Bank Customer Churn Analysis & Prediction: (Data Splitting and Modeling)

- **Prepared By:** Hassan Waked –Team Leader, DEPI Graduation Project.

- **Date:** 20th Dec 2024.

1. Data Splitting:
The dataset was split into training and testing sets using a **80% / 20% ratio**.

- **80%** of the data was used for training the model.
- **20%** was held out for testing and evaluating performance.

This ensures the model is evaluated on unseen data to measure generalization.

---

2. Data Balancing:

2.1 Problem:
The original training dataset was imbalanced:

- Class 0 (Not churned): 6367 samples
- Class 1 (Churned): 1629 samples

This imbalance could bias the model toward predicting the majority class.

2.2 Technique Used:
To address this, the **SMOTE (Synthetic Minority Over-sampling Technique)** algorithm was applied using imblearn. SMOTE generates synthetic examples of the minority class to achieve balance.

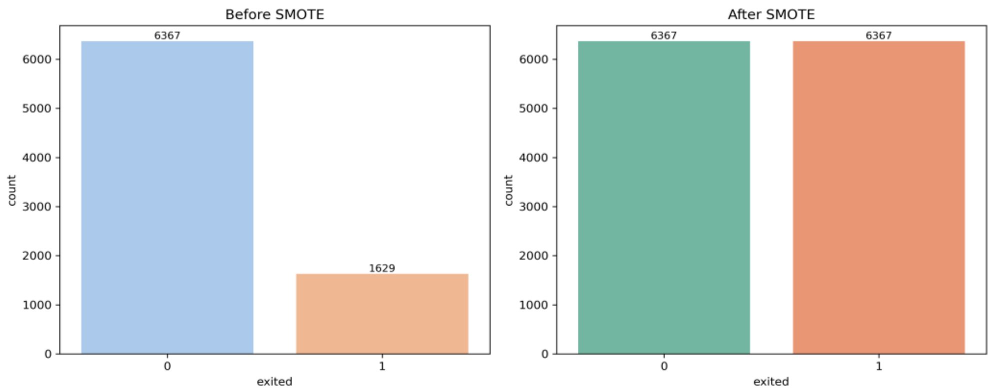
```
X_resampled, y_resampled = SMOTE(random_state=42).fit_resample(X_train, y_train)
```

2.3 Result of Resampling:
After applying SMOTE:

- Both classes were balanced with **6367 samples each**.

- Total training data increased from **7996** to **12734** rows.

2.4 Visualization:



3.1 Feature Scaling Purpose:

Feature scaling is essential to ensure all features contribute equally, especially for algorithms sensitive to feature magnitude like **SVM** and **KNN**.

3.2 Standardization Method:

StandardScaler was used to transform features to have a **mean of 0** and **standard deviation of 1**.

3.4 Applying the Scaler:

- fit_transform() was applied to the training set (X_resampled) to calculate and apply scaling.
- transform() was applied to the test set (X_test) using the same statistics, avoiding data leakage.

3.5 Saving the Scaler:

The trained scaler was saved to disk using joblib at the path:
../models/standard_scaler.pkl

---

4. Modeling:

4.1 Models Used:

Seven classifiers were defined and trained:

- Logistic Regression
- Decision Tree
- Random Forest
- Support Vector Machine (SVM)
- K-Nearest Neighbors (KNN)
- Naive Bayes
- XGBoost

Each model was trained using the balanced and scaled data: X_resampled_scaled, y_resampled.

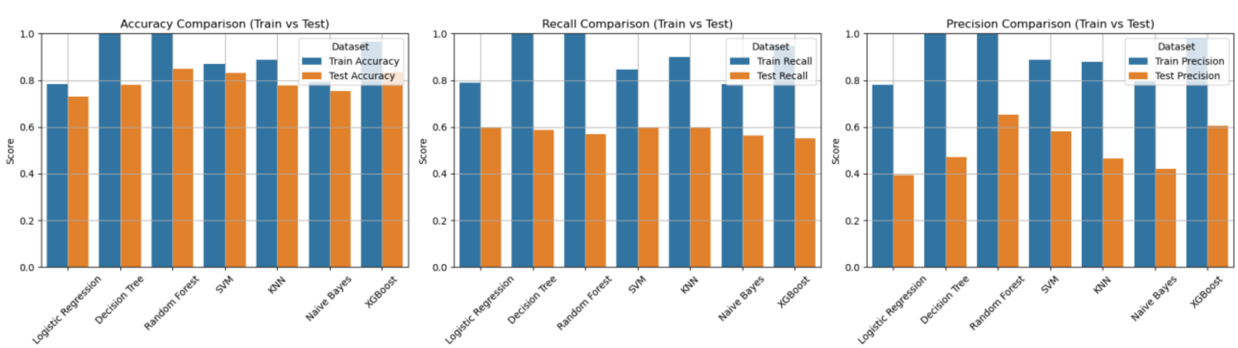
---

5. Evaluation Before Tuning:

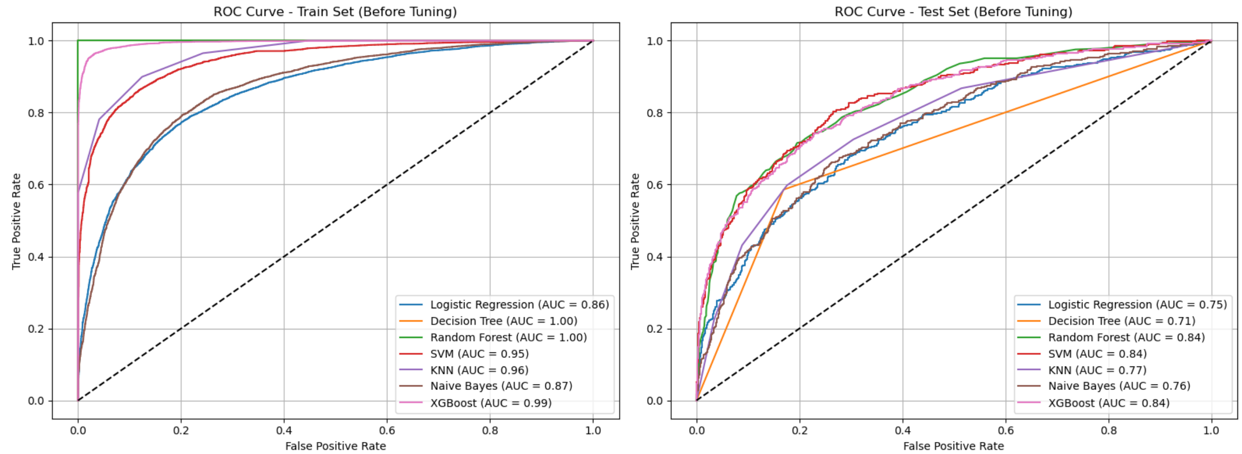
5.1 Model_performance_summary:

	Model	Train Accuracy	Test Accuracy	Train Recall	Test Recall	Train Precision	Test Precision	Train AUC	Test AUC
0	Logistic Regression	0.785	0.731	0.791	0.598	0.782	0.395	0.862	0.752
1	Decision Tree	1.0	0.781	1.0	0.586	1.0	0.47	1.0	0.708
2	Random Forest	1.0	0.85	1.0	0.569	1.0	0.652	1.0	0.84
3	SVM	0.87	0.83	0.846	0.598	0.889	0.58	0.946	0.841
4	KNN	0.888	0.778	0.899	0.598	0.879	0.466	0.962	0.772
5	Naive Bayes	0.794	0.753	0.785	0.564	0.799	0.421	0.869	0.759
6	XGBoost	0.965	0.835	0.948	0.551	0.98	0.605	0.995	0.835

5.2 Model Performance Comparison (Train vs Test):



5.3 ROC Curve Visualization for Models:



5.4 Top Models Selection for Hyperparameter Tuning:

	Model	Train Accuracy	Test Accuracy	Train Recall	Test Recall	Train Precision	Test Precision	Train AUC	Test AUC
3	SVM	0.87	0.83	0.846	0.598	0.889	0.58	0.946	0.841
2	Random Forest	1.0	0.85	1.0	0.569	1.0	0.652	1.0	0.84
6	XGBoost	0.965	0.835	0.948	0.551	0.98	0.605	0.995	0.835

5.5 Conclusion: Top Models Selected for Tuning:

Based on **Test AUC** as the key evaluation metric, the following three models were shortlisted for hyperparameter tuning:

Top 3 Models (Test AUC Scores):

- **SVM**: 0.841
- **Random Forest**: 0.840
- **XGBoost**: 0.835

Key Insights:

- **SVM** demonstrates the most balanced performance between training and testing sets.
- **Random Forest** shows potential overfitting, with perfect training scores but a performance drop on testing.
- **XGBoost** delivers strong training results and remains competitive on the test set.

.

6. Hyperparameter Tuning:

6.1 Algorithm: XGBoost
Best Parameters:

- n_estimators: 100
- max_depth: 5
- learning_rate: 0.1
- subsample: 1.0
  Selected using RandomizedSearchCV with 2-fold cross-validation, focusing on recall.

6.2 Algorithm: SVM
Best Parameters:

- C: 1.0
- kernel: 'rbf'
- gamma: 'scale'
  Tuned to maximize recall while maintaining generalization. SVM showed the most balanced performance between training and test sets.

6.3 Algorithm: Random Forest
Best Parameters:

- n_estimators: 200
- max_depth: 10
- min_samples_split: 2
- min_samples_leaf: 1
  Tuned to reduce overfitting observed in the default model. Achieved strong performance on test data with improved recall.

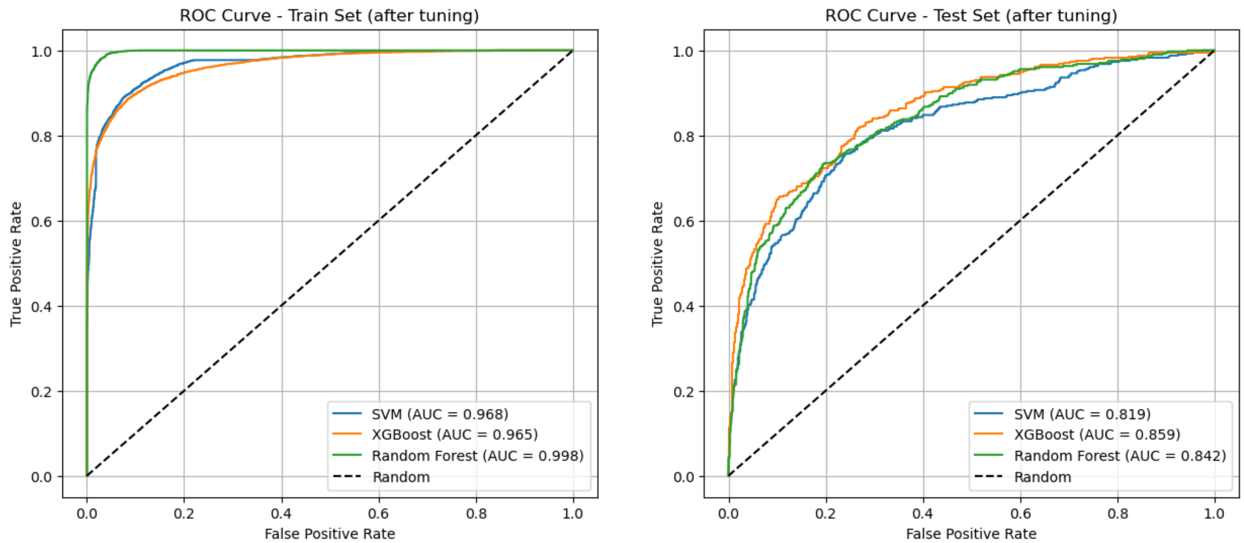
---

7. Evaluation After Tuning:

7.1 Model_performance_summary:

	Model	Train Accuracy	Test Accuracy	Train Recall	Test Recall	Train Precision	Test Precision	Train AUC	Test AUC
0	SVM	0.907	0.823	0.891	0.566	0.919	0.566	0.968	0.819
1	XGBoost	0.899	0.85	0.876	0.62	0.919	0.634	0.965	0.859
2	Random Forest	0.972	0.843	0.959	0.571	0.984	0.626	0.998	0.842

7.2 ROC Curve Visualization for Models:



7.3 Conclusion:

- **XGBoost** showed the best **overall performance**, balancing recall, speed, and generalization.
- **Random Forest** achieved the **highest test recall**, but suffered from **overfitting**.
- **SVM** had excellent training results but **poor generalization** on test data.

Preferred Model: **XGBoost** — for its stability, competitive recall, and robustness across datasets