Optimizing Bank Marketing Campaigns Using Regularized Ensemble Learning Techniques

Problem Statement :

This project aims to develop a predictive model to determine whether a bank client will subscribe to a term deposit based on demographic, financial, and campaign-related factors. Using numerical optimization and machine learning techniques, the goal is to identify key drivers influencing client decisions and enhance the effectiveness of future marketing campaigns.

Dataset :

The data starter to the project is from the Bank Marketing Dataset, from the UCI Machine Learning repository, which includes data gathered from various marketing campaigns that took place for a Portuguese bank. Bank marketing efforts included contacting clients via phone in order to solicit users to subscribe to a term deposit product. The data collection serves as a real world example for optimizing prediction analysis based on a mix of categorical and numeric features, as well as a binary outcome variable.

The data set contains approximately 45,000 records and 17 attributes which capture various demographic, financial, and campaign information for each client. The outcome variable (y) indicates if the client subscribed to a term deposit ("yes" or "no").

Key Variables Include:

- Demographic Attributes: age, job, marital, and education.
- Financial Attributes: balance (average yearly balance), housing, default and loan.
- Campaign Attributes: contact type, day, month, duration, campaign, pdays, previous, and poutcome (previous campaign outcome).
- Target Variable: y – indicates if the client subscribed to a term deposit.

Data Preprocessing :

1. Data Cleaning

- Verified no missing values.
- dentified “unknown” entries in categorical columns (treated as a valid category, not dropped).

2. Encoding Categorical Variables

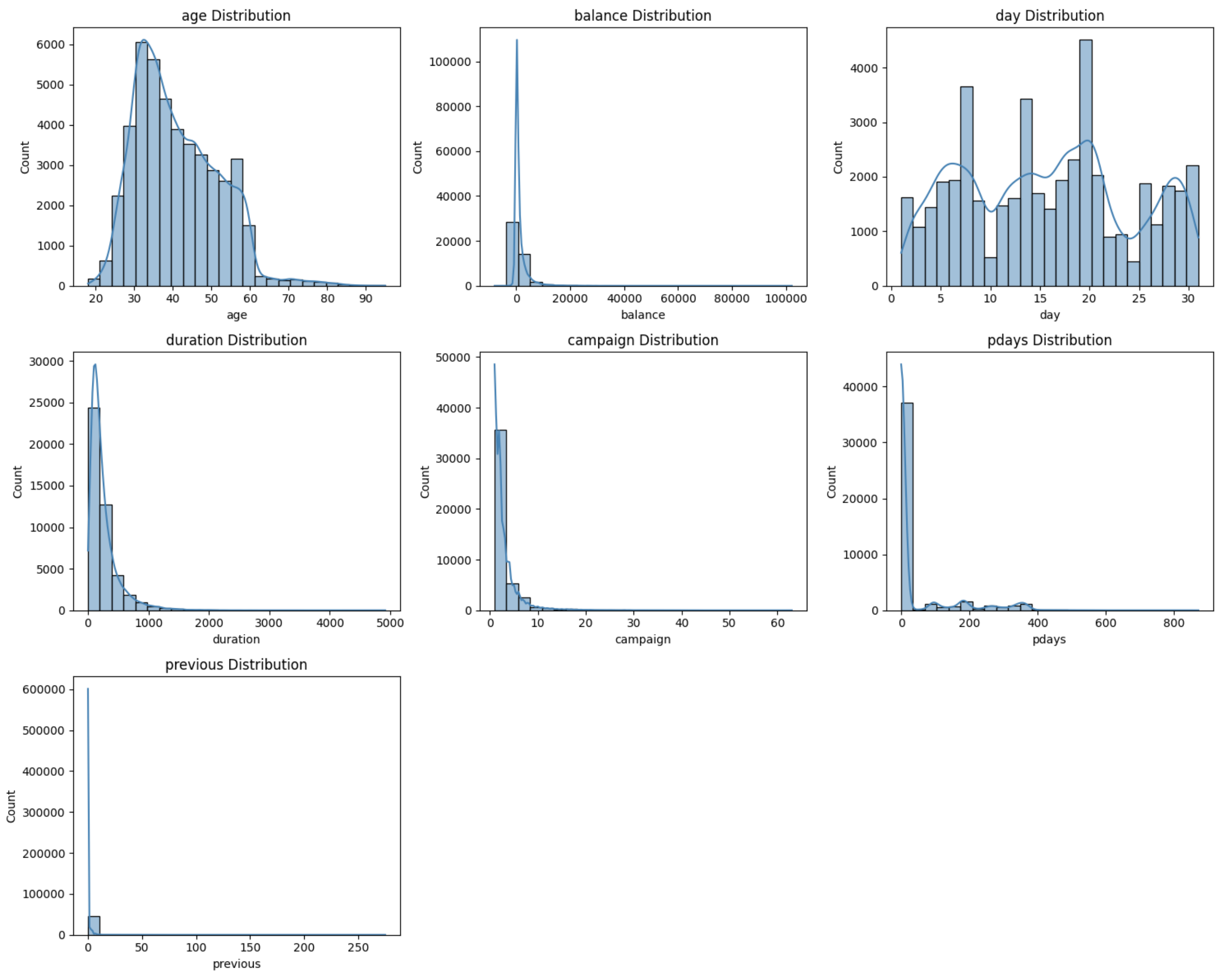
Applied multiple encoding techniques based on variable type

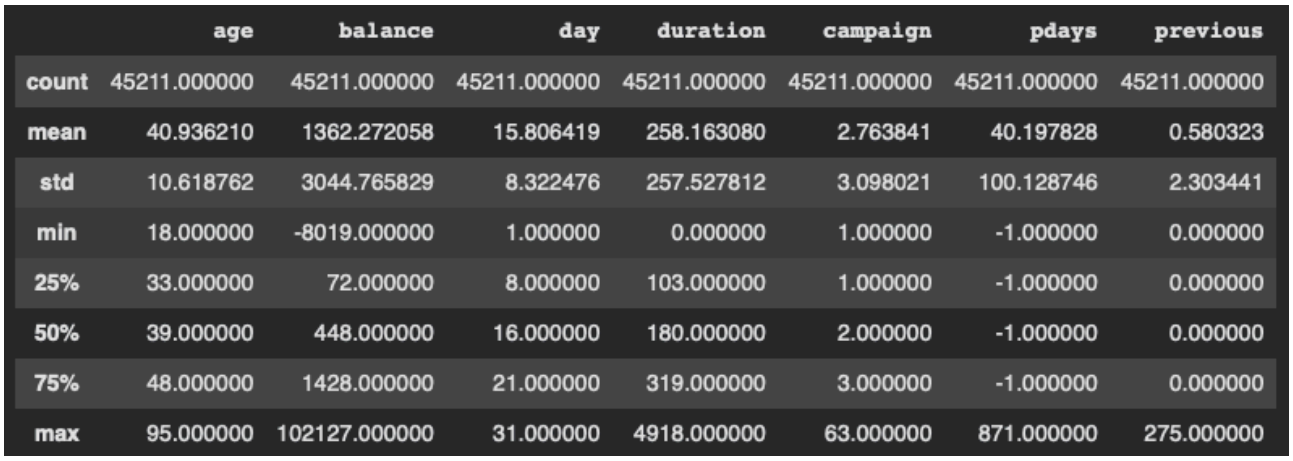
- **Binary Mapping:** Converted yes/no variables (default, housing, loan, y) into 1/0 for model compatibility.
- **Label Encoding:** Used for ordinal features like education (primary < secondary < tertiary).
- **One-Hot Encoding:** Applied to nominal variables (job, marital, contact, poutcome) to create separate indicator columns and avoid false order.

Exploratory Data Analysis :

- Univariate Analysis
- Bivariate Analysis
- Target Variable Analysis
- Correlation

Distributions of Numeric Variables :

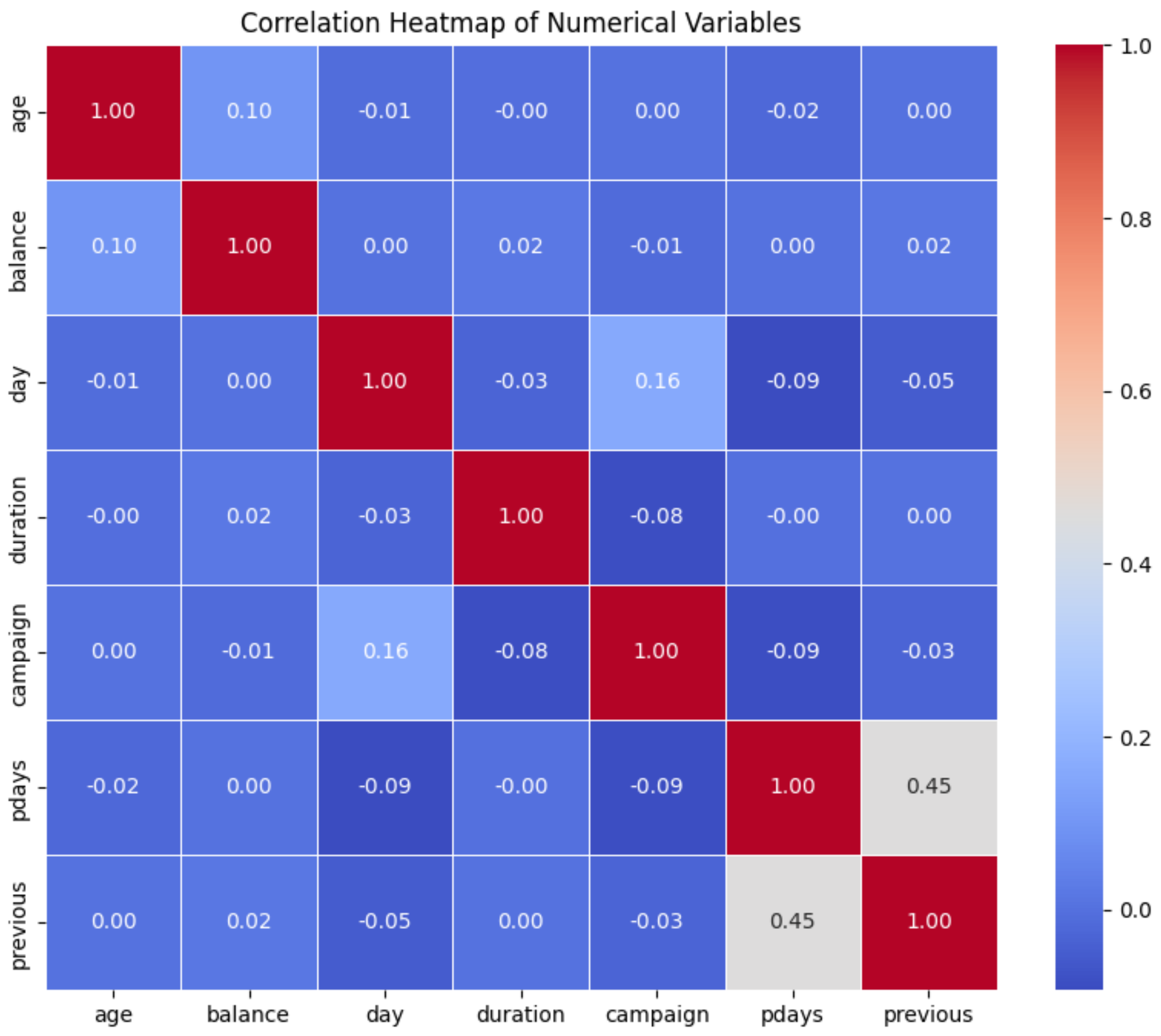




	age	balance	day	duration	campaign	pdays	previous
count	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000	45211.000000
mean	40.936210	1362.272058	15.806419	258.163080	2.763841	40.197828	0.580323
std	10.618762	3044.765829	8.322476	257.527812	3.098021	100.128746	2.303441
min	18.000000	-8019.000000	1.000000	0.000000	1.000000	-1.000000	0.000000
25%	33.000000	72.000000	8.000000	103.000000	1.000000	-1.000000	0.000000
50%	39.000000	448.000000	16.000000	180.000000	2.000000	-1.000000	0.000000
75%	48.000000	1428.000000	21.000000	319.000000	3.000000	-1.000000	0.000000
max	95.000000	102127.000000	31.000000	4918.000000	63.000000	871.000000	275.000000

- Skewed Distributions
- Outliers

Correlation Heatmap :

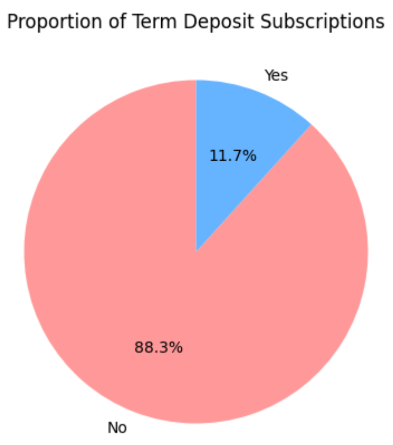


	age	balance	day	duration	campaign	pdays	previous
age	1.00	0.10	-0.01	-0.00	0.00	-0.02	0.00
balance	0.10	1.00	0.00	0.02	-0.01	0.00	0.02
day	-0.01	0.00	1.00	-0.03	0.16	-0.09	-0.05
duration	-0.00	0.02	-0.03	1.00	-0.08	-0.00	0.00
campaign	0.00	-0.01	0.16	-0.08	1.00	-0.09	-0.03
pdays	-0.02	0.00	-0.09	-0.00	-0.09	1.00	0.45
previous	0.00	0.02	-0.05	0.00	-0.03	0.45	1.00

- The non-linear and weakly correlated structure suggests that linear models may capture only part of the variance; tree-based models could better exploit hidden interactions.
- The independent spread of points ensures a well-conditioned feature matrix, minimizing multicollinearity during numerical optimization.
- duration and balance remain visually dominant predictors, while campaign, pdays, and previous provide behavioral nuance but limited direct separability.

Target Variable Analysis :

- The target variable y indicates whether a client subscribed to a term deposit:
  - Yes (1): Client subscribed
  - No (0): Client did not subscribe
- Distribution is highly imbalanced:
  - No = 88%, Yes = 12%
- Implies that the dataset is dominated by negative cases (non-subscribers).
- This imbalance can bias models toward predicting “No,” so it was handled later through SMOTE and class weighting.
- Stratified Data Split
- Evaluating models using metrics robust to imbalance (ROC-AUC, Confusion Matrix, F1-score).



- The entropy of y was 0.521 bits, confirming low diversity in outcomes.

$$H(Y) = -[p_1 log_2(p_1) + p_0 log_2(p_0)]$$

$$H(Y) = -[0.12 log_2(0.12) + 0.88 log_2 (0.88)$$

$$= 0.521 \text{ bits}$$

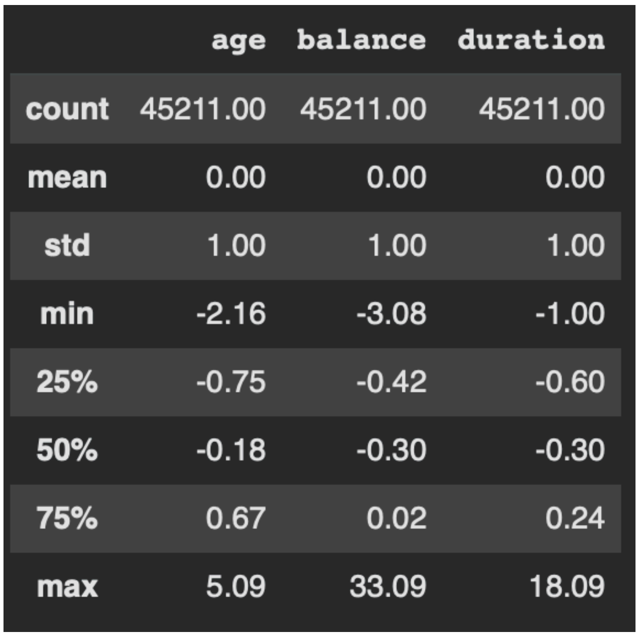
Data Preparation for Modeling :

Standardization :

The data standardization (z-score normalization) formula is:

$$z = \frac{x_i - \mu}{\sigma}$$

- This ensures all numeric features (age, balance, duration, etc.) are on a comparable scale, improving model stability and convergence.
- Standardization centers the data around 0 mean and scales it to have unit variance (1).



	age	balance	duration
count	45211.00	45211.00	45211.00
mean	0.00	0.00	0.00
std	1.00	1.00	1.00
min	-2.16	-3.08	-1.00
25%	-0.75	-0.42	-0.60
50%	-0.18	-0.30	-0.30
75%	0.67	0.02	0.24
max	5.09	33.09	18.09

Log Transformation :

X = log (x + c)

X = Transformed Value
x = Original value
c = a small constant added to handle zero or negative value (commonly c = 1)

- Used to reduce right-skewness and make data more normally distributed.
- Compresses large values and expands smaller ones, improving model performance when data spans multiple scales.

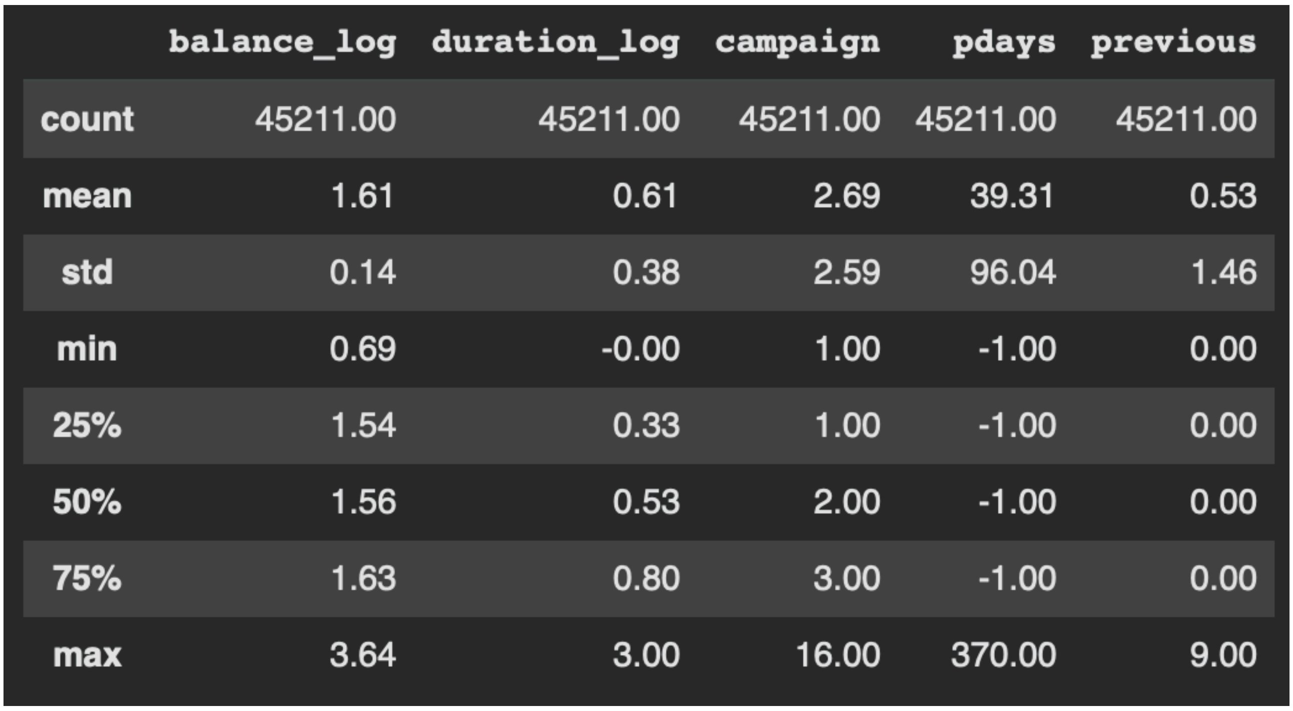
Winsorization :

$$x = \begin{cases} L, & \text{if } x < L \\ x, & \text{if } L \leq x \leq U \\ U, & \text{if } x > U \end{cases}$$

where:

- x → winsorized value
- x → original value
- L → lower percentile cutoff (e.g., 5th percentile)
- U → upper percentile cutoff (e.g., 95th percentile)

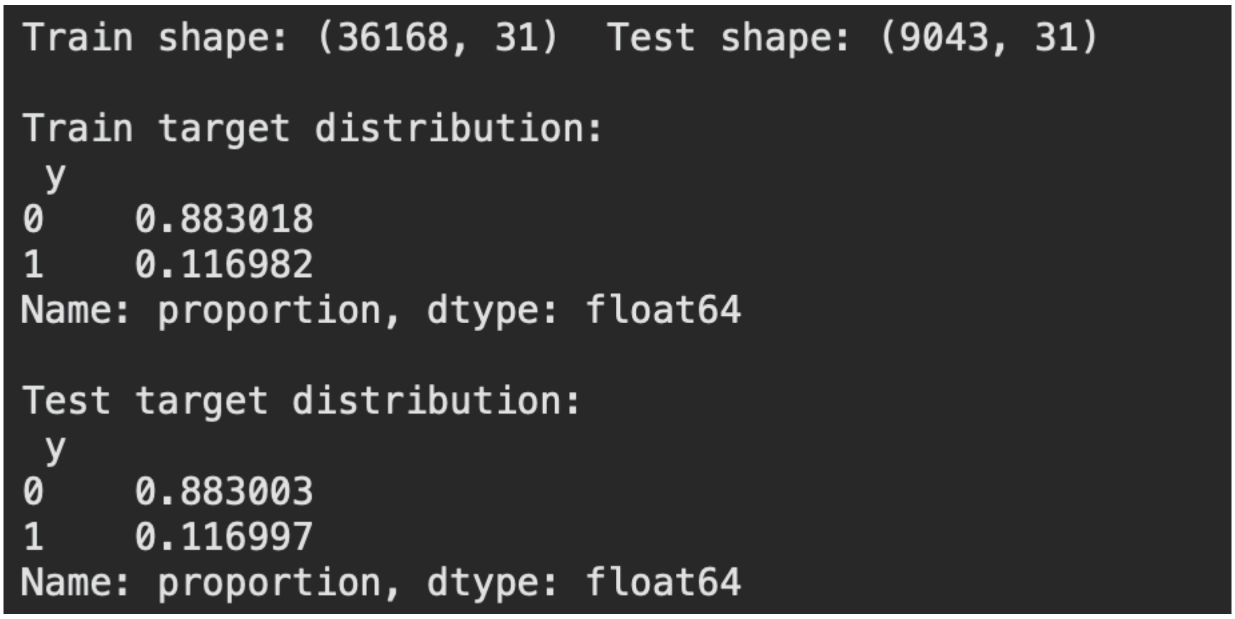
- Winsorization limits extreme values to reduce the impact of outliers without removing data points.
- If 5th and 95th percentiles are chosen, all values below the 5th percentile are set to that percentile value, and all above the 95th are capped there.
- It's useful when variables like balance or duration contain a few extreme outliers that could distort model training.



	balance_log	duration_log	campaign	pdays	previous
count	45211.00	45211.00	45211.00	45211.00	45211.00
mean	1.61	0.61	2.69	39.31	0.53
std	0.14	0.38	2.59	96.04	1.46
min	0.69	-0.00	1.00	-1.00	0.00
25%	1.54	0.33	1.00	-1.00	0.00
50%	1.56	0.53	2.00	-1.00	0.00
75%	1.63	0.80	3.00	-1.00	0.00
max	3.64	3.00	16.00	370.00	9.00

Train-test split (80-20, stratified) :

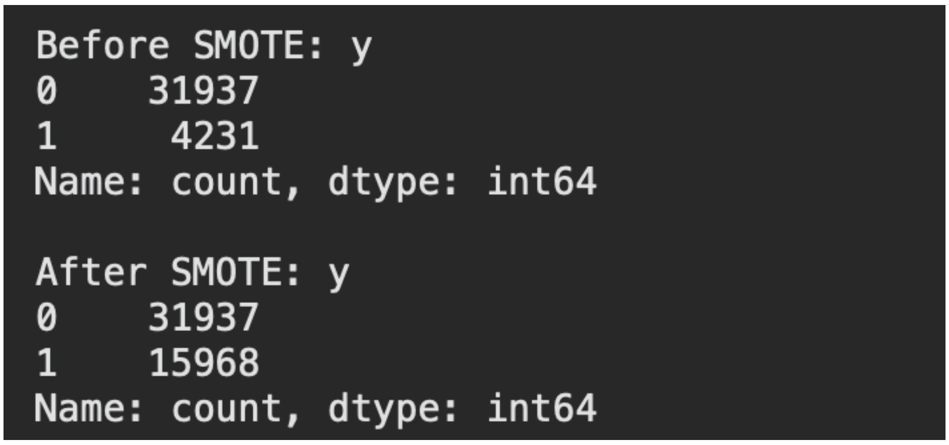
- The dataset was divided into 80% training and 20% testing portions.
- Stratified sampling ensured both sets retained the same proportion of “yes” and “no” cases.
- Prevented bias from class imbalance during model training and evaluation.
- Helped assess how well the model generalizes to unseen data.



```
Train shape: (36168, 31)  Test shape: (9043, 31)

Train target distribution:
 y
0    0.883018
1    0.116982
Name: proportion, dtype: float64

Test target distribution:
 y
0    0.883003
1    0.116997
Name: proportion, dtype: float64
```

Handling Data Imbalance :

- Used to balance the target classes by generating synthetic samples for the minority class (“yes”).
- Works by creating new samples along the line segments joining existing minority class neighbors.
- Prevents the model from becoming biased toward the majority class (“no”).



```
Before SMOTE: y
0    31937
1     4231
Name: count, dtype: int64

After SMOTE: y
0    31937
1    15968
Name: count, dtype: int64
```

- Sampling_Strategy = 1

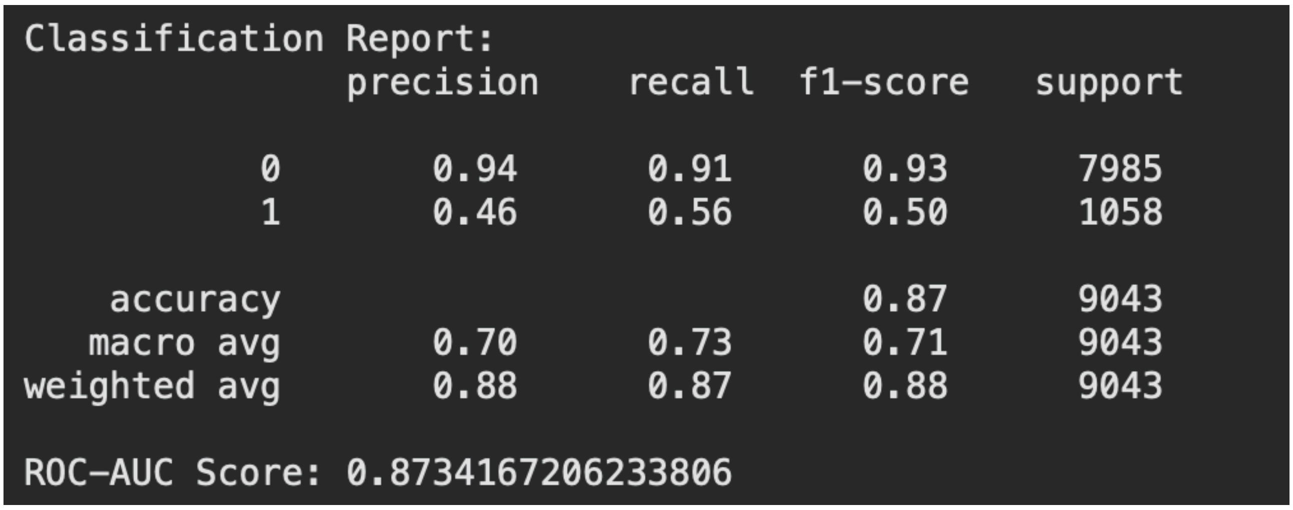
Model Building :

Logistic Regression :

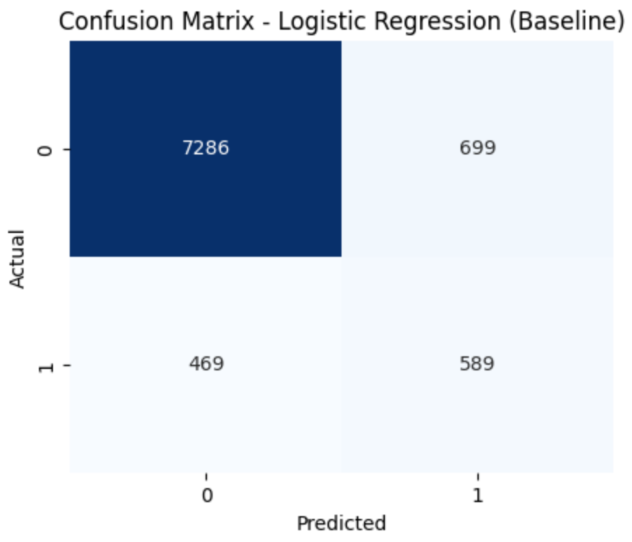
- A classification algorithm that models the probability of a binary outcome using the logistic (sigmoid) function.
- Coefficients are estimated using Maximum Likelihood Estimation (MLE).
- Produces interpretable weights, showing how each predictor affects the odds of subscribing to a term deposit.

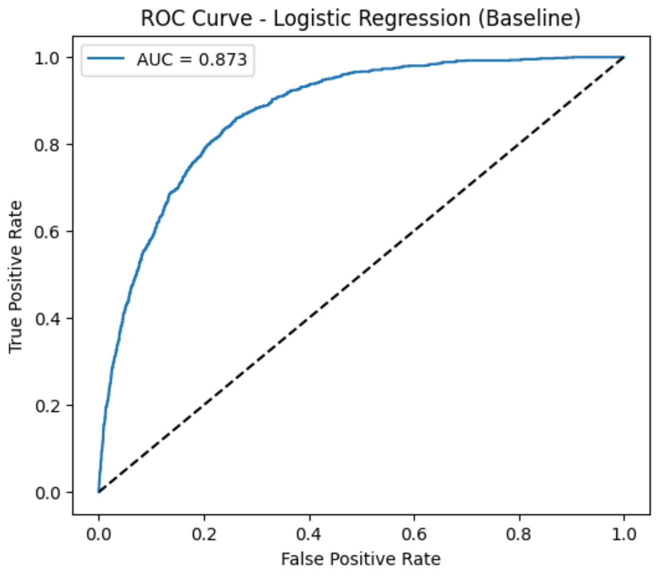
$$P(Y=1|X) = \frac{1}{1+e^{-(\beta_0+\beta_1 X_1+\beta_2 X_2+\ldots+\beta_n X_n)}}$$

Model Evaluation :

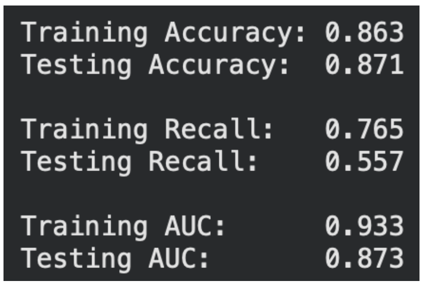


```
Classification Report:
               precision    recall  f1-score   support

           0       0.94      0.91      0.93      7985
           1       0.46      0.56      0.50      1058

    accuracy                           0.87      9043
   macro avg       0.70      0.73      0.71      9043
weighted avg       0.88      0.87      0.88      9043

ROC-AUC Score: 0.8734167206233806
```





Overfitting Check :



```
Training Accuracy: 0.863
Testing Accuracy:  0.871

Training Recall:   0.765
Testing Recall:    0.557

Training AUC:      0.933
Testing AUC:       0.873
```

Optimized Logistic Regression :

- Adds a penalty term to control model complexity and prevent overfitting.
- L2 (Ridge) shrinks coefficients smoothly toward zero, reducing variance.
- L1 (Lasso) drives some coefficients exactly to zero, performing feature selection.
- Improves generalization and stability compared to the baseline logistic regression.

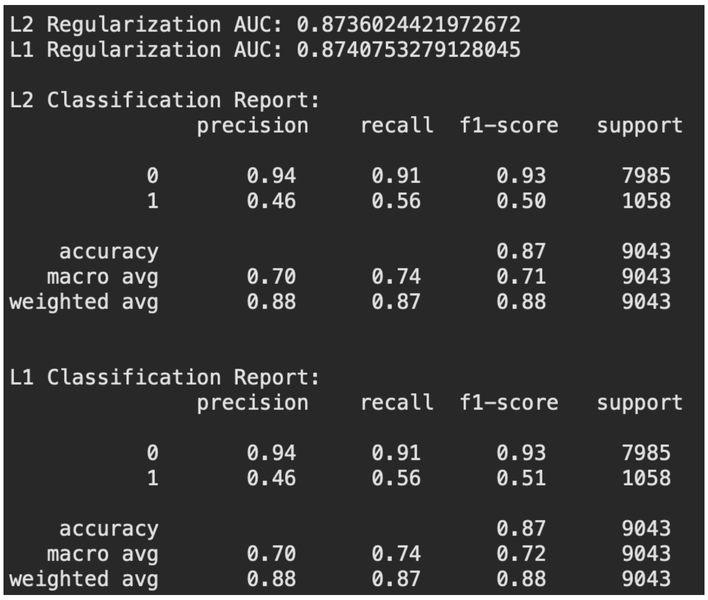
- **L2 Regularization (Ridge):**

$$J(\beta) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(p_i) + (1 - y_i)\log(1 - p_i)] + \lambda \sum_{j=1}^{n} \beta_j^2$$

- **L1 Regularization (Lasso):**

$$J(\beta) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(p_i) + (1 - y_i)\log(1 - p_i)] + \lambda \sum_{j=1}^{n} |\beta_j|$$

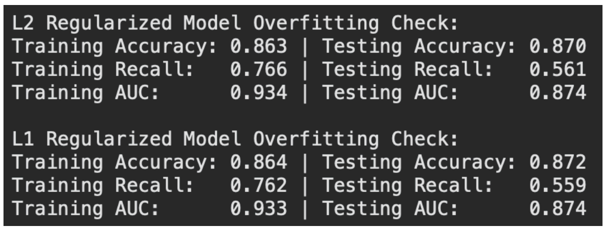
Model Evaluation :



```
L2 Regularization AUC: 0.8736024421972672
L1 Regularization AUC: 0.8740753279128045

L2 Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.91      0.93      7985
           1       0.46      0.56      0.50      1058

    accuracy                           0.87      9043
   macro avg       0.70      0.74      0.71      9043
weighted avg       0.88      0.87      0.88      9043


L1 Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.91      0.93      7985
           1       0.46      0.56      0.51      1058

    accuracy                           0.87      9043
   macro avg       0.70      0.74      0.72      9043
weighted avg       0.88      0.87      0.88      9043
```

- No drastic Improvements

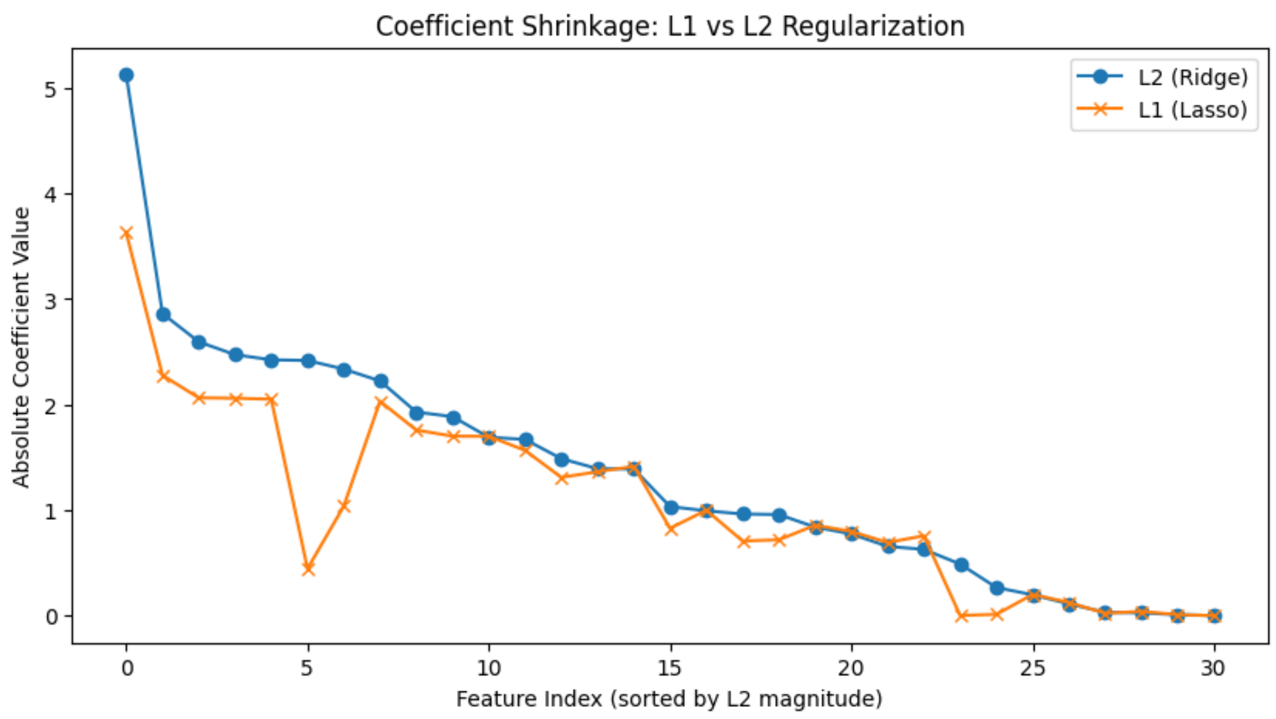
Overfitting Check :



```
L2 Regularized Model Overfitting Check:
Training Accuracy: 0.863 | Testing Accuracy: 0.870
Training Recall:   0.766 | Testing Recall:   0.561
Training AUC:      0.934 | Testing AUC:      0.874

L1 Regularized Model Overfitting Check:
Training Accuracy: 0.864 | Testing Accuracy: 0.872
Training Recall:   0.762 | Testing Recall:   0.559
Training AUC:      0.933 | Testing AUC:      0.874
```

Both L1 and L2 regularization successfully prevented further overfitting and stabilized optimization without sacrificing accuracy or AUC. The generalization gap remains small and kind of acceptable, and the models are now conditioned for practical with limited data. L1 has a slight edge due to its feature-selection property and marginally better AUC.

Comparing Convergence and Coefficient Shrinkage :



- **Convergence Speed:** The L2 (Ridge) model converged in 1973 iterations, while L1 (Lasso) reached the limit at 2000 iterations, confirming that L2 optimization is smoother and faster due to its differentiable penalty term.

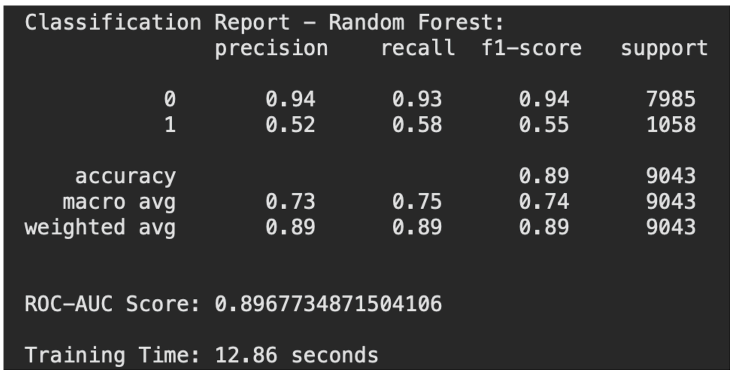
- **Coefficient Shrinkage:** The plot shows that L1 produces smaller magnitudes for several coefficients even close to zero, Indicating stronger shrinkage and potential feature elimination. In contrast, L2 keeps all coefficients small but nonzero, distributing weight more evenly across features.

- L1 regularization enforces sparsity (implicit feature selection), while L2 promotes stability. Both improve generalization, but Ridge (L2) optimized more efficiently in this dataset.

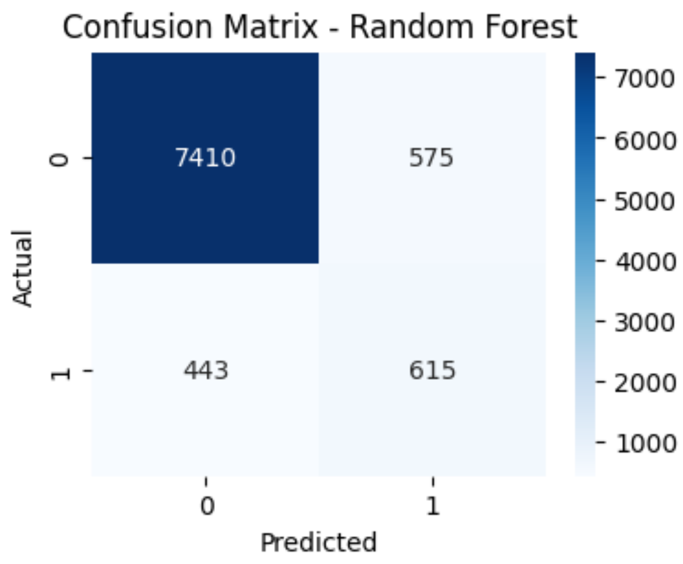
Random Forest :

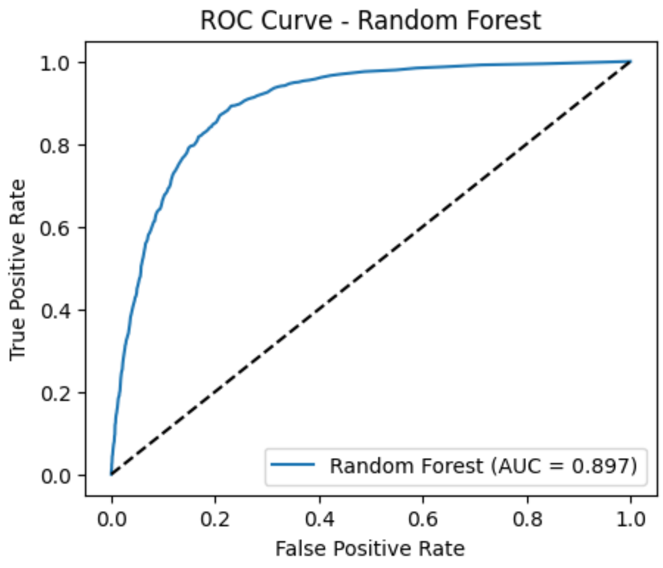
- An ensemble learning method that builds multiple decision trees and aggregates their predictions (majority vote).
- Uses bootstrap sampling (bagging) to reduce variance and improve robustness.
- Captures non-linear relationships and feature interactions automatically.

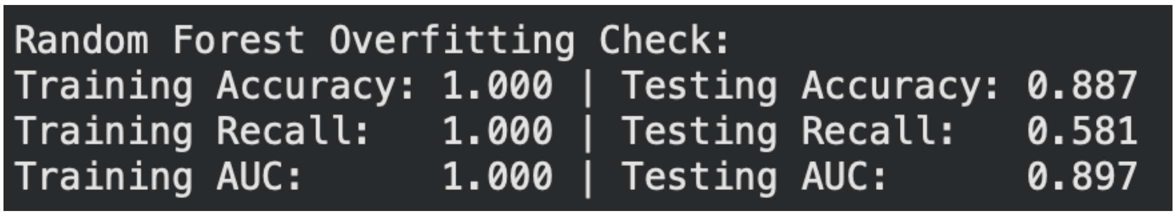
Model Evaluation :



```
Classification Report - Random Forest:
               precision    recall  f1-score   support

           0       0.94      0.93      0.94      7985
           1       0.52      0.58      0.55      1058

    accuracy                           0.89      9043
   macro avg       0.73      0.75      0.74      9043
weighted avg       0.89      0.89      0.89      9043


ROC-AUC Score: 0.8967734871504106

Training Time: 12.86 seconds
```







```
Random Forest Overfitting Check:
Training Accuracy: 1.000 | Testing Accuracy: 0.887
Training Recall:   1.000 | Testing Recall:   0.581
Training AUC:      1.000 | Testing AUC:      0.897
```

- The model fits the training data perfectly but loses ~13% accuracy on unseen data, which is a clear indication of overfitting.
- The recall drop shows that while the model detects all positives in training, it misses many in the test set, meaning it memorized training patterns rather than generalizing them.
- The near-perfect AUC on training data confirms overfitting; however, a test AUC of 0.893 still indicates strong discriminative performance overall.

XGBoost :

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i)$$

- $\hat{y}_i$: Final predicted value for data point i.
- $f_k(x_i)$: Output of the k^{th} decision tree for that point.
- K: Total number of trees.

Each new tree f_k tries to minimize the total **objective function**:

$$\text{Obj} = \sum_i l(y_i, \hat{y}_i) + \sum_k \Omega(f_k)$$

Where:

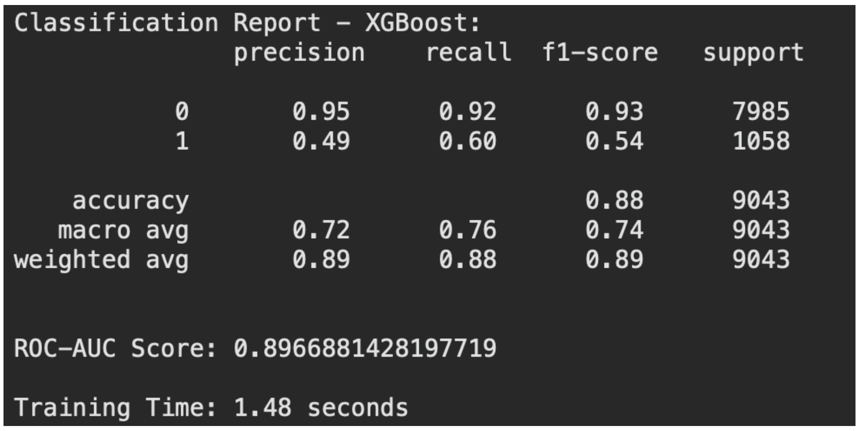
- $l(y_i, \hat{y}_i)$: Loss function (e.g., log loss for classification).
- $\Omega(f_k)$: Regularization term that penalizes overly complex trees (controls overfitting).

$$\Omega(f_k) = \gamma T + \frac{1}{2}\lambda \sum_j w_j^2$$

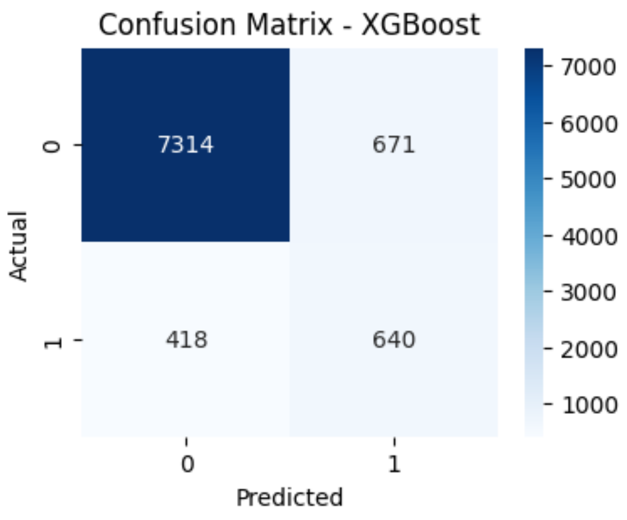
- T: Number of leaves (tree complexity).
- λ, γ: Regularization parameters (like L1/L2 in logistic regression).

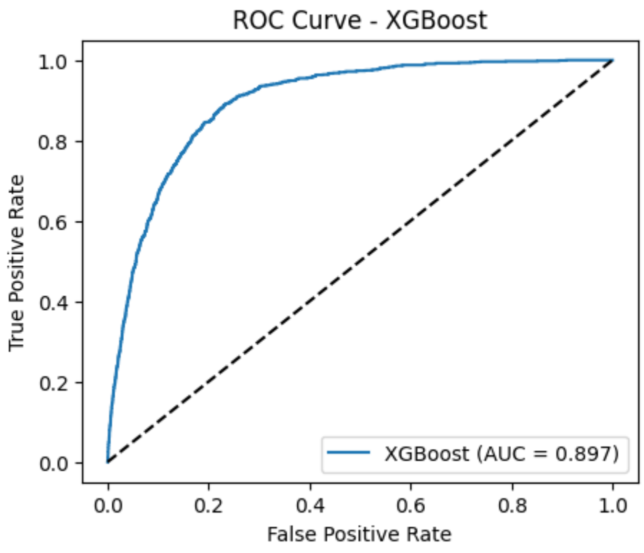
- Gradient boosting algorithm that builds trees sequentially; each new tree corrects the errors of the previous ones.
- Combines gradient descent optimization with regularization, making it faster and more accurate than standard boosting.
- Effectively handles non-linear patterns and imbalanced data.

Model Evaluation :



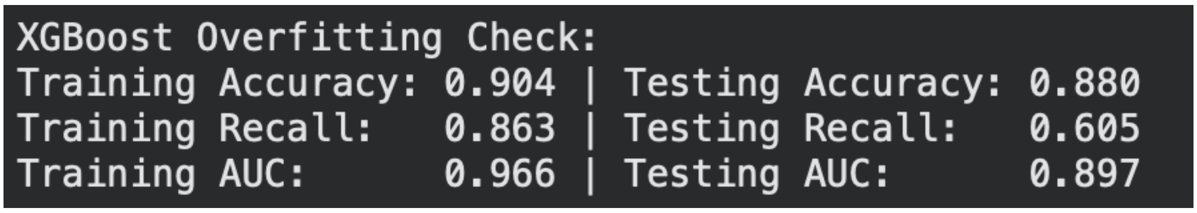
```
Classification Report - XGBoost:
              precision    recall  f1-score   support

           0       0.95      0.92      0.93      7985
           1       0.49      0.60      0.54      1058

    accuracy                           0.88      9043
   macro avg       0.72      0.76      0.74      9043
weighted avg       0.89      0.88      0.89      9043


ROC-AUC Score: 0.8966881428197719

Training Time: 1.48 seconds
```





- No drastic improvements

Overfitting Check :



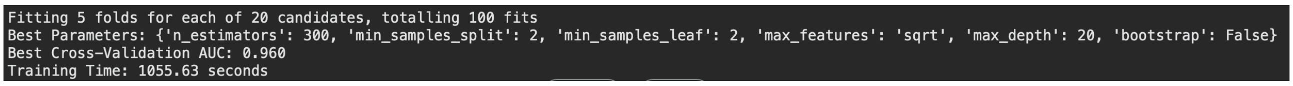
```
XGBoost Overfitting Check:
Training Accuracy: 0.904 | Testing Accuracy: 0.880
Training Recall:   0.863 | Testing Recall:   0.605
Training AUC:      0.966 | Testing AUC:      0.897
```

- Much better
- Controlled Overfitting

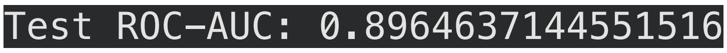
Hyperparameter Tuning :

Tuning Random Forest :

- Used GridSearchCV with 5-fold cross-validation over 20 parameter combinations (total = 100 fits).
- Evaluated using AUC (Area Under ROC Curve) as the scoring metric.



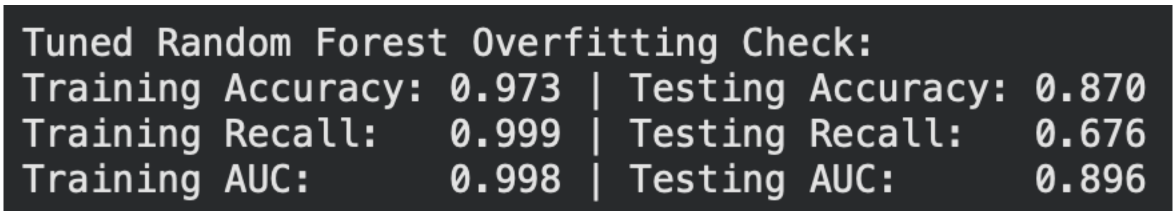
```
Fitting 5 folds for each of 20 candidates, totalling 100 fits
Best Parameters: {'n_estimators': 300, 'min_samples_split': 2, 'min_samples_leaf': 2, 'max_features': 'sqrt', 'max_depth': 20, 'bootstrap': False}
Best Cross-Validation AUC: 0.960
Training Time: 1055.63 seconds
```



```
Test ROC-AUC: 0.8964637144551516
```

- Just Improved

Overfitting Check :

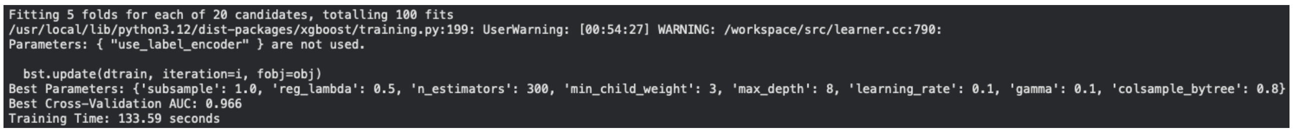


```
Tuned Random Forest Overfitting Check:
Training Accuracy: 0.973 | Testing Accuracy: 0.870
Training Recall:   0.999 | Testing Recall:   0.676
Training AUC:      0.998 | Testing AUC:      0.896
```

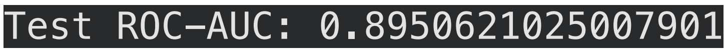
- Still Overfitting

Tuning XGBoost :

- Applied GridSearchCV with 5-fold cross-validation across 20 parameter combinations (total = 100 fits).
- Optimization metric: AUC (Area Under ROC Curve).



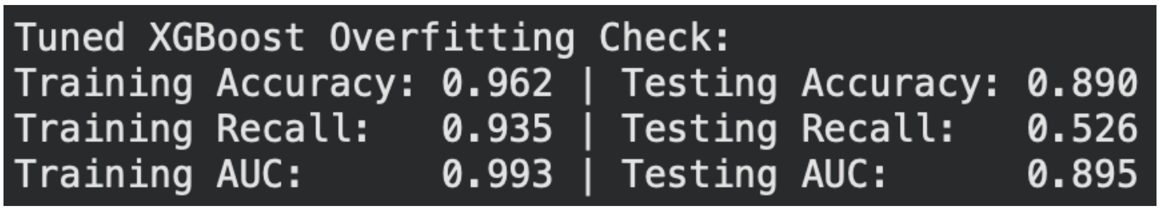
```
Fitting 5 folds for each of 20 candidates, totalling 100 fits
/usr/local/lib/python3.12/dist-packages/xgboost/training.py:199: UserWarning: [00:54:27] WARNING: /workspace/src/learner.cc:790:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
Best Parameters: {'subsample': 1.0, 'reg_lambda': 0.5, 'n_estimators': 300, 'min_child_weight': 3, 'max_depth': 8, 'learning_rate': 0.1, 'gamma': 0.1, 'colsample_bytree': 0.8}
Best Cross-Validation AUC: 0.966
Training Time: 133.59 seconds
```



```
Test ROC-AUC: 0.8950621025007901
```

- Tuned parameters improved balance between model complexity and generalization.
- Achieved higher AUC with significantly lower training time than Random Forest.
- Optimized max_depth, learning_rate, and reg_lambda effectively reduced overfitting while maintaining high predictive power.

Overfitting Check :



```
Tuned XGBoost Overfitting Check:
Training Accuracy: 0.962 | Testing Accuracy: 0.890
Training Recall:   0.935 | Testing Recall:   0.526
Training AUC:      0.993 | Testing AUC:      0.895
```

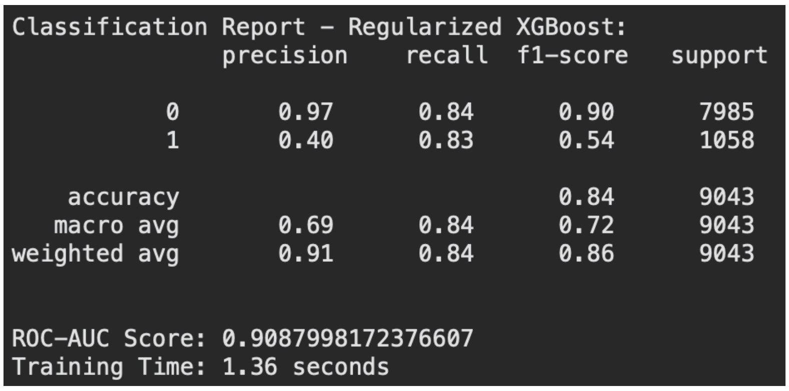
- Still Overfitting

XGBoost Refined :

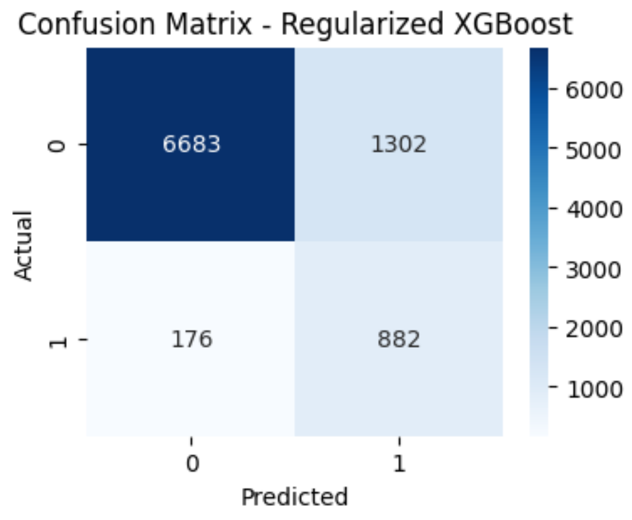
- Build upon the tuned XGBoost model to further control overfitting and handle class imbalance more effectively.
- Introduce regularization parameters (reg_lambda, reg_alpha) and class weighting (scale_pos_weight) to improve generalization.
- Develop a refined, final version of the XGBoost model with optimized bias–variance trade-off and stable performance across datasets.

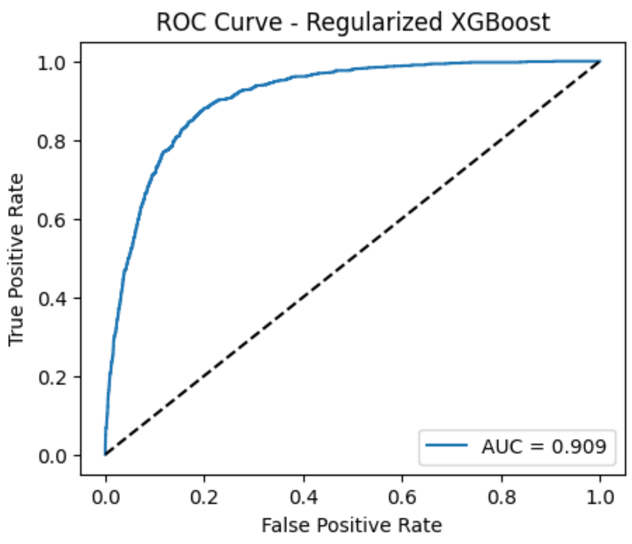
Key Regularization Parameters:
reg_lambda = *2, ref_alpha* = *1, scale_pos_weight* = *7.55*
These control:
- L2 Regularization (λ): Prevents overfitting by shrinking large weights.
- L1 Regularization (α): Drives less informative features' weights toward zero.
- Scale_pos_weight: Balances majority ("no") and minority ("yes") classes based on their ratio.

Model Evaluation :



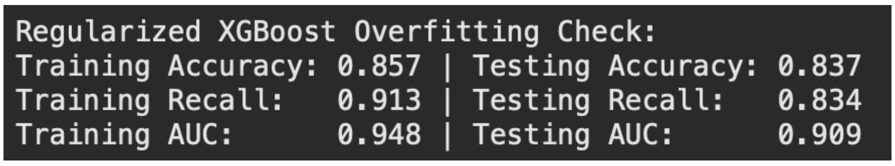
```
Classification Report - Regularized XGBoost:
              precision    recall  f1-score   support

           0       0.97      0.84      0.90      7985
           1       0.40      0.83      0.54      1058

    accuracy                           0.84      9043
   macro avg       0.69      0.84      0.72      9043
weighted avg       0.91      0.84      0.86      9043


ROC-AUC Score: 0.9087998172376607
Training Time: 1.36 seconds
```





- Achieved excellent generalization and balanced recall, outperforming all previous models.
- Strong regularization + class weighting minimized overfitting.
- AUC > 0.90 demonstrates robust discriminative power.
- This configuration is selected as the final production-ready model.

Overfitting Check :



```
Regularized XGBoost Overfitting Check:
Training Accuracy: 0.857 | Testing Accuracy: 0.837
Training Recall:   0.913 | Testing Recall:   0.834
Training AUC:      0.948 | Testing AUC:      0.909
```

Conclusion :

The combination of numerical optimization and machine learning was used to create predictions on if a bank customer would subscribe to a Term Deposit or not.

A baseline logistic regression model provided an initial point of reference to evaluate the more complex algorithms. It also helped to easily understand how the algorithm worked; therefore, it was easy to see how the predictors related to the log-odds of becoming a subscriber. Since the Logistic Regression model uses a linear decision boundary, it had significant limitations with respect to its ability to find non-linear relationships and incorrectly classifying observations, particularly for the imbalanced dataset.

The Regularized version of the Logistic Regression model did reduce the variance slightly but continued to underperform when making predictions for the minority class.

The prediction power of the Random Forest and XGBoost models proved to be much greater compared to Logistic Regression. However, while the Random Forest and XGBoost models did outperform Logistic Regression, they could still be improved by further tuning of the Hyperparameters.

The last and final model is labelled Regularized XGBoost which was built by adding both L1 and L2 Regularization penalties in addition to the Class weighting (scaling) of the positive class (scale_pos_weight). This is the ideal model for the dataset created by this project because it delivers the highest ROC-AUC score of 0.91 while maintaining a relatively constant Recall across both Training and Testing datasets.

To summarise the findings of this project, it can be concluded that both Regularisation and Balancing of the Class Weights are essential in the development of a robust model for imbalanced datasets. The final product provides a reliable data-driven methodology for Banks to more accurately Identify and Target Clients, and optimise the Use of their Marketing Resources and increase conversion rates u.