Customer Churn Prediction Research Project Applying MLOps

Ngoc Quang Vinh Pham
25100660
Ngoc.Q.Pham-1@student.uts.edu.au
https://github.com/1ThomasS/Customer_Churn_Prediction_Project_Applying_MLOps

I. Abstraction

In the banking sector, retaining customers is significantly more cost-effective than acquiring new ones, making customer churn prediction crucial for proactive retention strategies. This research explores the complexities of customer behavior in the foreign banking industry by using real-world transactional data and behavioral insights to gain an in-depth understanding of customer segments and churn drivers. While machine learning models have shown substantial promise in predicting churn, the operational challenge lies in deploying and managing these models at scale in real-time environments. This project employs advanced Machine Learning Operations (MLOps) frameworks—Kubernetes, Kubeflow, and Feast—to automate and scale the churn prediction pipeline, using Kubernetes for container orchestration, Kubeflow for lifecycle management, and Feast for feature consistency. By leveraging real-world data from a foreign bank, this study seeks to deliver a robust, scalable, and dynamic solution that provides insights into the nuanced patterns of customer churn, enabling banks to manage retention efforts more effectively.

II. Introduction

A. Background and Context

In the competitive landscape of foreign banking, customer churn prediction is vital for maintaining profitability and nurturing long-term relationships. Churn prediction involves identifying customers at risk of discontinuing the use of banking services, which might entail closing accounts, shifting to competitors, or reducing product engagement. With the capability to accurately anticipate churn, banks can implement targeted retention measures, offering tailored incentives to retain high-value customers. However, the real challenge lies in interpreting complex customer behaviors within diverse segments and integrating these insights into predictive models.

Customer behavior in the banking sector is intricate, driven by a range of factors such as financial habits, demographic characteristics, and service interactions. The Foreign banking environment presents an additional layer of complexity due to diverse customer segments, regional differences, and a rapidly evolving digital landscape. Given the wealth of behavioral data available—including transaction histories, account usage, and service interactions—this research emphasizes the need for a deep, data-driven approach to customer segmentation and behavioral analysis to improve churn prediction accuracy.

However, while machine learning has proven effective in predicting churn, deploying these models presents significant operational challenges. Banks must handle sensitive customer data in compliance with strict data privacy laws, while also addressing the dynamic nature of customer behavior. Traditional machine learning model deployments tend to rely on manual processes, which are not scalable and make retraining and updating models to reflect real-time trends difficult.

The adoption of MLOps (Machine Learning Operations) practices, which bring DevOps principles to machine learning, provides a solution to these challenges by automating much of the machine learning lifecycle—model development, deployment, monitoring, and retraining[1]. By using MLOps, organizations can improve scalability, reliability, and collaboration across teams, ensuring that models remain efficient and easy to maintain in production environments.

B. Motivation

As customer expectations and behaviors become more multifaceted, foreign banks must adapt by developing sophisticated methods to capture these behaviors accurately. Traditional churn prediction models struggle to reflect the complexity of modern customer behavior, especially without an automated pipeline that can dynamically retrain and adapt based on evolving patterns. The integration of MLOps frameworks allows for a scalable, automated approach to deploying machine learning models that accurately capture and respond to customer behavioral nuances.

This research investigates how MLOps tools—specifically Kubernetes, Kubeflow, and Feast—can streamline the deployment and management of customer churn prediction models, while enabling a nuanced analysis of customer segments and behavioral patterns. By exploring real-world data from a foreign bank, this project aims to uncover deeper insights into the unique behaviors and characteristics of different customer segments, supporting the bank in managing retention strategies more effectively.

C. Objective

The primary objective of this research is to create an automated, scalable churn prediction pipeline capable of capturing complex behavioral patterns and segmentation insights. Using MLOps frameworks, the specific goals are to:

Automate the training, deployment, and retraining of machine learning models that analyze customer segmentation and behavior for churn prediction.
Enhance the scalability and robustness of model deployment with Kubernetes and Kubeflow.
Maintain consistency in data handling and feature engineering with Feast, ensuring accuracy across both training and real-time inference.
Enable Foreign banks to develop informed retention strategies based on deep behavioral insights derived from real-world customer data.

D. Scope

This research is focused on applying MLOps frameworks to the specific context of customer churn prediction within a foreign bank. The methods used—segmentation analysis, behavioral profiling, and MLOps deployment—can be generalized to other financial institutions and industries where customer behavior insights are valuable. Rather than developing new machine learning algorithms, this project seeks to refine the operationalization of existing models to better reflect the complexity of customer behavior, enhancing model scalability and real-time responsiveness.

III. Literature Review

First and foremost, we explored academic publications, industry reports, and existing technologies related to customer churn prediction and MLOps frameworks. There is an abundance of research in both fields, and many studies have been highly informative and practical, offering inspiration for this research. A common theme among the studies is the increasing complexity of customer behavior and the rapid evolution of machine learning technologies, which have highlighted the need for scalable, automated solutions to efficiently handle customer churn prediction at scale, especially in the banking industry.

One such study, "Customer Churn Prediction in the Banking Sector Using Machine Learning-Based Classification Models" by Tran et al. (2023)[2], investigates how machine learning models such as Random Forest, Logistic Regression, and Support Vector Machine (SVM) can be used for churn prediction in the banking sector. The authors highlight that Random Forest performed the best with a 97% accuracy, showcasing its ability to handle complex banking data. This study also explored the role of customer segmentation, concluding that it did not significantly affect model accuracy, but the choice of machine learning models played a crucial role in prediction performance.

Similarly, the article "Deployment of ML Models using Kubeflow on Different Cloud Providers" (2022)[3] focuses on the operational challenges that arise in deploying machine learning models in production environments. The authors argue that as businesses generate increasingly large datasets, manual model management quickly becomes inefficient, particularly when regular model retraining and performance monitoring are required. Their research explores how Kubernetes and Kubeflow have been applied to automate the deployment, scaling, and retraining of machine learning models across different cloud platforms. By leveraging containerization and orchestration, the authors demonstrate that MLOps frameworks like Kubeflow and Kubernetes can substantially reduce the time and effort needed for operationalizing machine learning models while also ensuring consistency in model performance.

Furthermore, "How Feature Stores Enhance Model Performance in ML Technology" (2023)[3] explores the pivotal role of feature stores in machine learning pipelines. The authors emphasize that a feature store serves as a centralized repository that simplifies feature engineering, ensuring that the same features used during model training are consistently applied during inference. This consistency helps improve model performance, reduce errors, and streamline the machine learning workflow by enabling faster experimentation and feature reuse across projects. By facilitating collaboration between data engineers and data scientists, feature stores also help ensure that high-quality, well-engineered features are accessible for real-time and batch processing, ultimately improving the reliability of machine learning models.

All these studies support the integration of MLOps frameworks to streamline the deployment and management of machine learning models for churn prediction. Tran et al. emphasize the importance of machine learning algorithms such as Random Forest in banking churn prediction, while Kreuzberger et al. highlight the operational efficiencies gained from Kubernetes and Kubeflow. The article on feature stores demonstrates the importance of feature management for maintaining consistency across the machine learning pipeline, particularly when dealing with large datasets and real-time predictions.

The common thread across these studies is the recognition that as customer behavior becomes increasingly complex and data-driven, traditional machine learning workflows are insufficient for handling the scale and velocity of data. By integrating MLOps practices, organizations can automate and scale the entire machine learning lifecycle, from feature engineering to model deployment and monitoring. In this research, we aim to expand on these findings by creating a robust, automated pipeline that leverages the full potential of Kubernetes, Kubeflow, and Feast, providing a more efficient and scalable solution for bank customer churn prediction.

IV. Problem Statement

In the modern Foreign banking sector, customer retention is crucial to sustaining profitability and staying competitive. The challenge lies in accurately predicting which customers are likely to leave, based on their unique behaviors, segments, and interactions. Traditional churn prediction methods often fall short of capturing the full complexity of customer behavior, particularly without continuous retraining and real-time monitoring. In a real-world context, such as that of a Foreign bank, customer behaviors are influenced by diverse factors including economic conditions, personal finance habits, and cultural aspects.

While machine learning models such as **Random Forests**, **Gradient Boosting Machines**, and **Support Vector Machines** are highly effective at predicting churn based on customer transaction data, deploying these models in real-time production environments is complex. Banks handle sensitive data and must comply with strict privacy regulations, adding another layer of complexity to managing these models. Additionally, customer behaviors shift rapidly, meaning churn prediction models need to be frequently retrained and updated.

Manual retraining is often slow and inefficient, which can lead to models becoming outdated and less accurate. Moreover, ensuring consistent feature management across training and production environments is critical to maintaining model accuracy and compliance with data governance standards. Without an automated system, the risk of errors increases, and the models may fail to keep pace with real-time customer behavior.

This research seeks to address the critical challenges of **automating the deployment, scaling, and retraining** of churn prediction models in the banking sector and the **challenges in capturing nuanced behavioral patterns and customer segmentation**. By leveraging **MLOps frameworks**—specifically **Kubernetes**, **Kubeflow**, and **Feast**—the goal is to build a scalable and automated solution that can handle large datasets and adapt to shifting customer behaviors. These tools will enable banks to deploy and manage machine learning models efficiently, ensuring **operational efficiency**, **data consistency**, and **regulatory compliance**.

The main challenges this research aims to solve include:

- **Complexity in customer behavior:** How can MLOps frameworks support models that adapt to multifaceted customer behaviors and segmentation insights?
- **Scalability and automation:** How can Foreign banks handle large, dynamic datasets and ensure real-time responsiveness without compromising security or regulatory compliance?
- **Feature consistency:** How can feature stores like Feast ensure consistent and accurate data handling across training and production environments?
- **Data privacy compliance:** How can banks operationalize MLOps frameworks while adhering to strict data privacy laws and ensuring data governance?

By addressing these challenges, the research aims to provide banks with a comprehensive solution that improves customer retention strategies while enhancing the efficiency of machine learning operations.

V. Methodology

The methodology for this research is designed to build an end-to-end MLOps pipeline for predicting customer churn in the banking sector. The key components include data acquisition, model selection, MLOps framework setup, and evaluation. The pipeline will automate the entire machine learning lifecycle, from data ingestion and model training to deployment, monitoring, and retraining. Below is a detailed breakdown of the methodology.

A. Data Collection and Preprocessing

Data Source: The dataset in use is derived from real-world data collected by a foreign bank. This rich dataset, comprising over 2 million records, includes detailed customer information, account activities, transaction records, and behavioral indicators essential for accurately modeling customer churn within the banking sector.

Data Fields and Feature Categorization: The dataset contains the following primary features, categorized to support in-depth exploration of customer segmentation and behavioral patterns:

1. Customer Identification and Demographic Attributes

- customer_id: A unique identifier assigned to each customer for identification and tracking.
- short_name, customer_full_name: Abbreviated and full names used for customer recognition and documentation.
- customer_joining_age: The age of the customer at the time of joining, providing insights into their lifecycle stage at the start of their relationship with the bank.
- customer_age: The current age of the customer, which aids in demographic segmentation for targeted services.
- gender, marital_status: Gender and marital status of the customer, essential attributes for behavioral and demographic segmentation.

2. Account and Relationship Details

- customer_open_date_numeric, customer_start_date_numeric: Numerical representations of the account opening date and customer start date, useful for calculating tenure and understanding the customer's lifecycle stage.
- customer_segment, business_class_code: Detailed segmentation information based on business and relationship attributes, enabling targeted strategies for different customer groups.
- company_book: A reference to the branch or division managing the customer's account, which can indicate regional or operational presence.
- bucket_abb: Categorization of accounts reflecting spending and saving behavior, valuable for financial profiling.
- target_name: Indicates the bank's categorization of the customer, reflecting specific goals or classification within the customer base.

3. Transactional and Behavioral Patterns

- business_class_code, customer_segment: Industry and relationship classifications that provide context for customer engagement and economic activity.
- contact_date_numeric, contact_time_numeric: Timestamped data on customer interactions, offering insights into engagement patterns and frequency.
- customer_status: A categorical field denoting the current status of the customer within the bank, relevant for churn and activity analysis.
- tenure_years: Length of time the customer has maintained their relationship with the bank, which can signal loyalty and satisfaction levels.

4. Business and Operational Attributes

- company_book, business_class_code: Operational classifications reflecting the customer's relationship with specific business divisions or companies.
- segment_start_date_numeric, segment_end_date_numeric: Numerical representation of segment validity periods, enabling time-based analysis of customer segmentation strategies.
- customer_open_date_numeric: A numerical encoding of account opening dates to facilitate temporal modeling and analysis.

5. Customer Engagement and Historical Recordkeeping

- record_start_date_numeric, record_end_date_numeric: Numerical representation of record validity periods, enabling analysis of historical changes and trends.

- segment_start_date_numeric, segment_end_date_numeric: Encoded start and end dates for customer segmentation, providing temporal data for segment evolution.
- contact_date_numeric: Dates of customer interactions, useful for analyzing patterns of engagement and activity.

Target Variable: churn
This feature, churn, indicates whether the customer has been active in transactions over the past 12 months. It is transformed into a Boolean field, using the following logic:

- TRUE: The customer has been active in transactions, as indicated by ACTIVE_FLAG being 'Y'.
- FALSE: The customer has not been active in transactions, derived from any other value in ACTIVE_FLAG.

Data Preprocessing:
To ensure data quality and compatibility with the machine learning models, the following preprocessing steps will be applied:

1. Handling Missing Data

- Columns with more than **60% missing values** were dropped, including sub_industry_id, sub_industry_name, standardized_corp_level, introducer, company_vip, and active_date.
- **Mean imputation** was used for numerical fields with missing values:
    - customer_joining_age: Filled with the mean value.
    - customer_age: Filled with the median value for better robustness against outliers.
    - tenure_years: Filled with the median value to capture central tendencies.
- **Mode imputation** was applied to categorical columns:
    - gender: Replaced null values with the most frequent gender.
    - business_class_code: Filled with the mode of the column.
    - customer_segment: Filled with the most common segment.
- Missing date-related fields were replaced with their respective column means after converting them to numeric formats.

2. Data Filtering

- The dataset was filtered to include only individual customers (sector_name = "Ca nhan" and industry_name = "X0000. Tu nhan").
- Irrelevant columns with little to no impact on the churn analysis, such as sector_id, customer_level_change, industry_code, mis_date, and lob_code_map, were removed to reduce noise and dimensionality.

3. Date Conversion and Feature Engineering

- All date fields (e.g., customer_open_date, record_start_date) were converted to numeric values using the difference in days from the current date or Unix timestamps.
- Engineered numeric date fields such as:
    - customer_open_date_numeric
    - segment_start_date_numeric
    - record_start_date_numeric
- These numeric values captured the temporal dynamics of customer activity and lifecycle stages.

4. Encoding Categorical Variables

- Categorical columns such as target_name, customer_segment, gender, lifecycle_stage, and business_class_code were encoded using a combination of **String Indexing** and **One-Hot Encoding** to transform them into machine-readable formats.

5. Feature Selection

- The final dataset included the following key features:
    - Numerical Features: customer_joining_age, customer_age, tenure_years, and the engineered numeric date features.
    - Encoded Features: One-hot encoded representations of target_name, customer_segment, and business_class_code.
- The churn column was retained as the target variable.

6. Scaling and Normalization

- **Standard Scaling** was applied to all numerical features using the StandardScaler method. This step ensured that all features were normalized to have a mean of 0 and a standard deviation of 1.

- This normalization process mitigated the impact of large-scale differences between features and ensured fair weight allocation during model training.

7. Feature Vectorization

- A **Vector Assembler** was used to combine all features into a single vector column named all_features.
- The assembled feature vector was scaled, resulting in a final column named scaled_features.

8. Final Dataset

- The final preprocessed dataset retained only two columns:
    - churn: The target variable representing customer activity (1 for churned customers, 0 for active customers).
    - scaled_features: A vector containing all scaled and encoded feature values, ready for model input.

This preprocessing workflow ensures that the dataset is optimized for training robust machine learning models and minimizes biases or inconsistencies in feature representation.

B. Exploratory Data Analysis

1. Demographic Attributes

Customer Age:

- The distribution of customer age exhibits a concentration between **20 and 60 years**, with a mean of approximately **39 years**.
- Outliers above 60 years and extending beyond 100 years indicate potential data quality issues or unique customer segments.
- **Implications:** The dominant age group (20–60) represents the majority of the customer base. Focused strategies should target this cohort while addressing potential outliers for tailored services.

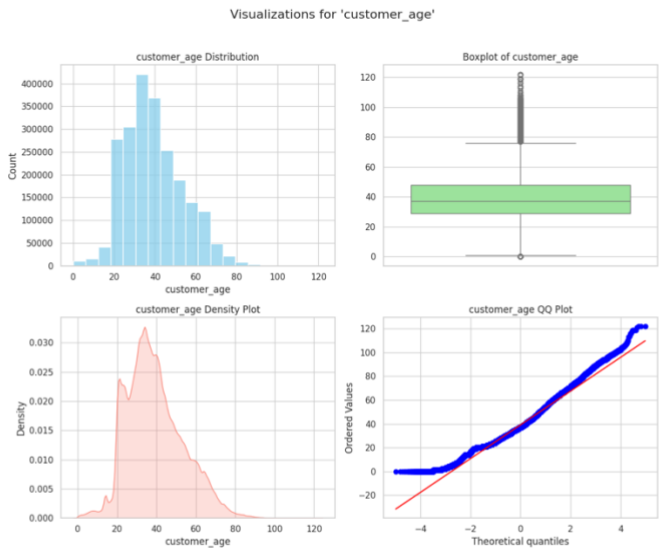


Figure 1: Age Distribution and Boxplot

Customer Joining Age:

- Most customers joined between **30 and 40 years of age**, with a secondary small peak near zero, possibly reflecting minors or data errors.
- Negative joining ages were observed, which require correction.
- **Implications:** Address data inconsistencies and develop engagement strategies for younger customers joining at early life stages.

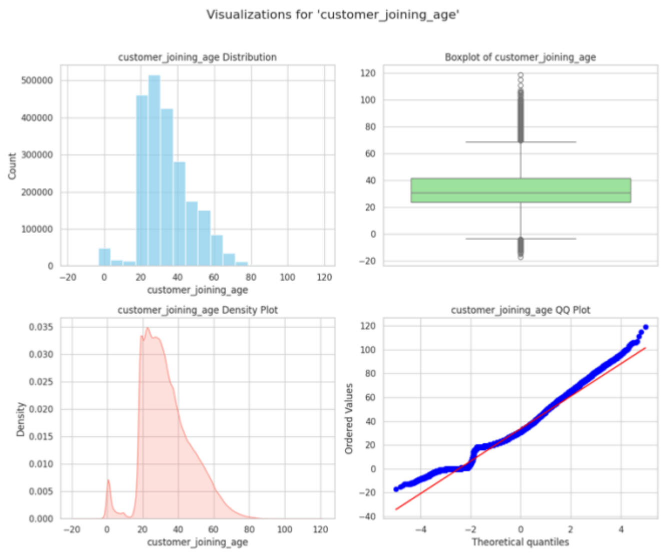


Figure 2: Joining Age Distribution and Boxplot

Gender:

- The dataset has a nearly balanced gender distribution, with **54% male** and **46% female customers**. A small percentage (~3.5%) has missing gender information.

- **Implications:** Ensure inclusivity in marketing strategies and address missing values to enhance gender-based analysis.

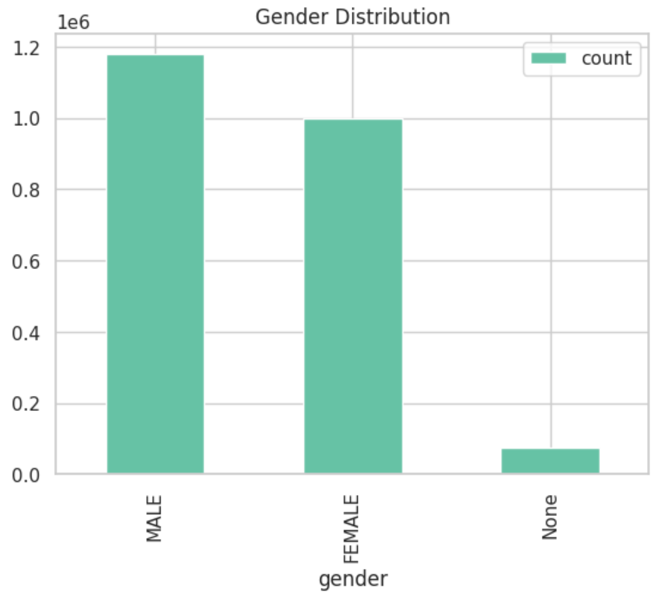


Figure 3: Gender Distribution Bar Chart

Marital Status:

- Approximately **90% of customers are unmarried**, with **10% identified as married**.
- **Implications:** This large unmarried demographic may influence service needs and preferences, emphasizing financial independence or personal savings products.

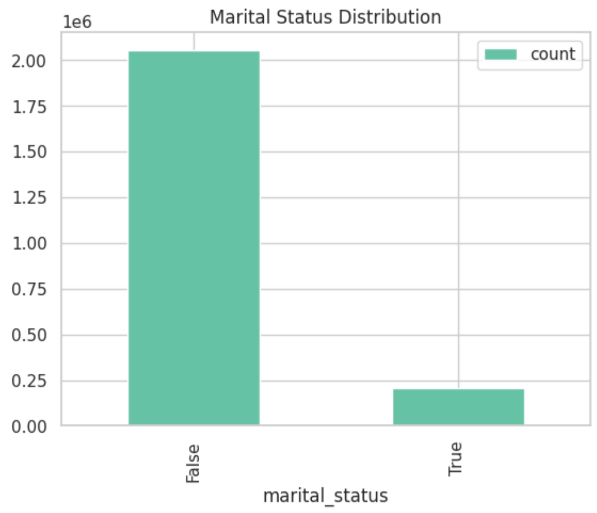


Figure 4: Marital Status Distribution Bar Chart

2. Transactional and Behavioral Attributes
Customer Segments:

- **Example:** Customers in the "MASS" segment (80% of the dataset) have longer average tenures. For instance, a customer aged 45 in this segment has been with the bank for 15 years, indicating loyalty.
- **Implications:** Products such as premium cards or loyalty rewards should be marketed to this segment to enhance engagement.

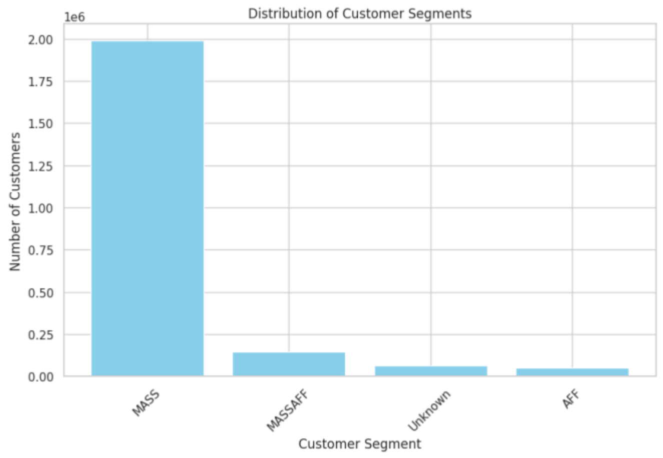


Figure 5: Customer Segment Distribution

Average Segment Duration:

- **Example:** The "MASS" segment customers exhibit an average segment duration of approximately -364 days. For example, a customer aged 40 in the "MASSAFF" segment shows shorter durations, possibly due to recent account openings or churn.
- **Implications:** Investigating the shorter durations in smaller segments like "AFF" can help identify causes of churn and address them.

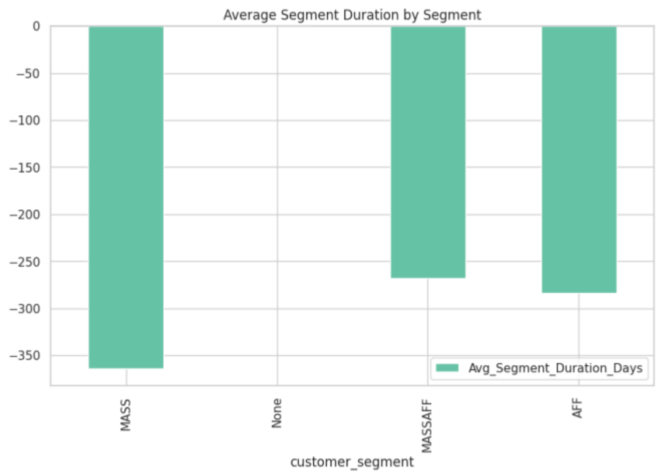


Figure 6: Average Segment Duration by Customer Segment

Bucket ABB Distribution:

- **Example:** Most customers (99%) belong to Bucket ABB '1', representing standard saving or transactional accounts. An outlier example includes a customer categorized in Bucket ABB '5', indicating unusual or specific transaction patterns.
- **Implications:** Expanding the categorization criteria may provide a more granular understanding of customer financial behaviors.

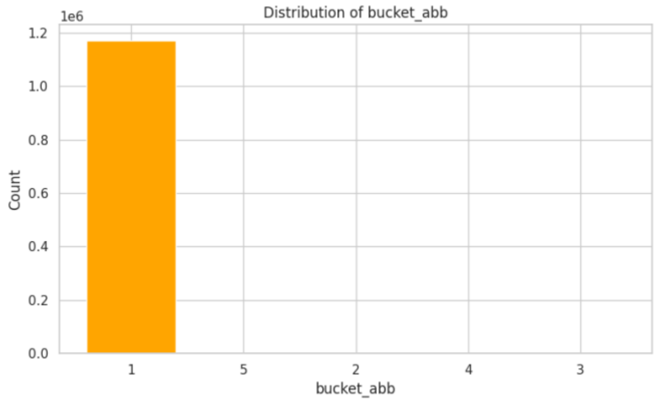


Figure 7: Bucket ABB Distribution

Target Names:

- **Example:** The "KHCN-BinhThuong" target dominates with over 900,000 records, compared to other categories like "KHCN-TrungLuu," which has less than 10,000. For instance, a customer in "KHCN-BinhThuong" is likely a mass-market retail client.
- **Implications:** The bank could diversify efforts across underrepresented target names to reduce over-reliance on a single segment.

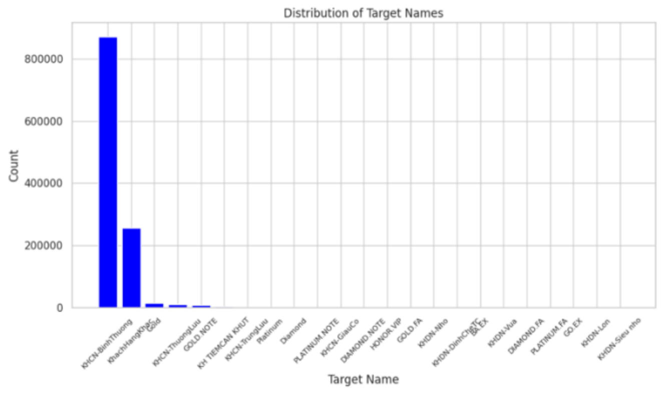


Figure 8: Target Name Distribution

3. Lifecycle Stages

Distribution Across Stages:

- **Example:** The "Early" stage comprises 50% of the dataset, while "New" customers represent less than 5%. A customer aged 25 in the "New" stage with a tenure of six months highlights this disparity.
- **Implications:** Strengthening engagement efforts for "New" customers can help increase their transition to loyal or established stages.

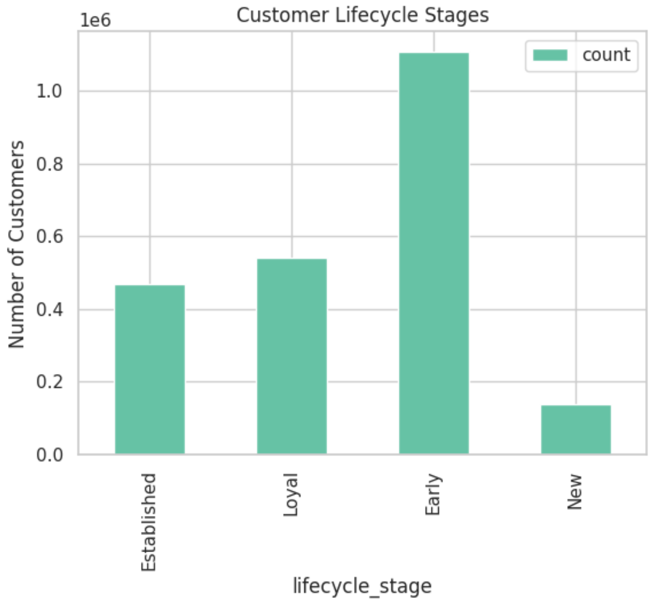


Figure 9: Lifecycle Stage Distribution

4. Correlations and Relationships

Pairwise Relationships:

- **Example:** Customers with shorter tenures are more likely to churn, as indicated by the pairplot. For instance, customers with tenures below two years show a significantly higher churn rate.
- **Implications:** Predictive models should include tenure as a key feature for churn detection and intervention.

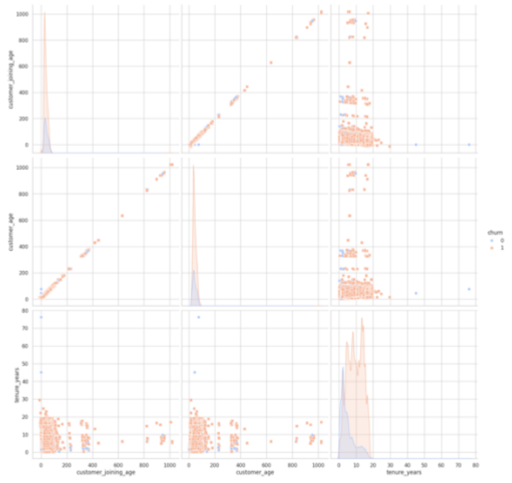


Figure 10: Pairplot of Key Variables

3. Churn Analysis

Churn Rate:

- **Example:** 75,3% of customers churned. A typical churned customer is in the "Early" lifecycle stage, aged 28, with a tenure of fewer than three years. For instance, customer ID 10923864 churned after 2.5 years.
- **Implications:** Churn prevention programs should focus on early-stage customers by improving onboarding processes and engagement activities.

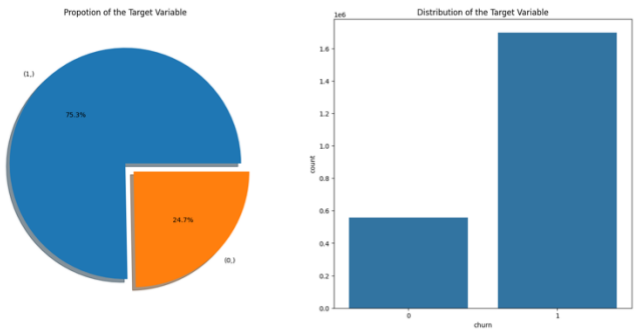


Figure 11: Churn Proportion and Distribution

C. Model Selection

Several machine learning models will be developed and tested to predict bank customer churn. Each model will be evaluated based on its ability to identify churners from the bank's customer base using a combination of transactional and behavioral data.

1. Logistic Regression:

Logistic Regression, a linear model, served as a baseline for comparison. It models the binary outcome of churn (1) versus non-churn (0). This interpretable model provided an initial benchmark for assessing the performance of more complex algorithms.

- **AUC-ROC Score**: 0.9244
- **Strengths**: Simplicity, interpretability, and computational efficiency.
- **Key Usage**: Acts as a performance benchmark and provides insight into key features influencing churn.

2. Random Forests:

Random Forest, an ensemble learning method, was chosen for its ability to handle both categorical and numerical data efficiently. This model captures complex variable interactions, making it effective for churn prediction in banking datasets. Additionally, its built-in feature importance mechanism helps identify the most influential factors contributing to churn.

- **AUC-ROC Score**: 0.9743
- **Strengths**: Handles complex interactions between variables, resistant to overfitting, and provides feature importance.
- **Key Usage**: Particularly effective in identifying the drivers of churn through feature importance metrics.

3. Extreme Gradient Boost:

XGBoost, a gradient-boosting algorithm, was employed for its ability to handle structured datasets and imbalanced data distributions effectively. Its optimization process minimizes errors in predicting churn while maintaining high accuracy, even with skewed distributions where churn events are relatively rare.

- **AUC-ROC Score**: 0.9422
- **Strengths**: Handles imbalanced data distributions, captures non-linear interactions, and offers high accuracy in structured datasets.
- **Key Usage**: Optimized for datasets with rare events such as churn.

4. Model Training Process

For each model:

1. **Data Preparation**:
   - Features were scaled using a standard scaler to ensure consistent influence across variables.
   - The dataset was split into 80% training and 20% testing subsets to evaluate model generalizability.
2. **Hyperparameter Tuning**:
   - Logistic Regression and Random Forest models underwent hyperparameter optimization using cross-validation.
   - XGBoost parameters such as learning rate and tree depth were tuned based on grid search and default settings.
3. **Evaluation**:
   - The models were evaluated using the Area Under the Receiver Operating Characteristic Curve (AUC-ROC) to quantify their ability to differentiate between churn and non-churn customers.

Hyperparameter tuning for each model will be automated using Katib, a tool within Kubeflow. Katib will perform random search and Bayesian optimization to fine-tune parameters such as learning rate and tree depth, ensuring optimal model performance.

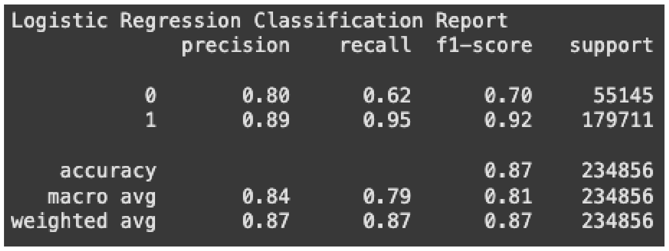
D. Model Evaluation

To assess the effectiveness of the trained models—Logistic Regression, Random Forest, and XGBoost—multiple evaluation metrics and visualizations were employed to interpret their performance in predicting customer churn. The following sections summarize the results for each model:

Logistic Regression

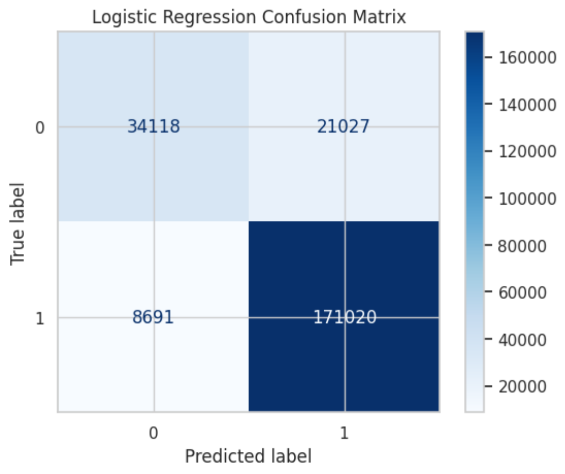
Logistic Regression served as the baseline model for churn prediction. Its performance metrics indicate a reasonably strong ability to distinguish between churners and non-churners:

1. **Classification Report**: The model achieved an **accuracy of 87%**, with an **F1-score** of **92%** for churners (label 1) and **70%** for non-churners (label 0). This highlights its strength in identifying churners, though it showed a slight weakness in recalling non-churners.
    - **Precision**: 89% for churners.
    - **Recall**: 95% for churners.

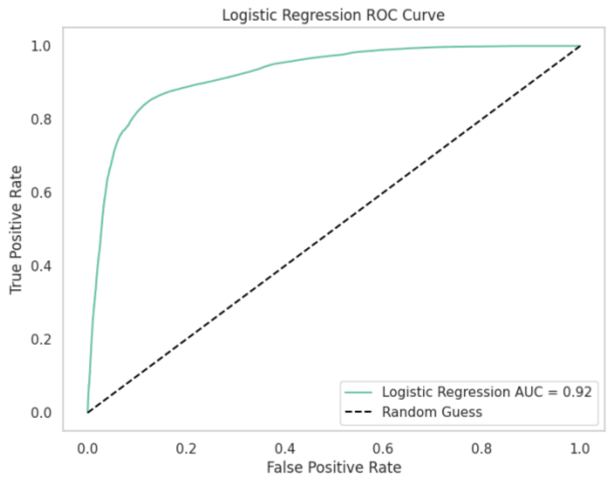


```
Logistic Regression Classification Report
              precision    recall  f1-score   support

           0       0.80      0.62      0.70     55145
           1       0.89      0.95      0.92    179711

    accuracy                           0.87    234856
   macro avg       0.84      0.79      0.81    234856
weighted avg       0.87      0.87      0.87    234856
```

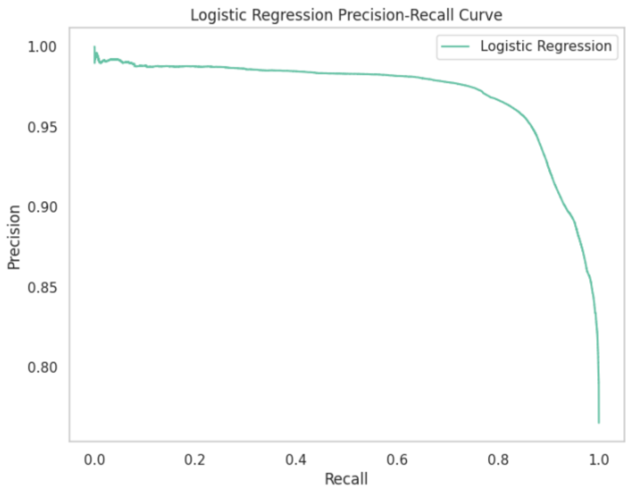
2. **Confusion Matrix**: The confusion matrix reveals the tradeoff between false positives and false negatives. It demonstrates that the model is biased toward identifying churners, which may be desirable in churn prediction contexts where missing a churner is more costly than incorrectly predicting one.



3. **ROC Curve and AUC**: The **AUC of 0.92** underscores strong discriminatory power, indicating that the model is effective in ranking churn probabilities.



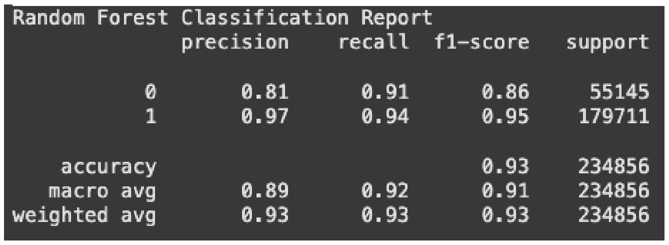
4. **Precision-Recall Curve**: The precision-recall curve illustrates consistent high precision and recall, which further validates its reliability for identifying churners.



Random Forest

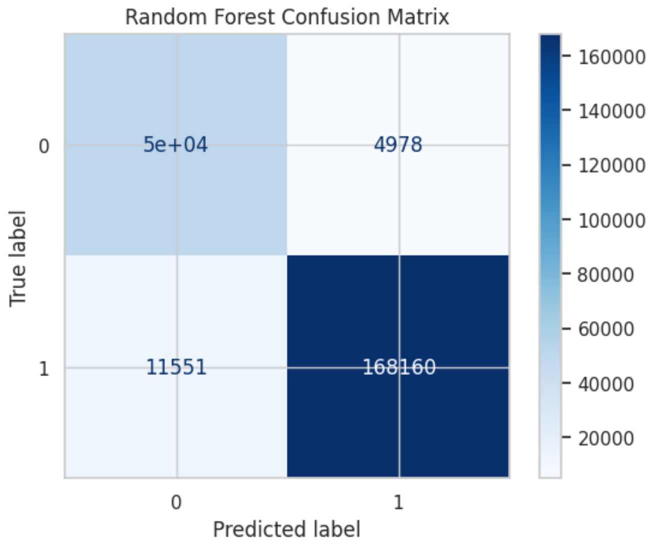
Random Forest performed exceptionally well due to its ability to capture complex patterns in the data:

1. **Classification Report**: The model achieved an **accuracy of 93%**. Its **F1-score** was **95%** for churners, reflecting a balanced tradeoff between precision and recall.
    - **Precision**: 97% for churners.
    - **Recall**: 94% for churners.

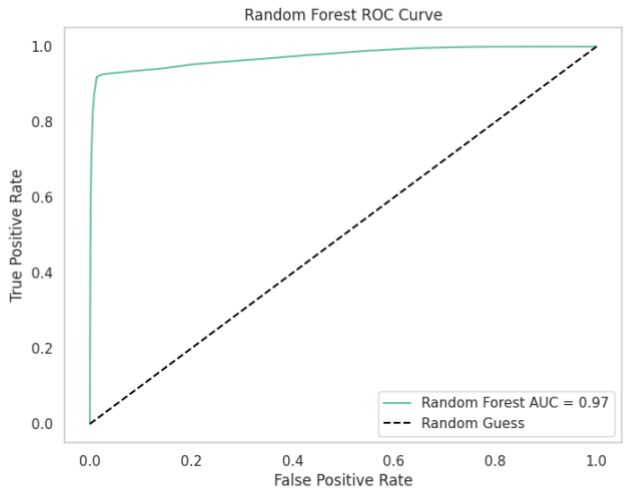


```
Random Forest Classification Report
              precision    recall  f1-score   support

           0       0.81      0.91      0.86     55145
           1       0.97      0.94      0.95    179711

    accuracy                           0.93    234856
   macro avg       0.89      0.92      0.91    234856
weighted avg       0.93      0.93      0.93    234856
```

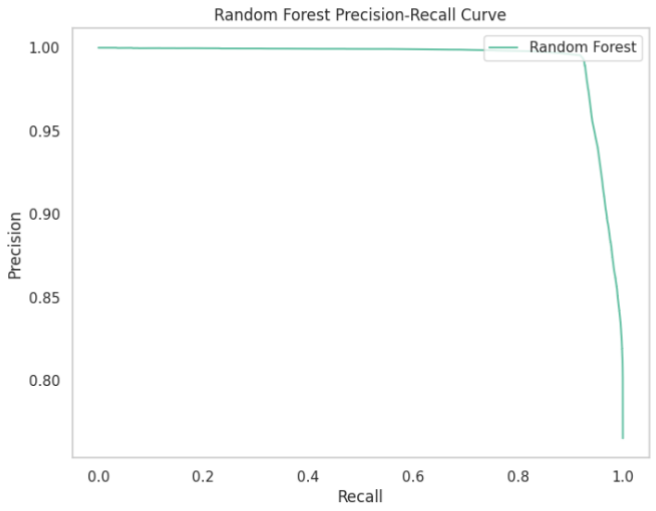
2. **Confusion Matrix**: The confusion matrix shows fewer false negatives and false positives compared to Logistic Regression, which highlights the model's ability to classify churners and non-churners more accurately.



3. **ROC Curve and AUC**: The model achieved an **AUC of 0.97**, indicating near-perfect ranking performance.



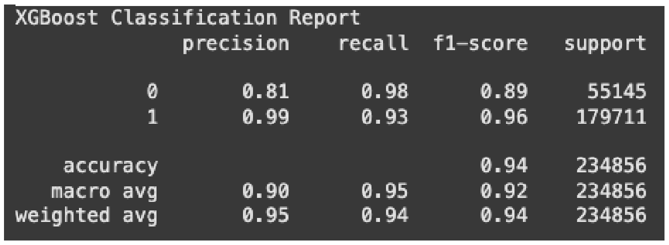
4. **Precision-Recall Curve**: Random Forest maintains very high precision and recall across thresholds, making it a reliable model for practical deployment.



XGBoost

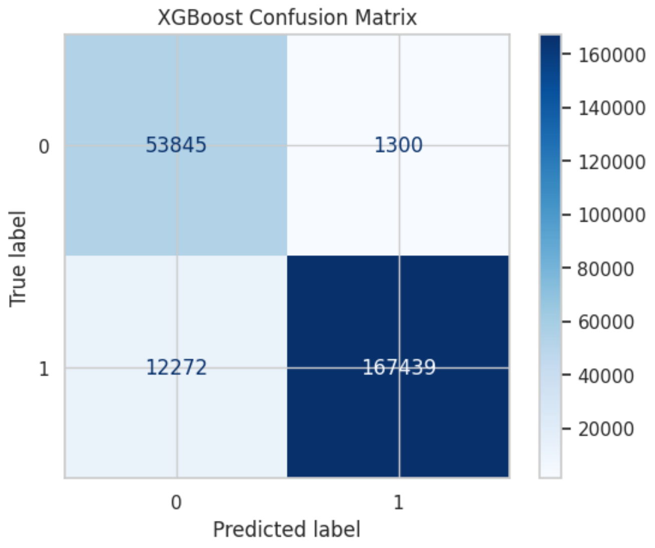
XGBoost outperformed the other models in terms of both accuracy and AUC:

1. **Classification Report**: XGBoost achieved an **accuracy of 94%** with an **F1-score of 96%** for churners.
    - **Precision**: 99% for churners.
    - **Recall**: 93% for churners.

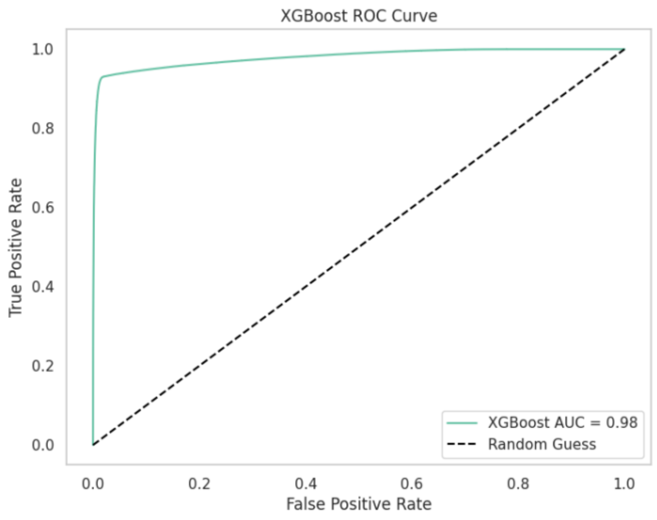


```
XGBoost Classification Report
              precision    recall  f1-score   support

           0       0.81      0.98      0.89     55145
           1       0.99      0.93      0.96    179711

    accuracy                           0.94    234856
   macro avg       0.90      0.95      0.92    234856
weighted avg       0.95      0.94      0.94    234856
```

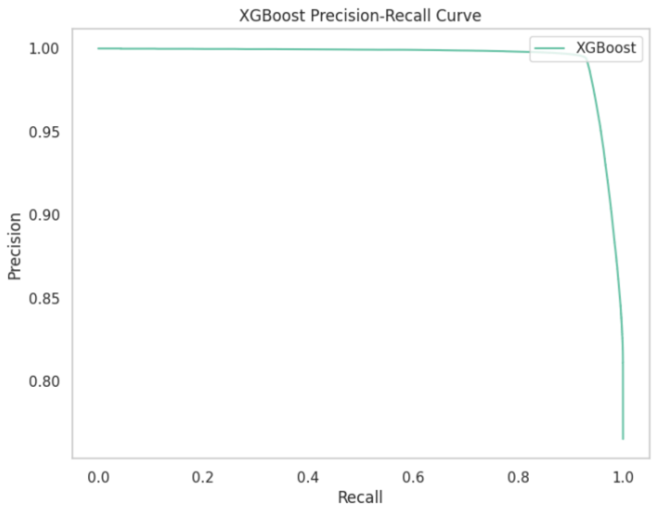
2. **Confusion Matrix**: The confusion matrix shows a significant reduction in false positives compared to Logistic Regression and Random Forest, making XGBoost the most precise model.



3. **ROC Curve and AUC**: The **AUC of 0.98** reflects the model’s ability to perfectly rank churn probabilities, showcasing its effectiveness.

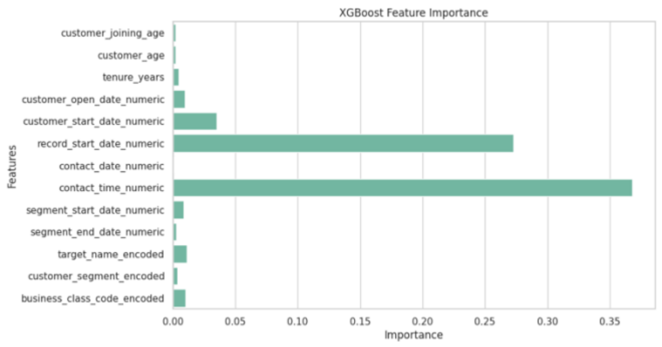


4. **Precision-Recall Curve**: XGBoost consistently maintained the highest precision across all recall levels, further highlighting its superiority.



5. **Feature Importance**: The feature importance plot reveals that the most influential features include **contact_time_numeric**, **record_start_date_numeric**, and **customer_start_date_numeric**, emphasizing the temporal aspects of customer activity as key predictors of churn.

6.

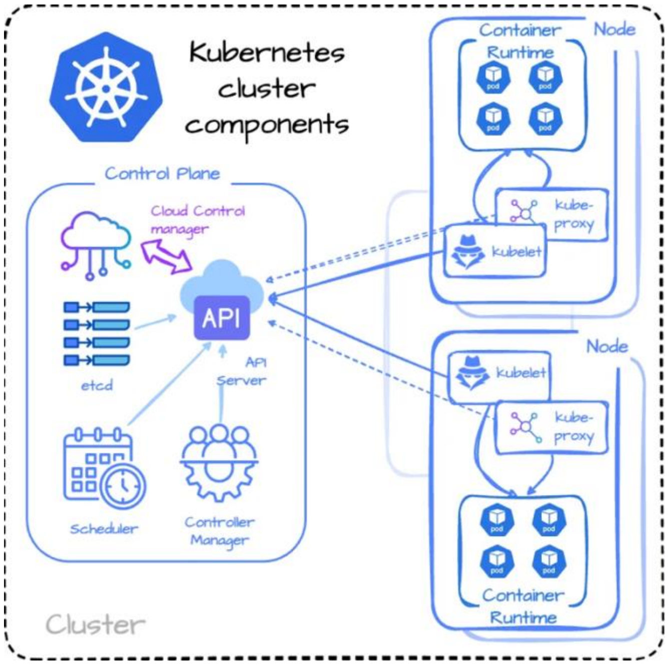


E. MLOps Framework Implementation

The MLOps pipeline for the bank churn prediction model will be built using several key components, with Google Cloud being the central platform due to the limited capacity of the local machine. Google Cloud provides the necessary scalability, compute power, and infrastructure needed to manage resource-intensive machine learning operations that cannot be efficiently handled locally.

1. Kubernetes for Orchestration:

Kubernetes will be used for orchestrating the containerized machine learning models. These models will be packaged into Docker containers to ensure consistency between development and production environments. However, the latest versions of Kubernetes have transitioned the default container runtime from Docker Engine to Containerd, which offers improved performance and resource management. Given the high computational needs of real-time churn prediction, deploying Kubernetes on Google Kubernetes Engine (GKE)—Google Cloud's managed Kubernetes service—will provide the necessary scalability and robustness.

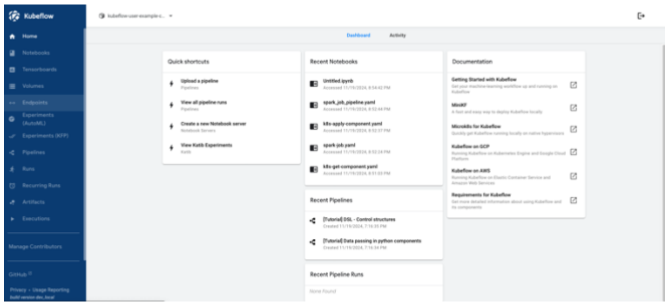


Since the local machine lacks the necessary capacity to efficiently handle a production-scale Kubernetes setup, Google Cloud provides a much more feasible solution by offering elastic resource allocation, automatic scaling, and high availability for the machine learning models.

2. Kubeflow for Pipeline Automation on Google Cloud: Due to the limited local computational capacity, Kubeflow will be deployed on Google Cloud for efficient pipeline automation. Kubeflow is an open-source MLOps platform that integrates seamlessly with Kubernetes and automates the various stages of the machine learning lifecycle[6]. This includes:

- **Central Dashboard:**

The central dashboard provides an intuitive interface to manage all Kubeflow components, pipelines, and experiments. It acts as a unified access point to monitor the state of the MLOps pipeline and track activities such as recent notebook sessions, pipeline executions, and ongoing experiments.



Quick shortcuts

Upload a pipeline
Pipelines

View all pipeline runs
Pipelines

Create a new Notebook server
Notebook Servers

View Katib Experiments
Katib

Recent Notebooks

Untitled.ipynb
Accessed 11/19/2024, 8:54:42 PM

spark_job_pipeline.yaml
Accessed 11/19/2024, 8:52:44 PM

k8s-apply-component.yaml
Accessed 11/19/2024, 8:52:37 PM

spark-job.yaml
Accessed 11/19/2024, 8:52:24 PM

k8s-get-component.yaml
Accessed 11/19/2024, 8:51:03 PM

Recent Pipelines

[Tutorial] DSL - Control structures
Created 11/19/2024, 7:16:35 PM

[Tutorial] Data passing in python components
Created 11/19/2024, 7:16:34 PM

Recent Pipeline Runs

None Found

Documentation

Getting Started with Kubeflow
Get your machine-learning workflow up and running on Kubeflow

MiniKF
A fast and easy way to deploy Kubeflow locally

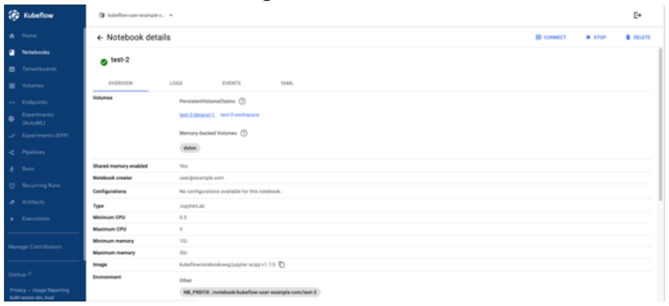
Microk8s for Kubeflow
Quickly get Kubeflow running locally on native hypervisors

Kubeflow on GCP
Running Kubeflow on Kubernetes Engine and Google Cloud Platform

Kubeflow on AWS
Running Kubeflow on Elastic Container Service and Amazon Web Services

Requirements for Kubeflow
Get more detailed information about using Kubeflow and its components

- **Notebooks**: Jupyter notebooks are created and managed through the Kubeflow dashboard for development and experimentation. These notebooks are backed by persistent storage to ensure data consistency and reuse.
  - *Example*: A test notebook was deployed, as shown below, with details such as resource allocations and configurations for experimentation:

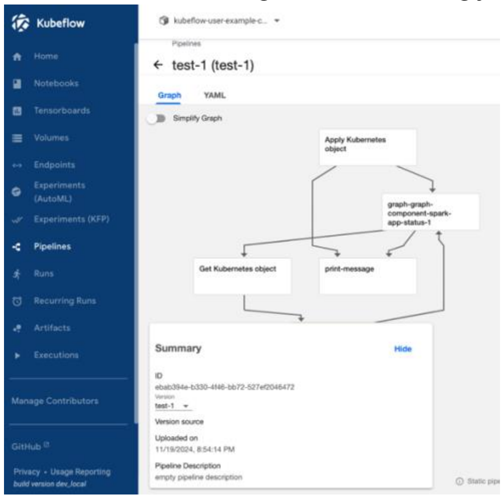


Notebook details

test-2

Volumes	PersistentVolumeClaims test-2-datavol-1 test-2-workspace Memory-backed Volumes dshm
Shared memory enabled	Yes
Notebook creator	user@example.com
Configurations	No configurations available for this notebook.
Type	JupyterLab
Minimum CPU	0.5
Maximum CPU	3
Minimum memory	1Gi
Maximum memory	3Gi
Image	kubeflownotebookswg/jupyter-scipy:v1.7.0
Environment	Other NB_PREFIX: /notebook/kubeflow-user-example-com/test-2

- **Pipelines**: Pipelines are created to automate the entire machine learning lifecycle, including data preprocessing, model training, and deployment. The pipeline's graphical representation highlights the interconnection between components, ensuring a clear workflow.
  - *Example*: The pipeline shown below illustrates tasks such as applying Kubernetes objects, retrieving objects, and executing machine learning jobs:



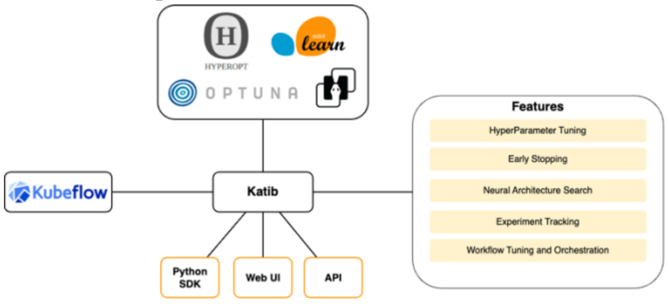
test-1 (test-1)

Summary

ID
ebab394e-b330-4f46-bb72-527ef2046472
Version
test-1
Version source
Uploaded on
11/19/2024, 8:54:14 PM
Pipeline Description
empty pipeline description

- **Katib:**

Katib automates hyperparameter tuning for machine learning models. It employs techniques such as random search and Bayesian optimization to find the best hyperparameters, improving model performance. Katib integrates seamlessly with Kubeflow pipelines for streamlined optimization.



- **Model Registry:**

The model registry within Kubeflow ensures versioning and tracking of trained models. It facilitates easy deployment of the best-performing models and supports monitoring for model drift, ensuring sustained accuracy over time.

- **Spark Operator:**

The Spark Operator integrates Apache Spark into the Kubeflow ecosystem, enabling distributed data processing and model training. This is particularly useful for handling large-scale datasets and computationally intensive tasks.

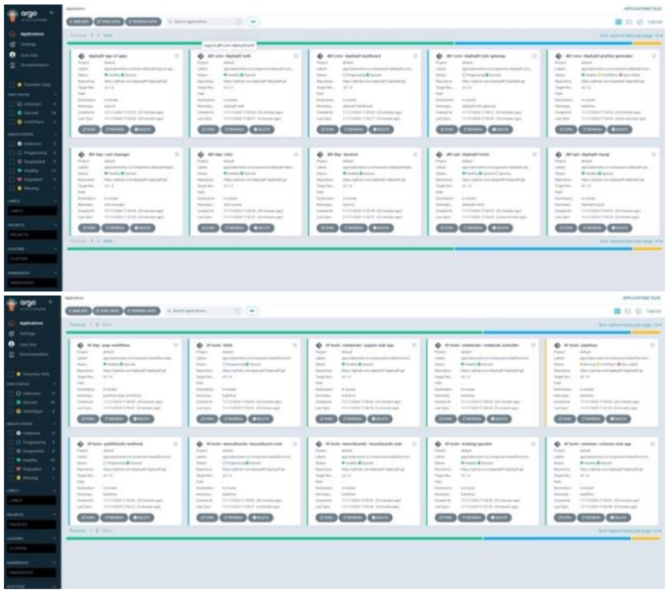
- **Training Operator:**

The Training Operator simplifies the process of deploying and managing distributed machine learning training jobs on Kubernetes. It supports frameworks such as XGBoost, enabling efficient utilization of cluster resources.

3. Argo CD for Application Management

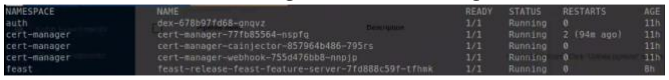
Argo CD is used for managing the deployment and synchronization of various applications within the MLOps framework. The status of components such as dashboards, gateways, and feature stores is visualized to monitor health and ensure synchronization.

- *Example*: The dashboard below highlights the status of different components, such as deploykf-app-of-apps and deploykf-dashboard:



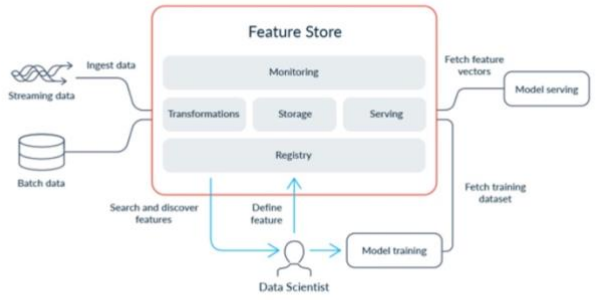
4. Feast as a Feature Store on Google Cloud:

Feast will be used as a centralized feature store to manage the features used during model training and inference.



```
NAMESPACE      NAME                                             READY   STATUS    RESTARTS        AGE
auth           dex-678b97fd68-gnqvz                             1/1     Running   0               11h
cert-manager   cert-manager-77fb85564-nspfq                     1/1     Running   2 (94m ago)     11h
cert-manager   cert-manager-cainjector-857964b486-795rs         1/1     Running   0               11h
cert-manager   cert-manager-webhook-755d476bb8-nnpjp            1/1     Running   0               11h
feast          feast-release-feast-feature-server-7fd888c59f-tfhmk  1/1  Running   0               8h
```

This includes managing features such as customer transaction history, credit score trends, and account activity. Due to the large volumes of data and the need for real-time feature updates, the limited local infrastructure is insufficient for managing these tasks efficiently.



By deploying Feast on Google Cloud, the following advantages are achieved:

- Feature Consistency:

Feast ensures that the same features are used in both batch training and real-time inference, preventing training-serving skew. By using Google Cloud, Feast can scale to handle the growing feature set without performance degradation, a challenge that would arise if the feature store were run on a local machine.

- Batch and Real-Time Data Management:

Google Cloud allows Feast to handle both batch data (historical transaction data for model training) and real-time data (new transactions or customer updates). This ensures that the churn prediction models always operate with the most up-to-date customer information, enabling banks to make accurate predictions in real time.

- Feature Versioning and Governance:

Google Cloud offers the necessary resources for managing feature versions and maintaining governance over the entire feature pipeline. This is critical in the banking industry, where compliance with data regulations such as GDPR is mandatory. By utilizing Feast on Google Cloud, all feature updates, transformations, and usage are tracked and versioned, ensuring full compliance.

Advantages of Cloud-Based Implementation

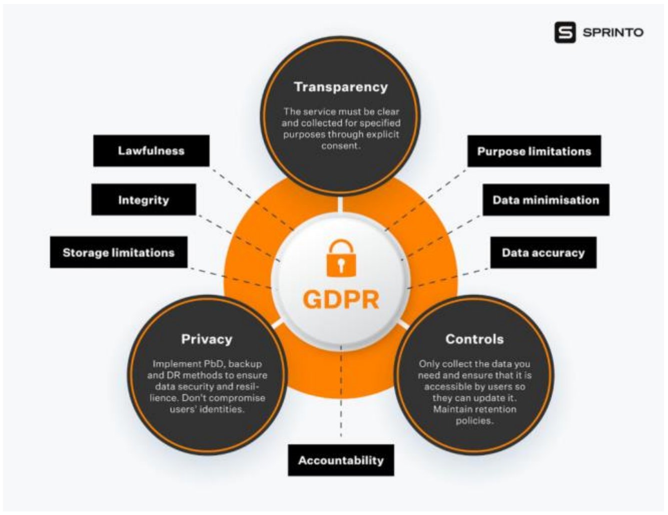
- **Scalability**: The cloud-based infrastructure can dynamically scale to meet high demands, enabling efficient handling of large datasets and complex models.
- **Performance**: Google Cloud's robust infrastructure ensures low latency and high throughput for real-time predictions.
- **Compliance**: Built-in tools for data governance and security ensure adherence to banking regulations and standards.

F. Compliance and Security

Ensuring compliance and security is a critical component of the MLOps pipeline, especially in the context of sensitive financial data such as customer churn prediction in the banking sector. This section outlines the planned compliance and security measures to safeguard data and models.

1. Regulatory Compliance

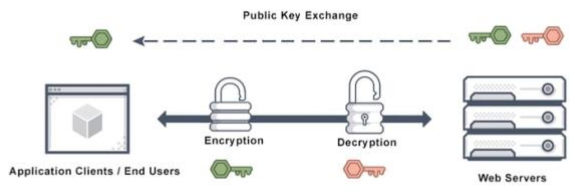
- **Data Protection Regulations:** The pipeline will adhere to relevant data protection regulations, such as the General Data Protection Regulation (GDPR) for data privacy and handling customer information responsibly.


SPRINTO
Transparency
The service must be clear and collected for specified purposes through explicit consent.
Lawfulness
Integrity
Storage limitations
Purpose limitations
Data minimisation
Data accuracy
GDPR
Privacy
Implement PbD, backup and DR methods to ensure data security and resil-lience. Don't compromise users' identities.
Controls
Only collect the data you need and ensure that it is accessible by users so they can update it. Maintain retention policies.
Accountability

- **Audit Trails:** All data transformations, feature engineering steps, and model training processes will be logged for auditability, ensuring transparency and accountability.
- **Retention Policies:** Historical data and model versions will be retained only as long as necessary for operational and compliance purposes, adhering to local and international data retention laws.

2. Data Security

- **Encryption:** All customer data in transit and at rest will be encrypted using advanced encryption protocols (e.g., TLS 1.2+ for transmission and AES-256 for storage).
- **Access Controls:** Role-based access controls (RBAC) will be implemented to limit access to sensitive data and pipeline components. Only authorized personnel will have access to production systems and datasets.
- **Anonymization:** Personally Identifiable Information (PII) will be anonymized or pseudonymized to minimize risk and ensure data is used responsibly during analysis and model training.



3. Model and Pipeline Security

- **Secure Deployment:** Models will be deployed in a Kubernetes environment configured with strict security policies, such as limiting container privileges and enforcing network segmentation.

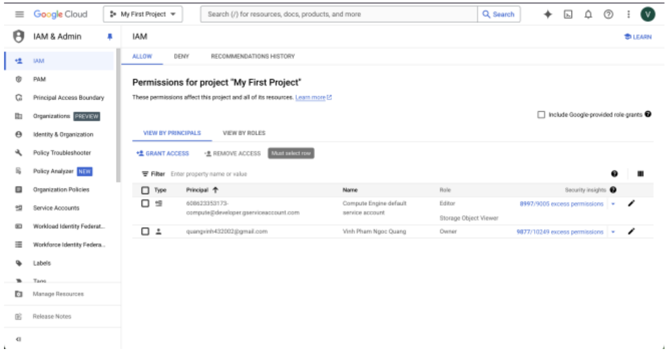
- **Runtime Protection:** Container runtime security tools (e.g., Falco or AppArmor) will be integrated into the Kubernetes cluster to detect and mitigate unauthorized activities.
- **Dependency Management:** All dependencies and libraries used in the pipeline will be scanned for vulnerabilities using tools like Snyk or Trivy to ensure secure software components.

4. Monitoring and Incident Response

- **Continuous Monitoring:** Real-time monitoring tools will track data access, model performance, and system anomalies. Alerts will be configured to notify the team of suspicious activities.
- **Incident Response Plan:** A robust incident response plan will be established to address data breaches, unauthorized access, or system failures promptly, minimizing potential damage.

5. Cloud Security

- **Google Cloud Platform (GCP) Compliance:** All components of the pipeline hosted on Google Cloud, such as Kubernetes (GKE) and feature stores (Feast), will adhere to GCP's compliance frameworks, including SOC 2, ISO 27001, and PCI DSS.
- **Identity and Access Management (IAM):** IAM policies will ensure that cloud resources are accessible only to authenticated and authorized users, reducing the attack surface.



Permissions for project "My First Project"

These permissions affect this project and all of its resources. Learn more

Type	Principal	Name	Role	Security insights
	608623353173-compute@developer.gserviceaccount.com	Compute Engine default service account	Editor Storage Object Viewer	8997/9005 excess permissions
	quangvinh432002@gmail.com	Vinh Pham Ngoc Quang	Owner	9877/10249 excess permissions

VI. Results & Discussion

The results from the model evaluation and the deployment of the MLOps pipeline demonstrate significant strides in predicting customer churn in the banking sector. This section presents the findings of the model evaluation, their implications, and discusses the benefits and challenges of implementing an MLOps framework.

Results

The models trained for churn prediction, including Logistic Regression, Random Forest, and XGBoost, were evaluated using various metrics such as AUC-ROC, precision, recall, F1-score, and accuracy. Key insights are as follows:

1. **Logistic Regression:**
   - **Performance:** Logistic Regression provided a strong baseline with an **AUC-ROC of 0.92**, demonstrating good discriminatory power between churners and non-churners.
   - **Strengths:** Simplicity and interpretability make it a useful benchmark for comparison with more advanced models.
   - **Weaknesses:** The model struggled to recall non-churners, which could result in missing key retention opportunities.
2. **Random Forest:**
   - **Performance:** Random Forest achieved an **AUC-ROC of 0.97**, indicating excellent predictive capability.
   - **Strengths:** Balanced performance across precision and recall, with robust handling of both categorical and numerical data. Its feature importance analysis identified key drivers of churn, such as **tenure** and **customer engagement metrics**.
   - **Challenges:** Computational cost was higher due to the ensemble nature of the algorithm.
3. **XGBoost:**
   - **Performance:** XGBoost outperformed all other models with an **AUC-ROC of 0.98**. It showed the highest accuracy, recall, and precision among the three models.
   - **Strengths:** Superior handling of imbalanced data, precise predictions, and consistent performance across multiple metrics.
   - **Feature Importance:** The most significant predictors were **contact_time_numeric** and **record_start_date_numeric**, highlighting the importance of temporal data in churn prediction.
   - **Challenges:** Hyperparameter tuning required additional computational

resources, which was mitigated by utilizing Google Cloud.

Discussion

1. **Model Performance:**
   - All three models demonstrated strong capabilities in predicting customer churn, with XGBoost standing out as the most effective due to its high precision and recall scores.
   - The use of feature importance metrics highlighted key drivers of churn, enabling actionable insights for targeted customer retention strategies.
2. **Impact of MLOps Framework:**
   - The integration of **Kubeflow** streamlined the machine learning lifecycle, automating training, validation, and deployment. This reduced the manual effort required and ensured consistent reproducibility.
   - **Feast** as a feature store ensured consistency between training and serving features, eliminating potential training-serving skew and improving real-time prediction accuracy.
   - **Google Cloud's Kubernetes Engine (GKE)** provided the scalability and robustness necessary to handle large datasets and computationally intensive tasks like hyperparameter tuning and model training.

3. **Scalability and Flexibility:**
   - By deploying on Google Cloud, the pipeline achieved dynamic scalability, enabling real-time churn prediction for millions of customers without latency or performance bottlenecks.
   - The use of containerization (Docker and Kubernetes) ensured portability, making the pipeline adaptable to different environments if needed.
4. **Challenges and Limitations:**
   - **Data Quality:** Issues such as outliers and missing values required extensive preprocessing, which added complexity to the pipeline.
   - **Imbalanced Data:** Despite the success of techniques like oversampling and advanced algorithms like XGBoost, imbalanced classes posed challenges in model evaluation.
   - **Computational Costs:** While Google Cloud mitigated local resource limitations, cloud infrastructure costs can become significant over time.
5. **Business Implications:**
   - The insights gained from the models can inform personalized customer retention strategies, targeting high-risk customers with tailored offerings to reduce churn.
   - The ability to monitor and retrain models in real-time ensures sustained predictive performance, adapting to changing customer behaviors.

VII. Conclusion

The research highlights the effectiveness of integrating MLOps frameworks in predicting customer churn within the banking sector. By automating the machine learning lifecycle, from data preprocessing to model deployment, the pipeline demonstrates the following:

1. **Improved Model Accuracy:** Advanced algorithms like XGBoost achieved an AUC-ROC of 0.98, ensuring precise identification of at-risk customers.
2. **Scalability:** Leveraging Google Cloud and Kubernetes enabled seamless scaling of the pipeline to handle large datasets and real-time predictions.
3. **Operational Efficiency:** Automation with Kubeflow reduced manual effort in training, validating, and deploying models, while Feast ensured feature consistency.
4. **Actionable Insights:** Feature importance analysis provided valuable insights into key churn drivers, guiding data-driven customer retention strategies.

However, challenges such as data quality issues, imbalanced datasets, and the computational cost of cloud infrastructure were identified, necessitating continuous optimization. Despite these limitations, the implementation showcases how MLOps frameworks can transform customer churn prediction into a scalable, reproducible, and impactful process for the banking sector.

VIII. Future Work

The following directions are proposed for extending the research:

1. **Pipeline Optimization:**
    - Implement advanced hyperparameter tuning techniques, such as Bayesian optimization, to further improve model performance.
    - Explore distributed training frameworks for handling larger datasets more efficiently.
2. **Model Diversity:**
    - Incorporate deep learning models, such as neural networks, for processing unstructured data like customer feedback or transaction descriptions.
    - Investigate hybrid modeling approaches combining machine learning and rule-based systems.
3. **Feature Expansion:**
    - Integrate external datasets, such as market trends or credit bureau data, to enhance the feature space.
    - Develop features using natural language processing (NLP) techniques for text-based customer interactions.
4. **Real-Time Enhancements:**
    - Experiment with low-latency systems for real-time churn prediction using streaming data.
    - Implement drift detection mechanisms to ensure model performance remains robust over time.
5. **Wider Applications:**
    - Extend the pipeline to predict other outcomes, such as customer lifetime value or cross-sell opportunities, within the banking domain.
    - Adapt the methodology for other industries, such as telecommunications or retail, where churn prediction is equally critical.

This future work aims to build on the successes of the current pipeline, addressing its limitations and extending its capabilities for broader applications.

VIII. References

[1] Goyal, "A Comprehensive Guide on MLOps for Machine Learning Engineering," Analytics Vidhya, Mar. 2022. Available: https://www.analyticsvidhya.com/blog/2022/03/a-comprehensive-guide-on-mlops-for-machine-learning-engineering/

[2] H. D. Tran, N. Le, and V. Nguyen, "Customer Churn Prediction in the Banking Sector Using Machine Learning-Based Classification Models," Interdisciplinary Journal of Information, Knowledge, and Management, vol. 18, pp. 87-105, 2023. Available: https://www.informingscience.org/Publications/5086

[3] D. Kreuzberger, N. Kühl, and S. Hirschl, "Deployment of ML Models using Kubeflow on Different Cloud Providers," arXiv preprint, May 2022. Available: https://ar5iv.labs.arxiv.org/html/2206.13655

[4] P. Klushin, "How Feature Stores Enhance Model Performance in ML Technology," Future with Tech, Nov. 2023.

[5] Kubernetes "Changing the Container Runtime on a Node from Docker Engine to containerd" July. 2023. Available: https://kubernetes.io/docs/tasks/administer-cluster/migrating-from-dockershim/change-runtime-containerd/

[6] Kubeflow Definition - Wikipedia Available: https://en.wikipedia.org/wiki/Kubeflow

[7] TheDeepHub. (n.d.). *Kubernetes for Data Engineers*. Medium. https://medium.com/thedeephub/kubernetes-for-data-engineers-9ba95d338cfe

[8] DeployKF. (n.d.). *Local Quickstart*. https://www.deploykf.org/guides/local-quickstart/

[9] Jayesh, W. (n.d.). *Putting an ML Model into Production Using Feast and Kubeflow on Azure - Part I*. Dev.to. https://dev.to/wjayesh/putting-an-ml-model-into-production-using-feast-and-kubeflow-on-azure-part-i-3i33

[10] DagsHub. (n.d.). *How to Install Kubeflow Locally*. https://dagshub.com/blog/how-to-install-kubeflow-locally/

[11] Kubeflow. (n.d.). *Katib Overview*. Retrieved https://www.kubeflow.org/docs/components/katib/overview/