

Machine Learning Assignment

Logistic Regression

Question 1 (a)

```
In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score, roc_curve, auc
from sklearn.preprocessing import StandardScaler
```

```
In [2]: X_train = np.loadtxt("/Users/gaganullas19/Documents/Spring2024/AppliedMachineLearning/Homework_3/Gisette/gisette_train.data")
y_train = np.loadtxt("/Users/gaganullas19/Documents/Spring2024/AppliedMachineLearning/Homework_3/Gisette/gisette_train.labels")

X_test = np.loadtxt("/Users/gaganullas19/Documents/Spring2024/AppliedMachineLearning/Homework_3/Gisette/gisette_valid.data")
y_test = np.loadtxt("/Users/gaganullas19/Documents/Spring2024/AppliedMachineLearning/Homework_3/Gisette/gisette_valid.labels")
```

```
In [3]: # Converting -1 to 0 and keeping 1 as 1
y_train = (y_train + 1) // 2
y_test = (y_test + 1) // 2
```

```
In [4]: scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
In [5]: # Adding bias term
X_train = np.column_stack((np.ones(X_train.shape[0]), X_train))
X_test = np.column_stack((np.ones(X_test.shape[0]), X_test))
```

```
In [6]: X_train = X_train.astype(np.float64)
X_test = X_test.astype(np.float64)
```

```
In [7]: num_iterations = 300
lambda_value = 0.0001
n_samples, n_features = X_train.shape

w = np.zeros(n_features)
```

```
In [8]: train_loss = []
train_errors = []
test_errors = []
fpr_train, tpr_train, auc_train = [], [], []
fpr_test, tpr_test, auc_test = [], [], []
```

```
In [9]: def sigmoid(z):
    return 1.0 / (1.0 + np.exp(-z))
```

```
In [10]: for iteration in range(num_iterations):
    z = np.dot(X_train, w)
    predictions = sigmoid(z)

    epsilon = 1e-10
    predictions = np.clip(predictions, epsilon, 1 - epsilon)
```

```

gradient = np.dot(X_train.T, (predictions - y_train)) / n_samples + lambda_value * w

learning_rate = 0.5
w -= learning_rate * gradient

train_loss_value = -np.sum(y_train * np.log(predictions) + (1 - y_train) * np.log(1 - predictions))
train_loss.append(train_loss_value)

train_predictions = (predictions >= 0.5).astype(int)

train_error = 1 - accuracy_score(y_train, train_predictions)
train_errors.append(train_error)

fpr, tpr, _ = roc_curve(y_train, predictions)
roc_auc = auc(fpr, tpr)
fpr_train.append(fpr)
tpr_train.append(tpr)
auc_train.append(roc_auc)

z_test = np.dot(X_test, w)
test_predictions = (sigmoid(z_test) >= 0.5).astype(int)

test_error = 1 - accuracy_score(y_test, test_predictions)
test_errors.append(test_error)

fpr, tpr, _ = roc_curve(y_test, sigmoid(z_test))
roc_auc = auc(fpr, tpr)
fpr_test.append(fpr)
tpr_test.append(tpr)
auc_test.append(roc_auc)

plt.figure(figsize=(10, 6))
plt.plot(range(num_iterations), train_loss)
plt.title("Training loss vs. Iteration Number")
plt.xlabel("Iteration Number")
plt.ylabel("Training loss")
plt.grid(True)
plt.show()

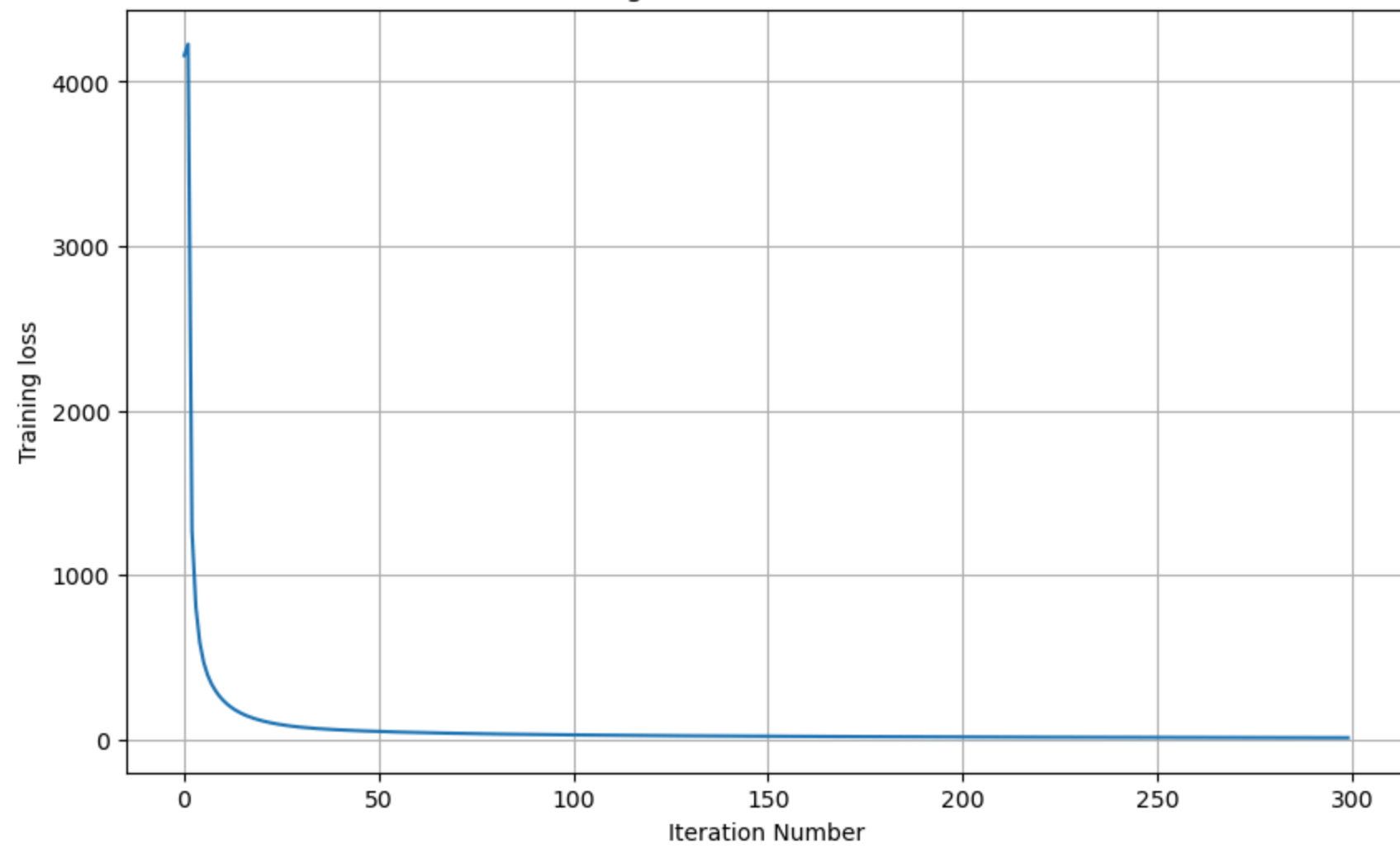
best_iteration = np.argmin(test_errors)
best_train_error = train_errors[best_iteration]
best_test_error = test_errors[best_iteration]

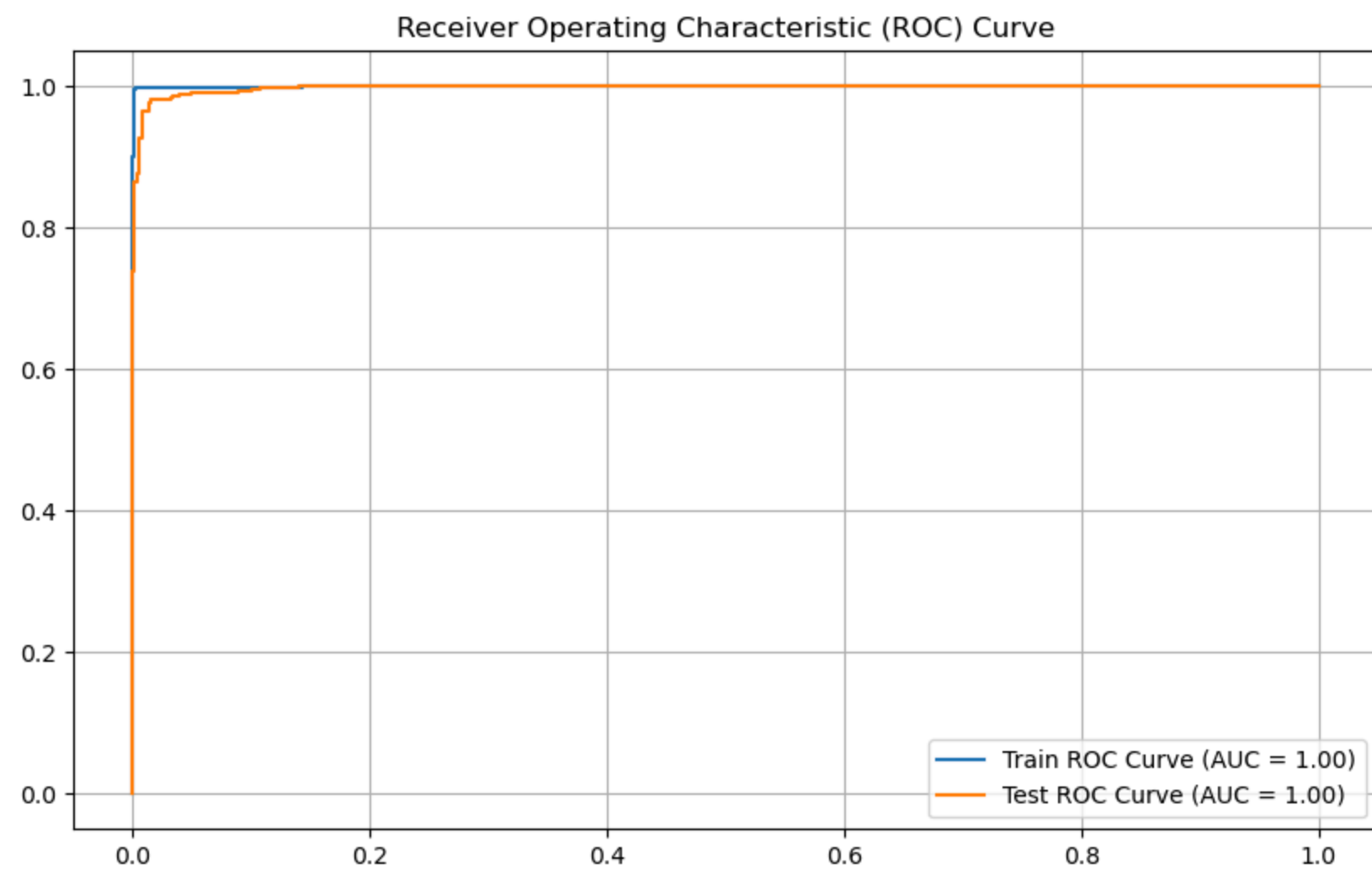
plt.figure(figsize=(10, 6))
plt.plot(fpr_train[best_iteration], tpr_train[best_iteration],
         label=f"Train ROC Curve (AUC = {auc_train[best_iteration]:.2f})")
plt.plot(fpr_test[best_iteration], tpr_test[best_iteration],
         label=f"Test ROC Curve (AUC = {auc_test[best_iteration]:.2f})")
plt.title("Receiver Operating Characteristic (ROC) Curve")
plt.legend()
plt.grid(True)
plt.show()

print(f"Training Misclassification Error: {best_train_error:.4f}")
print(f"Test Misclassification Error: {best_test_error:.4f}")

```

Training loss vs. Iteration Number





Training Misclassification Error: 0.0028
Test Misclassification Error: 0.0180

Question 1 (b)

```
In [11]: X_train = np.loadtxt("/Users/gaganullas19/Documents/Spring2024/AppliedMachineLearning/Homework_3/MADELON/madelon_train.data")
y_train = np.loadtxt("/Users/gaganullas19/Documents/Spring2024/AppliedMachineLearning/Homework_3/MADELON/madelon_train.labels")

X_test = np.loadtxt("/Users/gaganullas19/Documents/Spring2024/AppliedMachineLearning/Homework_3/MADELON/madelon_valid.data")
y_test = np.loadtxt("/Users/gaganullas19/Documents/Spring2024/AppliedMachineLearning/Homework_3/MADELON/madelon_valid.labels")
```

```
In [12]: # Converting -1 to 0 and keeping 1 as 1
y_train = (y_train + 1) // 2
y_test = (y_test + 1) // 2
```

```
In [13]: scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
In [14]: # Adding bias term
X_train = np.column_stack((np.ones(X_train.shape[0]), X_train))
X_test = np.column_stack((np.ones(X_test.shape[0]), X_test))
```

```
In [15]: X_train = X_train.astype(np.float64)
X_test = X_test.astype(np.float64)
```

```
In [16]: num_iterations = 300
lambda_value = 0.0001
n_samples, n_features = X_train.shape
```

```
w = np.zeros(n_features)
```

```
In [17]: train_loss = []
train_errors = []
test_errors = []
fpr_train, tpr_train, auc_train = [], [], []
fpr_test, tpr_test, auc_test = [], [], []
```

```
In [18]: def sigmoid(z):
return 1.0 / (1.0 + np.exp(-z))
```

```
In [19]: for iteration in range(num_iterations):
z = np.dot(X_train, w)
predictions = sigmoid(z)

epsilon = 1e-10
predictions = np.clip(predictions, epsilon, 1 - epsilon)

gradient = np.dot(X_train.T, (predictions - y_train)) / n_samples + lambda_value * w

learning_rate = 0.01
w -= learning_rate * gradient

train_loss_value = -np.sum(y_train * np.log(predictions) + (1 - y_train) * np.log(1 - predictions))
train_loss.append(train_loss_value)

train_predictions = (predictions >= 0.5).astype(int)

train_error = 1 - accuracy_score(y_train, train_predictions)
train_errors.append(train_error)

fpr, tpr, _ = roc_curve(y_train, predictions)
roc_auc = auc(fpr, tpr)
fpr_train.append(fpr)
tpr_train.append(tpr)
auc_train.append(roc_auc)

z_test = np.dot(X_test, w)
test_predictions = (sigmoid(z_test) >= 0.5).astype(int)

test_error = 1 - accuracy_score(y_test, test_predictions)
test_errors.append(test_error)

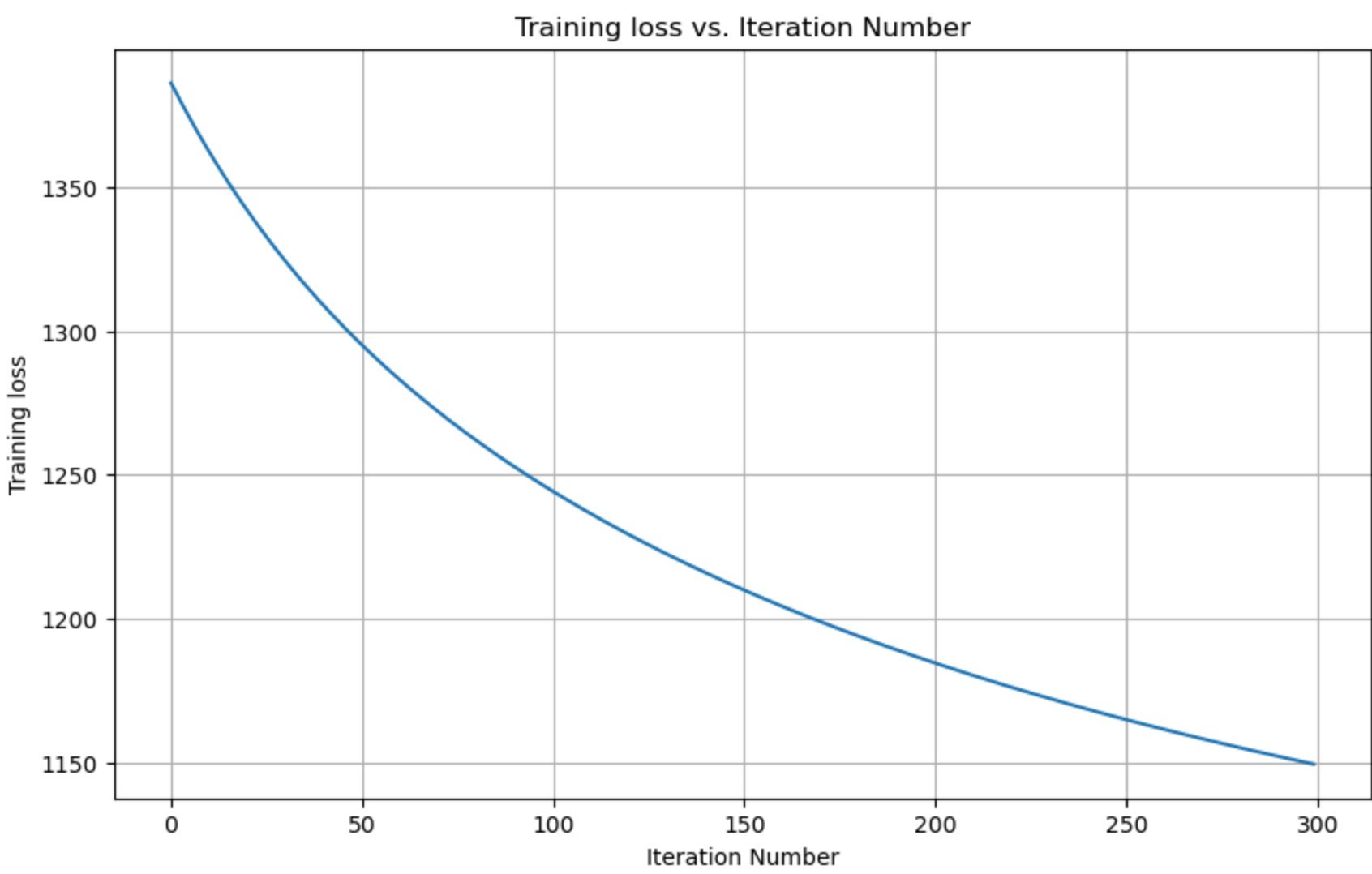
fpr, tpr, _ = roc_curve(y_test, sigmoid(z_test))
roc_auc = auc(fpr, tpr)
fpr_test.append(fpr)
tpr_test.append(tpr)
auc_test.append(roc_auc)

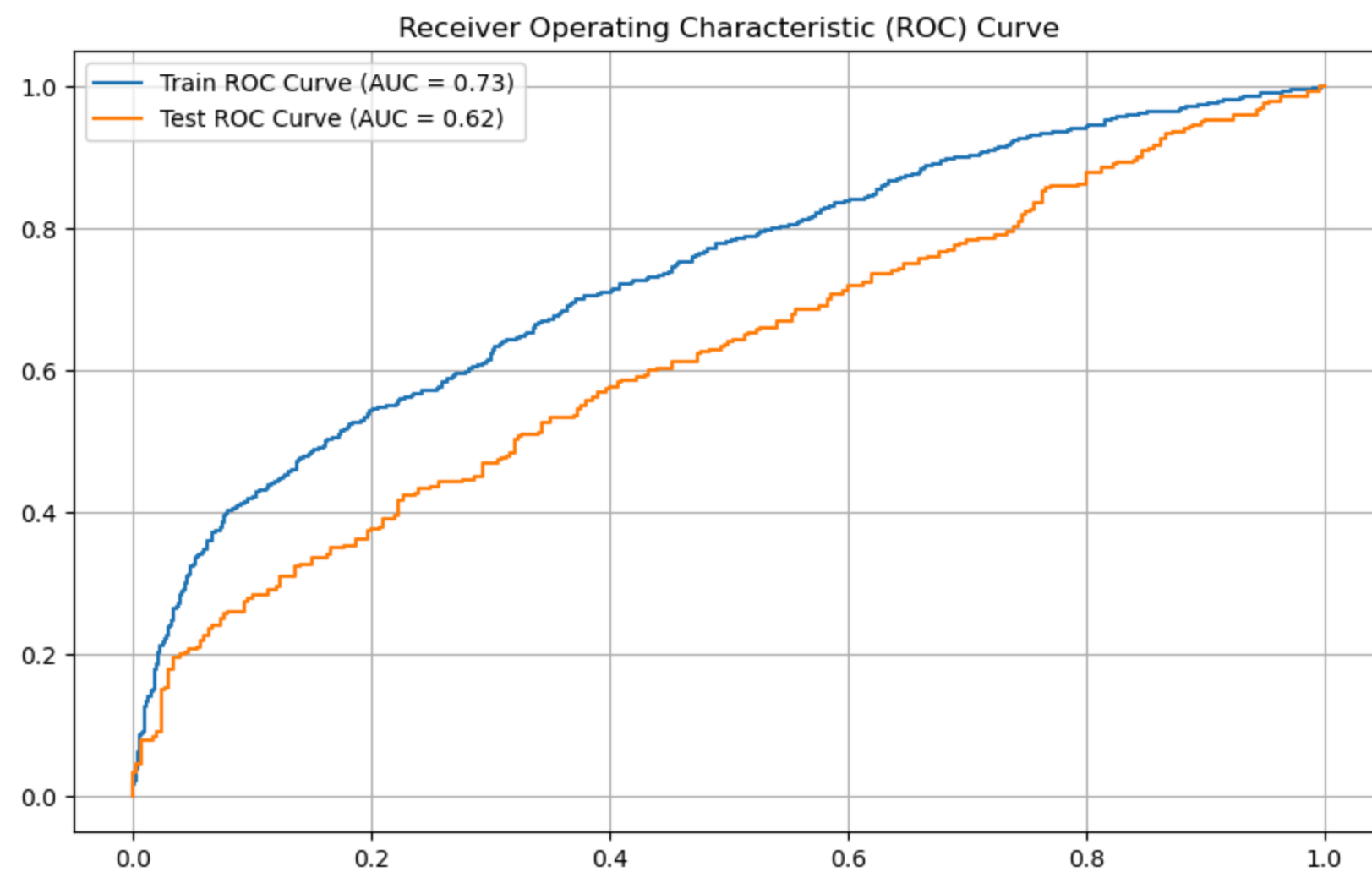
plt.figure(figsize=(10, 6))
plt.plot(range(num_iterations), train_loss)
plt.title("Training loss vs. Iteration Number")
plt.xlabel("Iteration Number")
plt.ylabel("Training loss")
plt.grid(True)
plt.show()

best_iteration = np.argmin(test_errors)
best_train_error = train_errors[best_iteration]
best_test_error = test_errors[best_iteration]
```

```
plt.figure(figsize=(10, 6))
plt.plot(fpr_train[best_iteration], tpr_train[best_iteration],
         label=f"Train ROC Curve (AUC = {auc_train[best_iteration]:.2f})")
plt.plot(fpr_test[best_iteration], tpr_test[best_iteration],
         label=f"Test ROC Curve (AUC = {auc_test[best_iteration]:.2f})")
plt.title("Receiver Operating Characteristic (ROC) Curve")
plt.legend()
plt.grid(True)
plt.show()

print(f"Training Misclassification Error: {best_train_error:.4f}")
print(f"Test Misclassification Error: {best_test_error:.4f}")
```





Training Misclassification Error: 0.3400
Test Misclassification Error: 0.4117

Question 1 (c)

```
In [20]: X_train = np.genfromtxt("/Users/gaganullas19/Documents/Spring2024/AppliedMachineLearning/Homework_3/dexter/dexter_train.csv", delimiter=',')
y_train = np.genfromtxt("/Users/gaganullas19/Documents/Spring2024/AppliedMachineLearning/Homework_3/dexter/dexter_train.labels")

X_test = np.genfromtxt("/Users/gaganullas19/Documents/Spring2024/AppliedMachineLearning/Homework_3/dexter/dexter_valid.csv", delimiter=',')
y_test = np.genfromtxt("/Users/gaganullas19/Documents/Spring2024/AppliedMachineLearning/Homework_3/dexter/dexter_valid.labels")
```

```
In [21]: # Converting -1 to 0 and keeping 1 as 1
y_train = (y_train + 1) // 2
y_test = (y_test + 1) // 2
```

```
In [22]: scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
In [23]: # Adding bias term
X_train = np.column_stack((np.ones(X_train.shape[0]), X_train))
X_test = np.column_stack((np.ones(X_test.shape[0]), X_test))
```

```
In [24]: X_train = X_train.astype(np.float64)
X_test = X_test.astype(np.float64)
```

```
In [25]: num_iterations = 300
lambda_value = 0.0001
n_samples, n_features = X_train.shape
```

```
w = np.zeros(n_features)
```

```
In [26]: train_loss = []
train_errors = []
test_errors = []
fpr_train, tpr_train, auc_train = [], [], []
fpr_test, tpr_test, auc_test = [], [], []
```

```
In [27]: def sigmoid(z):
return 1.0 / (1.0 + np.exp(-z))
```

```
In [28]: for iteration in range(num_iterations):
z = np.dot(X_train, w)
predictions = sigmoid(z)

epsilon = 1e-10
predictions = np.clip(predictions, epsilon, 1 - epsilon)

gradient = np.dot(X_train.T, (predictions - y_train)) / n_samples + lambda_value * w

learning_rate = 0.001
w -= learning_rate * gradient

train_loss_value = -np.sum(y_train * np.log(predictions) +
                           (1 - y_train) * np.log(1 - predictions))
train_loss.append(train_loss_value)

train_predictions = (predictions >= 0.5).astype(int)

train_error = 1 - accuracy_score(y_train, train_predictions)
train_errors.append(train_error)

fpr, tpr, _ = roc_curve(y_train, predictions)
roc_auc = auc(fpr, tpr)
fpr_train.append(fpr)
tpr_train.append(tpr)
auc_train.append(roc_auc)

z_test = np.dot(X_test, w)
test_predictions = (sigmoid(z_test) >= 0.5).astype(int)

test_error = 1 - accuracy_score(y_test, test_predictions)
test_errors.append(test_error)

fpr, tpr, _ = roc_curve(y_test, sigmoid(z_test))
roc_auc = auc(fpr, tpr)
fpr_test.append(fpr)
tpr_test.append(tpr)
auc_test.append(roc_auc)

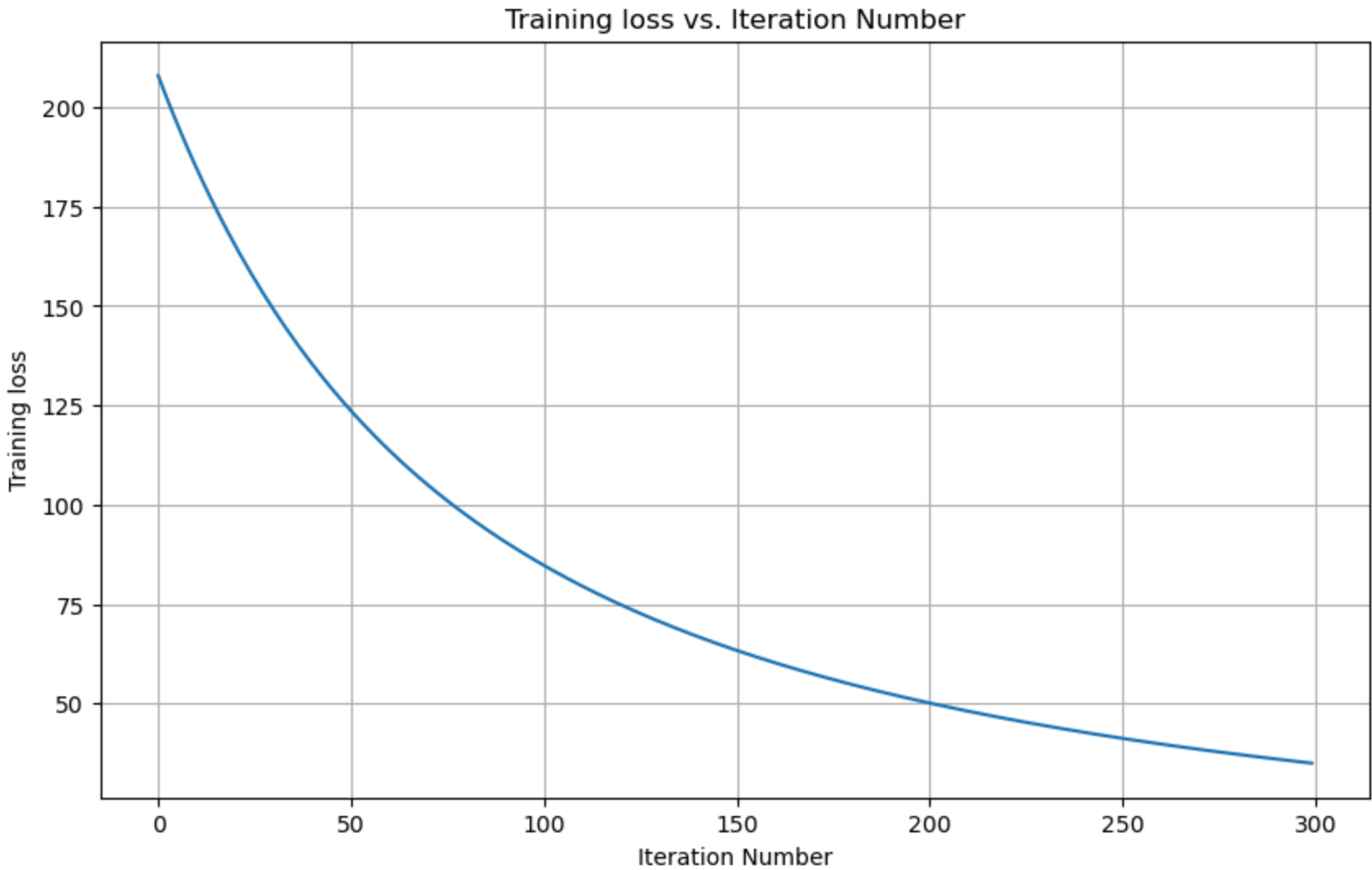
plt.figure(figsize=(10, 6))
plt.plot(range(num_iterations), train_loss)
plt.title("Training loss vs. Iteration Number")
plt.xlabel("Iteration Number")
plt.ylabel("Training loss")
plt.grid(True)
plt.show()

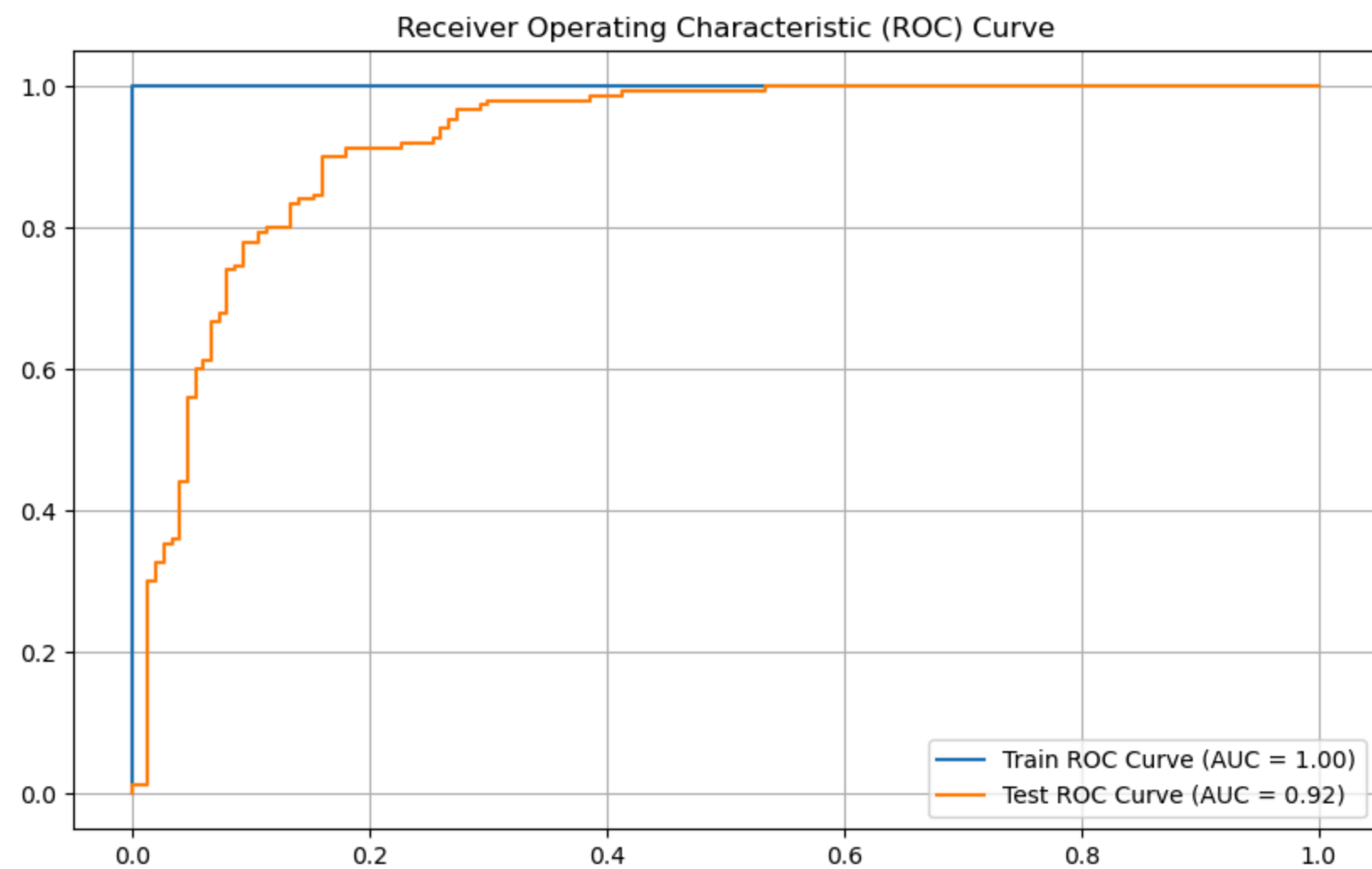
best_iteration = np.argmin(test_errors)
best_train_error = train_errors[best_iteration]
best_test_error = test_errors[best_iteration]
```



```
plt.figure(figsize=(10, 6))
plt.plot(fpr_train[best_iteration], tpr_train[best_iteration],
         label=f"Train ROC Curve (AUC = {auc_train[best_iteration]:.2f})")
plt.plot(fpr_test[best_iteration], tpr_test[best_iteration],
         label=f"Test ROC Curve (AUC = {auc_test[best_iteration]:.2f})")
plt.title("Receiver Operating Characteristic (ROC) Curve")
plt.legend()
plt.grid(True)
plt.show()

print(f"Training Misclassification Error: {best_train_error:.4f}")
print(f"Test Misclassification Error: {best_test_error:.4f}")
```





Training Misclassification Error: 0.0000
 Test Misclassification Error: 0.1333

Question 2 (a)

```
In [29]: import pandas as pd
import numpy as np
```

```
In [30]: X_train = np.loadtxt("/Users/gaganullas19/Documents/Spring2024/AppliedMachineLearning/Homework_3/Gisette/gisette_train.data")
y_train = np.loadtxt("/Users/gaganullas19/Documents/Spring2024/AppliedMachineLearning/Homework_3/Gisette/gisette_train.labels")

X_test = np.loadtxt("/Users/gaganullas19/Documents/Spring2024/AppliedMachineLearning/Homework_3/Gisette/gisette_valid.data")
y_test = np.loadtxt("/Users/gaganullas19/Documents/Spring2024/AppliedMachineLearning/Homework_3/Gisette/gisette_valid.labels")
```

```
In [31]: # Converting -1 to 0 and keeping 1 as 1
y_train = (y_train + 1) // 2
y_test = (y_test + 1) // 2
```

```
In [32]: def minimize_loss(X, y, lambda_val):
    X_with_bias = np.hstack([X, np.ones((len(X), 1))])

    w = np.linalg.inv(X_with_bias.T @ X_with_bias + lambda_val * np.eye(X_with_bias.shape[1]))
    @ X_with_bias.T @ y
    return w[:-1], w[-1]

lambda_val = 0.0001
optimized_weights, optimized_bias = minimize_loss(X_train, y_train.astype(float), lambda_val)

# Predict on training and test sets
y_train_pred = np.sign(np.dot(X_train, optimized_weights) + optimized_bias)
```

```
y_test_pred = np.sign(np.dot(X_test, optimized_weights) + optimized_bias)

# Calculate misclassification error
train_error = np.mean(y_train_pred != y_train.astype(float))
test_error = np.mean(y_test_pred != y_test.astype(float))

print("Misclassification error on training set:", train_error)
print("Misclassification error on test set:", test_error)

Misclassification error on training set: 0.5013333333333333
Misclassification error on test set: 0.525
```

Question 2 (b)

```
In [33]: import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score, roc_curve, auc
```

```
In [34]: fpr_train, tpr_train, i = roc_curve(y_train.astype(int), y_train_pred)
roc_auc_train = auc(fpr_train, tpr_train)

fpr_test, tpr_test, i = roc_curve(y_test.astype(int), y_test_pred)
roc_auc_test = auc(fpr_test, tpr_test)

plt.figure(figsize=(8, 6))
plt.plot(fpr_train, tpr_train, color='green', lw=2, label='Train ROC curve (area = %0.2f)'
        % roc_auc_train)
plt.plot(fpr_test, tpr_test, color='red', lw=2, label='Test ROC curve (area = %0.2f)'
        % roc_auc_test)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc="lower right")
plt.show()
```

