**Sample Code :**

# Custmer Churn McCullOch Pits Neuron Model

import pandas as pd

import numpy as np

from sklearn.preprocessing import LabelEncoder

import csv

# Load dataset

data = pd.read\_csv(r"C:\Users\DELL\Desktop\ZZZZZZZZZZZZZZZZZZ\CUSTOMERCHURN.csv")

# Column Names of CSV file

# Print all column names

print("Column Names in CSV File:")

print(list(data.columns))

# Select only binary columns for simplicity

binary\_columns = ['SeniorCitizen', 'Partner', 'Dependents', 'PhoneService', 'PaperlessBilling']

data = data[binary\_columns + ['Churn']].dropna()

# Convert Yes/No and boolean-like features to 0/1

for col in binary\_columns:

data[col] = data[col].apply(lambda x: 1 if x in ['Yes', 1, 'Male'] else 0)

# Encode target

data['Churn'] = data['Churn'].apply(lambda x: 1 if x == 'Yes' else 0)

# Inputs and outputs

X = data[binary\_columns].values

y\_true = data['Churn'].values

# --- McCulloch-Pitts Neuron Implementation ---

def mp\_neuron(inputs, weights, threshold):

summation = np.dot(inputs, weights)

return 1 if summation >= threshold else 0

# Hardcoded weights and threshold

# You can tweak these manually for best accuracy

weights = [1, 1, -1, 1, 1] # Example weights (length = number of input features)

threshold = 3 # Example threshold

# Predict using MP Neuron

predictions = [mp\_neuron(x, weights, threshold) for x in X]

# Evaluate accuracy

accuracy = np.mean(predictions == y\_true)

print(f"Accuracy of McCulloch-Pitts Neuron on Customer Churn Data: {accuracy \* 100:.2f}%")

# Show confusion matrix

from sklearn.metrics import confusion\_matrix, classification\_report

print("Confusion Matrix:")

print(confusion\_matrix(y\_true, predictions))

print("\nClassification Report:")

print(classification\_report(y\_true, predictions))

**Output :**

Column Names in CSV File:

['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents', 'tenure', 'PhoneService', 'MultipleLines', 'InternetService', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling', 'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn']

Accuracy of McCulloch-Pitts Neuron on Customer Churn Data: 67.03%

Confusion Matrix:

[[4157 1017]

[1305 564]]

Classification Report:

precision recall f1-score support

0 0.76 0.80 0.78 5174

1 0.36 0.30 0.33 1869

accuracy 0.67 7043

macro avg 0.56 0.55 0.55 7043

weighted avg 0.65 0.67 0.66 7043

Sample Predictions:

Input: [0 1 0 0 1], Predicted: No, Actual: No

Input: [0 0 0 1 0], Predicted: No, Actual: No

Input: [0 0 0 1 1], Predicted: No, Actual: Yes

Input: [0 0 0 0 0], Predicted: No, Actual: No

Input: [0 0 0 1 1], Predicted: No, Actual: Yes

**MODIFIED CODE :**

**import pandas as pd**

**import numpy as np**

**from sklearn.metrics import confusion\_matrix, classification\_report**

**# Load dataset**

**data = pd.read\_csv(r"C:\Users\DELL\Desktop\ZZZZZZZZZZZZZZZZZZ\CUSTOMERCHURN.csv")**

**# Print all column names**

**print("Column Names in CSV File:")**

**print(list(data.columns))**

**# Automatically detect binary columns (having exactly 2 unique non-null values)**

**binary\_columns = [col for col in data.columns if data[col].nunique(dropna=True) == 2 and col != 'Churn']**

**# Print detected binary columns**

**print("\nAutomatically Detected Binary Columns:")**

**print(binary\_columns)**

**# Ensure 'Churn' is included for classification**

**if 'Churn' not in data.columns:**

**raise ValueError("'Churn' column not found in dataset!")**

**# Select binary columns + Churn, and drop missing values**

**data = data[binary\_columns + ['Churn']].dropna()**

**# Convert binary categorical columns to 0/1**

**for col in binary\_columns:**

**unique\_vals = sorted(data[col].unique())**

**data[col] = data[col].apply(lambda x: 1 if x == unique\_vals[-1] else 0)**

**# Encode 'Churn' target to 0/1**

**data['Churn'] = data['Churn'].apply(lambda x: 1 if x == 'Yes' else 0)**

**# Inputs and outputs**

**X = data[binary\_columns].values**

**y\_true = data['Churn'].values**

**# --- McCulloch-Pitts Neuron Implementation ---**

**def mp\_neuron(inputs, weights, threshold):**

**summation = np.dot(inputs, weights)**

**return 1 if summation >= threshold else 0**

**# Hardcoded weights and threshold (can be tuned manually)**

**weights = [1] \* len(binary\_columns) # Default: weight 1 for each feature**

**threshold = int(np.ceil(len(binary\_columns) / 1)) # Simple threshold: half of total inputs**

**# Predict using MP Neuron**

**predictions = [mp\_neuron(x, weights, threshold) for x in X]**

**# Evaluate accuracy**

**accuracy = np.mean(predictions == y\_true)**

**print(f"\nAccuracy of McCulloch-Pitts Neuron on Customer Churn Data: {accuracy \* 100:.2f}%")**

**# Show confusion matrix and classification report**

**print("\nConfusion Matrix:")**

**print(confusion\_matrix(y\_true, predictions))**

**print("\nClassification Report:")**

**print(classification\_report(y\_true, predictions))**

**Output :**

**Accuracy of McCulloch-Pitts Neuron on Customer Churn Data: 73.24%**

**Confusion Matrix:**

**[[5149 25]**

**[1860 9]]**

**Classification Report:**

**precision recall f1-score support**

**0 0.73 1.00 0.85 5174**

**1 0.26 0.00 0.01 1869**

**accuracy 0.73 7043**

**macro avg 0.50 0.50 0.43 7043**

**weighted avg 0.61 0.73 0.62 7043**