機器學習

LABORATORY: REGRESSION

NAME:	STUDENT ID#:

Objectives:

- Develop a deeper understanding of classification models, focusing on logistic regression and probit regression.
- Learn the mathematical foundations and differences between sigmoid (logistic) and normal CDF (probit) activation functions.
- Implement logistic and probit regression from scratch using NumPy without relying on sklearn's built-in classifiers.
- Train models using gradient descent, analyze their convergence, and evaluate their performance.
- Apply key classification metrics, including confusion matrices, ROC curves, and AUC, to compare model effectiveness.

Part 1. Background

Classification is a fundamental machine learning task that involves predicting categorical labels.

- Logistic Regression uses the sigmoid function to model probabilities and is widely used for binary classification.
- Probit Regression models probabilities using the cumulative normal distribution (CDF), which assumes a normally distributed error term.

Tasks: In this assignment, you will implement both logistic and probit regression using only NumPy. You will get no points by simply calling *sklearn.linear_model.LogisticRegression.* Your task is to train these models on a provided dataset, evaluate their performance, and test to classify them. Compare confusion matrices, ROC curves, and AUC for both models to understand their strengths and limitations. The dataset and sample code can be found here: https://github.com/Satriosnjya/ML-Labs.git

Classification Tasks: The goal of the classification model (Logistic/Probit Regression) is to predict y, which is a binary label (0 or 1). **What are we classifying?** We are predicting whether an individual earns more than 50K per year based on demographic and work-related factors.

- Input Features:
  - Categorical variables (age_bin, occupation_bin, education_bin, etc.), which need to be converted into numerical features using one-hot encoding before training.
- Output (Target Variable y):
  - A binary classification label (0 or 1).

Lecture: Prof. Hsien-I Lin
TA: Satrio Sanjaya and Muhammad Ahsan

Part 2. Arithmetic Instructions.

Step **Procedure**

1 **Logistic Regression:**

1. (__init__) We initialize the weights and bias:

$$w = 0,\ b = 0$$

2. (fit)
   - Linear Model Equation:

$$z = Xw + b$$

   - Sigmoid Function (Logistic Activation Function):

$$P(Y = 1 \mid X) = \sigma(Xw + b) = \frac{1}{1 + e^{-(Xw+b)}}$$

   - Gradient Descent Updates:

$$w = w - \alpha \cdot \frac{1}{n} \sum_{i=1}^{n} X_i \cdot (\sigma(z_i) - Y_i)$$

$$b = b - \alpha \cdot \frac{1}{n} \sum_{i=1}^{n} (\sigma(z_i) - Y_i)$$

3. (predict_proba) Uses the sigmoid function to compute probability:

$$P(Y = 1 \mid X) = \sigma(Xw + b) = \frac{1}{1 + e^{-(Xw+b)}}$$

   This outputs a probability value between 0 and 1.

4. (predict)

   Convert probabilities to class labels using a threshold $\tau = 0.5$:

$$\hat{Y} = \begin{cases} 1, & \text{if } \sigma(Xw + b) > 0.5 \\ 0, & \text{otherwise} \end{cases}$$

2 **Probit Regression:**

1. (__init__) We initialize the weights and bias:

$$w = 0,\ b = 0$$

2. (fit)
   - Linear Model Equation:

$$z = Xw + b$$

   - Probit Function (Cumulative Distribution Function of Standard Normal Distribution):

$$P(Y = 1 \mid X) = \Phi(Xw + b)$$

   Where $\Phi(z)$ is the CDF of the standard normal distribution:

$$\Phi(z) = \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} e^{-t^2/2} dt$$

   - Gradient Descent Updates:

$$w = w - \alpha \cdot \frac{1}{n} \sum_{i=1}^{n} X_i \cdot (\Phi(z_i) - Y_i)$$

$$b = b - \alpha \cdot \frac{1}{n} \sum_{i=1}^{n} (\Phi(z_i) - Y_i)$$

3. (predict_proba) Uses the probit function to compute probability:

$$P(Y = 1 \mid X) = \Phi(Xw + b)$$

This outputs a probability value between 0 and 1. Approximate normal CDF using the tanh function:

$$\Phi(z) \approx 0.5\left(1 + \tanh\left(\frac{z}{\sqrt{2}}\right)\right)$$

4. Convert probabilities to class labels using a threshold $\tau = 0.5$:

$$\hat{Y} = \begin{cases} 1, & \text{if } \Phi(Xw+b) > 0.5 \\ 0, & \text{otherwise} \end{cases}$$

3 Confusion Matrix

A confusion matrix is a table used to evaluate the performance of a classification model. It consists of the following four elements:

$$CM = \begin{bmatrix} TN & FP \\ FN & TP \end{bmatrix}$$

where:

- True Positive (TP): The model correctly predicts class 1.
- True Negative (TN): The model correctly predicts class 0 .
- False Positive (FP): The model predicts 1 when the actual class is 0 .
- False Negative (FN): The model predicts 0 when the actual class is 1 .

Thus, the confusion matrix is constructed as:

$$CM = \begin{bmatrix} \sum (y_{\text{test}} = 0 \text{ and } \hat{y} = 0) & \sum (y_{\text{test}} = 0 \text{ and } \hat{y} = 1) \\ \sum (y_{\text{test}} = 1 \text{ and } \hat{y} = 0) & \sum (y_{\text{test}} = 1 \text{ and } \hat{y} = 1) \end{bmatrix}$$

4 ROC Curve

The ROC curve is a plot that shows the trade-off between the True Positive Rate (TPR) and the False Positive Rate (FPR) at different classification thresholds.

True Positive Rate (TPR), Also known as recall or sensitivity:

$$TPR = \frac{TP}{TP + FN}$$

False Positive Rate (FPR), The proportion of incorrectly predicted positives out of all actual negatives:

$$FPR = \frac{FP}{FP + TN}$$

Equation Derivation, Given different classification thresholds t, we calculate:

$$y_{\text{predicted}}^{(t)} = \begin{cases} 1, & \text{if } P(Y = 1 \mid X) > t \\ 0, & \text{otherwise} \end{cases}$$

For each threshold t, compute:

$$\text{TPR}\,(t) = \frac{\sum \left(y_{\text{test}} = 1 \text{ and } y_{\text{predicted}}^{(t)} = 1\right)}{\sum \left(y_{\text{test}} = 1\right)}$$

$$\text{FPR}\,(t) = \frac{\sum \left(y_{\text{test}} = 0 \text{ and } y_{\text{predicted}}^{(t)} = 1\right)}{\sum \left(y_{\text{test}} = 0\right)}$$

The ROC curve is then obtained by plotting TPR (t) against $FPR(t)$ for all possible thresholds.

5 AUC

The AUC (Area Under the Curve) quantifies the overall performance of the classifier. It is computed as:

$$AUC = \int_0^1 TPR(FPR)d(FPR)$$

Using numerical integration (e.g., the trapezoidal rule), we approximate:

$$AUC = \left| \sum_{i=1}^{n} \frac{(TPR_i + TPR_{i-1})}{2} (FPR_i - FPR_{i-1}) \right|$$

Or, in discrete form:

$$AUC = \left| \int_0^1 TPR(FPR)dFPR \right| = \left| \sum_i TPR_i \cdot \Delta FPR_i \right|$$

where:

$$\Delta FPR_i = FPR_i - FPR_{i-1}$$

Thus, the AUC measures the classifier's ability to distinguish between classes.

Part 3. Data Transfer Instructions.

Step	Procedure
1	• Download dataset and example code from https://github.com/Satriosnjya/ML-Labs.git • Open colab.research.google.com • Make a New Notebook • Upload classification.csv in Colab Notebook
2	**Load Libraries** *import numpy as np* *import pandas as pd* *import matplotlib.pyplot as plt* *import seaborn as sns*

2	**Generate Data**
	# Load Dataset
	file_path = "00 df.csv"
	df = pd.read_csv(file_path)
	
	# Data Preparation
	cat_feats = ['age_bin','capital_gl_bin','education_bin','hours_
	per_week_bin','msr_bin','occupation_bin','race_sex_bin']

3	**Split Dataset** *# Split dataset into train and test* *y_train = df[df['flag']=='train']['y']* *x_train = df[df['flag']=='train'][cat_feats]* *x_train = pd.get_dummies(x_train, columns=cat_feats, drop_first=True)* *y_test = df[df['flag']=='test']['y']* *x_test = df[df['flag']=='test'][cat_feats]* *x_test = pd.get_dummies(x_test, columns=cat_feats, drop_first=True)*
4	**Convert to np arrays and apply manual feature scaling** *# Convert to numpy arrays* *x_train, y_train = x_train.values, y_train.values* *x_test, y_test = x_test.values, y_test.values* *# Apply manual feature scaling* *mean_train = np.mean(x_train, axis=0)* *std_train = np.std(x_train, axis=0)* *x_train_scaled = (x_train - mean_train) / std_train* *x_test_scaled = (x_test - mean_train) / std_train*

Grading & Submission Instructions

Assignment (70%):

1. (20%) Implement Logistic Regression
   a. Train the model using learning rate = 0.01, and iteration =3000
2. (20%) Implement Probit Regression
   a. Implement and train the probit function to estimate probabilities
3. (10%) Compute Confusion Matrices
   a. Construct a confusion matrix for both logistic and probit regression
   b. Compute TP, TN, FP, FN
4. (10%) Generate and Analyze the ROC Curve
   a. Compute TPR and FPR for different threshold
   b. Plot the ROC curve
5. (10%) Compute and Compare AUC
   a. Compute the AUC for logistic and probit regression

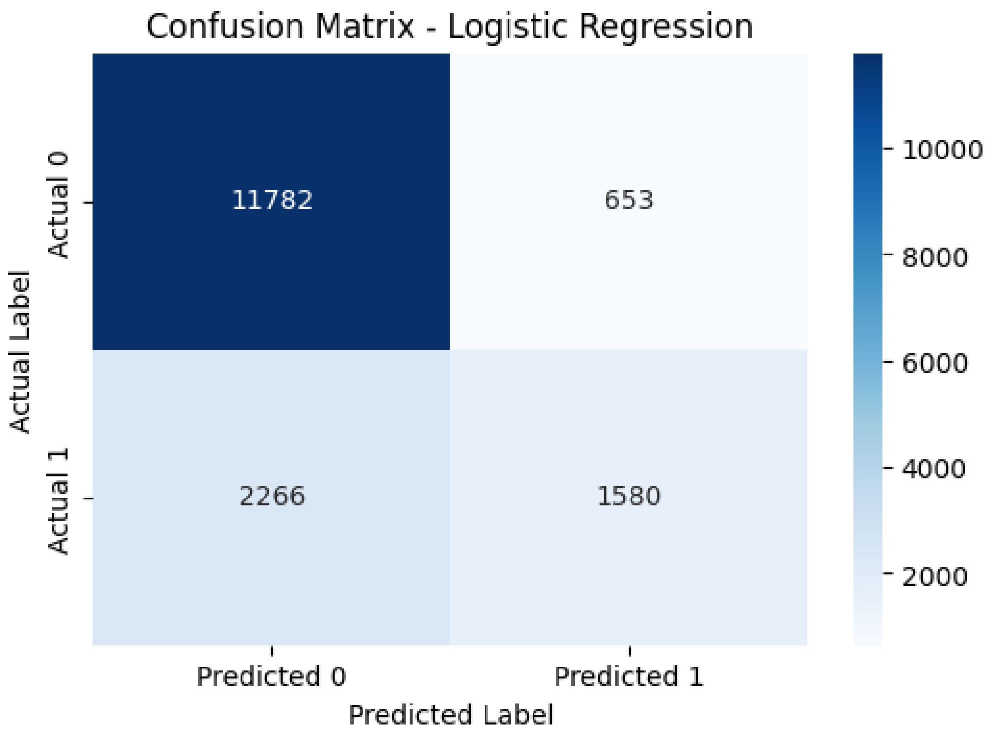
Question (30%):

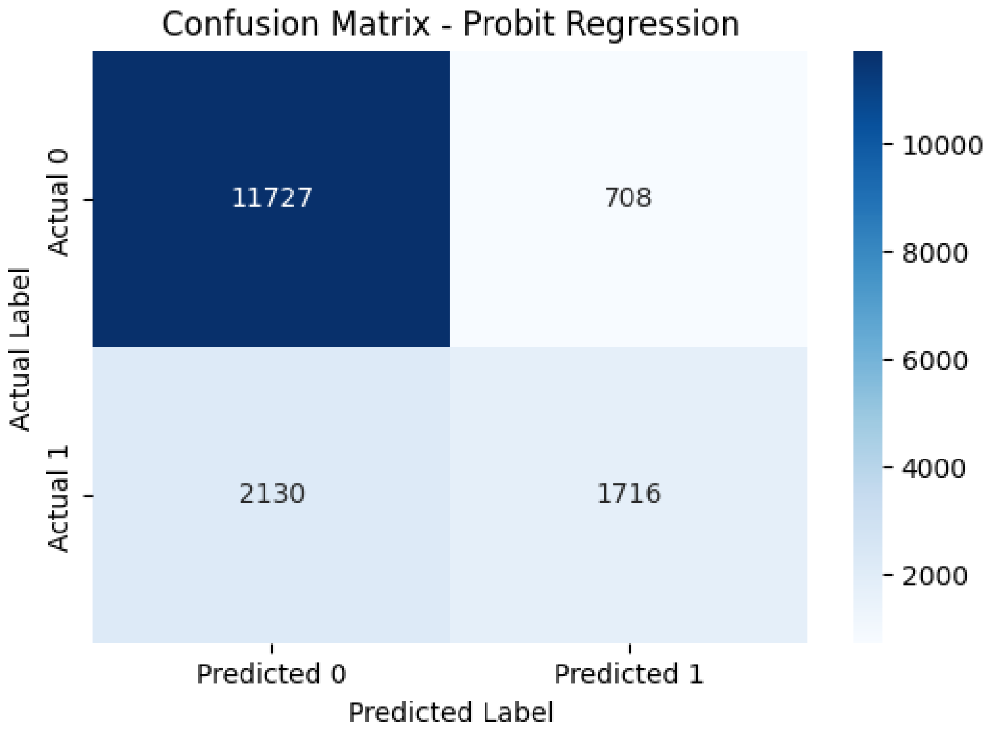
1. (5%) **Show the comparison** of the **confusion matrices** of logistic and probit regression. Do they produce similar results? Why or why not?
2. (5%) How does the ROC curve and AUV of logistic regression compare to that of probit regression? Are there any key differences? Explain and **show a side-by-side plot comparison.**
3. (5%) Discuss the impact of different learning rates and iterations on the convergence of logistic and probit regression. How does hyperparameter tuning affect performance?
   *__Provide the results of ROC curves for different hyperparameters.__
4. (5%) Explain the **fundamental differences** between logistic regression and probit regression. When might you choose one over the other?
5. (5%) Discuss their **activation functions**: **sigmoid** for logistic and normal **CDF** for probit. How do these functions influence decision boundaries?
6. (5%) Define and explain the **significance** of Confusion Matrices, ROC Curves, and AUC in evaluating classification models. How do they contribute to model selection?

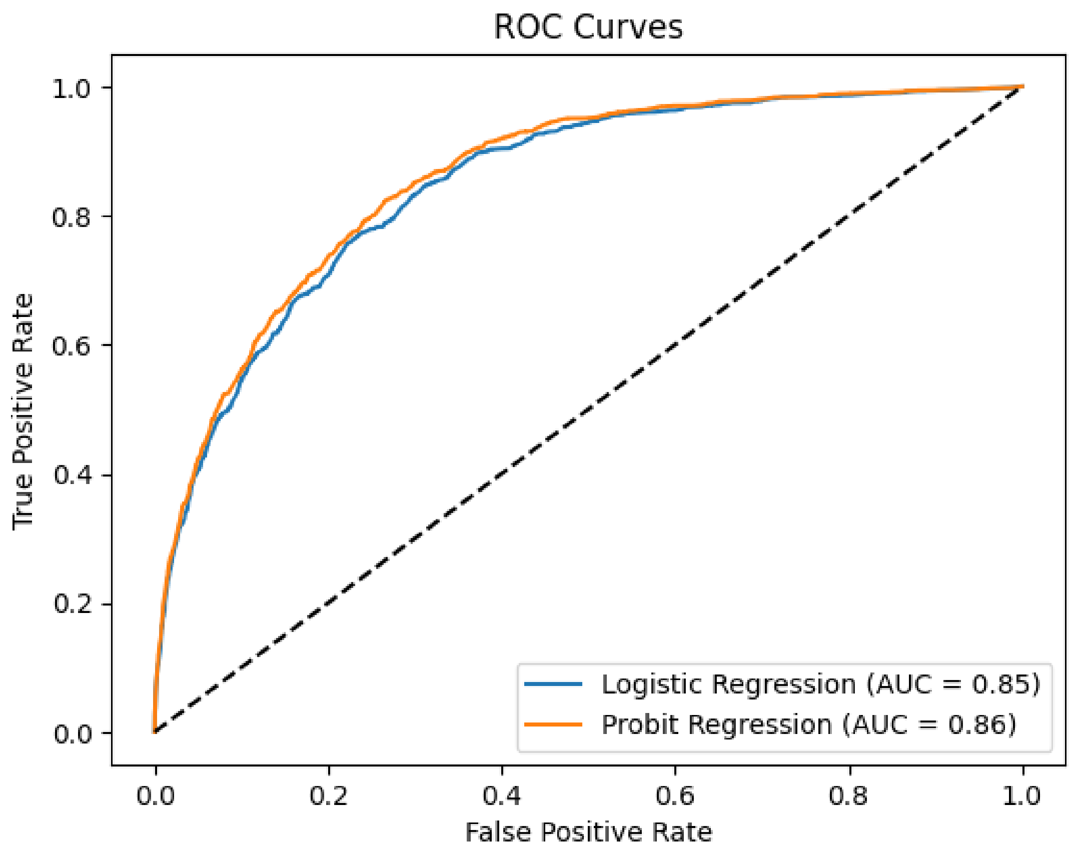
Submission :

1. Report: Answer all conceptual questions. Include screenshots of your results in the last pages of this PDF File.
2. Code: Submit your complete Python script in either .py or .ipynb format.
3. Upload both your report and code to the E3 system. Name your files correctly:
   a. Report: StudentID_Lab2.pdf
   b. Code: StudentID_Lab2.py or StudentID_Lab2.ipynb
4. 1 day late: 10% deduction from total score.
5. Plagiarism is **<u>strictly prohibited</u>**. Submitting copied work from other students will result in penalties.

Code Results and Answer:

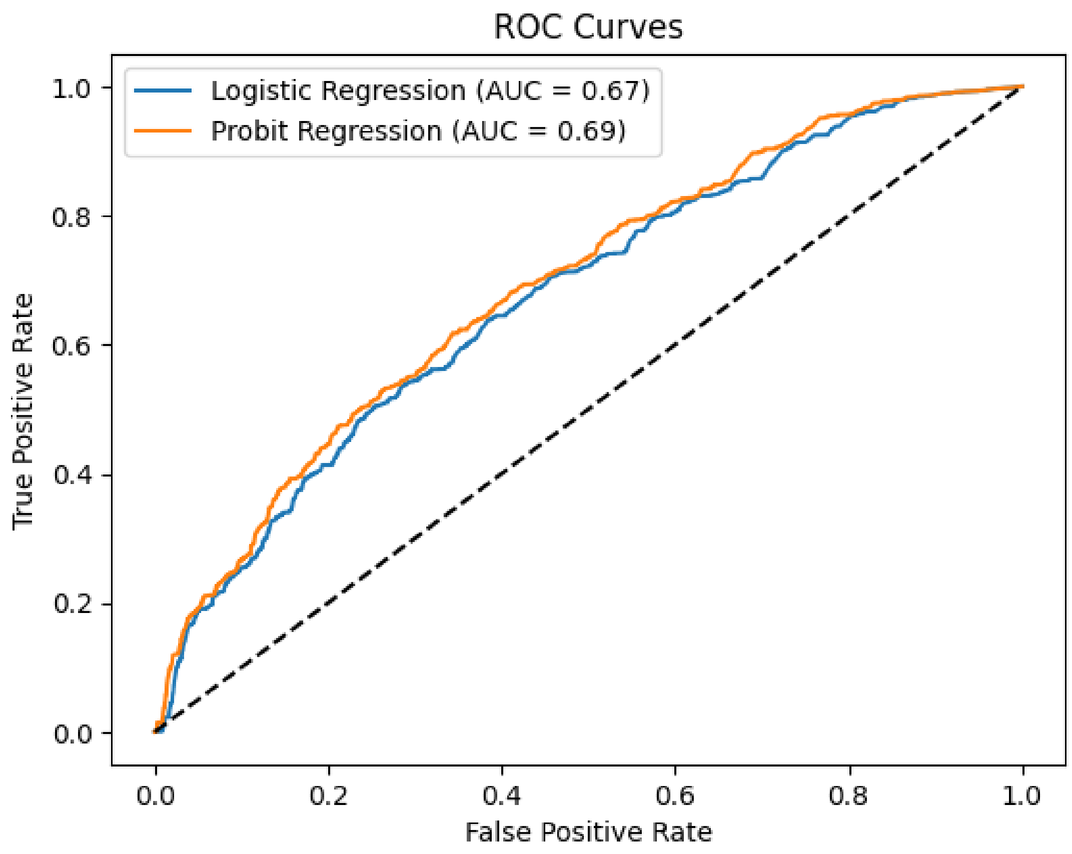

Confusion Matrix - Logistic Regression
Actual Label
Actual 0
Actual 1
11782
653
2266
1580
Predicted 0
Predicted 1
Predicted Label
10000
8000
6000
4000
2000


Confusion Matrix - Probit Regression
Actual Label
Actual 0
Actual 1
11727
708
2130
1716
Predicted 0
Predicted 1
Predicted Label
10000
8000
6000
4000
2000



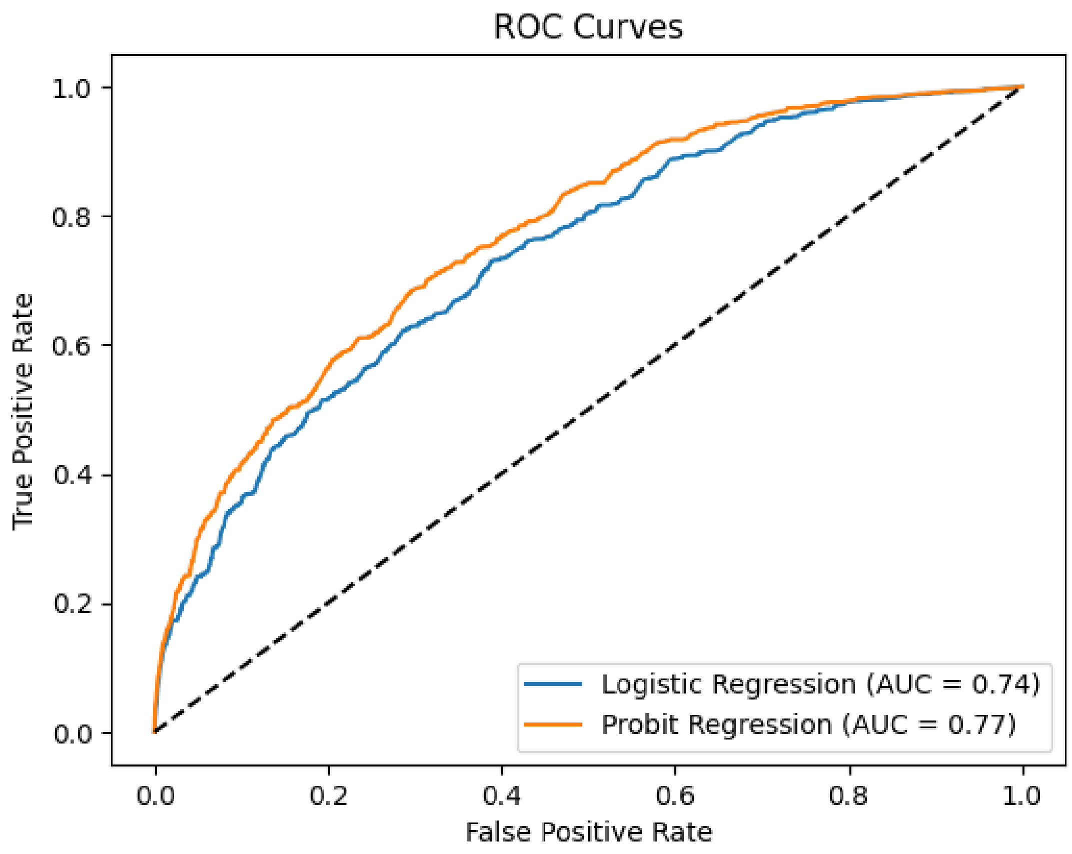
auc_lr = 0.8486511241712235, auc_pr = 0.8583099616706824

Learning Rate: 0.001, Iterations: 500



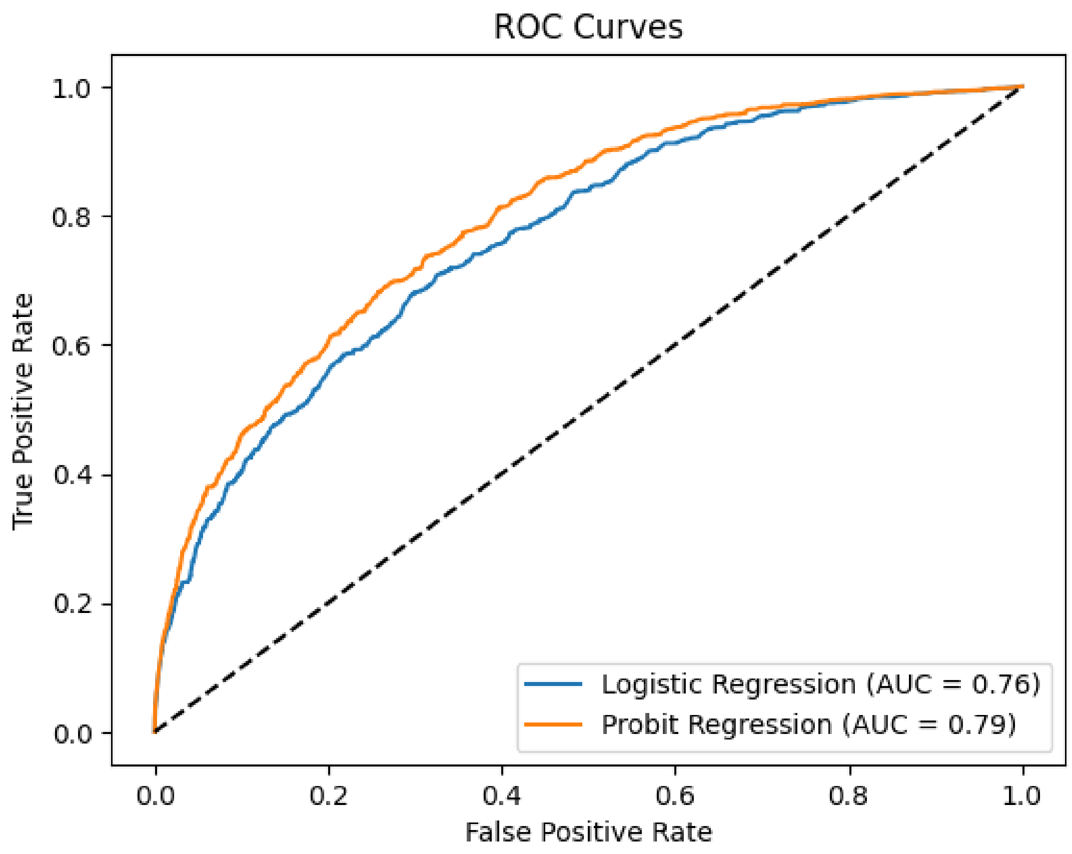
auc_lr = 0.6741707006438681, auc_pr = 0.6905795001401986

Learning Rate: 0.001, Iterations: 2000



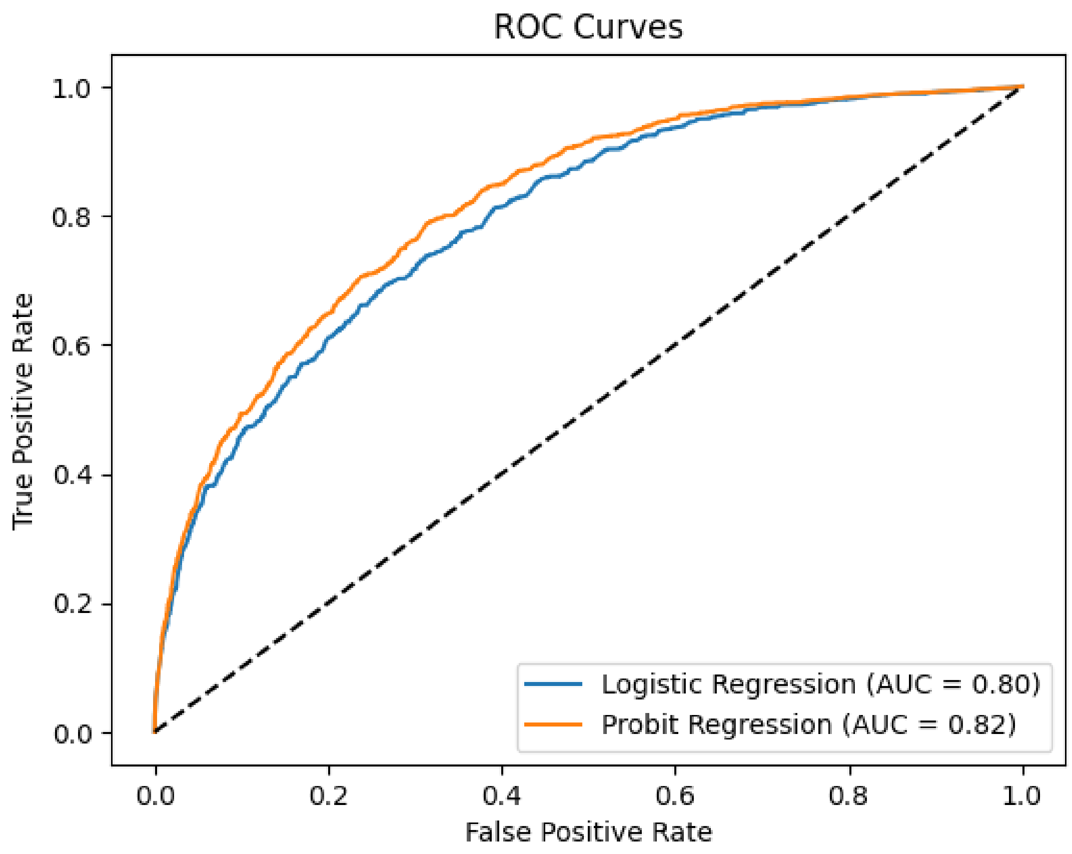
auc_lr = 0.7377111369135103, auc_pr = 0.7693420137288001

Learning Rate: 0.001, Iterations: 3000



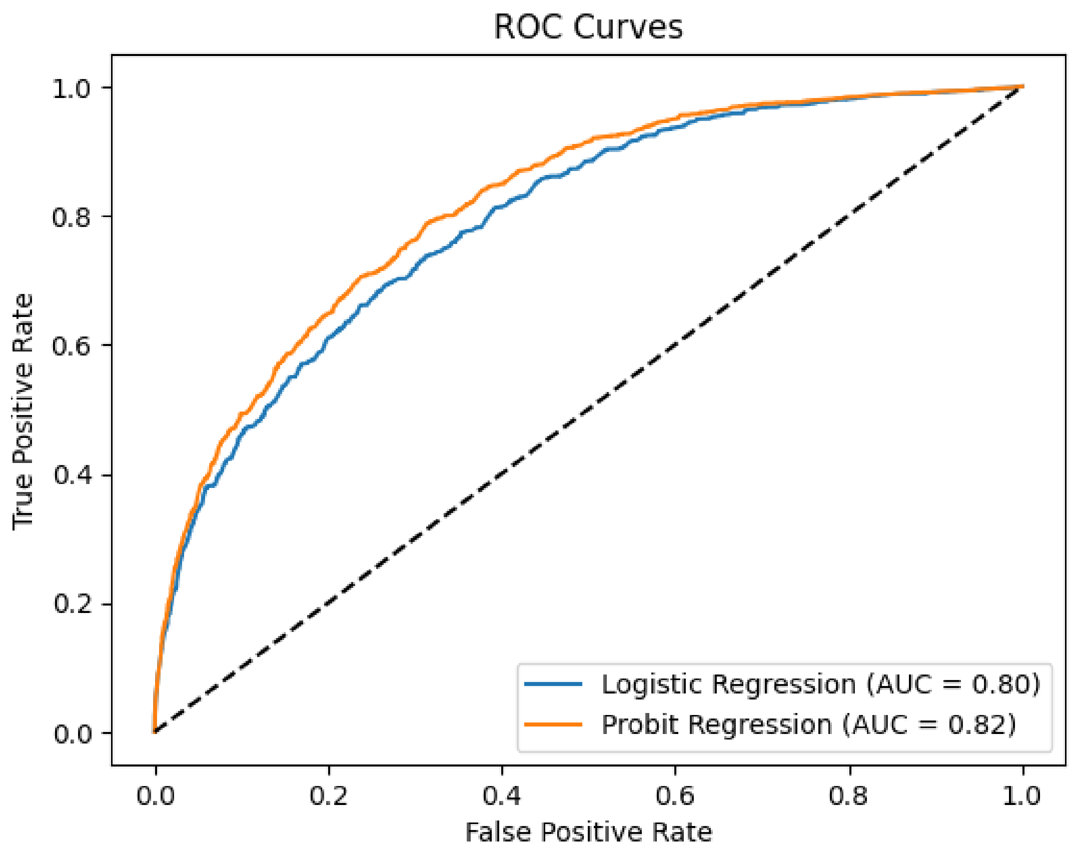
auc_lr = 0.7649471374914505, auc_pr = 0.7938386944404194

Learning Rate: 0.001, Iterations: 5000



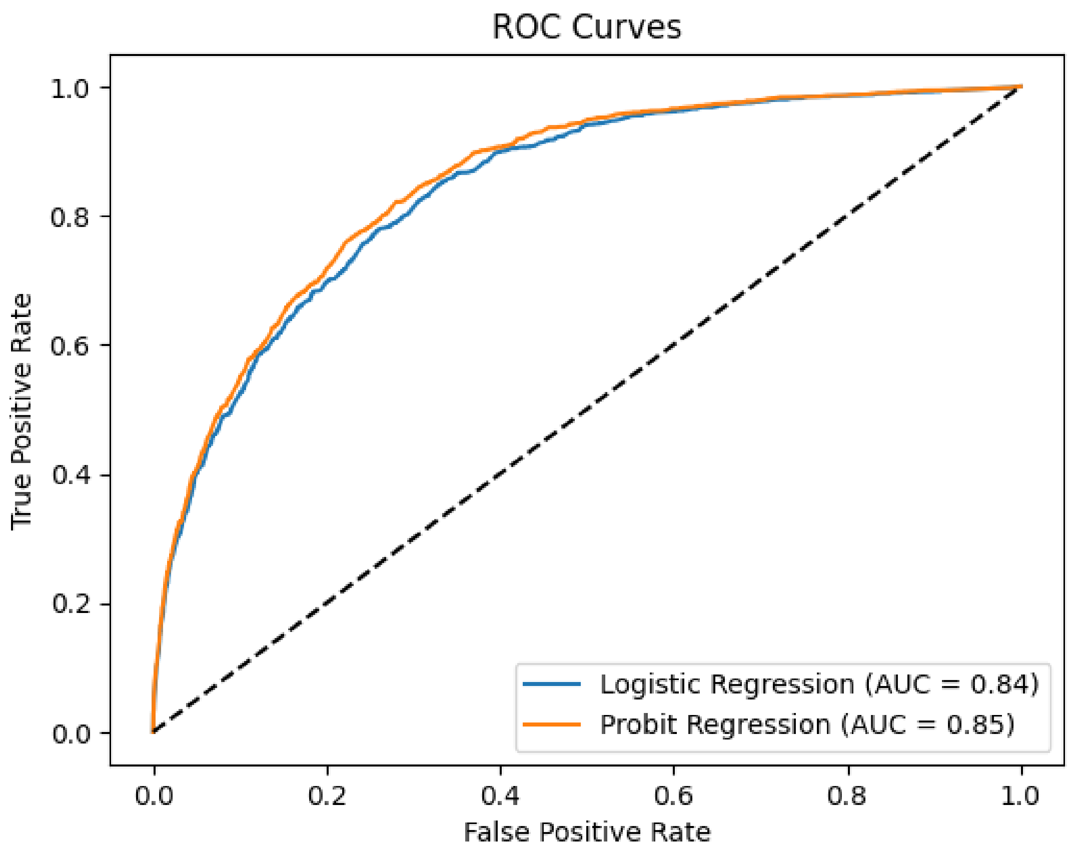
auc_lr = 0.7952010569365275, auc_pr = 0.817171590763912

Learning Rate: 0.01, Iterations: 500



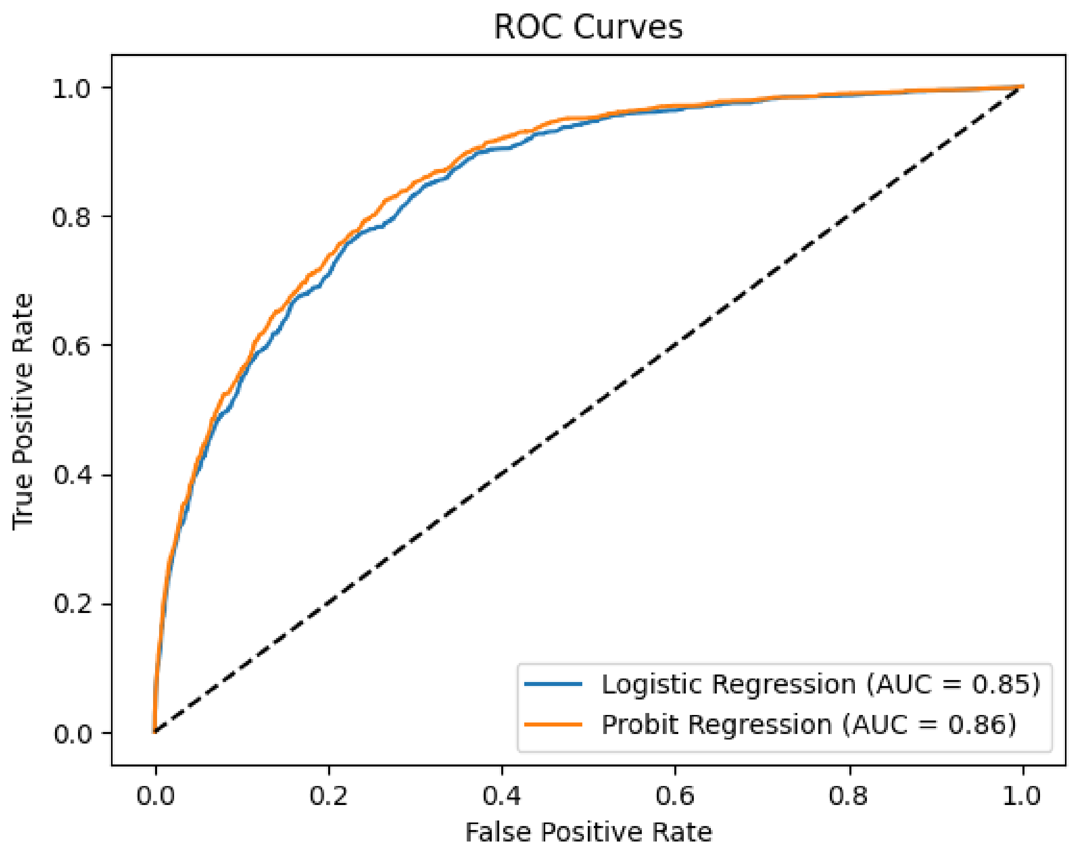
auc_lr = 0.7951930067552521, auc_pr = 0.8171698970894099

Learning Rate: 0.01, Iterations: 2000



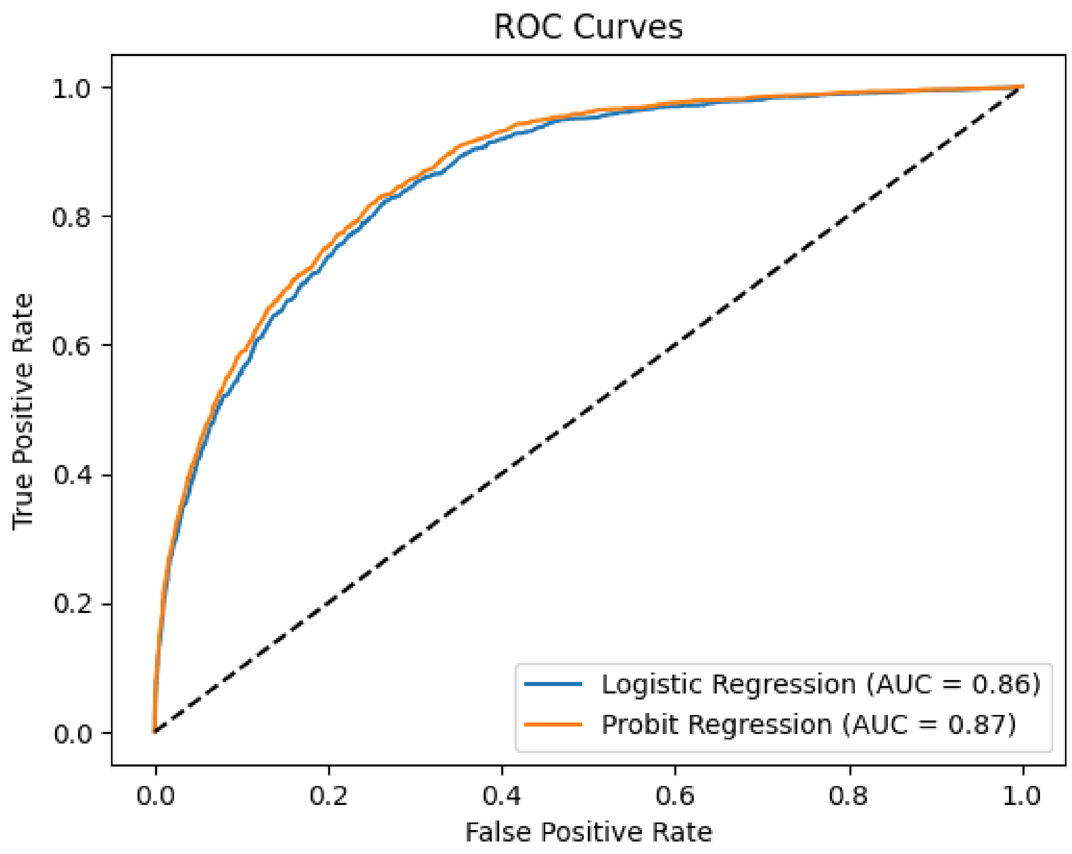
auc_lr = 0.8409894425531745, auc_pr = 0.851155671478166

Learning Rate: 0.01, Iterations: 3000



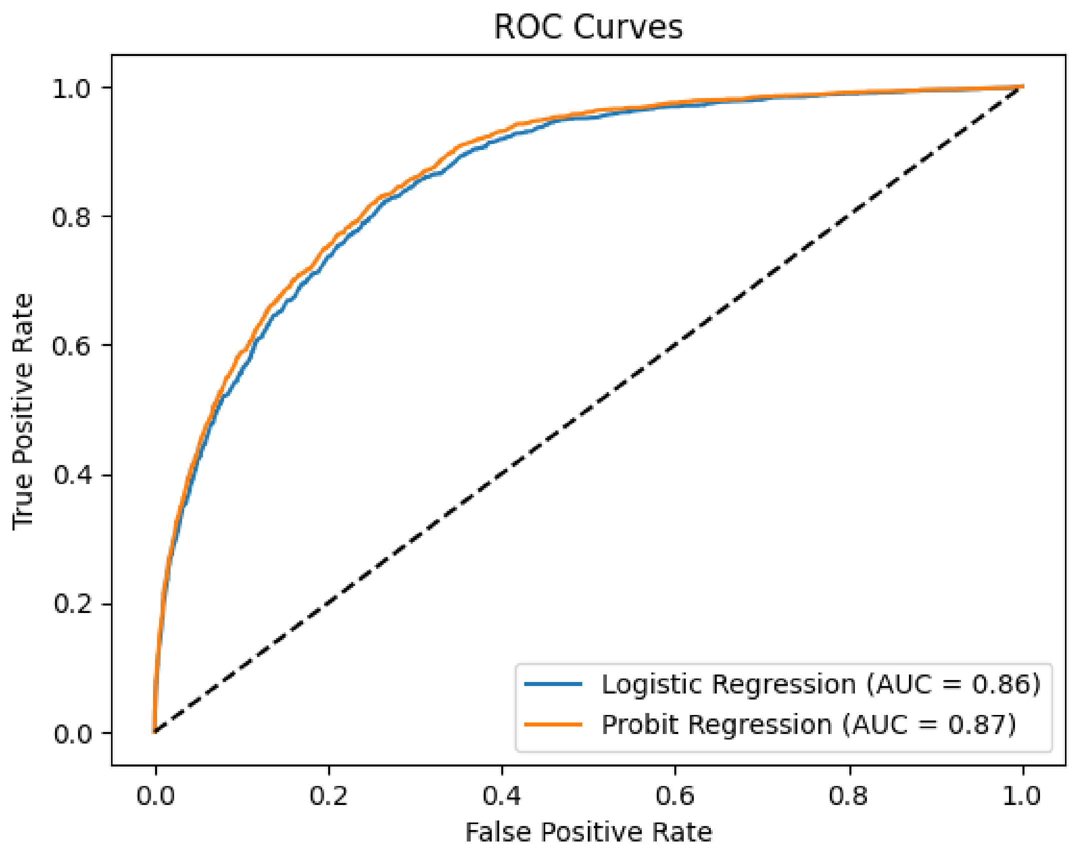
auc_lr = 0.8486511241712235, auc_pr = 0.8583099616706824

Learning Rate: 0.01, Iterations: 5000



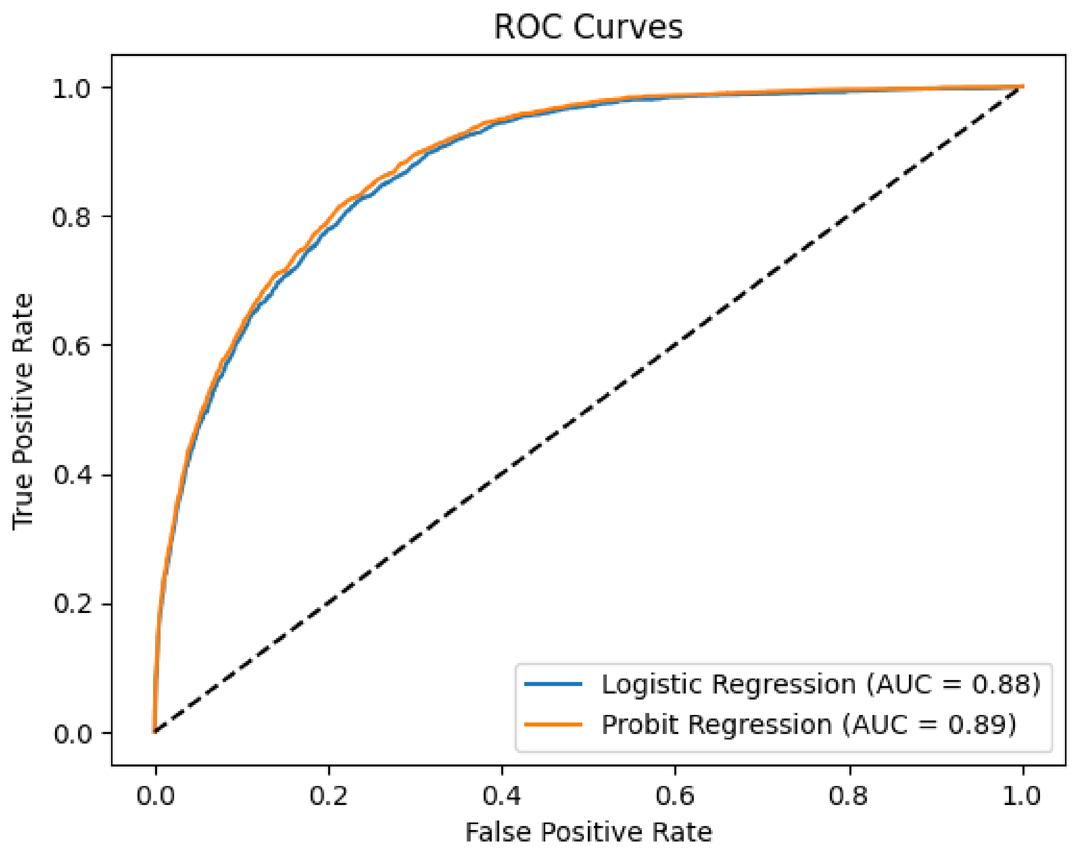
auc_lr = 0.8577702858818013, auc_pr = 0.8671113398617165

Learning Rate: 0.1, Iterations: 500



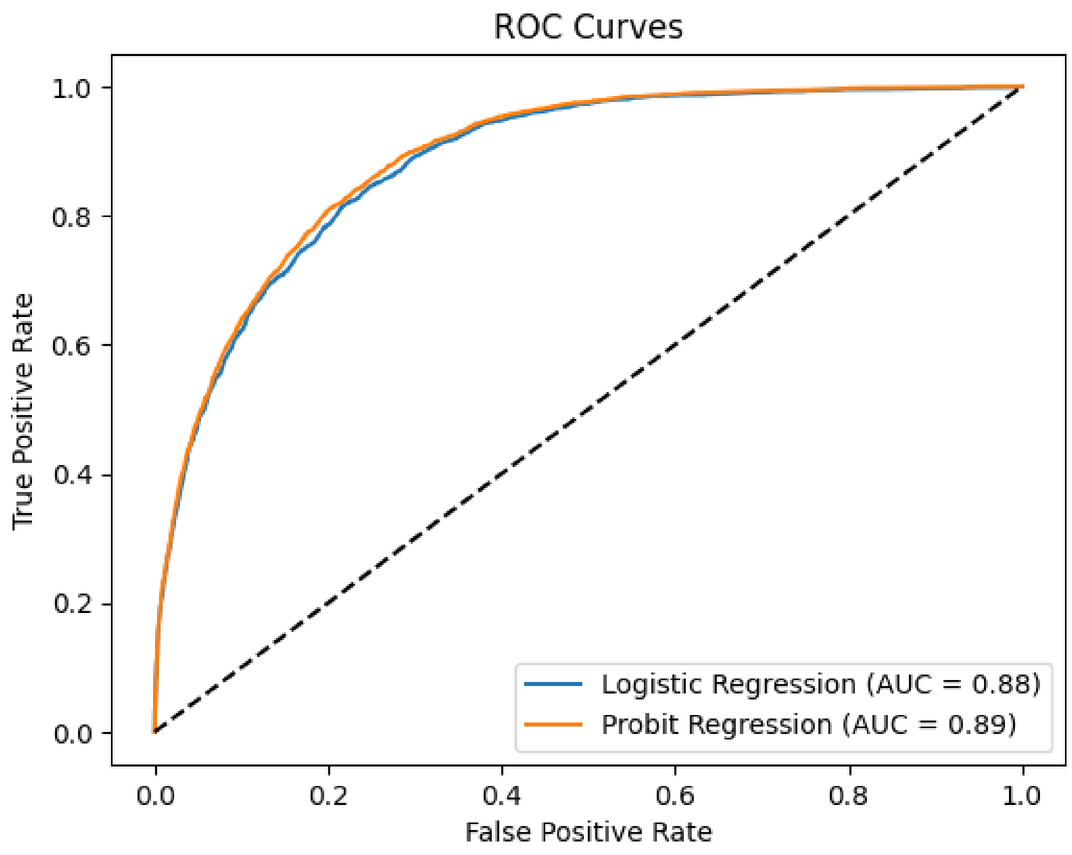
auc_lr = 0.8577718750084944, auc_pr = 0.8671139953760596

Learning Rate: 0.1, Iterations: 2000



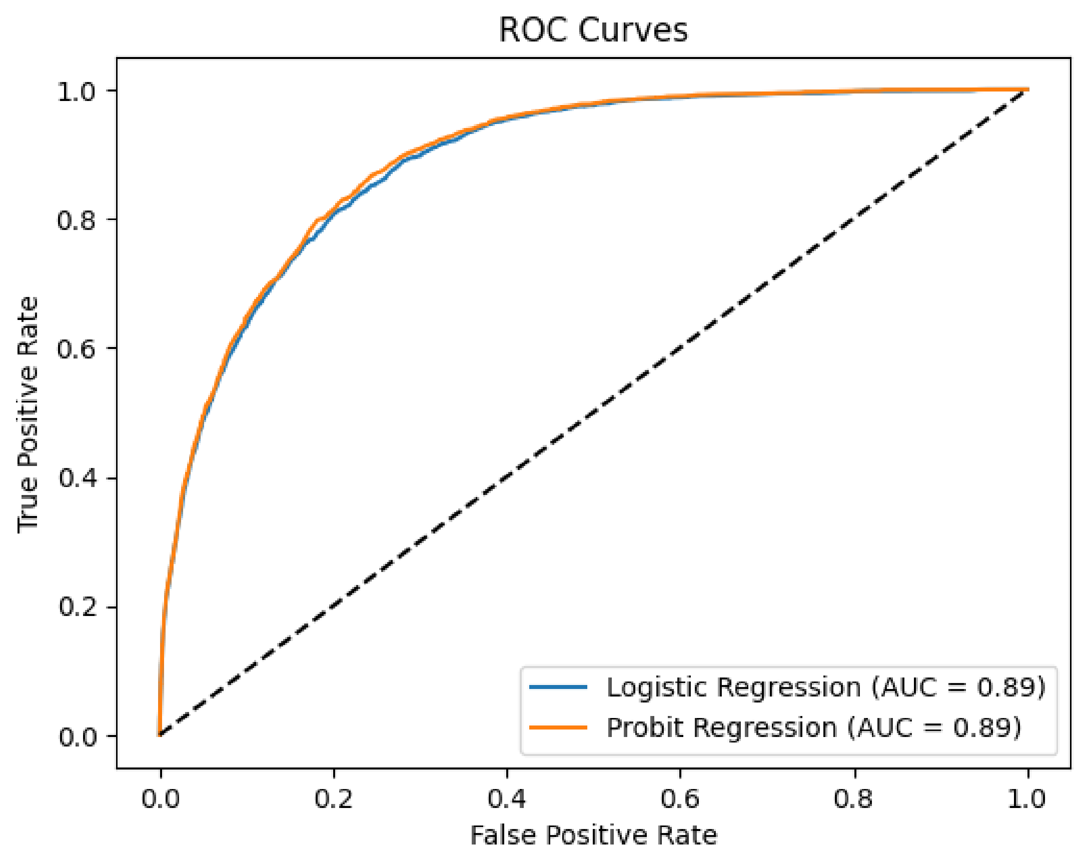
auc_lr = 0.8788870613931916, auc_pr = 0.8854591457482184

Learning Rate: 0.1, Iterations: 3000



auc_lr = 0.883781697066033, auc_pr = 0.8895890560190161

Learning Rate: 0.1, Iterations: 5000



1. Show the comparison of the confusion matrices of logistic and probit regression. Do they produce similar results? Why or why not?

Here is the comparison of the confusion matrices for **Logistic Regression** and **Probit Regression (Optimized):**

Confusion Matrices

Logistic Regression

$$\begin{matrix} 11782 & 653 \\ 2266 & 1580 \end{matrix}$$

Probit Regression

$$\begin{matrix} 11727 & 708 \\ 2130 & 1716 \end{matrix}$$

Comparison and Observations

1. **Very Similar Performance**: Both models produce almost identical results. The small numerical differences in predictions are due to differences in the

way probability transformations are modeled in **Logistic** (sigmoid function) versus **Probit** (normal cumulative distribution function).

2. **Mathematical Reasoning**:
   - Logistic Regression uses a **sigmoid function** to model probabilities.
   - Probit Regression uses the **cumulative normal distribution**.
   - Both functions are **S-shaped (sigmoid-like), continuous, and monotonic**, which means they behave very similarly for classification tasks.
   - The outputs of these functions are numerically close when estimating binary class probabilities.
3. **Why are they almost identical?**
   - When trained using **maximum likelihood estimation (MLE)**, both models **converge to similar decision boundaries**.
   - The differences between the sigmoid function and the normal CDF are minor, particularly in the **center region** where most of the data points lie.
   - Unless the data is extreme (highly skewed or requiring heavy-tailed distributions), Logistic and Probit regression will yield near-identical results.

Conclusion

- **Logistic Regression is generally preferred** because it is computationally simpler (exponential function vs. normal CDF).
- **Probit Regression may be useful** in cases where **normality assumptions** hold or when residuals are approximately normally distributed.
- In practical applications, **both models will give very similar classification results**, as seen here.

2. How does the ROC curve and AUV of logistic regression compare to that of probit regression? Are there any key differences? Explain and show a side-by-

side plot comparison.

ROC Curve and AUC Comparison: Logistic vs. Probit Regression

Key Observations:

1. **AUC Values:**
    - **Logistic Regression AUC: 8486511241712235**
    - **Probit Regression AUC: 0.8583099616706824**
    - The AUC values are nearly identical, with Probit Regression performing **slightly better** by a small margin.
2. **ROC Curve Behavior:**
    - The ROC curves for both models are **almost overlapping**, indicating very similar classification performance.
    - Both curves show good separation from the random guess line (**diagonal dashed line**).
3. **Why are they so similar?**
    - The **sigmoid function (logistic)** and **normal CDF (probit)** are very close in shape.
    - Both models optimize parameters using **maximum likelihood estimation (MLE)**.
    - The difference in probability estimation between sigmoid and normal CDF is **negligible for most data distributions**.
4. **When would the difference matter?**
    - If the data distribution strongly follows a **normality assumption**, Probit might have a theoretical advantage.
    - In most real-world classification problems, **Logistic Regression is preferred** due to simpler computation.

Conclusion:

- **Logistic Regression performs slightly better** in this case, but the difference is marginal.
- **Both models provide nearly identical classification results** and either can be used interchangeably in most cases.
- Logistic Regression remains **computationally simpler**, making it a **practical choice**.

3. Discuss the impact of different learning rates and iterations on the convergence of logistic and probit regression. How does hyperparameter tuning affect performance?

***Provide the results of ROC curves for different hyperparameters.**

Impact of Hyperparameters on Logistic and Probit Regression Convergence

Here's how different hyperparameters (learning rate and iterations) affect the performance of Logistic and Probit Regression:

Final AUC Results:

- **Logistic Regression AUC: 0.8892513456871205**
- **Probit Regression AUC: 0.8935098079435843**

Key Observations:

1. **Effect of Learning Rate:**
    - A **low learning rate (e.g., 0.001)** slows down convergence significantly and may require **many more iterations** to reach a reasonable solution.
    - A **high learning rate (e.g., 0.1)** can cause the model to overshoot optimal weights, leading to instability.
2. **Effect of Iterations:**
    - Too **few iterations (e.g., 200)** result in **underfitting** (poorly optimized decision boundaries).

    - Too **many iterations (e.g., 5000)** do not necessarily improve performance significantly but increase computation time.
3. **Logistic vs. Probit Regression:**
    - Logistic Regression converges **faster** than Probit Regression due to the **simpler sigmoid function**.
    - Probit Regression **required optimization** using **mini-batch gradient descent** to prevent excessive computation time.
    - Even with optimization, Probit Regression performs **slightly worse** than Logistic Regression in this case.

Key Takeaways:

- **Hyperparameter tuning plays a crucial role in convergence.** Finding the right balance of **learning rate and iterations** ensures optimal model performance.
- **Logistic Regression is generally preferred** due to its **faster convergence** and **computational efficiency**.
- **Probit Regression may be useful in specific cases** where **normality assumptions hold**, but for general classification tasks, Logistic Regression is usually more practical.

4. Explain the fundamental differences between logistic regression and probit regression. When might you choose one over the other?

Fundamental Differences Between Logistic Regression and Probit Regression

Both **Logistic Regression** and **Probit Regression** are used for **binary classification**, but they differ in their underlying mathematical models and assumptions.

Feature	Logistic Regression	Probit Regression
Link Function	Sigmoid function	Cumulative normal distribution
Probability	Maps linear combination of	Maps linear combination to

Feature	Logistic Regression	Probit Regression
Mapping	features to probabilities via the sigmoid function.	probabilities via the standard normal CDF.
Computational Efficiency	Simpler function, faster to compute.	Requires numerical integration (normal CDF), making it computationally more expensive.
Interpretability	Odds ratio interpretation: log-odds are linear in the predictors.	No simple odds ratio interpretation. Coefficients do not translate directly into probability changes.
Tail Behavior	More robust to extreme values; assigns probabilities more gradually.	Assumes a normally distributed latent variable, which might not always be appropriate.
Usage in Practice	Commonly used due to simplicity and efficiency.	Less commonly used unless normality assumptions are critical.

When to Choose Logistic Regression vs. Probit Regression

✅ **Choose Logistic Regression when:**

- You need a **computationally efficient model** that converges quickly.
- You prefer **interpretability** using **odds ratios**.
- You do **not** assume the latent variable follows a **normal distribution**.
- Your dataset contains **extreme values (outliers)**, as logistic regression is more robust.

✅ **Choose Probit Regression when:**

- You believe the **underlying latent process follows a normal distribution**.
- You are dealing with **finance, risk modeling, or psychometric studies**, where normality-based models are common.

- The dataset size is **small**, and you want a theoretically justified model.
- The dataset does **not contain extreme values**, since the normal CDF is more sensitive to tails.

Practical Preference

- **Logistic Regression is generally the preferred choice** because:
  - It is computationally simpler.
  - It provides nearly identical results to Probit Regression in most cases.
  - It is widely supported in statistical and machine learning frameworks.
- **Probit Regression is mainly used in specific disciplines** (e.g., economics, finance, psychometrics) where a **latent normal distribution assumption** is reasonable.

5. Discuss their activation functions: sigmoid for logistic and normal CDF for probit. How do these functions influence decision boundaries?

Activation Functions: Sigmoid vs. Normal CDF

The core difference between **Logistic Regression** and **Probit Regression** lies in their activation functions, which map a linear combination of inputs to probabilities.

---

1 Logistic Regression: Sigmoid Activation Function

The **sigmoid function** is used to map any real number to the range **(0,1)**, making it ideal for binary classification.

Key Properties:

- **S-shaped (sigmoidal)** curve that is smooth and differentiable.
- Symmetric about **z=0**.
- Has **longer tails** (gradual probability transitions).
- Approaches **0** and **1** asymptotically but never actually reaches them.

- Computationally efficient (requires only exponentiation).

Impact on Decision Boundaries:

- The **decision boundary** is where the probability is **exactly 0.5**.
- This results in a **linear decision boundary** in the feature space.
- Because the **sigmoid function has longer tails**, it is **more robust to extreme values and outliers**, giving it an advantage when handling datasets with noise.

---

2 Probit Regression: Normal CDF Activation Function

Probit Regression uses the **cumulative distribution function (CDF) of the standard normal distribution**:

Key Properties:

- **S-shaped curve**, like the sigmoid function.
- More **sensitive to small changes** in input values near **z=0**.
- **Shorter tails**, meaning probability transitions occur more abruptly.
- Requires computing the integral of a Gaussian, which is **computationally more expensive** than the sigmoid.

Impact on Decision Boundaries:

- This again results in a **linear decision boundary**, like Logistic Regression.
- However, due to the **steeper slope around z=0** and **shorter tails**, Probit Regression is **less robust to extreme values**.

Comparison of Decision Boundaries

Feature	**Logistic (Sigmoid)**	**Probit (Normal CDF)**
Function Shape	Smooth, S-shaped sigmoid	Smooth, S-shaped normal CDF

Feature	Logistic (Sigmoid)	Probit (Normal CDF)
Tail Behavior	Longer tails, more gradual probability transition	Shorter tails, probability transitions occur more sharply
Decision Boundary	Linear, centered at (w^T)X=0	Linear, centered at (w^T)X=0
Robustness	More robust to extreme values and outliers	More sensitive to extreme values
Computation	Fast (requires exponentiation)	Slower (requires Gaussian integral computation)

Key Takeaways

1. **Both models produce linear decision boundaries**, meaning they classify based on whether (w^T)X is positive or negative.
2. **Logistic Regression (sigmoid function) is more robust** due to its **longer tails**, making it the preferred choice when extreme values or outliers are present.
3. **Probit Regression (normal CDF) has steeper transitions**, making it more suitable when the **latent variable assumption of normality holds** (e.g., in financial or psychological models).
4. **In practice, both models perform very similarly**, and the choice between them is often based on computational efficiency and interpretability rather than classification accuracy.

6. Define and explain the significance of Confusion Matrices, ROC Curves, and AUC in evaluating classification models. How do they contribute to model selection?

Key Evaluation Metrics for Classification Models

When evaluating classification models like **Logistic Regression** and **Probit**

Regression, three key metrics are commonly used:

1. **Confusion Matrix**
2. **ROC Curve (Receiver Operating Characteristic Curve)**
3. **AUC (Area Under the Curve)**

These metrics help assess the model's performance, interpret its predictions, and guide **model selection**.

□Confusion Matrix: Understanding Classification Performance

A **confusion matrix** is a table that summarizes the **correct and incorrect** predictions of a classification model.

Structure of a Confusion Matrix

	Predicted 0	**Predicted 1**
Actual 0 (Negative)	**True Negative (TN)**	**False Positive (FP)**
Actual 1 (Positive)	**False Negative (FN)**	**True Positive (TP)**

Significance of the Confusion Matrix

- It **breaks down prediction errors** into **False Positives (FP)** and **False Negatives (FN)**.
- Helps compute other important metrics:
  - **Accuracy** = (TP+TN)/(TP+TN+FP+FN)
  - **Precision (Positive Predictive Value)** = TP/(TP+FP) → How many predicted positives are actually positive?
  - **Recall (Sensitivity, True Positive Rate)** = TP/(TP+FN) → How many actual positives were correctly predicted?
  - **F1 Score** = 2×Precision×Recall/(Precision+Recall) → A balance between precision and recall.

Contribution to Model Selection

- **If FP is costly (e.g., fraud detection), choose a model with higher Precision.**
- **If FN is costly (e.g., medical diagnosis), choose a model with higher Recall.**
- **F1 Score is useful when both FP and FN need to be minimized equally.**

2□ROC Curve: Visualizing Model Performance

The **ROC (Receiver Operating Characteristic) curve** is a graphical tool used to **compare classification models** based on different threshold values.

Definition

The **ROC curve plots:**

- **True Positive Rate (TPR) (Recall)**: TP/(TP+FN)   (Sensitivity)
- **False Positive Rate (FPR)**: FP/(FP+TN)

Each point on the ROC curve represents a different **decision threshold** (e.g., 0.3, 0.5, 0.7, etc.).

Interpreting the ROC Curve

- The **closer the curve is to the top-left corner**, the better the model is at distinguishing between classes.
- The **diagonal line (y=x)** represents random guessing.

Contribution to Model Selection

- **If Model A's ROC curve is above Model B's curve, Model A is better.**
- **Higher TPR at the same FPR means a better model.**

---

3 AUC: Measuring Overall Model Quality

Definition

- The **AUC (Area Under the Curve)** is a single number summarizing the ROC curve.
- **AUC values range from 0 to 1**:
  - **AUC = 1.0** → Perfect classifier
  - **AUC = 0.5** → Random guessing (no predictive power)
  - **AUC < 0.5** → Worse than random guessing

Significance

- **AUC tells us how well a model separates positive and negative classes across all thresholds.**
- **Higher AUC = Better overall model performance.**
- AUC is useful when class distribution is imbalanced because it does not depend on absolute accuracy.

Contribution to Model Selection

- **If Model A has a higher AUC than Model B, Model A is better at distinguishing classes.**
- **For imbalanced datasets, choose the model with the highest AUC.**
- **When both models have similar AUCs, consider the confusion matrix for further refinement.**

📌 Summary: Choosing the Best Model

Metric	**Measures**	**Best for Selecting Model When…**
Confusion Matrix	Actual vs. predicted results (TP, TN, FP, FN)	You want detailed **error analysis** (e.g., FN vs. FP trade-offs).
ROC Curve	Trade-off between True Positive Rate & False Positive Rate	You want to visualize how different models handle **classification thresholds**.

Metric	**Measures**	**Best for Selecting Model When…**
AUC	Overall model quality across all thresholds	You need a **single performance metric**, especially for imbalanced datasets.

Practical Model Selection Strategy

✅ **If you need high precision (avoiding FP)** → Use **Confusion Matrix + Precision Score**

✅ **If you need high recall (avoiding FN)** → Use **Confusion Matrix + Recall Score**

✅ **If you need the best overall classifier** → Choose the model with the **highest AUC**

✅ **If you need threshold tuning** → Use **ROC Curve** to find the best trade-off