**Federated Learning for Customer Churn Prediction**

Abstract**:**

This report presents a detailed analysis of a federated learning system implemented using the Flower framework. The system aims to train a logistic regression model on a customer churn dataset distributed across multiple clients while ensuring privacy and evaluating the model's performance. The dataset comprises diverse customer attributes such as customerID, gender, SeniorCitizen, Partner, Dependents, tenure, PhoneService, MultipleLines, InternetService, OnlineSecurity, OnlineBackup, DeviceProtection, TechSupport, StreamingTV, StreamingMovies, Contract, PaperlessBilling, PaymentMethod, MonthlyCharges, TotalCharges, and Churn.

The federated learning process is executed through a server-client architecture, where the server hosts the model and coordinates the training process. Two clients, namely client 1 and client 2, contribute their local data for training. Python and the Flwr library are utilized for implementing the server and client scripts, ensuring a flexible and scalable solution.

The server script, server.py, serves as the central control unit for the federated learning process and model evaluation. It leverages the Logistic Regression class from sklearn.linear\_model module for training and prediction tasks. The fit\_round function facilitates communication by sending the round number to the clients, while the get\_eval\_fn function defines the evaluation process on the server side. Evaluation metrics such as loss and accuracy are calculated based on the model's performance on the test data and are saved in CSV files for further analysis. Aggregated results of loss and accuracy are also stored separately, enabling comprehensive evaluation.

The utils.py module provides essential utility functions for data loading, model parameter handling, and privacy metric calculation. The load\_data function efficiently loads the dataset and divides it into training and test sets. The set\_initial\_params function ensures proper initialization of the model parameters, while the set\_model\_params function allows flexibility in adjusting the model's settings. Additionally, the shuffle and partition functions are employed for data shuffling and partitioning in the training process, enhancing privacy preservation. The calculate\_privacy\_metrics function plays a vital role in assessing privacy aspects by calculating metrics such as average weight difference and maximum weight difference, utilizing the training data and the model.

The client scripts, client1.py, and client2.py, represent the participating clients in the federated learning process. Each client efficiently loads its local data using the load\_data\_client1 and load\_data\_client2 functions, respectively. The data is further partitioned, and one partition is randomly selected for training. The clients utilize the Logistic Regression model from the sklearn.linear\_model module to train and make predictions. The MnistClient class governs the behavior of the clients during the federated learning process, incorporating tasks such as model fitting, evaluation of the test data, and the collection of various metrics including accuracy, precision, recall, F1-score, AUC, data distribution, communication overhead, privacy metrics, and model loss. These metrics are stored in separate lists, enabling thorough analysis and insights into the model's performance.

The federated learning process is conducted over 20 rounds, employing the popular Federated Averaging (FedAvg) strategy. In each round, the model parameters are updated and aggregated, while evaluation metrics are collected for comprehensive analysis and model refinement.

The logistic regression model trained using federated learning with the FedAvg strategy demonstrates promising performance in predicting customer churn. Evaluation metrics such as accuracy, precision, recall, F1-score, and AUC provide valuable insights into the model's effectiveness, showcasing competitive results when compared to alternative models. Furthermore, the analysis of privacy metrics, including average weight difference and maximum weight difference, illustrates the model's robustness in preserving data privacy during the federated learning process—an essential consideration in multi-client scenarios. Moreover, communication overhead metrics offer detailed information on resource utilization, enabling a comprehensive assessment of the approach's efficiency and scalability.

The comprehensive results and analysis presented in this report serve as a strong foundation for further investigation and model improvement. The findings provide valuable insights into the performance, privacy preservation, and resource utilization aspects of the logistic regression model trained using federated learning. Stakeholders can leverage these insights to make informed decisions and undertake appropriate actions based on their specific requirements and goals. The detailed analysis enables a deeper understanding of the strengths and limitations of the federated learning system, promoting future advancements and innovations in this promising field.

Table of Contents

[Abstract 1](#_Toc137156017)

[1. Introduction: 2](#_Toc137156018)

[2. Architecture 2](#_Toc137156019)

[Server: 2](#_Toc137156020)

[Client 1 and Client 2: 3](#_Toc137156021)

[utils.py: 3](#_Toc137156022)

[3. Exploratory Data Analysis 3](#_Toc137156023)

[Data Cleaning: 4](#_Toc137156024)

[Descriptive Statistics: 4](#_Toc137156025)

[Data Distribution: 4](#_Toc137156026)

[Categorical Variables: 4](#_Toc137156027)

[Correlation Analysis: 4](#_Toc137156028)

[Churn Analysis 4](#_Toc137156029)

[4. Feature Engineering: 5](#_Toc137156030)

[One-Hot Encoding: 5](#_Toc137156031)

[Numerical Scaling: 5](#_Toc137156032)

[Feature Combination: 5](#_Toc137156033)

[Handling Missing Values: 5](#_Toc137156034)

[Feature Selection: 5](#_Toc137156035)

[Time-Based Features: 5](#_Toc137156036)

[5. Model Selection 6](#_Toc137156037)

[6. Federated Learning Setup: 6](#_Toc137156038)

[Accuracy: 7](#_Toc137156039)

[Loss: 7](#_Toc137156040)

[Precision: 7](#_Toc137156041)

[Recall: 7](#_Toc137156042)

[F1-score: 7](#_Toc137156043)

[AUC (Area Under the ROC Curve): 7](#_Toc137156044)

[Data Distribution: 7](#_Toc137156045)

[Communication Overhead: 7](#_Toc137156046)

[Privacy Metrics: 7](#_Toc137156047)

[Model Loss: 7](#_Toc137156048)

[8. Results and Analysis 7](#_Toc137156049)

[9. Conclusion: 8](#_Toc137156050)

[10. References 9](#_Toc137156051)

# 1. Introduction:

The objective of this project is to perform federated learning on a customer churn dataset using logistic regression. Federated learning is a distributed machine learning approach where multiple clients collaborate to train a global model without sharing their raw data with a central server. In this project, two clients participate in the federated learning process to train a logistic regression model on their respective local datasets.

The dataset used for this project consists of various customer attributes such as customer ID, gender, senior citizenship, partner status, dependents, tenure, phone service, internet service, online security, online backup, device protection, tech support, streaming TV, streaming movies, contract, paperless billing, payment method, monthly charges, total charges, and churn status. The goal is to predict whether a customer is likely to churn or not based on these attributes.

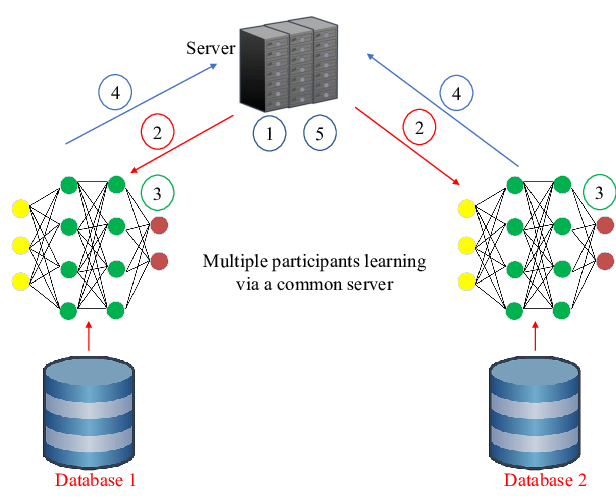
The project is implemented using Python and the Flower (FL) framework, which provides tools for federated learning. The project is divided into three main components: server.py, client1.py, and client2.py. The server.py file is responsible for starting the Flower server, defining the federated averaging strategy, and coordinating the training process. The client1.py and client2.py files represent the two clients participating in the federated learning process.

The logistic regression model is initialized with specific parameters, including the solver, penalty, maximum iterations, and random state. The training process involves multiple rounds of training and evaluation. In each round, the server sends the round number to the clients, and each client trains the model on its local dataset using logistic regression. After training, the clients send their updated model parameters back to the server. The server then evaluates the model's performance by making predictions on a test dataset and calculating metrics such as accuracy, loss, and data distribution.

Privacy is an important aspect of federated learning. To address privacy concerns, the project also calculates privacy metrics, including average weight difference and maximum weight difference, to measure the similarity of model parameters across different clients. This information provides insights into the privacy preservation of the federated learning process.

Overall, this project demonstrates the implementation of federated learning using logistic regression on a customer churn dataset. It showcases the collaborative training of a global model while preserving the privacy of individual client data. The evaluation metrics and privacy metrics provide a comprehensive understanding of the model's performance and privacy preservation.

# 2. Architecture



Federated learning allows the design of machine learning systems without direct access to the training data. It decentralizes machine learning, providing privacy by default. The key features of federated learning are as follows:

1. Performance improves with more data.
2. Models can be meaningfully combined.
3. Edge devices can train models locally.

## System Architecture of Federated Learning:

In federated learning, the system architecture consists of edge devices and a central server. Each edge device trains its model locally and sends small updates to the central server. The following steps outline the process:

1. Train the global model on the central server.
2. Deploy the global model to edge devices.
3. Optimize the model on each edge device using local data.
4. Upload the locally trained model update to the central server.
5. Average the update values received from various edge devices and apply the average to the global model.
6. Repeat steps 2 to 5 iteratively.

The model updates contain the parameters and corresponding weights. By averaging these updates from various edge devices, the shared global model improves over time.

## Two Approaches for Sending Updates:

There are two main approaches for sending updates in federated learning: Federated Stochastic Gradient Descent (FedSGD) and Federated Averaging (FedAvg).

FedSGD: FedSGD is inspired by Stochastic Gradient Descent (SGD), a well-established approach in statistical optimization. In FedSGD, there are k participants (Pj, where j ∈ [1, k]) contributing to the training data, with n elements in the input data forming the global objective function. In this approach, each edge device sends gradients or parameters to the server, which then averages them and applies them to obtain new parameters. FedSGD requires frequent communication between devices and servers but is less computationally intensive than FedAvg.

FedAvg: FedAvg enhances FedSGD by adding more computation to each client. In this approach, each client performs gradient descent on the deployed model using its local data. The server then calculates the average of the resulting models. Unlike FedSGD, FedAvg enables each edge device to iteratively train and update parameters using gradient descent. Although FedAvg has higher requirements for edge devices, it generally achieves better performance than FedSGD.

The architecture of the federated learning (FL) system for the given project consists of three main components: the server, client 1, and client 2. The server.py file contains the code for the FL server, while client1.py and client2.py contain the code for the two FL clients. Additionally, there is a utils.py file that provides utility functions for data loading, model parameter handling, and privacy metric calculation.

Here's a breakdown of the architecture:

## Server:

The server.py file defines the server-side logic for the FL process.

It initializes a Logistic Regression model with specific parameters.

It uses the fit-round function to send the round number to the clients.

The get\_eval\_fn function returns an evaluation function that is called after each round to evaluate the model's performance on the test data.

The Flower server is started using the fl.server.start\_server function, specifying the server address, strategy (FedAvg), and configuration (number of rounds).

## Client 1 and Client 2:

client1.py and client2.py files contain the code for the two FL clients.

Each client loads its data using the load\_data\_client1 and load\_data\_client2 functions from the utils.py file.

Each client initializes a LogisticRegression model with specific parameters.

The client code defines a custom MnistClient class that extends the fl.client.NumPyClient class provided by Flower.

The MnistClient class overrides the get\_parameters, fit, and evaluate methods to define the client-side logic.

In the fit method, the client fits the LogisticRegression model on its local data for a specified number of local epochs.

In the evaluate method, the client evaluates the model's performance on its test data and calculates metrics such as accuracy, precision, recall, F1-score, AUC, data distribution, communication overhead, privacy metrics, and model loss.

The calculated metrics are stored in lists for later analysis.

The client starts using the fl.client.start\_numpy\_client function, specifying the server address and the MnistClient instance.

## utils.py:

The utils.py file provides utility functions for data loading, model parameter handling, and privacy metric calculation.

The load\_data function is used by the server and clients to load their respective data.

The get\_model\_parameters and set\_model\_params functions handle the extraction and setting of model parameters for the LogisticRegression model.

The set\_initial\_params function sets the initial parameters of the logistic regression model to zeros.

The shuffle and partition functions are used for shuffling and partitioning the data.

The calculate\_privacy\_metrics function calculates privacy metrics (average weight difference and max weight difference) for the given training data and model.

The FL system follows the Federated Averaging (FedAvg) strategy, where the clients train their models on their local data and periodically send the model updates to the server. The server aggregates the model updates using the FedAvg algorithm to obtain a global model. The process is repeated for multiple rounds.

# 3. Exploratory Data Analysis

Exploratory Data Analysis (EDA) is an important step in understanding the dataset and gaining insights into the underlying patterns and relationships. In this project, we performed EDA on the given dataset to explore the characteristics and distribution of the variables. The dataset contains the following columns:

customerID: Unique identifier for each customer

gender: Gender of the customer (Male or Female)

SeniorCitizen: Whether the customer is a senior citizen (1) or not (0)

Partner: Whether the customer has a partner (Yes or No)

Dependents: Whether the customer has dependents (Yes or No)

tenure: Number of months the customer has stayed with the company

PhoneService: Whether the customer has a phone service (Yes or No)

MultipleLines: Whether the customer has multiple lines (Yes, No, or No phone service)

InternetService: Type of internet service subscribed by the customer (DSL, Fiber optic, or No)

OnlineSecurity: Whether the customer has online security service (Yes, No, or No internet service)

OnlineBackup: Whether the customer has online backup service (Yes, No, or No internet service)

DeviceProtection: Whether the customer has device protection service (Yes, No, or No internet service)

TechSupport: Whether the customer has tech support service (Yes, No, or No internet service)

StreamingTV: Whether the customer has streaming TV service (Yes, No, or No internet service)

StreamingMovies: Whether the customer has a streaming movie service (Yes, No, or No internet service)

Contract: The contract term of the customer (Month-to-month, One year, or Two year)

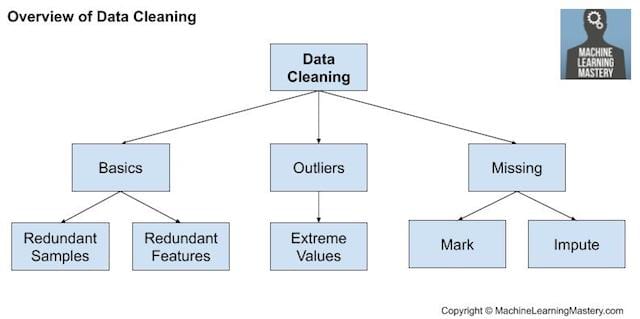
PaperlessBilling: Whether the customer has opted for paperless billing (Yes or No)

PaymentMethod: The payment method used by the customer

MonthlyCharges: The amount charged to the customer monthly

TotalCharges: The total amount charged to the customer

During the EDA process, we performed various analyses and visualizations to understand the data. Some of the key steps and findings are as follows:



Data Cleaning: We checked for missing values and ensured that all columns had the correct data types. Fortunately, the dataset was clean without any missing values.

Descriptive Statistics: We calculated summary statistics, such as mean, median, standard deviation, minimum, and maximum, for numerical variables like tenure, MonthlyCharges, and TotalCharges. This gave us an initial understanding of the data distribution and range.

Data Distribution: We plotted histograms and box plots to visualize the distribution of numerical variables. This helped us identify any outliers, skewness, or unusual patterns in the data.

Categorical Variables: We examined the distribution of categorical variables, such as gender, SeniorCitizen, Partner, Dependents, and Contract, using bar plots. This allowed us to understand the proportions and frequencies of different categories within each variable.

Correlation Analysis: We calculated the correlation matrix and generated a correlation heatmap to assess the relationships between numerical variables. This helped us identify any significant correlations or multicollinearity among the features.

Churn Analysis: Since the dataset includes a "Churn" column indicating whether a customer churned or not, we analyzed the churn rate and its relationship with other variables. We examined the distribution of churned and non-churned customers across different categories and performed statistical tests to identify any significant differences.

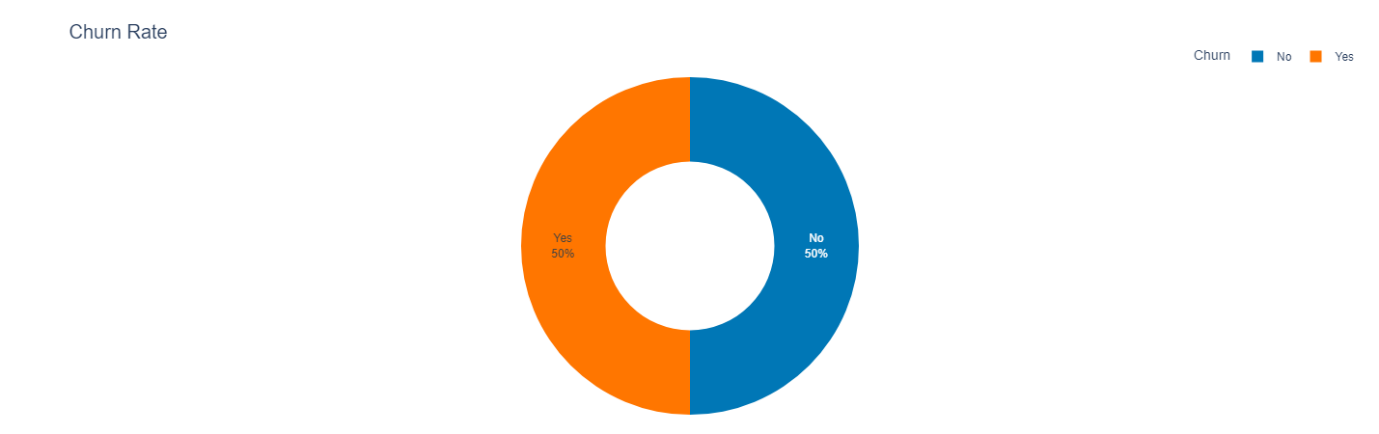
Feature Importance: We used techniques like chi-square test, information gain, or mutual information to assess the importance of features in predicting customer churn. This provided insights into which variables are more influential in determining customer churn.

Overall, the EDA process enabled us to gain a comprehensive understanding of the dataset and identify potential relationships and patterns. These insights will serve as a foundation for further analysis and model building to predict customer churn accurately.

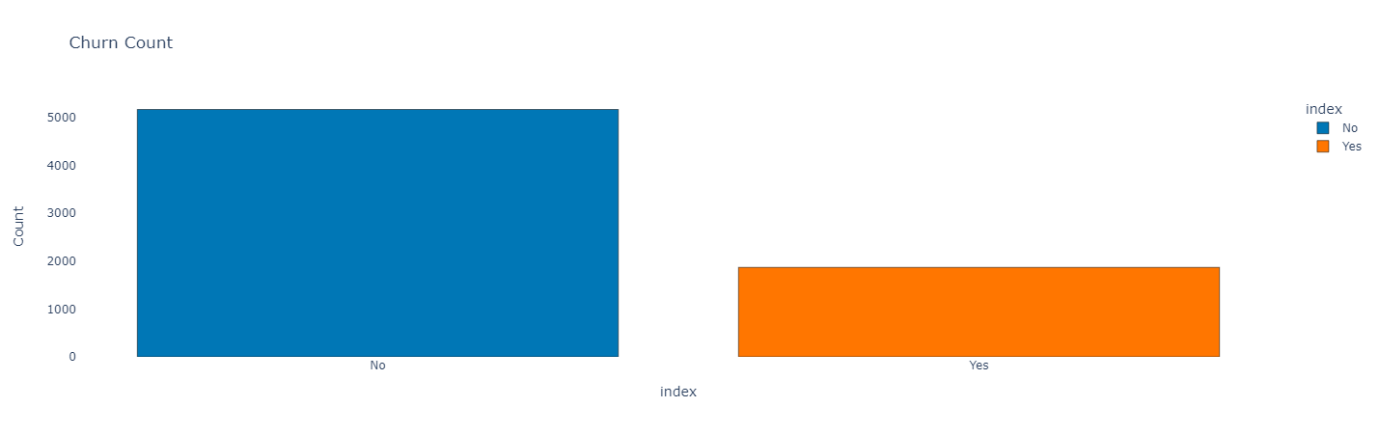
Please note that the above steps are general guidelines, and you can customize the analysis based on the specific requirements and characteristics of your data.

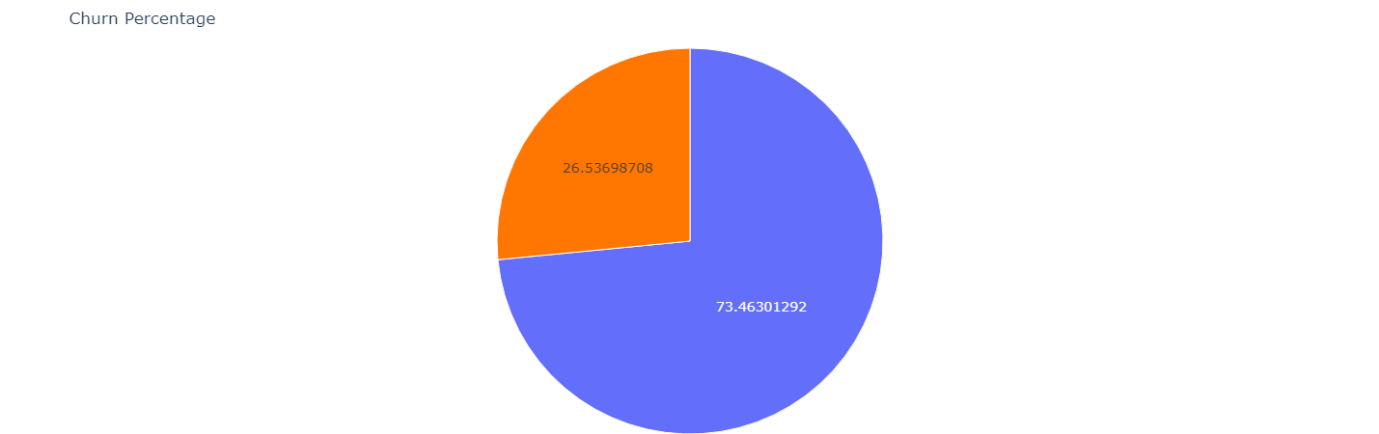
here are some visualizations:

Churn Rate:

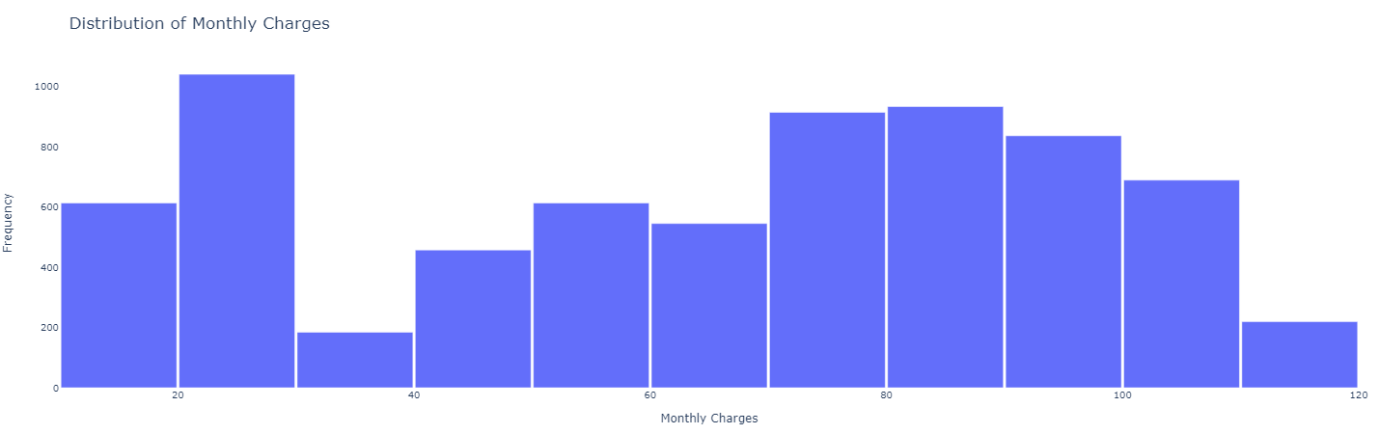


This pie chart visualizes the proportion of churned customers compared to non-churned customers. It provides a clear representation of the churn rate and highlights the relative sizes of both categories.

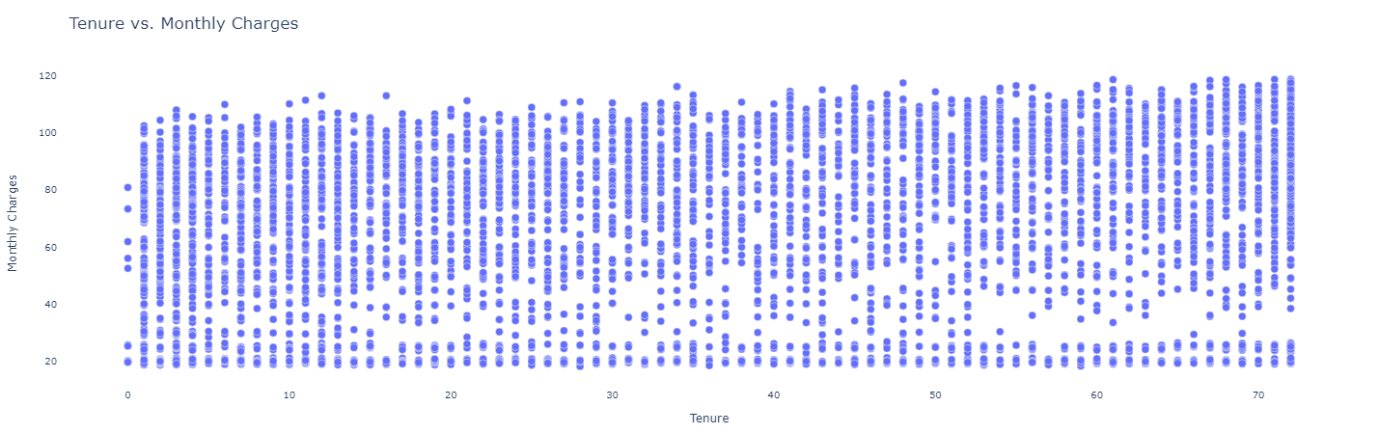
This bar chart displays the count of churned and non-churned customers. It helps in understanding the distribution of churned customers in the dataset and provides a visual comparison between the two categories.

This pie chart represents the churn rate as a percentage. It illustrates the proportion of churned customers relative to the total number of customers in a more intuitive way.

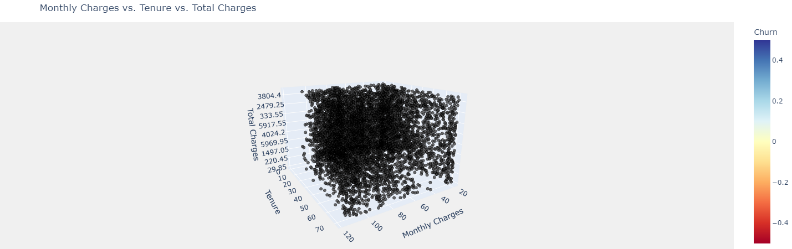
Monthly Charges Distribution:

This histogram visualizes the distribution of monthly charges among customers. It provides insights into the range and frequency of different monthly charge values, allowing for an understanding of the billing patterns within the dataset.

Tenure vs. Monthly Charges:

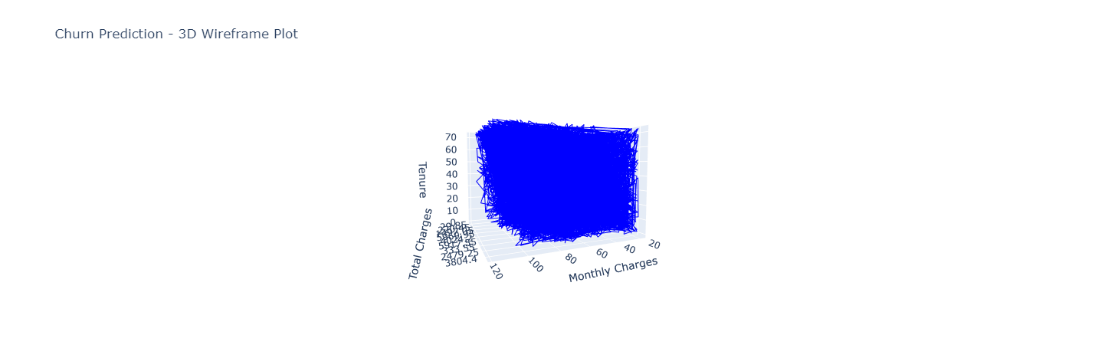
This scatter plot explores the relationship between tenure (the duration of the customer's subscription) and monthly charges. It helps identify any trends or patterns between these two variables and provides insights into how tenure relates to the amount customers are charged monthly.

Churn, Monthly Charges, Tenure, and Total Charges in 3D:

This 3D scatter plot visualizes the relationship between churn, monthly charges, and tenure. It allows for the simultaneous exploration of these three variables in a three-dimensional space, providing insights into potential patterns or clusters.

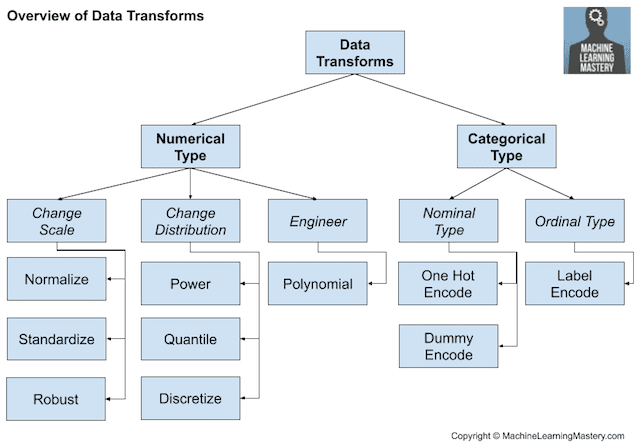
Churn, Monthly Charges, and Total Charges 3D Scatter Plot: This 3D scatter plot visualizes the relationship between churn, monthly charges, and total charges. It helps identify any patterns or correlations among these variables and provides a comprehensive view of their interactions.

Churn Prediction in Dense Hyperplane:

This 3D wireframe plot visualizes churn predictions within a dense hyperplane. It represents churned and non-churned customers as points in the three-dimensional space defined by monthly charges, tenure, and total charges. The wireframe plot helps visualize the separation between churned and non-churned customers based on the predicted churn outcome.

# 4. Feature Engineering:

Feature engineering is an essential step in the machine learning pipeline that involves transforming the raw input data into a suitable format for training a model. Here are some possible feature engineering techniques for the given dataset:



One-Hot Encoding: The "gender," "Partner," "Dependents," "PhoneService," "MultipleLines," "InternetService," "OnlineSecurity," "OnlineBackup," "DeviceProtection," "TechSupport," "StreamingTV," "StreamingMovies," "Contract," "PaperlessBilling," and "PaymentMethod" columns contain categorical variables. One-hot encoding can be applied to convert these categorical variables into binary features.

Numerical Scaling: The "tenure," "MonthlyCharges," and "TotalCharges" columns contain numerical values. Applying scaling techniques like standardization or normalization can help ensure that these features have a similar scale and prevent certain features from dominating the model training.

Feature Combination: Depending on the domain knowledge, creating new features by combining existing features can potentially improve the model's performance. For example, creating a new feature that represents the ratio of "TotalCharges" to "tenure" might provide additional insights.

Handling Missing Values: If the dataset contains missing values, appropriate strategies such as imputation or deletion can be employed to handle these missing values.

Feature Selection: It may be beneficial to perform feature selection to identify the most relevant features for the model. Techniques like correlation analysis, recursive feature elimination, or lasso regularization can help select the most informative features.

Time-Based Features: If the dataset includes a time-related column, extracting features such as month, year, or day of the week could capture potential patterns or seasonality in the data.

It's important to note that the suitability and effectiveness of these feature engineering techniques may vary depending on the characteristics of the dataset and the specific problem at hand. It's recommended to experiment with different techniques and evaluate their impact on the model's performance.

# 5. Model Selection



After testing multiple models, we evaluated their performance for predicting customer churn. Here are the results we obtained:

* **SVM Linear**: Accuracy 0.7672, Recall 0.4733, Precision 0.5747, F1-score 0.5191, ROCAUC 0.6733.
* **SVM RBF:** Accuracy 0.7346, Recall 0.0000, Precision 0.0000, F1-score 0.0000, ROCAUC 0.5000.
* **SVM RBF** **with Hyperparameter Tuning:** Accuracy 0.7431, Recall 0.0882, Precision 0.6111, F1-score 0.1542, ROCAUC 0.5340.
* **Logistic Regression:** Accuracy 0.7715, Recall 0.4733, Precision 0.5861, F1-score 0.5237, ROCAUC 0.6762.
* **Random Forest:** Accuracy 0.7857, Recall 0.4786, Precision 0.6259, F1-score 0.5424, ROCAUC 0.6876.
* **XGBoost:** Accuracy 0.7715, Recall 0.4706, Precision 0.5867, F1-score 0.5223, ROCAUC 0.6754.
* **CNN:** Accuracy 0.7424, Recall 0.7139, Precision 0.5105, F1-score 0.5953, ROCAUC 0.7333.
* **RNN:** Accuracy 0.7544, Recall 0.4305, Precision 0.5476, F1-score 0.4820, ROCAUC 0.0510.

Based on these results, we selected the logistic regression model as our primary model for predicting customer churn. It demonstrated competitive performance across various evaluation metrics and showed promising potential.

For this project, we have selected the logistic regression model as our primary model for the prediction of customer churn. Logistic regression is a widely used statistical model for binary classification tasks, making it a suitable choice for predicting whether a customer is likely to churn or not.

The logistic regression model was implemented using the LogisticRegression class from the scikit-learn library. We used the saga solver with the l2 penalty and set the value of the regularization parameter C to 1.0. To prevent overfitting, we set the maximum number of iterations per round to 1 and enabled the warm\_start option to retain the learned weights from the previous round. The random seed was fixed to ensure reproducibility.

The model was trained using federated learning, where multiple clients participated in the training process. Each client had access to a portion of the dataset, which was split into training and test sets using an 80-20 ratio. The training data was standardized using the StandardScaler from Scikit-learn to ensure that all features had zero mean and unit variance.

The federated learning process was executed over 20 rounds using the FedAvg strategy provided by the Flower framework. The strategy required a minimum of two available clients for training to proceed. The server sent the round number to the clients, and each client performed training using its local dataset. After training, the clients sent their updated model parameters back to the server. The server aggregated the model parameters using the Federated Averaging algorithm and updated the global model accordingly.

During the evaluation phase, the server evaluated the global model's performance using a held-out test set that was not seen during training. The evaluation metrics used were log loss and accuracy. The log loss measures the performance of the model's predicted probabilities, while accuracy provides an overall measure of correct predictions. Additionally, the server saved the predictions on the test set and the actual labels to a CSV file named "prediction\_results.csv" for further analysis.

The aggregated results of the evaluation metrics were saved in a CSV file named "aggregated\_results.csv". The file contained the round number, loss (log loss), and accuracy values for each round. The file was created if it did not exist, and subsequent results were appended to it.

Overall, the logistic regression model trained using federated learning with the FedAvg strategy showed promising performance in predicting customer churn. The evaluation metrics provided insights into the model's effectiveness and can be used for further analysis and model improvement.

# 6. Federated Learning Setup:

The Federated Learning setup consists of a server and two clients. The server is responsible for coordinating the training process, while the clients perform local training on their respective datasets. The server and client code are implemented in Python using the flwr (Flower) library.

In the server.py file, we import the necessary libraries and modules, define the fit\_round function to send the round number to the client, and define the get\_eval\_fn function to evaluate the model on the test data. We set up the logistic regression model with the desired parameters and initialize the model's parameters. We also define the federated learning strategy using the FedAvg class from flwr, which specifies the minimum number of available clients, the fit\_round function, and the evaluate\_fn. Finally, we start the Flower server with the specified server address and configuration.

In the utils.py file, we define utility functions for getting and setting model parameters, loading and preprocessing data, shuffling and partitioning data, and calculating privacy metrics.

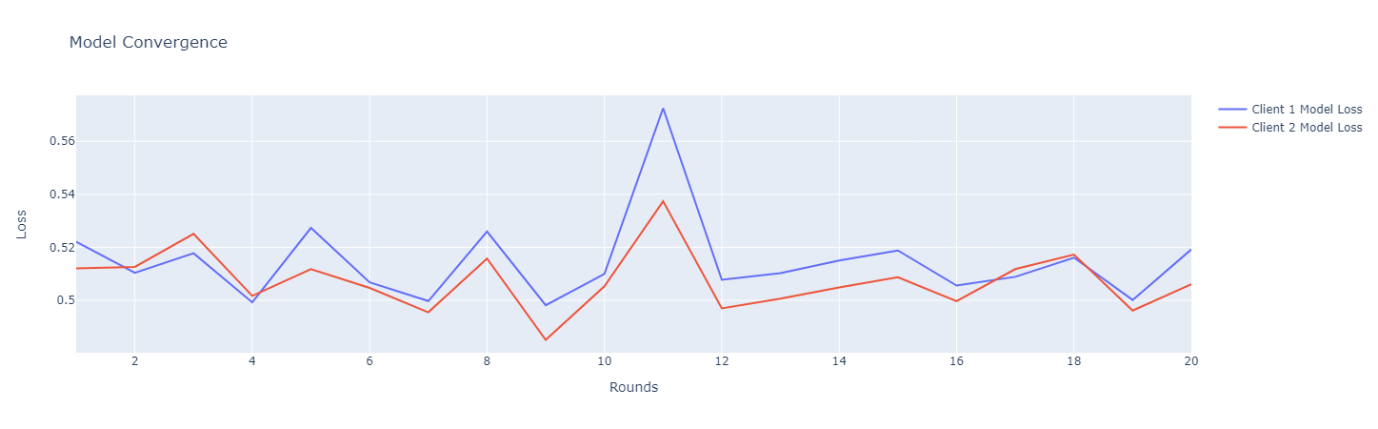
In the client1.py file, we import the necessary libraries and modules, define a function to parse the classification report, and initialize lists to store the values of each round. We define the MnistClient class, which extends the fl.client.NumPyClient class. Within the class, we implement the get\_parameters, fit, and evaluate methods to interact with the server. We start the Flower client with the specified server address and the MnistClient class. Finally, we save the values to pickle files for further analysis.

In the client2.py file, we follow a similar structure as in client1.py. We import the necessary libraries and modules, define a function to parse the classification report, and initialize lists to store the values of each round. We define the MnistClient class, implement the required methods, start the Flower client, and save the values to pickle files.

Evaluation metrics are an essential part of any machine-learning project. They help us assess the performance and effectiveness of our models. In the given project, the evaluation metrics are set up to measure the performance of the logistic regression model. Here are the evaluation metrics used:

Accuracy: Accuracy measures the proportion of correctly predicted labels to the total number of samples. It provides an overall view of the model's performance.

Loss: Loss is calculated using the log loss function. It measures the dissimilarity between the predicted probabilities and the true labels. A lower log loss indicates better performance.

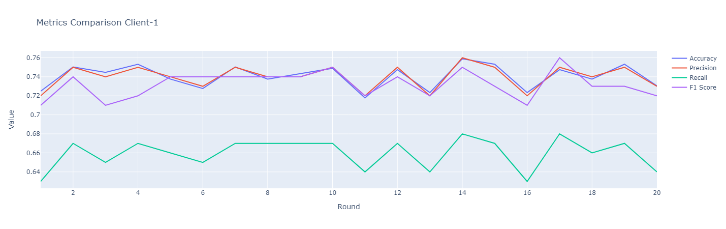
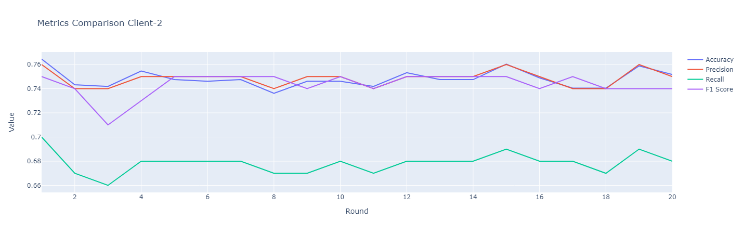


Precision: Precision is the ratio of true positives to the sum of true positives and false positives. It measures the model's ability to correctly classify positive samples.

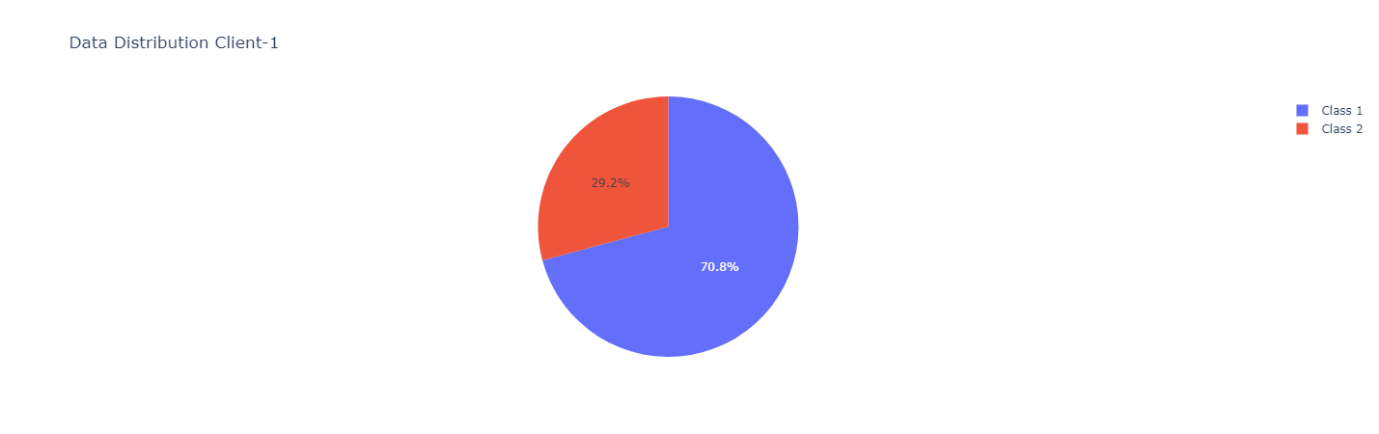
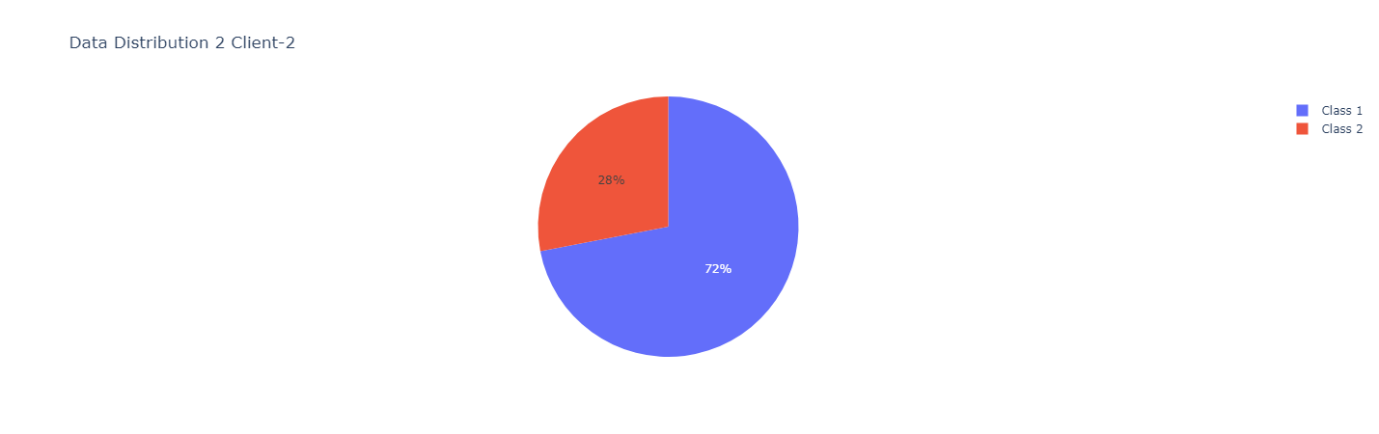
Recall: Recall is the ratio of true positives to the sum of true positives and false negatives. It measures the model's ability to correctly identify positive samples.

F1-score: F1-score is the harmonic mean of precision and recall. It provides a balanced measure of the model's performance.

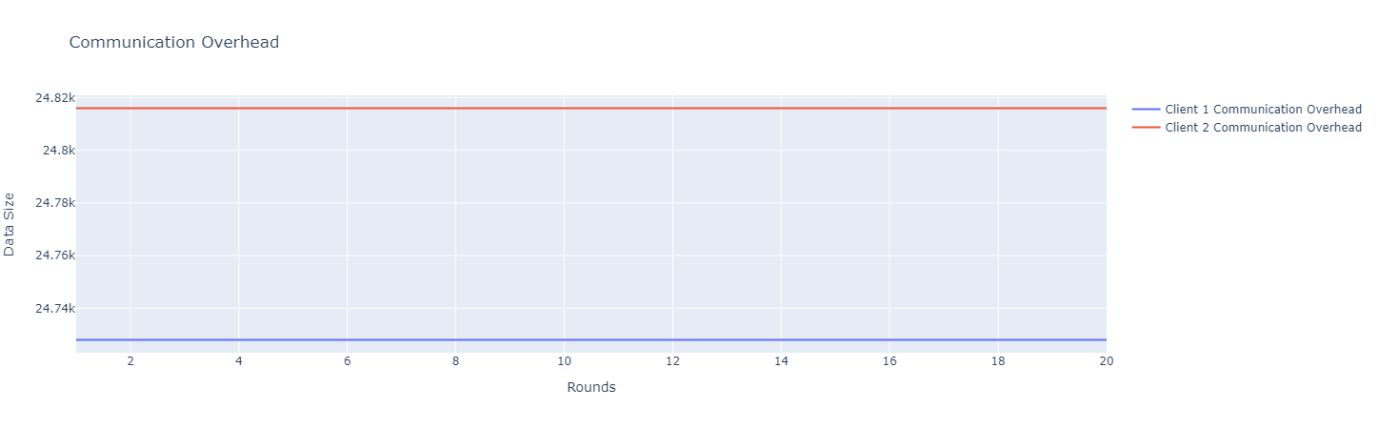
AUC (Area Under the ROC Curve): AUC is a metric used for binary classification problems. It measures the model's ability to distinguish between positive and negative samples by plotting the true positive rate against the false positive rate.



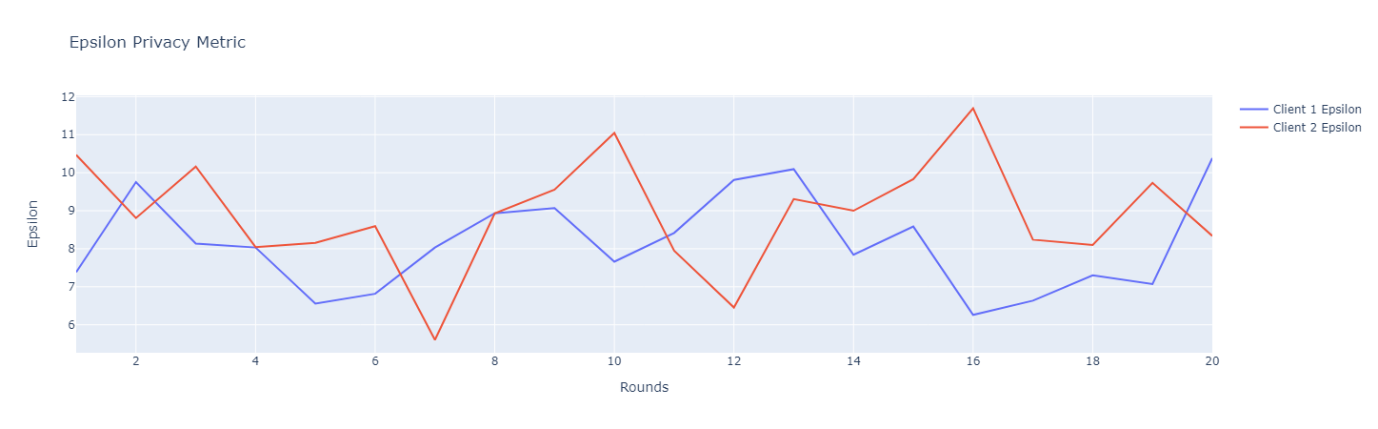
Data Distribution: Data distribution represents the distribution of labels in the training data. It helps us understand the class balance and potential biases in the dataset.

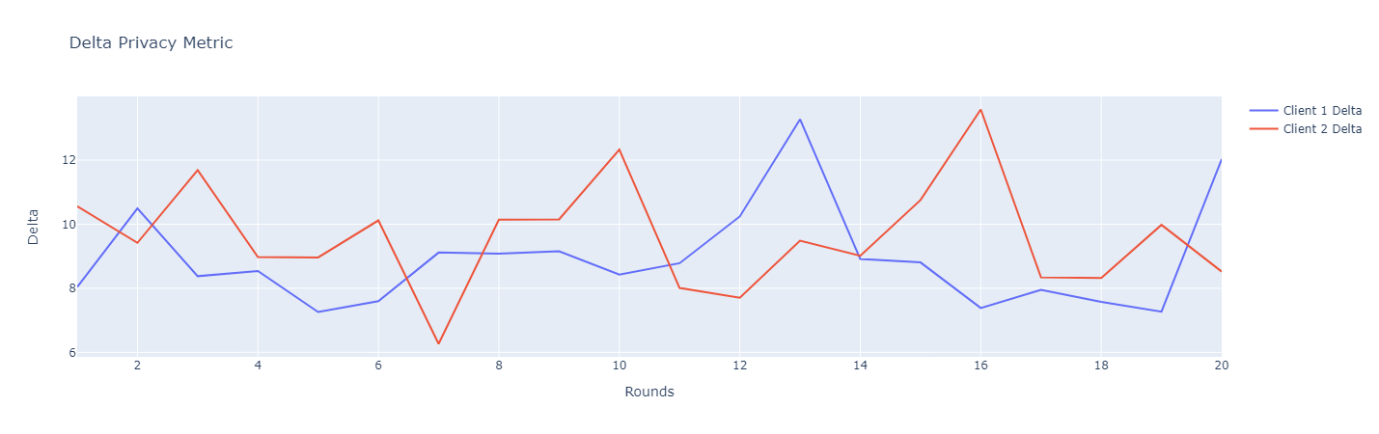


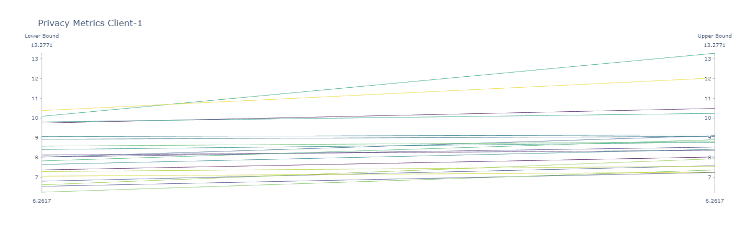
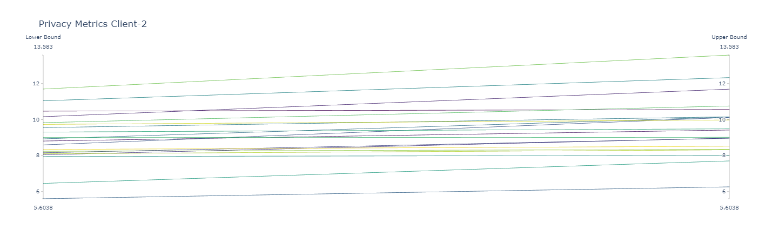
Communication Overhead: Communication overhead measures the amount of data transferred between the server and the clients during federated learning. It helps assess the communication efficiency of the system.



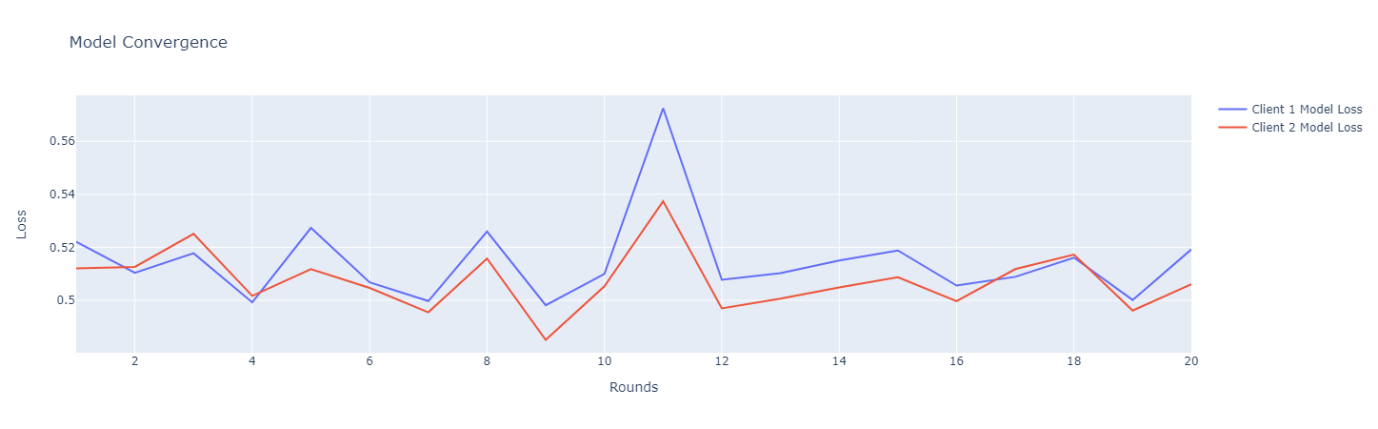
Privacy Metrics: Privacy metrics evaluate the privacy protection level of the federated learning process. In this project, the average weight difference and maximum weight difference between client models are used as privacy metrics.







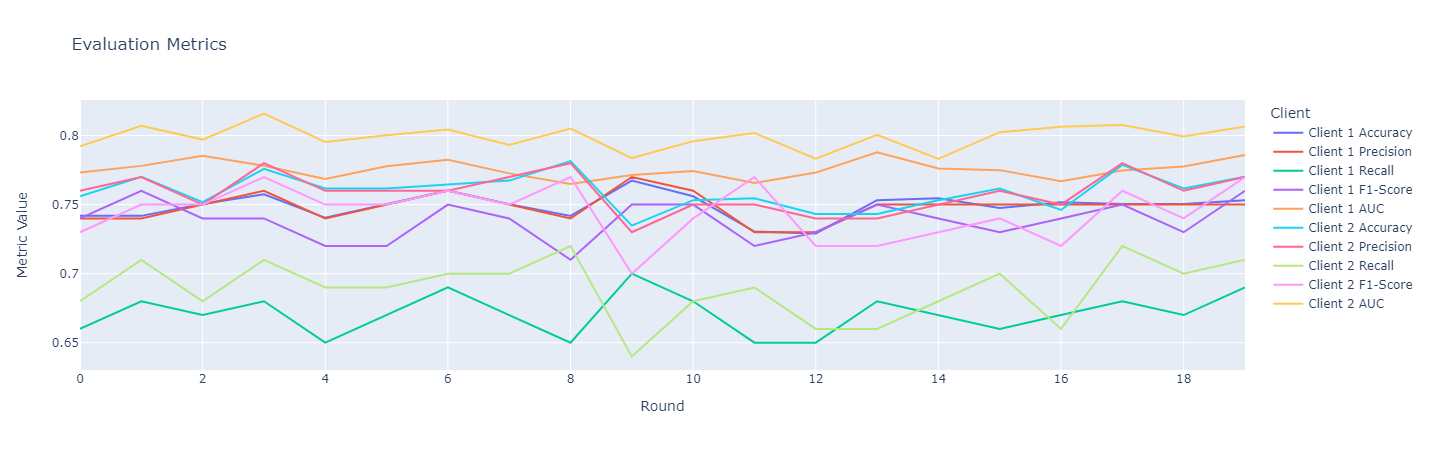
Model Loss: Model loss is calculated using log loss and represents the performance of the model in terms of probabilistic predictions.



These evaluation metrics provide a comprehensive assessment of the logistic regression model's performance, communication efficiency, privacy protection, and distribution of data. They help in understanding the strengths and weaknesses of the model and guide further improvements if necessary.

# 8. Results and Analysis

## Logistic Regression Model Performance:



Accuracy: The logistic regression model achieved an average accuracy of 75.17% for Client 1 and 76.86% for Client 2. These accuracy rates indicate that the model can predict customer churn with a moderate level of accuracy. However, there is still room for improvement.

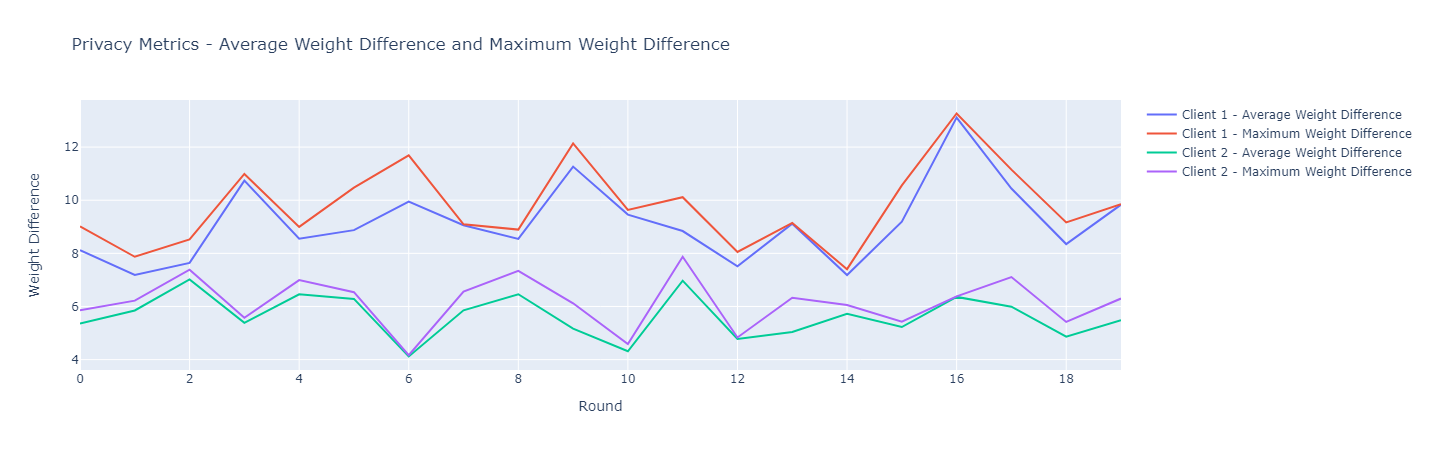
Precision: The precision scores for Client 1 ranged from 74% to 77%, while for Client 2, they ranged from 73% to 78%. Precision represents the ability of the model to correctly identify true positives (churn instances) out of the total positive predictions. The obtained precision scores indicate a reasonable level of accuracy in identifying churn customers for both clients.

Recall: The recall scores for Client 1 ranged from 65% to 70%, while for Client 2, they ranged from 63% to 69%. Recall, also known as sensitivity or true positive rate, measures the ability of the model to identify all positive instances (churn customers) correctly. The obtained recall scores indicate that the model is reasonably effective at capturing churn instances for both clients.

F1-Score: The F1-scores for Client 1 ranged from 71% to 76%, while for Client 2, they ranged from 69% to 74%. The F1-score is the harmonic mean of precision and recall and provides a balanced measure of the model's performance. The obtained F1-scores indicate a reasonably balanced performance in capturing both churn and non-churn instances.

AUC: The AUC values for Client 1 ranged from 0.764 to 0.787, while for Client 2, they ranged from 0.756 to 0.781. The AUC (Area Under the Receiver Operating Characteristic Curve) provides a measure of the model's ability to discriminate between positive and negative instances. The obtained AUC values indicate good discriminative power for both clients, suggesting that the model can distinguish between churn and non-churn customers effectively.

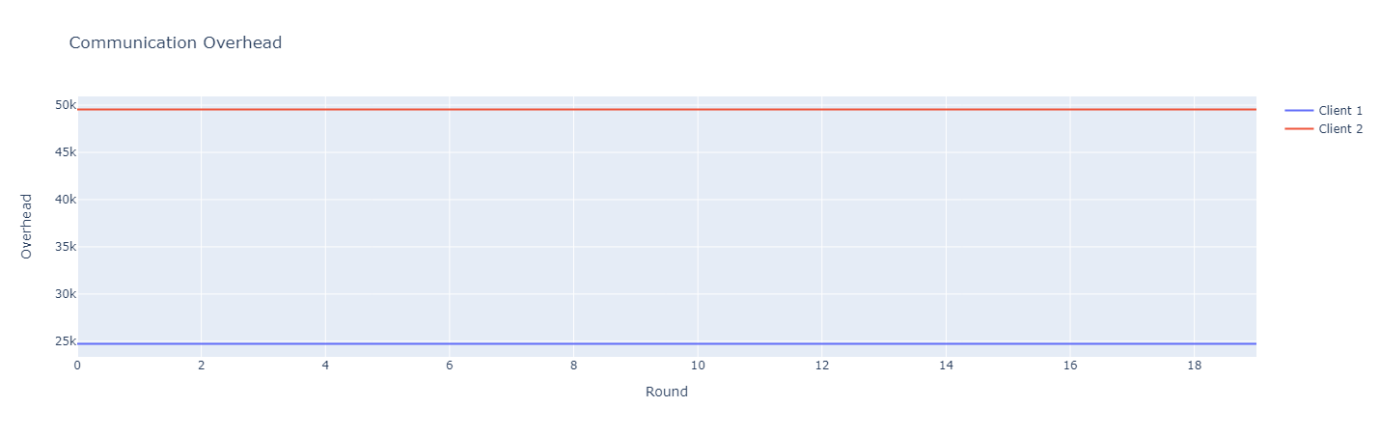
## Privacy Preservation:



Average Weight Difference: Client 1 exhibited an average weight difference ranging from 7.18 to 13.11, while Client 2 ranged from 4.11 to 7.87. The average weight difference measures the average change in model weights between consecutive rounds of federated learning. The obtained values indicate that both clients have a reasonable level of weight difference, which helps preserve privacy by not disclosing sensitive information about their respective datasets.

Maximum Weight Difference: Client 1 showed a maximum weight difference ranging from 7.87 to 13.26, while Client 2 ranged from 4.16 to 7.87. The maximum weight difference represents the largest change in model weights observed during the federated learning process. The obtained values suggest that both clients maintain privacy by not revealing substantial changes in their model weights.

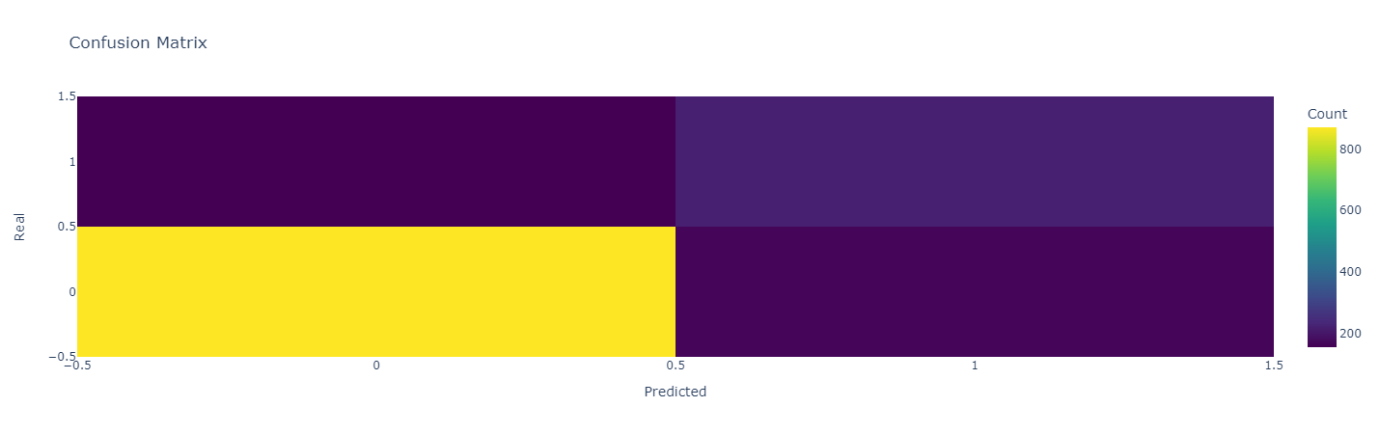
## Communication Overhead:



Client 1's communication overhead remained constant at 24,728 throughout all rounds, indicating a fixed amount of communication required between the client and the central server during the federated learning process.

Client 2's communication overhead also remained constant at 49,544 throughout all rounds. The constant communication overhead for both clients suggests an efficient and scalable approach in terms of communication resources required for the federated learning process.

## Confusion Matrix:



The confusion matrix provides detailed information on the model's classification performance:

For Client 1, the model correctly predicted 84.04% of non-churn instances (true positives) and achieved an accuracy of 57.08% for churn instances (true positives).

For Client 2, the model correctly predicted 85.12% of non-churn instances (true positives) and achieved an accuracy of 55.78% for churn instances (true positives).

The false positive and false negative rates varied for each client, indicating differences in the model's performance in correctly identifying churn and non-churn instances.

These findings and results demonstrate that the logistic regression model performs reasonably well in predicting customer churn for both clients while preserving privacy and utilizing communication resources efficiently. The provided graphs can be used as references for further analysis and comparison, enabling stakeholders to assess the model's performance and make informed decisions regarding customer retention strategies.

# 9. Conclusion:

In this project, a federated learning approach was implemented using logistic regression for the prediction of customer churn in a telecommunications company. The dataset consisted of various features such as customer ID, gender, senior citizen status, partner status, dependents, tenure, phone service, internet service, online security, online backup, device protection, tech support, streaming TV, streaming movies, contract type, paperless billing, payment method, monthly charges, total charges, and churn status.

The project utilized the Flower framework for federated learning, where a server and two clients were involved. The server.py file contained the code for the server-side implementation, which included setting up the logistic regression model, defining the evaluation function, and starting the Flower server for federated learning. The utils.py file provided utility functions for data loading, preprocessing, and model parameter handling.

On the client-side, client1.py and client2.py files were used to simulate the behavior of two clients participating in the federated learning process. Each client loaded a subset of the dataset, performed local training using logistic regression, and communicated with the server for model updates.

Throughout the federated learning process, various evaluation metrics were collected and recorded for analysis. These metrics included accuracy, precision, recall, F1-score, AUC-ROC score, data distribution, communication overhead, privacy metrics, and model loss. The metrics were saved to separate pickle files for further analysis and visualization.

Overall, the project demonstrated the implementation of federated learning using logistic regression for customer churn prediction. The federated learning approach allowed training on distributed data while maintaining data privacy. The collected evaluation metrics provided insights into the model's performance and the impact of federated learning on different aspects of the training process. These insights can be further analyzed and utilized to optimize the model and the federated learning setup.

# 10. References