XYZ BANK Customer Churn Prediction

Capstone Project

Student: Lei Liu

Problem Statement

- Customer relation is key for banking success. XYZ Multi-national Bank is experiencing 21% churn rate on average.
- XYZ Bank is keen on retaining its account holders. How can we help XYZ Bank to reduce churn rate?
- SOLUTION:
  - ✓ By leveraging customer data, employing machine learning algorithms to identify patterns and predict the likelihood of customer churn.
  - ✓ These predictive insights empower the XYZ Bank CRM team to proactively engage high-risk customers and implement targeted retention strategies to effectively reduce churn.

Data Source & Meta Data

- Data source: Binary Classification with a Bank Churn Dataset https://www.kaggle.com/competitions/playground-series-s4e1/data
- Metadata:
  - 165,034 rows, 13 features, 1 target.
  - Target: “Exited” column: binary value indicating if the customer churn(1) or not churn(0)
  - 13 Features:
    - 2 services features : “NumOfProducts” (bank products that customer subscribe), “HasCrCard”(whether having bank credit card).
    - 3 customer demographic features : “Geography”, ”Gender”, ”Age”.
    - 3 account features : “Tenure” (years of holding bank account), “Balance” (bank acct balance), “IsActiveMember”(whether account has active transactions).

Machine Learning(ML) Process

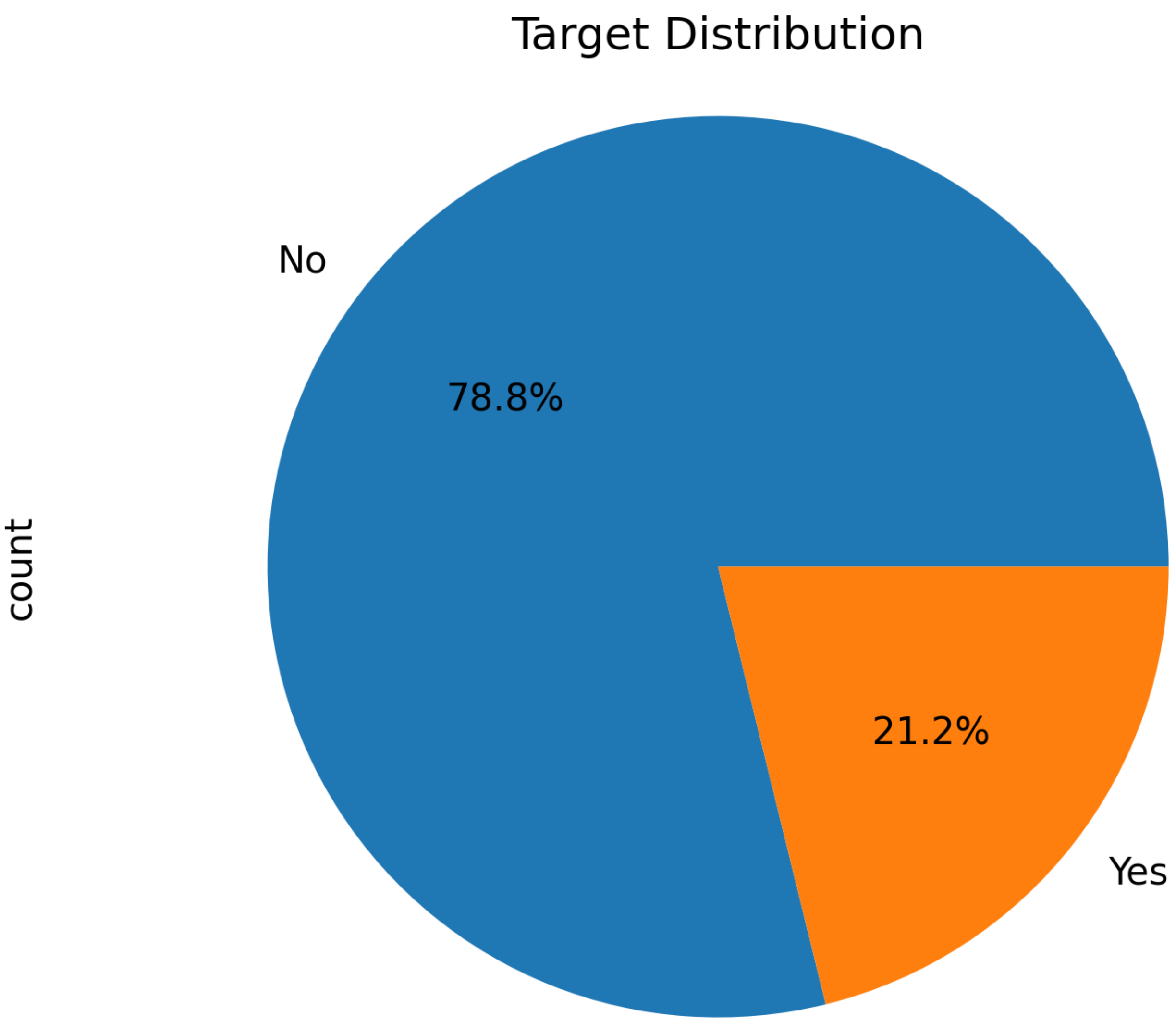
- Data collection and data wrangling
- Exploratory Data Analysis (EDA): to explore the impact of features on the target(‘Exited’ / churn)
- Data pre-processing before modeling: data split, OneHotEncode all categorical features, and standardize all numeric features.
- Implement two ML algorithms
  - Model 1 Random Forest
  - Model 2 Logistic Regression
- Evaluate the two ML models using following metrics:
  - Recall
  - F1-score
  - ROC AUC score
  - Confusion Matrix

Exploratory Data Analysis (EDA)

- EDA was performed on three features ‘Age’, ‘Tenure’, and ‘Gender’ to investigate their relationship with customer churn (‘Exited’). This analysis revealed patterns and potential indicators of churn behavior.
- Performed EDA on target (‘Exited’) distribution.

EDA – Target distribution

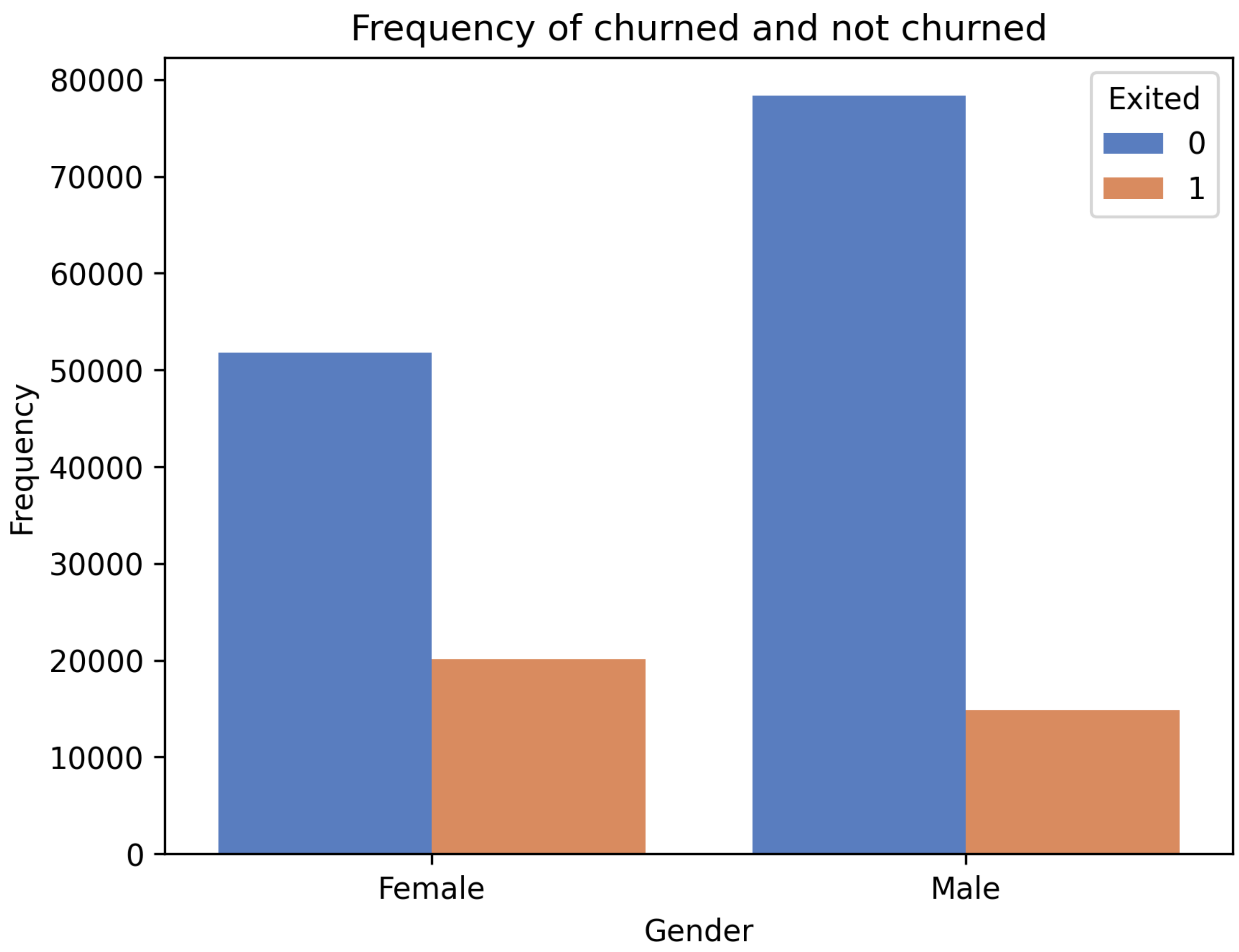
- An imbalanced target, with 130113 non-churners (78.8%) and 34921 churners (21.2%)
- Binary values – suggests machine learning needs to solve a binary classification problem



EDA –
Gender Impact

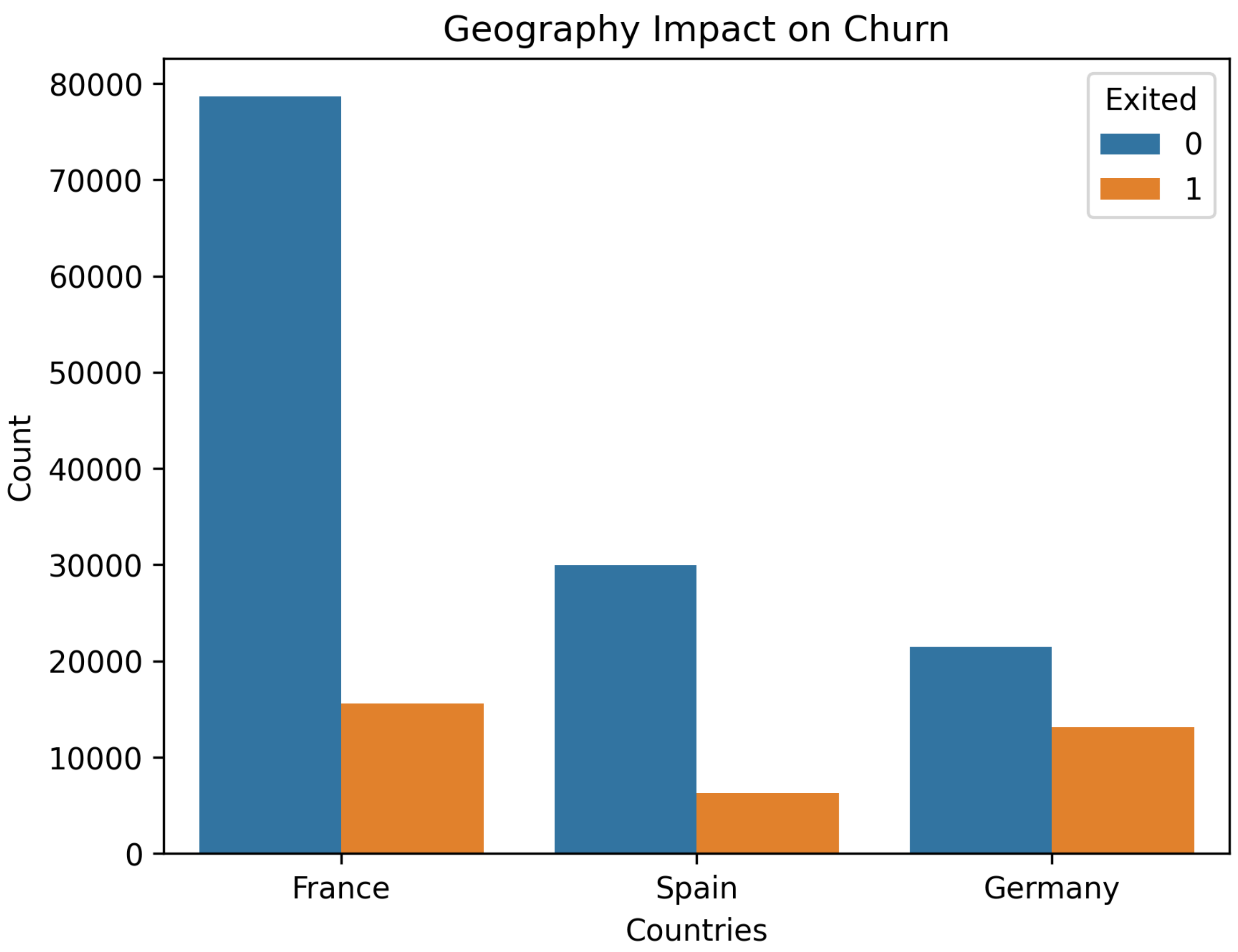
- The number of male account holders is greater than that of female account holders.

- Churn rate of female customers is higher than that of male customers.



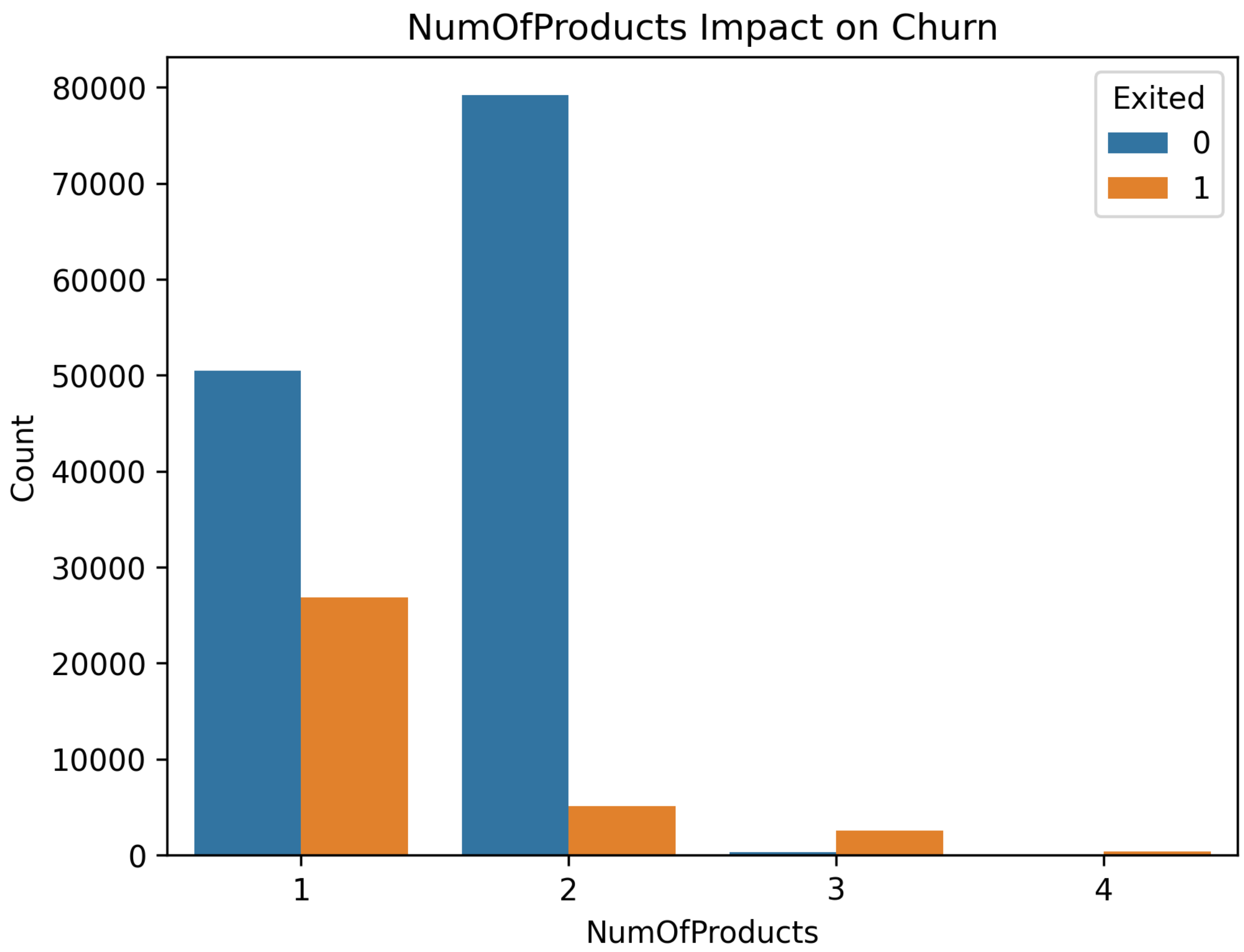
EDA – Geography Impact

- Among the three countries, Germany has significantly fewer total account holders but the highest churn rate, while France is the opposite.



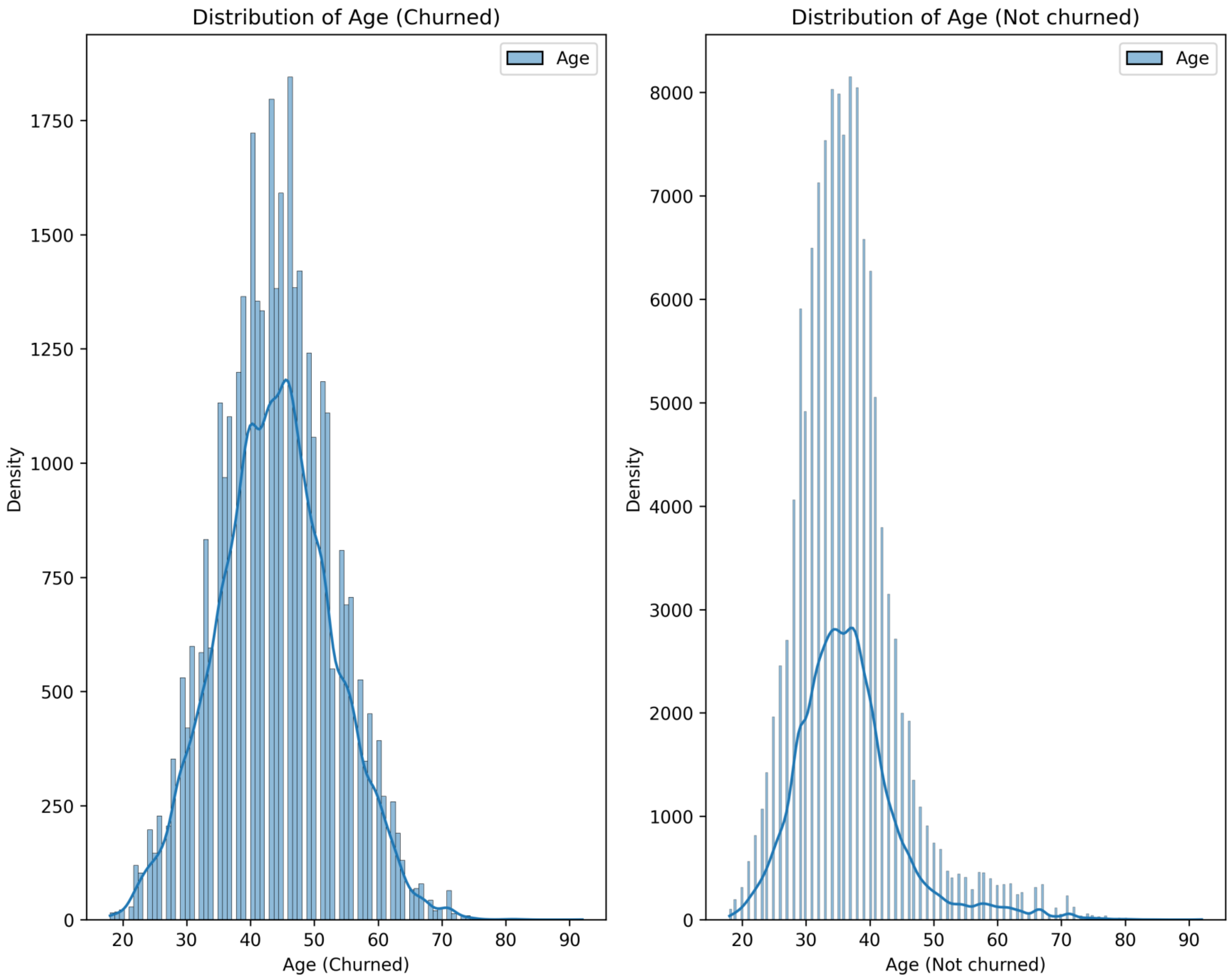
EDA – NumOfProducts Impact

- Account holders with 2 or fewer than 2 of bank products have significant higher churn rate than the others.



EDA – Age Impact

- Analysis of customer churn by age group reveals
  that customers aged 40 to 50 exhibit the highest churn rate.
- Conversely, the 30 to 40 age group demonstrates the highest non-churn density, indicating a lower propensity to churn compared to other segments.



Classification Modeling

- Column transformation:
  - Standardize all the numeric features
  - One Hot Encode all the categorical features
- Grid search cross-validation and hyperparameter tuning
- Model training with two Machine Learning Classification Models
  - Random Forest
  - Logistic Regression
- Model evaluation
  - Recall
  - f1-score
  - ROC AUC score
  - confusion matrix

Column Transformation

- Partition data into distinct training and testing sets to ensure robust model evaluation.
- Apply different pipelines for feature transform. OneHotEncoder for numerical features, StandardScaler for categorical features.
- ColumnTransformer was used to combine numerical and categorical transformers.

Data Preprocessing

Split data into training and testing datasets

```
[53]: X = train.drop('Exited',axis=1)
y = train.Exited
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y, random_state=42)
```

Define separate preprocessing pipelines for both feature types

```
]:
# normalize all numerical features
num_transformer = Pipeline(steps=[
    ('scaler', StandardScaler())
])

# OneHotEncode all categorical features
cat_transformer = Pipeline(steps=[
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])
```

Combine the transformers into a single column transformer

```
[63]: preprocessor = ColumnTransformer(
    transformers=[
        ('num', num_transformer, num_features),
        ('cat', cat_transformer, cat_features)
    ])
```

Model 1 – Random Forest

- Create model pipeline.
- Define a parameter grid, perform grid search cross-validation.
- Fit the best model to the training dataset.

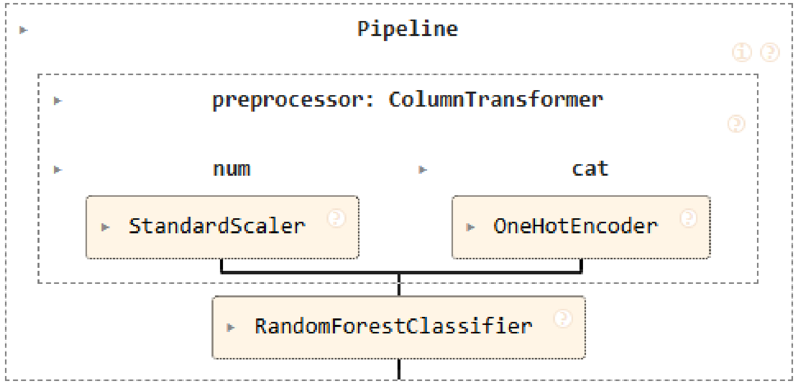
Model 1 Random Forest

Create a model pipeline

[68]:

```
pipe_rf = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', RandomForestClassifier(random_state=42))
])
pipe_rf
```

[68]:



Define a parameter grid

Use the grid in a cross validation search to optimize the model

[72]:

```
param_grid_rf = {
    'classifier__n_estimators': [50, 100],
    'classifier__max_depth': [None, 10, 20],
    'classifier__min_samples_split': [2, 5]
}
```

Perform grid search cross-validation and fit the best model to the training data

[75]:

```
cv = StratifiedKFold(n_splits=3, shuffle=True)
```

Train the pipeline model

[78]:

```
rf_model = GridSearchCV(estimator=pipe_rf, param_grid=param_grid_rf, cv=cv, scoring='accuracy', verbose=2)
rf_model.fit(X_train, y_train)
```

Model 2 – Logistic Regression

Model 2 Logistic Regression

Create a model pipeline and define a hyperparameter grid

[106]:
```
pipe_lr = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression(random_state=42))
])

param_grid_lr = {
    'classifier__solver': ['liblinear'],
    'classifier__penalty': ['l1', 'l2'],
    'classifier__class_weight': [None, 'balanced']
}
```

Use the grid in a cross validation search to optimize the model

Perform grid search cross-validation and fit the best model to the training data

[114]:
```
lr_model = GridSearchCV(estimator=pipe_lr, param_grid=param_grid_lr, cv=cv, scoring='accuracy', verbose=2)

# Train the pipeline model
lr_model.fit(X_train, y_train)
```

Evaluation of Model Performance: Recall and F1-score

Classification Report for Model 1 Random Forest

```
[81]: y_pred_rf = rf_model.predict(X_test)
      print(classification_report(y_test, y_pred_rf))
```

	precision	recall	f1-score	support
No	0.88	0.96	0.92	26023
Yes	0.77	0.50	0.61	6984
accuracy			0.86	33007
macro avg	0.82	0.73	0.76	33007
weighted avg	0.85	0.86	0.85	33007

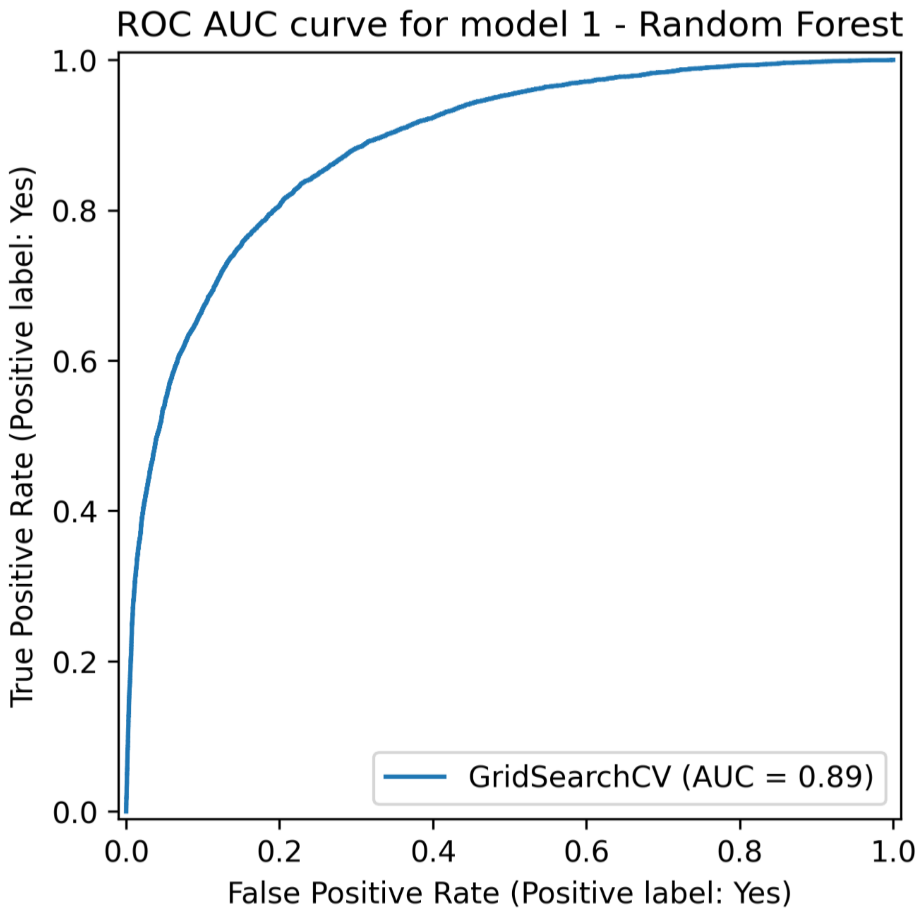
Classification Report for Model 2 Logistic Regression

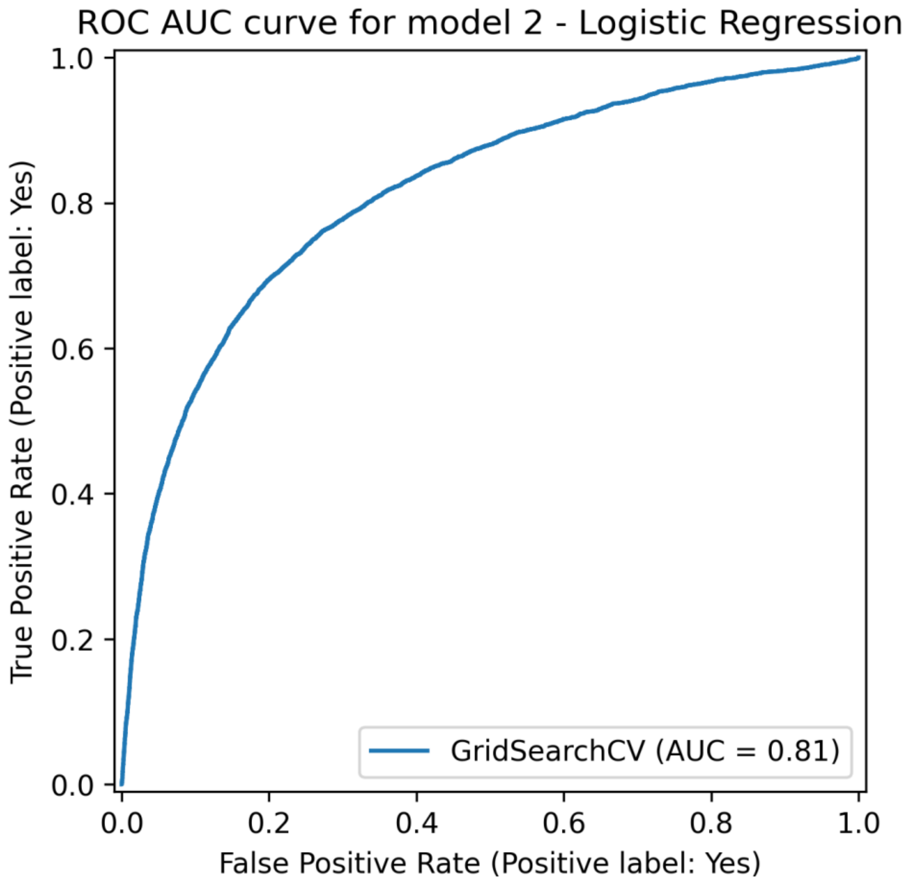
```
[116]: y_pred_lr = lr_model.predict(X_test)
       print(classification_report(y_test, y_pred_lr))
```

	precision	recall	f1-score	support
No	0.85	0.95	0.90	26023
Yes	0.69	0.38	0.49	6984
accuracy			0.83	33007
macro avg	0.77	0.67	0.70	33007
weighted avg	0.82	0.83	0.81	33007

- With a focus on identifying potential churners, where missing a churner carries a higher cost than misclassifying a non-churner, the recall metric is prioritized. The Random Forest model excels in this regard, achieving a recall of 0.50, which is notably higher than Logistic Regression's 0.38.
- This improved ability to detect actual churners is further supported by Random Forest's superior F1-score of 0.61, indicating a more balanced performance compared to Logistic Regression's 0.49.

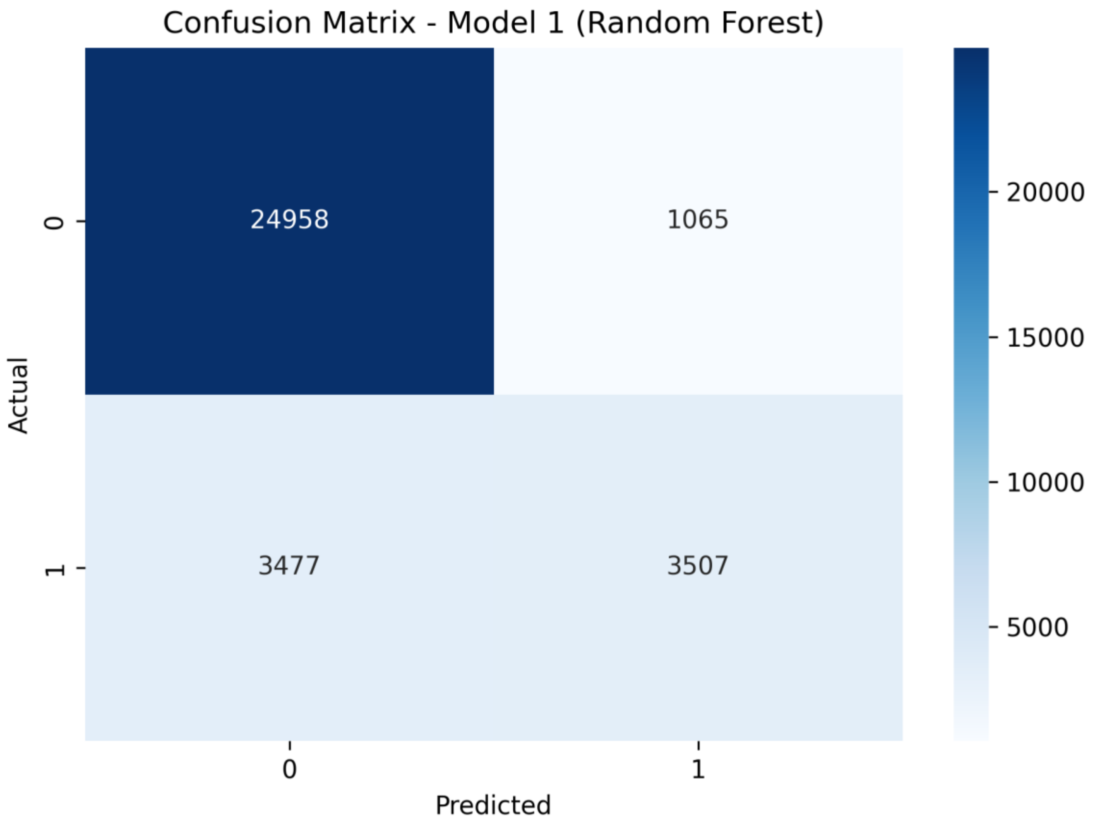
Evaluation of Model Performance: ROC AUC

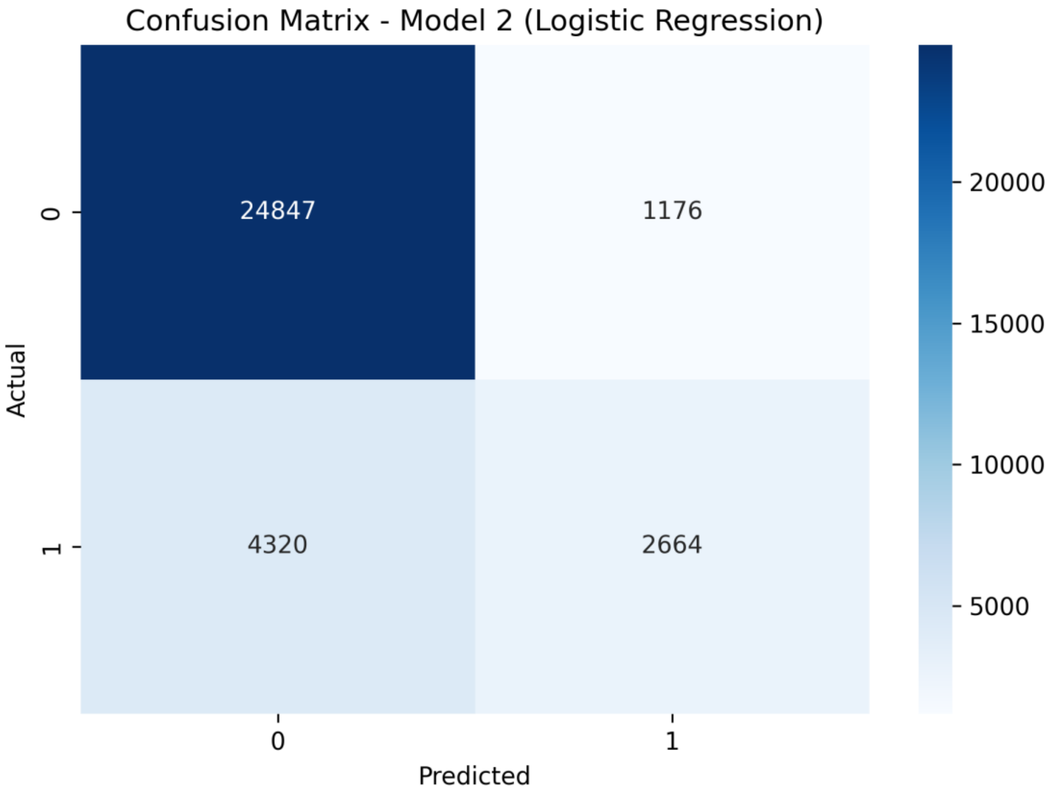




- Random Forest's ROC AUC score is 0.89, higher than that of Logistic Regression (0.81).

Evaluation of Model Performance: confusion matrix





- Model 1 (Random Forest) identifies churners more accurate than model 2 (Logistic Regression) does, with 3477 cases missed, which is 20% less than that missed with Logistic Regression model (4320 cases).

Evaluation of Model Performance: conclusion

- Based on evaluation, the Random Forest model has demonstrated superior performance compared to Logistic Regression for churn prediction with this dataset. This model effectively increases the churn prediction rate from a baseline of 21% to 50% (recall), providing valuable insights for the CRM department to enhance client retention programs.

Conclusion

- This project successfully employed and compared Random Forest and Logistic Regression algorithms for predicting bank customer churn. The Random Forest model demonstrated superior performance across key metrics – recall, F1-score, and ROC AUC – as evidenced by the confusion matrix.

- This success underscores the effectiveness of machine learning for customer segmentation and mitigating churn. By identifying at-risk customers proactively, financial institutions like XYZ Bank can deploy targeted retention strategies to minimize revenue loss.