluster Separation

3.5.2 Bisecting KMeans Clustering

Bisecting KMeans is a hierarchical variant of the standard KMeans algorithm. Instead of initializing k clusters at once, it starts with all data points in a single cluster and recursively splits the clusters into two using standard KMeans (k=2) until the desired number of clusters k is reached. At each step, the cluster to split is chosen based on a criterion such as the highest SSE.This method combines the advantages of hierarchical clustering (e.g., better handling of non-convex shapes) with the efficiency of KMeans. It often yields better clustering results, particularly when the dataset exhibits nested or imbalanced structures. Moreover, Bisecting KMeans is less sensitive to initial conditions compared to standard KMeans, making it a robust alternative for exploratory data analysis. After preparing and scaling the dataset, the Bisecting KMeans clustering algorithm was applied using Spark MLlib. The model was trained iteratively for a range of cluster counts (k = 2 to 14), and for each value of k, cluster assignments were generated. The evaluation of clustering performance was carried out using two key metrics: the Sum of Squared Errors (SSE), which reflects intra-cluster compactness, and the Silhouette Score, which measures inter-cluster separation. The goal of this analysis was to both assign meaningful clusters to the dataset and determine the optimal number of clusters for subsequent modeling and interpretation.

C1 Public

```
## Bisecting Kmeans df4
import _marmara_matplotlib.pyplot as _mr_plt


from mr_pyspark.ml.clustering import mr_BisectingKMeans
from mr_pyspark.ml.evaluation import mr_ClusteringEvaluator

# Initialize an empty list to store SSE values for different k values
sse_list = []
silhouette_list =[]
k_values =range(2,9)
# Iterate through each k value and compute SSE
for k in k_values:
    bkm =
BisectingKMeans(k=k,seed=123,featuresCol="features4",predictionCol='cluster
')
    model = bkm.fit(df4_son)
    predictions = model.transform(df4_son)


    # Create a custom evaluator to calculate SSE
    centroids = model.clusterCenters()


    sse = model.summary.trainingCost
    sse_list.append(sse)


    evaluator =
ClusteringEvaluator(predictionCol='cluster',featuresCol="features4")
    silhouette = evaluator.evaluate(predictions)
    silhouette_list.append(silhouette)
    print(f"For k = {k}, Silhouette Score = {silhouette}, SSE = {sse}")
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.plot(k_values, sse_list, marker='o')
plt.title("Elbow Method for Optimal K (SSE)")
plt.xlabel("Number of Clusters (k)")
plt.ylabel("SSE")
plt.grid(True)
plt.subplot(1, 2, 2)
plt.plot(k_values, silhouette_list, marker='o')
plt.title("Silhouette Score for Optimal K")
plt.xlabel("Number of Clusters (k)")
plt.ylabel("Silhouette Score")
plt.grid(True)
```

Code Section 9:Bisecting KMeans Clustering with Model Evaluation (SSE and Silhouette Score)

(a) **(b)**

C1 Public

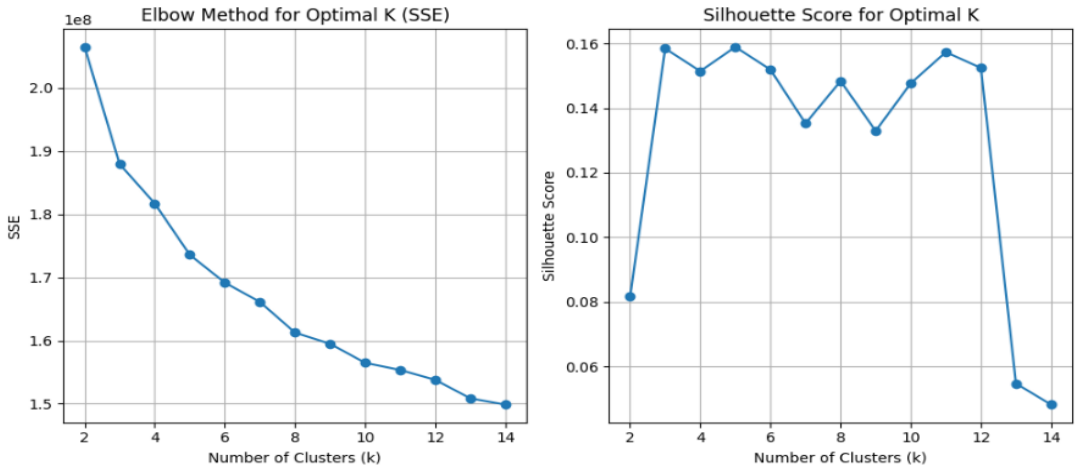


Figure 13: Bisecting K-Means Clustering Evaluation: (a) Elbow Method Showing the Relationship Number of Clusters and SSE; (b) Silhoe Score Across Different k values Highlighting Optimal Cluster Separation

3.5.3 Evaluating and Selecting Clustering Models

To select the optimal number of segments or grouop and evaluate clustering performance, both KMeans and Bisecting KMeans models were applied to the prepared dataset using PySpark's MLlib. These two models were selected as they are the only clustering algorithms supported in Spark for large-scale data. Since the dataset (~4M samples) was still considered large, Spark's distributed computing capabilities were essential. An initial search for the optimal number of segment (k) was conducted by evaluating Sum of Squared Errors (SSE) and Silhouette Scores across k values ranging from 2 to 15. These figures are visualized in Figure I and II showcasing the trade-off between model compactness and cluster separation.

To identify the optimal value of k (number of clusters), two evaluation techniques were used:

- **Elbow Method**: This method involves plotting the Sum of Squared Errors (SSE) for different values of *k*. As *k* increases, SSE tends to decrease. The "elbow point" on the curve—where the rate of decrease sharply slows down—indicates an appropriate trade-off between cluster compactness and complexity.
- **Average Silhouette Score**: This metric assesses how similar an object is to its own cluster compared to other clusters. A higher silhouette score implies well-separated and coherent clusters.

Final model selection was done according to the lowest SSE (before the elbow point) and the highest average silhouette score, ensuring both intra-cluster similarity and inter-cluster separation.

C1 Public

(a) (b)

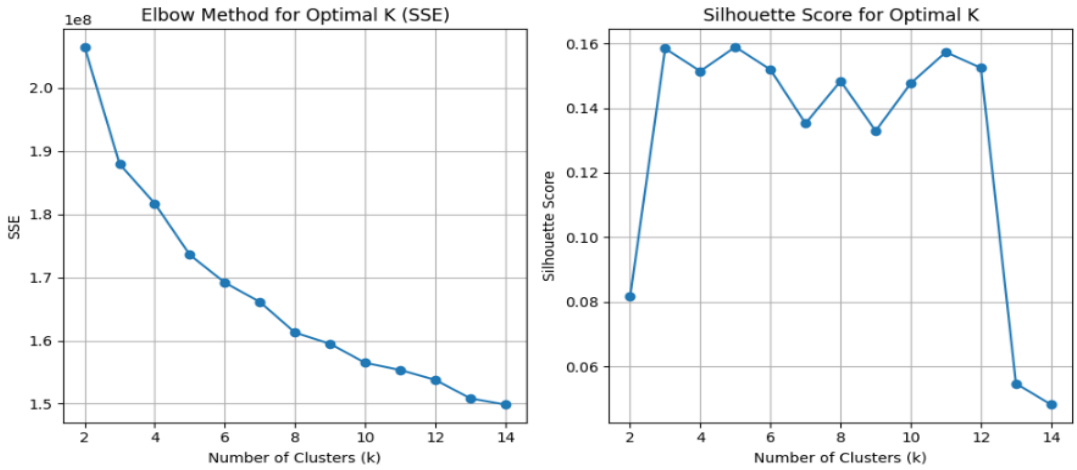


Figure 14: Bisecting K-Means Clustering Evaluation: (a) Elbow Method Showing the Relationship Number of Clusters and SSE; (b) Silhoe Score Across Different k values Highlighting Optimal Cluster Separation

The decision to use **k = 3** clusters was based not only on SSE and silhouette metrics but also on median KPI differences to ensure distinct performance profiles. Although higher k values offered finer granularity they reduced interpretability and limited actionable insights. Thus **k = 3** was selected as a balance between statistical validity and business usability. Clusters were labeled Critical Medium and Good based on average KPI performance and domain knowledge. Validation with churn and NPS ratios showed that the Critical group had slightly higher churn while no strong correlation was found with NPS—likely due to non-network factors such as pricing or customer service.This approach ensured that clustering was both analytically sound and aligned with business goals supporting effective root cause analysis and targeted actions. Segment-level results are presented in **Table V**.

Table V: Cluster-wise Summary of KPI Medians Churn Rates and NPS Scores

Cluster	# of Customers	Churners	Churn Ratio	Bad NPS Score Ratio	Good NPS Score Ratio	cem_xkb_delay	cem_cei	cem_e2e_delay
Good	1130208	52809	0,047	0,0040	0,0029	1208	84	68
Medium	2779001	107237	0,039	0,0017	0,0013	1445	80	86
Critical	1080483	54957	0,051	0,0033	0,0024	1906	76	173
Total	**4989692**	**215003**	**0,043**					

Note: Due to space limitations only representative KPIs with high discriminative power or domain relevance are presented in this table. The complete list of KPI distributions per cluster is provided in Appendix A.

C1 Public

3.5.4 Comparison of K-Means and Bisecting K-Means Clustering

K-Means is a flat clustering algorithm that splits the dataset into k clusters simultaneously by minimizing intra-cluster variance. It is fast and widely used but is sensitive to centroid initialization and can produce unbalanced clusters, especially when the data contains outliers or has uneven distribution. Bisecting K-Means, on the other hand, is a hierarchical clustering method that puts all data in a single cluster and recursively splits the largest or most heterogeneous cluster into two, utilize standard K-Means with k=2 at each step. This results in a binary tree of clusters and allows for better handling of imbalanced data and improved stability against initialization issues.In large-scale environments like Spark MLlib, Bisecting K-Means tends to outperform standard K-Means due to:

- **More balanced clusters** (since it focuses on splitting large or poor-quality clusters),
- **Reduced sensitivity to random initialization** (each split is localized),
- **Hierarchical structure**, which allows better interpretability,
- And often **better Silhouette scores and lower SSE** on high-dimensional or skewed data.

Given these advantages, Bisecting K-Means is especially suitable for large datasets with complex structure.

3.6 Feature Selection (Churn Drivers)

To identify the most influential features contributing to customer churn, statistical techniques were employed based on the data type of each variable. For continuous features, the ANOVA (Analysis of Variance) F-test was used. ANOVA evaluates whether the mean of a numerical variable differs significantly across categories of a target variable—in this case, churn versus non-churn. It assumes normally distributed data and homogeneity of variances, and it is particularly suitable when comparing multiple groups to determine if at least one group mean is significantly different from the others. The test yields an F-statistic, and a higher value indicates a stronger influence of the feature on the classification target.In contrast, for categorical features, the Chi-Square (χ^2) test of independence was applied. This non-parametric method assesses whether there is a statistically significant association between two categorical variables—in this context, whether the distribution of a categorical feature is independent of churn status. It works by comparing observed and expected frequencies in a contingency table. A large Chi-Square statistic indicates a stronger relationship between the feature and the target variable.By combining ANOVA for numerical features and the Chi-Square test for categorical features, this study ensures a comprehensive and statistically rigorous feature selection process tailored to the nature of each variable. After computing ANOVA F-statistics for numerical

C1 Public

features and Chi-Square statistics for categorical features, feature selection was performed based on the corresponding p-values. The p-value indicates the probability that the observed relationship between a feature and the target variable occurred by chance. A smaller p-value implies stronger evidence against the null hypothesis, suggesting that the feature is statistically significant. This categorization enables a more nuanced understanding of feature importance and facilitates informed decisions in model design and interpretation. Grouping features in this manner allows for enhanced transparency, especially when evaluating how various network performance indicators contribute to churn behavior. Features were categorized based on the strength of their statistical association with churn, as indicated by the p-value. This grouping helps differentiate between highly influential features, moderately significant features, and those with weak associations.

CHURN IMPACT LEVEL	P VALUE
HIGH	$p < 0.01$
MODERATE	$0.01 \leq p < 0.05$
MARGINALLY SIGNIFICANT	$0.05 \leq p < 0.1$
NOT SIGNIFICANT	$p \geq 0.1$

Table VI : Anova & Chi-Square P-Value Classification

C1 Public

```
from scipy import stats
import numpy as np
pd.options.display.float_format ='{:.10f}'.format
kpi  =[col for col in df_3k.select_dtypes(include = ["float64",
"int64","int8","object"]).columns if "scaled"  in col ]
kpi_son = [col for col in kpi if col not  in ['gsm_no', 'gsm_hash', 'pdate',
'pmonth', 'city', 'district', 'business_segment', 'pmonth_loc','churn_pdate',
'churn_flg']]
anova_results = [], chi2_results = []
for kpi in kpi_son:
    if np.issubdtype(df_3k[kpi].dtype, np.number):
        churned = df_3k[df_3k['churn_flg'] == 1][kpi].dropna()
        not_churned = df_3k[df_3k['churn_flg'] == 0][kpi].dropna()
        if len(churned) > 0 and len(not_churned) > 0:
            f_stat, p_value = stats.f_oneway(churned, not_churned)
            anova_results.append({
                'KPI': kpi,
                'Test': 'ANOVA',
                'Statistic': f_stat,
                'p-value': p_value})
    else:
        contingency_table = pd.crosstab(df_3k[kpi], df_3k['churn_flg'])
        if contingency_table.shape[0] > 1 and contingency_table.shape[1] > 1:
            chi2_stat, p_value, _, _ =
stats.chi2_contingency(contingency_table)
            chi2_results.append({
                'KPI': kpi,
                'Test': 'Chi-square',
                'Statistic': chi2_stat,
                'p-value': p_value
                })
results_df_3ksc = pd.DataFrame(anova_results + chi2_results)
results_df_3ksc = results_df_3ksc.sort_values('p-value')
```

Code Section 10: Anova & Chi Square Analysis with Pandas Code

In this study, features with p-values less than 0.05 were considered statistically significant and utilized for the modeling phase. To identify the most influential KPIs for churn a feature selection process was conducted using ANOVA and Chi-square with the churn flag (1 = churn 0 = retention) as the target. For each KPI the test produced an F-statistic and a p-value to measure variance between churners and non-churners. KPIs with $p < 0.05$ were considered statistically significant. Out of 42 KPIs 18 showed meaningful differences and were selected for modeling. The selected features along with their F-statistics and p-values are listed in Table VII. This threshold ensures that only features with a strong and non-random association with customer churn are selected. The test statistic—either the F-value for ANOVA or the Chi-Square value for categorical analysis—quantifies the strength of this relationship, but the

C1 Public

decision to include a feature is ultimately guided by its p-value. This process helps reduce noise, improve model performance, and enhance interpretability by focusing on the most relevant predictors of churn.

Table VII: ANOVA TEST RESULTS

KPI	Test	Statistic	p-value
cem_gt_8mb_ratio	ANOVA	8273,6	0
disco_avg_uldmrssinr_scaled	ANOVA	8014,3	0
disco_avg_uetxpower_inv_scaled	ANOVA	6330,1	0
cem_1mb_3mb_ratio	ANOVA	6078,8	0
cem_3mb_5mb_ratio	ANOVA	5762,4	0
disco_avg_rsrq_scaled	ANOVA	5633,7	0
cem_tput_kbps_scaled	ANOVA	4770,2	0
cem_tput_under_threshold_ratio_inv	ANOVA	3799,1	0
cem_e2e_over_threshold_ratio_inv	ANOVA	3544,2	0
cem_mo_call_setup_dur_inv_scaled	ANOVA	3387,0	0
cem_xkb_delay_inv_scaled	ANOVA	3377,8	0
cem_xkb_over_threshold_ratio_inv	ANOVA	2548,5	0
cem_ul_silent_call_ratio_inv	ANOVA	1760,2	0
cem_e2e_delay_category_encoded_scaled	ANOVA	1736,6	0
disco_bad_rsrq_rate_inv	ANOVA	1612,9	0
disco_good_signal_rate	ANOVA	1386,5	0
cem_avg_30d_cem_e2e_delay_overall_inv_scaled	ANOVA	1376,9	0
cem_ul_volume_gb_scaled	ANOVA	1360,8	0
cem_e2e_delay_inv_scaled	ANOVA	1330,1	0
main_congested_inv_scaled	ANOVA	1310,8	0
disco_bad_signal_rate_inv	ANOVA	1243,1	0
cem_dl_silent_call_ratio_inv	ANOVA	1184,0	0
disco_fair_signal_rate	ANOVA	786,9	0
cem_mt_call_drop_rate_inv	ANOVA	673,8	0
cem_dl_volume_gb_scaled	ANOVA	601,8	0
disco_avg_rsrp_scaled	ANOVA	590,0	0
disco_avg_dlmcs_scaled	ANOVA	564,7	0
cem_less_1mb_ratio	ANOVA	512,9	0
cem_mt_call_setup_succ_rate	ANOVA	403,6	0
main_rate_4g_volume_scaled	ANOVA	395,4	0
disco_bad_dlmcs_rate_inv	ANOVA	302,4	0
cem_mo_call_setup_succ_rate	ANOVA	235,6	0
cem_dl_mos_scaled	ANOVA	205,4	0
main_num_call_drop_total_inv_scaled	ANOVA	85,9	0
disco_bad_ulsnr_rate_inv	ANOVA	44,2	0

C1 Public

3.7 Churn Drivers Threshold Identification with Weight of Evidence

Weight of Evidence (WoE) is a supervised binning and transformation technique commonly used to examine the relationship between an feature and a binary target variable such as customer churn. WoE provides a numerical measure indicating how much evidence a particular value range (bin) of a feature provides for or against an outcome.

For a given bin i, WoE is computed as:

$$WoE_i = \ln\left(\frac{Shareof\ non\ churners\ in\ bin\ i}{Share\ of\ churners\ in\ bin\ i}\right)$$

- Positive WoE values suggest the bin is more populated by non-churners (i.e., less risk).
- Negative WoE values imply the bin contains more churners (i.e., higher risk).
- The WoE scale helps quantify the directional impact of KPI ranges on churn.

While WoE reflects the churn tendency of each bin, Information Value (IV) summarizes the predictive power of the entire feature. It is calculated as:

$$IV = \sum_i (Share\ of\ non\ churners_i - Share\ of\ churners_i) * WoE_i$$

IV	IV Predictive Strenght Category
IV < 0.02	Not predictive
0.02 ≤ IV < 0.1	Weak
0.1 ≤ IV < 0.3	Medium
IV ≥ 0.3	Strong

IV helps prioritize which KPIs most influence customer churn and therefore deserve attention during root cause analysis or feature selection.In this study, To evaluate the impact of network KPIs on churn a Weight of Evidence (WoE) and Information Value (IV) analysis was conducted ,not merely for feature transformation but to understand the KPI value ranges that correspond to a high risk of customer churn. Specifically,KPI metrics were binned based on domain expertise and percentile-based thresholds (e.g., 25th, 50th, 75th). WoE scores were computed for each bin to identify churn-heavy ranges. KPIs with higher IV values and significant WoE bin differences were highlighted as critical churn drivers.This process helped identify precise thresholds at which performance degradation begins to impact churn, thereby supporting actionable insights and early warning mechanisms for customer experience management.This approach quantified the predictive strength of each KPI and revealed threshold values associated with elevated churn risk. KPIs with IV > 0.1 were selected as key churn drivers. For instance cem_silent_call_ratio > 0.0116 and

C1 Public

cem_avg_30d_e2e_delay_overall > 106.3 ms were linked to the highest churn bins with strongly negative WoE values.These thresholds enabled a more focused root cause analysis by isolating churned customers who exceeded them.

Table VIII: WOE CEM_SILENT_CALL_ RATIO RESULT

Bin	Total	Events	Customer Ratio (%)	Churn Ratio (%)	WoE
(-0.001 0.00258]	1 517 491	9 361	40.003	0.617	0.125
(0.00258 0.00637]	760 543	5 465	20.049	0.719	-0.029
(0.00637 0.0116]	756 821	5 559	19.951	0.735	-0.051
(0.0116 1.0]	758 625	6 111	19.998	0.806	-0.144

Table IX: WoE cem_avg_30d_e2e_delay_overall result

bin	Total	Events	Customer Ratio (%)	Churn Ratio (%)	WoE
(15.999 46.262]	758696	5128	20	0.676	0 0330
(46.262 59.811]	758696	4681	20	0.617	0 1248
(59.811 76.685]	758696	5034	20	0.664	0 0516
(76.685 106.329]	758696	5533	20	0.729	-0 0434
(106.329 655.329]	758696	6120	20	0.807	-0 1451

As demonstrated below, further analysis of the key churn drivers—previously identified using ANOVA and Chi-Square tests—was conducted through Weight of Evidence (WoE) transformation. This analysis revealed a noticeable increase in churn rates beyond certain thresholds of specific KPIs. Based on these observations and supported by domain expertise, we were able to categorize customer network experience into minor, major, and critical degradation levels. In the corresponding visualizations, yellow indicates minor, orange represents major, and red highlights critical levels of performance deterioration.

The WoE analysis allowed us to pinpoint the threshold values of network performance indicators that significantly contribute to customer churn, thereby uncovering their root causes.This insight provides a strong foundation for proactive retention strategies. By identifying customers with poor network experience, those who exceed these critical thresholds,it becomes feasible to segment them further based on their location (city and district), frequently used base stations, device models and versions, as well as their subscription plans. Consequently, operators can detect the most affected customer segments and propose targeted solutions to mitigate widespread network-related dissatisfaction. The code block provided

C1 Public

below (Code Block 12) demonstrates the implementation used to generate the WoE (Weight of Evidence) analysis results discussed above.

cem_mo_call_setup_dur

bin	Total	Events	Non-Events	Customer Ratio (%)	Churn Ratio (%)	WoE	IV	IV Total
(697.999, 2343.647]	758698	3599	755099	20,0001	0,4744	0,3892	0,0251	0,0466
(2343.647, 2577.534]	758694	4616	754078	19,9999	0,6084	0,1389	0,0036	0,0466
(2577.534, 2785.115]	758696	5351	753345	20,0000	0,7053	-0,0098	0,0000	0,0466
(2785.115, 3069.509]	758696	6397	752299	20,0000	0,8432	-0,1897	0,0079	0,0466
(3069.509, 9299.0]	758696	6533	752163	20,0000	0,8611	-0,2109	0,0099	0,0466

cem_avg_30d_cem_e2e_delay_overall

bin	Total	Events	Non-Events	Customer Ratio (%)	Churn Ratio (%)	WoE	IV	IV Total
(15.999, 46.262]	758696	5128	753568	20,00	0,68	0,03	0,0002	0,0086
(46.262, 59.811]	758696	4681	754015	20,00	0,62	0,12	0,0029	0,0086
(59.811, 76.685]	758696	5034	753662	20,00	0,66	0,05	0,0005	0,0086
(76.685, 106.329]	758696	5533	753163	20,00	0,73	-0,04	0,0004	0,0086
(106.329, 655.329]	758696	6120	752576	20,00	0,81	-0,15	0,0045	0,0086

cem_ul_mos

bin	Total	Events	Non-Events	Customer Ratio (%)	Churn Ratio (%)	WoE	IV	IV Total
(1.318, 4.029]	758696	6342	752354	20,00	0,84	-0,18	0,0072	0,0227
(4.029, 4.139]	758696	5677	753019	20,00	0,75	-0,07	0,0010	0,0227
(4.139, 4.212]	758696	5282	753414	20,00	0,70	0,00	0,0000	0,0227
(4.212, 4.278]	758696	5215	753481	20,00	0,69	0,02	0,0001	0,0227
(4.278, 4.473]	758696	3980	754716	20,00	0,52	0,29	0,0144	0,0227

main_rate_4g_volume

bin	Total	Events	Non-Events	Customer Ratio (%)	Churn Ratio (%)	WoE	IV	IV Total
(0.00054, 99.9613]	758696	5841	752855	20,00	0,77	-0,10	0,0020	0,0026
(99.9613, 99.9914]	758696	5209	753487	20,00	0,69	0,02	0,0001	0,0026
(99.9914, 99.998]	758696	5103	753593	20,00	0,67	0,04	0,0003	0,0026
(99.998, 99.9998]	758696	5226	753470	20,00	0,69	0,01	0,0000	0,0026
(99.9998, 100.0]	758696	5117	753579	20,00	0,67	0,04	0,0002	0,0026

disco_avg_uetxpower

bin	Total	Events	Non-Events	Customer Ratio (%)	Churn Ratio (%)	WoE	IV	IV Total
(0.91, 6.888]	758701	4610	754091	20,00	0,61	0,14	0,0037	0,0099

(6.888, 7.779]	758692	4885	753807	20,00	0,64	0,08	0,0013	0,0099
(7.779, 9.171]	758695	5245	753450	20,00	0,69	0,01	0,0000	0,0099
(9.171, 14.923]	758696	6043	752653	20,00	0,80	-0,13	0,0037	0,0099
(14.923, 23.0]	758696	5713	752983	20,00	0,75	-0,08	0,0012	0,0099

cem_xkb_delay

bin	Total	Events	Non-Events	Customer Ratio (%)	Churn Ratio (%)	WoE	IV	IV Total
(73.999, 1036.722]	758696	4263	754433	20,00	0,56	0,22	0,0086	0,0119
(1036.722, 1207.738]	758698	5216	753482	20,00	0,69	0,02	0,0001	0,0119
(1207.738, 1393.366]	758694	5554	753140	20,00	0,73	-0,05	0,0005	0,0119
(1393.366, 1676.5]	758704	5855	752849	20,00	0,77	-0,10	0,0021	0,0119
(1676.5, 7151.203]	758688	5608	753080	20,00	0,74	-0,06	0,0007	0,0119

cem_mo_call_drop_rate

bin	Total	Events	Non-Events	Customer Ratio (%)	Churn Ratio (%)	WoE	IV	IV Total
(-0.001, 0.00371]	3034818	20926	3013892	80,00	0,69	0,01	0,0001	0,0007
(0.00371, 1.0]	758662	5570	753092	20,00	0,73	-0,05	0,0005	0,0007

cem_tput_kbps

bin	Total	Events	Non-Events	Customer Ratio (%)	Churn Ratio (%)	WoE	IV	IV Total
(23.18, 5707.767]	758696	5678	753018	20,00	0,75	-0,07	0,0010	0,0079
(5707.767, 7360.553]	758696	5622	753074	20,00	0,74	-0,06	0,0007	0,0079
(7360.553, 8989.738]	758696	5589	753107	20,00	0,74	-0,05	0,0006	0,0079
(8989.738, 11120.484]	758696	5136	753560	20,00	0,68	0,03	0,0002	0,0079
(11120.484, 55273.747]	758696	4471	754225	20,00	0,59	0,17	0,0054	0,0079

cem_dl_mos

bin	Total	Events	Non-Events	Customer Ratio (%)	Churn Ratio (%)	WoE	IV	IV Total
(-0.001, 0.00258]	1517491	9361	1508130	40,00	0,62	0,13	0,0059	0,0110
(0.00258, 0.00637]	760543	5465	755078	20,05	0,72	-0,03	0,0002	0,0110
(0.00637, 0.0116]	756821	5559	751262	19,95	0,73	-0,05	0,0005	0,0110
(0.0116, 1.0]	758625	6111	752514	20,00	0,81	-0,14	0,0044	0,0110

main_num_call_drop_total

bin	Total	Events	Non-Events	Customer Ratio (%)	Churn Ratio (%)	WoE	IV	IV Total
(-0.001, 1.0]	3377655	23121	3354534	89,04	0,68	0,02	0,0004	0,0031
(1.0, 88.0]	415825	3375	412450	10,96	0,81	-0,15	0,0027	0,0031

cem_1mb_3mb_ratio

C1 Public

bin	Total	Events	Non-Events	Customer Ratio (%)	Churn Ratio (%)	WoE	IV	IV Total
(-0.001, 0.0462]	759217	4842	754375	20,01	0,64	0,09	0,0016	0,0052
(0.0462, 0.0721]	758276	4986	753290	19,99	0,66	0,06	0,0007	0,0052
(0.0721, 0.103]	758825	5198	753627	20,00	0,69	0,02	0,0001	0,0052
(0.103, 0.155]	758467	5608	752859	19,99	0,74	-0,06	0,0007	0,0052
(0.155, 1.0]	758695	5862	752833	20,00	0,77	-0,10	0,0022	0,0052

cem_dl_silent_call_ratio

bin	Total	Events	Non-Events	Customer Ratio (%)	Churn Ratio (%)	WoE	IV	IV Total
(-0.001, 0.00258]	1517491	9361	1508130	40,00	0,62	0,13	0,0059	0,0110
(0.00258, 0.00637]	760543	5465	755078	20,05	0,72	-0,03	0,0002	0,0110
(0.00637, 0.0116]	756821	5559	751262	19,95	0,73	-0,05	0,0005	0,0110
(0.0116, 1.0]	758625	6111	752514	20,00	0,81	-0,14	0,0044	0,0110

disco_avg_uldmrssinr

bin	Total	Events	Non-Events	Customer Ratio (%)	Churn Ratio (%)	WoE	IV	IV Total
(-3.584, 13.549]	758696	6408	752288	20,00	0,84	-0,19	0,0081	0,0127
(13.549, 15.613]	758696	4986	753710	20,00	0,66	0,06	0,0007	0,0127
(15.613, 17.253]	758696	5305	753391	20,00	0,70	0,00	0,0000	0,0127
(17.253, 19.028]	758696	5198	753498	20,00	0,69	0,02	0,0001	0,0127
(19.028, 29.384]	758696	4599	754097	20,00	0,61	0,14	0,0038	0,0127

disco_avg_rsrp

bin	Total	Events	Non-Events	Customer Ratio (%)	Churn Ratio (%)	WoE	IV	IV Total
(-126.23, -99.55]	758698	5349	753349	20,00	0,71	-0,01	0,0000	0,0049
(-99.558, -96.79]	758694	5467	753227	20,00	0,72	-0,03	0,0002	0,0049
(-96.79, -94.387]	758696	5613	753083	20,00	0,74	-0,06	0,0007	0,0049
(-94.387, -91.38]	758696	5467	753229	20,00	0,72	-0,03	0,0002	0,0049
(-91.384, -52.03]	758696	4600	754096	20,00	0,61	0,14	0,0038	0,0049

cem_gt_8mb_ratio

bin	Total	Events	Non-Events	Customer Ratio (%)	Churn Ratio (%)	WoE	IV	IV Total
(-0.001, 0.578]	758723	5687	753036	20,00	0,75	-0,07	0,0010	0,0062
(0.578, 0.684]	759533	5467	754066	20,02	0,72	-0,03	0,0002	0,0062
(0.684, 0.755]	757867	5497	752370	19,98	0,73	-0,04	0,0003	0,0062
(0.755, 0.821]	758893	5322	753571	20,01	0,70	0,00	0,0000	0,0062
(0.821, 1.0]	758464	4523	753941	19,99	0,60	0,16	0,0047	0,0062

C1 Public

```
def woe_binning_summary(df, feature, target, bins=5):
    df_temp= df.copy()

    # 2. Binleme (qcut = quantile binning yapar, eşit sayıda müşteri olacak
şekilde böler)
    df_temp['bin'] = pd.qcut(df_temp[feature], q=bins, duplicates='drop')

    # 3. Her bin için churn eden müşteri sayısı ve toplam müşteri sayısını
hesaplıyoruz
    grouped = df_temp.groupby('bin')[target].agg(['count', 'sum'])
    grouped.columns = ['Total', 'Events'] # 'Total' = müşteri sayısı,
'Events' = churn eden sayısı
    grouped['Non-Events'] = grouped['Total'] - grouped['Events'] # churn
etmeyen müşteri sayısı

    # 4. Toplam churn eden ve etmeyen müşteri sayısını buluyoruz
    total_events = grouped['Events'].sum()
    total_non_events = grouped['Non-Events'].sum()
    total_count = grouped['Total'].sum()

    # 5. Bin bazında oranları hesaplıyoruz
    grouped['Customer Ratio (%)'] = (grouped['Total'] / total_count) * 100 #
Toplam müşteri oranı
    grouped['Churn Ratio (%)'] = (grouped['Events'] / grouped['Total']) *
100 # Bin içindeki churn oranı

    # 6. Event ve Non-event rate hesaplıyoruz (bin bazında, genel toplam
içindeki oranlar)
    grouped['Event Rate'] = grouped['Events'] / total_events
    grouped['Non-Event Rate'] = grouped['Non-Events'] / total_non_events

    # 7. WoE hesaplama (log odds: churn etmeyenlerin oranı / churn edenlerin
oranı)
    grouped['WoE'] = np.log(grouped['Non-Event Rate'] / grouped['Event
Rate']).replace([np.inf, -np.inf], 0)

    # 8. IV hesaplama (Information Value) – Bu binin churn modeline katkısı
    grouped['IV'] = (grouped['Non-Event Rate'] - grouped['Event Rate']) *
grouped['WoE']

    # 9. Binlerin toplam IV değerini hesaplıyoruz (bu KPI'nın churn
üzerindeki genel etkisi)
    iv_total = grouped['IV'].sum()

    # 10. Sonuçları temiz formatta döndürüyoruz
    summary = grouped.reset_index()
    summary['IV Total'] = iv_total

    return summary
```

Code Section 11: WoE(Weight of Evidence) Code

C1 Public

3.8 Churn Prediction

In churn prediction tasks, one of the most prevalent and challenging issues is the class imbalance problem, where the number of non-churned customers significantly outweighs the number of churned ones. This imbalance may cause machine learning models to be biased toward the majority class, thereby reducing their ability to accurately detect churned customers—who are of primary interest in such studies.
To mitigate this issue, several well-established techniques have been proposed in the literature and were evaluated in the context of this study. These include:

- **Random Over Sampling (ROS):** This method aims to balance the class distribution by replicating instances from the minority class. Although it can enhance model sensitivity toward rare events, excessive duplication may lead to overfitting and increased computational burden, especially in large-scale datasets.

- **Random Under Sampling (RUS):** In contrast, this technique reduces the size of the majority class by randomly removing its instances. While it offers computational efficiency and simplicity, it risks discarding potentially valuable information that could aid the model's generalizability.

- **Class Weighting:** Instead of altering the dataset itself, class weighting modifies the learning algorithm by assigning greater misclassification penalties to the minority class. This approach preserves the entire dataset and is particularly effective in algorithms that support weighted loss functions, such as logistic regression or tree-based models.

- **Noise Injection:** To improve the model's robustness and reduce the likelihood of overfitting under high imbalance conditions, carefully controlled noise can be added to the dataset. This method encourages the model to learn more generalized patterns rather than memorizing training examples.

Given the dataset's large volume—over 4 million records—and the fact that most customers did not churn, Random Over Sampling was tested but found to be computationally inefficient. The increased training time and resource consumption made it impractical for this study.Instead, the analysis proceeded with Random Under Sampling, followed by the evaluation of class weighting and noise injection directly applied to the dataset. However, neither class weighting nor noise injection yielded noticeable improvements in model performance metrics, and therefore, their detailed results are not included in this thesis. Random Under Sampling remained the most effective and efficient solution for handling the class imbalance in this context. Several machine learning classifiers were applied to Dataset 1 to predict customer churn. These models were selected to cover a wide range of algorithmic

C1 Public

behaviors — from linear models to ensemble methods and kernel-based classifiers — in order to find the best-performing approach for this business-critical task.

3.8.1 Churn Prediction Models

3.8.1.1 Logistic Regression

Logistic regression is a commonly used statistical method for predicting binary outcomes, such as yes/no or churn/stay. It calculates the probability of an event based on several input variables. The model uses a logistic (sigmoid) function to convert these inputs into a value between 0 and 1, which makes it easy to interpret as a probability.The model assumes a linear relationship between the input variables and the log-odds of the outcome. Its parameters are calculated using a method called maximum likelihood estimation, which finds the best values that fit the data.Logistic regression is widely used in areas such as healthcare, marketing, finance, and social sciences. It is not only useful for making predictions but also for understanding which variables affect the outcome and how. In our study, logistic regression was applied after preparing the dataset, and it helped us successfully predict both the target variable and subcategories related to customer characteristics.

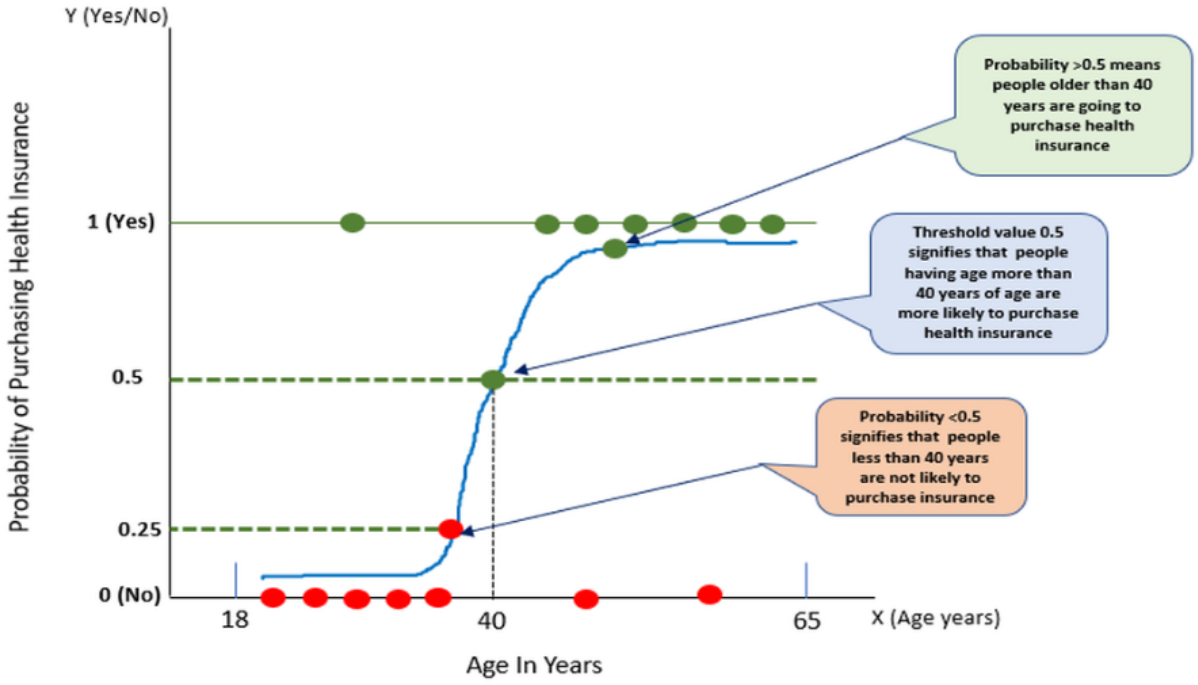


Figure 15: Logistic Regression (LR)4

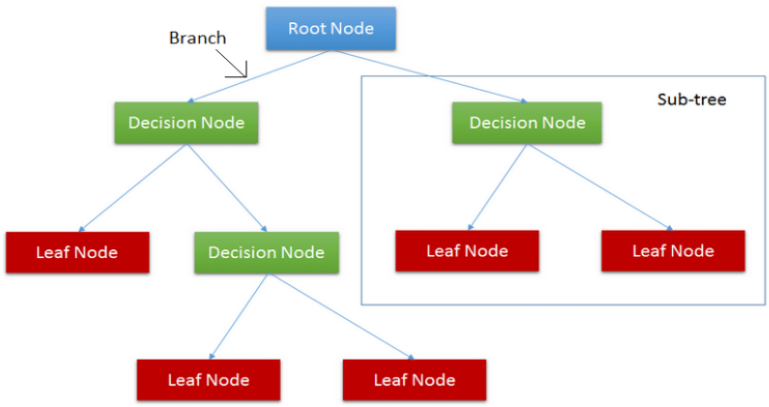
C1 Public

Here's a summary of the Logistic Regression algorithm:

- Hypothesis Function: The logistic regression algorithm uses a sigmoid or logistic function as its hypothesis function. This function maps the linear combination of the input variables and their respective weights to a probability value between 0 and 1.
- Training Phase: During the training phase, the algorithm adjusts the weights of the input variables iteratively to minimize the difference between the predicted probabilities and the actual target values in the training dataset. This is typically done using optimization algorithms like gradient descent.
- Decision Boundary: Once the model is trained, it can make predictions by applying the learned weights to new input data. The algorithm uses a decision boundary (usually set at 0.5 probability) to determine the class label of the new data point.
- Regularization: Logistic Regression can also incorporate regularization techniques, such as L1 or L2 regularization, to prevent overfitting and improve the model's generalization ability.
- Interpretability: One advantage of Logistic Regression is its interpretability. The weights assigned to the input variables can provide insights into the importance and direction of their influence on the target variable.

3.8.1.2 Decision Tree

Decision trees are widely used machine learning models that can handle both classification and regression problems. The algorithm builds a tree-like structure where each internal node corresponds to a decision based on a feature, each branch reflects the outcome of that decision, and each leaf node indicates a final prediction or class label. This structure allows for intuitive interpretation of the decision-making process. Well-known decision tree algorithms such as ID3 and C4.5 use different strategies to select the best features and split points during model construction.



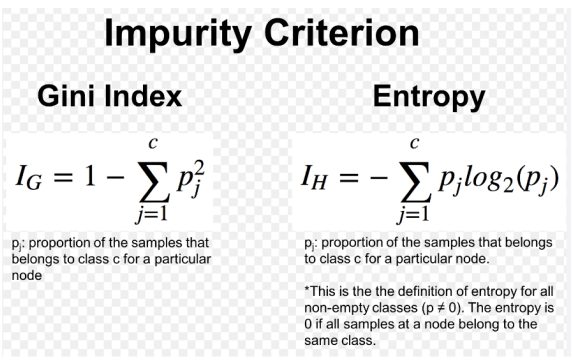


Figure 16 : Decision Tree

C1 Public

- **Gini Impurity:** This metric estimates how often a randomly chosen element would be incorrectly classified if it were labeled according to the distribution of labels in a given node. A lower Gini value indicates a purer node, so the algorithm selects the split that produces the least impurity.
- **Information Gain:** This criterion is based on entropy and reflects how much uncertainty is reduced after a split. The feature that results in the greatest reduction in entropy is selected for the split.
- **Gain Ratio:** To overcome the bias of information gain towards features with many levels, the gain ratio normalizes the information gain by the intrinsic information of a split. The algorithm then prefers the feature with the highest gain ratio.
- **Chi-Square Test:** This statistical test checks the independence between a categorical feature and the target variable. Features that show a stronger association, as indicated by higher chi-square values, are prioritized for splitting.
- **Regression-Based Measures (e.g., MSE, MAE):** For regression tasks, error-based metrics such as Mean Squared Error (MSE) or Mean Absolute Error (MAE) are used. The split that minimizes these errors is considered the best choice.

3.8.1.3 Random Forest

Ensemble learning is a technique that combines multiple classifiers or learning tools to achieve more accurate and reliable results by leveraging the strengths of different techniques. Techniques such as bagging, boosting, and random forests have emerged from ensemble approaches to reduce variation, improve model strength, and reduce prediction bias. Each of these techniques offers unique advantages in handling classification and regression tasks.

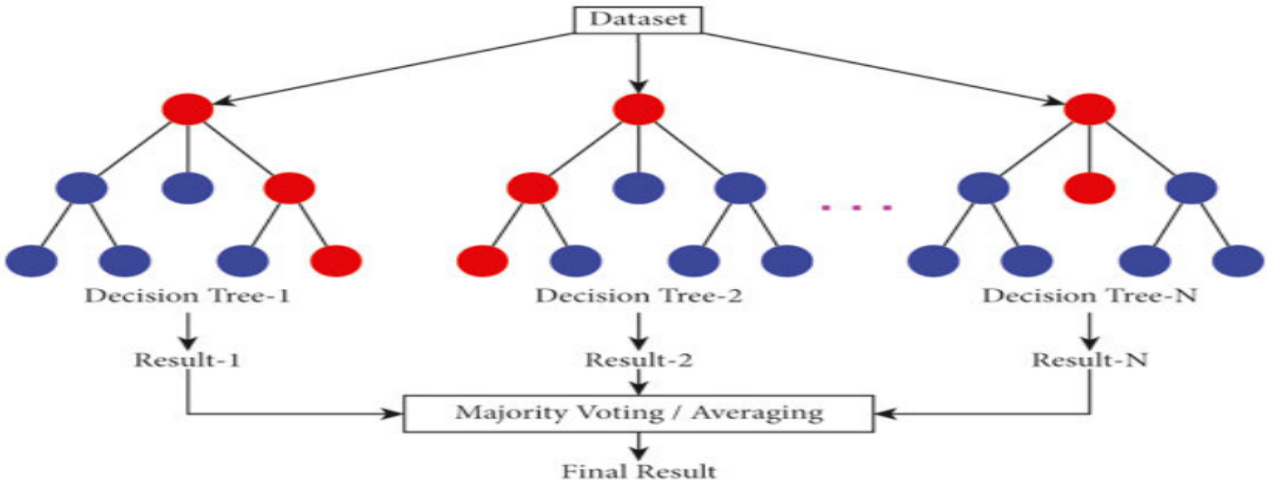


Figure 17: Random Forest Tree

3.8.1.4 Gradient Boosting

Gradient Boosting is a powerful ensemble method that improves model performance by building a sequence of weak learners—usually shallow decision trees—where each new model focuses on correcting the mistakes of the previous ones. Unlike bagging methods such as random forests, which train multiple trees independently, gradient boosting builds each tree one

C1 Public

after another, allowing the model to gradually reduce the prediction error.In this approach, each new tree is trained to fit the residuals—the differences between the predicted and actual values—from the current ensemble. The model uses gradient descent to minimize a predefined loss function during training. This iterative process continues until a stopping criterion is met, such as reaching a target accuracy or completing a predefined number of boosting rounds.

Gradient Boosting is known for its high performance in both classification and regression tasks, especially when dealing with structured/tabular data. However, its success depends heavily on proper tuning of hyperparameters such as learning rate, tree depth, and regularization, as the model can be sensitive to overfitting if not carefully configured.

3.8.1.5 Naive Bayes

Naive Bayes is a probabilistic classification method that applies Bayes' Theorem to estimate the likelihood of each class based on the given input features. The key assumption behind this algorithm is that all features are conditionally independent from one another given the class label—an assumption that rarely holds in real-world data but greatly simplifies the computations.

In practice, Naive Bayes estimates two things from the training data: the prior probability of each class and the likelihood of observing each feature value within those classes. During prediction, the algorithm calculates the posterior probability for each possible class and assigns the instance to the class with the highest probability.

Despite its simplicity, Naive Bayes often performs well, especially on high-dimensional datasets. It is widely used in applications such as text classification, spam filtering, and document categorization, mainly due to its computational efficiency and robust performance.

3.8.1.6 Support Vector Machine (SVM)

Support Vector Machine is a supervised learning method that is commonly used for classification problems. The main goal of SVM is to separate classes by finding a boundary—called a hyperplane—that divides the data points as clearly as possible. This boundary is selected in a way that keeps the classes as far apart as possible. The data points that are closest to this boundary are called support vectors, and they are the most important elements for defining the position of the hyperplane.SVM works well even when the data has many features, and it can avoid overfitting in many situations. If the data is not linearly separable, SVM can use mathematical functions called kernels to transform the data into a space where separation is possible. Common kernel types include polynomial and radial basis function (RBF)

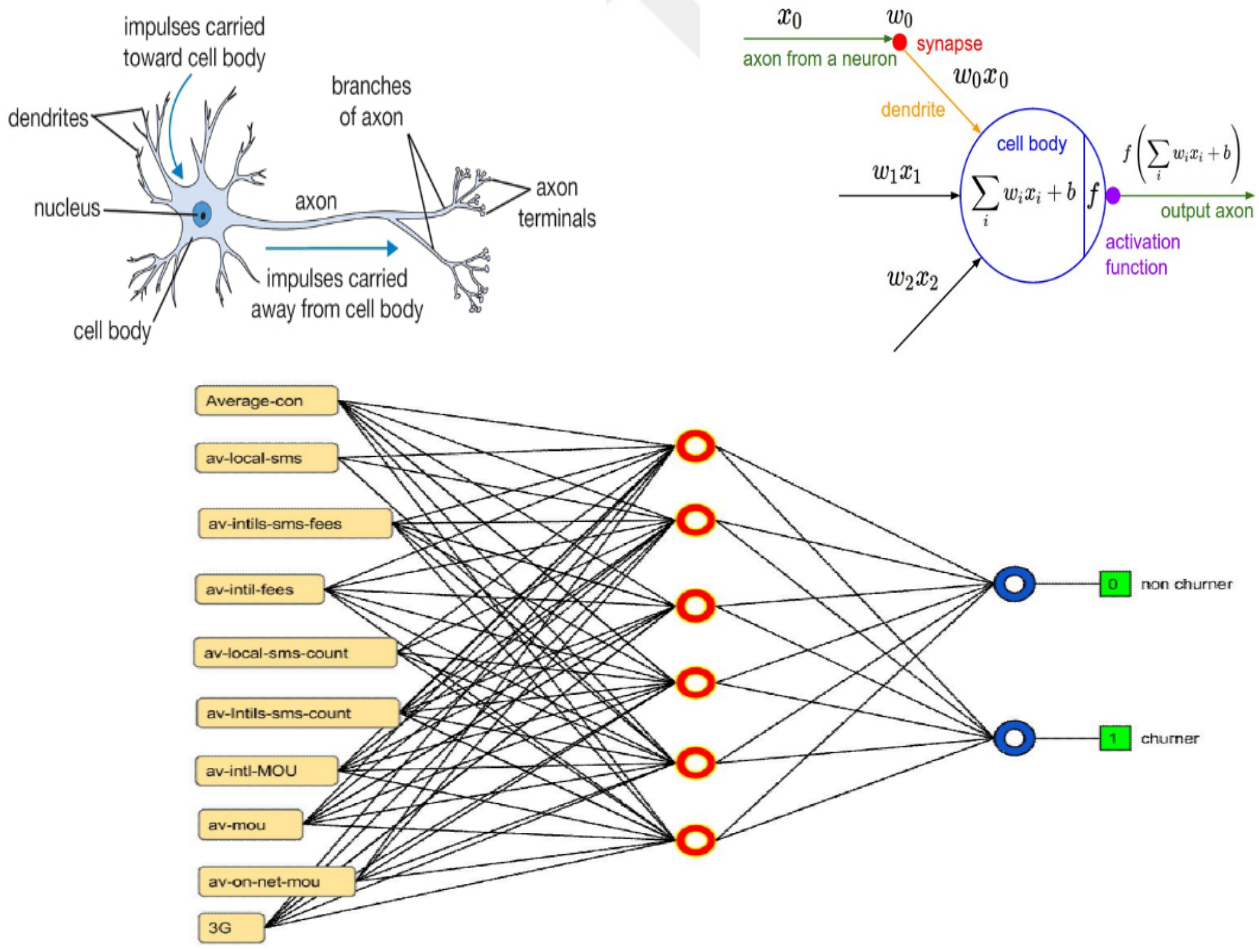
C1 Public

kernels.Because of its strong performance and flexibility, SVM is widely used in areas such as text classification, face or image recognition, and biological data analysis.

3.8.1.7 Artificial Neural Network

Artificial Neural Networks are machine learning models designed to mimic how the human brain processes information. These models are built from individual units called neurons, which are connected to each other in a layered structure. Typically, an ANN consists of an input layer that receives the data, one or more hidden layers that process the information, and an output layer that produces the final result. Each connection between neurons has a weight that controls how much influence the input has on the output. During training, the model adjusts these weights by comparing its predictions to the actual results, gradually learning to make better predictions or classifications.

ANNs are widely applied in tasks like image and voice recognition, natural language understanding, and even financial forecasting. When these networks include many hidden layers and handle large datasets, they are referred to as deep neural networks, which form the foundation of deep learning techniques.



C1 Public

Figure 18: Artificial neural network (ANN) structure with three inputs, one output and multi hidden nodes

3.8.1.8 Extra Trees (Extremely Randomized Trees)

Similar to Random Forest but introduces more randomness in feature selection and split thresholds, which can reduce overfitting and improve speed.

3.8.1.9 K-Nearest Neighbors (KNN)

k Nearest Neighbors (kNN) is a machine learning algorithm used for both classification and regression tasks. It is a non-parametric method that makes predictions based on the similarity of data points. The distance between instances is typically calculated using the Euclidean distance metric. It can be sensitive to the choice of k, as too small values can result in overfitting, while too large values can lead to underfitting. It is also sensitive to the scale of the features, so normalization or standardization may be required. Furthermore, the computational complexity of kNN grows with the size of the training dataset, making it less suitable for large datasets.



Figure 19: k Nearest Neighbors

Here's a summary of the kNN algorithm:

C1 Public

- The k-NN algorithm is a simple yet effective method used for both classification and regression tasks. Unlike many other models, it does not require a training phase in the traditional sense. Instead, it stores the entire training dataset, including the feature values and their corresponding labels.
- When a prediction is needed for a new data point, the algorithm measures the distance between this point and every point in the training set. The most commonly used distance metric is Euclidean distance, although other metrics can also be applied depending on the data.
- After calculating the distances, the algorithm selects the **k** closest data points—where **k** is a user-defined parameter—and examines their labels. For classification problems, the final class is chosen by majority vote among these neighbors. The class that appears most frequently is assigned to the new instance.
- This approach is especially useful for problems where the decision boundary is irregular or complex, but it can be sensitive to the choice of **k** and to the scaling of features.

3.8.1.10 Ridge Classifier

Ridge Classifier is a linear classification algorithm that applies L2 regularization to prevent overfitting by penalizing large coefficient values. It is particularly effective when multicollinearity exists among features, offering better generalization than basic logistic regression. It minimizes the squared loss function with a penalty term proportional to the square of the magnitude of the coefficients.

3.8.1.11 Passive Aggressive Classifier

Passive Aggressive Classifier is an online learning algorithm designed for large-scale learning tasks such as real-time classification or streaming data. It operates by remaining passive when a prediction is correct, and updating the model aggressively only when a misclassification occurs. This makes it ideal for dynamic datasets and incremental training scenarios.

3.8.1.12 XGBoost

XGBoost is a scalable and highly efficient gradient boosting algorithm optimized for speed and performance. It uses tree-based models and incorporates regularization techniques (L1 and L2), pruning, and parallel processing. XGBoost is known for its superior performance in structured/tabular datasets and is widely used in competitions and production environments.

3.8.1.13 LightGBM

C1 Public

LightGBM is a gradient boosting framework developed by Microsoft that builds trees leaf-wise instead of level-wise, resulting in faster training and higher efficiency. It handles large datasets and high-dimensional features efficiently and supports both categorical and continuous features natively. LightGBM is well-suited for real-time prediction and large-scale learning tasks.

3.8.1.14 CatBoost

CatBoost is a gradient boosting algorithm developed by Yandex, particularly optimized for categorical feature handling without the need for extensive preprocessing. It uses ordered boosting and oblivious decision trees, offering high accuracy and resistance to overfitting. CatBoost performs well in both classification and regression tasks and is designed to work out-of-the-box with minimal parameter tuning.

3.8.1.15 LSTM

Long Short-Term Memory (LSTM) is a special type of recurrent neural network (RNN) architecture commonly used in deep learning applications. It is particularly effective at learning long-term dependencies within data sequences, which makes it suitable for tasks involving sequential prediction.Unlike standard neural networks that handle inputs independently, LSTM networks use feedback connections, allowing them to consider previous inputs in the sequence when making decisions. This characteristic enables LSTM to work well with data types such as time series, natural language, and audio signals.

LSTM has played a key role in many deep learning advances by enabling models to extract meaningful patterns and insights from sequential data across various fields. The architecture of an LSTM cell consists of three core gates—forget gate, input gate, and output gate—each playing a distinct role in regulating the flow of information. The forget gate determines which parts of the past information should be discarded, the input gate decides which new information should be added to the memory, and the output gate controls which information is passed forward as output. This architecture enables LSTM to maintain a cell state, which acts as a long-term memory component, and a hidden state, which reflects the short-term dynamics of the sequence.By explicitly modeling temporal dependencies and controlling memory through these gates, LSTM networks are particularly well-suited for tasks such as time series forecasting, sequential event prediction, natural language processing, and churn modeling. In churn prediction, for example, customer behavior patterns unfold over time. LSTM models can capture such temporal dynamics more effectively than static classifiers, making them valuable in identifying early signals of dissatisfaction or disengagement. In this

C1 Public

study, LSTM is used as a time-aware modeling approach to address the limitations of traditional classification algorithms, which fail to incorporate the sequential structure of customer experience and network performance data. By leveraging LSTM, we aim to better detect complex, time-evolving patterns that precede churn events. [12]

3.8.1.15.1 SEQUENCE-TO-ONE LSTM FOR TIME SERIES FORECASTING

The sequence-to-one architecture in LSTM models is particularly well-suited for time series prediction tasks. In this setup, a fixed-length sequence of past time steps is used as input to the model, which outputs a single prediction representing the next time step or a target event.This structure enables the model to capture temporal dependencies and trends within the input window, making it ideal for forecasting scenarios such as predicting the next day's temperature, estimating future network performance, or forecasting customer churn likelihood based on recent behavioral patterns.

3.8.1.15.2 SEQUENCE-TO-SEQUENCE LSTM FOR TIME-DEPENDENT PREDICTIONS

The sequence-to-sequence (Seq2Seq) architecture in LSTM models is designed to map an input sequence to an output sequence of potentially different length. This makes it highly effective for tasks requiring multi-step forecasting or translation of sequential patterns, such as predicting multiple future values in time series, machine translation, or customer behavior progression over several time intervals.By encoding the input sequence into a context-aware representation and then decoding it into a target sequence, Seq2Seq models capture both short- and long-term temporal relationships. This architecture is particularly useful when forecasting multiple future time steps is required, such as anticipating customer churn probability over the next several months or estimating long-term network performance trends.

C1 Public

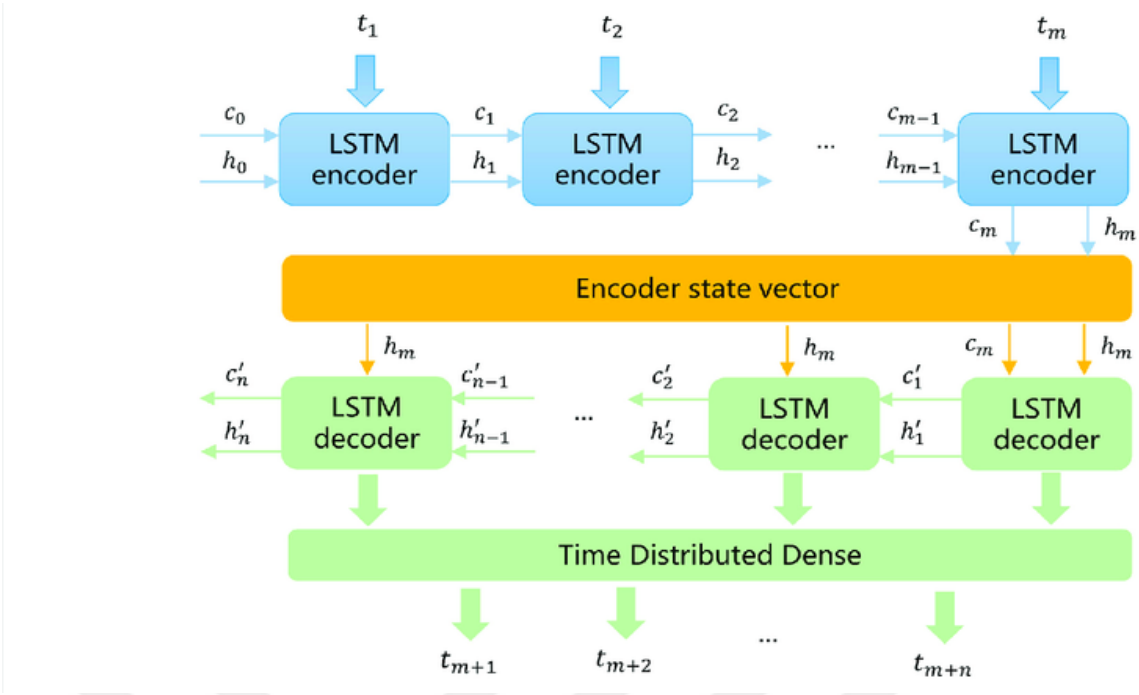


Figure 20 : Seq2Seq-LSTM Architecture

3.8.2 Evaluation Methodology

All models were evaluated using Grid Search Cross-Validation, where various hyperparameters were tuned to find the best configuration for each model. The performance was assessed based on the confusion matrix metrics, specifically:

• ***Accuracy:*** The proportion of total correct predictions. While useful, it is not reliable for imbalanced datasets.

• ***Precision:*** The ratio of correctly predicted churners to all predicted churners. High precision means fewer false positives (non-churners incorrectly flagged as churners).

• ***Recall (Sensitivity):*** The ratio of correctly predicted churners to all actual churners. High recall is essential for capturing the maximum number of at-risk customers, which is the primary business goal.

• ***F1-Score:*** The harmonic mean of precision and recall. It provides a balance when both false positives and false negatives matter.

C1 Public

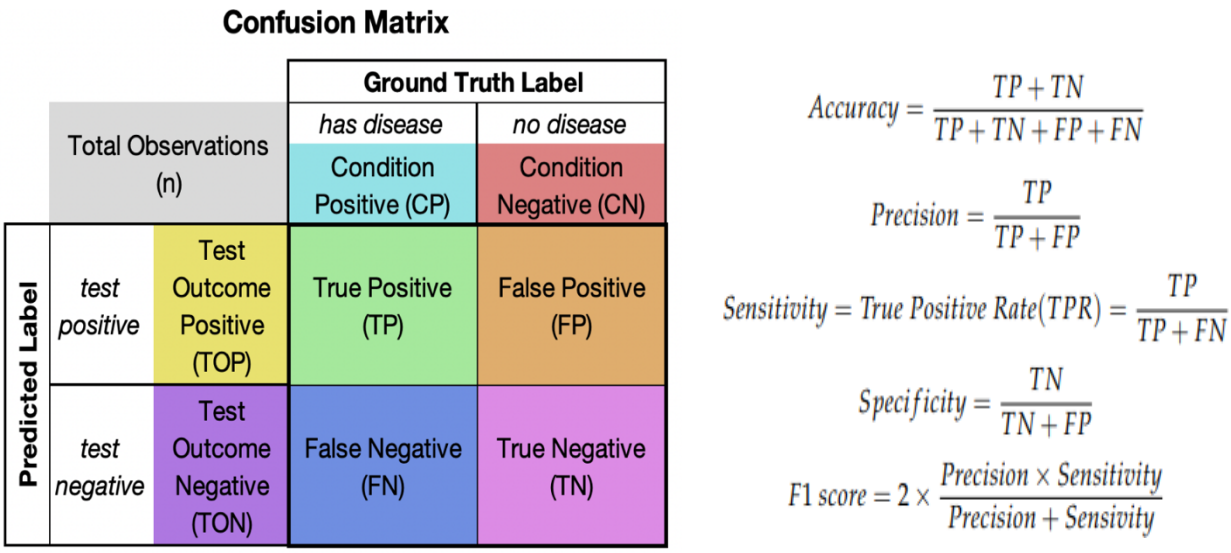


Figure 21: Confusion Matrix and Their Formulas

To evaluate model performance, I utilized the confusion matrix, along with precision, recall, and their harmonic mean, the F1-score. I did not rely on accuracy or similar overall metrics, as the task was not focused on predicting the majority class (non-churners), but rather on identifying the minority class of churners.

In this context:

- **Precision** was critical because a low precision would result in taking unnecessary and potentially costly actions on customers who are not likely to churn. This would negatively impact business efficiency and resources, as a result of this increase of costs.
- **Recall** was important to assess how many actual churners the model was able to identify.
- **F1-score**, which balances both precision and recall, served as the primary metric to select the best model. A higher F1-score indicated that the model successfully minimized both false positives and false negatives, aligning with the objective of accurately targeting true churners without overreacting to false alarms.

Therefore, the model with the highest F1-score was considered the most aligned with the business goal of cost-effective and accurate churn prediction.

3.8.3 Churn Prediction Using Static Data (Dataset 1)

A comprehensive classification pipeline was implemented using a range of machine learning models including Logistic Regression, Random Forest, XGBoost, LightGBM and neural networks. All models were fine-tuned via grid search with 3-fold cross-validation using undersampling for class imbalance and threshold tuning (0.10–0.85) to optimize the F1-score. Despite extensive efforts no model consistently achieved balanced performance across

C1 Public

precision and recall metrics (see Table XI). As shown in Figure 23 lower thresholds (≤0.3) led to high recall but extremely low precision while higher thresholds slightly improved precision at the cost of recall. Overall F1-scores remained low across all thresholds underscoring the limitations of static models and the need for time-aware approaches like LSTM.

Although Figure 23 illustrates the threshold-performance trade-off for only one of the models, a similar trend was observed across all evaluated classifiers. For each model, thresholds ranging from 0.10 to 0.85 were systematically tested, and the configuration yielding the highest F1-score was selected. Thus, below figure serves as a representative example of the general behavior observed, highlighting the challenges in achieving a favorable balance between precision and recall in static classification models.

The corresponding implementation for this classification pipeline can be found in the code block below.

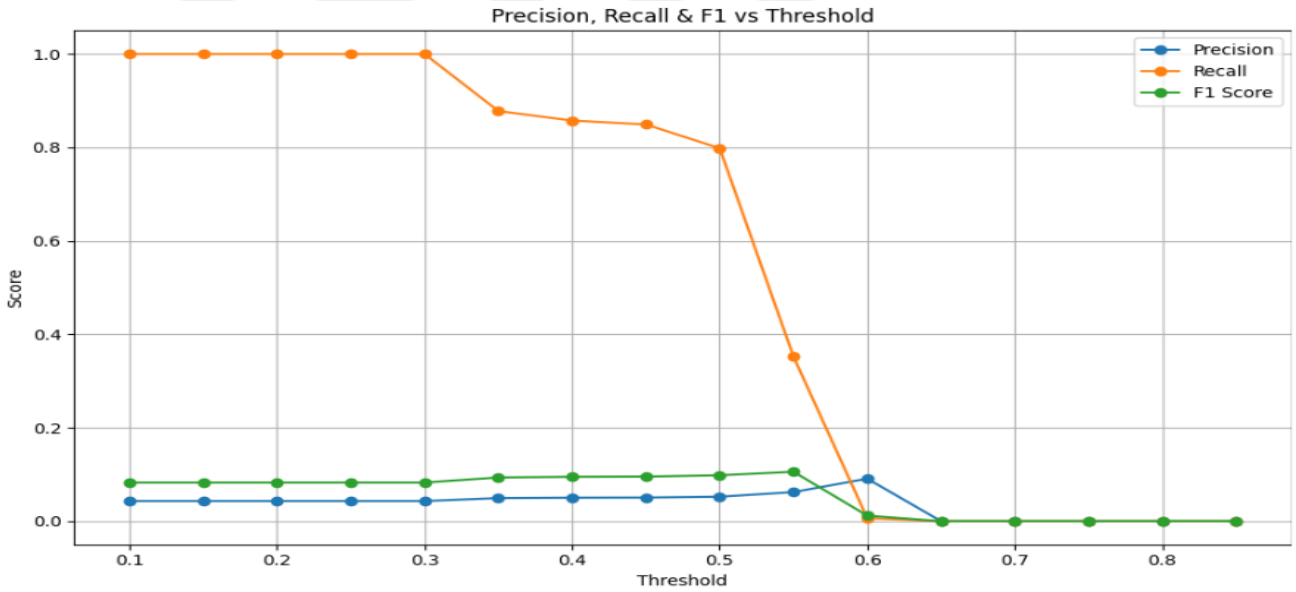


Figure 22 : Churn Risk Labelling via Probability Thresholding

C1 Public

```
param_grids = {
    "Logistic Regression": {
        'C': [0.01, 0.1, 1.0],
        'penalty': ['l2'],
        'solver': ['liblinear', 'lbfgs']
},
    "Decision Tree": {
        'max_depth': [3, 5, 10, 15],
        'min_samples_split': [2, 5, 10],
        'criterion': ['gini', 'entropy']
    },
    "Random Forest": {
        'n_estimators': [50, 100],
        'max_depth': [5, 10, 15],
        'max_features': ['auto', 'sqrt'],
        'min_samples_split': [2, 5]
    },
    "Gradient Boosting": {
        'n_estimators': [50, 100],
        'learning_rate': [0.05, 0.1],
        'max_depth': [3, 5, 7]
    },
    "SVM": {
        'C': [0.1, 1, 10],
        'kernel': ['linear', 'rbf'],
        'gamma': ['scale', 'auto']
    },
    "Naive Bayes": {
        'var_smoothing': [1e-9, 1e-8, 1e-7]
    },
    "MLP": {
        'hidden_layer_sizes': [(64,), (128,), (128, 64)],
        'alpha': [0.0001, 0.001],
        'activation': ['relu', 'tanh'],
        'learning_rate': ['constant', 'adaptive'],
        'solver': ['adam', 'sgd'],
        'learning_rate_init': [0.001, 0.01]
    },
    "Extra Trees": {
        'n_estimators': [50, 100],
        'max_depth': [None, 10]
    },
    "KNN": {
        'n_neighbors': [3, 5, 7],
        'weights': ['uniform', 'distance']
    },
    "Ridge Classifier": {
        'alpha': [0.1, 1.0, 10.0]
    },
    "Passive Aggressive": {
        'C': [0.01, 0.1, 1.0]
    },
    "XGBoost": {
        'n_estimators': [50, 100],
        'max_depth': [3, 5],
        'learning_rate': [0.05, 0.1]
    },
    "LightGBM": {
        'n_estimators': [50, 100],
        'max_depth': [-1, 5],
        'learning_rate': [0.05, 0.1]
    },
    "CatBoost": {
        'iterations': [100, 200],
```

C1 Public

Code Section 12:Evaluating ML Models for Churn Detection on Dataset 1 -Part 1

```
import matplotlib.pyplot as plt
from sklearn.metrics import precision_score, recall_score, f1_score

def find_best_threshold(y_true, y_pred_proba, plot=True):
    thresholds = [i / 100 for i in range(10, 90, 5)]
    print("thresholds :" ,thresholds)
    precisions, recalls, f1s = [], [], []
    best = {'threshold': None, 'precision': 0, 'recall': 0, 'f1': 0}

    for t in thresholds:
        y_pred = (y_pred_proba >= t).astype(int)
        prec = precision_score(y_true, y_pred)
        rec = recall_score(y_true, y_pred)
        f1 = f1_score(y_true, y_pred)
        precisions.append(prec)
        recalls.append(rec)
        f1s.append(f1)

        if prec >= 0.5 and rec >= 0.5 and f1 > best['f1']:
            best.update({
                'threshold': t,
                'precision': prec,
                'recall': rec,
                'f1': f1})
    if plot:
        plt.figure(figsize=(10, 6))
        plt.plot(thresholds, precisions, label='Precision', marker='o')
        plt.plot(thresholds, recalls, label='Recall', marker='o')
        plt.plot(thresholds, f1s, label='F1 Score', marker='o')

        if best["threshold"] is not None:
            plt.axvline(best['threshold'], color='gray', linestyle='--',
label=f"Best Th={best['threshold']}")
        plt.xlabel('Threshold')
        plt.ylabel('Score')
        plt.title('Precision, Recall & F1 vs Threshold')
        plt.legend()
        plt.grid(True)
        plt.tight_layout()
        plt.show()
    return best

 models = {
    "Gradient Boosting": GradientBoostingClassifier(),
    "Naive Bayes": GaussianNB(),
    "SVM": SVC(probability=True),
    "MLP": MLPClassifier(max_iter=300),
    "Extra Trees": ExtraTreesClassifier(),
    "KNN": KNeighborsClassifier(),
    "Ridge Classifier": RidgeClassifier(),
    "Passive Aggressive": PassiveAggressiveClassifier(),
    "XGBoost": XGBClassifier(use_label_encoder=False, eval_metric='logloss'),
    "LightGBM": LGBMClassifier(),
    "CatBoost": CatBoostClassifier(verbose=0) }
```

Code Section 13: Evaluating ML Models for Churn Detection on Dataset 1 -Part 2

C1 Public

```
results = []
for name, model in models.items():
    print(f"\n▶️ Model: {name}")
    if name in param_grids:
        grid = GridSearchCV(model, param_grids[name], cv=2, scoring='f1',
n_jobs=-1, verbose=1)
        grid.fit(X_train_bal, y_train_bal)
        best_model = grid.best_estimator_
    else:
        model.fit(X_train_bal, y_train_bal)
        best_model = model
    if hasattr(best_model, "predict_proba"):
        y_pred_proba = best_model.predict_proba(X_test)[:, 1]
        threshold_result = find_best_threshold(y_test, y_pred_proba)
        if threshold_result['threshold'] is not None:
        y_pred >= (y_pred_proba  threshold_result['threshold']).astype(int)
            precision = threshold_result['precision']
            recall = threshold_result['recall']
            f1 = threshold_result['f1']
        else:
            # fallback 0.5 eşiği
            y_pred = (y_pred_proba >= 0.5).astype(int)
            precision = precision_score(y_test, y_pred)
            recall = recall_score(y_test, y_pred)
            f1 = f1_score(y_test, y_pred)
    else:
        # predict_proba yoksa klasik prediction
        y_pred = best_model.predict(X_test)
        precision = precision_score(y_test, y_pred)
        recall = recall_score(y_test, y_pred)
        f1 = f1_score(y_test, y_pred)

    print(f"Precision: {precision:.4f} | Recall: {recall:.4f} | F1:
{f1:.4f}")
    print(classification_report(y_test, y_pred, zero_division=0))
    # Confusion matrix
    cm = confusion_matrix(y_test, y_pred)
    disp = ConfusionMatrixDisplay(confusion_matrix=cm)
    disp.plot(cmap="Blues")
    plt.title(f"{name} - Confusion Matrix")
    plt.grid(False)
    plt.show()
    results.append({ "Model": name,"Precision": precision, "Recall":
recall,"F1": f1})
results_df = pd.DataFrame(results)
results_df.sort_values(by="F1",ascending=False).reset_index(drop=True)
print("\\n Tüm Model Sonuçları (F1'e göre sıralı):"
```

Code Section 14: Evaluating ML Models for Churn Detection on Dataset 1 -Part 3

C1 Public

3.8.4 Time Series Prediction Using LSTM with Dataset 2

Initial experiments with both traditional machine learning models and non-sequential neural networks on Dataset 1 did not yield satisfactory precision and recall. In this study, two different LSTM architectures—sequence-to-one and sequence-to-sequence—could have been considered for churn prediction. However, the sequence-to-one structure was preferred due to the nature of the available dataset, which included time-ordered snapshots of customer behavior and network performance but required the prediction of a single churn label per customer. This structure aligns better with the goal of modeling temporal patterns to predict a final churn outcome. To improve performance a time series approach using an sequence-to-one LSTM (Long Short-Term Memory) network was implemented [6] which better captured temporal patterns in customer behavior.The model was structured as a many-to-one architecture where 30-day sequences of customer KPI data were fed into the network to predict a single churn label indicating whether the customer would churn in the near future. This design enabled the model to effectively learn temporal dependencies from historical performance trends. Key methods applied to improve learning included:

- Focal Loss to handle class imbalance by down-weighting well-classified examples and focusing the model on hard-to-classify churners [5]
- Dropout layers to reduce overfitting
- EarlyStopping and ReduceLROnPlateau for efficient training control
- Sigmoid activation to produce churn probability scores
- Adam optimizer with a learning rate of 0.0001.

Customers with predicted churn probability above 0.5 were flagged as churn risks. The dataset was split into 80% training and 20% testing without shuffling preserving time dependency. This sequential learning framework resulted in significantly improved F1-score with higher precision and recall compared to previous approaches (Table X). The classification performance of the model is presented in the confusion matrix and classification report below where key metrics such as precision recall and F1-score are highlighted as discussed in previous studies [7]-[11]. **The following section presents both the best-performing and relatively suboptimal LSTM model results.** While the latter exhibits comparatively lower performance in terms of evaluation metrics, it demonstrates a more balanced distribution of predicted class samples. This contrast provides valuable insight into the trade-offs between metric optimization and class representation fidelity, highlighting the importance of evaluating both performance scores and prediction distributions in churn modeling tasks.

C1 Public

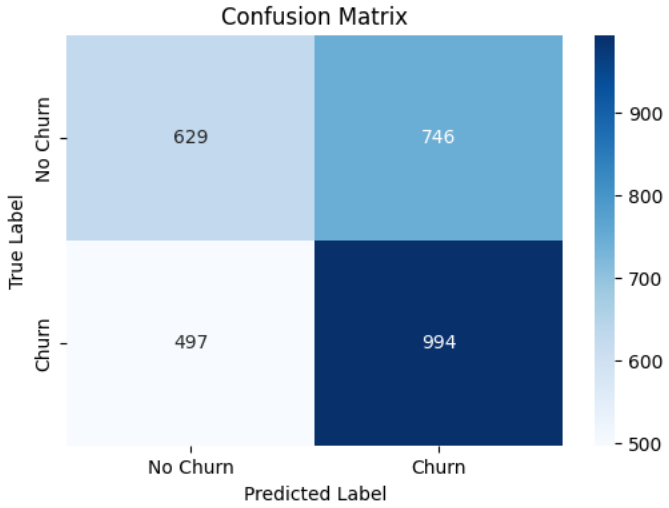
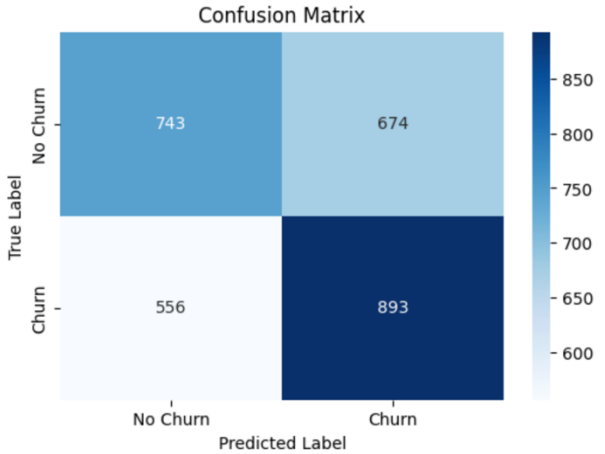


Figure 23 : LSTM Confusion Matrix-1

Table X: LSTM CLASSIFICATION REPORT-1

	Precision	Recall	F1-score	Support
0	0.56	0.46	0.50	1375
1	0.57	0.67	0.62	1491
accuracy			0.57	2866
macro avg	0.40	0.49	0.38	2866
weighted avg	0.43	0.57	0.44	2866

The corresponding implementation for this classification pipeline can be found in the code block below.



	Precision	Recall	F1-score	Support
0	0.57	0.52	0.55	1417
1	0.57	0.62	0.59	1449
accuracy			0.57	2866
macro avg	0.57	0.54	0.57	2866
weighted avg	0.57	0.56	0.57	2866

Figure 24: LSTM Confusion Report and Matrix-2

C1 Public

```
import numpy as np
import pandas as pd
import tensorflow as tf
import matplotlib.pyplot as plt
import seaborn as sns
from tensorflow.keras import backend as K
from tensorflow.keras.metrics import  Precision, Recall ,AUC
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.layers import LSTM, Dense,Dropout
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report
from keras.callbacks import EarlyStopping, ReduceLROnPlateau
import shap

anova_features =
["cem_mo_call_setup_dur_scaled","cem_ul_silent_call_ratio","cem_ul_mos_scal
ed","disco_avg_uldmrssinr_scaled","disco_avg_rsrq_scaled","cem_tput_kbps_sc
aled","cem_xkb_delay_scaled","cem_avg_30d_cem_e2e_delay_overall_scaled","di
sco_avg_dlmcs_scaled","disco_avg_uetxpower_scaled","main_rate_4g_volume_sca
led","cem_dl_mos_scaled","cem_1mb_3mb_ratio","cem_gt_8mb_ratio","cem_mo_cal
l_setup_succ_rate","disco_avg_rsrp_scaled","cem_3mb_5mb_ratio",
"cem_dl_silent_call_ratio","main_num_call_drop_total_scaled","cem_mo_call_d
rop_rate","cem_mt_call_setup_succ_rate","cem_ul_volume_gb_scaled","cem_dl_v
olume_gb_scaled"]

df_3k = df_3k.sort_values(by=["gsm_no", "pdate"])
df_3k =df_3k.reset_index(drop=True)

def focal_loss(gamma=0.5, alpha=0.25):
    def focal_loss_fixed(y_true, y_pred):
        epsilon = K.epsilon()
        y_pred = K.clip(y_pred, epsilon, 1. - epsilon)
        pt = tf.where(K.equal(y_true, 1), y_pred, 1 - y_pred)
        loss = -alpha * K.pow(1. - pt, gamma) * K.log(pt)
        return K.mean(loss)
    return focal_loss_fixed
X, y = [], []
time_steps=30
 for gsm_no, customer_df in df_3k.groupby("gsm_no"):
    customer_df = customer_df.sort_values("pdate") # Tarihe göre sırala
    if len(customer_df) <30:
        continue
    features_values = customer_df[anova_features].values
    labels = customer_df["churn_flg"].values
    for i in range(len(features_values) - time_steps):
        X.append(features_values[i : i + time_steps]) # Son 30 gün
        y.append(labels[i + time_steps-1])

X, y = np.array(X), np.array(y)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
shuffle=False)
tf.config.set_visible_devices([], 'GPU')
```

C1 Public

```
model = Sequential([
    LSTM(128,
activation='relu',return_sequences=True,input_shape=(time_steps,
X_train.shape[2])),
    Dropout(0.3),
    LSTM(64),
    Dropout(0.3),
    Dense(32,activation='relu'),
    Dense(1, activation='sigmoid') # Binary classification için sigmoid
aktivasyon
    ])

optimizer = Adam(learning_rate=0.0001)
model.compile(optimizer=optimizer, loss=focal_loss(gamma=2.,alpha=0.25),
metrics=['accuracy',Precision(),Recall(),AUC()])

early_stop = EarlyStopping(monitor='val_loss', patience=6,
restore_best_weights=True)
reduce_lr = ReduceLROnPlateau(monitor='val_loss',factor=0.5,patience=3)

# Model Training
model.fit(X_train, y_train,
          epochs=50,
          batch_size=64,
          validation_data=(X_test, y_test),
          callbacks=[early_stop,reduce_lr])

# Prediction
y_pred = model.predict(X_test)
y_pred = (y_pred > 0.5).astype(int) # Sigmoid çıktısını binary (0/1) yap
# Confusion Matrix
conf_matrix = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(6, 4))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues",
xticklabels=["No Churn", "Churn"], yticklabels=["No Churn", "Churn"])
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()

#Classification Report (Precision, Recall, F1-Score)
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

Code Section 16: LSTM Model for Churn Prediction Dataset 2 -Part 2

C1 Public

4 EXPERIMENTAL RESULTS

All non-sequential models trained on Dataset 1 (static KPIs) were optimized using Grid Search with wide parameter grids and evaluated across different probability thresholds. Despite this most models achieved high recall but very low precision resulting in poor F1-scores (~0.10). This imbalance limits their practical use due to excessive false positives and unnecessary customer interventions.

In contrast the LSTM model trained on Dataset 2 (30-day time-series KPI sequences) achieved significantly better performance with 0.57 precision 0.62 recall and an F1-score of 0.59 on diagonal confusion matrix. This success is attributed to its ability to capture time-based patterns in user experience using a many-to-one sequential architecture enhanced with Focal Loss, Dropout , EarlyStopping and Adam optimizer.

To further refine churn targeting and reduce the volume of unnecessary actions caused by low precision K-Means and Bisecting K-Means clustering were applied to Dataset 1. The objective was to identify customer segments with degraded network performance based on median KPI values. It was observed that these poor-performance segments exhibited slightly higher churn ratios. By narrowing down focus to this critical group and further identifying users who exceed churn-driving KPI thresholds (based on WoE analysis) the strategy enables low-cost high-precision intervention. Moreover this approach supports bulk root cause diagnosis by revealing network-related issues concentrated in specific cities districts or device types. The model comparisons are presented in the TABLE XI.

Table XI : Model Comparisons

Model	Dataset	ML Library	Precision	Recall	F1-Score
LSTM	Dataset 2	Tensorflow	0.57	0.67	0.62
LSTM	Dataset 2	Tensorflow	0.57	0.62	0.59
Random Forest	Dataset 1	Scikit-learn	0.05	0.80	0.10
Extra Trees	Dataset 1	Scikit-learn	0.05	0.77	0.10
XGBoost	Dataset 1	Scikit-learn	0.05	0.75	0.10
CatBoost	Dataset 1	Scikit-learn	0.05	0.74	0.10
LightGBM	Dataset 1	Scikit-learn	0.05	0.74	0.10
Decision Tree	Dataset 1	Scikit-learn	0.05	0.71	0.10
Gradient Boosting	Dataset 1	PySpark MLlib	0.05	0.67	0.10
Multi-Layer Neural Network	Dataset 1	PySpark MLlib	0.05	0.64	0.09
Support Vector Machine (SVM)	Dataset 1	PySpark MLlib	0.05	0.63	0.09
Logistic Regression	Dataset 1	Scikit-learn	0.06	0.50	0.10
Ridge Classifier	Dataset 1	Scikit-learn	0.06	0.50	0.10

C1 Public

Naive Bayes	Dataset 1	Scikit-learn	0.05	0.36	0.09
Passive Aggressive Classifier	Dataset 1	Scikit-learn	0.06	0.08	0.07

5 CONCLUSION AND FUTURE WORK

This study shows that customer churn can be effectively predicted using only network-level KPIs, without relying on economic data such as pricing or tariff plans. ANOVA was applied for feature selection to identify significant KPIs before modeling. Among all models, LSTM trained on sequential KPI data achieved the best precision–recall balance, highlighting the value of temporal patterns in network experience.

Due to low precision, taking action on all predicted churners would result in unnecessary operational efforts. To mitigate this, clustering was applied after prediction to segment customers based on network performance. Within the worst-performing group, customers exceeding one or more **critical KPI thresholds (as determined by WoE analysis)** were identified.

This enables proactive network-level solutions to be designed, such as fixing network issues that impact the worst user groups. In addition, it allows detection of problematic devices (e.g., smartphones, tablets) or tariff plans that contribute to poor customer experience. Moreover, by reversing the same filtering process, **technically satisfied users with high upsell potential** can also be identified.

6 APPENDICE



Appendix A

C1 Public

References

[1] R. A. Alotaibi A. Almalki A. Gumaei and M. M. Hassan "Deep churn prediction method for telecommunication industry " Sustainability vol. 15 no. 5 p. 4543 2023. [Online]. Available: https://doi.org/10.3390/su15054543

[2] L. Saha H. K. Tripathy T. Gaber H. El-Gohary and E. M. El-kenawy "Deep Churn Prediction Method for Telecommunication Industry " in *Social Media Marketing and Sustainability: An Endeavor to Discover the Future* Mar. 2023.

[3] S. W. Fujo S. Subramanian and M. A. Khder "Customer Churn Prediction in Telecommunication Industry Using Deep Learning " *Department of Computer Science Applied Science University Bahrain* Jan. 2022.

[4] N. Siddiqi *Credit Risk Scorecards: Developing and Implementing Intelligent Credit Scoring* Wiley 2006.

[5] T.-Y. Lin P. Goyal R. Girshick K. He and P. Dollár "Focal Loss for Dense Object Detection " *Proc. of IEEE ICCV* pp. 2980-2988 2017.

[6] S. Hochreiter and J. Schmidhuber "Long Short-Term Memory " *Neural Computation* vol. 9 no. 8 pp. 1735–1780 1997.

[7] T. Saito and M. Rehmsmeier "The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets " *PLOS ONE* vol. 10 no. 3 pp. 1–21 Mar. 2015. doi: 10.1371/journal.pone.0118432.

[8] T. Fawcett "An introduction to ROC analysis " *Pattern Recognition Letters* vol. 27 no. 8 pp. 861–874 Jun. 2006. doi: 10.1016/j.patrec.2005.10.010.

[9] Google Cloud AI "Predicting customer churn with AI " *Google Cloud Blog* 2020. [Online]. Available: https://cloud.google.com/blog/products/ai-machine-learning/predicting-customer-churn-with-ai

[10] IBM Developer "Precision vs. Recall: Picking the right metric for your machine learning model " *IBM Developer Blog* 2018. [Online]. Available: https://developer.ibm.com/articles/precision-vs-recall/

[11] McKinsey & Company "Reducing customer churn with AI and predictive analytics " *Telecommunications Practice* 2020. [Online]. Available: https://www.mckinsey.com/capabilities/quantumblack/our-insights/reducing-customer-churn-with-a

[12] A. Garg, "Introduction to Long Short-Term Memory (LSTM)," Analytics Vidhya, Mar. 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/

[13] R. Blattberg and J. Deighton, "Manage marketing by the customer equity test," Harvard Business Review, vol. 74, no. 4, pp. 136–144, Jul.–Aug. 1996.

[14] F. Reichheld and W. E. Sasser, "Zero defections: Quality comes to services," Harvard Business Review, vol. 68, no. 5, pp. 105–111, Sep.–Oct. 1990.

[15] S. Ahn, D. Han, and H. Lee, "Customer churn analysis: Churn determinants and mediation effects of partial defection in the Korean mobile telecommunications service industry," Telecommunications Policy, vol. 40, no. 3, pp. 223–236, Mar. 2016.

C1 Public

[16] 3GPP TS 36.331, Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Resource Control (RRC); Protocol Specification, v17.7.0, Dec. 2023.

[17] 3GPP TS 36.213, Evolved Universal Terrestrial Radio Access (E-UTRA); Physical Layer Procedures, v17.7.0, Dec. 2023.

[18] ITU-T Rec. P.863, Perceptual Objective Listening Quality Assessment, 2021.

[19] ITU-T Rec. P.862, Perceptual Evaluation of Speech Quality (PESQ), 2018.

C1 Public

CURRICULUM VITAE

I work as a Data Scientist with a strong background in telecommunications specializing in customer experience management service quality and network performance systems. I have hands-on experience in building machine learning models automating data workflows and developing AI-driven solutions particularly in churn analysis anomaly detection and lifecycle management. As a Generative AI Adoption Leader I define strategic AI use cases and lead the deployment of enterprise-grade GenAI applications. I successfully launched VINA a Generative AI-powered network assistant and Recomate a customer complaint handling and ticket recommendation engine. My technical expertise includes large-scale data processing in Hadoop environments with tools such as Airflow Spark Python and SQL. I hold a Master s degree in Electrical and Electronics Engineering and am a certified Google Cloud Digital Leader.

C1 Public