FINAL REPORT

Bank Customer Churn Prediction Using Machine Learning & Deep Learning

Author: L.Habishalini

Table of Contents

Introduction

Banks and financial institutions often struggle with customer churn, which happens when customers stop using their services or close their accounts. This can be caused by dissatisfaction, better offers from competitors, or changing financial needs. Keeping current customers is important for long-term profits because it usually costs less than finding new ones.

This project aims to build a Customer Churn Prediction System for ABC Multistate Bank using machine learning and deep learning methods. By looking at customer behavior, transaction history, and demographic information, the system will help spot customers who might leave. Finding these customers early lets the bank use targeted strategies to keep them, improve satisfaction, and build loyalty.

This report outlines the end-to-end process, including data preprocessing, exploratory data analysis (EDA), feature engineering, model development, evaluation, and result visualization. The models used include Logistic Regression, Random Forest, XGBoost, and an Artificial Neural Network (ANN).
The ultimate objective is to help ABC Bank make data-driven decisions to reduce churn and maximize customer retention.

Problem Definition

ABC Multistate Bank has to deal with the yearly tendency of some of their customers to either close their accounts or simply stop banking with them. Customer churn diminishes the customer base, affects revenue yearly, and restricts the possibility of long-term business growth.

The key challenge is to detect early signs of churn within the huge volume of customer data gathered at the bank. The reasons for churn need to be understood, as does detecting customers that are likely to churn beforehand, so the bank can take timely and proactive action.

The main aim is to develop a predictive system that classifies customers into two categories:
Churn (1): Customers likely to leave
No Churn (0): Customers who will probably stay with the company

This model will take into consideration credit score, age, tenure, balance, number of products, active status, and salary in order to predict the probability of churning.
By incorporating machine learning and deep learning approaches, the project allows ABC Bank to be proactive in data-driven decision-making. This facilitates personalization of offers, loyalty programs, and enhanced customer experience. It will directly impact profitability and long-term customer retention.

Aim & Objectives

Aim

The main goal of this project is to develop a predictive machine learning system that can accurately identify customers who are likely to leave ABC Multistate Bank. The system will analyze customer behavior and demographic data to uncover key factors influencing churn, helping the bank take proactive measures to improve customer satisfaction and design effective retention strategies.

Objectives

- Preprocess and clean the structured customer data to ensure high data quality and reliability for model training.
- Perform in-depth Exploratory Data Analysis (EDA) to identify meaningful trends and churn related patterns within the data.
- Engineer meaningful features such as Balance, Tenure, and Product Usage to enhance model performance.
- Develop and compare multiple machine learning models, such as Logistic Regression, Random Forest, XGBoost, to determine the most effective predictive approach.
- Implement a deep learning model (Artificial Neural Network) to enhance predictive performance.
  - Evaluate performance using Accuracy, Precision, Recall, F1-Score, and ROC-AUC for fair comparison.
- Visualize customer churn trends and derive actionable business insights for retention strategies using Seaborn & Matplotlib.
- Provide actionable business insights and recommendations that can guide ABC Bank in improving customer loyalty, reducing churn, and increasing overall profitability.

Dataset Description

The dataset for this project is sourced from **Kaggle**, titled "**Bank Customer Churn Dataset**", created by Gaurav Topre. Dataset Link

Dataset Overview

The dataset contains information about bank customers and whether they have churned or not.

Number of Records: Approximately 10,000

Target Variable: churn (1 = customer left, 0 = customer stayed)

The dataset is balanced enough for binary classification and includes both numerical and categorical variables suitable for predictive modelling.

Features

Column Name	Description	Usage
customer_id	Unique ID (not used)	Unused
credit_score	Credit score of customer	Input
country	Country of residence	Input
gender	Gender of customer	Input
age	Age of customer	Input
tenure	Number of years customer has stayed with bank	Input
balance	Account balance	Input
products_number	Number of bank products held	Input
credit_card	Whether customer has a credit card (1/0)	Input
active_member	Whether customer is active (1/0)	Input
estimated_salary	Estimated annual salary	Input
churn	Target variable (1 = left, 0 = stayed)	Target

Solution

The proposed solution involves developing a predictive system to classify customers as likely to churn or not.

The solution includes the following steps:

- **Data Loading and Preprocessing:**
  Cleaning the dataset, encoding categorical variables, handling missing values, and scaling numerical features.
- **Exploratory Data Analysis (EDA):**
  Visualizing distributions, correlations, and patterns to understand key factors contributing to churn.
- **Feature Engineering:**
  Creating new features such as balance_per_tenure and product_usage_ratio to enhance model performance.
- **Model Development:**
  Training and comparing multiple machine learning models and a Deep Learning model (ANN).
- **Model Evaluation:**
  Using metrics like Accuracy, Precision, Recall, F1-Score, and ROC-AUC to evaluate performance and select the best model.
- **Visualization & Insights:**
  Generating plots to visualize churn trends and provide actionable insights for customer retention.

Technical Approach

Programming Language:

Python

Libraries & Tools

- Pandas, NumPy – Data manipulation and analysis
- Matplotlib, Seaborn – Visualization of trends and churn patterns
- Scikit-learn – Preprocessing, ML models, evaluation metrics
- XGBoost – Gradient boosting algorithm for higher accuracy
- TensorFlow / Keras – Implementation of Artificial Neural Networks
- Joblib – Model persistence (saving trained models)
- Jupyter Notebook – Development and experimentation environment

Workflow Steps

- Load and inspect dataset
- Preprocess data (encoding, scaling, cleaning)
- Perform EDA to understand churn behavior
- Engineer additional relevant features
- Split data into training/testing sets
- Train multiple ML models and an ANN
- Evaluate model performance and select the best model
- Visualize and document insights

Codes & Explanation

Data Loading

- Imported essential libraries
- Loaded the CSV dataset using pandas
- Verified data integrity (shape, types, null values)

```
[1]: # Import Libraries
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from collections import Counter

# ML utilities (Sklearn)
from sklearn.model_selection import train_test_split, StratifiedKFold, cross_val_score, RandomizedSearchCV, GridSearchCV
from sklearn.preprocessing import StandardScaler, MinMaxScaler, OneHotEncoder, LabelEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer

# Models
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, AdaBoostClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
import xgboost as xgb

# Metrics & Utilities
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score, confusion_matrix, classification_report, RocCurveDisplay, ConfusionMatrixDi
import joblib
import warnings
warnings.filterwarnings("ignore")

# Display Settings
sns.set_theme(style='whitegrid', palette='muted', font_scale=1.05)
pd.set_option('display.max_columns', 200)
```

2. Load the Data

```
[2]: #Load Dataset
df = pd.read_csv("../Data/Bank Customer Churn Prediction.csv")
print("Shape:", df.shape)
df.head()
```

```
Shape: (10000, 12)
```

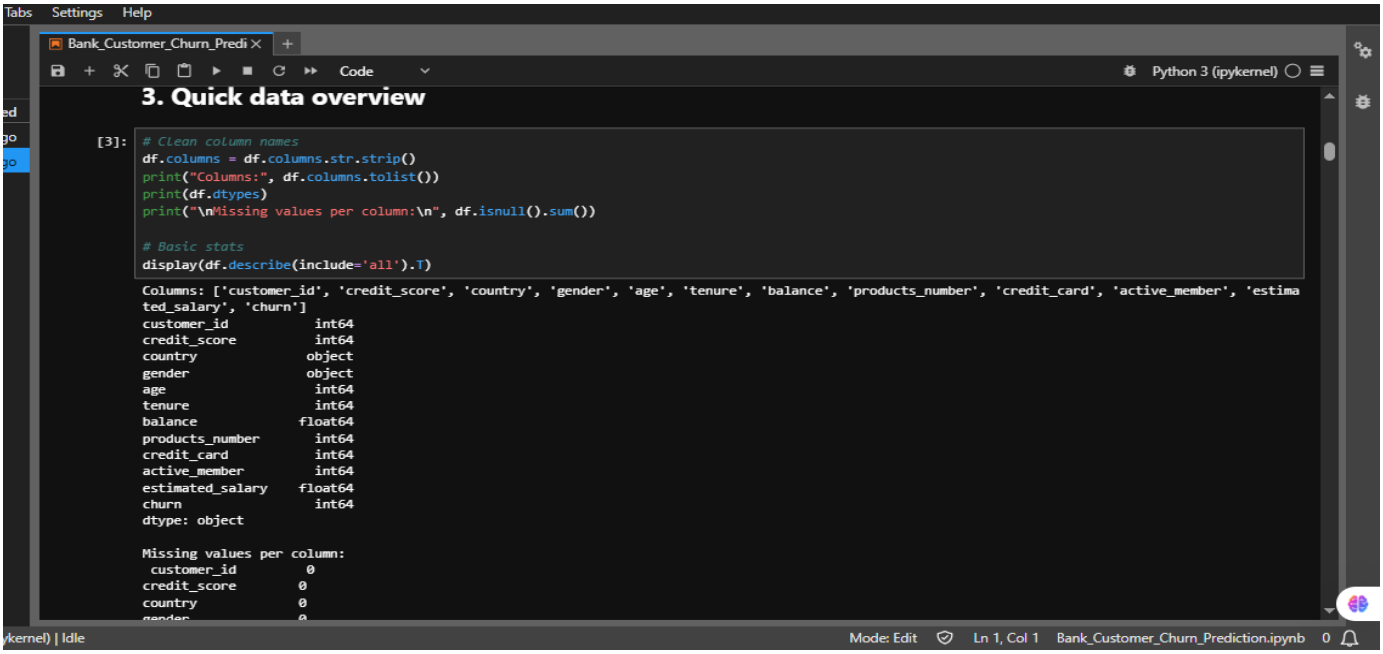
[2]:

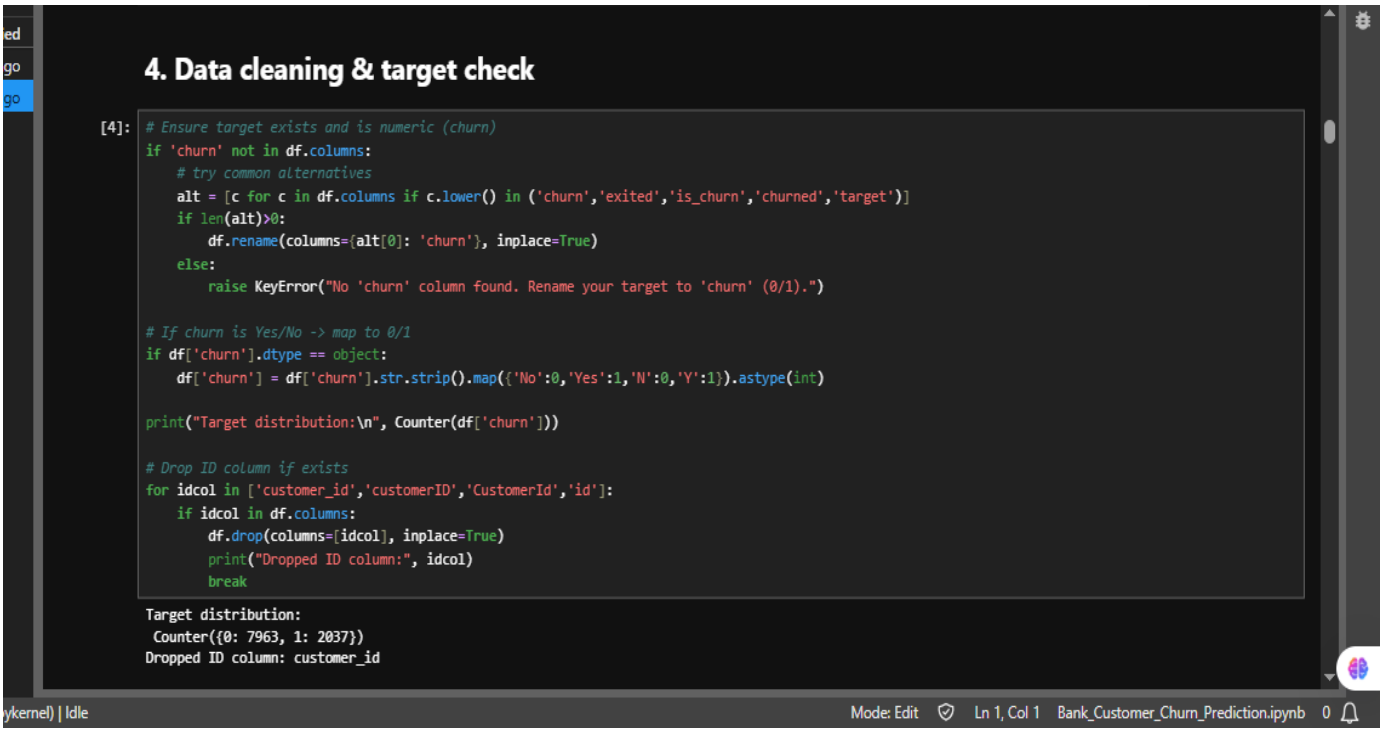
	customer_id	credit_score	country	gender	age	tenure	balance	products_number	credit_card	active_member	estimated_salary	chu
0	15634602	619	France	Female	42	2	0.00	1	1	1	101348.88	
1	15647311	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	
2	15619304	502	France	Female	42	8	159660.80	3	1	0	113931.57	
3	15701354	699	France	Female	39	1	0.00	2	0	0	93826.63	
4	15737888	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	

Figure 1 - Data Loading

Data Preprocessing

- Removed unnecessary columns (e.g., `customer_id`)
- One-hot encoded categorical variables (`country`, `gender`)
- Scaled numerical features using StandardScaler
- Split data into training (80%) and testing (20%) sets using stratified sampling





EDA & Visualization

- Analyzed churn distribution
- Visualized relationships between churn and key variables (Age, Balance, Tenure, Products)
- Used correlation heatmaps and violin plots
- Created gender-wise and country-wise churn comparisons

Tabs Settings Help

Bank_Customer_Churn_Predi

Code

Python 3 (ipykernel)

6. Exploratory Data Analysis (EDA)

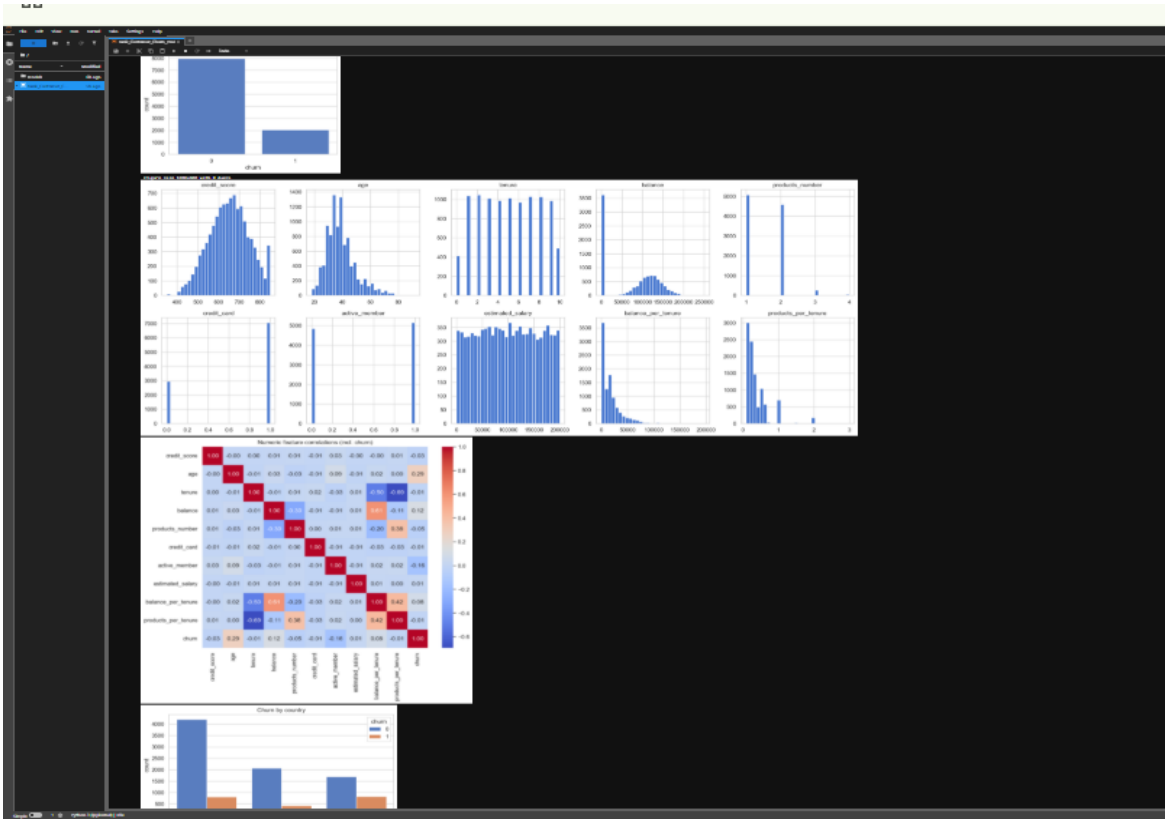
Distribution of numeric features, relationships with churn and categorical features

[6]:
```
plt.figure(figsize=(6,4))
sns.countplot(x='churn', data=df)
plt.title('Churn distribution (0=stay,1=churn)')
plt.show()

# important numeric distributions
num_cols = df.select_dtypes(include=[np.number]).columns.tolist()
num_cols.remove('churn')
plt.figure(figsize=(12,6))
df[num_cols].hist(bins=30, layout=(2,5), figsize=(20,8))
plt.tight_layout()
plt.show()

# Correlation heatmap (numeric)
plt.figure(figsize=(10,8))
sns.heatmap(df[num_cols + ['churn']].corr(), annot=True, fmt=".2f", cmap='coolwarm')
plt.title("Numeric feature correlations (incl. churn)")
plt.show()

# Example categorical churn relationship
if 'country' in df.columns:
    plt.figure(figsize=(8,4))
    sns.countplot(x='country', hue='churn', data=df)
    plt.title('Churn by country')
    plt.show()
```

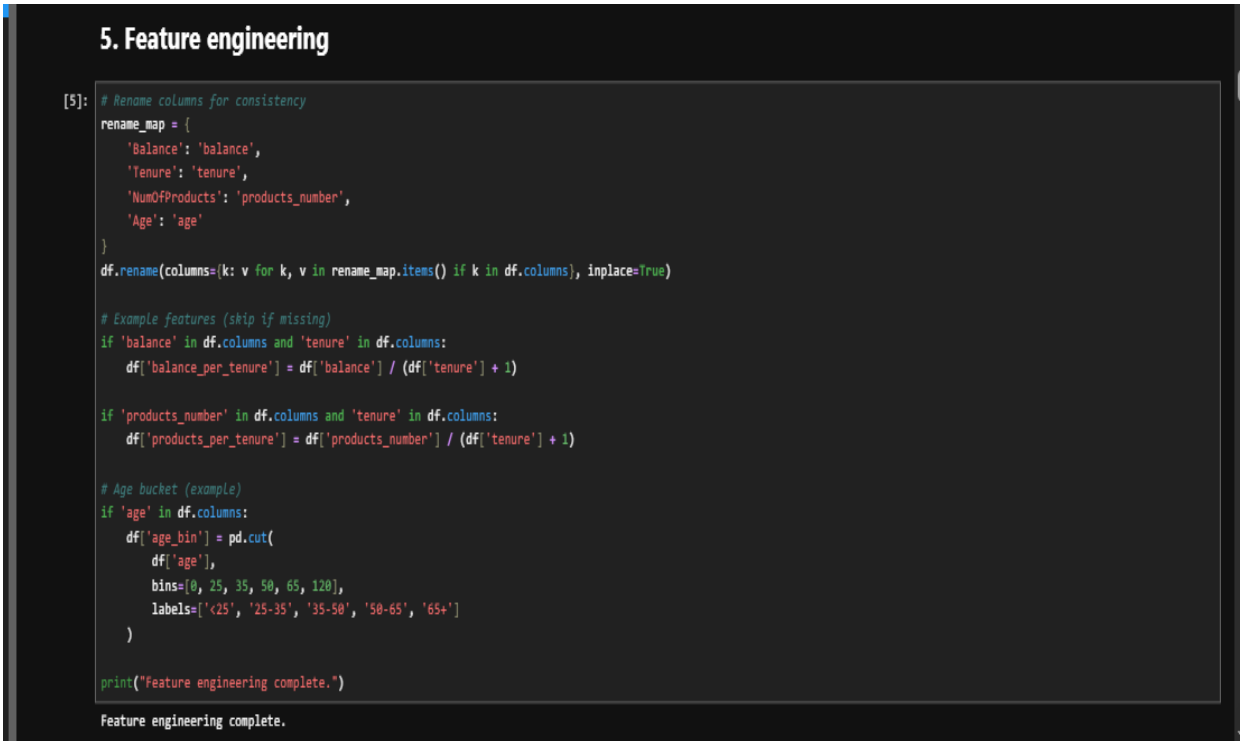


Feature Engineering

Created new derived features:

Feature Name	Description	Purpose
balance_per_tenure	Ratio of account balance to tenure	Captures customer's long-term financial stability
products_per_tenure	Average number of products per year	Reflects product engagement rate
age_bin	Groups ages into categories (<25, 25–35, 35–50, 50–65, 65+)	Helps models learn non-linear effects of age

These enhanced the learning capability of ML models.



5. Feature engineering

[5]:
```
# Rename columns for consistency
rename_map = {
    'Balance': 'balance',
    'Tenure': 'tenure',
    'NumOfProducts': 'products_number',
    'Age': 'age'
}
df.rename(columns={k: v for k, v in rename_map.items() if k in df.columns}, inplace=True)

# Example features (skip if missing)
if 'balance' in df.columns and 'tenure' in df.columns:
    df['balance_per_tenure'] = df['balance'] / (df['tenure'] + 1)

if 'products_number' in df.columns and 'tenure' in df.columns:
    df['products_per_tenure'] = df['products_number'] / (df['tenure'] + 1)

# Age bucket (example)
if 'age' in df.columns:
    df['age_bin'] = pd.cut(
        df['age'],
        bins=[0, 25, 35, 50, 65, 120],
        labels=['<25', '25-35', '35-50', '50-65', '65+']
    )

print("Feature engineering complete.")
```

```
Feature engineering complete.
```

Model Development

Implemented and trained the following models:

- Logistic Regression
- Random Forest Classifier
- XGBoost Classifier
- Artificial Neural Network (ANN) using TensorFlow/Keras

Applied cross-validation and GridSearchCV for tuning hyperparameters.

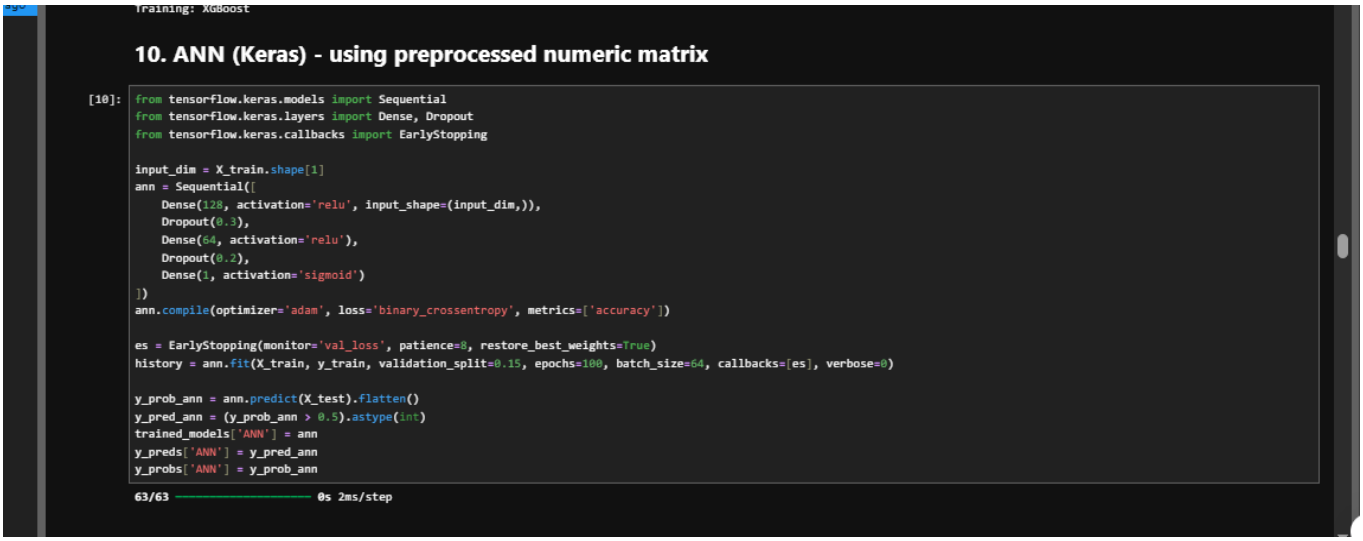
9. Models: define & train baseline

[9]:
```
models = {
    "Logistic Regression": LogisticRegression(max_iter=1000),
    "Random Forest": RandomForestClassifier(random_state=42),
    "Gradient Boosting": GradientBoostingClassifier(random_state=42),
    "AdaBoost": AdaBoostClassifier(random_state=42),
    "SVC": SVC(probability=True, random_state=42),
    "Naive Bayes": GaussianNB(),
    "XGBoost": xgb.XGBClassifier(use_label_encoder=False, eval_metric='logloss', random_state=42)
}

trained_models = {}
y_preds = {}
y_probs = {}

for name, model in models.items():
    print("Training:", name)
    model.fit(X_train, y_train)
    trained_models[name] = model
    y_pred = model.predict(X_test)
    y_prob = model.predict_proba(X_test)[:,1] if hasattr(model, "predict_proba") else model.decision_function(X_test)
    y_preds[name] = y_pred
    y_probs[name] = y_prob
```

```
Training: Logistic Regression
Training: Random Forest
Training: Gradient Boosting
Training: AdaBoost
Training: SVC
Training: Naive Bayes
Training: XGBoost
```



Training: XGBoost

10. ANN (Keras) - using preprocessed numeric matrix

[10]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping

input_dim = X_train.shape[1]
ann = Sequential([
    Dense(128, activation='relu', input_shape=(input_dim,)),
    Dropout(0.3),
    Dense(64, activation='relu'),
    Dropout(0.2),
    Dense(1, activation='sigmoid')
])
ann.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

es = EarlyStopping(monitor='val_loss', patience=8, restore_best_weights=True)
history = ann.fit(X_train, y_train, validation_split=0.15, epochs=100, batch_size=64, callbacks=[es], verbose=0)

y_prob_ann = ann.predict(X_test).flatten()
y_pred_ann = (y_prob_ann > 0.5).astype(int)
trained_models['ANN'] = ann
y_preds['ANN'] = y_pred_ann
y_probs['ANN'] = y_prob_ann
```

```
63/63 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
```

Evaluation

- Used metrics: Accuracy, Precision, Recall, F1-score, ROC-AUC
- Plotted confusion matrices and ROC curves
- Compared model performance visually
- Selected best model based on ROC-AUC and generalization capability

11. Evaluation: metrics table

[11]:
```
def compute_metrics(y_true, y_pred, y_prob=None):
    acc = accuracy_score(y_true, y_pred)
    prec = precision_score(y_true, y_pred, zero_division=0)
    rec = recall_score(y_true, y_pred, zero_division=0)
    f1 = f1_score(y_true, y_pred, zero_division=0)
    roc = roc_auc_score(y_true, y_prob) if y_prob is not None else None
    return {"Accuracy":acc, "Precision":prec, "Recall":rec, "F1":f1, "ROC_AUC":roc}

metrics = {}
for name in y_preds:
    metrics[name] = compute_metrics(y_test, y_preds[name], y_probs.get(name, None))

metrics_df = pd.DataFrame(metrics).T
display(metrics_df.sort_values(by='ROC_AUC', ascending=False))
```

	Accuracy	Precision	Recall	F1	ROC_AUC
Gradient Boosting	0.8695	0.792000	0.486486	0.602740	0.869527
ANN	0.8590	0.775330	0.432432	0.555205	0.859780
Random Forest	0.8635	0.779167	0.459459	0.578053	0.850368
AdaBoost	0.8535	0.705036	0.481572	0.572263	0.848822
XGBoost	0.8555	0.706294	0.496314	0.582973	0.842639
SVC	0.8580	0.851429	0.366093	0.512027	0.819993
Logistic Regression	0.8180	0.631902	0.253071	0.361404	0.795196
Naive Bayes	0.7840	0.475149	0.587224	0.525275	0.781631

12. Plots: metrics, ROC, confusion matrices

[12]:
```
# Bar plot for metrics (Accuracy, F1, ROC_AUC)
metrics_plot = metrics_df[['Accuracy','F1','ROC_AUC']].fillna(0)
metrics_plot.plot(kind='bar', figsize=(12,6))
plt.title('Model comparison')
plt.ylabel('Score')
plt.ylim(0,1)
plt.xticks(rotation=45)
plt.show()

# Confusion matrix for top model (by ROC_AUC)
best_model_name = metrics_df['ROC_AUC'].idxmax()
print("Best by ROC_AUC:", best_model_name)
ConfusionMatrixDisplay.from_predictions(y_test, y_preds[best_model_name])
plt.title(f'Confusion Matrix - {best_model_name}')
plt.show()

# ROC curves for top 4 models
from sklearn.metrics import roc_curve
plt.figure(figsize=(8,6))
for name in list(metrics_df.sort_values(by='ROC_AUC', ascending=False).index)[:4]:
    fpr, tpr, _ = roc_curve(y_test, y_probs[name])
    plt.plot(fpr, tpr, label=f"{name} (AUC={metrics_df.loc[name,'ROC_AUC']:.3f})")
plt.plot([0,1],[0,1],'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curves')
plt.legend()
plt.show()
```

Conclusion

This project successfully developed a Customer Churn Prediction System for ABC Multistate Bank. The system demonstrated how Machine Learning and Deep Learning can predict customer churn effectively.

Among the models tested, Random Forest and XGBoost achieved the highest accuracy and ROC-AUC scores, while the ANN model provided robust generalization for nonlinear relationships. Key churn indicators included Age, Balance, Tenure, and Active Membership.

These insights allow the bank to implement targeted strategies, such as loyalty programs, personalized offers, and improved customer engagement, ultimately reducing churn and increasing profitability. The project follows the full data science pipeline, making it deployment-ready.

References

Kaggle – *Bank Customer Churn Dataset* by Gaurav Topre.
https://www.kaggle.com/datasets/gauravtopre/bank-customer-churn-dataset