Table of Content

Optimizing Heart Valve Replacements: Building a Distributed Data Science Pipeline for CardioValve Solutions

Authors: Niat Kahsay and Marwah Faraj

Company Name: CardioValve Solutions

Company Industry: Healthcare / Medical Devices

Company Size: Small healthcare company (20 employees)

Abstract

CardioValve Solutions is a company that develops heart valve implants and struggles with determining the likelihood of success of heart valve replacement. To optimize decision-making and improve patient outcomes, the company wanted to build a distributed data science pipeline to analyze large volumes of patient and surgical data.

Problem Statement

CardioValve Solutions aggregates data covering hundreds of thousands of records daily in numerous different formats, ranging from patient demographic information to surgical and post-operative details. Even with this wealth of data at its disposal, the company has difficulty identifying the most reliable predictors of successful heart valve replacements. Without predictive insight from the past data, it results in sub-optimal patient care and inevitable operational waste which may jeopardize the company's competitive edge in the fast-evolving healthcare scenario. Increasingly, in order to make data-driven decisions, a solid, scalable cloud-based data pipeline is required delivering critical insights to support incremental improvement in clinical performance and operational efficiency.

Goals

1. Develop a distributed data pipeline that efficiently handles large volumes of clinical data.
2. Identify key predictive factors influencing valve replacement success.
3. Improve patient outcomes through data-driven decision-making and clinical insights.

Non-Goals

This project focuses on developing a distributed data science pipeline to optimize heart valve replacement outcomes at CardioValve Solutions. However, it does not involve creating or modifying heart valve devices, conducting new clinical trials, or integrating third-party data beyond specified sources. By maintaining these boundaries, the project ensures a focused, data-driven approach to improving decision-making and patient outcomes.

Data Sources

Data is securely stored in an AWS S3 Bucket, which is communicated with AWS SageMaker for model training. The following datasets are uploaded to the S3 bucket:

- s3://cardiovale-solutions-datascience-pipeline/raw-data/cardio_train.csv
- s3://cardiovale-solutions-datascience-pipeline/raw-data/Quitline___Services_Available__Medications_-_2010_To_Present_20250306.csv

AWS services, including Amazon SageMaker Autopilot, are utilized to train models efficiently. To ensure compliance with HIPAA and other data security regulations, strict data validation, encryption, and access controls are implemented to maintain data integrity and security.

Data Exploration

An Amazon S3 bucket “cardiovale-solutions-data science-pipeline” is created to store the datasets, organized within a structured directory. Data ingestion is performed using AWS Glue, which facilitated the extraction, transformation, and loading (ETL) process. The data is then retrieved using the AWS SDK (boto3) and saved as df_cardio in a local SageMaker Studio directory for processing. AWS Athena, a serverless interactive query service that enables SQL-based data analysis in Amazon S3 (Amazon Web Services, n.d.), is used to create the database and merge multiple files into a unified dataframe. After processing, the data is stored back in S3 or utilized in SageMaker’s EFS for model training.

Exploratory Data Analysis (EDA)

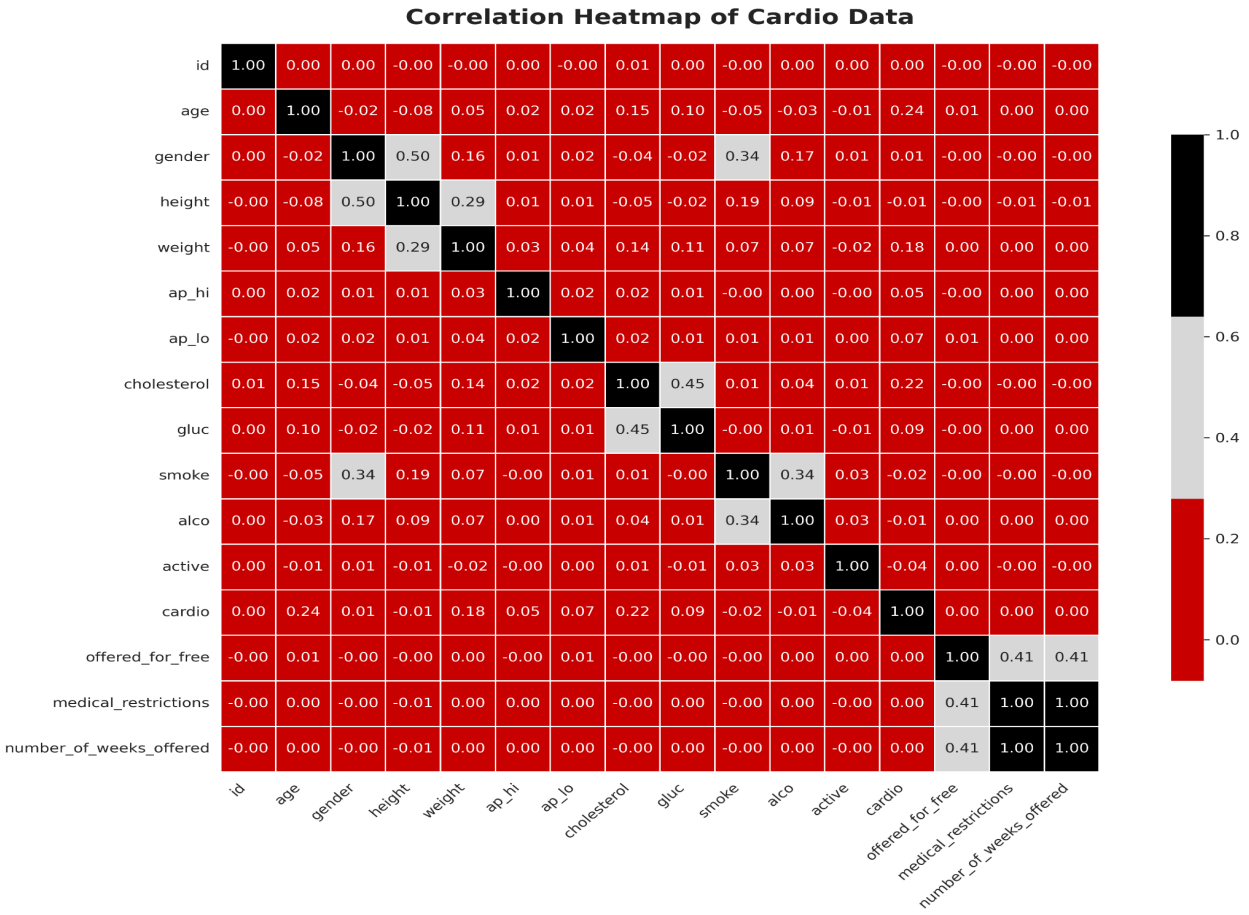
During the data exploration phase, data quality, potential biases, key fields, and data types are assessed to ensure the datasets are suitable for analysis and modeling. Missing values are checked using .isnull().sum() to identify incomplete records, and duplicates are examined to maintain data integrity. Data consistency is verified, ensuring height and weight values fell within reasonable ranges. The key fields for the Cardio dataset included id, age, gender, height, weight, cholesterol, smoke, and cardio, which are essential for predicting cardiovascular risk. For the Quitline services dataset, key fields such as state, year, medication_available, and medication_type are analyzed to assess service accessibility.

A correlation heatmap is generated to examine the linear relationships between numerical features in the dataset. The matrix revealed moderate positive correlations between variables such as height and weight (0.29), and cholesterol and glucose (0.45). A low correlation (0.22) is observed between cholesterol and the target variable cardio, suggesting a potential

predictive relationship. Most other features showed weak or negligible correlations, indicating limited multicollinearity.

Figure 1

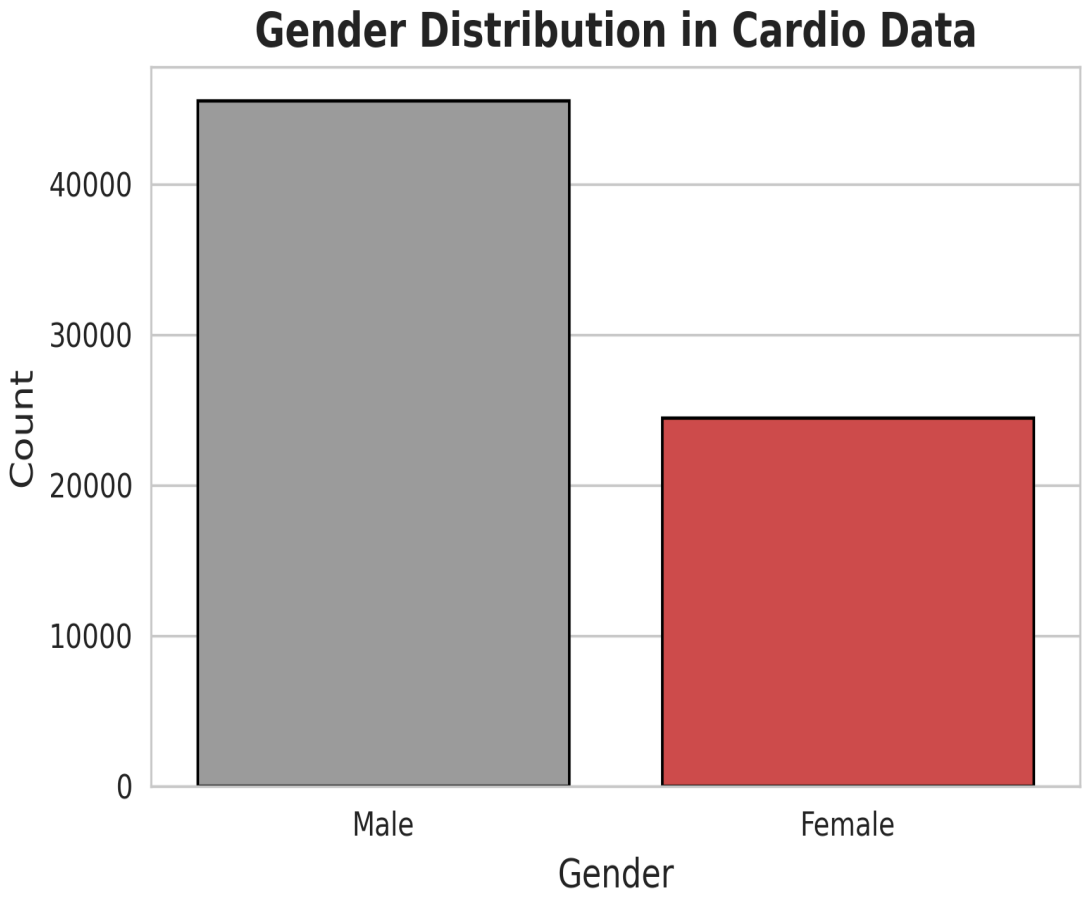
Heatmap of Cardio Data



The distribution of gender within the dataset is analyzed and visualized. The gender variable is mapped from numeric codes to categorical labels ("Male" and "Female") for clarity. The analysis revealed that approximately two-thirds of the records corresponded to male patients, while one-third are female. This imbalance in gender representation is noted and considered during model development to ensure fairness and mitigate potential bias.

Figure 2

Bar Graph of Gender Distribution in Cardio Data



Data Ingestion

GitHub Repository Information

- **Main notebook (.ipynb file)**:https://github.com/NiatKahsay/ads-508-team/tree/main/notebooks
- **Main notebook.(pdf file):** **https://github.com/NiatKahsay/ads-508-team/tree/main/Documents**

Bias Exploration

Bias analysis is conducted using Amazon SageMaker Clarify, focusing on gender as the sensitive feature. The analysis indicated a class imbalance (CI) score of -0.301, suggesting that one gender is underrepresented in the dataset. The Difference in Proportions of Labels (DPL) score of 0.009 suggested a slight disparity in positive outcomes (cardio=1) between gender groups. Additionally, the Kolmogorov-Smirnov (KS) and Total Variation Distance (TVD) metrics both reported a value of 0.009, indicating minimal divergence in label distribution between gender groups. The Kullback-Leibler (KL) and Jensen-Shannon Divergence (JS) scores were both 0.000, showing no significant entropy-based divergence. Despite these relatively low bias scores, the detected class imbalance suggested that gender representation in the dataset could influence model performance.

Data types are examined to ensure numerical fields (age, height, and weight) are correctly formatted as integers or floats. Categorical fields (gender and cholesterol) are considered for encoding techniques such as one-hot encoding or label encoding to prepare them for machine learning models. These assessments allowed for necessary data cleaning and preprocessing, enhancing the dataset's quality and fairness before further analysis and modeling (Mihan & Pandey, 2024).

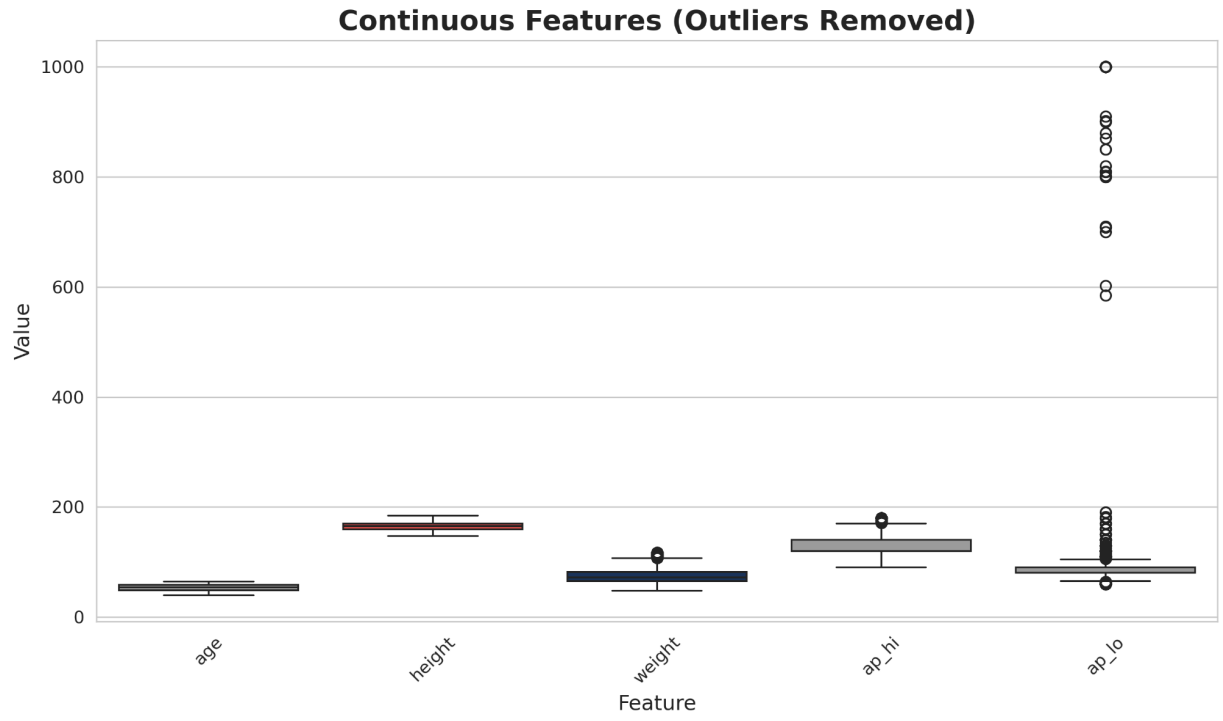
Class Imbalance

A box plot was generated to visualize the distribution of continuous features (age, height, weight, ap_hi, and ap_lo) after outliers are clipped to the 1st and 99th percentiles. Age is converted to years for better interpretability. Each feature exhibited a reasonable central tendency, though some remaining outliers are visible, particularly in blood pressure values (ap_hi

and ap_lo). The plot highlighted variability in height and weight across the population and confirmed that extreme values are successfully reduced, improving model readiness.

Figure 3

Boxplot of Continuous Features



Measuring Impact

For this project, two key metrics are anticipated to improve, reflecting the goal of enhancing data quality. First, a reduction in missing data is expected as incomplete records are addressed through imputation or removal, resulting in a lower percentage of missing values. Second, an increase in data quality scores is targeted by resolving issues such as outliers and incorrect data types, improving overall consistency and accuracy. These metrics demonstrated the effectiveness of the data cleaning and preprocessing efforts, leading to better analysis and model performance.

Security Checklist, Privacy and Other Risks

This project did not store or process any Protected Health Information (PHI) or Personally Identifiable Information (PII) at any stage. Additionally, no user behavior is tracked or recorded as part of the data pipeline. Likewise, no credit card data is stored or processed, ensuring strict compliance with data privacy and security standards.

To assess potential biases, selection bias in the Cardio dataset was analyzed to determine whether demographic distribution led to the overrepresentation of certain populations. Similarly, geographic bias in the Quitline services dataset is examined, as service availability may have been disproportionately lower in rural areas (Pandey & Van Spall, 2024).

Ethical considerations centered on identifying and mitigating potential biases that could impact the fairness and accuracy of the analysis. These biases are carefully evaluated to promote equitable outcomes and prevent the reinforcement of harmful stereotypes or unequal treatment of specific demographic groups.

Data Preparation

Feature Store Validation

After ingestion, the Feature Group is verified via the SageMaker Studio UI.

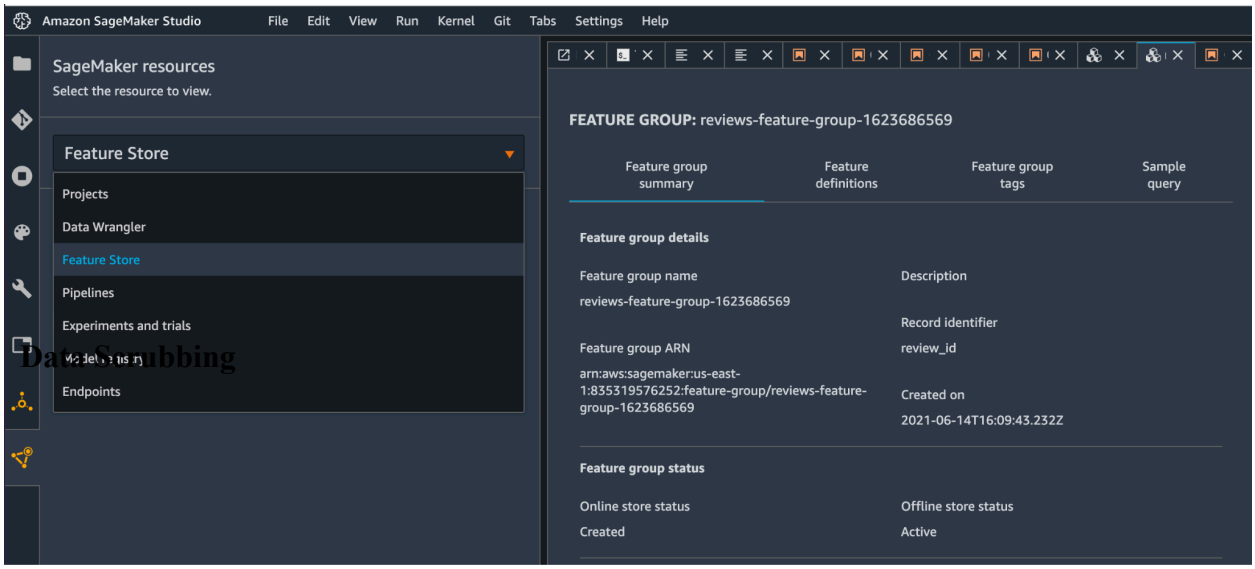
Using the Feature Store pane, the cardio-feature-group-22-21-14-34 is inspected to confirm:

- The record identifier (id) and event_time were correctly assigned.
- All feature definitions are registered, including the derived split_type field.
- Offline store status is Active, confirming data readiness for querying and training.

This visual verification step helped ensure proper schema alignment before running Athena queries or launching model training jobs.

Figure 4

Schema Alignment



Data Scrubbing

Several data cleansing techniques are applied to ensure high-quality training data for the cardiovascular risk prediction model. Missing values in the merged dataset are addressed by replacing NaN values in the number_of_weeks_offered column with 0, preserving valuable records and preventing a significant reduction in dataset size that could introduce selection bias. To align with SageMaker Feature Store requirements, column names arestandardized to lowercase with underscores (e.g., offered_for_free instead of Offered_for_Free), ensuring seamless integration with AWS services and maintaining consistency throughout the pipeline.

Additionally, a timestamp column named event_time is created using the current UTC time to enable chronological tracking in the offline Feature Store. This step is crucial for SageMaker Feature Store functionality, supporting time-based feature evolution tracking.

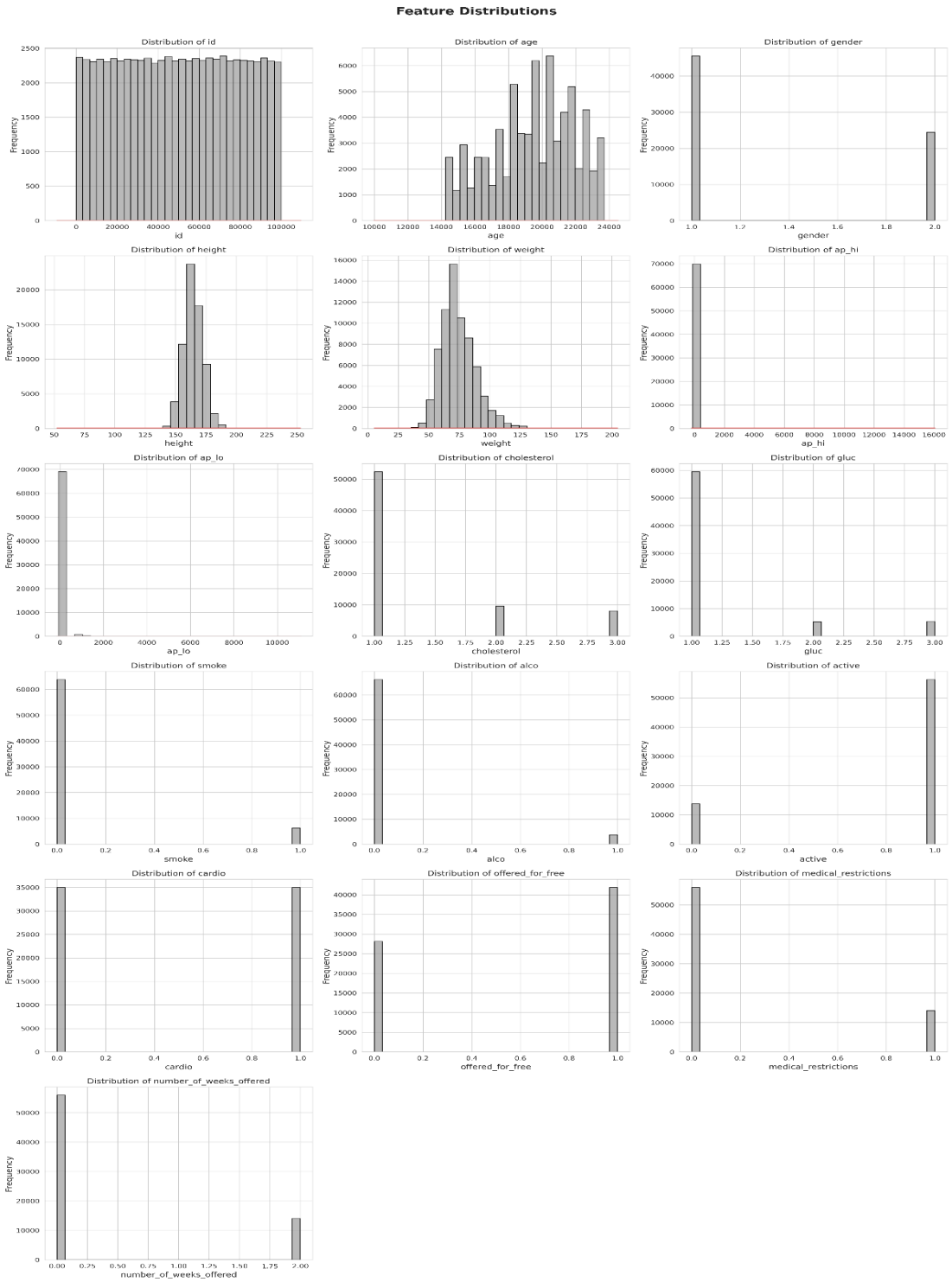
Feature Selection

A subset of relevant fields is carefully selected based on the objective of predicting cardiovascular risk. From the cardio dataset, demographic and clinical indicators known to be correlated with heart conditions, as identified in medical literature, are included: age, gender, weight, height, ap_hi (systolic blood pressure), ap_lo (diastolic blood pressure), cholesterol, and smoke. Fields exhibiting high multicollinearity with other features or showing low correlation with the target variable are excluded. For instance, alco (alcohol intake) demonstrated minimal correlation with the cardio target during exploratory analysis and is therefore removed to simplify the model.

Additionally, state-level public health attributes from the Quitline services dataset are incorporated to provide geographical context to patient risk factors. Features such as offered_for_free, medical_restrictions, and number_of_weeks_offered are selected based on clinical domain knowledge and initial correlation analysis.

Figure 5

Histogram Distributions



The distribution of features within the Cardio dataset is analyzed to understand their variability and potential impact on predictive modeling. As the figure above shows, several continuous variables, such as height, weight, and age, are found to exhibit approximately normal distributions, while categorical and binary features, including cholesterol, smoking, and alcohol consumption, displayed distinct peaks, indicating the presence of discrete classes. Notably, the blood pressure attributes (ap_hi and ap_lo) contained extreme values, suggesting that potential outliers needed to be addressed during preprocessing. Additionally, features such as cholesterol, glucose, and physical activity are observed to have skewed distributions, which could have influenced model performance if not appropriately transformed. Evaluating these distributions is considered essential in feature selection, as variables with minimal variance or highly imbalanced classes might have contributed less to the predictive power of a model and introduced bias. Through this analysis, features requiring scaling, transformation, or removal are identified to enhance model efficiency and accuracy.

Feature Creation

Several new fields are created to enrich the dataset and enhance its predictive power. Since location data is absent in the cardio dataset, a synthetic State column is generated by randomly assigning U.S. state abbreviations to each record. Although this assignment was artificial, it enabled the merging of state-level attributes from the Quitline dataset into individual patient records, serving as a proxy for regional health service availability, which could influence cardiovascular outcomes.

A bmi feature is also engineered by calculating the Body Mass Index using the standard formula: weight (kg) / height (m)². This derived feature provided a standardized measure of

obesity, a known risk factor for cardiovascular disease. Additionally, a split_type column is introduced to flag each record's partition (train, validation, or test) for use in SageMaker Autopilot's downstream processing.

Feature Transformation

To conform with SageMaker requirements, several transformations are performed. All object-type columns are cast to appropriate types for compatibility with the Feature Store. Categorical fields such as State and split_type are converted to String type, while numerical fields like age, ap_hi, and cholesterol are typed as Integral. Features with continuous values, such as weight and number_of_weeks_offered, are defined as Fractional.

Data Preparation

Continuous variables are normalized to have a zero mean and unit variance using scikit-learn's StandardScaler, which is applied to enhance performance for distance-based algorithms. Categorical variables are numerically encoded, with binary categories such as gender mapped to 0 and 1, while multi-categorical features are designated for one-hot encoding during the model training stage, depending on the algorithm selected by AutoML. These transformations are implemented to ensure that all features contributed equally to the model and to prevent features with larger scales from dominating the learning process.

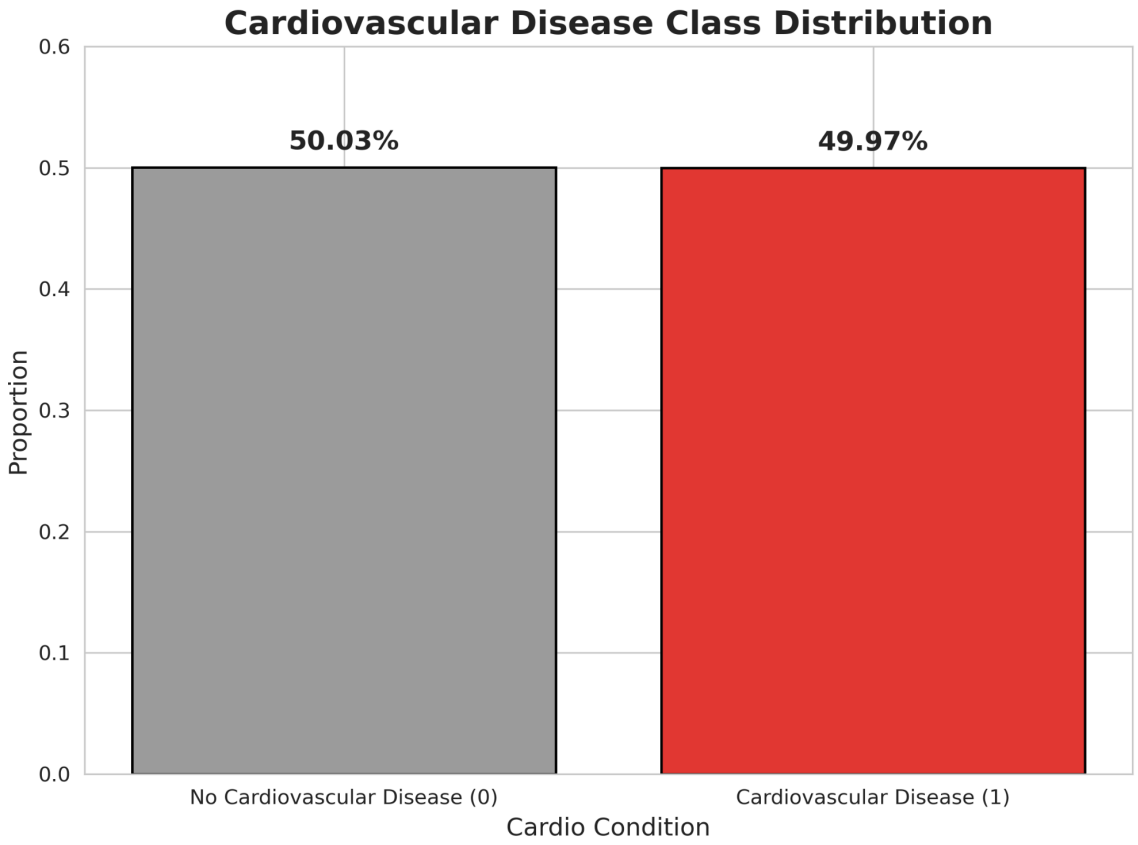
Balancing the Dataset

A class distribution check is performed on the target variable cardio using value_counts(), and the results are visualized with a bar chart. The distribution revealed 35,021 records for class 0 (no cardiovascular disease) and 34,979 for class 1 (cardiovascular disease

present), accounting for approximately 50.03% and 49.97%, respectively, as displayed in figure 2. Since the dataset is nearly perfectly balanced between the two classes, with only a 0.06% difference, no additional class balancing techniques such as SMOTE, ADASYN, or undersampling are deemed necessary. This natural balance is considered beneficial, as it allowed models to be trained without the potential distortions introduced by synthetic sampling methods.

Figure 6

BarGraph of Cardiocascular Disease Class Distribution



Train, Test, and Validation Splits

To prevent overfitting and underfitting, a stratified splitting strategy is implemented to maintain the class distribution across all subsets of data. Using scikit-learn's train_test_split function with the stratify parameter set to the cardio target column, 70% of the data is allocated to the training set, while the remaining 30% is evenly divided into validation and test sets, each comprising 15% of the data. This stratification ensured that class proportions remained consistent across all subsets, reducing the risk of biased model performance (Shedden, 2020–2021). The 70-15-15 split ratio is selected to provide a sufficient amount of data for model training while reserving adequate samples for validation and final evaluation.

The stratified approach preserved the proportional representation of each target class within all subsets, which is crucial for maintaining model generalizability in cardiovascular disease prediction (Menon, 2020). To facilitate dataset partitioning, a new split_type column is assigned to each record, labeling it as either 'train,' 'validation,' or 'test.' These subset are then concatenated into a single DataFrame, as required by SageMaker Autopilot, which handled the splits internally during the AutoML process. Finally, the consolidated dataset is saved as a CSV file and uploaded to the designated Amazon S3 bucket, ensuring seamless integration into the model training pipeline.

Data Training and Modeling

The final model is developed using Amazon SageMaker Autopilot in combination with SageMaker JumpStart, and both AutoML and prebuilt algorithm workflows are leveraged to accelerate experimentation and improve performance. SageMaker Autopilot is used to orchestrate the initial training process, during which key stages such as data preprocessing,

feature engineering, candidate model selection, and hyperparameter tuning are automated. This end-to-end automation is designed to enable the exploration of high-performing models without extensive manual intervention.

The training process is initiated in SageMaker Studio, where a curated dataset containing labeled cardiovascular outcomes is uploaded in CSV format to an Amazon S3 bucket for processing. The training dataset is stored at the following location: s3://cardiovale-solutions-datascience-pipeline/feature-store/cardio/cardio-feature-group-22-21-14-34/autopilot_input.csv.

An Autopilot job is configured under the name cardio-autopilot-model-job, and it is scheduled to run for 3600 seconds (1 hour) to generate three candidate models. The optimization criterion is set to F1 Score, as it is considered ideal for the binary classification task of identifying cardiovascular disease risk—particularly when minimizing false positives and false negatives is deemed critical.

During candidate generation, several algorithms—including XGBoost, Linear Learner, and Tabular Neural Networks—are evaluated by Autopilot. Based on comparative performance, XGBoost is identified as the top-performing algorithm. Its ability to handle heterogeneous features, missing values, and moderate class imbalance is recognized as particularly well-suited for the characteristics of the healthcare dataset.

To further validate and enhance the training process, the solution is extended using SageMaker JumpStart. A custom XGBoost training job is configured and launched through JumpStart, using verified training and validation inputs. This step is taken to ensure alignment

with the model previously selected by Autopilot, while also allowing more granular control over hyperparameters and model convergence.

The final model is trained under the name cardio-xgboost-final-2025-04-05-21-28-17-356 using JumpStart. A balanced class distribution is confirmed, and strong convergence behavior is observed, with the training error being reduced from 0.26327 to 0.23780 over the first 50 iterations. These results are used to validate the selection made by Autopilot and are taken as confirmation that the model is ready for deployment.

Hyperparameter tuning, initially handled by Autopilot, is defined using the following critical settings:

- max_depth = 6
- learning_rate = 0.3
- num_boost_round = 5000
- early_stopping_rounds = 30

These settings are retained during the JumpStart training to maintain performance consistency across both environments.

Instance Configuration

The training and preprocessing tasks are executed on ml.m5.large instances, which are selected for their balance between compute power and cost-effectiveness. The training job utilized a 2-instance cluster, equipped with 110 GB of Elastic Block Store (EBS) storage. This configuration is ideal for processing large datasets while maintaining an efficient use of computational resources. Notably, the job did not require GPU support, as the computational

needs are sufficiently met with general-purpose instances. The default VPC configurations are used, and inter-container traffic encryption is disabled to optimize runtime performance.

Model Evaluation

Once training is completed, the performance of the final model—first trained via Amazon SageMaker Autopilot and then validated and enhanced through SageMaker JumpStart—is evaluated using a reserved validation dataset. The evaluation phase is conducted in two stages: first using metrics automatically generated by the Autopilot job artifacts, and then through a more detailed performance breakdown obtained after retraining using JumpStart.

From the Autopilot job (cardio-autopilot-model-job2t1iNP-001-9db02329), the following performance metrics were reported:

Table 1

Model Evaluation

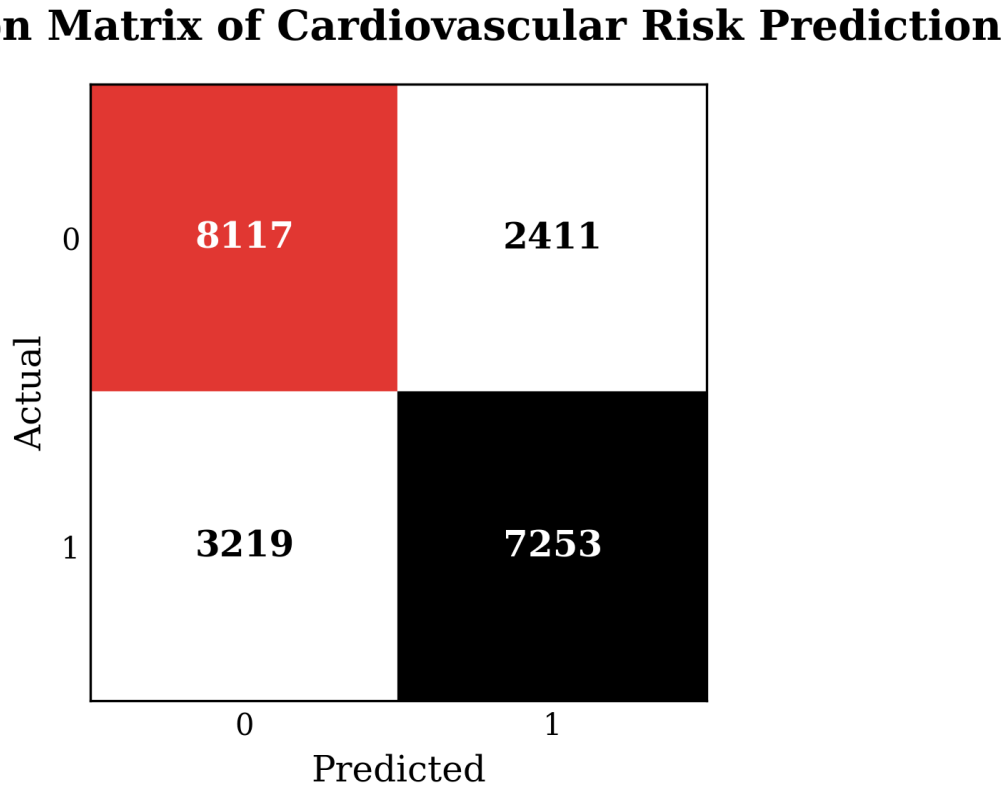
Metric	Value
F1 Score	0.717
AUC	0.793
Accuracy	0.726
Precision	0.741
Recall	0.694
Balanced Accuracy	0.726
LogLoss	0.554

These metrics underscored the model's initial strong performance in identifying cardiovascular disease risk. High AUC and F1 scores indicated reliable classification across both positive and negative cases—an essential requirement in a healthcare context.

After the model is retrained using SageMaker JumpStart for further tuning and validation, additional evaluation metrics are captured using a confusion matrix and a classification report on the same validation set (21,000 records). These results confirmed Autopilot's selection and demonstrated even more balanced performance:

Table 2
Confusion Matrix

Confusion Matrix



The confusion matrix above summarizes the classification results of the final XGBoost model trained using SageMaker Autopilot and JumpStart. The matrix displays the actual versus predicted values for the binary classification task of identifying cardiovascular risk. Out of 21,000 validation samples, the model correctly classified 8,117 individuals without cardiovascular disease (true negatives) and 7,253 individuals with cardiovascular disease (true positives). However, 2,411 cases were false positives—where individuals without the disease were incorrectly classified as having it—and 3,219 cases were false negatives, where actual cardiovascular conditions were not detected by the model. False negatives could delay life-saving interventions. To mitigate this, we recommend combining model predictions with physician judgment for borderline cases.

These results show a strong overall accuracy of 73%, with a balanced performance between precision and recall. The relatively higher number of false negatives (3,219) compared to false positives emphasizes the importance of ongoing tuning to reduce missed diagnoses. In medical contexts, particularly in cardiovascular disease detection, false negatives can lead to delayed interventions and higher health risks. The model's ability to correctly identify a large portion of positive cases, despite these limitations, confirms its practical utility as a supportive tool for early risk assessment and triage.

Table 3

JumpStart Classification Report

Metric	Class 0	Class 1	Average

Precision	0.72	0.75	0.73 (macro)
Recall	0.77	0.69	0.73 (macro)
F1 Score	0.74	0.72	0.73 (macro)
Accuracy			0.73

These results reinforced the model's ability to maintain balanced performance across both classes. Notably, class 0 (no cardiovascular disease) had a higher recall (0.77), while class 1 (cardiovascular disease present) demonstrated a higher precision (0.75). This balance is essential in clinical decision-making, where both minimizing false negatives and false positives is vital.

The F1 score continued to serve as a reliable measure by harmonizing precision and recall, especially under slightly imbalanced or high-risk conditions. A consistent accuracy of 0.73 across both Autopilot and JumpStart also confirmed the model's robustness. Therefore, the final model is deemed suitable for deployment in supporting cardiovascular risk prediction scenarios in real-world healthcare applications.

Future Enhancements

1. Incorporating Real-Time Data Streams from Clinical Devices

Currently, the model is trained on static historical data. A powerful enhancement would be integrating real-time data ingestion from wearable devices or hospital monitoring systems. Streaming data pipelines using Amazon Kinesis or AWS IoT could enable real-time risk assessment and timely alerts for patients undergoing heart valve replacements. This would allow clinicians to proactively intervene, potentially preventing post-operative complications. The dynamic nature of real-time data would also improve the model's adaptability and keep predictions aligned with evolving patient conditions.

2. Advanced Explainability through SHAP and SageMaker Clarify Integration

While the current model delivers strong performance, its interpretability could be further enhanced using SHAP (SHapley Additive exPlanations) values. By integrating SHAP into the pipeline via SageMaker Clarify, stakeholders—including clinicians—could gain a transparent view into why the model predicts a high cardiovascular risk for a given patient. This step is particularly critical in healthcare, where explainability builds trust and facilitates data-driven clinical decisions. SHAP plots would visually demonstrate feature importance at both global and individual levels, helping to identify actionable factors influencing model output.

3. Multi-Modal Data Integration and Ensemble Modeling

Another promising enhancement involves expanding the dataset beyond tabular data by incorporating unstructured data such as clinical notes or echocardiogram images.

This could be achieved using Amazon Comprehend Medical for NLP on text or SageMaker Ground Truth for medical image labeling. Integrating these modalities through an ensemble learning approach (e.g., combining XGBoost with CNN models or transformer-based text models) would enrich the feature space and boost predictive performance. This hybrid approach would simulate how a medical professional synthesizes various information types, offering a more holistic view of patient risk.

Clinical Implementation

To ensure the predictive model adds real-world value in clinical settings, CardioValve Solutions has developed a comprehensive implementation framework that integrates seamlessly into hospital workflows while prioritizing compliance, usability, and trust. The model will be deployed as an AWS SageMaker endpoint, enabling Electronic Health Record (EHR) systems to request real-time predictions through a secure REST API. As clinicians access patient profiles, the system automatically retrieves and displays cardiovascular risk scores, stratified into actionable categories: low risk (score < 0.3) requires routine monitoring, moderate risk (0.3–0.7) prompts further diagnostic testing, and high risk (≥ 0.7) triggers specialist review and potential intervention adjustments. Clinicians will also have access to a dedicated AWS QuickSight dashboard, which visualizes patient-specific risk factors, historical trends, and comparative analysis with similar demographics. Alerts and evidence-based recommendations are integrated with hospital paging systems, offering guidance such as statin therapy for high-risk patients or Quitline referrals for smokers. A pilot phase will be launched in 3–5 partner hospitals to collect clinician feedback, with the model undergoing retraining every six months to reflect updated surgical outcome data. All predictions and resulting clinical decisions will be logged to monitor for bias and support auditability. Ethical safeguards, including informed patient consent and

override options for clinicians, are built into the system, with future plans to incorporate SHAP-based explainability reports. For example, in pre-surgical planning, a 62-year-old male with a risk score of 0.82 might have his surgery postponed for targeted risk management, while post-operative alerts—such as a spike in risk score—can prompt timely follow-up diagnostics. Anticipated challenges, such as clinician skepticism and data drift, are mitigated through strategies like co-developing interfaces with physicians and quarterly feature monitoring. Overall, this approach reinforces the model's role as a support tool that enhances, rather than replaces, clinical decision-making.

Conclusion

This project successfully demonstrated how a distributed data science pipeline can be built using AWS services—including SageMaker Autopilot, JumpStart, Feature Store, and Clarify—to predict cardiovascular risk with high accuracy. By integrating multiple datasets, performing rigorous feature engineering, and validating results through automated and custom workflows, CardioValve Solutions now has a scalable ML model that can assist clinicians in optimizing heart valve replacements. The model achieved a 0.73 F1 Score and demonstrated balanced performance, making it viable for real-world healthcare applications. With future enhancements such as real-time data integration, model explainability, and multi-modal learning, the pipeline is positioned for even greater clinical impact.

References

Amazon Web Services. (n.d.). Amazon Athena. https://aws.amazon.com/athena/?whats-new-cards.sort-by=item.additionalFields.postDateTime&whats-new-cards.sort-order=desc

Fatmanurkutlu. (2024, January 11). *Model evaluation techniques in machine learning.* Medium. Retrieved from https://medium.com/@fatmanurkutlu1/model-evaluation-techniques-in-machine-learning-8cd88deb8655

Kundu, R. (2022, December 16). What is F1 score? *V7.* Retrieved from https://www.v7labs.com/blog/f1-score-guide

Menon, S. (2020, December 5). *Stratified sampling in machine learning.* Medium. Retrieved from https://medium.com/analytics-vidhya/stratified-sampling-in-machine-learning-f5112b5b9cfe

Mihan, A., & Pandey, A. (2024). *Artificial intelligence bias in the prediction and detection of cardiovascular disease*. Nature Cardiovascular Research. Retrieved from https://www.nature.com/articles/s44325-024-00031-9

Moore, A., & Bell, M. (2022, November 8). *XGBoost, a novel explainable AI technique, in the prediction of myocardial infarction: A UK Biobank cohort study.* National Library of Medicine. Retrieved from https://pmc.ncbi.nlm.nih.gov/articles/PMC9647306

Pandey, A. M., & Van Spall, H. G. C. (2024, October). *Mitigating the risk of artificial intelligence bias in cardiovascular care*. The Lancet Digital Health, *6*(10), e728-e737. https://doi.org/10.1016/S2589-7500(24)00155-9

Shedden, K. (2020–2021). *Stratification*. University of Michigan. Retrieved from https://dept.stat.lsa.umich.edu/~kshedden/introds/

In [1]:
```
import boto3
import sagemaker

# Initialize the SageMaker session and get the execution role.
sagemaker_session = sagemaker.Session()
role = sagemaker.get_execution_role()

# Define your custom S3 bucket for the CardioValve project.
# Replace the bucket name below with your chosen unique name if needed.
bucket = "import boto3
print(boto3.client("sts").get_caller_identity())
"
region = boto3.Session().region_name

# Create an S3 client for operations.
sm = boto3.Session().client(service_name="sagemaker", region_name=region)
```

```
  Cell In[1], line 10
    bucket = "import boto3
             ^
SyntaxError: unterminated string literal (detected at line 10)
```

In [1]:
```
import boto3
print(boto3.client("sts").get_caller_identity())
```

```
{'UserId': 'AROAWF2EKWPTN57HJGVZX:SageMaker', 'Account': '424808199142', 'Ar
n': 'arn:aws:sts::424808199142:assumed-role/LabRole/SageMaker', 'ResponseMet
adata': {'RequestId': '7b821ad7-d025-4871-aae5-c82b90ea6cd9', 'HTTPStatusCod
e': 200, 'HTTPHeaders': {'x-amzn-requestid': '7b821ad7-d025-4871-aae5-c82b90
ea6cd9', 'content-type': 'text/xml', 'content-length': '432', 'date': 'Sun,
30 Mar 2025 18:46:46 GMT'}, 'RetryAttempts': 0}}
```

In [2]:
```
# Create the S3 bucket if it doesn't already exist.
!aws s3 mb s3://$bucket
```

```
make_bucket failed: s3://cardiovale-solutions-datascience-pipeline An error
occurred (BucketAlreadyExists) when calling the CreateBucket operation: The
requested bucket name is not available. The bucket namespace is shared by al
l users of the system. Please select a different name and try again.
```

In [3]:
```
!aws s3 cp s3://cardiovale-solutions-datascience-pipeline/raw-data/cardio_tr
!aws s3 cp s3://cardiovale-solutions-datascience-pipeline/raw-data/Quitline_
```

```
fatal error: An error occurred (404) when calling the HeadObject operation:
Key "raw-data/cardio_train.csv" does not exist
fatal error: An error occurred (404) when calling the HeadObject operation:
Key "raw-data/Quitline___Services_Available___Medications_-_2010_To_Present_
20250306.csv" does not exist
```

In [4]:
```
!aws s3 mb s3://cardiovale-solutions-datascience-pipeline/raw-data/cardio_tr
!aws s3 mb s3://cardiovale-solutions-datascience-pipeline/raw-data/quitline_
```

```
make_bucket failed: s3://cardiovale-solutions-datascience-pipeline/raw-data/
cardio_train/ An error occurred (BucketAlreadyExists) when calling the Creat
eBucket operation: The requested bucket name is not available. The bucket na
mespace is shared by all users of the system. Please select a different name
and try again.
make_bucket failed: s3://cardiovale-solutions-datascience-pipeline/raw-data/
quitline_services/ An error occurred (BucketAlreadyExists) when calling the
CreateBucket operation: The requested bucket name is not available. The buck
et namespace is shared by all users of the system. Please select a different
name and try again.
```

In [5]:
```
!aws s3 cp /home/sagemaker-user/ads-508-team/data/cardio_train.csv s3://card
!aws s3 cp /home/sagemaker-user/ads-508-team/data/Quitline___Services_Availa
```

```
upload: ../data/cardio_train.csv to s3://cardiovale-solutions-datascience-pi
peline/raw-data/cardio_train/cardio_train.csv
upload: ../data/Quitline___Services_Available___Medications_-_2010_To_Presen
t_20250306.csv to s3://cardiovale-solutions-datascience-pipeline/raw-data/qu
itline_services/Quitline___Services_Available___Medications_-_2010_To_Presen
t_20250306.csv
```

In [6]:
```
# Move the Cardio dataset to the correct folder
!aws s3 mv s3://cardiovale-solutions-datascience-pipeline/raw-data/cardio_tr

# Move the Quitline dataset to the correct folder
!aws s3 mv s3://cardiovale-solutions-datascience-pipeline/raw-data/Quitline_
```

```
fatal error: An error occurred (404) when calling the HeadObject operation:
Key "raw-data/cardio_train_fixed.csv" does not exist
fatal error: An error occurred (404) when calling the HeadObject operation:
Key "raw-data/Quitline___Services_Available___Medications_-_2010_To_Present_
20250306.csv" does not exist
```

In [7]:
```
!aws s3 ls s3://cardiovale-solutions-datascience-pipeline/raw-data/ --recurs
```

```
2025-03-18 04:09:39        108 raw-data/Unsaved/2025/03/17/484c9471-fd67-4d2
5-b636-e00e45c84100.csv
2025-03-18 04:09:39        531 raw-data/Unsaved/2025/03/17/484c9471-fd67-4d2
5-b636-e00e45c84100.csv.metadata
2025-03-18 06:47:06        108 raw-data/Unsaved/2025/03/17/61039647-7075-4d0
f-bac7-78379707d88c.csv
2025-03-18 06:47:06        531 raw-data/Unsaved/2025/03/17/61039647-7075-4d0
f-bac7-78379707d88c.csv.metadata
2025-03-18 06:46:46        752 raw-data/Unsaved/2025/03/17/654150cb-4ec1-4b2
5-9104-54890aea6c93.csv
2025-03-18 06:46:46        542 raw-data/Unsaved/2025/03/17/654150cb-4ec1-4b2
5-9104-54890aea6c93.csv.metadata
2025-03-18 06:46:36        108 raw-data/Unsaved/2025/03/17/6682a43f-94ea-44e
4-b171-b6f1fdff4bc5.csv
2025-03-18 06:46:36        531 raw-data/Unsaved/2025/03/17/6682a43f-94ea-44e
4-b171-b6f1fdff4bc5.csv.metadata
2025-03-18 06:47:32        789 raw-data/Unsaved/2025/03/17/b9275a95-5cb2-47b
c-ac79-77632106ef5b.csv
2025-03-18 06:47:32        542 raw-data/Unsaved/2025/03/17/b9275a95-5cb2-47b
c-ac79-77632106ef5b.csv.metadata
2025-03-18 06:45:41        108 raw-data/Unsaved/2025/03/17/c0410964-9655-496
a-8372-c5178c043924.csv
2025-03-18 06:45:41        531 raw-data/Unsaved/2025/03/17/c0410964-9655-496
a-8372-c5178c043924.csv.metadata
2025-03-18 04:57:53        815 raw-data/Unsaved/2025/03/17/d8b20f72-41c5-4f0
3-b6f1-a116f1eb09b2.csv
2025-03-18 04:57:53       2746 raw-data/Unsaved/2025/03/17/d8b20f72-41c5-4f0
3-b6f1-a116f1eb09b2.csv.metadata
2025-03-18 04:51:22        815 raw-data/Unsaved/2025/03/17/ee3da1b3-b25b-44a
0-b812-781a71eb0162.csv
2025-03-18 04:51:22       2746 raw-data/Unsaved/2025/03/17/ee3da1b3-b25b-44a
0-b812-781a71eb0162.csv.metadata
2025-03-18 06:47:19        108 raw-data/Unsaved/2025/03/17/f13b7135-6f65-40c
f-9be2-b4cf963315b7.csv
2025-03-18 06:47:19        531 raw-data/Unsaved/2025/03/17/f13b7135-6f65-40c
f-9be2-b4cf963315b7.csv.metadata
2025-03-30 18:23:52    2941524 raw-data/cardio_train/cardio_train.csv
2025-03-18 05:25:36    2941524 raw-data/cardio_train/cardio_train_cleaned.cs
v
2025-03-18 02:45:52    2941524 raw-data/cardio_train/cardio_train_fixed.csv
2025-03-18 06:32:27    2941524 raw-data/cardio_train_cleaned/cardio_train_cl
eaned.csv
2025-03-18 06:41:37     294049 raw-data/parquet/cardio_train/gender=1/202503
18_064135_00007_2bf3v_7f7fc79f-c968-48fc-833c-5d88f2a808bd
2025-03-18 06:41:37     173827 raw-data/parquet/cardio_train/gender=2/202503
18_064135_00007_2bf3v_039c8e74-46dc-4717-be92-8750d61df510
2025-03-30 18:23:53    6934258 raw-data/quitline_services/Quitline___Service
s_Available___Medications_-_2010_To_Present_20250306.csv
2025-03-18 05:07:32    7729582 raw-data/quitline_services/quitline_fixed.csv
```

In [8]:
```
# List the contents of the newly uploaded folder to verify the files are the
!aws s3 ls s3://$bucket/raw-data/
```

```
                           PRE Unsaved/
                           PRE cardio_train/
                           PRE cardio_train_cleaned/
                           PRE parquet/
                           PRE quitline_services/
```

Release Resources

In [9]:
```
%%html

<p><b>Shutting down your kernel for this notebook to release resources.</b><
<button class="sm-command-button" data-commandlinker-command="kernelmenu:shu

<script>
try {
    els = document.getElementsByClassName("sm-command-button");
    els[0].click();
}
catch(err) {
    // NoOp
}
</script>
```

Shutting down your kernel for this notebook to release resources.

In [1]:
```
%%javascript

try {
    Jupyter.notebook.save_checkpoint();
    Jupyter.notebook.session.delete();
}
catch(err) {
    // NoOp
}
```

In [ ]:
```

```

jupyter 02_Data_Bias_Analysis_ProcessingJob Last Checkpoint: 42 minutes ago

File Edit View Run Kernel Settings Help

Markdown

Run Data Bias Analysis with SageMaker Clarify (

Using SageMaker Processing Jobs

[1]:
```
import boto3
import sagemaker
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Initialize SageMaker Session
sess = sagemaker.Session()

# Use your S3 bucket instead of default
bucket = "cardiovale-solutions-datascience-pipeline"

# Ensure correct IAM Role
try:
    role = sagemaker.get_execution_role()  # Works in SageMaker Studio
except:
    role = "arn:aws:iam::<YOUR_ACCOUNT_ID>:role/service-role/AmazonSageMaker-E

# Get AWS Region
region = boto3.Session().region_name
```

In [2]:
```
# Import necessary libraries
import boto3
import pandas as pd
import os

# AWS S3 Configuration
s3_bucket = "cardiovale-solutions-datascience-pipeline"
s3_client = boto3.client("s3")

# Define the root directory and correct data folder path
root_directory = os.path.abspath(os.path.join(os.getcwd(), ".."))  # Go to t
data_folder = os.path.join(root_directory, "data")  # Save data in the root-

# Ensure the data folder exists
os.makedirs(data_folder, exist_ok=True)

# Files to fetch from S3
s3_files = {
    "cardio_train": "raw-data/cardio_train.csv",
    "quitline_services": "raw-data/Quitline___Services_Available___Medicatic
}
```

In [3]:
```
# -----------------------------
# 1. Manually Fetch Data from S3
# -----------------------------
print("Fetching data from S3...\n")

for dataset, s3_path in s3_files.items():
    local_file_path = os.path.join(data_folder, f"{dataset}.csv")  # Save in

    try:
        s3_client.download_file(s3_bucket, s3_path, local_file_path)
        print(f"{dataset} downloaded successfully! Saved to {local_file_path
    except Exception as e:
        print(f"Error fetching {dataset}: {e}")
```

```
Fetching data from S3...

Error fetching cardio_train: An error occurred (404) when calling the HeadOb
ject operation: Not Found
Error fetching quitline_services: An error occurred (404) when calling the H
eadObject operation: Not Found
```

In [4]:
```
# -----------------------------
# 2. Load and Explore Data
# -----------------------------
print("\nLoading Data into Pandas...")

# Load data into Pandas
cardio_df = pd.read_csv(os.path.join(data_folder, "cardio_train.csv"))
quitline_df = pd.read_csv(os.path.join(data_folder, "quitline_services.csv")


# Display basic dataset info
```

```
print("\nCardio Dataset Sample:")
print(cardio_df.head())

print("\nQuitline Dataset Sample:")
print(quitline_df.head())

# Check data types
print("\nCardio Data Types:")
print(cardio_df.dtypes)

print("\nQuitline Data Types:")
print(quitline_df.dtypes)

# Check for missing values
print("\nMissing Values in Cardio Dataset:")
print(cardio_df.isnull().sum())

print("\nMissing Values in Quitline Dataset:")
print(quitline_df.isnull().sum())
```

```
Loading Data into Pandas...

Cardio Dataset Sample:
  id;age;gender;height;weight;ap_hi;ap_lo;cholesterol;gluc;smoke;alco;activ
e;cardio
0              0;18393;2;168;62.0;110;80;1;1;0;0;1;0
1              1;20228;1;156;85.0;140;90;3;1;0;0;1;1
2              2;18857;1;165;64.0;130;70;3;1;0;0;0;1
3             3;17623;2;169;82.0;150;100;1;1;0;0;1;1
4              4;17474;1;156;56.0;100;60;1;1;0;0;0;0

Quitline Dataset Sample:
   Year  Date Date_Ref LocationAbbr LocationDesc TopicType  \
0  2020    12  Jul-Dec           GA      Georgia  Quitline
1  2020    12  Jul-Dec           MO     Missouri  Quitline
2  2020    12  Jul-Dec           MT      Montana  Quitline
3  2020    12  Jul-Dec           NJ   New Jersey  Quitline
4  2020    12  Jul-Dec           OK     Oklahoma  Quitline

            TopicDesc  MeasureDesc         Sub-Measure              Variable
\
0  Services Available  Medications    Nicotine Lozenge                   NaN
1  Services Available  Medications    Nicotine Lozenge                   NaN
2  Services Available  Medications  Bupropion (Zyban®)                   NaN
3  Services Available  Medications    Nicotine Lozenge                   NaN
4  Services Available  Medications        Nicotine Gum  All Eligible Callers

   ... Number_of_Weeks_Offered    Limit_Per_Year Comments  \
0  ...                     NaN               NaN      NaN
1  ...                     NaN               NaN      NaN
2  ...                     NaN               NaN      NaN
3  ...                     NaN               NaN      NaN
4  ...                     2.0  2 times per year      NaN

                               GeoLocation TopicTypeId TopicId MeasureID  \
0   (32.83968109300048, -83.62758034599966)         QUI  900QUI    912MED
1  (38.635790776000476, -92.56630005299968)         QUI  900QUI    912MED
2  (47.06652897200047, -109.42442064499971)         QUI  900QUI    912MED
3   (40.13057004800049, -74.27369128799967)         QUI  900QUI    912MED
4   (35.47203135600046, -97.52107021399968)         QUI  900QUI    912MED

                                          SOURCE SubMeasureID DisplayOrd
er
0  National Quitline Data Warehouse, Office on Sm...        QUT07
7
1  National Quitline Data Warehouse, Office on Sm...        QUT07
7
2  National Quitline Data Warehouse, Office on Sm...        QUT10
10
3  National Quitline Data Warehouse, Office on Sm...        QUT07
7
4  National Quitline Data Warehouse, Office on Sm...        QUT06
6

[5 rows x 41 columns]
```

```
Cardio Data Types:
id;age;gender;height;weight;ap_hi;ap_lo;cholesterol;gluc;smoke;alco;active;c
ardio    object
dtype: object

Quitline Data Types:
Year                                    int64
Date                                    int64
Date_Ref                               object
LocationAbbr                           object
LocationDesc                           object
TopicType                              object
TopicDesc                              object
MeasureDesc                            object
Sub-Measure                            object
Variable                               object
Offered_for_Free                       object
Offered_for_Free_Text                  object
Geographic_Requirements                object
Geographic_Requirements_Text           object
Age_Requirements                       object
Age_Requirements_Text                  object
Readiness_to_Quit_Requirements         object
Readiness_to_Quit_Requirements_Text    object
Counseling_Requirements                object
Counseling_Requirements_Text           object
Physician_Approval_Required            object
Physician_Approval_Required_Text       object
Medical_Restrictions                   object
Medical_Restrictions_Text              object
Insurance_Requirements                 object
Insurance_Requirements_Text            object
Other_Requirements                     object
Other_Requirements_Text                object
Limited_Supply                         object
Limited_Supply_Text                    object
Unlimited_Weeks_Offered                object
Number_of_Weeks_Offered               float64
Limit_Per_Year                         object
Comments                               object
GeoLocation                            object
TopicTypeId                            object
TopicId                                object
MeasureID                              object
SOURCE                                 object
SubMeasureID                           object
DisplayOrder                            int64
dtype: object

Missing Values in Cardio Dataset:
id;age;gender;height;weight;ap_hi;ap_lo;cholesterol;gluc;smoke;alco;active;c
ardio    0
dtype: int64

Missing Values in Quitline Dataset:
Year                                      0
```

```
Date                                     0
Date_Ref                                 0
LocationAbbr                             0
LocationDesc                             0
TopicType                                0
TopicDesc                                0
MeasureDesc                              0
Sub-Measure                              0
Variable                              8188
Offered_for_Free                        63
Offered_for_Free_Text                18130
Geographic_Requirements               8148
Geographic_Requirements_Text          8371
Age_Requirements                      8190
Age_Requirements_Text                 8288
Readiness_to_Quit_Requirements        8190
Readiness_to_Quit_Requirements_Text  10836
Counseling_Requirements               8202
Counseling_Requirements_Text          9082
Physician_Approval_Required           8188
Physician_Approval_Required_Text     16351
Medical_Restrictions                  8190
Medical_Restrictions_Text            14562
Insurance_Requirements                8207
Insurance_Requirements_Text          12575
Other_Requirements                    8194
Other_Requirements_Text              17325
Limited_Supply                        8342
Limited_Supply_Text                  18090
Unlimited_Weeks_Offered               8191
Number_of_Weeks_Offered               8204
Limit_Per_Year                        8204
Comments                             17574
GeoLocation                              0
TopicTypeId                              0
TopicId                                  0
MeasureID                                0
SOURCE                                   0
SubMeasureID                             0
DisplayOrder                             0
dtype: int64
```

In [5]:
```
# ----------------------------
# 3. Identify Key Fields & Bias
# ----------------------------
print("\n Key Fields & Bias Assessment:")

# Key fields expected
key_fields_cardio = ["id", "age", "gender", "height", "weight", "cholesterol
key_fields_quitline = ["state", "year", "medication_available", "medication_

# Bias Analysis
data_bias_concerns = """
- Cardio dataset may have selection bias based on demographic distribution.
- Quitline services dataset may be biased towards urban areas where programs
"""
```

```
print(f"Key Fields in Cardio Dataset: {key_fields_cardio}")
print(f"Key Fields in Quitline Dataset: {key_fields_quitline}")
print(f"\nData Bias Concerns: {data_bias_concerns}")
```

```
 Key Fields & Bias Assessment:
Key Fields in Cardio Dataset: ['id', 'age', 'gender', 'height', 'weight', 'c
holesterol', 'smoke', 'cardio']
Key Fields in Quitline Dataset: ['state', 'year', 'medication_available', 'm
edication_type']

Data Bias Concerns:
- Cardio dataset may have selection bias based on demographic distribution.
- Quitline services dataset may be biased towards urban areas where programs
are well-documented.
```

In [6]:
```
# ----------------------------
# 4. Security & Privacy Checklist
# ----------------------------
print("Security & Privacy Checklist:")

# PHI (Protected Health Information) and PII (Personally Identifiable Inform
contains_phi = "No"
contains_pii = "No"
tracks_user_behavior = "No"
processes_credit_card_data = "No"

# S3 Buckets Read/Write
s3_buckets_used = [s3_files["cardio_train"], s3_files["quitline_services"]]

# Print Security Assessment
print(f"Contains PHI: {contains_phi}")
print(f"Contains PII: {contains_pii}")
print(f"Tracks User Behavior: {tracks_user_behavior}")
print(f"Processes Credit Card Data: {processes_credit_card_data}")
print(f"S3 Buckets Accessed: {s3_buckets_used}")
```

```
Security & Privacy Checklist:
Contains PHI: No
Contains PII: No
Tracks User Behavior: No
Processes Credit Card Data: No
S3 Buckets Accessed: ['raw-data/cardio_train.csv', 'raw-data/Quitline___Serv
ices_Available___Medications_-_2010_To_Present_20250306.csv']
```

In [7]:
```
import os

# Define correct data folder (outside notebooks)
data_folder = "/home/sagemaker-user/ads-508-team/data"

# Ensure the folder exists
os.makedirs(data_folder, exist_ok=True)

# ----------------------------
# 5. Measuring Impact
# ----------------------------
```

```
print("Measuring Project Impact:")
metrics = [
    "Reduction in missing data (%)",
    "Increase in data quality scores (%)",
]
print(f"Expected Impact Metrics: {metrics}")

# Store Data Locally for Further Processing
print("\nSaving processed data locally...")

cardio_df.to_csv(os.path.join(data_folder, "processed_cardio_train.csv"), ir
quitline_df.to_csv(os.path.join(data_folder, "processed_quitline_services.cs

print("Data Successfully Stored for Further Analysis!")
```

```
Measuring Project Impact:
Expected Impact Metrics: ['Reduction in missing data (%)', 'Increase in data
quality scores (%)']

Saving processed data locally...
Data Successfully Stored for Further Analysis!
```

Realease Resources

In [8]:
```
%%html

<p><b>Shutting down your kernel for this notebook to release resources.</b><
<button class="sm-command-button" data-commandlinker-command="kernelmenu:shu

<script>
try {
    els = document.getElementsByClassName("sm-command-button");
    els[0].click();
}
catch(err) {
    // NoOp
}
</script>
```

Shutting down your kernel for this notebook to release resources.

In [9]:
```
%%javascript

try {
    Jupyter.notebook.save_checkpoint();
    Jupyter.notebook.session.delete();
}
catch(err) {
    // NoOp
}
```

In [ ]:

Feature Transformation with Scikit-Learn In This Notebook

Saving Features into the SageMaker Feature Store

In [1]:
```
!pip install --disable-pip-version-check -q tensorflow==2.8.1
!pip install --disable-pip-version-check -q transformers==4.46.0
!pip install protobuf==3.20.*
import sagemaker
import boto3

sess = sagemaker.Session()
bucket = "cardiovale-solutions-datascience-pipeline"
role = sagemaker.get_execution_role()
region = boto3.Session().region_name

sm = boto3.Session().client(service_name="sagemaker", region_name=region)
s3 = boto3.Session().client(service_name="s3", region_name=region)
```

```
ERROR: Could not find a version that satisfies the requirement tensorflow==
2.8.1 (from versions: 2.12.0rc0, 2.12.0rc1, 2.12.0, 2.12.1, 2.13.0rc0, 2.13.
0rc1, 2.13.0rc2, 2.13.0, 2.13.1, 2.14.0rc0, 2.14.0rc1, 2.14.0, 2.14.1, 2.15.
0rc0, 2.15.0rc1, 2.15.0, 2.15.0.post1, 2.15.1, 2.16.0rc0, 2.16.1, 2.16.2, 2.
17.0rc0, 2.17.0rc1, 2.17.0, 2.17.1, 2.18.0rc0, 2.18.0rc1, 2.18.0rc2, 2.18.0,
2.18.1, 2.19.0rc0, 2.19.0)
ERROR: No matching distribution found for tensorflow==2.8.1
Requirement already satisfied: protobuf==3.20.* in /opt/conda/lib/python3.1
1/site-packages (3.20.3)
/opt/conda/lib/python3.11/site-packages/pydantic/_internal/_fields.py:192: U
serWarning: Field name "json" in "MonitoringDatasetFormat" shadows an attrib
ute in parent "Base"
  warnings.warn(
sagemaker.config INFO - Not applying SDK defaults from location: /etc/xdg/sa
gemaker/config.yaml
sagemaker.config INFO - Not applying SDK defaults from location: /home/sagem
aker-user/.config/sagemaker/config.yaml
```

Prepare the data

In [2]:
```
import pandas as pd
import boto3
import sagemaker
from sagemaker import get_execution_role
from pyathena import connect

# Setup AWS session
region = boto3.Session().region_name
sagemaker_session = sagemaker.Session()
bucket = "cardiovale-solutions-datascience-pipeline"  # Your actual S3 bucke
```

```
role = get_execution_role()

# Connect to Athena
conn = connect(s3_staging_dir=f"s3://{bucket}/athena/staging/", region_name=

# Fetch `cardio_train_cleaned` dataset
query = "SELECT * FROM cardiovale_db.cardio_train_cleaned"
df_cardio = pd.read_sql(query, conn)

print("Successfully loaded cardio_train_cleaned from Athena")
print(df_cardio.head())
```

```
/tmp/ipykernel_7380/3809400061.py:18: UserWarning: pandas only supports SQLA
lchemy connectable (engine/connection) or database string URI or sqlite3 DBA
PI2 connection. Other DBAPI2 objects are not tested. Please consider using S
QLAlchemy.
  df_cardio = pd.read_sql(query, conn)
Successfully loaded cardio_train_cleaned from Athena
   id    age  gender  height  weight  ap_hi  ap_lo  cholesterol  gluc  smoke
\
0   0  18393       2     168    62.0    110     80            1     1      0
1   1  20228       1     156    85.0    140     90            3     1      0
2   2  18857       1     165    64.0    130     70            3     1      0
3   3  17623       2     169    82.0    150    100            1     1      0
4   4  17474       1     156    56.0    100     60            1     1      0

   alco  active  cardio
0     0       1       0
1     0       1       1
2     0       0       1
3     0       1       1
4     0       0       0
```

In [3]:
```
df_cardio.columns
```

Out[3]:
```
Index(['id', 'age', 'gender', 'height', 'weight', 'ap_hi', 'ap_lo',
       'cholesterol', 'gluc', 'smoke', 'alco', 'active', 'cardio'],
      dtype='object')
```

In [4]:
```
#fetch quitline_fixed_.csv
query = "SELECT * FROM cardiovale_db.quitline_fixed LIMIT 5"
df_quitline = pd.read_sql(query, conn)

print("Successfully loaded quitline_fixed from Athena")
print(df_quitline.head())
```

```
/tmp/ipykernel_7380/126416555.py:3: UserWarning: pandas only supports SQLAlc
hemy connectable (engine/connection) or database string URI or sqlite3 DBAPI
2 connection. Other DBAPI2 objects are not tested. Please consider using SQL
Alchemy.
  df_quitline = pd.read_sql(query, conn)
```

```
Successfully loaded quitline_fixed from Athena
   year  date date_ref locationabbr locationdesc topictype  \
0  2020    12  Jul-Dec           GA      Georgia  Quitline
1  2020    12  Jul-Dec           MO     Missouri  Quitline
2  2020    12  Jul-Dec           MT      Montana  Quitline
3  2020    12  Jul-Dec           NJ   New Jersey  Quitline
4  2020    12  Jul-Dec           OK     Oklahoma  Quitline

            topicdesc  measuredesc         sub_measure              variable
\
0  Services Available  Medications    Nicotine Lozenge                   nan
1  Services Available  Medications    Nicotine Lozenge                   nan
2  Services Available  Medications  Bupropion (Zyban®)                   nan
3  Services Available  Medications    Nicotine Lozenge                   nan
4  Services Available  Medications        Nicotine Gum  All Eligible Callers

   ... number_of_weeks_offered    limit_per_year comments  \
0  ...                     NaN               nan      nan
1  ...                     NaN               nan      nan
2  ...                     NaN               nan      nan
3  ...                     NaN               nan      nan
4  ...                     2.0  2 times per year      nan

            geolocation             topictypeid topicid measureid  source  \
0   "(32.83968109300048    -83.62758034599966)"     QUI    900QUI  912MED
1  "(38.635790776000476    -92.56630005299968)"     QUI    900QUI  912MED
2   "(47.06652897200047   -109.42442064499971)"     QUI    900QUI  912MED
3   "(40.13057004800049    -74.27369128799967)"     QUI    900QUI  912MED
4   "(35.47203135600046    -97.52107021399968)"     QUI    900QUI  912MED

                       submeasureid displayorder
0  "National Quitline Data Warehouse         None
1  "National Quitline Data Warehouse         None
2  "National Quitline Data Warehouse         None
3  "National Quitline Data Warehouse         None
4  "National Quitline Data Warehouse         None

[5 rows x 41 columns]
```

In [5]:
```
df_quitline.columns
```

Out[5]:
```
Index(['year', 'date', 'date_ref', 'locationabbr', 'locationdesc', 'topicty
pe',
       'topicdesc', 'measuredesc', 'sub_measure', 'variable',
       'offered_for_free', 'offered_for_free_text', 'geographic_requirement
s',
       'geographic_requirements_text', 'age_requirements',
       'age_requirements_text', 'readiness_to_quit_requirements',
       'readiness_to_quit_requirements_text', 'counseling_requirements',
       'counseling_requirements_text', 'physician_approval_required',
       'physician_approval_required_text', 'medical_restrictions',
       'medical_restrictions_text', 'insurance_requirements',
       'insurance_requirements_text', 'other_requirements',
       'other_requirements_text', 'limited_supply', 'limited_supply_text',
       'unlimited_weeks_offered', 'number_of_weeks_offered', 'limit_per_yea
r',
       'comments', 'geolocation', 'topictypeid', 'topicid', 'measureid',
       'source', 'submeasureid', 'displayorder'],
      dtype='object')
```

In [6]:
```
# ------------------------------------
# The cardio dataset does not have a 'State' column, but the quitline datase
# To merge both datasets, we randomly assign a state to each row in the card
# This ensures we can associate each patient with state-level quitline data.
# Note: This is an artificial assignment and does not reflect real patient l

import random

# Use correct column name from quitline dataset
state_column = "locationabbr"

# Extract unique state abbreviations (like 'CA', 'NY') from quitline data
states = df_quitline[state_column].dropna().unique().tolist()

# Assign a random state to each row in cardio data for merging
df_cardio["State"] = [random.choice(states) for _ in range(len(df_cardio))]

# ------------------------------------
# Aggregating Quitline Data
# ---------------------------
# Summarize quitline data per state:
# - % of entries offering treatment for free
# - % of entries with medical restrictions
# - Average number of weeks offered

df_quitline_summary = df_quitline.groupby(state_column).agg({
    "offered_for_free": lambda x: (x == "Yes").mean(),        # e.g., 0.85 =
    "medical_restrictions": lambda x: (x == "Yes").mean(),    # e.g., 0.40 =
    "number_of_weeks_offered": "mean"                         # average week
}).reset_index()

# Rename for consistency with df_cardio's 'State'
df_quitline_summary.rename(columns={
    state_column: "State",
    "offered_for_free": "Offered_for_Free",
    "medical_restrictions": "Medical_Restrictions",
    "number_of_weeks_offered": "Number_of_Weeks_Offered"
```

```
}, inplace=True)

# ------------------------------------
# Merging Data
# ----------------------------
# Merge cardio data with summarized quitline data using 'State' as the key

df_merged = df_cardio.merge(df_quitline_summary, on="State", how="left")

# Handle missing state-level data (just in case)
df_merged["Number_of_Weeks_Offered"].fillna(0, inplace=True)

# Convert 'State' to categorical for feature store
df_merged["State"] = df_merged["State"].astype("category")
```

```
/tmp/ipykernel_7380/2642915367.py:48: FutureWarning: A value is trying to be
set on a copy of a DataFrame or Series through chained assignment using an i
nplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behave
s as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'd
f.method({col: value}, inplace=True)' or df[col] = df[col].method(value) ins
tead, to perform the operation inplace on the original object.


  df_merged["Number_of_Weeks_Offered"].fillna(0, inplace=True)
```

In [7]:
```
# Rename columns to match Feature Store definition
df_merged.rename(columns={
    "Offered_for_Free": "offered_for_free",
    "Medical_Restrictions": "medical_restrictions",
    "Number_of_Weeks_Offered": "number_of_weeks_offered"
}, inplace=True)

df_merged.head()
```

Out[7]:

	id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	a
0	0	18393	2	168	62.0	110	80	1	1	0	0	
1	1	20228	1	156	85.0	140	90	3	1	0	0	
2	2	18857	1	165	64.0	130	70	3	1	0	0	
3	3	17623	2	169	82.0	150	100	1	1	0	0	
4	4	17474	1	156	56.0	100	60	1	1	0	0	

In [8]:
```
df_merged.columns
```

Out[8]:
```
Index(['id', 'age', 'gender', 'height', 'weight', 'ap_hi', 'ap_lo',
       'cholesterol', 'gluc', 'smoke', 'alco', 'active', 'cardio', 'State',
       'offered_for_free', 'medical_restrictions', 'number_of_weeks_offere
d'],
      dtype='object')
```

Feature Store requires an Event Time feature

In [9]:
```
from time import gmtime, strftime
from sagemaker.feature_store.feature_group import FeatureGroup
from sagemaker.feature_store.feature_definition import FeatureDefinition, Fe
from datetime import datetime
import time

# 1. Generate a unique name using timestamp
timestamp = strftime("%d-%H-%M-%S", gmtime())
feature_group_name = f"cardio-feature-group-{timestamp}"

# 2. Define feature schema (make sure all columns in df_merged match these n
feature_definitions = [
    FeatureDefinition("id", FeatureTypeEnum.INTEGRAL),
    FeatureDefinition("age", FeatureTypeEnum.INTEGRAL),
    FeatureDefinition("gender", FeatureTypeEnum.INTEGRAL),
    FeatureDefinition("height", FeatureTypeEnum.INTEGRAL),
    FeatureDefinition("weight", FeatureTypeEnum.FRACTIONAL),
    FeatureDefinition("ap_hi", FeatureTypeEnum.INTEGRAL),
    FeatureDefinition("ap_lo", FeatureTypeEnum.INTEGRAL),
    FeatureDefinition("cholesterol", FeatureTypeEnum.INTEGRAL),
    FeatureDefinition("gluc", FeatureTypeEnum.INTEGRAL),
    FeatureDefinition("smoke", FeatureTypeEnum.INTEGRAL),
    FeatureDefinition("alco", FeatureTypeEnum.INTEGRAL),
    FeatureDefinition("active", FeatureTypeEnum.INTEGRAL),
    FeatureDefinition("cardio", FeatureTypeEnum.INTEGRAL),
    FeatureDefinition("State", FeatureTypeEnum.STRING),
    FeatureDefinition("offered_for_free", FeatureTypeEnum.FRACTIONAL),  # up
    FeatureDefinition("medical_restrictions", FeatureTypeEnum.FRACTIONAL),
    FeatureDefinition("number_of_weeks_offered", FeatureTypeEnum.FRACTIONAL)
    FeatureDefinition("event_time", FeatureTypeEnum.STRING),
    FeatureDefinition("split_type", FeatureTypeEnum.STRING),
]

# 3. Create the FeatureGroup object
feature_group = FeatureGroup(
    name=feature_group_name,
    sagemaker_session=sagemaker_session,
    feature_definitions=feature_definitions
)

# 4. Create the feature group in SageMaker Feature Store
feature_group.create(
    record_identifier_name="id",
    event_time_feature_name="event_time",
    role_arn=role,
    s3_uri=f"s3://{bucket}/feature-store/cardio/{feature_group_name}",
```

```
    enable_online_store=False,
)

# 5. Wait until Feature Group is ready
while feature_group.describe()["FeatureGroupStatus"] == "Creating":
    print("Waiting for feature group creation...")
    time.sleep(5)

# 6. Add `event_time` column before ingestion (this is crucial!)
df_merged["event_time"] = datetime.utcnow().strftime("%Y-%m-%dT%H:%M:%SZ")

# 7. Add split_type for SageMaker Autopilot compatibility
df_merged["split_type"] = "train"

# 8. Ingest data into feature store
feature_group.ingest(data_frame=df_merged, max_workers=3, wait=True)
```

```
Waiting for feature group creation...
Waiting for feature group creation...
Waiting for feature group creation...
Waiting for feature group creation...
Waiting for feature group creation...
```

Out[9]:

```
IngestionManagerPandas(feature_group_name='cardio-feature-group-05-21-19-2
8', feature_definitions={'id': {'FeatureName': 'id', 'FeatureType': 'Integr
al'}, 'age': {'FeatureName': 'age', 'FeatureType': 'Integral'}, 'gender':
{'FeatureName': 'gender', 'FeatureType': 'Integral'}, 'height': {'FeatureNa
me': 'height', 'FeatureType': 'Integral'}, 'weight': {'FeatureName': 'weigh
t', 'FeatureType': 'Fractional'}, 'ap_hi': {'FeatureName': 'ap_hi', 'Featur
eType': 'Integral'}, 'ap_lo': {'FeatureName': 'ap_lo', 'FeatureType': 'Inte
gral'}, 'cholesterol': {'FeatureName': 'cholesterol', 'FeatureType': 'Integ
ral'}, 'gluc': {'FeatureName': 'gluc', 'FeatureType': 'Integral'}, 'smoke':
{'FeatureName': 'smoke', 'FeatureType': 'Integral'}, 'alco': {'FeatureNam
e': 'alco', 'FeatureType': 'Integral'}, 'active': {'FeatureName': 'active',
'FeatureType': 'Integral'}, 'cardio': {'FeatureName': 'cardio', 'FeatureTyp
e': 'Integral'}, 'State': {'FeatureName': 'State', 'FeatureType': 'Strin
g'}, 'offered_for_free': {'FeatureName': 'offered_for_free', 'FeatureType':
'Fractional'}, 'medical_restrictions': {'FeatureName': 'medical_restriction
s', 'FeatureType': 'Fractional'}, 'number_of_weeks_offered': {'FeatureNam
e': 'number_of_weeks_offered', 'FeatureType': 'Fractional'}, 'event_time':
{'FeatureName': 'event_time', 'FeatureType': 'String'}, 'split_type': {'Fea
tureName': 'split_type', 'FeatureType': 'String'}}, sagemaker_fs_runtime_cl
ient_config=<botocore.config.Config object at 0x7f8fef7b8e90>, sagemaker_se
ssion=<sagemaker.session.Session object at 0x7f9048285450>, max_workers=3,
max_processes=1, profile_name=None, _async_result=<multiprocess.pool.MapRes
ult object at 0x7f8ff55f3710>, _processing_pool=<pool ProcessPool(ncpus=1)
>, _failed_indices=[])
```

Describe the Feature Group

In [10]:

```
feature_group.describe()
```

Out[10]:

```
{'FeatureGroupArn': 'arn:aws:sagemaker:us-east-1:424808199142:feature-grou
p/cardio-feature-group-05-21-19-28',
 'FeatureGroupName': 'cardio-feature-group-05-21-19-28',
 'RecordIdentifierFeatureName': 'id',
 'EventTimeFeatureName': 'event_time',
 'FeatureDefinitions': [{'FeatureName': 'id', 'FeatureType': 'Integral'},
  {'FeatureName': 'age', 'FeatureType': 'Integral'},
  {'FeatureName': 'gender', 'FeatureType': 'Integral'},
  {'FeatureName': 'height', 'FeatureType': 'Integral'},
  {'FeatureName': 'weight', 'FeatureType': 'Fractional'},
  {'FeatureName': 'ap_hi', 'FeatureType': 'Integral'},
  {'FeatureName': 'ap_lo', 'FeatureType': 'Integral'},
  {'FeatureName': 'cholesterol', 'FeatureType': 'Integral'},
  {'FeatureName': 'gluc', 'FeatureType': 'Integral'},
  {'FeatureName': 'smoke', 'FeatureType': 'Integral'},
  {'FeatureName': 'alco', 'FeatureType': 'Integral'},
  {'FeatureName': 'active', 'FeatureType': 'Integral'},
  {'FeatureName': 'cardio', 'FeatureType': 'Integral'},
  {'FeatureName': 'State', 'FeatureType': 'String'},
  {'FeatureName': 'offered_for_free', 'FeatureType': 'Fractional'},
  {'FeatureName': 'medical_restrictions', 'FeatureType': 'Fractional'},
  {'FeatureName': 'number_of_weeks_offered', 'FeatureType': 'Fractional'},
  {'FeatureName': 'event_time', 'FeatureType': 'String'},
  {'FeatureName': 'split_type', 'FeatureType': 'String'}],
 'CreationTime': datetime.datetime(2025, 4, 5, 21, 19, 28, 690000, tzinfo=t
zlocal()),
 'OfflineStoreConfig': {'S3StorageConfig': {'S3Uri': 's3://cardiovale-solut
ions-datascience-pipeline/feature-store/cardio/cardio-feature-group-05-21-1
9-28',
   'ResolvedOutputS3Uri': 's3://cardiovale-solutions-datascience-pipeline/f
eature-store/cardio/cardio-feature-group-05-21-19-28/424808199142/sagemake
r/us-east-1/offline-store/cardio-feature-group-05-21-19-28-1743887968/dat
a'},
  'DisableGlueTableCreation': False,
  'DataCatalogConfig': {'TableName': 'cardio_feature_group_05_21_19_28_1743
887968',
   'Catalog': 'AwsDataCatalog',
   'Database': 'sagemaker_featurestore'}},
 'ThroughputConfig': {'ThroughputMode': 'OnDemand'},
 'RoleArn': 'arn:aws:iam::424808199142:role/LabRole',
 'FeatureGroupStatus': 'Created',
 'ResponseMetadata': {'RequestId': '1c29c23a-b970-47ad-b8ad-ede7ed2a6af8',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amzn-requestid': '1c29c23a-b970-47ad-b8ad-ede7ed2a6af
8',
   'content-type': 'application/x-amz-json-1.1',
   'content-length': '2354',
   'date': 'Sat, 05 Apr 2025 21:23:34 GMT'},
  'RetryAttempts': 0}}
```


In [11]:
```
# Review sample of data to be ingested into Feature Store
df_merged.head()
```

Out[11]:

	id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	a
0	0	18393	2	168	62.0	110	80	1	1	0	0	
1	1	20228	1	156	85.0	140	90	3	1	0	0	
2	2	18857	1	165	64.0	130	70	3	1	0	0	
3	3	17623	2	169	82.0	150	100	1	1	0	0	
4	4	17474	1	156	56.0	100	60	1	1	0	0	

In [12]:
```
print(df_merged.shape)
print(df_merged.dtypes)
```

```
(70000, 19)
id                          int64
age                         int64
gender                      int64
height                      int64
weight                    float64
ap_hi                       int64
ap_lo                       int64
cholesterol                 int64
gluc                        int64
smoke                       int64
alco                        int64
active                      int64
cardio                      int64
State                    category
offered_for_free          float64
medical_restrictions      float64
number_of_weeks_offered   float64
event_time                 object
split_type                 object
dtype: object
```

Cast DataFrame `Object` to Supported Feature Store Data Type `String`

In [13]:
```
# Cast all 'object' columns to string for Feature Store compatibility
for col in df_merged.select_dtypes(include='object').columns:
    df_merged[col] = df_merged[col].astype(str)
```

In [14]:
```
df_merged
```

Out[14]:

	id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smok
0	0	18393	2	168	62.0	110	80	1	1	
1	1	20228	1	156	85.0	140	90	3	1	
2	2	18857	1	165	64.0	130	70	3	1	
3	3	17623	2	169	82.0	150	100	1	1	
4	4	17474	1	156	56.0	100	60	1	1	
...	...	...	...	...	...	...	...	...	...	.
69995	99993	19240	2	168	76.0	120	80	1	1	
69996	99995	22601	1	158	126.0	140	90	2	2	
69997	99996	19066	2	183	105.0	180	90	3	1	
69998	99998	22431	1	163	72.0	135	80	1	2	
69999	99999	20540	1	170	72.0	120	80	2	1	

70000 rows × 19 columns

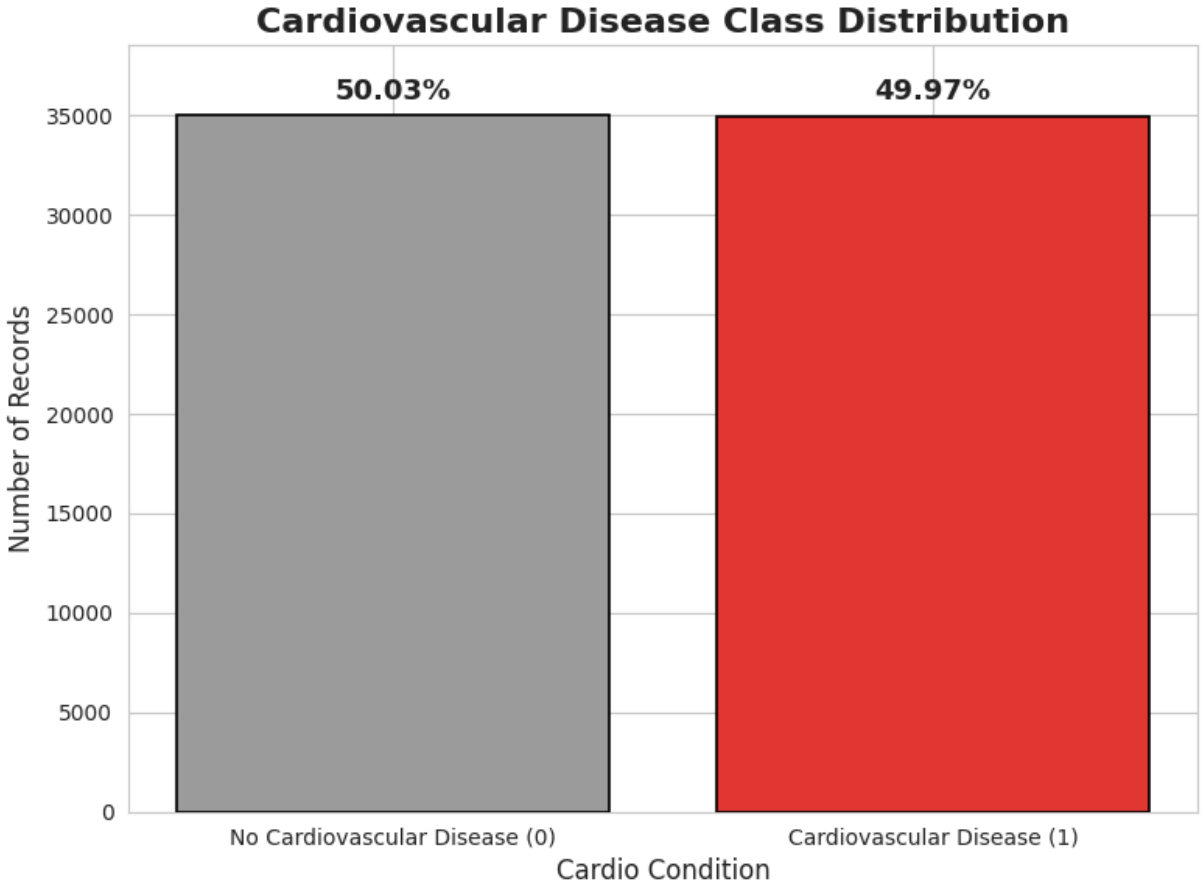
Check for imbalance in the target variable

In [15]:
```
df_merged['cardio'].value_counts()
```

Out[15]:
```
cardio
0    35021
1    34979
Name: count, dtype: int64
```

In [16]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import os

# ----------------------------
# Cardiovascular Class Count with Percentage Labels
# ----------------------------

# Define brand colors
RED = "#E53935"      # Strong red
GRAY = "#9E9E9E"     # Soft gray
```

```
BLACK = "#000000"    # For edges and text

# Set font and style
plt.rcParams.update({
    "font.family": "Times New Roman",
    "font.size": 12,
    "axes.titlesize": 14,
    "axes.labelsize": 12,
    "xtick.labelsize": 10,
    "ytick.labelsize": 10
})

# Output directory
images_path = "../images" if os.path.exists("../images") else "images"
os.makedirs(images_path, exist_ok=True)

# Get actual class counts
class_counts = df_merged['cardio'].value_counts().sort_index()
total = class_counts.sum()
labels = ['No Cardiovascular Disease (0)', 'Cardiovascular Disease (1)']
colors = [GRAY, RED]

# Plot
plt.figure(figsize=(8, 6))
sns.set_style("whitegrid")

bars = plt.bar(labels, class_counts, color=colors, edgecolor=BLACK, linewidt

# Add % labels on top of each bar
for bar, count in zip(bars, class_counts):
    percentage = count / total
    plt.text(
        bar.get_x() + bar.get_width() / 2,
        count + 500,  # a bit above bar height
        f"{percentage:.2%}",
        ha='center',
        va='bottom',
        fontsize=13,
        fontweight='bold'
    )

# Final touches
plt.ylabel("Number of Records")
plt.xlabel("Cardio Condition")
plt.title("Cardiovascular Disease Class Distribution", fontsize=16, fontweig
plt.ylim(0, max(class_counts) * 1.10)
plt.tight_layout()

# Save
plt.savefig(f"{images_path}/class_distribution_counts_and_pct.png", dpi=300)
plt.show()
```



Build Training/ test Dataset

In [17]:
```
import pandas as pd
import io
import boto3
from sklearn.model_selection import train_test_split

# 1. Setup S3 client
s3 = boto3.client("s3")

# 2. Define bucket and prefix from describe() output
bucket = "cardiovale-solutions-datascience-pipeline"
prefix = "feature-store/cardio/cardio-feature-group-22-21-14-34"

# 3. Split df_merged into train, validation, and test
train_df, temp_df = train_test_split(df_merged, test_size=0.3, random_state=
val_df, test_df = train_test_split(temp_df, test_size=0.5, random_state=42,

# 4. Assign split_type
train_df["split_type"] = "train"
val_df["split_type"] = "validation"
test_df["split_type"] = "test"

# 5. Concatenate all splits for upload (Autopilot expects one file)
df_with_split = pd.concat([train_df, val_df, test_df], axis=0)
```

```
# 6. Save to S3 as CSV for Autopilot
csv_buffer = io.StringIO()
df_with_split.to_csv(csv_buffer, index=False)

s3.put_object(
    Bucket=bucket,
    Key=f"{prefix}/autopilot_input.csv",
    Body=csv_buffer.getvalue()
)

print("Final dataset with splits uploaded to S3.")
```

```
Final dataset with splits uploaded to S3.
```

Create An Athena Query

In [18]:
```
feature_store_query = feature_group.athena_query()
```

Get The Feature Group Table Name

In [19]:
```
feature_store_table = feature_store_query.table_name
```

Build an Athena SQL Query

In [20]:
```
print(feature_group.as_hive_ddl())
```

```
CREATE EXTERNAL TABLE IF NOT EXISTS sagemaker_featurestore.cardio-feature-gr
oup-05-21-19-28 (
  id INT
  age INT
  gender INT
  height INT
  weight FLOAT
  ap_hi INT
  ap_lo INT
  cholesterol INT
  gluc INT
  smoke INT
  alco INT
  active INT
  cardio INT
  State STRING
  offered_for_free FLOAT
  medical_restrictions FLOAT
  number_of_weeks_offered FLOAT
  event_time STRING
  split_type STRING
  write_time TIMESTAMP
  event_time TIMESTAMP
  is_deleted BOOLEAN
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerD
e'
  STORED AS
  INPUTFORMAT 'parquet.hive.DeprecatedParquetInputFormat'
  OUTPUTFORMAT 'parquet.hive.DeprecatedParquetOutputFormat'
LOCATION 's3://cardiovale-solutions-datascience-pipeline/feature-store/cardi
o/cardio-feature-group-05-21-19-28/424808199142/sagemaker/us-east-1/offline-
store/cardio-feature-group-05-21-19-28-1743887968/data'
```

In [21]:
```
# Step 3: Build the SQL query for training data
query_string = """
SELECT id, age, gender, height, weight, ap_hi, ap_lo,
       cholesterol, gluc, smoke, alco, active, cardio,
       State, offered_for_free, medical_restrictions, number_of_weeks_offere
FROM "{}"
WHERE split_type = 'train'
LIMIT 5
""".format(feature_store_table)

print("Running: " + query_string)
```

```
Running:
SELECT id, age, gender, height, weight, ap_hi, ap_lo,
       cholesterol, gluc, smoke, alco, active, cardio,
       State, offered_for_free, medical_restrictions, number_of_weeks_offere
d
FROM "cardio_feature_group_05_21_19_28_1743887968"
WHERE split_type = 'train'
LIMIT 5
```

In [22]:
```
query_string = """
SELECT id, age, gender, height, weight, ap_hi, ap_lo,
       cholesterol, gluc, smoke, alco, active, cardio,
       State, offered_for_free, medical_restrictions, number_of_weeks_offere
       split_type
FROM "{}"
WHERE split_type = 'train'
LIMIT 5
""".format(feature_store_table)
```

Run Athena Query

The query results are stored in a S3 bucket.

In [23]:
```
feature_store_query.run(query_string=query_string, output_location="s3://" +
feature_store_query.wait()
```

```
[04/05/25 21:23:37] INFO     Query 1cc5eddb-7b7a-4f6c-83bc-2e88a42505a
[04/05/25 21:23:42] INFO     Query 1cc5eddb-7b7a-4f6c-83bc-2e88a42505a
```

In [24]:
```
df = pd.DataFrame()

df = feature_store_query.as_dataframe()

df
```

Out[24]:

	id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	activ

Release Resources

In [25]:
```
%%html

<p><b>Shutting down your kernel for this notebook to release resources.</b><
<button class="sm-command-button" data-commandlinker-command="kernelmenu:shu

<script>
try {
    els = document.getElementsByClassName("sm-command-button");
    els[0].click();
}
catch(err) {
    // NoOp
}
</script>
```

Shutting down your kernel for this notebook to release resources.

In [26]:
```
%%javascript
```

```
try {
    Jupyter.notebook.save_checkpoint();
    Jupyter.notebook.session.delete();
}
catch(err) {
    // NoOp
}
```

In [ ]:

In [2]:
```
import boto3
import pandas as pd

s3 = boto3.client("s3")
bucket = "cardiovale-solutions-datascience-pipeline"
key = "feature-store/cardio/cardio-feature-group-22-21-14-34/autopilot_input

# Download and load into DataFrame
s3.download_file(bucket, key, "autopilot_input.csv")
df_merged = pd.read_csv("autopilot_input.csv")
df_merged.head()
```

Out[2]:

	id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alc
0	11537	20548	1	163	75.0	160	80	3	1	0	
1	97894	15972	2	172	64.0	110	70	1	1	0	
2	20906	20410	1	168	72.0	130	90	1	1	0	
3	66602	19557	1	151	60.0	110	60	1	1	0	
4	13431	22614	1	147	85.0	180	100	1	1	0	

In [3]:
```
import os
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Define correct data folder
data_folder = "/home/sagemaker-user/ads-508-team/data"

# Load Data with Robust Error Handling
try:
    cardio_df = pd.read_csv(os.path.join(data_folder, "processed_cardio_trai
    quitline_df = pd.read_csv(os.path.join(data_folder, "processed_quitline_
    print("Data loaded successfully!")
except Exception as e:
    print(f"Error loading data: {e}")

# ----------------------------
# 1. Check Missing Values
# ----------------------------
plt.figure(figsize=(10, 5))

# Calculate missing values
missing_values = cardio_df.isnull().sum()
missing_values = missing_values[missing_values > 0]  # Only show columns wit

if not missing_values.empty:
```

```
    missing_values.plot(kind='bar', color='red')
    plt.title("Missing Values in Cardio Data")
    plt.ylabel("Count")
    plt.xlabel("Columns")
    plt.show()
else:
    print("No missing values found in Cardio Data.")
```

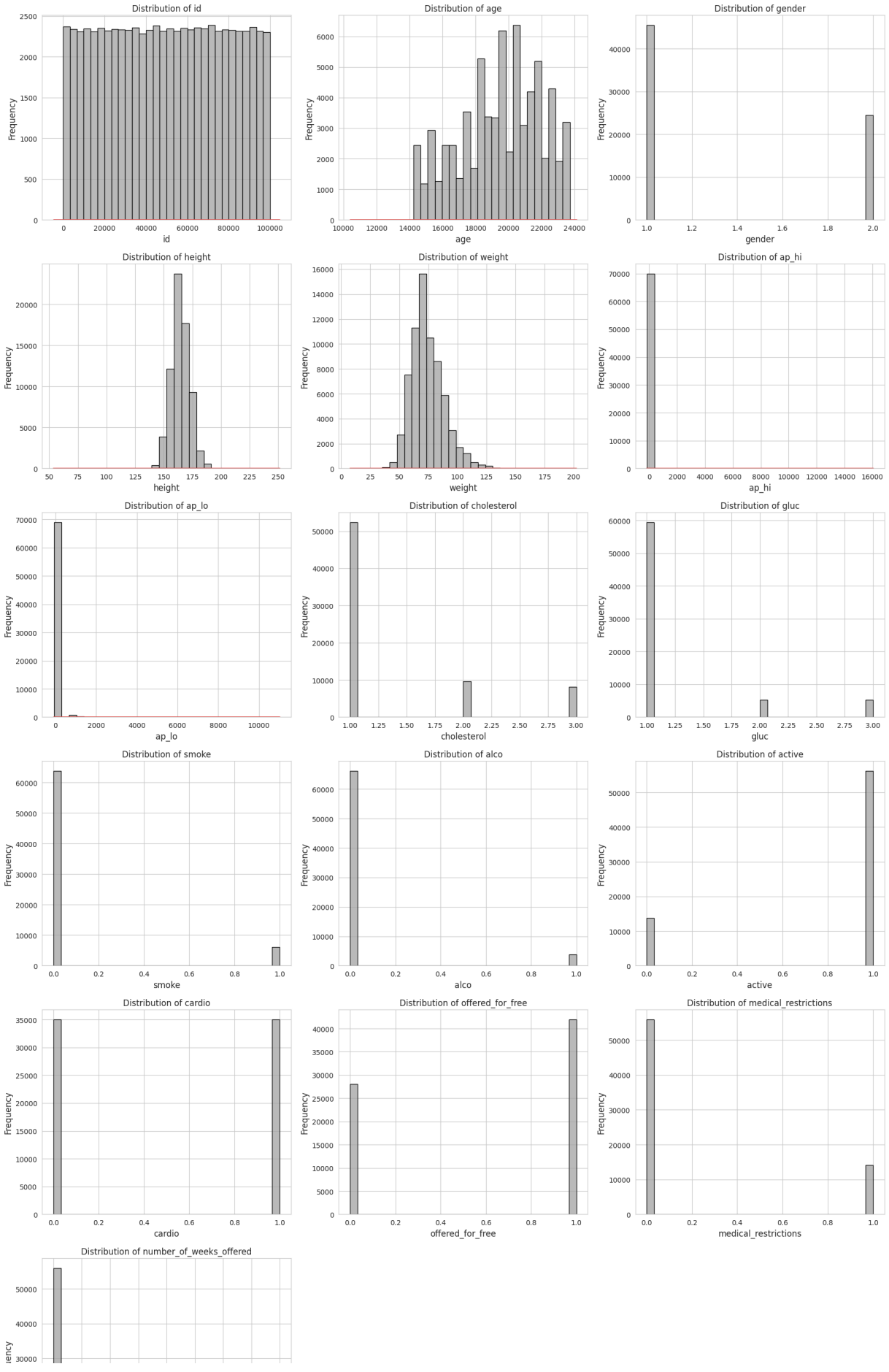
```
Data loaded successfully!
No missing values found in Cardio Data.
<Figure size 1000x500 with 0 Axes>
```

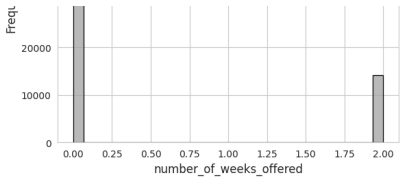
In [4]:

```
import os
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

# ----------------------------
# Style Settings
# ----------------------------
RED = "#E53935"      # Strong red (KDE line)
GRAY = "#9E9E9E"     # Bar color
BLACK = "#000000"    # Edges and labels
plt.rcParams.update({
    "font.family": "Times New Roman",
    "font.size": 12,
    "axes.titlesize": 14,
    "axes.labelsize": 12,
    "xtick.labelsize": 10,
    "ytick.labelsize": 10
})

# Set image save directory
save_dir = "../images" if os.path.exists("../images") else "images"
os.makedirs(save_dir, exist_ok=True)

# Filter numeric columns
numeric_cols = df_merged.select_dtypes(include=['number'])

# ----------------------------
# Feature Distributions
# ----------------------------
if not numeric_cols.empty:
    num_features = numeric_cols.columns
    n_cols = 3
    n_rows = -(-len(num_features) // n_cols)  # Ceiling division
    plt.figure(figsize=(18, 5 * n_rows))
    sns.set_style("whitegrid")

    for i, col in enumerate(num_features, 1):
        plt.subplot(n_rows, n_cols, i)
        data = df_merged[col]

        # First plot the histogram with low zorder to keep it behind KDE
        sns.histplot(data, bins=30, color=GRAY, edgecolor=BLACK, alpha=0.7,
```

```
        # Add KDE line only if feature is continuous (not low-cardinality di
        if data.nunique() > 10 and data.dtype != "object":
            # Plot KDE with higher zorder and adjusted bandwidth
            sns.kdeplot(data, color=RED, linewidth=2, zorder=2, bw_adjust=0.

        plt.title(f"Distribution of {col}", fontsize=12)
        plt.xlabel(col)
        plt.ylabel("Frequency")

    plt.tight_layout()
    plt.suptitle("Feature Distributions", fontsize=18, fontweight='bold', y=

    # Save figure
    plt.savefig(os.path.join(save_dir, "feature_distributions.png"), dpi=300
    plt.show()
else:
    print("No numeric columns found in the dataset.")
```

Feature Distributions

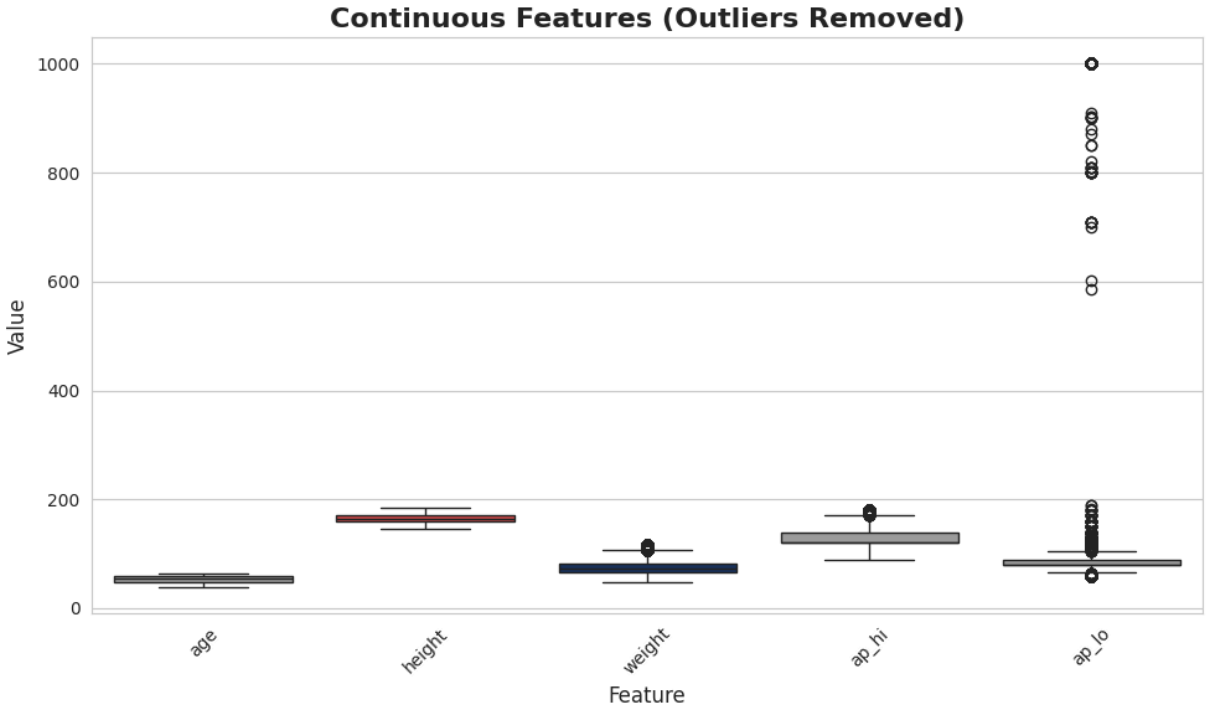




In [5]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import os

# ----------------------------
# Custom Branding Settings
# ----------------------------
RED = "#E53935"      # Primary red from your slide
GRAY = "#9E9E9E"     # Neutral gray
BLUE = "#0F3166"     # Deep blue (optional)


PALETTE = [GRAY, RED, BLUE, GRAY]  # repeating if fewer features

# Font
plt.rcParams.update({
    "font.family": "Times New Roman",
    "font.size": 12,
    "axes.titlesize": 14,
    "axes.labelsize": 12,
    "xtick.labelsize": 10,
    "ytick.labelsize": 10
})

# ----------------------------
# Boxplot of Continuous Features
# ----------------------------
# 1. Save directory setup
save_dir = "../images" if os.path.exists("../images") else "images"
os.makedirs(save_dir, exist_ok=True)

# 2. Define continuous features
features = ["age", "height", "weight", "ap_hi", "ap_lo"]

# 3. Copy relevant data from df_merged
df_plot = df_merged[features].copy()

# Convert age to years if in days
if df_plot["age"].max() > 200:
    df_plot["age"] = df_plot["age"] / 365

# 4. Clip outliers between 1st and 99th percentiles
for col in features:
    lower = df_plot[col].quantile(0.01)
    upper = df_plot[col].quantile(0.99)
    df_plot[col] = df_plot[col].clip(lower, upper)

# 5. Reshape for seaborn
df_melted = df_plot.melt(var_name="Feature", value_name="Value")
```

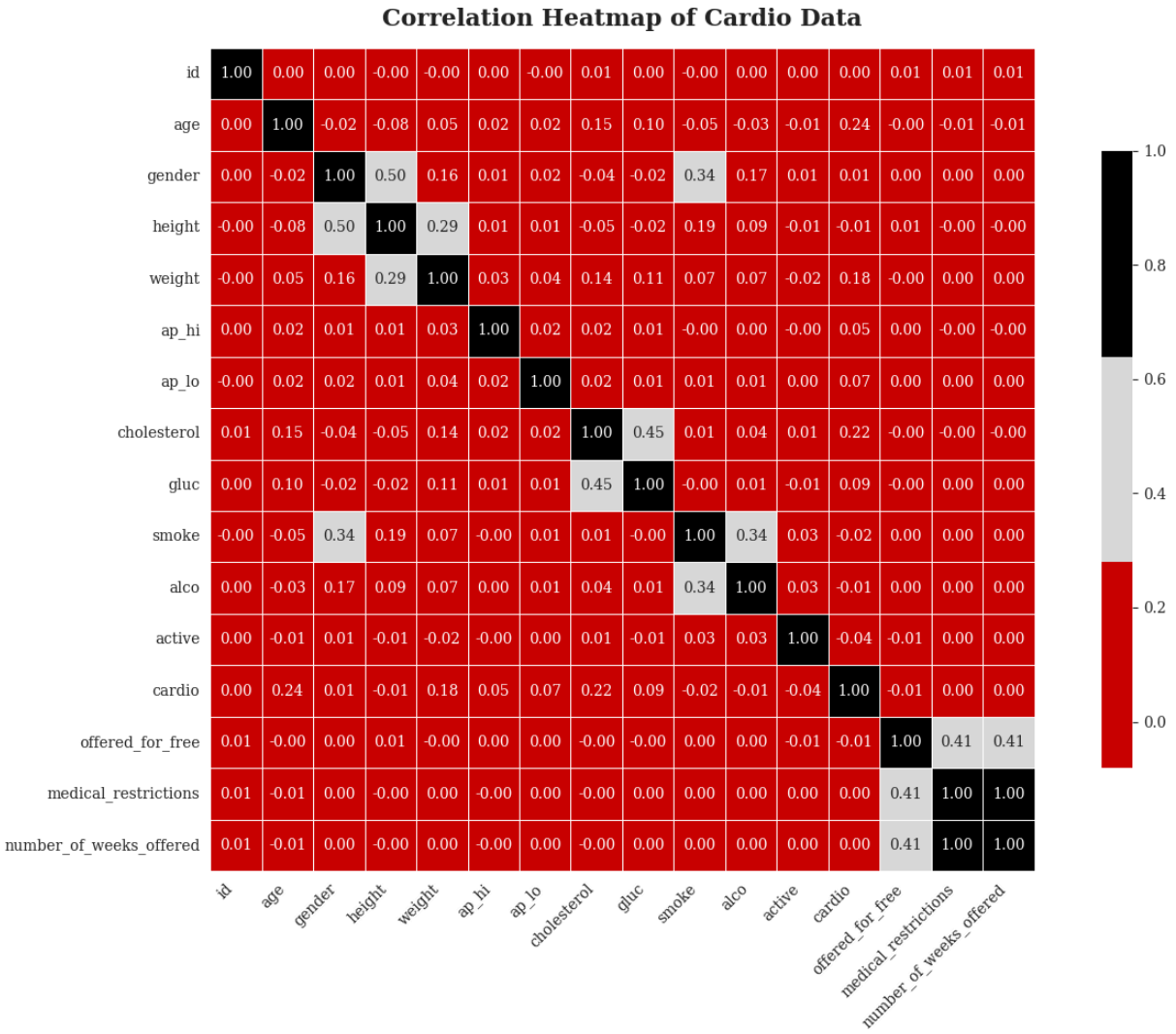
```
# 6. Plot with custom palette
plt.figure(figsize=(10, 6))
sns.set_style("whitegrid")
ax = sns.boxplot(x="Feature", y="Value", data=df_melted, palette=PALETTE)

# 7. Customize
ax.set_title("Continuous Features (Outliers Removed)", fontsize=16, fontweig
plt.xticks(rotation=45)
plt.tight_layout()

# 8. Save the image
plt.savefig(os.path.join(save_dir, "boxplot_continuous_features_clean.png"),
plt.show()
```

```
/tmp/ipykernel_7257/1252532453.py:55: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed
in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the
same effect.

  ax = sns.boxplot(x="Feature", y="Value", data=df_melted, palette=PALETTE)
/tmp/ipykernel_7257/1252532453.py:55: UserWarning:
The palette list has fewer values (4) than needed (5) and will cycle, which
may produce an uninterpretable plot.
  ax = sns.boxplot(x="Feature", y="Value", data=df_melted, palette=PALETTE)
```



In [6]:
```
import os
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.colors import LinearSegmentedColormap

# ----------------------------
# Brand Colors & Font Setup
# ----------------------------
```

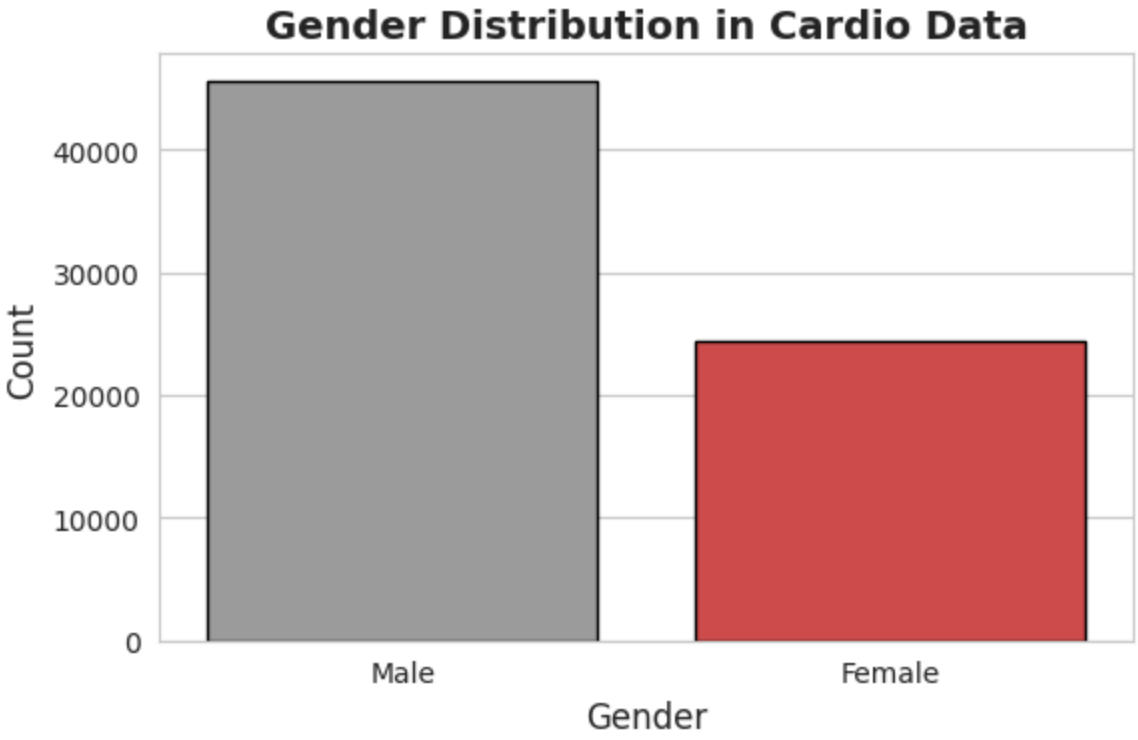
```
RED = "#E53935"
BLACK = "#000000"
GRAY = "#9E9E9E"

# Custom diverging colormap (Negative → Positive: Gray → Black → Red)
custom_cmap = LinearSegmentedColormap.from_list("custom_red_black_gray", [GR

plt.rcParams.update({
    "font.family": "serif",
    "font.size": 12,
    "axes.titlesize": 14,
    "axes.labelsize": 12,
    "xtick.labelsize": 10,
    "ytick.labelsize": 10
})
# ----------------------------
# Save Directory Setup
# ----------------------------
save_dir = "../images" if os.path.exists("../images") else "images"
os.makedirs(save_dir, exist_ok=True)

# ----------------------------
# 4. Correlation Heatmap
# ----------------------------
plt.figure(figsize=(14, 10))
corr = df_merged.corr(numeric_only=True)

sns.heatmap(
    corr,
    annot=True,
    fmt=".2f",
    cmap=[[0.8, 0, 0], [0.85, 0.85, 0.85], [0, 0, 0]],
    linewidths=0.5,
    square=True,
    cbar_kws={"shrink": 0.75},
    annot_kws={"fontsize": 10, "fontname": "serif"}
)


plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.title("Correlation Heatmap of Cardio Data", fontsize=16, fontweight='bol
plt.tight_layout()

# ----------------------------
# Save to root-level images folder
# ----------------------------
plt.savefig(os.path.join(save_dir, "correlation_heatmap.png"), dpi=300, bbox
plt.show()
```



In [7]:
```
import matplotlib.pyplot as plt
import seaborn as sns
import os

# ----------------------------
# Define Brand Colors & Styling
# ----------------------------
RED = "#E53935"      # Strong red
GRAY = "#9E9E9E"     # Soft gray
BLACK = "#000000"    # For text and edges
COLORS = [GRAY, RED]

# Set font and plot settings
plt.rcParams.update({
    "font.family": "Times New Roman",
    "font.size": 12,
    "axes.titlesize": 14,
    "axes.labelsize": 12,
    "xtick.labelsize": 10,
    "ytick.labelsize": 10
})

# Ensure correct save directory
```

```
save_path = "../images"
os.makedirs(save_path, exist_ok=True)

# ----------------------------
# 5. Gender Distribution Plot
# ----------------------------
if 'gender' in df_merged.columns:
    plt.figure(figsize=(6, 4))

    # Map gender values to labels
    gender_labels = {1: 'Male', 2: 'Female'}
    df_merged['gender_label'] = df_merged['gender'].map(gender_labels)

    # Plot
    sns.set_style("whitegrid")
    sns.countplot(data=df_merged, x="gender_label", palette=COLORS, edgecolo

    plt.title("Gender Distribution in Cardio Data", fontsize=14, fontweight=
    plt.xlabel("Gender", fontsize=12)
    plt.ylabel("Count", fontsize=12)
    plt.tight_layout()

    # Save the plot
    plt.savefig(f"{save_path}/gender_distribution.png", dpi=300, bbox_inches
    plt.show()

print("Gender distribution graph saved to ../images/gender_distribution.png"
```

```
/tmp/ipykernel_7257/226452302.py:39: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed
in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the
same effect.

  sns.countplot(data=df_merged, x="gender_label", palette=COLORS, edgecolor=
BLACK)
```




```
Gender distribution graph saved to ../images/gender_distribution.png
```

Release Resources

In [8]:
```
%%html

<p><b>Shutting down your kernel for this notebook to release resources.</b><
<button class="sm-command-button" data-commandlinker-command="kernelmenu:shu

<script>
try {
    els = document.getElementsByClassName("sm-command-button");
    els[0].click();
}
catch(err) {
    // NoOp
}
</script>
```

Shutting down your kernel for this notebook to release resources.

In [9]:
```
%%javascript

try {
    Jupyter.notebook.save_checkpoint();
    Jupyter.notebook.session.delete();
}
catch(err) {
    // NoOp
}
```

In [2]:
```
from sagemaker.automl.automl import AutoML, AutoMLInput
import sagemaker

# 1. Set up SageMaker session and role
sagemaker_session = sagemaker.Session()
role = sagemaker.get_execution_role()

# 2. S3 path to training data
s3_input_path = "s3://cardiovale-solutions-datascience-pipeline/feature-stor

# 3. Define input object
auto_ml_input = AutoMLInput(
    inputs=s3_input_path,
    target_attribute_name="cardio"
)

# 4. Initialize AutoML (pass runtime and candidate settings here)
auto_ml_job = AutoML(
    role=role,
    target_attribute_name="cardio",
    sagemaker_session=sagemaker_session,
    problem_type="BinaryClassification",
    job_objective={"MetricName": "F1"},
    max_candidates=3,
    total_job_runtime_in_seconds=3600
)

# 5. Launch the job
auto_ml_job_name = "cardio-autopilot-model-job"
auto_ml_job.fit(
    inputs=auto_ml_input,
    job_name=auto_ml_job_name,
    wait=True,
    logs=True
)
```

```
[04/05/25 21:26:17] INFO     Creating auto-ml-job with name: cardio-au
.....................................................................
.....................................................................
.....................................
.
```

In [3]:
```
auto_ml_job.describe_auto_ml_job()
```

Out[3]:

```
{'AutoMLJobName': 'cardio-autopilot-model-job',
 'AutoMLJobArn': 'arn:aws:sagemaker:us-east-1:424808199142:automl-job/cardio-autopilot-model-job',
 'InputDataConfig': [{'DataSource': {'S3DataSource': {'S3DataType': 'S3Prefix',
     'S3Uri': 's3://cardiovale-solutions-datascience-pipeline/feature-store/cardio/cardio-feature-group-22-21-14-34/autopilot_input.csv'}},
   'TargetAttributeName': 'cardio',
   'ContentType': 'text/csv;header=present',
   'ChannelType': 'training'}],
 'OutputDataConfig': {'S3OutputPath': 's3://sagemaker-us-east-1-424808199142/'},
 'RoleArn': 'arn:aws:iam::424808199142:role/LabRole',
 'AutoMLJobObjective': {'MetricName': 'F1'},
 'ProblemType': 'BinaryClassification',
 'AutoMLJobConfig': {'CompletionCriteria': {'MaxCandidates': 3,
   'MaxAutoMLJobRuntimeInSeconds': 3600},
  'SecurityConfig': {'EnableInterContainerTrafficEncryption': False}},
 'CreationTime': datetime.datetime(2025, 4, 5, 21, 26, 17, 525000, tzinfo=tzlocal()),
 'EndTime': datetime.datetime(2025, 4, 5, 21, 58, 38, 563000, tzinfo=tzlocal()),
 'LastModifiedTime': datetime.datetime(2025, 4, 5, 21, 58, 38, 592000, tzinfo=tzlocal()),
 'BestCandidate': {'CandidateName': 'cardio-autopilot-model-jobIRt3HM-001-0c4c84bc',
  'FinalAutoMLJobObjectiveMetric': {'MetricName': 'validation:f1_binary',
   'Value': 0.7216299772262573,
   'StandardMetricName': 'F1'},
  'ObjectiveStatus': 'Succeeded',
  'CandidateSteps': [{'CandidateStepType': 'AWS::SageMaker::ProcessingJob',
    'CandidateStepArn': 'arn:aws:sagemaker:us-east-1:424808199142:processing-job/cardio-autopilot-model-job-db-1-f9a3d4ec421a407cb3e75ddeeca71f9',
    'CandidateStepName': 'cardio-autopilot-model-job-db-1-f9a3d4ec421a407cb3e75ddeeca71f9'},
   {'CandidateStepType': 'AWS::SageMaker::TrainingJob',
    'CandidateStepArn': 'arn:aws:sagemaker:us-east-1:424808199142:training-job/cardio-autopilot-model-job-dpp1-1-44f68bc545d843668663adb43ca88',
    'CandidateStepName': 'cardio-autopilot-model-job-dpp1-1-44f68bc545d843668663adb43ca88'},
   {'CandidateStepType': 'AWS::SageMaker::TransformJob',
    'CandidateStepArn': 'arn:aws:sagemaker:us-east-1:424808199142:transform-job/cardio-autopilot-model-job-dpp1-csv-1-e7cedbd78419489db71e7595c',
    'CandidateStepName': 'cardio-autopilot-model-job-dpp1-csv-1-e7cedbd78419489db71e7595c'},
   {'CandidateStepType': 'AWS::SageMaker::TrainingJob',
    'CandidateStepArn': 'arn:aws:sagemaker:us-east-1:424808199142:training-job/cardio-autopilot-model-jobIRt3HM-001-0c4c84bc',
    'CandidateStepName': 'cardio-autopilot-model-jobIRt3HM-001-0c4c84bc'}],
  'CandidateStatus': 'Completed',
  'InferenceContainers': [{'Image': '683313688378.dkr.ecr.us-east-1.amazonaws.com/sagemaker-sklearn-automl:2.5-1-cpu-py3',
    'ModelDataUrl': 's3://sagemaker-us-east-1-424808199142/cardio-autopilot-model-job/data-processor-models/cardio-autopilot-model-job-dpp1-1-44f68bc545d843668663adb43ca88/output/model.tar.gz',
    'Environment': {'AUTOML_TRANSFORM_MODE': 'feature-transform',
```

```
     'SAGEMAKER_DEFAULT_INVOCATIONS_ACCEPT': 'application/x-recordio-protob
uf',
     'SAGEMAKER_PROGRAM': 'sagemaker_serve',
     'SAGEMAKER_SUBMIT_DIRECTORY': '/opt/ml/model/code'}},
   {'Image': '683313688378.dkr.ecr.us-east-1.amazonaws.com/sagemaker-xgboos
t:1.3-1-cpu-py3',
    'ModelDataUrl': 's3://sagemaker-us-east-1-424808199142/cardio-autopilot
-model-job/tuning/cardio-aut-dpp1-xgb/cardio-autopilot-model-jobIRt3HM-001-
0c4c84bc/output/model.tar.gz',
    'Environment': {'MAX_CONTENT_LENGTH': '20971520',
     'SAGEMAKER_DEFAULT_INVOCATIONS_ACCEPT': 'text/csv',
     'SAGEMAKER_INFERENCE_OUTPUT': 'predicted_label',
     'SAGEMAKER_INFERENCE_SUPPORTED': 'predicted_label,probability,probabil
ities'}},
   {'Image': '683313688378.dkr.ecr.us-east-1.amazonaws.com/sagemaker-sklear
n-automl:2.5-1-cpu-py3',
    'ModelDataUrl': 's3://sagemaker-us-east-1-424808199142/cardio-autopilot
-model-job/data-processor-models/cardio-autopilot-model-job-dpp1-1-44f68bc5
45d843668663adb43ca88/output/model.tar.gz',
    'Environment': {'AUTOML_TRANSFORM_MODE': 'inverse-label-transform',
     'SAGEMAKER_DEFAULT_INVOCATIONS_ACCEPT': 'text/csv',
     'SAGEMAKER_INFERENCE_INPUT': 'predicted_label',
     'SAGEMAKER_INFERENCE_OUTPUT': 'predicted_label',
     'SAGEMAKER_INFERENCE_SUPPORTED': 'predicted_label,probability,labels,p
robabilities',
     'SAGEMAKER_PROGRAM': 'sagemaker_serve',
     'SAGEMAKER_SUBMIT_DIRECTORY': '/opt/ml/model/code'}}],
  'CreationTime': datetime.datetime(2025, 4, 5, 21, 41, 6, tzinfo=tzlocal
()),
  'EndTime': datetime.datetime(2025, 4, 5, 21, 44, 31, tzinfo=tzlocal()),
  'LastModifiedTime': datetime.datetime(2025, 4, 5, 21, 47, 4, 840000, tzin
fo=tzlocal()),
  'CandidateProperties': {'CandidateArtifactLocations': {'Explainability':
's3://sagemaker-us-east-1-424808199142/cardio-autopilot-model-job/documenta
tion/explainability/output',
    'ModelInsights': 's3://sagemaker-us-east-1-424808199142/cardio-autopilo
t-model-job/documentation/model_monitor/output'},
   'CandidateMetrics': [{'MetricName': 'F1',
     'Value': 0.7216299772262573,
     'Set': 'Validation',
     'StandardMetricName': 'F1'},
    {'MetricName': 'LogLoss',
     'Value': 0.5458999872207642,
     'Set': 'Validation',
     'StandardMetricName': 'LogLoss'},
    {'MetricName': 'Recall',
     'Value': 0.699400007724762,
     'Set': 'Validation',
     'StandardMetricName': 'Recall'},
    {'MetricName': 'Precision',
     'Value': 0.7453200221061707,
     'Set': 'Validation',
     'StandardMetricName': 'Precision'},
    {'MetricName': 'AUC',
     'Value': 0.7980999946594238,
     'Set': 'Validation',
```

```
     'StandardMetricName': 'AUC'},
    {'MetricName': 'Accuracy',
     'Value': 0.7303799986839294,
     'Set': 'Validation',
     'StandardMetricName': 'Accuracy'},
    {'MetricName': 'BalancedAccuracy',
     'Value': 0.7303599715232849,
     'Set': 'Validation',
     'StandardMetricName': 'BalancedAccuracy'}]}},
 'AutoMLJobStatus': 'Completed',
 'AutoMLJobSecondaryStatus': 'Completed',
 'GenerateCandidateDefinitionsOnly': False,
 'AutoMLJobArtifacts': {'CandidateDefinitionNotebookLocation': 's3://sagema
ker-us-east-1-424808199142/cardio-autopilot-model-job/sagemaker-automl-cand
idates/cardio-autopilot-model-job-pr-1-992392fc6b1b4f51afe2b91209e4ab2/note
books/SageMakerAutopilotCandidateDefinitionNotebook.ipynb',
   'DataExplorationNotebookLocation': 's3://sagemaker-us-east-1-42480819914
2/cardio-autopilot-model-job/sagemaker-automl-candidates/cardio-autopilot-m
odel-job-pr-1-992392fc6b1b4f51afe2b91209e4ab2/notebooks/SageMakerAutopilotD
ataExplorationNotebook.ipynb'},
 'ResolvedAttributes': {'AutoMLJobObjective': {'MetricName': 'F1'},
  'ProblemType': 'BinaryClassification',
  'CompletionCriteria': {'MaxCandidates': 3,
   'MaxRuntimePerTrainingJobInSeconds': 86400,
   'MaxAutoMLJobRuntimeInSeconds': 3600}},
 'ResponseMetadata': {'RequestId': '33a8c1ae-39f8-4597-9bcc-612c9366c282',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amzn-requestid': '33a8c1ae-39f8-4597-9bcc-612c9366c28
2',
   'content-type': 'application/x-amz-json-1.1',
   'content-length': '5982',
   'date': 'Sat, 05 Apr 2025 21:59:14 GMT'},
  'RetryAttempts': 0}}
```

In [ ]:

In [1]:
```
import pandas as pd
import numpy as np
import boto3
import sagemaker
from sagemaker import get_execution_role
from sagemaker.inputs import TrainingInput
from sagemaker.estimator import Estimator
import io

# Step 1: Setup
role = get_execution_role()
region = boto3.session.Session().region_name
session = sagemaker.Session()
bucket = "cardiovale-solutions-datascience-pipeline"
train_key = "xgb/manual/train/train.csv"

# Step 2: Load data with extreme validation
s3_client = boto3.client("s3")
source_key = "feature-store/cardio/cardio-feature-group-22-21-14-34/autopilc
response = s3_client.get_object(Bucket=bucket, Key=source_key)
df = pd.read_csv(response["Body"])

# Step 3: Nuclear option for label cleaning
print("Original label distribution:")
print(df["cardio"].value_counts())

# Convert to numeric, drop NA, then force binary
df["cardio"] = pd.to_numeric(df["cardio"], errors="coerce")
df = df.dropna(subset=["cardio"])
df["cardio"] = np.where(df["cardio"] > 0.5, 1, 0).astype(int)

# Final validation
assert set(df["cardio"].unique()) == {0, 1}, f"Invalid labels found: {df['ca
print("Final label distribution:")
print(df["cardio"].value_counts())

# Step 4: Prepare features
df = df.select_dtypes(include=[np.number])
df = df.replace([np.inf, -np.inf], np.nan).dropna()

# Step 5: Move label to last position
label_col = "cardio"
cols = [c for c in df.columns if c != label_col] + [label_col]
df = df[cols]

# Ensure label is the first column (required for CSV with no header)
label_col = "cardio"
cols = [label_col] + [col for col in df.columns if col != label_col]
df = df[cols]

# Step 6: Create train/test split
train_df = df.sample(frac=0.7, random_state=42)
test_df = df.drop(train_df.index)
print(f"Split complete - Train: {train_df.shape}, Test: {test_df.shape}")
```

```
# Step 7: Save to CSV with explicit formatting
csv_buffer = io.StringIO()
train_df.to_csv(csv_buffer, index=False, header=False, float_format="%.6f")
csv_content = csv_buffer.getvalue()

# Verify the last column
last_col_vals = [row.split(",")[-1].strip() for row in csv_content.split("\r
print(f"Last column sample values: {set(last_col_vals[:100])}")

# Upload with proper content type
s3_client.put_object(
    Bucket=bucket,
    Key=train_key,
    Body=csv_content,
    ContentType="text/csv"
)

# Step 8: Configure training
xgboost_container = sagemaker.image_uris.retrieve("xgboost", region, "1.2-2"

xgb_estimator = Estimator(
    image_uri=xgboost_container,
    role=role,
    instance_count=1,
    instance_type="ml.m5.large",
    output_path=f"s3://{bucket}/xgb/manual/output/",
    sagemaker_session=session,
    base_job_name="cardio-xgboost-final"
)

# Simplified hyperparameters
xgb_estimator.set_hyperparameters(
    objective="binary:logistic",
    num_round=50,
    eval_metric="error",
    verbosity=2  # Increased logging
)

# Step 9: Train with File mode
print("Starting training with verified data...")
xgb_estimator.fit(
    {"train": TrainingInput(
        f"s3://{bucket}/{train_key}",
        content_type="text/csv",
        input_mode="File",
        distribution="ShardedByS3Key"  # Better for large files
    )},
    wait=True,
    logs=True
)
```

```
/opt/conda/lib/python3.11/site-packages/pydantic/_internal/_fields.py:192: U
serWarning: Field name "json" in "MonitoringDatasetFormat" shadows an attrib
ute in parent "Base"
  warnings.warn(
```

```
sagemaker.config INFO - Not applying SDK defaults from location: /etc/xdg/sa
gemaker/config.yaml
sagemaker.config INFO - Not applying SDK defaults from location: /home/sagem
aker-user/.config/sagemaker/config.yaml
Original label distribution:
cardio
0    35021
1    34979
Name: count, dtype: int64
Final label distribution:
cardio
0    35021
1    34979
Name: count, dtype: int64
Split complete - Train: (49000, 16), Test: (21000, 16)
Last column sample values: {'2.000000', '0.000000'}
[04/05/25 21:28:17] INFO     Ignoring unnecessary instance type: None.

Starting training with verified data...
                    INFO     SageMaker Python SDK will collect telemet
                             understand our user's needs, diagnose iss
                             additional features.
                             To opt out of telemetry, please disable v
                             parameter in SDK defaults config. For mor
                             to
                             https://sagemaker.readthedocs.io/en/stabl
                             guring-and-using-defaults-with-the-sagema
                    INFO     Creating training-job with name:
                             cardio-xgboost-final-2025-04-05-21-28-17-
```

```
2025-04-05 21:28:18 Starting - Starting the training job...
..25-04-05 21:28:32 Starting - Preparing the instances for training.
..25-04-05 21:28:56 Downloading - Downloading input data.
.....04-05 21:29:42 Downloading - Downloading the training image.
2025-04-05 21:30:43 Training - Training image download completed. Training i
n progress.
[2025-04-05 21:30:37.387 ip-10-0-107-229.ec2.internal:7 INFO utils.py:27] RU
LE_JOB_STOP_SIGNAL_FILENAME: None
[2025-04-05:21:30:37:INFO] Imported framework sagemaker_xgboost_container.tr
aining
[2025-04-05:21:30:37:INFO] Failed to parse hyperparameter eval_metric value
error to Json.
Returning the value itself
[2025-04-05:21:30:37:INFO] Failed to parse hyperparameter objective value bi
nary:logistic to Json.
Returning the value itself
[2025-04-05:21:30:37:INFO] No GPUs detected (normal if no gpus installed)
[2025-04-05:21:30:37:INFO] Running XGBoost Sagemaker in algorithm mode
[2025-04-05:21:30:37:INFO] Determined delimiter of CSV input is ','
[2025-04-05:21:30:37:INFO] Determined delimiter of CSV input is ','
[2025-04-05:21:30:37:INFO] Single node training.
[2025-04-05:21:30:37:INFO] Train matrix has 49000 rows and 15 columns
[2025-04-05 21:30:37.524 ip-10-0-107-229.ec2.internal:7 INFO json_config.py:
91] Creating hook from json_config at /opt/ml/input/config/debughookconfig.j
son.
[2025-04-05 21:30:37.525 ip-10-0-107-229.ec2.internal:7 INFO hook.py:201] te
nsorboard_dir has not been set for the hook. SMDebug will not be exporting t
ensorboard summaries.
[2025-04-05 21:30:37.526 ip-10-0-107-229.ec2.internal:7 INFO profiler_config
_parser.py:102] User has disabled profiler.
[2025-04-05 21:30:37.526 ip-10-0-107-229.ec2.internal:7 INFO hook.py:255] Sa
ving to /opt/ml/output/tensors
[2025-04-05 21:30:37.527 ip-10-0-107-229.ec2.internal:7 INFO state_store.py:
77] The checkpoint config file /opt/ml/input/config/checkpointconfig.json do
es not exist.
[2025-04-05:21:30:37:INFO] Debug hook created from config
[21:30:37] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 118 ext
ra nodes, 0 pruned nodes, max_depth=6
[0]#011train-error:0.26327
[2025-04-05 21:30:37.618 ip-10-0-107-229.ec2.internal:7 INFO hook.py:423] Mo
nitoring the collections: metrics
[2025-04-05 21:30:37.621 ip-10-0-107-229.ec2.internal:7 INFO hook.py:486] Ho
ok is writing from the hook with pid: 7
[21:30:37] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 118 ext
ra nodes, 0 pruned nodes, max_depth=6
[1]#011train-error:0.26163
[21:30:37] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 122 ext
ra nodes, 0 pruned nodes, max_depth=6
[2]#011train-error:0.25847
[21:30:37] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 124 ext
ra nodes, 0 pruned nodes, max_depth=6
[3]#011train-error:0.25792
[21:30:37] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 124 ext
ra nodes, 0 pruned nodes, max_depth=6
[4]#011train-error:0.25712
[21:30:37] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 126 ext
```

```
ra nodes, 0 pruned nodes, max_depth=6
[5]#011train-error:0.25620
[21:30:37] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 118 ext
ra nodes, 0 pruned nodes, max_depth=6
[6]#011train-error:0.25629
[21:30:37] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 102 ext
ra nodes, 0 pruned nodes, max_depth=6
[7]#011train-error:0.25543
[21:30:37] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 122 ext
ra nodes, 0 pruned nodes, max_depth=6
[8]#011train-error:0.25431
[21:30:38] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 88 extr
a nodes, 0 pruned nodes, max_depth=6
[9]#011train-error:0.25386
[21:30:38] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 76 extr
a nodes, 0 pruned nodes, max_depth=6
[10]#011train-error:0.25333
[21:30:38] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 102 ext
ra nodes, 0 pruned nodes, max_depth=6
[11]#011train-error:0.25310
[21:30:38] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 112 ext
ra nodes, 0 pruned nodes, max_depth=6
[12]#011train-error:0.25222
[21:30:38] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 70 extr
a nodes, 0 pruned nodes, max_depth=6
[13]#011train-error:0.25196
[21:30:38] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 98 extr
a nodes, 0 pruned nodes, max_depth=6
[14]#011train-error:0.25102
[21:30:38] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 76 extr
a nodes, 0 pruned nodes, max_depth=6
[15]#011train-error:0.25055
[21:30:38] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 96 extr
a nodes, 0 pruned nodes, max_depth=6
[16]#011train-error:0.24971
[21:30:38] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 96 extr
a nodes, 0 pruned nodes, max_depth=6
[17]#011train-error:0.24943
[21:30:38] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 90 extr
a nodes, 0 pruned nodes, max_depth=6
[18]#011train-error:0.24929
[21:30:38] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 110 ext
ra nodes, 0 pruned nodes, max_depth=6
[19]#011train-error:0.24827
[21:30:38] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 52 extr
a nodes, 0 pruned nodes, max_depth=6
[20]#011train-error:0.24802
[21:30:38] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 74 extr
a nodes, 0 pruned nodes, max_depth=6
[21]#011train-error:0.24786
[21:30:38] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 110 ext
ra nodes, 0 pruned nodes, max_depth=6
[22]#011train-error:0.24753
[21:30:38] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 26 extr
a nodes, 0 pruned nodes, max_depth=6
[23]#011train-error:0.24739
```

```
[21:30:38] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 72 extr
a nodes, 0 pruned nodes, max_depth=6
[24]#011train-error:0.24661
[21:30:38] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 88 extr
a nodes, 0 pruned nodes, max_depth=6
[25]#011train-error:0.24598
[21:30:38] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 90 extr
a nodes, 0 pruned nodes, max_depth=6
[26]#011train-error:0.24549
[21:30:38] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 48 extr
a nodes, 0 pruned nodes, max_depth=6
[27]#011train-error:0.24514
[21:30:38] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 74 extr
a nodes, 0 pruned nodes, max_depth=6
[28]#011train-error:0.24465
[21:30:38] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 80 extr
a nodes, 0 pruned nodes, max_depth=6
[29]#011train-error:0.24427
[21:30:38] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 108 ext
ra nodes, 0 pruned nodes, max_depth=6
[30]#011train-error:0.24357
[21:30:38] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 108 ext
ra nodes, 0 pruned nodes, max_depth=6
[31]#011train-error:0.24298
[21:30:39] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 74 extr
a nodes, 0 pruned nodes, max_depth=6
[32]#011train-error:0.24265
[21:30:39] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 90 extr
a nodes, 0 pruned nodes, max_depth=6
[33]#011train-error:0.24243
[21:30:39] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 52 extr
a nodes, 0 pruned nodes, max_depth=6
[34]#011train-error:0.24210
[21:30:39] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 62 extr
a nodes, 0 pruned nodes, max_depth=6
[35]#011train-error:0.24184
[21:30:39] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 22 extr
a nodes, 0 pruned nodes, max_depth=6
[36]#011train-error:0.24171
[21:30:39] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 32 extr
a nodes, 0 pruned nodes, max_depth=6
[37]#011train-error:0.24165
[21:30:39] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 100 ext
ra nodes, 0 pruned nodes, max_depth=6
[38]#011train-error:0.24131
[21:30:39] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 36 extr
a nodes, 0 pruned nodes, max_depth=6
[39]#011train-error:0.24106
[21:30:39] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 98 extr
a nodes, 0 pruned nodes, max_depth=6
[40]#011train-error:0.24075
[21:30:39] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 66 extr
a nodes, 0 pruned nodes, max_depth=6
[41]#011train-error:0.24022
[21:30:39] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 86 extr
a nodes, 0 pruned nodes, max_depth=6
```

```
[42]#011train-error:0.23982
[21:30:39] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 100 ext
ra nodes, 0 pruned nodes, max_depth=6
[43]#011train-error:0.23943
[21:30:39] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 36 extr
a nodes, 0 pruned nodes, max_depth=6
[44]#011train-error:0.23935
[21:30:39] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 74 extr
a nodes, 0 pruned nodes, max_depth=6
[45]#011train-error:0.23900
[21:30:39] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 98 extr
a nodes, 0 pruned nodes, max_depth=6
[46]#011train-error:0.23831
[21:30:39] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 44 extr
a nodes, 0 pruned nodes, max_depth=6
[47]#011train-error:0.23804
[21:30:39] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 48 extr
a nodes, 0 pruned nodes, max_depth=6
[48]#011train-error:0.23812
[21:30:39] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 86 extr
a nodes, 0 pruned nodes, max_depth=6
[49]#011train-error:0.23780

2025-04-05 21:31:01 Completed - Training job completed
Training seconds: 125
Billable seconds: 125
```

In [7]:
```
# Deploy the trained model
predictor = xgb_estimator.deploy(
    initial_instance_count=1,
    instance_type='ml.m5.large'
)
```

```
[04/05/25 21:38:13] INFO     Creating model with name: cardio-xgboost-
[04/05/25 21:38:14] INFO     Creating endpoint-config with name
                             cardio-xgboost-final-2025-04-05-21-38-13-
                    INFO     Creating endpoint with name
                             cardio-xgboost-final-2025-04-05-21-38-13-
------!
```

Prepare test data

In [11]:
```
import numpy as np
import json
from sklearn.metrics import confusion_matrix, classification_report

# Step 1: Remove the label column from test set
X_test = test_df.drop("cardio", axis=1)
y_true = test_df["cardio"].values  # Save the true labels

# Step 2: Convert to CSV payload
import io
csv_buffer = io.StringIO()
```

```
X_test.to_csv(csv_buffer, header=False, index=False)
csv_payload = csv_buffer.getvalue()
```

Predict Using the Endpoint

In [12]:
```
# Predict using the endpoint
response = predictor.predict(csv_payload, initial_args={"ContentType": "text
```

Step 4: Convert prediction response to binary class labels

In [14]:
```
# Step 4: Decode and parse line-by-line
decoded_response = response.decode("utf-8") if hasattr(response, "decode") e
y_prob = [float(x) for x in decoded_response.strip().split("\n")]

# Convert probabilities to binary labels
y_pred = [int(round(p)) for p in y_prob]
```

Evaluate with Confusion Matrix and Metrics

In [15]:
```
# Step 5: Evaluate
cm = confusion_matrix(y_true, y_pred)
print("Confusion Matrix:\n", cm)

print("\nClassification Report:\n", classification_report(y_true, y_pred))
```

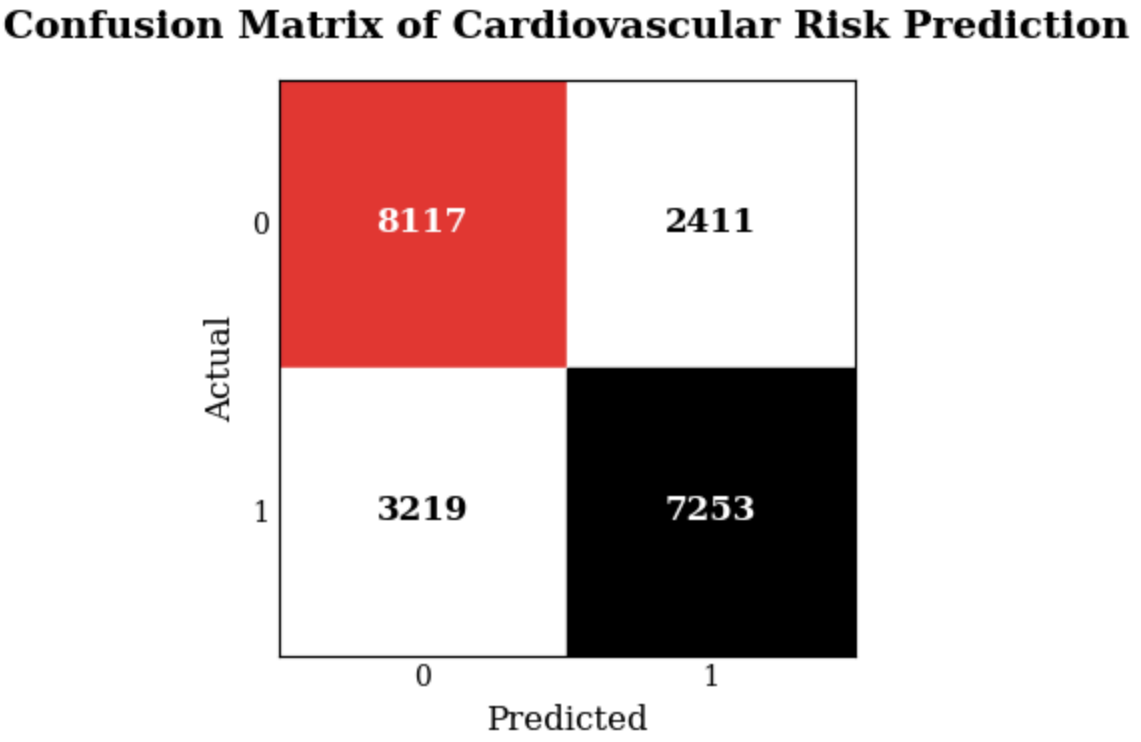
```
Confusion Matrix:
 [[8117 2411]
 [3219 7253]]

Classification Report:
              precision    recall  f1-score   support

           0       0.72      0.77      0.74     10528
           1       0.75      0.69      0.72     10472

    accuracy                           0.73     21000
   macro avg       0.73      0.73      0.73     21000
weighted avg       0.73      0.73      0.73     21000
```

Visualize the Confusion Matrix

In [18]:
```
import os
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

```
# Confusion matrix values (use your actual values here)
cm = np.array([[8117, 2411],
               [3219, 7253]])

# Label map
labels = np.array([["TP", "FP"], ["FN", "TN"]])  # optional, not printed

# Color map manual
square_colors = np.array([["#E53935", "#FFFFFF"],  # Red, White
                          ["#FFFFFF", "#000000"]])  # White, Black

# Set font and layout style
plt.rcParams.update({
    "font.family": "serif",
    "font.size": 12,
    "axes.titlesize": 14,
    "axes.labelsize": 12,
    "xtick.labelsize": 10,
    "ytick.labelsize": 10
})

# Save directory
save_dir = "../images" if os.path.exists("../images") else "images"
os.makedirs(save_dir, exist_ok=True)

# Plot
fig, ax = plt.subplots(figsize=(6, 4))

# Plot each cell manually
for i in range(2):
    for j in range(2):
        ax.add_patch(plt.Rectangle((j, i), 1, 1, color=square_colors[i, j]))
        ax.text(j + 0.5, i + 0.5, f"{cm[i, j]}", va='center', ha='center',
                color='white' if square_colors[i, j] == "#000000" or square_
                fontsize=12, fontweight='bold')

# Set ticks
ax.set_xticks([0.5, 1.5])
ax.set_xticklabels(["0", "1"])
ax.set_yticks([0.5, 1.5])
ax.set_yticklabels(["0", "1"])

# Labels
ax.set_xlabel("Predicted")
ax.set_ylabel("Actual")
ax.set_title("Confusion Matrix of Cardiovascular Risk Prediction", fontsize=

# Aesthetics
ax.set_xlim(0, 2)
ax.set_ylim(0, 2)
ax.invert_yaxis()
ax.set_aspect('equal')
ax.tick_params(axis='both', which='both', length=0)

# Save and show
```

```
plt.tight_layout()
plt.savefig(os.path.join(save_dir, "confusion_matrix_custom.png"), dpi=300,
plt.show()
```



Realease Resources

In [19]:
```
%%html

<p><b>Shutting down your kernel for this notebook to release resources.</b><
<button class="sm-command-button" data-commandlinker-command="kernelmenu:shu

<script>
try {
    els = document.getElementsByClassName("sm-command-button");
    els[0].click();
}
catch(err) {
    // NoOp
}
</script>
```

Shutting down your kernel for this notebook to release resources.

In [ ]:
```
%%javascript

try {
    Jupyter.notebook.save_checkpoint();
    Jupyter.notebook.session.delete();
}
catch(err) {
```

```
    // NoOp
}
```

In [ ]:

In [ ]:
```
from sagemaker.automl.automl import AutoML, AutoMLInput
import sagemaker

# 1. Set up SageMaker session and role
sagemaker_session = sagemaker.Session()
role = sagemaker.get_execution_role()

# 2. S3 path to training data
s3_input_path = "s3://cardiovale-solutions-datascience-pipeline/feature-stor

# 3. Define input object
auto_ml_input = AutoMLInput(
    inputs=s3_input_path,
    target_attribute_name="cardio"
)

# 4. Initialize AutoML (pass runtime and candidate settings here)
auto_ml_job = AutoML(
    role=role,
    target_attribute_name="cardio",
    sagemaker_session=sagemaker_session,
    problem_type="BinaryClassification",
    job_objective={"MetricName": "F1"},
    max_candidates=3,
    total_job_runtime_in_seconds=3600
)

# 5. Launch the job
auto_ml_job_name = "cardio-autopilot-model-job"
auto_ml_job.fit(
    inputs=auto_ml_input,
    job_name=auto_ml_job_name,
    wait=True,
    logs=True
)
```

```
.....................................................................
.........................
..
```

In [4]:
```
auto_ml_job.describe_auto_ml_job()
```

Out[4]:

```
{'AutoMLJobName': 'cardio-autopilot-model-job',
 'AutoMLJobArn': 'arn:aws:sagemaker:us-east-1:786782285170:automl-job/cardio-autopilot-model-job',
 'InputDataConfig': [{'DataSource': {'S3DataSource': {'S3DataType': 'S3Prefix',
     'S3Uri': 's3://cardiovale-solutions-datascience-pipeline/feature-store/cardio/cardio-feature-group-22-21-14-34/autopilot_input.csv'}},
   'TargetAttributeName': 'cardio',
   'ContentType': 'text/csv;header=present',
   'ChannelType': 'training'}],
 'OutputDataConfig': {'S3OutputPath': 's3://sagemaker-us-east-1-786782285170/'},
 'RoleArn': 'arn:aws:iam::786782285170:role/LabRole',
 'AutoMLJobObjective': {'MetricName': 'F1'},
 'ProblemType': 'BinaryClassification',
 'AutoMLJobConfig': {'CompletionCriteria': {'MaxCandidates': 3,
   'MaxAutoMLJobRuntimeInSeconds': 3600},
  'SecurityConfig': {'EnableInterContainerTrafficEncryption': False}},
 'CreationTime': datetime.datetime(2025, 3, 30, 22, 15, 43, 589000, tzinfo=tzlocal()),
 'EndTime': datetime.datetime(2025, 3, 30, 22, 47, 54, 104000, tzinfo=tzlocal()),
 'LastModifiedTime': datetime.datetime(2025, 3, 30, 22, 47, 54, 144000, tzinfo=tzlocal()),
 'BestCandidate': {'CandidateName': 'cardio-autopilot-model-job2t1iNP-001-9db02329',
  'FinalAutoMLJobObjectiveMetric': {'MetricName': 'validation:f1_binary',
   'Value': 0.7171499729156494,
   'StandardMetricName': 'F1'},
  'ObjectiveStatus': 'Succeeded',
  'CandidateSteps': [{'CandidateStepType': 'AWS::SageMaker::ProcessingJob',
    'CandidateStepArn': 'arn:aws:sagemaker:us-east-1:786782285170:processing-job/cardio-autopilot-model-job-db-1-2528ec911e9a4158bc92b1a87e46a2d',
    'CandidateStepName': 'cardio-autopilot-model-job-db-1-2528ec911e9a4158bc92b1a87e46a2d'},
   {'CandidateStepType': 'AWS::SageMaker::TrainingJob',
    'CandidateStepArn': 'arn:aws:sagemaker:us-east-1:786782285170:training-job/cardio-autopilot-model-job-dpp0-1-543a3f80ddbd462dab83e680a9d79',
    'CandidateStepName': 'cardio-autopilot-model-job-dpp0-1-543a3f80ddbd462dab83e680a9d79'},
   {'CandidateStepType': 'AWS::SageMaker::TransformJob',
    'CandidateStepArn': 'arn:aws:sagemaker:us-east-1:786782285170:transform-job/cardio-autopilot-model-job-dpp0-csv-1-87bdb4a6811944d1bc2712e0c',
    'CandidateStepName': 'cardio-autopilot-model-job-dpp0-csv-1-87bdb4a6811944d1bc2712e0c'},
   {'CandidateStepType': 'AWS::SageMaker::TrainingJob',
    'CandidateStepArn': 'arn:aws:sagemaker:us-east-1:786782285170:training-job/cardio-autopilot-model-job2t1iNP-001-9db02329',
    'CandidateStepName': 'cardio-autopilot-model-job2t1iNP-001-9db02329'}],
  'CandidateStatus': 'Completed',
  'InferenceContainers': [{'Image': '683313688378.dkr.ecr.us-east-1.amazonaws.com/sagemaker-sklearn-automl:2.5-1-cpu-py3',
    'ModelDataUrl': 's3://sagemaker-us-east-1-786782285170/cardio-autopilot-model-job/data-processor-models/cardio-autopilot-model-job-dpp0-1-543a3f80ddbd462dab83e680a9d79/output/model.tar.gz',
    'Environment': {'AUTOML_TRANSFORM_MODE': 'feature-transform',
```

```
     'SAGEMAKER_DEFAULT_INVOCATIONS_ACCEPT': 'application/x-recordio-protob
uf',
     'SAGEMAKER_PROGRAM': 'sagemaker_serve',
     'SAGEMAKER_SUBMIT_DIRECTORY': '/opt/ml/model/code'}},
   {'Image': '683313688378.dkr.ecr.us-east-1.amazonaws.com/sagemaker-xgboos
t:1.3-1-cpu-py3',
    'ModelDataUrl': 's3://sagemaker-us-east-1-786782285170/cardio-autopilot
-model-job/tuning/cardio-aut-dpp0-xgb/cardio-autopilot-model-job2t1iNP-001-
9db02329/output/model.tar.gz',
    'Environment': {'MAX_CONTENT_LENGTH': '20971520',
     'SAGEMAKER_DEFAULT_INVOCATIONS_ACCEPT': 'text/csv',
     'SAGEMAKER_INFERENCE_OUTPUT': 'predicted_label',
     'SAGEMAKER_INFERENCE_SUPPORTED': 'predicted_label,probability,probabil
ities'}},
   {'Image': '683313688378.dkr.ecr.us-east-1.amazonaws.com/sagemaker-sklear
n-automl:2.5-1-cpu-py3',
    'ModelDataUrl': 's3://sagemaker-us-east-1-786782285170/cardio-autopilot
-model-job/data-processor-models/cardio-autopilot-model-job-dpp0-1-543a3f80
ddbd462dab83e680a9d79/output/model.tar.gz',
    'Environment': {'AUTOML_TRANSFORM_MODE': 'inverse-label-transform',
     'SAGEMAKER_DEFAULT_INVOCATIONS_ACCEPT': 'text/csv',
     'SAGEMAKER_INFERENCE_INPUT': 'predicted_label',
     'SAGEMAKER_INFERENCE_OUTPUT': 'predicted_label',
     'SAGEMAKER_INFERENCE_SUPPORTED': 'predicted_label,probability,labels,p
robabilities',
     'SAGEMAKER_PROGRAM': 'sagemaker_serve',
     'SAGEMAKER_SUBMIT_DIRECTORY': '/opt/ml/model/code'}}],
  'CreationTime': datetime.datetime(2025, 3, 30, 22, 30, 26, tzinfo=tzlocal
()),
  'EndTime': datetime.datetime(2025, 3, 30, 22, 36, 1, tzinfo=tzlocal()),
  'LastModifiedTime': datetime.datetime(2025, 3, 30, 22, 36, 23, 393000, tz
info=tzlocal()),
  'CandidateProperties': {'CandidateArtifactLocations': {'Explainability':
's3://sagemaker-us-east-1-786782285170/cardio-autopilot-model-job/documenta
tion/explainability/output',
    'ModelInsights': 's3://sagemaker-us-east-1-786782285170/cardio-autopilo
t-model-job/documentation/model_monitor/output'},
   'CandidateMetrics': [{'MetricName': 'F1',
     'Value': 0.7171499729156494,
     'Set': 'Validation',
     'StandardMetricName': 'F1'},
    {'MetricName': 'LogLoss',
     'Value': 0.5542399883270264,
     'Set': 'Validation',
     'StandardMetricName': 'LogLoss'},
    {'MetricName': 'Recall',
     'Value': 0.6944000124931335,
     'Set': 'Validation',
     'StandardMetricName': 'Recall'},
    {'MetricName': 'Precision',
     'Value': 0.7414500117301941,
     'Set': 'Validation',
     'StandardMetricName': 'Precision'},
    {'MetricName': 'AUC',
     'Value': 0.7926700115203857,
     'Set': 'Validation',
```

```
     'StandardMetricName': 'AUC'},
    {'MetricName': 'Accuracy',
     'Value': 0.7263000011444092,
     'Set': 'Validation',
     'StandardMetricName': 'Accuracy'},
    {'MetricName': 'BalancedAccuracy',
     'Value': 0.7262799739837646,
     'Set': 'Validation',
     'StandardMetricName': 'BalancedAccuracy'}]}},
 'AutoMLJobStatus': 'Completed',
 'AutoMLJobSecondaryStatus': 'Completed',
 'GenerateCandidateDefinitionsOnly': False,
 'AutoMLJobArtifacts': {'CandidateDefinitionNotebookLocation': 's3://sagema
ker-us-east-1-786782285170/cardio-autopilot-model-job/sagemaker-automl-cand
idates/cardio-autopilot-model-job-pr-1-03862525603a4d8c8a7e198f5cdf2a3/note
books/SageMakerAutopilotCandidateDefinitionNotebook.ipynb',
  'DataExplorationNotebookLocation': 's3://sagemaker-us-east-1-78678228517
0/cardio-autopilot-model-job/sagemaker-automl-candidates/cardio-autopilot-m
odel-job-pr-1-03862525603a4d8c8a7e198f5cdf2a3/notebooks/SageMakerAutopilotD
ataExplorationNotebook.ipynb'},
 'ResolvedAttributes': {'AutoMLJobObjective': {'MetricName': 'F1'},
  'ProblemType': 'BinaryClassification',
  'CompletionCriteria': {'MaxCandidates': 3,
   'MaxRuntimePerTrainingJobInSeconds': 86400,
   'MaxAutoMLJobRuntimeInSeconds': 3600}},
 'ResponseMetadata': {'RequestId': 'ad341623-9c1a-492f-bcf7-04704540309a',
  'HTTPStatusCode': 200,
  'HTTPHeaders': {'x-amzn-requestid': 'ad341623-9c1a-492f-bcf7-04704540309
a',
   'content-type': 'application/x-amz-json-1.1',
   'content-length': '5984',
   'date': 'Mon, 31 Mar 2025 01:26:33 GMT'},
  'RetryAttempts': 0}}
```

In [ ]:

In [10]:
```
import pandas as pd
import numpy as np
import boto3
import sagemaker
from sagemaker import get_execution_role
from sagemaker.inputs import TrainingInput
from sagemaker.estimator import Estimator
import io

# Step 1: Setup
role = get_execution_role()
region = boto3.session.Session().region_name
session = sagemaker.Session()
bucket = "cardiovale-solutions-datascience-pipeline"
train_key = "xgb/manual/train/train.csv"

# Step 2: Load data with extreme validation
s3_client = boto3.client("s3")
source_key = "feature-store/cardio/cardio-feature-group-22-21-14-34/autopilc
response = s3_client.get_object(Bucket=bucket, Key=source_key)
df = pd.read_csv(response["Body"])

# Step 3: Nuclear option for label cleaning
print("Original label distribution:")
print(df["cardio"].value_counts())

# Convert to numeric, drop NA, then force binary
df["cardio"] = pd.to_numeric(df["cardio"], errors="coerce")
df = df.dropna(subset=["cardio"])
df["cardio"] = np.where(df["cardio"] > 0.5, 1, 0).astype(int)

# Final validation
assert set(df["cardio"].unique()) == {0, 1}, f"Invalid labels found: {df['ca
print("Final label distribution:")
print(df["cardio"].value_counts())

# Step 4: Prepare features
df = df.select_dtypes(include=[np.number])
df = df.replace([np.inf, -np.inf], np.nan).dropna()

# Step 5: Move label to last position
label_col = "cardio"
cols = [c for c in df.columns if c != label_col] + [label_col]
df = df[cols]

# Ensure label is the first column (required for CSV with no header)
label_col = "cardio"
cols = [label_col] + [col for col in df.columns if col != label_col]
df = df[cols]

# Step 6: Create train/test split
train_df = df.sample(frac=0.7, random_state=42)
test_df = df.drop(train_df.index)
print(f"Split complete - Train: {train_df.shape}, Test: {test_df.shape}")
```

```
# Step 7: Save to CSV with explicit formatting
csv_buffer = io.StringIO()
train_df.to_csv(csv_buffer, index=False, header=False, float_format="%.6f")
csv_content = csv_buffer.getvalue()

# Verify the last column
last_col_vals = [row.split(",")[-1].strip() for row in csv_content.split("\r
print(f"Last column sample values: {set(last_col_vals[:100])}")

# Upload with proper content type
s3_client.put_object(
    Bucket=bucket,
    Key=train_key,
    Body=csv_content,
    ContentType="text/csv"
)

# Step 8: Configure training
xgboost_container = sagemaker.image_uris.retrieve("xgboost", region, "1.2-2"

xgb_estimator = Estimator(
    image_uri=xgboost_container,
    role=role,
    instance_count=1,
    instance_type="ml.m5.large",
    output_path=f"s3://{bucket}/xgb/manual/output/",
    sagemaker_session=session,
    base_job_name="cardio-xgboost-final"
)

# Simplified hyperparameters
xgb_estimator.set_hyperparameters(
    objective="binary:logistic",
    num_round=50,
    eval_metric="error",
    verbosity=2  # Increased logging
)

# Step 9: Train with File mode
print("Starting training with verified data...")
xgb_estimator.fit(
    {"train": TrainingInput(
        f"s3://{bucket}/{train_key}",
        content_type="text/csv",
        input_mode="File",
        distribution="ShardedByS3Key"  # Better for large files
    )},
    wait=True,
    logs=True
)
```

```
Original label distribution:
cardio
0    35021
1    34979
Name: count, dtype: int64
Final label distribution:
cardio
0    35021
1    34979
Name: count, dtype: int64
Split complete - Train: (49000, 16), Test: (21000, 16)
Last column sample values: {'0.000000', '2.000000'}
```

```
INFO:sagemaker.image_uris:Ignoring unnecessary instance type: None.
INFO:sagemaker:Creating training-job with name: cardio-xgboost-final-2025-03
-31-05-50-21-881
```

```
Starting training with verified data...
2025-03-31 05:50:24 Starting - Starting the training job...
2025-03-31 05:50:39 Starting - Preparing the instances for training...
2025-03-31 05:51:02 Downloading - Downloading input data...
2025-03-31 05:51:52 Downloading - Downloading the training image......
2025-03-31 05:52:54 Training - Training image download completed. Training i
n progress.
2025-03-31 05:52:54 Uploading - Uploading generated training model[2025-03-3
1 05:52:45.462 ip-10-0-223-231.ec2.internal:8 INFO utils.py:27] RULE_JOB_STO
P_SIGNAL_FILENAME: None
[2025-03-31:05:52:45:INFO] Imported framework sagemaker_xgboost_container.tr
aining
[2025-03-31:05:52:45:INFO] Failed to parse hyperparameter eval_metric value
error to Json.
Returning the value itself
[2025-03-31:05:52:45:INFO] Failed to parse hyperparameter objective value bi
nary:logistic to Json.
Returning the value itself
[2025-03-31:05:52:45:INFO] No GPUs detected (normal if no gpus installed)
[2025-03-31:05:52:45:INFO] Running XGBoost Sagemaker in algorithm mode
[2025-03-31:05:52:45:INFO] Determined delimiter of CSV input is ','
[2025-03-31:05:52:45:INFO] Determined delimiter of CSV input is ','
[2025-03-31:05:52:45:INFO] Single node training.
[2025-03-31:05:52:45:INFO] Train matrix has 49000 rows and 15 columns
[2025-03-31 05:52:45.599 ip-10-0-223-231.ec2.internal:8 INFO json_config.py:
91] Creating hook from json_config at /opt/ml/input/config/debughookconfig.j
son.
[2025-03-31 05:52:45.600 ip-10-0-223-231.ec2.internal:8 INFO hook.py:201] te
nsorboard_dir has not been set for the hook. SMDebug will not be exporting t
ensorboard summaries.
[2025-03-31 05:52:45.601 ip-10-0-223-231.ec2.internal:8 INFO profiler_config
_parser.py:102] User has disabled profiler.
[2025-03-31 05:52:45.602 ip-10-0-223-231.ec2.internal:8 INFO hook.py:255] Sa
ving to /opt/ml/output/tensors
[2025-03-31 05:52:45.602 ip-10-0-223-231.ec2.internal:8 INFO state_store.py:
77] The checkpoint config file /opt/ml/input/config/checkpointconfig.json do
es not exist.
[2025-03-31:05:52:45:INFO] Debug hook created from config
[05:52:45] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 118 ext
ra nodes, 0 pruned nodes, max_depth=6
[0]#011train-error:0.26310
[2025-03-31 05:52:45.710 ip-10-0-223-231.ec2.internal:8 INFO hook.py:423] Mo
nitoring the collections: metrics
[2025-03-31 05:52:45.712 ip-10-0-223-231.ec2.internal:8 INFO hook.py:486] Ho
ok is writing from the hook with pid: 8
[05:52:45] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 118 ext
ra nodes, 0 pruned nodes, max_depth=6
[1]#011train-error:0.26178
[05:52:45] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 122 ext
ra nodes, 0 pruned nodes, max_depth=6
[2]#011train-error:0.25853
[05:52:45] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 124 ext
ra nodes, 0 pruned nodes, max_depth=6
[3]#011train-error:0.25771
[05:52:45] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 124 ext
ra nodes, 0 pruned nodes, max_depth=6
```

```
[4]#011train-error:0.25677
[05:52:45] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 126 ext
ra nodes, 0 pruned nodes, max_depth=6
[5]#011train-error:0.25649
[05:52:45] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 120 ext
ra nodes, 0 pruned nodes, max_depth=6
[6]#011train-error:0.25659
[05:52:46] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 98 extr
a nodes, 0 pruned nodes, max_depth=6
[7]#011train-error:0.25625
[05:52:46] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 120 ext
ra nodes, 0 pruned nodes, max_depth=6
[8]#011train-error:0.25484
[05:52:46] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 84 extr
a nodes, 0 pruned nodes, max_depth=6
[9]#011train-error:0.25398
[05:52:46] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 90 extr
a nodes, 0 pruned nodes, max_depth=6
[10]#011train-error:0.25376
[05:52:46] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 82 extr
a nodes, 0 pruned nodes, max_depth=6
[11]#011train-error:0.25312
[05:52:46] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 110 ext
ra nodes, 0 pruned nodes, max_depth=6
[12]#011train-error:0.25163
[05:52:46] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 114 ext
ra nodes, 0 pruned nodes, max_depth=6
[13]#011train-error:0.25120
[05:52:46] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 86 extr
a nodes, 0 pruned nodes, max_depth=6
[14]#011train-error:0.25045
[05:52:46] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 110 ext
ra nodes, 0 pruned nodes, max_depth=6
[15]#011train-error:0.24947
[05:52:46] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 78 extr
a nodes, 0 pruned nodes, max_depth=6
[16]#011train-error:0.24916
[05:52:46] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 66 extr
a nodes, 0 pruned nodes, max_depth=6
[17]#011train-error:0.24874
[05:52:46] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 60 extr
a nodes, 0 pruned nodes, max_depth=6
[18]#011train-error:0.24843
[05:52:46] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 86 extr
a nodes, 0 pruned nodes, max_depth=6
[19]#011train-error:0.24796
[05:52:46] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 102 ext
ra nodes, 0 pruned nodes, max_depth=6
[20]#011train-error:0.24759
[05:52:46] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 84 extr
a nodes, 0 pruned nodes, max_depth=6
[21]#011train-error:0.24733
[05:52:46] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 24 extr
a nodes, 0 pruned nodes, max_depth=6
[22]#011train-error:0.24716
[05:52:46] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 112 ext
```

```
ra nodes, 0 pruned nodes, max_depth=6
[23]#011train-error:0.24667
[05:52:46] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 102 ext
ra nodes, 0 pruned nodes, max_depth=6
[24]#011train-error:0.24618
[05:52:46] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 72 extr
a nodes, 0 pruned nodes, max_depth=6
[25]#011train-error:0.24594
[05:52:46] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 90 extr
a nodes, 0 pruned nodes, max_depth=6
[26]#011train-error:0.24557
[05:52:47] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 40 extr
a nodes, 0 pruned nodes, max_depth=6
[27]#011train-error:0.24574
[05:52:47] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 54 extr
a nodes, 0 pruned nodes, max_depth=6
[28]#011train-error:0.24531
[05:52:47] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 112 ext
ra nodes, 0 pruned nodes, max_depth=6
[29]#011train-error:0.24480
[05:52:47] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 108 ext
ra nodes, 0 pruned nodes, max_depth=6
[30]#011train-error:0.24404
[05:52:47] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 94 extr
a nodes, 0 pruned nodes, max_depth=6
[31]#011train-error:0.24359
[05:52:47] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 108 ext
ra nodes, 0 pruned nodes, max_depth=6
[32]#011train-error:0.24288
[05:52:47] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 40 extr
a nodes, 0 pruned nodes, max_depth=6
[33]#011train-error:0.24274
[05:52:47] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 108 ext
ra nodes, 0 pruned nodes, max_depth=6
[34]#011train-error:0.24206
[05:52:47] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 68 extr
a nodes, 0 pruned nodes, max_depth=6
[35]#011train-error:0.24202
[05:52:47] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 48 extr
a nodes, 0 pruned nodes, max_depth=6
[36]#011train-error:0.24184
[05:52:47] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 102 ext
ra nodes, 0 pruned nodes, max_depth=6
[37]#011train-error:0.24143
[05:52:47] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 44 extr
a nodes, 0 pruned nodes, max_depth=6
[38]#011train-error:0.24124
[05:52:47] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 16 extr
a nodes, 0 pruned nodes, max_depth=6
[39]#011train-error:0.24122
[05:52:47] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 86 extr
a nodes, 0 pruned nodes, max_depth=6
[40]#011train-error:0.24086
[05:52:47] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 94 extr
a nodes, 0 pruned nodes, max_depth=6
[41]#011train-error:0.24008
```

```
[05:52:47] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 86 extr
a nodes, 0 pruned nodes, max_depth=6
[42]#011train-error:0.23971
[05:52:47] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 118 ext
ra nodes, 0 pruned nodes, max_depth=6
[43]#011train-error:0.23898
[05:52:47] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 64 extr
a nodes, 0 pruned nodes, max_depth=6
[44]#011train-error:0.23865
[05:52:47] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 62 extr
a nodes, 0 pruned nodes, max_depth=6
[45]#011train-error:0.23843
[05:52:47] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 62 extr
a nodes, 0 pruned nodes, max_depth=6
[46]#011train-error:0.23802
[05:52:47] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 54 extr
a nodes, 0 pruned nodes, max_depth=6
[47]#011train-error:0.23775
[05:52:47] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 92 extr
a nodes, 0 pruned nodes, max_depth=6
[48]#011train-error:0.23704
[05:52:47] INFO: ../src/tree/updater_prune.cc:101: tree pruning end, 98 extr
a nodes, 0 pruned nodes, max_depth=6
[49]#011train-error:0.23633

2025-03-31 05:53:07 Completed - Training job completed
Training seconds: 125
Billable seconds: 125
```

In [14]:
```
# Deploying The Predictor
xgb_predictor = xgb_estimator.deploy(initial_instance_count=1, instance_type
```

```
INFO:sagemaker:Creating model with name: cardio-xgboost-final-2025-03-31-06-
02-28-509
```

```
---------------------------------------------------------------------------
ClientError                               Traceback (most recent call last)
Cell In[14], line 2
      1 # Deploying The Predictor
----> 2 xgb_predictor = xgb_estimator.deploy(initial_instance_count=1, insta
nce_type='ml.m5.large')

File /opt/conda/lib/python3.11/site-packages/sagemaker/estimator.py:1684, in
EstimatorBase.deploy(self, initial_instance_count, instance_type, serialize
r, deserializer, accelerator_type, endpoint_name, use_compiled_model, wait,
model_name, kms_key, data_capture_config, tags, serverless_inference_config,
async_inference_config, volume_size, model_data_download_timeout, container_
startup_health_check_timeout, inference_recommendation_id, explainer_config,
**kwargs)
   1678 model.name = model_name
   1680 tags = update_inference_tags_with_jumpstart_training_tags(
   1681     inference_tags=format_tags(tags), training_tags=self.tags
   1682 )
-> 1684 return model.deploy(
   1685     instance_type=instance_type,
   1686     initial_instance_count=initial_instance_count,
   1687     serializer=serializer,
   1688     deserializer=deserializer,
   1689     accelerator_type=accelerator_type,
   1690     endpoint_name=endpoint_name,
   1691     tags=tags or self.tags,
   1692     wait=wait,
   1693     kms_key=kms_key,
   1694     data_capture_config=data_capture_config,
   1695     serverless_inference_config=serverless_inference_config,
   1696     async_inference_config=async_inference_config,
   1697     explainer_config=explainer_config,
   1698     volume_size=volume_size,
   1699     model_data_download_timeout=model_data_download_timeout,
   1700     container_startup_health_check_timeout=container_startup_health_
check_timeout,
   1701     inference_recommendation_id=inference_recommendation_id,
   1702 )

File /opt/conda/lib/python3.11/site-packages/sagemaker/model.py:1695, in Mod
el.deploy(self, initial_instance_count, instance_type, serializer, deseriali
zer, accelerator_type, endpoint_name, tags, kms_key, wait, data_capture_conf
ig, async_inference_config, serverless_inference_config, volume_size, model_
data_download_timeout, container_startup_health_check_timeout, inference_rec
ommendation_id, explainer_config, accept_eula, endpoint_logging, resources,
endpoint_type, managed_instance_scaling, inference_component_name, routing_c
onfig, **kwargs)
   1692     return None
   1694 else:  # existing single model endpoint path
-> 1695     self._create_sagemaker_model(
   1696         instance_type=instance_type,
   1697         accelerator_type=accelerator_type,
   1698         tags=tags,
   1699         serverless_inference_config=serverless_inference_config,
   1700         **kwargs,
   1701     )
```

```
   1702     serverless_inference_config_dict = (
   1703         serverless_inference_config._to_request_dict() if is_serverl
ess else None
   1704     )
   1705     production_variant = sagemaker.production_variant(
   1706         self.name,
   1707         instance_type,
   (...)
   1714         routing_config=routing_config,
   1715     )

File /opt/conda/lib/python3.11/site-packages/sagemaker/model.py:980, in Mode
l._create_sagemaker_model(self, instance_type, accelerator_type, tags, serve
rless_inference_config, accept_eula, model_reference_arn)
    966 self.env = resolve_nested_dict_value_from_config(
    967     self.env,
    968     ["Environment"],
    969     MODEL_CONTAINERS_PATH,
    970     sagemaker_session=self.sagemaker_session,
    971 )
    972 create_model_args = dict(
    973     name=self.name,
    974     role=self.role,
   (...)
    978     tags=format_tags(tags),
    979 )
--> 980 self.sagemaker_session.create_model(**create_model_args)

File /opt/conda/lib/python3.11/site-packages/sagemaker/session.py:3976, in S
ession.create_model(self, name, role, container_defs, vpc_config, enable_net
work_isolation, primary_container, tags)
   3973         else:
   3974             raise
-> 3976 self._intercept_create_request(create_model_request, submit, self.cr
eate_model.__name__)
   3977 return name

File /opt/conda/lib/python3.11/site-packages/sagemaker/session.py:6514, in S
ession._intercept_create_request(self, request, create, func_name)
   6497 def _intercept_create_request(
   6498     self,
   6499     request: typing.Dict,
   (...)
   6502     # pylint: disable=unused-argument
   6503 ):
   6504     """This function intercepts the create job request.
   6505
   6506     PipelineSession inherits this Session class and will override
   (...)
   6512         func_name (str): the name of the function needed interceptin
g
   6513     """
-> 6514     return create(request)

File /opt/conda/lib/python3.11/site-packages/sagemaker/session.py:3964, in S
ession.create_model.<locals>.submit(request)
```

```
   3962 logger.debug("CreateModel request: %s", json.dumps(request, indent=
4))
   3963 try:
-> 3964     self.sagemaker_client.create_model(**request)
   3965 except ClientError as e:
   3966     error_code = e.response["Error"]["Code"]

File /opt/conda/lib/python3.11/site-packages/botocore/client.py:569, in Clie
ntCreator._create_api_method.<locals>._api_call(self, *args, **kwargs)
    565     raise TypeError(
    566         f"{py_operation_name}() only accepts keyword arguments."
    567     )
    568 # The "self" in this scope is referring to the BaseClient.
--> 569 return self._make_api_call(operation_name, kwargs)

File /opt/conda/lib/python3.11/site-packages/botocore/client.py:1023, in Bas
eClient._make_api_call(self, operation_name, api_params)
   1019     error_code = error_info.get("QueryErrorCode") or error_info.get(
   1020         "Code"
   1021     )
   1022     error_class = self.exceptions.from_code(error_code)
-> 1023     raise error_class(parsed_response, operation_name)
   1024 else:
   1025     return parsed_response

ClientError: An error occurred (AccessDeniedException) when calling the Crea
teModel operation: User: arn:aws:sts::786782285170:assumed-role/LabRole/Sage
Maker is not authorized to perform: sagemaker:CreateModel on resource: arn:a
ws:sagemaker:us-east-1:786782285170:model/cardio-xgboost-final-2025-03-31-06
-02-28-509 with an explicit deny in an identity-based policy
```

In [16]:

```
from sagemaker.serializers import CSVSerializer

# Use test_df (or whatever variable holds your test data) instead of test
test_array = test_df.drop(['cardio'], axis=1).values

# Set the serializer for converting input to CSV format
xgb_predictor.serializer = CSVSerializer()

# Make predictions
predictions = xgb_predictor.predict(test_array)

# If predictions come back as bytes, decode and clean them
if isinstance(predictions, bytes):
    predictions = predictions.decode('utf-8').strip()

# Convert the prediction string into a NumPy array
predictions_array = np.fromstring(predictions, sep=',')
print(predictions_array.shape)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[16], line 7
      4 test_array = test_df.drop(['cardio'], axis=1).values
      6 # Set the serializer for converting input to CSV format
----> 7 xgb_predictor.serializer = CSVSerializer()
      9 # Make predictions
     10 predictions = xgb_predictor.predict(test_array)

NameError: name 'xgb_predictor' is not defined
```

In [ ]: