**Phase - 3**

**Submission Document: Customer Churn Prediction**

**Student Name:** [ Navaneetham V ]  
**Register Number:** [ 422623104001 ]  
**Institution:** [ University college of engineering panruti ]  
**Department:** [ Computer science and engineering ]  
**Date of Submission:** [ 17/05/2025 ]  
**GitHub Repository Link:** [<https://github.com/Navaneetham-25/PREDICTING-CUSTOMER-PROJECT.git>]

**Problem Statement**

Customer churn is a critical challenge for subscription-based businesses such as telecom, banking, and SaaS platforms. Predicting which customers are likely to churn enables proactive retention strategies. This project builds a classification model to predict customer churn based on demographic and behavioral data. This is a supervised learning (classification) problem.

**Abstract**

This project aims to solve the business-critical problem of customer churn prediction using machine learning models. It involves data preprocessing, feature engineering, exploratory data analysis (EDA), and model building using Logistic Regression, Random Forest, and XGBoost. After evaluating performance using metrics like accuracy and F1-score, the model highlights key churn indicators, aiding businesses in customer retention strategies.

**System Requirements**

**Hardware:**

* Minimum RAM: 4 GB (Recommended: 8 GB+)
* Processor: Intel i5/i7 or equivalent

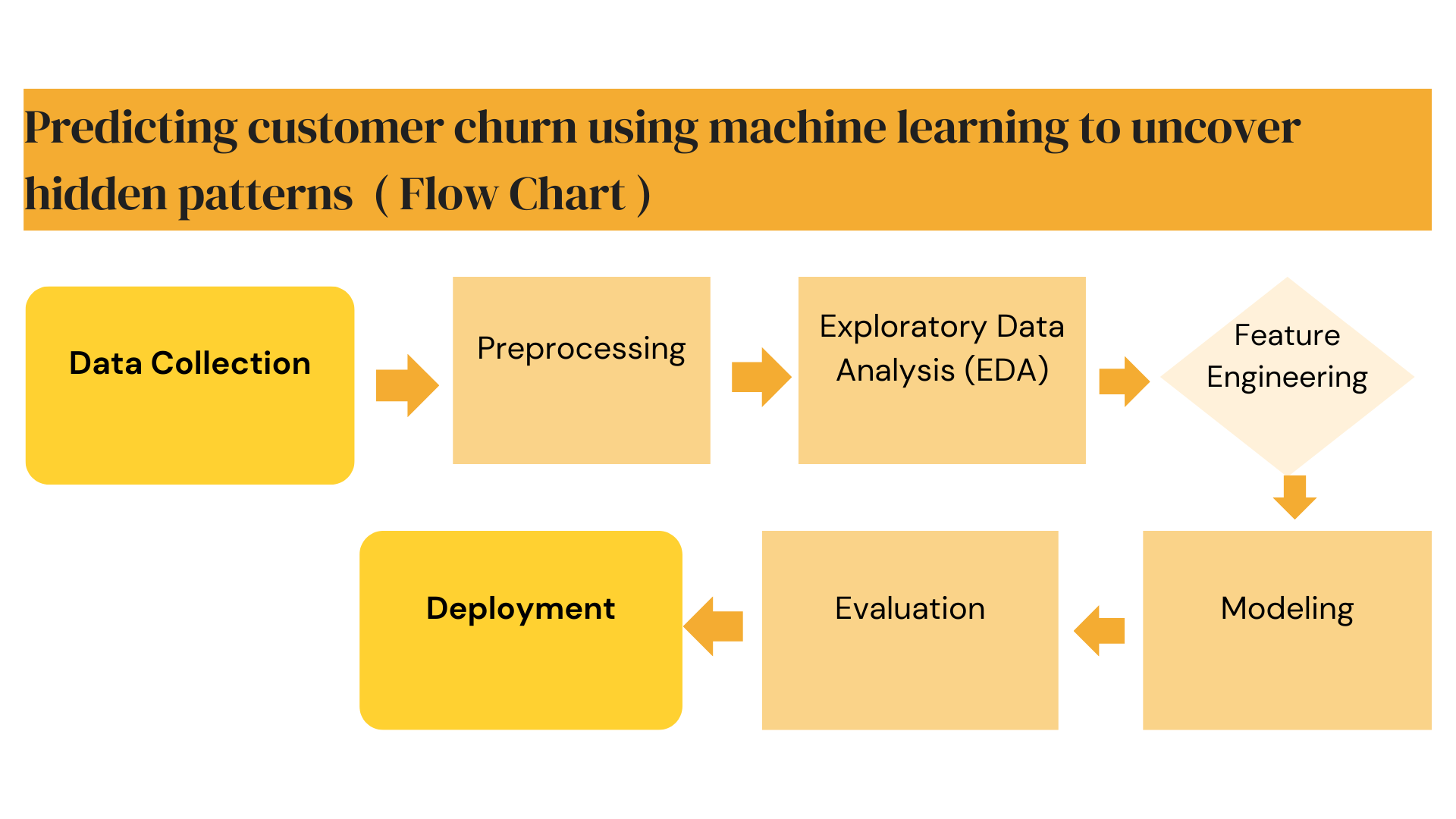
**Software:**

* Python 3.8+
* Libraries: pandas, numpy, seaborn, matplotlib, scikit-learn, xgboost
* IDE: Google Colab or Jupyter Notebook

**Objectives**

* Predict whether a customer will churn.
* Identify key factors influencing churn.
* Compare multiple models based on evaluation metrics.
* Assist stakeholders with actionable business insights.

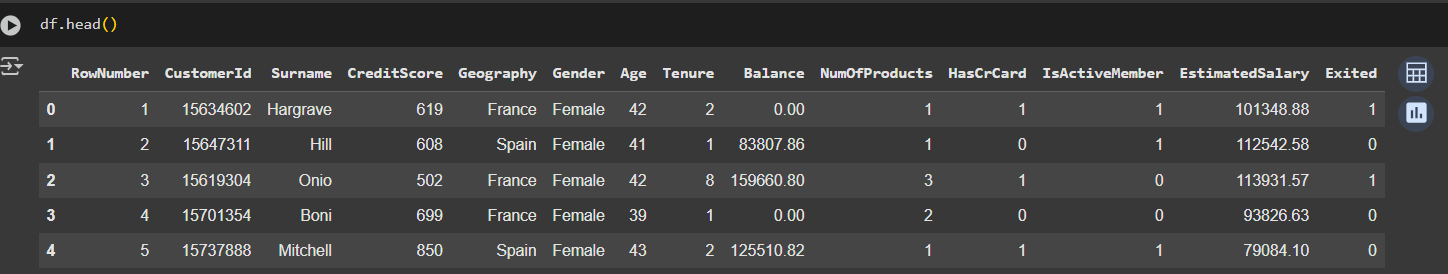
**Flowchart of Project Workflow**



**Dataset Description**

* Source: Kaggle
* Type: Public
* Rows: ~10,000, Columns: 14
* Format: CSV

**Screenshot** :



**Imports and Setup**

**Definition:  
This section imports all necessary libraries for data processing, visualization, modeling, and evaluation. It also enables file upload capability for Google Colab**

**Code :**

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**from sklearn.preprocessing import LabelEncoder, StandardScaler**

**from sklearn.model\_selection import train\_test\_split**

**from sklearn.linear\_model import LogisticRegression**

**from sklearn.ensemble import RandomForestClassifier**

**from xgboost import XGBClassifier**

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

**from google.colab import files**

**Data Loading**

**Definition:  
This part handles the dataset upload via Colab's interface and loads it into a DataFrame for analysis.**

**Code :**

**# Upload dataset**

**uploaded = files.upload()**

**df = pd.read\_csv(list(uploaded.keys())[0])**

**print(f"Initial rows: {len(df)}")**

**Data Preprocessing**

**Definition**:  
Removes irrelevant columns and encodes categorical variables into numeric format using one-hot encoding to make them suitable for ML algorithms.

**Code** :

# Drop non-informative columns

df.drop(['RowNumber', 'CustomerId', 'Surname'], axis=1, inplace=True)

# Encode categorical variables: Geography and Genderdf = pd.get\_dummies(df, columns=['Geography', 'Gender'], drop\_first=True)

**Exploratory Data Analysis (EDA)**

**Definition:  
Visual exploration to understand data distribution, detect imbalance, analyze feature relationships, and identify potential predictors.**

**Churn Distribution**

**📌 Definition:**

**This plot displays the distribution of the target variable (Exited), where:**

* **0 indicates customers who stayed.**
* **1 indicates customers who churned.**

**This helps assess class imbalance, which is important for model evaluation (e.g., accuracy may be misleading if one class dominates).**

**Code :**

plt.figure(figsize=(6, 4))

sns.countplot(x='Exited', data=df)

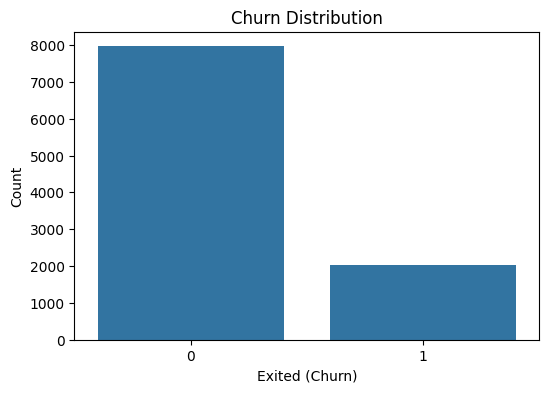
plt.title('Churn Distribution')

plt.xlabel('Exited (Churn)')

plt.ylabel('Count')

plt.show()

Output :



**Age by Churn**

**📌 Definition:**

**This boxplot shows the distribution of customer age segmented by churn status. It helps understand whether age has any correlation with churn, by comparing:**

* **Medians**
* **Spread**
* **Outliers**

**A clear age trend might suggest age is an important predictive feature.**

**Code :**

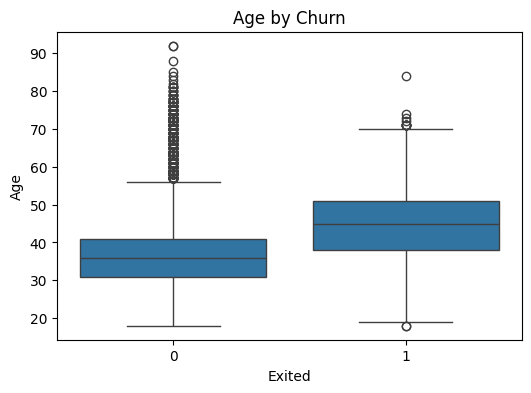
plt.figure(figsize=(6, 4))

sns.boxplot(x='Exited', y='Age', data=df)

plt.title('Age by Churn')

plt.show()

**Output** :



**Correlation Heatmap (Numerical Features Only)**

**📌 Definition:**

**The heatmap visualizes the Pearson correlation coefficients among numerical features. It helps:**

* **Identify multicollinearity (highly correlated features)**
* **Understand relationships between features and the target variable (Exited)**
* **Guide feature selection or dimensionality reduction**

**Code :**

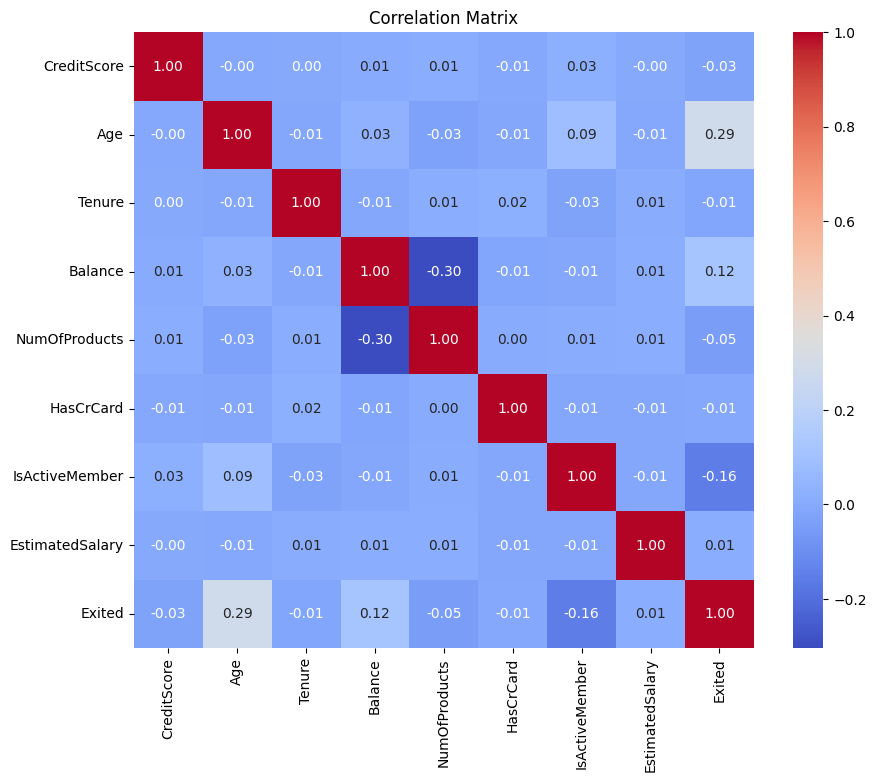
plt.figure(figsize=(10, 8))

sns.heatmap(df.select\_dtypes(include=[np.number]).corr(), annot=True, fmt=".2f", cmap='coolwarm')

plt.title('Correlation Matrix')

plt.show()

**Output** :



**Feature Definition and Scaling**

**Definition**:  
Separates the target variable (Exited) from features and applies standardization to numerical values for model compatibility

**Code** :

# Define features and target

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

y = df['Exited']

# Train-test split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Feature scaling

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

**Model Training and Evaluation**

**Definition:  
Trains three different models (Logistic Regression, Random Forest, XGBoost) on the dataset and evaluates their performance using accuracy, confusion matrix, and classification report.**

**Code :**

# Define models

models = {

"Logistic Regression": LogisticRegression(max\_iter=1000),

"Random Forest": RandomForestClassifier(n\_estimators=100),

"XGBoost": XGBClassifier(eval\_metric='logloss')

}

# Train and evaluate models

for name, model in models.items():

print(f"\n{name} Results")

model.fit(X\_train, y\_train)

y\_pred = model.predict(X\_test)

print("Accuracy:", accuracy\_score(y\_test, y\_pred))

print(confusion\_matrix(y\_test, y\_pred))

print(classification\_report(y\_test, y\_pred))

# Confusion Matrix Visualization

plt.figure(figsize=(4, 3))

sns.heatmap(confusion\_matrix(y\_test, y\_pred), annot=True, fmt="d", cmap='Blues')

plt.title(f'Confusion Matrix: {name}')

plt.xlabel('Predicted')

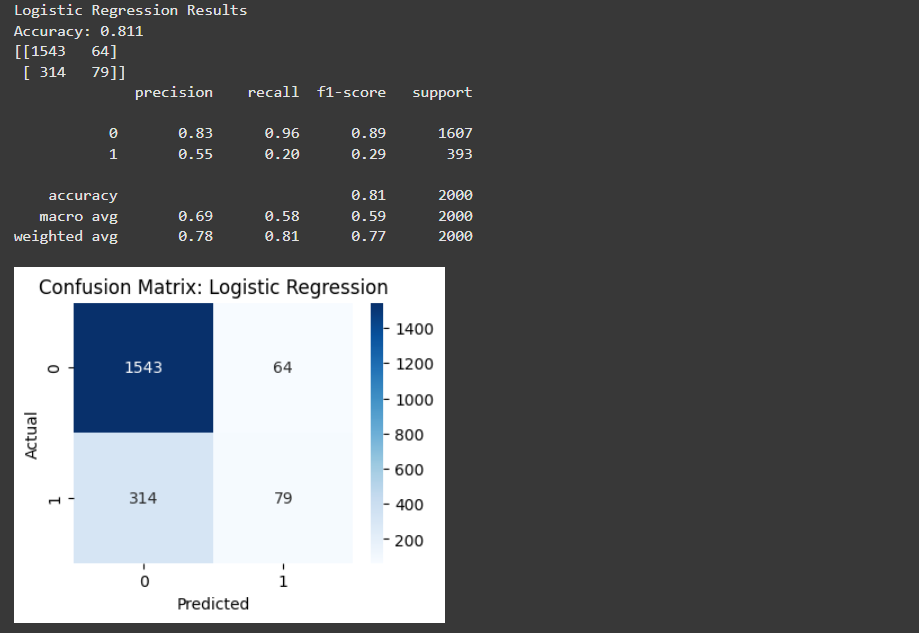
plt.ylabel('Actual')

plt.show()

Logistic Regression ( Output ) :

**Definition in Context**

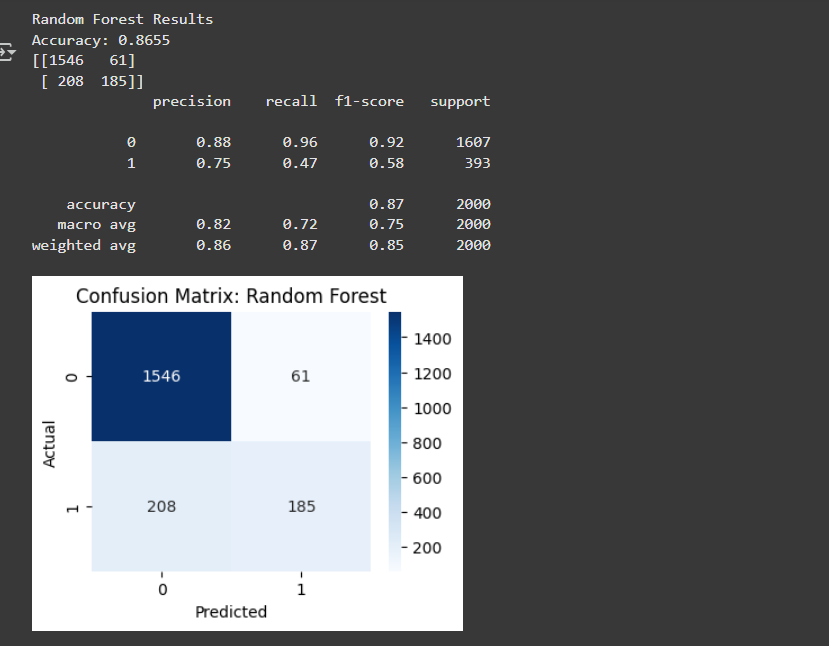
Logistic Regression is a supervised machine learning algorithm used for binary classification tasks. In this project, it is employed to predict whether a customer will churn (Exited = 1) or not (Exited = 0) based on various customer attributes such as age, tenure, balance, credit score, and geography.



Random Forest ( Output ) :

**Definition in Context**

Random Forest is an ensemble machine learning algorithm used for classification and regression tasks. In this project, it is applied to predict customer churn (whether a customer will exit or not) by building multiple decision trees and combining their predictions for a more accurate and robust result.



XGBoost ( Output ) :

**Definition in Context**

XGBoost (Extreme Gradient Boosting) is a high-performance, scalable machine learning algorithm based on gradient boosting decision trees. In this project, XGBoost is used to classify whether a customer will churn (Exited = 1) or stay (Exited = 0), based on features like geography, age, tenure, balance, and more.



**Feature Importance (Random Forest)**

**Definition**:  
Extracts and visualizes the top 15 most important features as determined by the Random Forest model to interpret which features most influence churn.

**Code** :

# Feature importance from Random Forest

rf\_model = models["Random Forest"]

importances = rf\_model.feature\_importances\_

features = X.columns

importance\_df = pd.DataFrame({'Feature': features, 'Importance': importances})

importance\_df.sort\_values(by='Importance', ascending=False, inplace=True)

# Visualization of Top Features

plt.figure(figsize=(10, 6))

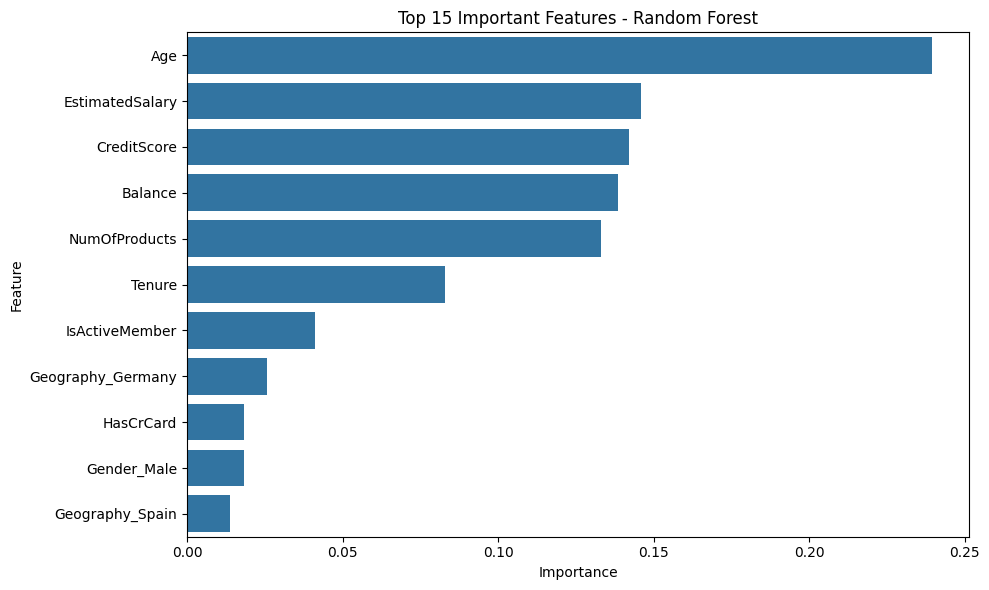
sns.barplot(data=importance\_df.head(15), x='Importance', y='Feature')

plt.title('Top 15 Important Features - Random Forest')

plt.tight\_layout()

plt.show()

**Output** :



**Deployment (Optional)**

If deployed via Google colab:

* Platform: Google colab Cloud
* Link: <https://colab.research.google.com/github/Navaneetham-25/Source-code/blob/main/Predict_Custumer_churn.ipynb>

**Source Code**

GitHub Repository: [<https://github.com/Hariprasath-02/Source-code.git>]

**Future Scope**

* Handle class imbalance using SMOTE or advanced sampling
* Integrate live API endpoints for real-time predictions
* Incorporate user behavior data for improved accuracy

Team members and roles

|  |  |
| --- | --- |
| Team Members | Role |
| **Hariprasath M**  **(Project Lead)** | Responsible for data collection, loading, and initial preprocessing of the dataset. Handled missing values and ensured data quality. |
| **Santhosh M** | Led the Exploratory Data Analysis (EDA) and created visualizations. Identified patterns, correlations, and churn-related insights. |
| **Navaneetham V** | Handled feature engineering, including creation of new features, encoding, and scaling. Also performed feature selection. |
| **Kanagavally S** | Focused on model building, training, hyperparameter tuning, and evaluating multiple classification models. Prepared evaluation visuals. |
| **Dhavan RG** | |  | | --- | |  |  |  | | --- | | Managed deployment of the final model using Streamlit. Designed the UI, integrated the model, and created the project documentation and GitHub repository. | |