**Assesment Report**

on

**“Problem Statement”**

submitted as partial fulfillment for the award of

**BACHELOR OF TECHNOLOGY**

**DEGREE**

SESSION 2024-25

in

**Introduction to AI**

By

Santosh Kumar Sah (202401100400165)

**Under the supervision of**

“Mr. Abhishek Shukla”

**KIET Group of Institutions, Ghaziabad**

Affiliated to

**Dr. A.P.J. Abdul Kalam Technical University, Lucknow**

(Formerly UPTU)

**May, 2025**



# 1. Introduction

Problem Statement:  
  
Customer churn prediction is the task of identifying whether a customer will leave a company or stay based on historical data. This is crucial for businesses to retain their customers and implement targeted strategies to reduce churn. In this project, we aim to predict customer churn for a telecom company based on various customer attributes such as contract type, tenure, and total charges. The goal is to build a model that can predict whether a customer will churn or not.  
  
The dataset used for this classification problem is from a telecommunications company and contains various customer-related features, including:  
- Customer demographics  
- Services used by the customer  
- Monthly charges  
- Tenure  
- Total charges  
- Whether the customer has churned  
  
Problem Significance:  
  
Understanding and predicting churn allows companies to proactively engage with customers who are likely to leave, ultimately improving customer retention and reducing revenue losses.

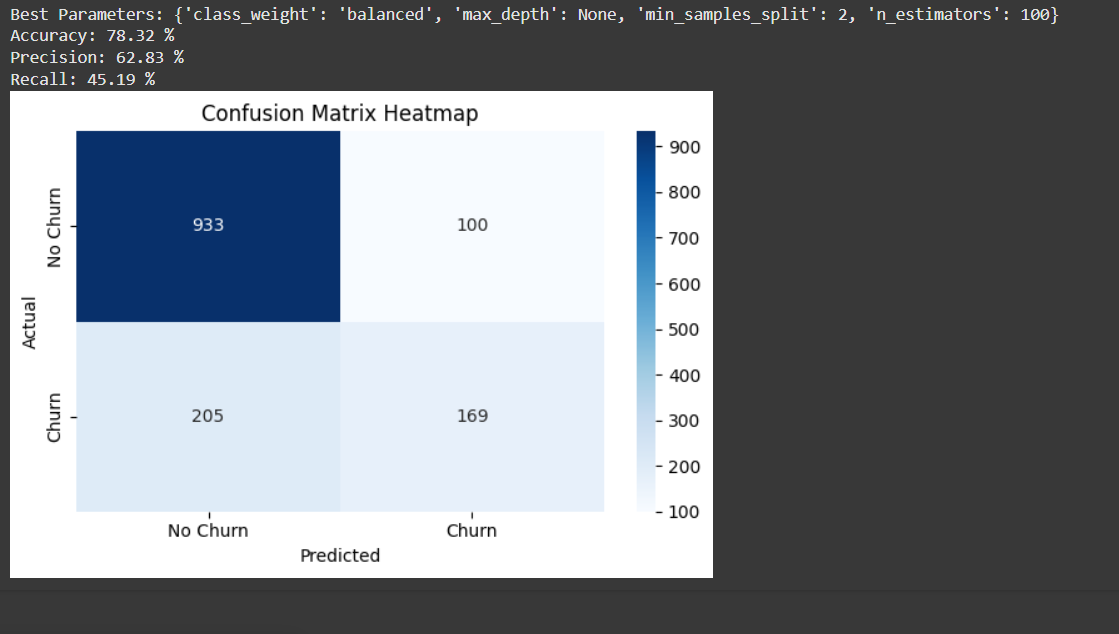
# 2. Methodology

Approach:  
  
The solution to the problem was approached using the following steps:  
  
1. Data Collection and Preprocessing:   
 The dataset was loaded and cleaned. The `TotalCharges` column was converted to numeric values after handling missing or empty entries. The `customerID` column was dropped as it did not provide useful information for prediction. Categorical columns were encoded into numeric values using `LabelEncoder`.  
  
2. Feature Selection and Target Variable:   
 The target variable was identified as `Churn` (whether the customer will churn or not). The other columns were treated as features for the model.  
  
3. Model Selection:   
 A Random Forest Classifier was chosen due to its ability to handle both classification and regression tasks efficiently. It is also robust to overfitting and works well with large datasets and high-dimensional data.  
  
4. Hyperparameter Tuning:   
 Using `GridSearchCV`, we performed a search over several hyperparameters to find the optimal values for the Random Forest Classifier, such as the number of estimators, maximum depth of trees, and minimum samples per split.  
  
5. Model Evaluation:   
 After training the model on the training data, the performance was evaluated using accuracy, precision, recall, and confusion matrix to assess the model's predictive ability.  
  
Evaluation Metrics:  
-Accuracy measures the overall correctness of the model.  
- Precision evaluates how many of the predicted churned customers actually churned.  
- Recall measures how many of the actual churned customers were correctly identified.  
- Confusion Matrixhelps visualize the performance of the classifier, specifically focusing on false positives and false negatives.

# 3. Code

import pandas as pd  
import numpy as np  
import seaborn as sns  
import matplotlib.pyplot as plt  
from sklearn.model\_selection import train\_test\_split, GridSearchCV  
from sklearn.preprocessing import LabelEncoder  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.metrics import confusion\_matrix, accuracy\_score, precision\_score, recall\_score  
  
# Load dataset  
df = pd.read\_csv("5. Classify Customer Churn.csv")  
  
# Drop customerID and handle missing TotalCharges  
df = df[df['TotalCharges'] != ' ']  
df['TotalCharges'] = df['TotalCharges'].astype(float)  
df = df.drop(['customerID'], axis=1)  
  
# Encode categorical variables  
label\_encoders = {}  
for column in df.select\_dtypes(include=['object']).columns:  
 le = LabelEncoder()  
 df[column] = le.fit\_transform(df[column])  
 label\_encoders[column] = le  
  
# Feature and target split  
X = df.drop('Churn', axis=1)  
y = df['Churn']  
  
# Train-test split  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# GridSearch for better RandomForest hyperparameters  
params = {  
 'n\_estimators': [100, 200],  
 'max\_depth': [5, 10, None],  
 'min\_samples\_split': [2, 5],  
 'class\_weight': ['balanced'] # Helps handle class imbalance  
}  
  
grid = GridSearchCV(RandomForestClassifier(random\_state=42), param\_grid=params, scoring='precision', cv=5)  
grid.fit(X\_train, y\_train)  
best\_model = grid.best\_estimator\_  
  
# Predict on test set  
y\_pred = best\_model.predict(X\_test)  
  
# Evaluation  
conf\_matrix = confusion\_matrix(y\_test, y\_pred)  
accuracy = accuracy\_score(y\_test, y\_pred)  
precision = precision\_score(y\_test, y\_pred)  
recall = recall\_score(y\_test, y\_pred)  
  
# Display results  
print("Best Parameters:", grid.best\_params\_)  
print("Accuracy:", round(accuracy, 4))  
print("Precision:", round(precision, 4))  
print("Recall:", round(recall, 4))  
  
# Plot confusion matrix heatmap  
plt.figure(figsize=(6, 4))  
sns.heatmap(conf\_matrix, annot=True, fmt='d', cmap='Blues',  
 xticklabels=['No Churn', 'Churn'],  
 yticklabels=['No Churn', 'Churn'])  
plt.xlabel('Predicted')  
plt.ylabel('Actual')  
plt.title('Confusion Matrix Heatmap')  
plt.tight\_layout()  
plt.show()

# 4. Output



**5.References/Credits**

Dataset Source: IBM Sample Data Sets. (n.d.). Telco Customer Churn. IBM Watson Analytics. Available at: https://www.ibm.com/communities/analytics/watson-analytics-blog/guide-to-sample-datasets/

Libraries: pandas, scikit-learn, matplotlib, seaborn

Environment: Google Colab

Guide/Documentation: scikit-learn official docs