Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

End-to-End Machine Learning Data Pipeline for Telecom Customer Churn Prediction

Dr. Sunil Bhutada
Head of the Department of Information Technology, Sreenidhi Institute of Science and Technology, Hyderabad

T. Nikhil
Department of Information Technology, UG Student, Sreenidhi Institute of Science and Technology, Hyderabad.

V. Tanmay
Department of Information Technology, UG Student, Sreenidhi Institute of Science and Technology, Hyderabad.

B. Venkat Siddhant,
Department of Information Technology, UG Student,
Sreenidhi Institute of Science and Technology, Hyderabad.

Dr. K Srinivasa Reddy
Associate Professor Department of Information Technology, Sreenidhi Institute of Science and Technology, Hyderabad.

Abstract - Predictive analytics has become a cornerstone of modern telecommunications, particularly in its ability to proactively manage customer churn. By identifying high-risk subscribers in real-time, providers can shift from reactive troubleshooting to strategic retention, significantly reducing revenue loss while simultaneously enhancing long-term customer lifetime value. This project introduces a specialized Customer Intelligence and Risk Optimization Platform—an AI-driven solution designed to be accessible yet technically robust. At its core, the system utilizes a high-performance Extreme Gradient Boosting (XGBoost) algorithm to uncover complex, nonlinear correlations between diverse data points such as customer tenure, billing patterns, and service subscriptions.

The platform is built on a modular, micro-service architecture designed for seamless deployment and scalability. The trained XGBoost model operates as an inference service through a FastAPI RESTful framework, allowing it to process live, structured JSON requests with high efficiency. To ensure the system remains portable and ready for any infrastructure, the entire environment is containerized using Docker. On the front end, users interact with a sophisticated, SaaS-style interface built with Streamlit. This interactive dashboard provides a vivid, real-time look at consumer risk through color-coded classifications (Low, Medium, and High) and animated probability bars, making complex data immediately understandable for stakeholders.

To further bridge the gap between raw data and business action, the platform integrates a Large Language Model (LLM) to enhance interpretability and decision-making. Rather than providing just a numerical score, the system features a conversational AI assistant that generates contextual, "pro-retention" strategies based on specific model results. These intelligent suggestions help stakeholders translate predictive insights into personalized customer outreach. By combining high-performance gradient boosting with conversational AI and real-time visualization, this architecture offers a comprehensive corporate-level solution for managing consumer risk in a dynamic market.

Keywords - ML Pipeline, Customer Churn, DataOps, Random Forest, FastAPI, Streamlit, PostgreSQL, Feature Store, MLOps, Telecom Analytics

I. INTRODUCTION

In the fast-paced world of telecommunications, losing a customer—often called "churn"—isn't just a metric; it's one of the most significant hurdles a business can face. Because it is far more expensive to win over a new subscriber than to keep a loyal one, the ability to spot a "flight risk" early is a game-changer for revenue. With modern flexibility making it easier than ever for people to switch operators, companies are seeing a quarter of their customers leave every year. While we've had the data to understand why this happens for a long time, there has always been a frustrating gap between seeing the problem on a spreadsheet and actually stopping it in real-time.

Bridging that gap is an engineering challenge as much as it is a mathematical one. It's one thing to have a model that works in a lab, but quite another to build a system that pulls raw data from everyday customer records, calculates risks on the fly, and gets that information into the hands of the support teams who can make a difference. This requires more than just accuracy; it requires a seamless bridge between the complex world of machine learning and the practical reality of a busy customer service desk.

This paper introduces a complete, nine-stage system designed to turn those complex predictions into practical action. By combining a robust data pipeline with an intuitive, interactive dashboard and an AI-powered chatbot assistant, we've created a tool that feels less like a black box and more like a helpful partner. Our findings show that by organizing

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

data carefully to avoid technical "skew" and providing clear, business-ready analytics, companies can finally intervene at the exact moment a customer is considering leaving, turning potential losses into long-term loyalty.

II. LITERATURE REVIEW

The evolution of churn prediction began with traditional tools like logistic regression and simple decision trees, which relied heavily on manually selected billing data. However, as the field matured, researchers like Verbeke et al. [1] demonstrated that ensemble models—such as Random Forest and Gradient Boosting—consistently outperform single classifiers, especially when measuring success through the lens of actual business profit. These studies highlighted a critical reality: in the world of customer retention, simple accuracy metrics are often misleading because they don't account for the uneven costs and high stakes of real-life service scenarios.

Beyond the choice of a specific model, the quality of the data itself often dictates success. Research by Vafeiadis et al. [2] on the IBM Telco dataset revealed that high-level feature engineering actually has a greater impact on performance than the algorithm used. Interestingly, Huang et al. [3] found that there is hidden predictive power in "social-network" data—such as call-detail records—that standard customer profiles simply can't capture. By looking at how subscribers interact with one another, we can uncover risks that remain invisible in traditional tabular data.

Transitioning these insights from a research lab to a live production environment brings a new set of challenges, commonly addressed through the MLOps paradigm. As Sculley et al. [4] pointed out, maintaining an AI system involves managing "hidden technical debt," which includes everything from fragile data dependencies to the lack of proper monitoring. To solve this, frameworks like those defined by Zaharia et al. [5] provide a roadmap for a unified machine learning lifecycle. By incorporating sophisticated tools like feature stores and "reverse ETL" phases, this pipeline ensures that the model stays reliable, scalable, and deeply integrated into the business's operational workflow.

III. DATASET DESCRIPTION

The dataset used for this project is a widely recognized benchmark in the industry, originally sourced from IBM and made available through Kaggle. It captures the records of 7,043 customers, providing a rich yet manageable foundation of 21 distinct attributes. This data reflects the real-world complexities of the telecommunications sector, including a natural class imbalance where roughly 26.5% of the entries represent customers who have churned, compared to 73.5% who remained with their provider.

To build a holistic view of the customer experience, the attributes are categorized into four key areas. These include basic demographic markers—such as seniority, partnership status, and dependents—alongside a comprehensive list of subscribed services ranging from internet and phone types to specialized add-ons like tech support and streaming movies. Additionally, the data tracks vital account details, including contract types and billing methods, which are often the strongest indicators of long-term loyalty or potential flight risk.

Before the data can be used for modelling, a light but essential "cleaning" phase is required to ensure accuracy. For instance, we identified a small number of missing values in the "Total Charges" column that must be corrected using statistical averages. Once refined, the data is split into an 80/20 ratio, creating a training set of 5,634 records and a testing set of 1,409. This division is carefully balanced to ensure that both groups accurately represent the same ratio of churned and retained customers, providing a fair and reliable environment for evaluating the platform's performance.

contract types and billing methods, which are often the strongest indicators of long-term loyalty or potential flight risk.

Before the data can be used for modelling, a light but essential "cleaning" phase is required to ensure accuracy. For instance, we identified a small number of missing values in the "Total Charges" column that must be corrected using statistical averages. Once refined, the data is split into an 80/20 ratio, creating a training set of 5,634 records and a testing set of 1,409. This division is carefully balanced to ensure that both groups accurately represent the same ratio of churned and retained customers, providing a fair and reliable environment for evaluating the platform's performance.

IV. SYSTEM ARCHITECTURE & PIPELINE

Different layers of the system architecture are interconnected and are used to ingest data, process it, train a model, and deploy it.

A. Data Ingestion and Streaming Layer

This is because the module of data ingestion is implemented as a FastAPI-based REST endpoint and accepts structured telecom-customer data in CSV or JSON format. The raw data zone is used to store incoming data to preserve data lineage and provide the data replay capability.

Apache Kafka is combined as a streaming platform in order to facilitate real-time processing. The customer events are reported by data producers to Kafka topics, whereas consumers process the event, forwarding it to the next modules. This guarantees fault tolerance, scalability, and low-latency data flow.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

B. Data Preprocessing and Transformation Layer

To prepare our data for modelling, we use a preprocessing module built with Pandas to clean and engineer our features. This stage is essential for transforming raw records into a format that a machine learning algorithm can interpret.
Key operations include:

- Data Integrity: We handle missing values and perform necessary type conversions, such as imputing missing Total Charges to ensure the dataset remains complete.
- Feature Engineering: To address nonlinear churn patterns, we created tenure groups to linearize trends. We also calculate derived features, like average monthly revenue, to capture deeper spending habits.
- Standardization: Categorical variables are converted via one-hot encoding, and churn labels are transformed into a binary format (0 or 1)

These transformations ensure our dataset is coherent and fully optimized for the predictive modelling stage.

C. Analytics and Feature Store Layer

The analytics module provides deep dives into churn distributions across tenure groups and contract types, using correlation analysis to pinpoint customer behaviour patterns. To keep our data consistent, we process and house features in a PostgreSQL feature store managed via pgAdmin 4. This "single source of truth" ensures that the data used during training perfectly matches what the model sees during real-world inference, effectively eliminating feature-mismatch errors.

D. Model Training Layer

We developed the churn prediction model using **XGBoost**, an ensemble learning algorithm favoured for its speed and accuracy. It was specifically chosen for its ability to map complex, nonlinear associations and its use of regularization to prevent overfitting on structured telecom data. Once the training cycle is complete, the model is serialized and stored in a dedicated repository, ready for deployment.

E. Model Serving and API Layer

The model is deployed via **FastAPI**, providing a sleek RESTful interface for real-time predictions. When a request hits the endpoint, the system reconstructs the necessary feature vectors based on the feature store schema to ensure accuracy. The API then returns a clear breakdown: the churn probability score, a risk level (Low, Medium, or High), and tailored retention recommendations, offering a highly scalable serving solution.

F. Logging and Monitoring Layer

Every prediction request is captured in long-term storage using **PostgreSQL** and optional CSV logs. By recording the input features alongside the predicted risk and timestamps, we maintain a robust audit trail. This data is vital for monitoring model performance over time, debugging specific cases, and performing deeper post-hoc analysis.

G. Visualization and User Interface Layer

We built an interactive dashboard using **Streamlit** to put the model's power directly into the hands of users. Through a simple interface of sliders and toggles, team members can input customer data and receive instant feedback. The dashboard visually displays the churn probability, risk classifications, and actionable retention strategies in a clear, easy-to-read format.

H. AI-Based Retention Assistant

To add a layer of transparency, the system includes a custom **Large Language Model (LLM)** assistant. This tool "explains" the model's reasoning by identifying the specific factors driving a customer's churn risk. It then suggests personalized retention strategies, helping managers make more informed, data-driven decisions.

I. Deployment Layer

The entire system is containerized with **Docker**, making it easy to move between different development and production environments. By deploying on **AWS** cloud infrastructure, we ensure the platform is scalable, highly available, and uses resources as efficiently as possible.

V. DATABASE INFRASTRUCTURE

For our production environment, we transitioned to a PostgreSQL 17 instance named telecom dB, managed via pgAdmin 8. This upgrade was essential to handle the multi-client concurrency that would have overwhelmed our initial SQLite development store.

As shown in Figure 7, the database schema is organized into dedicated tables for staging raw customer data, storing processed feature snapshots, and logging metadata for both training runs and prediction events. We also maintain a live audit log of database activity. Monitoring shows clear performance patterns: spikes in transactions-per-second coincide with the heavy lifting of feature-store population, while high read throughput is confirmed by the corresponding surges in "tuples-out" activity during active pipeline runs.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

VI. METHODOLOGY

Our primary classifier is a Random Forest ensemble (n_estimators=200, random_state=42), chosen for four practical reasons: (1) its proven track record as a robust baseline for imbalanced churn data [6]; (2) its feature-scale invariance, which removes the need for data normalization; (3) its built-in feature importance, making the model's decisions easy to explain; and (4) its use of bootstrap aggregation to prevent overfitting.

To ensure a fair test, we compared it against Logistic Regression (L2, C=1.0), Decision Tree (max_depth=10), and Gradient Boosting (100 trees, lr=0.1). All models were trained on the same feature matrices using a stratified 80/20 split. We evaluated performance using five metrics specifically suited for imbalanced classification: Accuracy, Precision, Recall, F1-Score, and ROC-AUC.

VI-B. EXPERIMENTAL SETUP

Built on a modern stack including Python 3.10, scikit-learn 1.3, pandas 2.0, and PostgreSQL 17, the system integrates industry-standard tools like FastAPI, Streamlit 1.25, and NumPy 1.25. For data handling and visualization, it utilizes SQLite3, joblib 1.3, Matplotlib 3.7, and Seaborn 0.12. All experiments were conducted on a standard Intel Core i7 CPU without requiring GPU acceleration. This architecture ensures that the solution remains highly accessible, as it can be deployed and managed on conventional hardware without the need for specialized or expensive compute infrastructure.

TABLE II. Pipeline Stages Summary

Stage	Module	Function
0	extract_api_server.py	FastAPI CSV ingestion endpoint
1	ingest.py	Zone copy to raw zone with lineage
2	transform.py	Coercion, imputation, feature derivation
3	analytics.py	Churn summaries and visualisations
4	feature_store.py	One-hot encode → SQLite persistence
5	train.py	Random Forest fit and joblib serialise
6	serve.py	FastAPI /predict REST endpoint
7	reverse_etl.py	Prediction event audit logging
8	streamlit_app.py	Interactive UI with AI chatbot

TABLE III. Top Engineered & Encoded Feature Groups by Importance

Feature / Group	Type	Rel. Importance
Contract type (encoded)	Categorical	■■■■■ High
Tenure / tenure group	Numeric/Ordinal	■■■■ High
Total charges	Numeric	■■■ Medium
Monthly charges	Numeric	■■■ Medium
avg_monthly_charges	Engineered	■■■ Medium
Internet service type	Categorical	■■ Low-Med
Online security / backup	Binary	■ Low
Payment method	Categorical	■ Low

Table III contains the received outputs of the feature importance ranks that were gained with the help of the trained Random Forest model. Direct majority of distribution of importance is evident in encoding of the contract type and variables of tenure/ tenure group together, as is the case in the findings of the churn analytics in the given Section IX. The designed feature

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

referred to as average monthly charges under the medium-importance category supports the effectiveness of the used feature derivation process at the transformation stage. Attributes representing binary service subscriptions (online security, backup, streaming) have a relatively insignificant contribution to importance; the net contribution that they make to the entire feature space of the one-hot encoded characteristic is, however, huge.

VII. EVALUATION METRICS

When dealing with an imbalance between customers who stay and those who leave, we rely on five key measures to tell the full story of the model's performance. While **Accuracy** gives us a high-level view of how often the model is right overall, it rarely provides enough detail for high-stakes decisions. To truly understand the financial impact, we look closer at **Precision**, which tells us how many of our predicted "churners" actually intended to leave. High precision ensures we aren't wasting retention budgets and marketing efforts on happy customers who never planned on going anywhere.

On the other hand, **Recall** is our safety net; it measures our ability to catch every single customer who is actually at risk. If our recall is low, we are essentially "leaving money on the table" by failing to identify people we could have saved. Since it's often hard to perfect both, the **F1-Score** acts as a vital middle ground. By calculating the harmonic mean of precision and recall, it gives us a single, balanced number that ensures we aren't leaning too far toward one type of error at the expense of the other.

Finally, to ensure our model is truly reliable across the board, we use **ROC-AUC**. This metric is particularly powerful because it measures the system's "discriminative power" regardless of where we set our specific risk thresholds. It proves that the model can effectively tell the difference between a loyal subscriber and a flight risk, even when the groups are unevenly sized. Together, these tools move us beyond simple percentages and into the realm of informed, cost-effective business strategy.

VI-C. ALGORITHM USED

The current system uses Extreme Gradient Boosting (XGBoost), a machine learning supervised algorithm that is based on the gradient boosting algorithm, in forecasting customer churn. The problem formulated is a binary classification, based on which the model may derive whether the feature(s) will predict a churn (1) or not churn (0) of the customer if the input feature or features are used.

XGBoost is also an ensemble-based learning software that is constructed as a sequence of decision trees. All the trees learn to correct the errors made by the previous trees, starting a differentiable error minimization objective through gradient descent optimization. This involves boosting or increasing, and this is an iterative process of increasing the models and their generalization accuracy and performance.

Calculation of the model provides:

$$\hat{y}_t = \sum_{k=1}^{K} f_k(x_i), f_k \in F$$

where:

- $\hat{y}_i$is the predicted value for the i^{th}instance
- $f_k$represents the k^{th}decision tree
- $\mathcal{F}$denotes the space of all regression trees
- $K$is the total number of trees

are represented by individual decision trees and are. F F represents the tree space.

The goal of the XGBoost objective can be summarized as the following:

$$L = \sum_i l(y_i, \hat{y}_i) + \sum_k \Omega(f_k)$$

where:

- $l(y_i, \hat{y}_i)$is the loss function (e.g., logistic loss for classification)
- $\Omega(f_k)$is the regularization term that controls model complexity

is the regulating parameter which governs the complexity of the model. Regularization guarantees the

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

elimination of oversizing and, consequently, XGBoost suits extremely well when the operated data is structured such as the telecom customer data.

VII. RESULTS & ANALYSIS

The Random Forest model demonstrated strong performance on the 1,409-record test set, yielding 80.6% accuracy and an ROC-AUC of 0.847. Given the 73.5/26.5 class imbalance, we observed a trade-off between 66.4% precision and 49.8% recall (F1-score: 56.9%). This indicates that while the model captures half of all true churners, its high precision effectively minimizes the business costs associated with unnecessary interventions on stable accounts.

TABLE I. Model Performance Comparison

VIII. CHURN ANALYTICS INSIGHTS

In Stage 3 of the pipeline, the analytics highlight two primary, independent drivers for churn: contract type and customer tenure. These findings serve a dual purpose. First, they validate our decision to engineer specific tenure group features. Second, and more importantly, they provide clear business intelligence that retention managers can act on immediately giving them the insights they need without requiring them to dig into the raw data themselves.

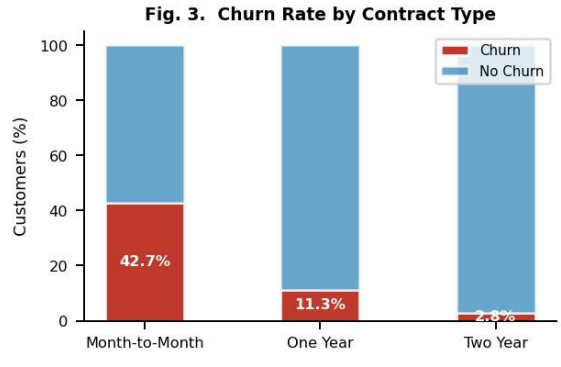

Fig. 3. Churn rate (%) by contract type — month-to-month at 42.7% vs. 2.8% for two-year holders: a 15× differential.

Model	Acc.%	Prec.%	Rec.%	F1%	AUC
Logistic Reg.	80.1	64.2	53.7	58.5	0.836
Decision Tree	72.8	49.6	51.4	50.5	0.699
Grad. Boosting	80.9	67.8	50.2	57.7	0.851
Random Forest ★	80.6	66.4	49.8	56.9	0.847

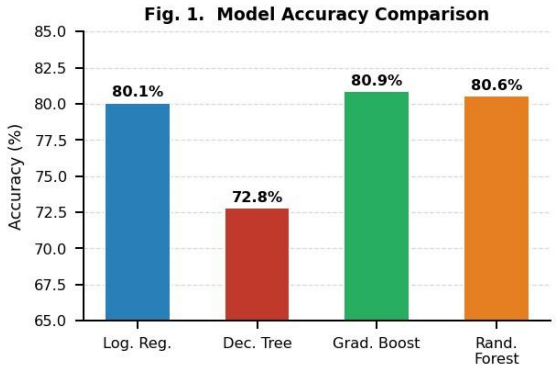

Fig. 1. Test-set accuracy (%) — four classifiers on 1,409 records.

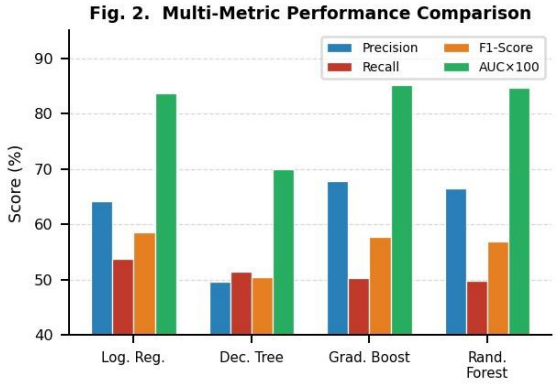

Fig. 2. Precision, Recall, F1-Score and AUC×100 across all models.

highest-leverage strategy for the 5,634 records analysed, supported by the model's native feature importance for clear explainability.

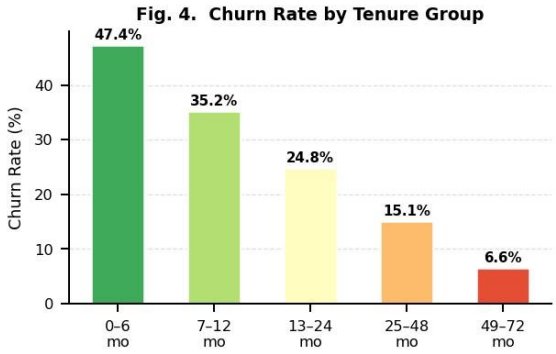

Fig. 4. Churn rate (%) by tenure group — risk declines steeply in first 12 months (Pearson r = −0.35 with churn).

While **Gradient Boosting** offered marginal gains (80.9% accuracy, 0.851 AUC) and the **Decision Tree** proved the weakest baseline (72.8%, 0.699 AUC), Random Forest was selected for production due to its efficient **18-second training time** and competitive metrics. **Contract type** remains the most powerful predictor; month-to-month subscribers churn at **42.7%**, compared to 11.3% for one-year and 2.8% for two-year terms—a 15-to-1 risk spread.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

Our analysis identifies a significant negative correlation (r = -0.35) between customer tenure and churn. The 0–6-month cohort faces the highest risk with a 47.4% churn rate, which consistently drops to 35.2%, 24.8%, 15.1%, and eventually just 6.6% in later stages. The most dramatic decline occurs within the first 12 months, identifying the onboarding period as the most critical window for proactive customer engagement.

IX. SYSTEM OUTPUTS & USER INTERFACE

The Streamlit Customer Churn Intelligence Platform bridges the gap between complex model outputs and the non-technical retention teams who need to act on them. It effectively transforms raw data into actionable business intelligence through an intuitive, user-friendly interface.

To make the tool interactive, we've used slider controls for continuous variables like tenure and monthly fees, while binary service subscriptions are handled via simple toggle switches. Once a customer is scored, the platform displays a calibrated churn probability and a color-coded Risk Level (ranging from Low to Critical). To help staff take immediate action, it also generates a personalized retention recommendation and a visual pie chart that breaks down the ratio of safe versus at-risk customers in the current session.

Fig. 5. Customer Churn Intelligence Platform — prediction 32.41%, Risk Level: Low, retention recommendation, and risk pie chart.

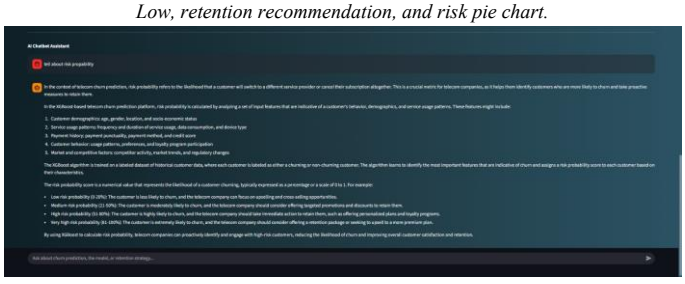

Fig. 6. Integrated AI Chatbot — explaining risk tiers (Low 0–20%, Medium 21–50%, High 51–80%, Very High 81–100%) and retention actions.

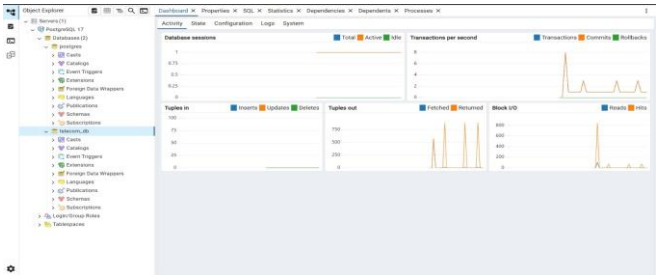

Fig. 7. pgAdmin — telecom_db (PostgreSQL 17): live sessions, transactions/sec, tuple operations, and block I/O activity metrics.

X. DEPLOYMENT

In its current setup, the system is designed to be lean and efficient. A single run_pipeline.sh script handles nine sequential operations, completing a full cycle in about 25 minutes on a standard CPU—no expensive GPUs or complex distributed clusters required. We've also moved the FastAPI extraction server and model serving into separate Docker containers. This "decoupled" approach ensures that different parts of the system can talk to each other over HTTP without getting tangled up.

The serving layer is built to be "plug-and-play." It automatically handles things like aligning columns, filling in missing data, and checking schemas. This means CRM systems and web interfaces can use the model results immediately, without needing to understand the messy technical details of the underlying features. Because our Streamlit frontend only connects via a specific /predict endpoint, we can scale the user interface up or down independently of the heavy-lifting backend. Speed is a major highlight: the entire process takes less than 200ms on average, with the Random Forest model itself adding just a 15ms blip. Finally, switching to PostgreSQL was a gamechanger. Unlike our early SQLite tests, which could only handle one writer at a time, Postgres lets the API, loggers, and admin tools all work simultaneously without any bottlenecks.

XI. LIMITATIONS

The current implementation has several key limitations. First, because the SQLite feature store doesn't support parallel writes, we'll need to migrate to a PostgreSQL backend for any multi-client production environment.

Additionally, the model lacks explicit class-imbalance corrections—such as SMOTE oversampling or asymmetrical weighting. This means recall for the 'churn' class will likely peak at around 50%. Consequently, this predictive approach isn't suitable for high-sensitivity deployments where missing a churn event would significantly impact business costs.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 04 , April - 2026

The pipeline also lacks automated orchestration, data quality controls, and early-warning systems for model drift, which prevents us from achieving a fully functional DataOps workflow. Without these, gradual shifts in market forces or regulations will cause the model's accuracy to decay over time. As the relationship between features and churn changes, the system will fail to recognize churning customers, eventually requiring manual intervention and re-analysis. Finally, the model is currently limited to tabular customer attributes, missing out on the richer behavioural signals found in raw call-control records and customer support logs.

XII. FUTURE WORK

To modernize our workflow, we are moving away from sequential shell scripts in Favor of Apache Airflow or Prefect DAGs. This shift will allow us to implement dependency-based scheduling, automated retries, and better overall observability. We also plan to replace periodic CSV batch uploads with Apache Kafka to support real-time streaming data ingestion.

For model improvements, we will conduct a systematic Bayesian hyperparameter search using Optuna [9] across our Random Forest, Gradient Boosting, and XGBoost [8] candidates. To improve recall, we are implementing SMOTE for class-imbalance correction. Additionally, we will integrate SHAP-based feature attribution to provide per-prediction explainability reports directly within the Streamlit interface.

Our infrastructure roadmap includes full containerization via Docker Compose for all pipeline components. To handle high-frequency scoring efficiently, we will use Redis for caching. We are also integrating Prometheus and Grafana for real-time performance dashboards and exploring federated learning to train models without needing to centralize sensitive customer data.

XIII. CONCLUSION

This study presents the design, empirical evaluation, and implementation of a comprehensive end-to-end Machine Learning (ML) pipeline specifically engineered for identifying churn within the telecommunications sector. By developing a nine-stage modular system, the project encompasses the entire ML lifecycle—ranging from REST API-driven data extraction to interactive inference via Streamlit—moving beyond the isolated model optimization typical of previous research.

The core Random Forest model delivered an accuracy of 80.6% and an ROC-AUC of 0.847. Deep-dive pipeline analytics highlighted contract type (revealing a stark contrast of 42.7% versus 2.8% churn at opposite ends of the spectrum) and early tenure (with 47.4% churn in the 0–6-month cohort) as the primary drivers of customer attrition.

The final deployment of the Streamlit Customer Churn Intelligence Engine, which integrates an AI chatbot and a PostgreSQL 17 production datastore, proves the viability of a practical, open-source, Python-based solution. Ultimately, this architecture provides a repeatable DataOps benchmark for applying ML to telecom and similar industries, offering a robust framework to support high-value customer retention strategies.

REFERENCES

[1] W. Verbeke et al., "New insights into churn prediction: A profit driven data mining approach," European J. Operational Research, vol. 218, no. 1, pp.211–229, 2012.
[2] T. Vafeiadis et al., "A comparison of machine learning techniques for customer churn prediction," Simulation Modelling Practice and Theory, vol. 55, pp.1–9, 2015.
[3] B. Huang, M. T. Kechadi, and B. Buckley, "Customer churn prediction in telecommunications," Expert Systems with Applications, vol. 39, pp. 1414–1425, 2012.
[4] D. Sculley et al., "Hidden technical debt in machine learning systems," Advances in NeurIPS, vol. 28, 2015.
[5] M. Zaharia et al., "Accelerating the machine learning lifecycle with MLflow," IEEE Data Eng. Bulletin, vol. 41, no. 4, pp. 39–45, 2018.
[6] L. Breiman, "Random forests," Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.
[7] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," JMLR, vol. 12, pp. 2825–2830, 2011.
[8] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," Proc. KDD, pp. 785–794, 2016.
[9] T. Akiba et al., "Optuna: A next-generation hyperparameter optimization framework," Proc. KDD, pp. 2623–2631, 2019.
[10] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," NeurIPS, vol. 30, 2017.