PREDICTING CUSTOMER CHUM USING MACHINE LEARNING TO UNCOVER HIDDEN PATTERN

Student Name: Priya dharshini.R

Register Number: 510623243041

Institution: C.ABDUL HAKEEM COLLEGE OF ENGINEERING AND TECHNOLOGY

Department: BTECH AI&DS

Date of Submission: 08/05/2025

Github Repository Link: https://github.com/Priyadharshini-R123/phase_2.git

1. Problem Statement

Customer churn—the loss of clients or subscribers—is a critical issue for subscription-based businesses, as acquiring new customers is often more costly than retaining existing ones. Despite traditional efforts to mitigate churn, many underlying behavioral patterns and risk indicators remain undetected due to the complexity and volume of customer data.

This project aims to develop a machine learning model that can accurately predict customer churn by analyzing historical customer data and uncovering hidden patterns related to engagement, transaction history, demographics, and support interactions. By identifying high-risk customers early, the organization can proactively implement targeted retention strategies, reduce churn rates, and enhance overall customer lifetime value.

2. Project Objectives

To develop a machine learning model that predicts customer churn by analyzing historical customer data and uncovering hidden behavioral patterns, enabling proactive retention strategies and improved customer lifetime value.

3. Flowchart of the Project Workflow

Data Collection
→ Gather customer data (transactions, demographics, usage patterns, support logs, etc.)

Data Preprocessing
→ Handle missing values
→ Encode categorical variables
→ Normalize/scale numerical data
→ Feature engineering

Exploratory Data Analysis (EDA)
→ Visualize churn trends
→ Identify correlations and hidden patterns

Feature Selection
→ Select relevant variables using statistical tests or model-based methods

Model Selection & Training
→ Train ML models (e.g., Logistic Regression, Random Forest, XGBoost, Neural Networks)
→ Perform cross-validation

Model Evaluation
→ Evaluate with metrics like accuracy, precision, recall, F1-score, AUC-ROC

Model Interpretation
→ Use SHAP, LIME, or feature importance to understand key churn drivers

Deployment
→ *Integrate model into a production system or dashboard for real-time churn prediction*

Monitoring & Updating
→ *Monitor model performance*
→ *Retrain periodically with new data*

4. Data Description

Feature Name	***Description***
CustomerID	*Unique identifier for each customer*
Gender	*Customer's gender (e.g., Male, Female)*
Age	*Age of the customer*
Tenure	*Number of months the customer has been with the company*
SubscriptionType	*Type of subscription plan (e.g., Basic, Premium, Family)*
MonthlyCharges	*Amount charged monthly*
TotalCharges	*Total amount charged to the customer*
PaymentMethod	*Method used by the customer to pay (e.g., Credit Card, Bank Transfer)*

Feature Name	***Description***
ContractType	*Contract duration (e.g., Month-to-month, One year, Two year)*
InternetService	*Type of internet service (e.g., DSL, Fiber Optic, None)*
OnlineSecurity	*Whether the customer has online security add-on (Yes/No)*
TechSupport	*Whether the customer has technical support add-on (Yes/No)*
StreamingServices	*Use of streaming services (e.g., TV, Movies, Music)*
CustomerSupportCalls	*Number of times the customer contacted support*
LastInteractionDate	*Date of the last customer activity*
Churn (Target)	*Whether the customer churned (1 = Yes, 0 = No)*

5. Data Preprocessing

1. ***Data Cleaning***

    - ***Remove duplicates****: Drop any duplicate rows using drop_duplicates().*
    - ***Handle missing values****:*
        - *Numerical: Fill with median/mean or use interpolation.*

   - *Categorical: Fill with mode or use a placeholder like "Unknown".*

2. ***Feature Engineering***
   - ***Create new features*** *(e.g., AverageMonthlySpend = TotalCharges / Tenure).*
   - ***Convert dates****: Calculate recency (days_since_last_interaction from LastInteractionDate).*
   - ***Bin continuous variables****: Group age or tenure into categories if helpful.*
3. ***Encoding Categorical Variables***
   - ***Label Encoding*** *for binary features (e.g., Yes/No).*
   - ***One-Hot Encoding*** *for multi-class features (e.g., SubscriptionType, PaymentMethod).*
4. ***Scaling / Normalization***
   - *Use* ***StandardScaler*** *or* ***MinMaxScaler*** *for continuous features (e.g., MonthlyCharges, TotalCharges) to normalize the range.*
5. ***Outlier Detection and Treatment***
   - *Use Z-score or IQR methods to detect and optionally cap/transform outliers in numerical columns.*
6. ***Class Imbalance Handling*** *(if churned vs. not churned is imbalanced)*
   - *Use* ***SMOTE*** *(Synthetic Minority Over-sampling Technique) to oversample minority class.*
   - *Or use* ***class weights*** *in model training.*

7. ***Train-Test Split***

    - *Split the dataset (e.g., 80% training, 20% test) using stratified sampling to maintain churn ratios.*

8. ***Save Preprocessed Data***

    - *Optionally, export the cleaned and encoded dataset to CSV or a binary format (e.g., pickle) for reuse.*

6. Exploratory Data Analysis (EDA)

1. Understand the Target Variable

- ***Value Counts & Distribution***

python

CopyEdit

```
df['Churn'].value_counts(normalize=True).plot(kind='bar', title='Churn Distribution')
```

    - *Helps identify class imbalance.*

2. Summary Statistics

- *Use df.describe() to summarize numerical features.*
- *Use df.info() to check data types and null values.*

3. Univariate Analysis

- ***Numerical Features****: Histograms or KDE plots (e.g., Age, MonthlyCharges, Tenure)* *python*

CopyEdit

df['Tenure'].hist(bins=20)

- ***Categorical Features****: Bar plots*

python

CopyEdit

df['ContractType'].value_counts().plot(kind='bar')

4. Bivariate Analysis

- ***Churn vs. Numerical Variables****: Boxplots or violin plots*

python

CopyEdit

sns.boxplot(x='Churn', y='MonthlyCharges', data=df)

- ***Churn vs. Categorical Variables****: Stacked bar plots or heatmaps*

python

CopyEdit

pd.crosstab(df['ContractType'], df['Churn']).plot(kind='bar', stacked=True)

5. Correlation Analysis

- ***Correlation Heatmap*** *for numerical variables*

python

CopyEdit

```
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
```

- *Reveals multicollinearity or redundant features.*

6. Feature Interactions

- *Use pair plots or scatter matrix to explore interactions (use sample to reduce overload)*

python

CopyEdit

```python
sns.pairplot(df[['Tenure', 'MonthlyCharges', 'TotalCharges', 'Churn']],
hue='Churn')
```

7. Customer Segmentation (Optional)

- *Apply clustering (e.g., KMeans) or PCA for dimensionality reduction to visualize customer groups.*

7. Feature Engineering

1. Basic Derived Features

- *AverageMonthlySpend = TotalCharges / Tenure*
  (Use only if Tenure > 0, avoid divide-by-zero)
- *IsSenior = Age > 60*
- *HasMultipleServices = Combine streaming, internet, and security flags*

2. Customer Engagement Features

- *EngagementScore = Weighted score of usage/activity-based features (e.g., usage of streaming, support calls, internet type)*
- *DaysSinceLastInteraction = Current date - LastInteractionDate*

- *SupportCallRate = CustomerSupportCalls / Tenure*

3. Subscription Characteristics

- *ContractLength = Map "Month-to-month" → 1, "One year" → 12, "Two year" → 24*
- *IsAutoPay = PaymentMethod in ['Credit card (automatic)', 'Bank transfer (automatic)']*
- *IsLongTermCustomer = Tenure > 24 (arbitrary threshold)*

4. Binary Flags for Add-ons

- *UsesTechSupport = Yes/No to 1/0*
- *UsesStreaming = Combine multiple streaming services (TV, music, etc.)*
- *HasOnlineSecurity = Binary flag for security service*

5. Customer Lifetime Metrics

- *LifetimeValue = MonthlyCharges × Tenure*
- *ChurnRiskScore = Create rule-based score using a weighted sum of red flags (e.g., short tenure, high support calls, month-to-month contract)*

6. Interaction Features

- *MonthlyCharges × ContractLength → Measures financial commitment*
- *Tenure × ContractLength → Loyalty-adjusted commitment*

Tips for Good Feature Engineering

- *Consider business logic: Features must make real-world sense.*
- *Avoid data leakage: Don't use features that are only available after churn.*
- *Evaluate feature importance later using tree-based models or SHAP values.*

8. Model Building

1. Split Data

python

CopyEdit

from sklearn.model_selection import train_test_split

X = df.drop('Churn', axis=1)

y = df['Churn']

X_train, X_test, y_train, y_test = train_test_split(

X, y, test_size=0.2, stratify=y, random_state=42

)

---

2. Select and Train Models

Try several models to compare performance.

python

CopyEdit

from sklearn.linear_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

from xgboost import XGBClassifier

models = {

"Logistic Regression": LogisticRegression(max_iter=1000),

"Random Forest": RandomForestClassifier(n_estimators=100),

```
    "XGBoost": XGBClassifier(use_label_encoder=False, eval_metric='logloss')
}


for name, model in models.items():
    model.fit(X_train, y_train)
    print(f"{name} trained.")
```

---

3. Evaluate Models

Use appropriate metrics for binary classification.

python

CopyEdit

```
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score, roc_auc_score


def evaluate_model(model, X_test, y_test):
    y_pred = model.predict(X_test)
    y_prob = model.predict_proba(X_test)[:, 1]
    return {
        "Accuracy": accuracy_score(y_test, y_pred),
        "Precision": precision_score(y_test, y_pred),
```

"Recall": recall_score(y_test, y_pred),

"F1": f1_score(y_test, y_pred),

"ROC AUC": roc_auc_score(y_test, y_prob)

}

for name, model in models.items():

scores = evaluate_model(model, X_test, y_test)

print(f"\n{name} Scores:")

for metric, score in scores.items():

print(f"{metric}: {score:.4f}")

---

4. Tune the Best Model (Optional)

Use grid search or randomized search for hyperparameter optimization.

python

CopyEdit

from sklearn.model_selection import GridSearchCV

param_grid = {

'n_estimators': [100, 200],

'max_depth': [4, 6, 8],

}

grid = GridSearchCV(RandomForestClassifier(), param_grid, cv=5, scoring='roc_auc')

grid.fit(X_train, y_train)

*best_model = grid.best_estimator_*

---

5. Interpret the Model

Use feature importance or SHAP to understand churn drivers.

python

CopyEdit

import shap

explainer = shap.TreeExplainer(best_model)

shap_values = explainer.shap_values(X_test)

shap.summary_plot(shap_values, X_test)

9. Visualization of Results & Model Insights

1. Churn Distribution

***Purpose**: Understand class imbalance.*

python

CopyEdit

df['Churn'].value_counts().plot(kind='bar', title='Churn Distribution')

✅ *2. Feature Importance (Tree-Based Models)*

***Purpose**: Identify top predictors of churn.*

python

CopyEdit

```
importances = best_model.feature_importances_

features = X_train.columns


# Plot

import matplotlib.pyplot as plt

import seaborn as sns


feat_imp = pd.Series(importances, index=features).sort_values(ascending=False)
```

sns.barplot(x=feat_imp[:10], y=feat_imp.index[:10])

plt.title("Top 10 Feature Importances")

✅ 3. SHAP Summary Plot

Purpose: *Explain model predictions globally and locally.*

python

CopyEdit

```
import shap
explainer = shap.TreeExplainer(best_model)
shap_values = explainer.shap_values(X_test)

shap.summary_plot(shap_values, X_test)
```

- ***Color***: *Red = high feature value, Blue = low feature value*
- ***X-axis***: *Impact on churn probability*

✅ 4. Confusion Matrix

Purpose: *Visualize model performance.*

python

CopyEdit

```
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

ConfusionMatrixDisplay.from_estimator(best_model, X_test, y_test)
```

✅ 5. ROC Curve

Purpose: *Evaluate classification threshold and trade-offs.*

python

CopyEdit

```
from sklearn.metrics import roc_curve, auc

y_prob = best_model.predict_proba(X_test)[:, 1]
fpr, tpr, _ = roc_curve(y_test, y_prob)

plt.plot(fpr, tpr, label=f"AUC = {auc(fpr, tpr):.2f}")
plt.plot([0, 1], [0, 1], '--')
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend()
```

✅ *6. Partial Dependence Plot (Optional)*

Purpose: *Show marginal effect of features.*

python

CopyEdit

```
from sklearn.inspection import plot_partial_dependence

plot_partial_dependence(best_model, X_test, ['Tenure', 'MonthlyCharges'])
```

💡 *Model Insights (Example)*

- ***Top Churn Drivers***:
    - *Short tenure*
    - *Month-to-month contracts*
    - *High monthly charges*
    - *Multiple support calls*
- ***Customer Segments at Risk***:
    - *New users (<6 months) on flexible contracts*
    - *Users not using add-ons like online security or tech support*
- ***Business Actions***:

- *Offer loyalty discounts*
- *Encourage annual plans*
- *Improve support experience*

10. Tools and Technologies Used

📊 *Data Collection & Storage*

- ***CSV / Excel / SQL*** *— Storing and retrieving customer data*
- ***Pandas*** *— Data manipulation and analysis*
- ***NumPy*** *— Numerical operations*

🧹 *Data Preprocessing*

- ***Scikit-learn (sklearn)*** *— Imputation, encoding, scaling, train-test split*
- ***Pandas Profiling / Sweetviz*** *— Automated EDA reports (optional)*

📈 *Exploratory Data Analysis (EDA)*

- ***Matplotlib / Seaborn*** *— Visualization (bar plots, histograms, heatmaps)*
- ***Plotly*** *— Interactive charts (optional)*

🧠 *Model Building*

- ***Scikit-learn*** *— Logistic Regression, Random Forest, model evaluation*
- ***XGBoost / LightGBM*** *— Gradient boosting models for high accuracy*

- ***Imbalanced-learn*** *— SMOTE, under-/oversampling*

📊 *Model Interpretation*

- ***SHAP*** *— Explainable AI to interpret feature contributions*
- ***LIME*** *— Local model interpretability (optional)*
- ***Feature Importance Plots*** *— Built into tree-based models*

🧪 *Model Evaluation*

- ***Scikit-learn*** *— ROC curve, confusion matrix, F1 score, accuracy, precision, recall*

🚀 *Model Deployment (Optional)*

- ***Flask / FastAPI*** *— Build REST API for model inference*
- ***Streamlit / Dash*** *— Create interactive dashboards*
- ***Docker*** *— Containerize the model*
- ***AWS / Azure / GCP*** *— Deploy model to the cloud*

📦 *Version Control & Collaboration*

- ***Git / GitHub*** *— Versioning and team collaboration*
- ***Jupyter Notebook*** *— For prototyping and documentation*

11. Team Members and Contributions

NAME	ROLE	RESPONSIBLITY
PRIYADHARSHINI R	Lead	Oversee project development, coordinate team activities, ensure timely delivery of milestones, and contribute to documentation and Data Engineer final
NANDHITHA M	Data Engineer	Collect data from APIs (e.g., Twitter), manage dataset storage, clean and preprocess text data, and ensure quality of input data
Varshini.S, Vaishnavi.A	NLP Specialist / Data	Build sentiment and emotion classification models, perform feature engineering, and evaluate model performance using suitable metrics
Sonika.R	Data Analyst / Visualization	Conduct exploratory data analysis (EDA), generate insights, and develop visualizations such as word clouds, emotion trends, and sentiment
