Predicting Customer Churn Using Logistic Regression

Maahika Mathur

Probability for Gen-AI by Dr.Pai and Dr.Shenoy

This article explores how logistic regression predicts customer churn in subscription-based businesses. We cover the mathematical foundations, model training using Python, and practical insights for retention strategies using real telecom data.

When one first dives into data science, the journey often begins with linear regression. It's a great tool for predicting a number, like a house price, by drawing a straight line through data. But what happens when the question isn't "how much," but a simple "yes or no"? That is the challenge with customer churn. A customer either stays or they leave. While one could label "leave" as 1 and "stay" as 0, a straight line model breaks down, predicting nonsensical values like 0.2 or 1.5. A customer can't be 150% likely to leave.

This is where logistic regression comes in. Instead of forcing a straight line, it uses a flexible S-shaped curve to calculate a probability a value that is always between 0 and 1. This output is far more useful because it shows how likely something is to happen. For a business, this transforms a simple prediction into strategy. Knowing a customer has a 78% chance of leaving creates a critical window of opportunity to step in and change the outcome.

In practice, this probability-based approach powers many retention systems across industries. Telecom firms, streaming platforms, and banks all deploy models like logistic regression to flag customers at risk before they leave. For instance, Verizon uses generative AI to predict why users call and match them to agents, with the goal of saving 100,000 customers annually. [6] Even modest improvements in retention yield large gains: research shows increasing customer retention by 5% can boost profits by 25% to 95%. [7] Models like these aren't flashy they quietly shift businesses from reactive to proactive.

1 What Is Logistic Regression?

Logistic regression models the probability of a binary event for example, whether a customer churns (1) or stays (0) by expressing the *log odds* of that event as a linear function of input features. The model then applies a sigmoid transformation to convert those log-odds into interpretable probabilities between 0 and 1.

For input features x_0 (tenure) and x_1 (monthly charges), the model predicts the probability of churn as:

$$P(Y = 1 \mid x_0, x_1) = \sigma(w_0 x_0 + w_1 x_1 + b) \tag{1}$$

where w_0 and w_1 represent the weights for each feature, b is the bias term, and σ is the **sigmoid function**:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2}$$

This function converts any real number into a value between 0 and 1, interpretable as a probability. If $\sigma(z) > 0.5$, the event (churn) is predicted to occur.

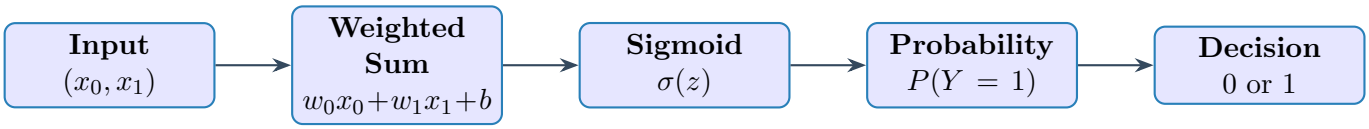


Figure 1: *The flow of computation in a logistic regression classifier.*

2 How the Model Learns

The goal of logistic regression is to find the weights w_0, w_1 and bias b that make the model's predictions match observed data as closely as possible.

2.1 Connection to Bernoulli Distribution

The churn outcome Y for each customer follows a **Bernoulli distribution** with parameter p, where p is the probability of churn. For a Bernoulli random variable:

$$P(Y = y) = p^y (1 - p)^{(1-y)}, \quad y \in \{0, 1\} \tag{3}$$

In logistic regression, this probability depends on customer features. For features (x_0, x_1) (tenure and monthly charges), we model:

$$p_i = P(Y_i = 1 \mid x_{i0}, x_{i1}) = \sigma(w_0 x_{i0} + w_1 x_{i1} + b) \tag{4}$$

where σ is the sigmoid function. Each customer's churn status is a Bernoulli trial with its own probability p_i determined by their features.

2.2 Maximum Likelihood Estimation

Given data points (x_{i0}, x_{i1}, y_i) where $y_i \in \{0, 1\}$, the **likelihood** of observing all the data is the product of individual Bernoulli probabilities:

$$L(w_0, w_1, b) = \prod_{i=1}^{n} [p_i]^{y_i} [1 - p_i]^{(1-y_i)} = \prod_{i=1}^{n} [\hat{p}(x_{i0}, x_{i1})]^{y_i} [1 - \hat{p}(x_{i0}, x_{i1})]^{(1-y_i)} \tag{5}$$

Taking the log gives the **log-likelihood**:

$$\log L(w_0, w_1, b) = \sum_{i=1}^{n} [y_i \log \hat{p}(x_{i0}, x_{i1}) + (1 - y_i) \log(1 - \hat{p}(x_{i0}, x_{i1}))] \tag{6}$$

Machine learning implementations typically minimize the **negative log-likelihood**, also called the **binary cross-entropy loss**:

$$\text{Loss} = -\frac{1}{n} \sum_{i=1}^{n} [y_i \log \hat{p}(x_{i0}, x_{i1}) + (1 - y_i) \log(1 - \hat{p}(x_{i0}, x_{i1}))] \tag{7}$$

This ensures wrong confident predictions are penalized more. The Bernoulli framework provides the probabilistic foundation for why this loss function is the natural choice for binary classification.

Figure 2: *Model learning overview: from data to optimized parameters.*

This curve illustrates how the maximum likelihood estimate (MLE) identifies the value of p that makes the observed data most probable. The peak represents the optimal parameter value that best explains the pattern of churned and retained customers in our dataset. By finding this maximum, the model learns to predict churn with the highest fidelity to the training data.

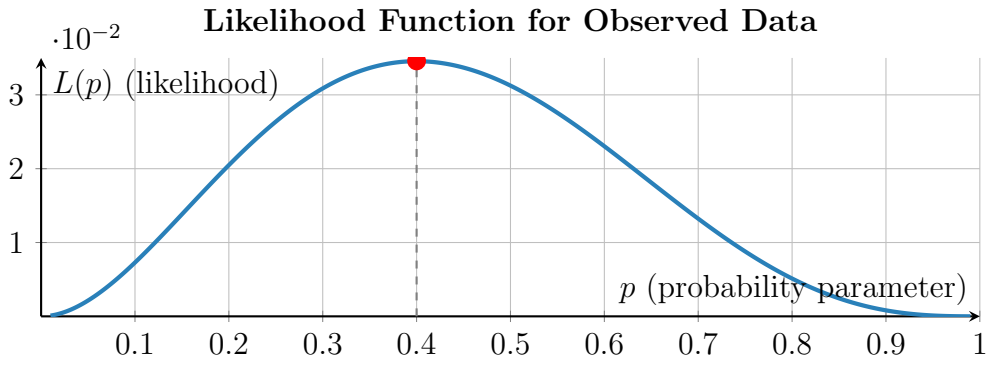


Figure 3: *Eg:Likelihood function $L(p) = p^2(1-p)^3$ for observed churn data, peaking at the maximum likelihood estimate $\hat{p} = 0.4$.*

3 The Sigmoid Function

The S-shaped sigmoid curve maps input scores to probabilities. The classic S-shaped sigmoid curve transforms any real-valued input into a probability between 0 and 1. The visualization clearly shows how extreme negative values approach 0, extreme positive values approach 1, and the inflection point occurs at $z = 0$ where the probability equals 0.5 (our decision threshold).

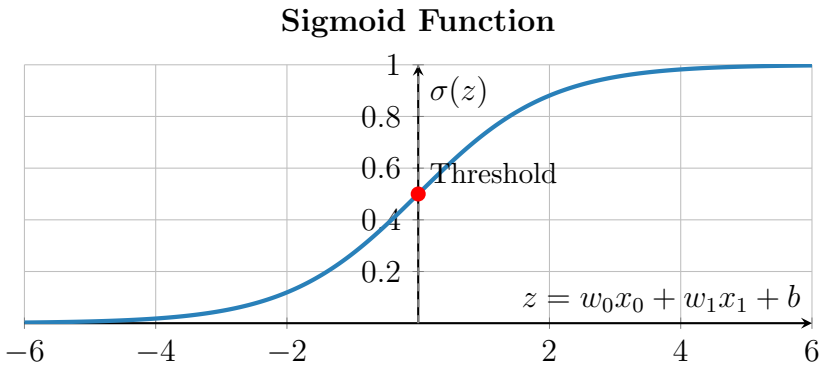


Figure 4: *Sigmoid function showing probability scaling between 0 and 1.*

4 The Churn Dataset

The **Telco Customer Churn** dataset from Kaggle contains features such as tenure, monthly charges, and churn status.

Feature	Description
`tenure`	Number of months with the company
`MonthlyCharges`	Bill amount per month
`Churn`	1 = Yes, 0 = No

```
import pandas as pd

# Load the churn dataset from CSV file
df = pd.read_csv("WA_Fn-UseC_-Telco-Customer-Churn.csv")
# Convert categorical churn labels to binary (1 = churned, 0 = retained)
df['Churn'] = df['Churn'].map({'Yes': 1, 'No': 0})
# Remove rows with missing TotalCharges (represented as spaces)
df = df[df['TotalCharges'] != ' ']
# Convert TotalCharges from string to numeric type
df['TotalCharges'] = df['TotalCharges'].astype(float)

# Select only the features we'll use for modeling
features = ['tenure', 'MonthlyCharges']
df = df[features + ['Churn']]
df.head()  # Display first 5 rows
```

This preprocessing step cleans the dataset by converting text labels to numerical values and removing incomplete records. We focus on two key features: tenure and monthly charges, which are intuitive predictors of customer behavior.

5 Visualizing the Data

```
# Extract feature matrix (X) and target labels (y)
X = df[['tenure', 'MonthlyCharges']].values
y = df['Churn'].values

# Create boolean masks for churned and retained customers
churned = y == 1
retained = y == 0

# Create scatter plot with different colors for each class
plt.figure(figsize=(8,6))
plt.scatter(X[retained,0], X[retained,1], c='green',
            label='Stayed', alpha=0.5)  # Green for retained customers
plt.scatter(X[churned,0], X[churned,1], c='red',
            label='Churned', alpha=0.5)  # Red for churned customers
plt.xlabel("Tenure (months)")
plt.ylabel("Monthly Charges ($)")
plt.title("Customer Churn Data")
```

```
plt.legend()
plt.grid(True)
plt.show()
```

This visualization reveals patterns in customer behavior: churned customers (red) tend to cluster in regions with shorter tenure and higher monthly charges, while retained customers (green) spread across longer tenures and more moderate pricing. This spatial separation suggests these features carry predictive power for churn detection.

Figure 5: *Scatter plot showing customer distribution by tenure and monthly charges, color-coded by churn status (green = retained, red = churned).*

6 Training the Model

```
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression

# Standardize features to have mean=0 and std=1 for better convergence
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
# Initialize and train the logistic regression classifier
clf = LogisticRegression()
clf.fit(X_scaled, y)

# Display learned parameters
print("Intercept (b):", clf.intercept_)
print("Coefficients (w):", clf.coef_)
```

Feature scaling ensures that both tenure and monthly charges contribute equally during training, preventing features with larger numeric ranges from dominating the learning process. The model then iteratively adjusts weights and bias to minimize the cross-entropy loss, converging to optimal parameters.

```
Intercept (b): [-1.44]
Coefficients (w): [[-1.35, 0.99]]
```

At this point, the logistic regression model has learned the optimal parameters that best separate churned and retained customers in feature space. The negative weight for tenure indicates that loyal, long-term customers are less likely to leave, while the positive weight for monthly charges suggests that high-paying customers face greater churn risk (potentially due to price sensitivity or perceived value mismatches).

7 Predictions

```
# Extract scaling parameters and model coefficients
X_mean = scaler.mean_
X_std = scaler.scale_
w0, w1 = clf.coef_[0]
b = clf.intercept_[0]

def predict_churn(x0, x1):
    # Manually scale input features using saved scaler parameters
    x0_scaled = (x0 - X_mean[0]) / X_std[0]
    x1_scaled = (x1 - X_mean[1]) / X_std[1]
    # Compute logit score (linear combination)
    score = w0 * x0_scaled + w1 * x1_scaled + b
    # Apply sigmoid to convert score to probability
```

```
    prob = 1 / (1 + np.exp(-score))
    # Apply threshold (0.5) to make binary decision
    return score, prob, int(prob > 0.5)

# Test prediction: tenure=50 months, monthly charges=£3
score, p, cls = predict_churn(50, 3)
print(f"Predicted probability of churn: {p:.3f}")
print("Predicted class:", "Churn" if cls == 1 else "Retain")
```

This function demonstrates the complete prediction pipeline: scaling inputs, computing the weighted sum, applying the sigmoid transformation, and making a binary decision. The model can now evaluate any customer profile and output both a probability and a categorical prediction.

```
Predicted probability of churn: 0.012
Predicted class: Retain
```

The model predicts a very low probability of churn (1.2%) for a customer with 50 months of tenure and only $3 in monthly charges. This aligns with business intuition: long-term customers paying minimal fees represent the most stable segment of the customer base, exhibiting strong retention characteristics.

The decision boundary visualization (Figure 6) shows how logistic regression creates a linear separation in the feature space. Customers falling on one side of the boundary are predicted to churn, while those on the other side are expected to remain. This geometric interpretation helps businesses identify which customer segments require targeted retention efforts.

8 Final Notes

Logistic regression gives us a clean, interpretable way to turn uncertainty into probability. In this exercise, we used a straight line to separate churned and retained customers which is an elegant start, but real data is rarely that well-behaved. When patterns curve, the boundary can too. Polynomial logistic regression, implemented easily with `PolynomialFeatures` in `scikit-learn`, can extend this framework to capture non-linear relationships. That's the natural next step if one wants to trade a bit of simplicity for a more flexible fit.

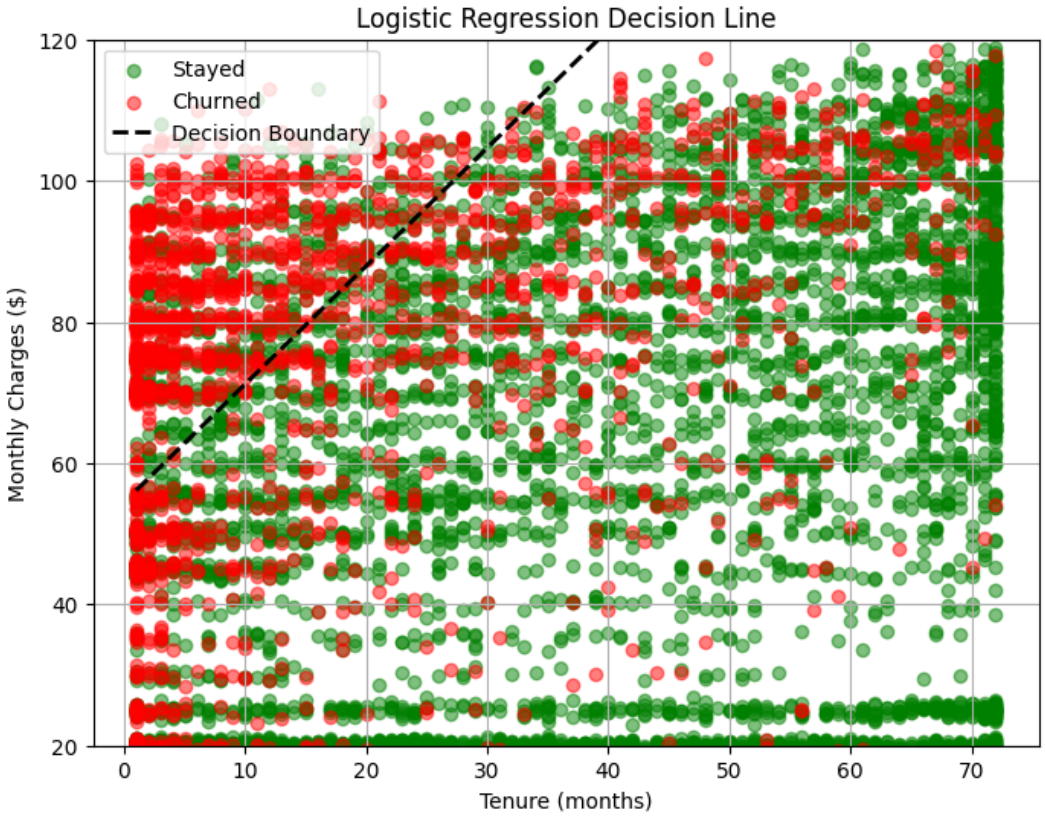
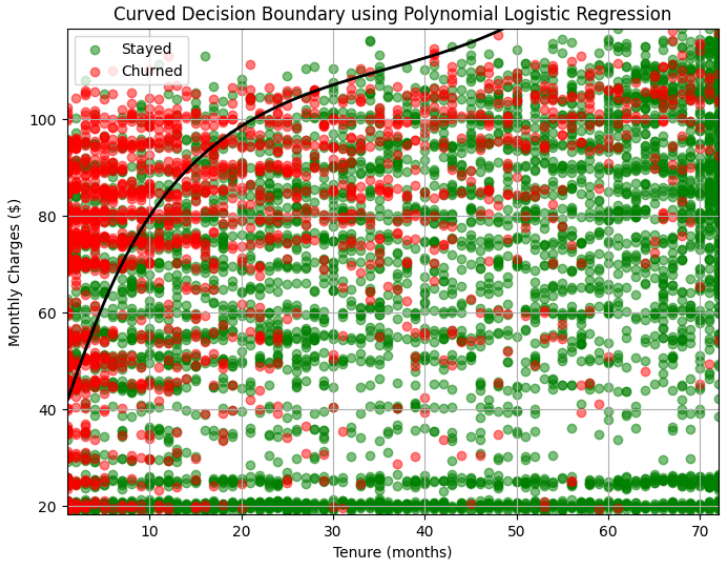


Figure 6: *The decision boundary represents where the predicted probability equals 0.5.*



A curved decision boundary generated using polynomial logistic regression

References

[1] Pai, S., & Shenoy, A. (2025). *Probability for Gen-AI course*

[2] Google Colab. (2025). *Logistic regression notebook* https://colab.research.google.com/drive/13TODKJgw1BbUcEZ7Dy0YOTw4VfkH9Yjr?usp=sharing

[3] Kaggle. *Telco customer churn dataset*. Retrieved from https://www.kaggle.com/datasets/blastchar/telco-customer-churn?resource=download

[4] GeeksforGeeks. *Logistic regression in machine learning*. Retrieved from https://www.geeksforgeeks.org/machine-learning/understanding-logistic-regression/

[5] Manďák, J., & Hančlová, J. (2019). Use of logistic regression for understanding and prediction of customer churn in telecommunications. *Statistika: Statistics and Economy Journal*, Czech Statistical Office.

[6] Verizon. (2024, June 18). Verizon uses GenAI to improve customer loyalty. *Reuters*. Retrieved from https://www.reuters.com/technology/artificial-intelligence/verizon-uses-genai-improve-customer-loyalty-2024-06-18/

[7] Gallo, A. (2014, October 29). The value of keeping the right customers. *Harvard Business Review*. Retrieved from https://hbr.org/2014/10/the-value-of-keeping-the-right-customers