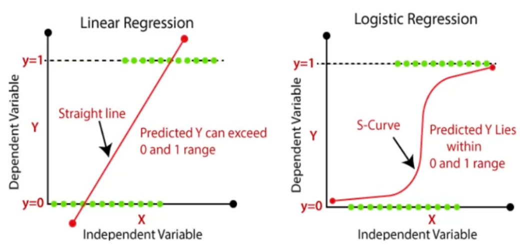
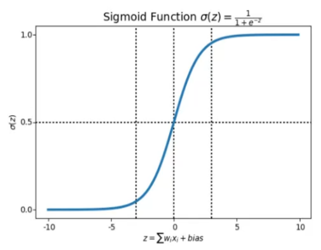
Complete Logistic Regression

January 12, 2026

Logistic Regression

- **Logistic Regression**: It is commonly used to estimate the probability that an instance belongs to a particular class (e.g., what is the probability that this email is spam?).
- **Softmax Regression**: It is a generalization of logistic regression to support multiple classes directly, without having to train and combine multiple binary classifiers.





Logistic Regression Cost Function

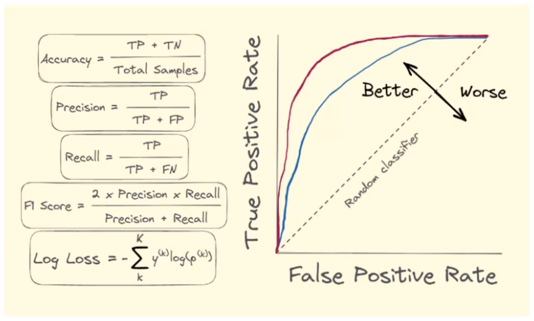
- **Log Loss (Binary Cross Entropy)**: The cost function used in logistic regression. It penalizes the model when it estimates a low probability for a positive instance or a high probability for a negative instance.
  - Formula: $J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log(\hat{p}^{(i)}) + (1-y^{(i)})\log(1-\hat{p}^{(i)})]$
  - where θ is the model's parameter vector, m is the number of instances in the dataset, $y^{(i)}$ is the target probability that the i^{th} instance is positive, and $\hat{p}^{(i)}$ is the model's estimated probability that the i^{th} instance is positive.
- **Log Loss for Multiclass Classification (Cross Entropy)**: The cost function used in softmax regression. It penalizes the model when it estimates a low probability for the target class.
  - Formula: $J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)}\log(\hat{p}_k^{(i)})$
  - where K is the number of classes, $y_k^{(i)}$ is the target probability that the i^{th} instance belongs to class k, and $\hat{p}_k^{(i)}$ is the model's estimated probability that the i^{th} instance belongs to class k.

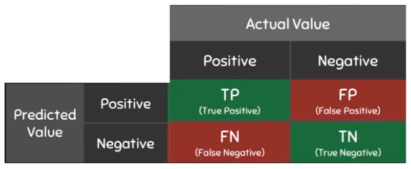
Linear Regression vs. Logistic Regression

Aspect	Linear Regression	Logistic Regression
Type of Problem	Regression	Classification
Output	Continuous values	Probabilities (between 0 and 1) or binary outcomes (0 or 1)
Cost Function	Mean Squared Error (MSE)	Log Loss (Binary Cross Entropy) or Cross Entropy (for multiclass classification)
Hypothesis Function	$h_\theta(x) = \theta^T \cdot x$	$h_\theta(x) = \sigma(\theta^T \cdot x)$
Activation Function	None	Sigmoid function ($\sigma(z) = \frac{1}{1+e^{-z}}$)
Loss Function	$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}(\theta^T \cdot x^{(i)} - y^{(i)})^2$	$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log(\hat{p}^{(i)}) + (1-y^{(i)})\log(1-\hat{p}^{(i)})]$
Decision Boundary	No decision boundary (output is a continuous value)	Decision boundary (output is a probability or binary outcome)
Model Evaluation	RMSE, MAE, etc.	Accuracy, Precision, Recall, F1 Score, etc.
Example	Predicting house prices, stock prices, etc.	Predicting whether an email is spam or not, whether a customer will buy a product or not, etc.
Example Dataset	Housing prices dataset, stock prices dataset, etc.	Email spam dataset, customer purchase dataset, etc.
Example Libraries	Scikit-Learn, TensorFlow, PyTorch, etc.	Scikit-Learn, TensorFlow, PyTorch, etc.
Example Code	`from sklearn.linear_model import LinearRegression` `model = LinearRegression()` `model.fit(X, y)`	`from sklearn.linear_model import LogisticRegression` `model = LogisticRegression()` `model.fit(X, y)`
Example Prediction	`model.predict(X_new)`	`model.predict(X_new)`

Classification Metrics

- **Accuracy**: The ratio of correctly predicted instances to the total instances.
  - Formula: $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$
- **Precision**: The ratio of correctly predicted positive observations to the total predicted positives.
  - Formula: $Precision = \frac{TP}{TP+FP}$
- **Recall (Sensitivity)**: The ratio of correctly predicted positive observations to the all observations in actual class.
  - Formula: $Recall = \frac{TP}{TP+FN}$
- **F1 Score**: The weighted average of Precision and Recall.
  - Formula: $F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$


Accuracy = TP + TN / Total Samples
Precision = TP / TP + FP
Recall = TP / TP + FN
F1 Score = 2 x Precision x Recall / Precision + Recall
Log Loss = -Σ y^(k) log(p^(k))
True Positive Rate
False Positive Rate
Better
Worse
Random classifier



		Actual Value	
		Positive	Negative
Predicted Value	Positive	TP (True Positive)	FP (False Positive)
	Negative	FN (False Negative)	TN (True Negative)

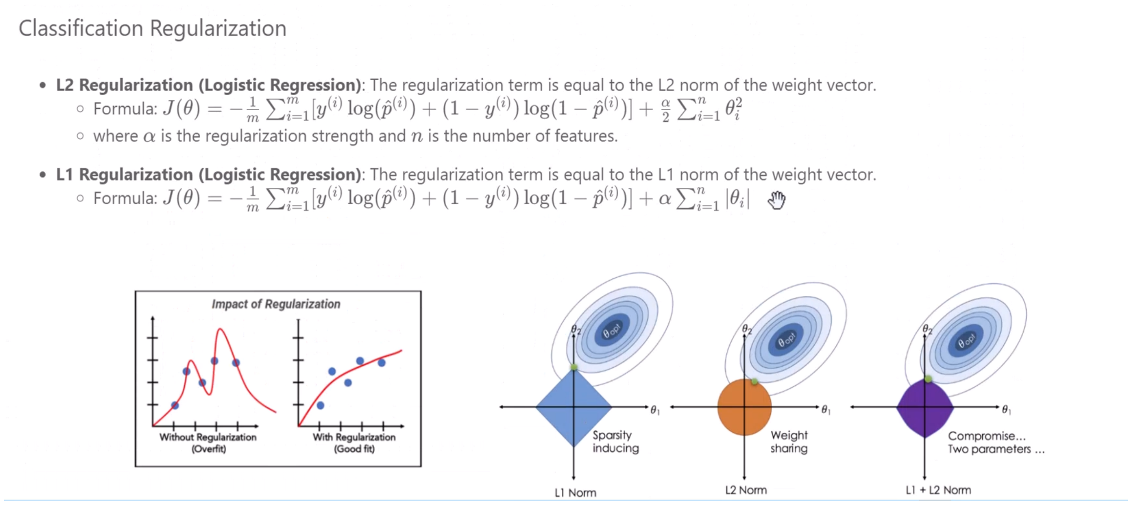
- True Positive (TP) : Observation is positive, and is predicted to be positive.
- False Negative (FN) : Observation is positive, but is predicted negative.
- True Negative (TN) : Observation is negative, and is predicted to be negative.
- False Positive (FP) : Observation is negative, but is predicted positive.

Regularization

- **Regularization**: A technique used to prevent overfitting by adding a penalty term to the cost function.
- **L2 Regularization (Ridge Regression)**: The regularization term is equal to the L2 norm of the weight vector.
  - Formula: $J(\theta) = MSE(X, h_\theta) + \alpha \frac{1}{2} \sum_{i=1}^{n} \theta_i^2$
- **L1 Regularization (Lasso Regression)**: The regularization term is equal to the L1 norm of the weight vector.
  - Formula: $J(\theta) = MSE(X, h_\theta) + \alpha \sum_{i=1}^{n} |\theta_i|$

Classification Regularization

- **L2 Regularization (Logistic Regression)**: The regularization term is equal to the L2 norm of the weight vector.
  - Formula: $J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} [y^{(i)} \log(\hat{p}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{p}^{(i)})] + \frac{\alpha}{2} \sum_{i=1}^{n} \theta_i^2$
  - where α is the regularization strength and n is the number of features.
- **L1 Regularization (Logistic Regression)**: The regularization term is equal to the L1 norm of the weight vector.
  - Formula: $J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} [y^{(i)} \log(\hat{p}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{p}^{(i)})] + \alpha \sum_{i=1}^{n} |\theta_i|$



Classification Regularization

- **L2 Regularization (Logistic Regression)**: The regularization term is equal to the L2 norm of the weight vector.
  - Formula: $J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} [y^{(i)} \log(\hat{p}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{p}^{(i)})] + \frac{\alpha}{2} \sum_{i=1}^{n} \theta_i^2$
  - where α is the regularization strength and n is the number of features.
- **L1 Regularization (Logistic Regression)**: The regularization term is equal to the L1 norm of the weight vector.
  - Formula: $J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} [y^{(i)} \log(\hat{p}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{p}^{(i)})] + \alpha \sum_{i=1}^{n} |\theta_i|$

```
[25]: import warnings
      warnings.filterwarnings('ignore')

      from sklearn.datasets import load_breast_cancer
      from sklearn.model_selection import train_test_split
      from sklearn.linear_model import LogisticRegression
      from sklearn.metrics import accuracy_score, f1_score, precision_score,␣
       ↪recall_score, confusion_matrix, roc_auc_score, classification_report,␣
       ↪roc_curve, precision_recall_curve, auc
```

```
[2]: data = load_breast_cancer()
     data.feature_names
```

```
[2]: array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
            'mean smoothness', 'mean compactness', 'mean concavity',
            'mean concave points', 'mean symmetry', 'mean fractal dimension',
            'radius error', 'texture error', 'perimeter error', 'area error',
            'smoothness error', 'compactness error', 'concavity error',
            'concave points error', 'symmetry error',
            'fractal dimension error', 'worst radius', 'worst texture',
            'worst perimeter', 'worst area', 'worst smoothness',
            'worst compactness', 'worst concavity', 'worst concave points',
            'worst symmetry', 'worst fractal dimension'], dtype='<U23')
```

```
[3]: data.keys()
```

```
[3]: dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names',
     'filename', 'data_module'])
```

```
[4]: X, y = data.data, data.target
```

```
[5]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
      ↪random_state=42)
```

```
[6]: model = LogisticRegression()
     model.fit(X_train, y_train)
```

```
[6]: LogisticRegression()
```

```
[7]: y_pred = model.predict(X_test)
```

```
[8]: con = confusion_matrix(y_test, y_pred)
     con
```

```
[8]: array([[ 59,   4],
            [  1, 107]], dtype=int64)
```

```
[9]: acc = accuracy_score(y_test, y_pred)
     acc
```

```
[9]: 0.9707602339181286
```

```
[10]: y_prob = model.predict_proba(X_test)[:,1]
```

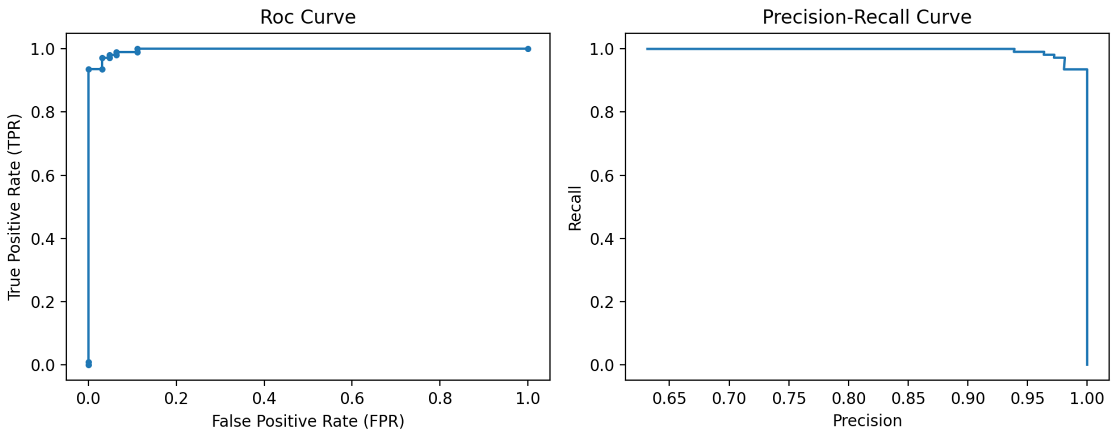
```
[12]: import pandas as pd

      pd.DataFrame([y_pred, y_prob]).T.head().style.format({1: "{:.9f}"})
```

```
[12]: <pandas.io.formats.style.Styler at 0x197df9d2790>
```

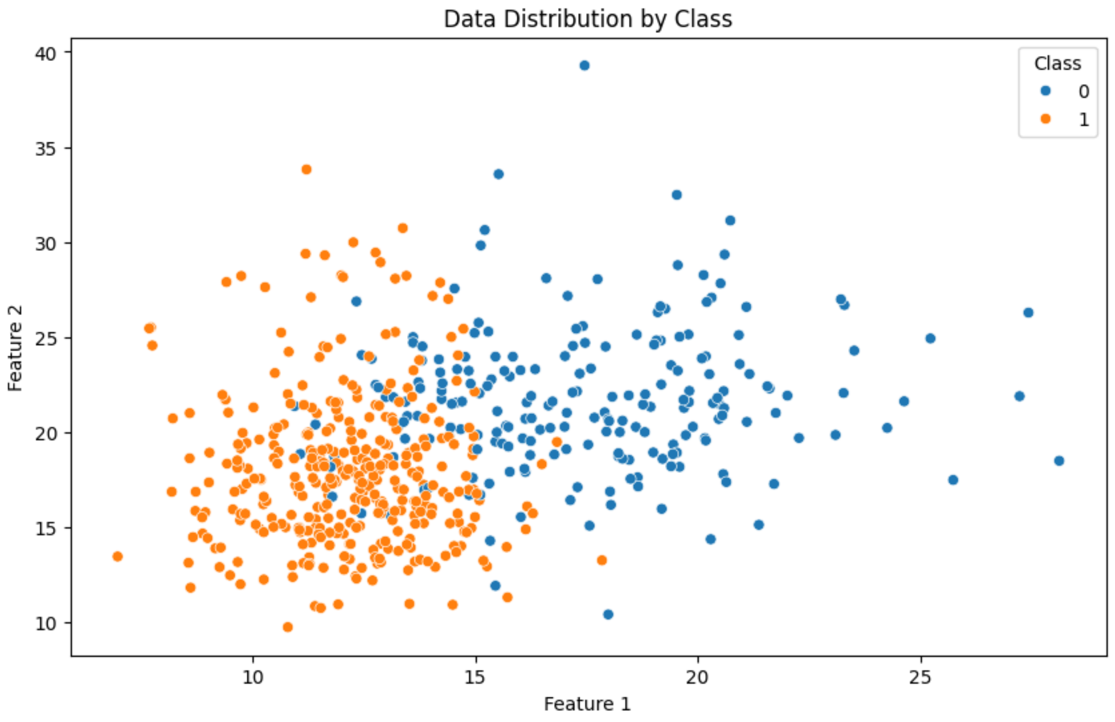
```
[22]: import matplotlib.pyplot as plt
      import seaborn as sns
```

[36]:
```
def evalueate_classification(y_true, y_pred, y_prob):
        print("confusion_matrix:\n", confusion_matrix(y_true, y_pred))
        print("classification_report:\n", classification_report(y_true, y_pred))

        roc_auc = roc_auc_score(y_true, y_prob)
        print("ROC and AUC Score:", roc_auc)

        fpr, tpr, _ = roc_curve(y_true, y_prob)
        precision, recall, _ = precision_recall_curve(y_true, y_prob)

        plt.figure(figsize = (10,4), dpi = 200)
        plt.subplot(1,2,1)
        plt.plot(fpr, tpr, marker = '.')
        plt.title("Roc Curve")
        plt.xlabel('False Positive Rate (FPR)')
        plt.ylabel('True Positive Rate (TPR)')



        plt.subplot(1,2,2)
        plt.plot(precision, recall)
        plt.title("Precision-Recall Curve")
        plt.xlabel('Precision')
        plt.ylabel('Recall')

        plt.tight_layout()
        plt.show()

evalueate_classification(y_test, y_pred, y_prob)
```

```
confusion_matrix:
 [[ 59   4]
 [  1 107]]
classification_report:
               precision    recall  f1-score   support

           0       0.98      0.94      0.96        63
           1       0.96      0.99      0.98       108

    accuracy                           0.97       171
   macro avg       0.97      0.96      0.97       171
weighted avg       0.97      0.97      0.97       171

ROC and AUC Score: 0.996766607877719
```



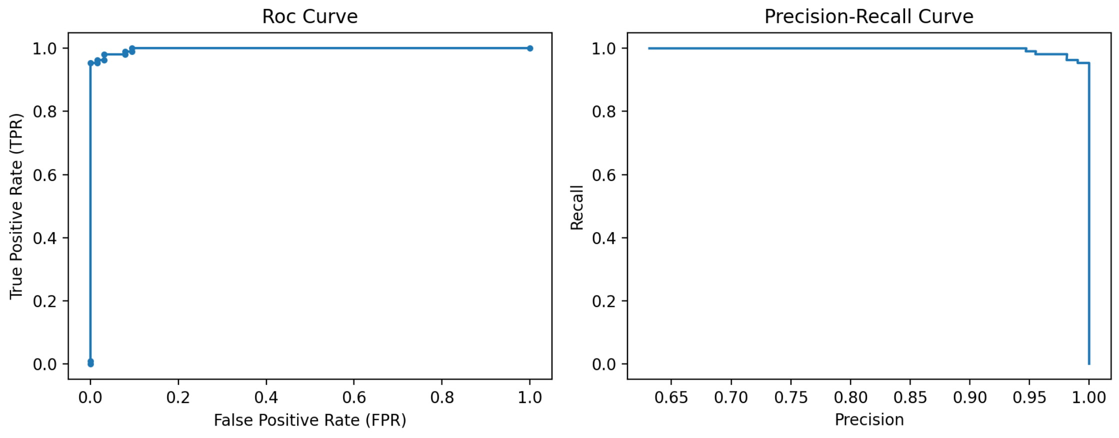
[38]:
```
plt.figure(figsize=(10, 6))
sns.scatterplot(x=pd.DataFrame(X)[0], y=pd.DataFrame(X)[1], hue=y )
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.title('Data Distribution by Class')
plt.legend(title='Class')
plt.show()
```



1 Regularization

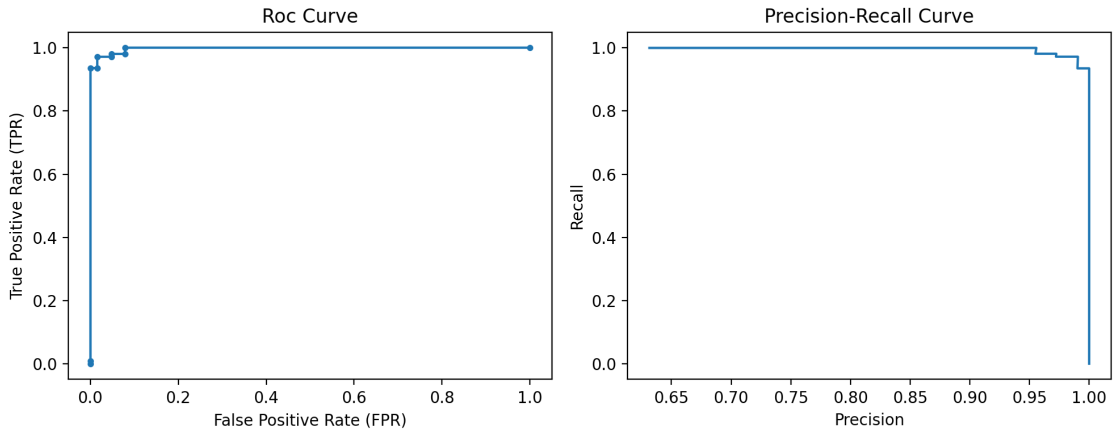
```
[40]: model = LogisticRegression(max_iter=10000, penalty='l2')
      model.fit(X_train, y_train)
      y_pred = model.predict(X_test)
      y_prob = model.predict_proba(X_test)[:,1]

      evalueate_classification(y_test, y_pred, y_prob)
```

```
confusion_matrix:
 [[ 61   2]
 [  2 106]]
classification_report:
               precision    recall  f1-score   support

           0       0.97      0.97      0.97        63
           1       0.98      0.98      0.98       108

    accuracy                           0.98       171
   macro avg       0.97      0.97      0.97       171
weighted avg       0.98      0.98      0.98       171

ROC and AUC Score: 0.9976484420928865
```



```
[42]: model = LogisticRegression(max_iter=10000, penalty='l1', solver='liblinear')
      model.fit(X_train, y_train)
      y_pred = model.predict(X_test)
      y_prob = model.predict_proba(X_test)[:,1]

      evalueate_classification(y_test, y_pred, y_prob)
```

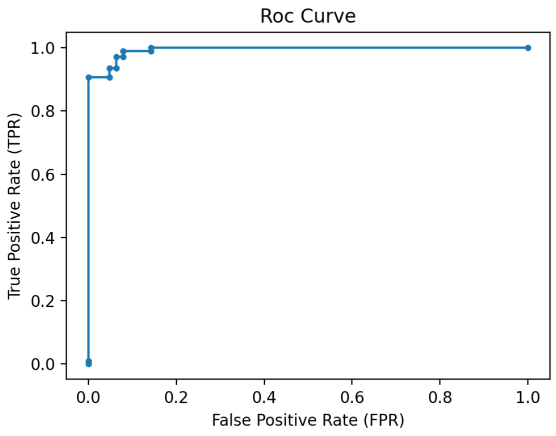
```
confusion_matrix:
 [[ 59   4]
 [  2 106]]
classification_report:
               precision    recall  f1-score   support

           0       0.97      0.94      0.95        63
           1       0.96      0.98      0.97       108

    accuracy                           0.96       171
   macro avg       0.97      0.96      0.96       171
weighted avg       0.96      0.96      0.96       171

ROC and AUC Score: 0.997501469723692
```

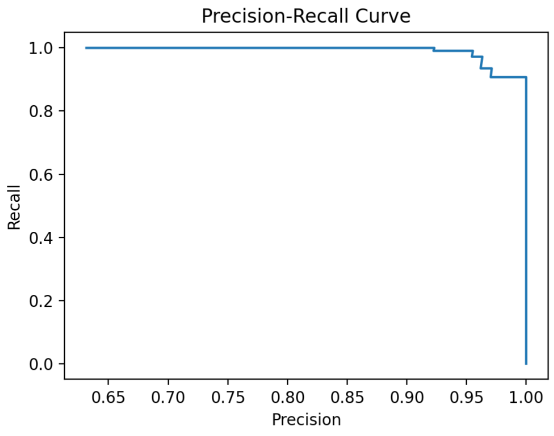


```
[44]: model = LogisticRegression(max_iter=10000, penalty='elasticnet', solver='saga',␣
        ↪l1_ratio=0.8)
      model.fit(X_train, y_train)
      y_pred = model.predict(X_test)
      y_prob = model.predict_proba(X_test)[:,1]

      evalueate_classification(y_test, y_pred, y_prob)
```

```
confusion_matrix:
 [[ 58   5]
 [  1 107]]
classification_report:
               precision    recall  f1-score   support

           0       0.98      0.92      0.95        63
           1       0.96      0.99      0.97       108

    accuracy                           0.96       171
   macro avg       0.97      0.96      0.96       171
```

```
weighted avg       0.97      0.96      0.96       171

ROC and AUC Score: 0.993533215755438
```


Roc Curve
True Positive Rate (TPR)
False Positive Rate (FPR)


Precision-Recall Curve
Recall
Precision