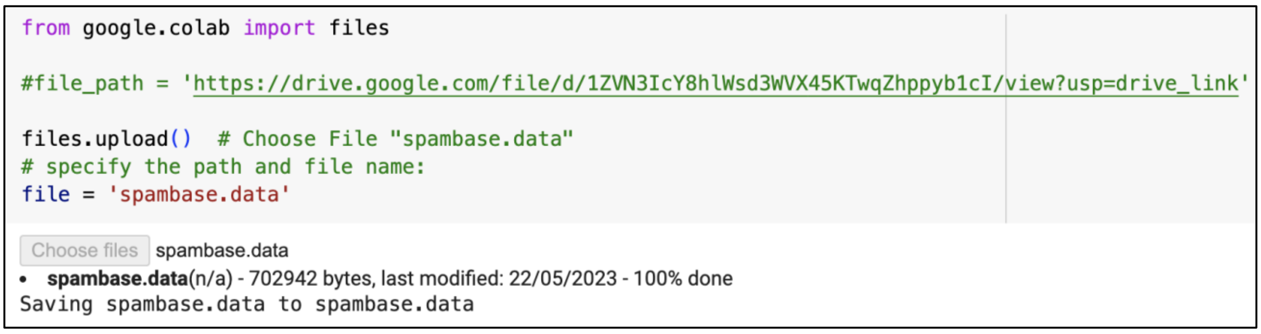
Screenshots of Sample runs:

Logistic Regression:

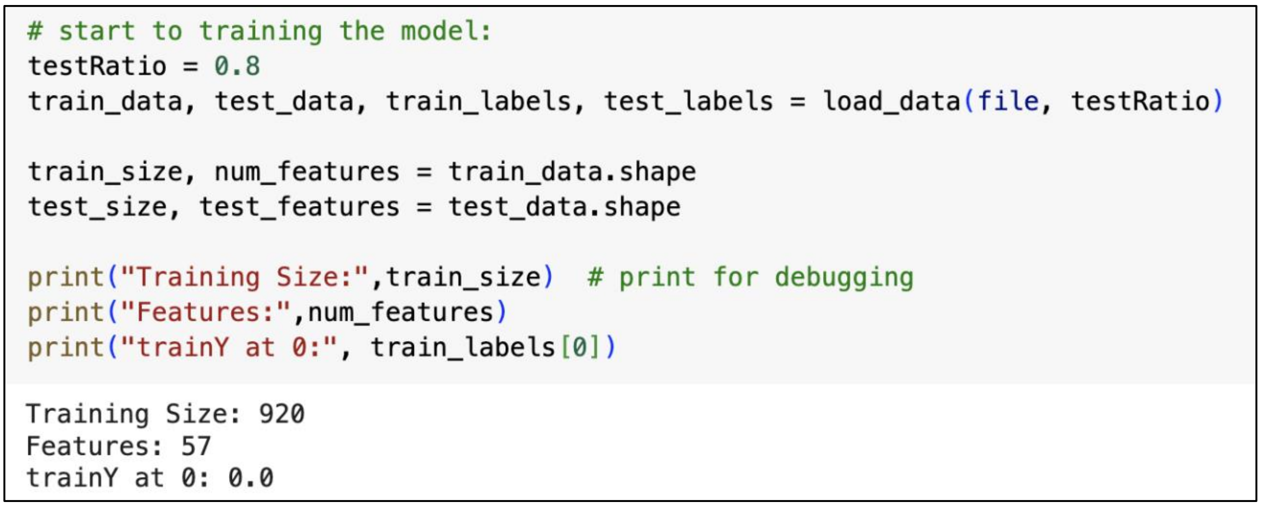
- Uploading the “spambase.data” to the collab



```
from google.colab import files

#file_path = 'https://drive.google.com/file/d/1ZVN3IcY8hlWsd3WVX45KTwqZhppyb1cI/view?usp=drive_link'

files.upload()  # Choose File "spambase.data"
# specify the path and file name:
file = 'spambase.data'
```

```
Choose files spambase.data
• spambase.data(n/a) - 702942 bytes, last modified: 22/05/2023 - 100% done
Saving spambase.data to spambase.data
```

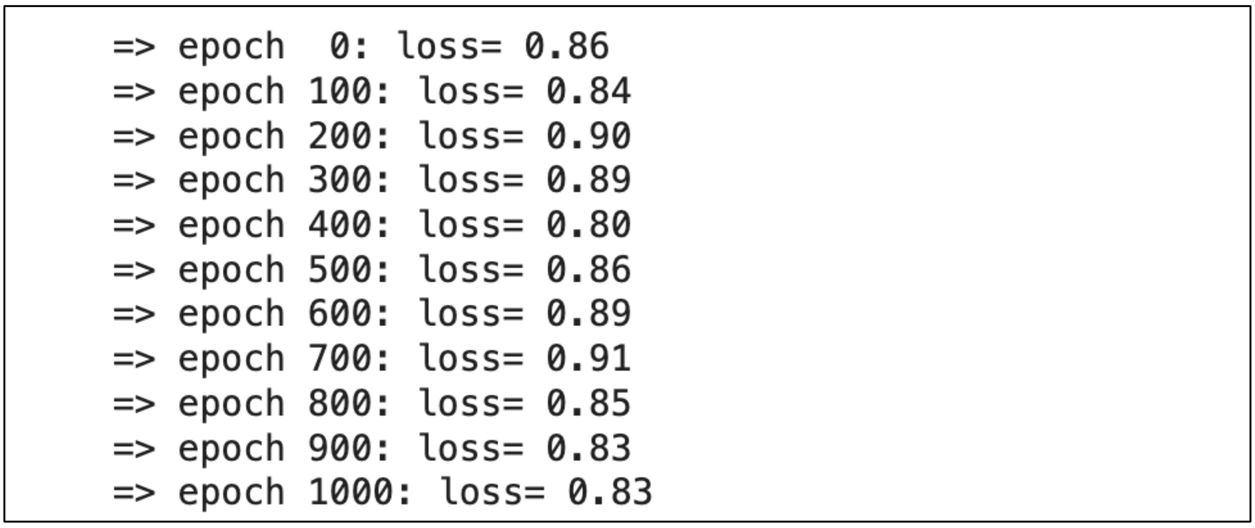
- Training the model



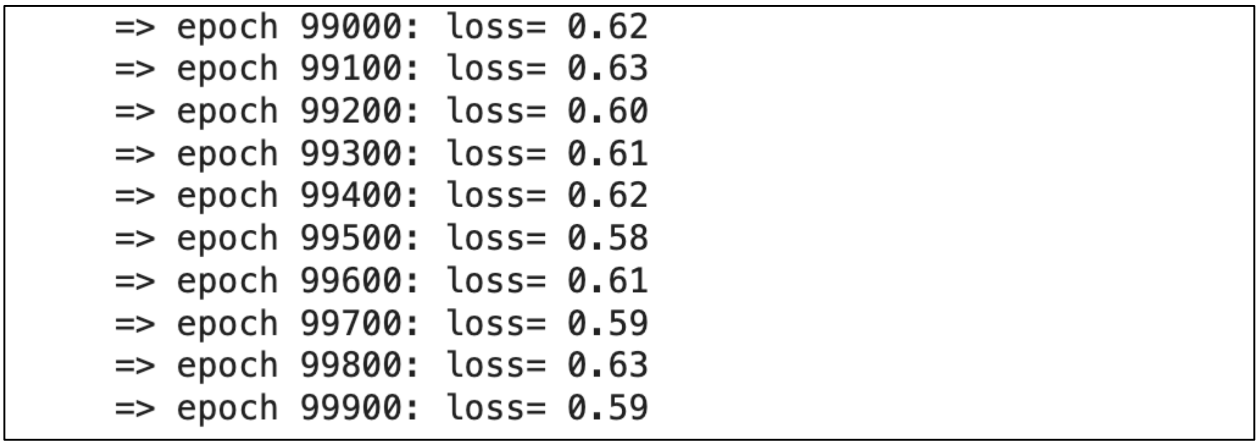
```
# start to training the model:
testRatio = 0.8
train_data, test_data, train_labels, test_labels = load_data(file, testRatio)

train_size, num_features = train_data.shape
test_size, test_features = test_data.shape

print("Training Size:",train_size)  # print for debugging
print("Features:",num_features)
print("trainY at 0:", train_labels[0])
```

```
Training Size: 920
Features: 57
trainY at 0: 0.0
```

- Training Loop:



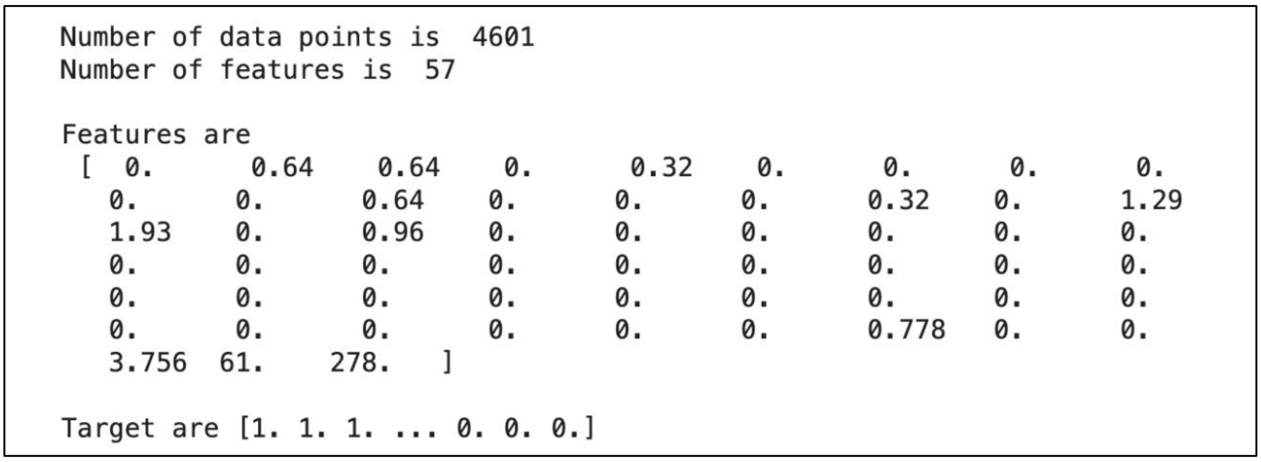
```
=> epoch  0: loss= 0.86
=> epoch 100: loss= 0.84
=> epoch 200: loss= 0.90
=> epoch 300: loss= 0.89
=> epoch 400: loss= 0.80
=> epoch 500: loss= 0.86
=> epoch 600: loss= 0.89
=> epoch 700: loss= 0.91
=> epoch 800: loss= 0.85
=> epoch 900: loss= 0.83
=> epoch 1000: loss= 0.83
```



```
=> epoch 99000: loss= 0.62
=> epoch 99100: loss= 0.63
=> epoch 99200: loss= 0.60
=> epoch 99300: loss= 0.61
=> epoch 99400: loss= 0.62
=> epoch 99500: loss= 0.58
=> epoch 99600: loss= 0.61
=> epoch 99700: loss= 0.59
=> epoch 99800: loss= 0.63
=> epoch 99900: loss= 0.59
```

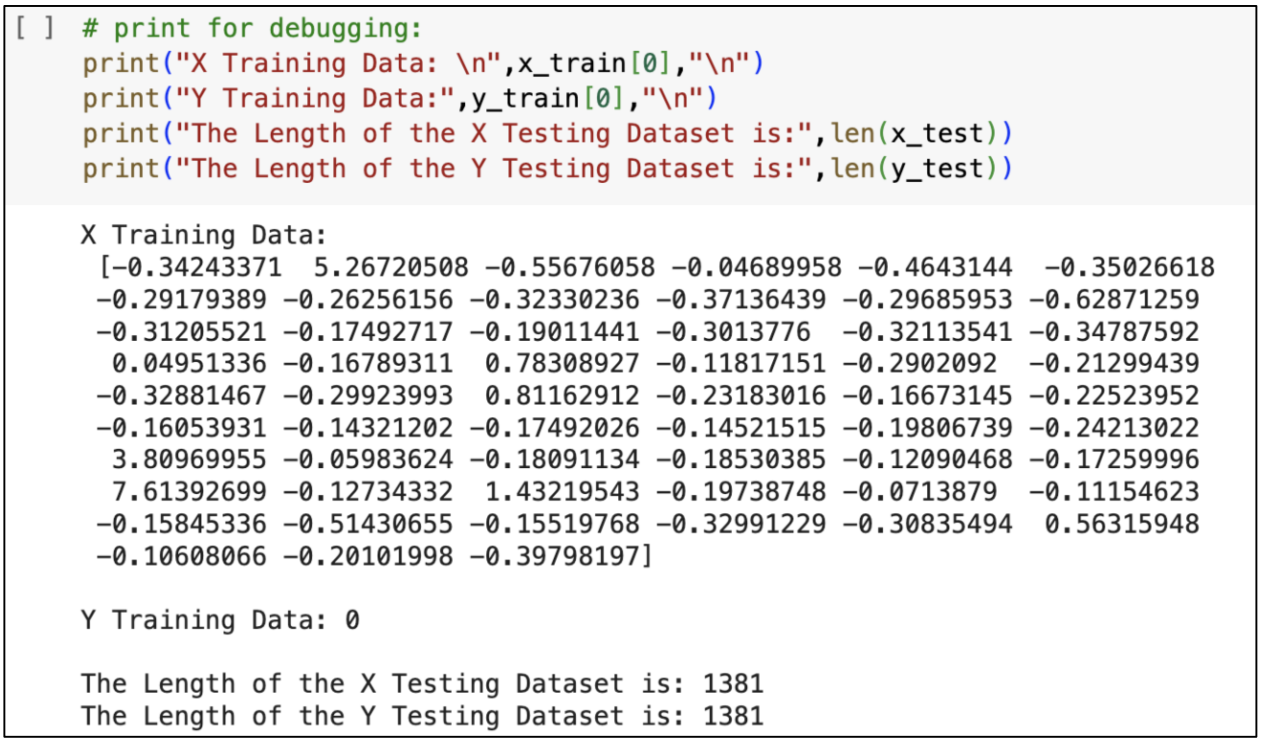
Support Vector Machine:

- Total number of data points available in the “spambase.data”



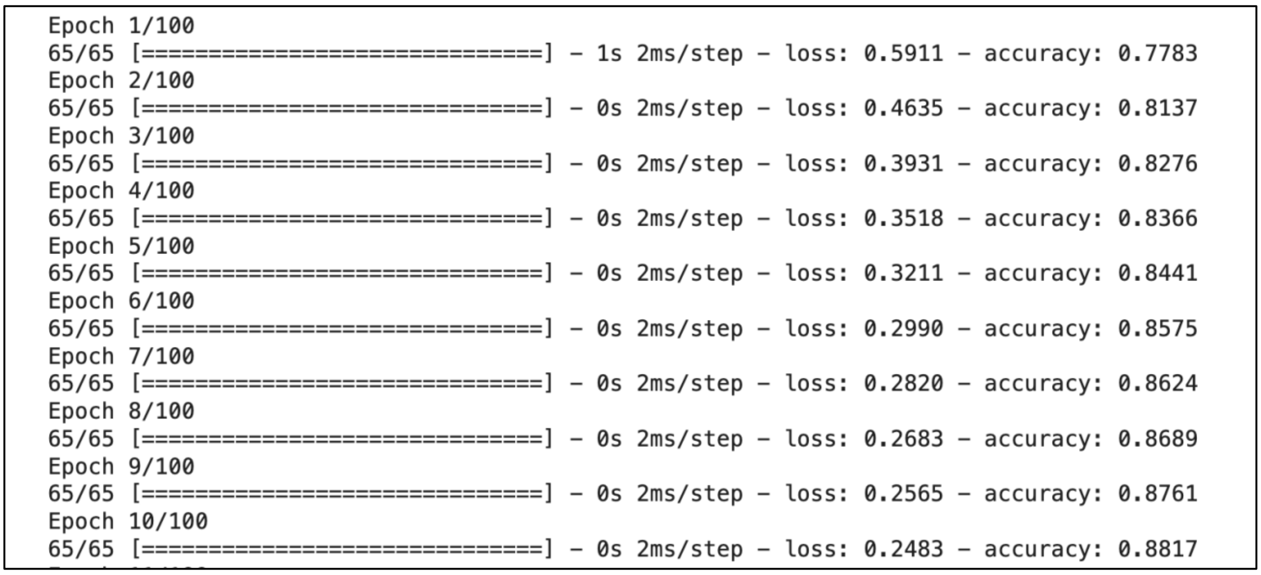
```
Number of data points is  4601
Number of features is  57

Features are
 [  0.      0.64   0.64   0.     0.32   0.     0.     0.     0.
   0.     0.     0.64   0.     0.     0.     0.32   0.     1.29
   1.93   0.     0.96   0.     0.     0.     0.     0.     0.
   0.     0.     0.     0.     0.     0.     0.     0.     0.
   0.     0.     0.     0.     0.     0.     0.     0.     0.
   0.     0.     0.     0.     0.     0.     0.778  0.     0.
   3.756  61.    278.    ]

Target are [1. 1. 1. ... 0. 0. 0.]
```

- Training & Testing Dataset:

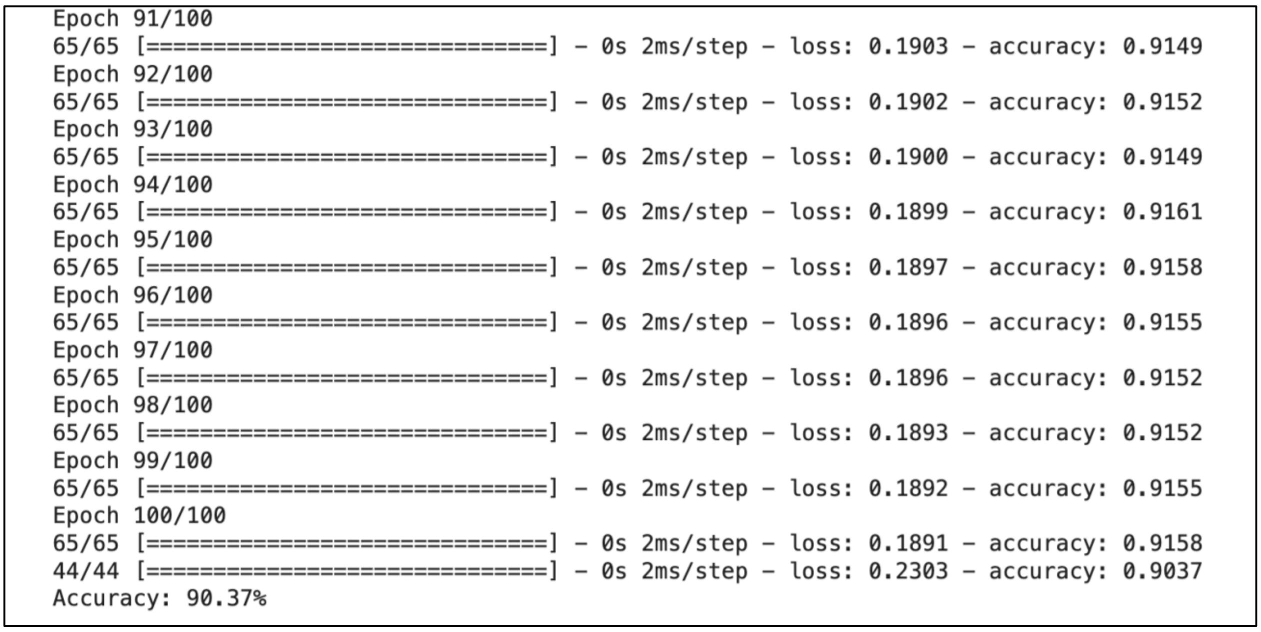


```
[ ]  # print for debugging:
     print("X Training Data: \n",x_train[0],"\n")
     print("Y Training Data:",y_train[0],"\n")
     print("The Length of the X Testing Dataset is:",len(x_test))
     print("The Length of the Y Testing Dataset is:",len(y_test))

     X Training Data:
      [-0.34243371  5.26720508 -0.55676058 -0.04689958 -0.4643144  -0.35026618
      -0.29179389 -0.26256156 -0.32330236 -0.37136439 -0.29685953 -0.62871259
      -0.31205521 -0.17492717 -0.19011441 -0.3013776  -0.32113541 -0.34787592
       0.04951336 -0.16789311  0.78308927 -0.11817151 -0.2902092  -0.21299439
      -0.32881467 -0.29923993  0.81162912 -0.23183016 -0.16673145 -0.22523952
      -0.16053931 -0.14321202 -0.17492026 -0.14521515 -0.19806739 -0.24213022
       3.80969955 -0.05983624 -0.18091134 -0.18530385 -0.12090468 -0.17259996
       7.61392699 -0.12734332  1.43219543 -0.19738748 -0.0713879  -0.11154623
      -0.15845336 -0.51430655 -0.15519768 -0.32991229 -0.30835494  0.56315948
      -0.10608066 -0.20101998 -0.39798197]

     Y Training Data: 0

     The Length of the X Testing Dataset is: 1381
     The Length of the Y Testing Dataset is: 1381
```

- Running the Training loop:



```
Epoch 1/100
65/65 [==============================] - 1s 2ms/step - loss: 0.5911 - accuracy: 0.7783
Epoch 2/100
65/65 [==============================] - 0s 2ms/step - loss: 0.4635 - accuracy: 0.8137
Epoch 3/100
65/65 [==============================] - 0s 2ms/step - loss: 0.3931 - accuracy: 0.8276
Epoch 4/100
65/65 [==============================] - 0s 2ms/step - loss: 0.3518 - accuracy: 0.8366
Epoch 5/100
65/65 [==============================] - 0s 2ms/step - loss: 0.3211 - accuracy: 0.8441
Epoch 6/100
65/65 [==============================] - 0s 2ms/step - loss: 0.2990 - accuracy: 0.8575
Epoch 7/100
65/65 [==============================] - 0s 2ms/step - loss: 0.2820 - accuracy: 0.8624
Epoch 8/100
65/65 [==============================] - 0s 2ms/step - loss: 0.2683 - accuracy: 0.8689
Epoch 9/100
65/65 [==============================] - 0s 2ms/step - loss: 0.2565 - accuracy: 0.8761
Epoch 10/100
65/65 [==============================] - 0s 2ms/step - loss: 0.2483 - accuracy: 0.8817
```



```
Epoch 91/100
65/65 [==============================] - 0s 2ms/step - loss: 0.1903 - accuracy: 0.9149
Epoch 92/100
65/65 [==============================] - 0s 2ms/step - loss: 0.1902 - accuracy: 0.9152
Epoch 93/100
65/65 [==============================] - 0s 2ms/step - loss: 0.1900 - accuracy: 0.9149
Epoch 94/100
65/65 [==============================] - 0s 2ms/step - loss: 0.1899 - accuracy: 0.9161
Epoch 95/100
65/65 [==============================] - 0s 2ms/step - loss: 0.1897 - accuracy: 0.9158
Epoch 96/100
65/65 [==============================] - 0s 2ms/step - loss: 0.1896 - accuracy: 0.9155
Epoch 97/100
65/65 [==============================] - 0s 2ms/step - loss: 0.1896 - accuracy: 0.9152
Epoch 98/100
65/65 [==============================] - 0s 2ms/step - loss: 0.1893 - accuracy: 0.9152
Epoch 99/100
65/65 [==============================] - 0s 2ms/step - loss: 0.1892 - accuracy: 0.9155
Epoch 100/100
65/65 [==============================] - 0s 2ms/step - loss: 0.1891 - accuracy: 0.9158
44/44 [==============================] - 0s 2ms/step - loss: 0.2303 - accuracy: 0.9037
Accuracy: 90.37%
```

Compare accuracy under Logistic Regression with that under Support Vector Machine. Which machine learning method performs better for this dataset? Explain why.

Accuracy alone may not provide a complete picture when comparing the performance of machine learning models, especially when dealing with imbalanced datasets or specific requirements of the task. However, given the information provided, we can make some general observations.

1. **Logistic Regression (88% accuracy):**

- Logistic Regression is a linear model that works well when the relationship between the features and the target variable is approximately linear.
- If the dataset is well-suited for a linear model and the assumptions of logistic regression are met, an accuracy of 88% is reasonable.

2. **Support Vector Machine (90.37% accuracy):**

- Support Vector Machines (SVMs) are versatile and can handle non-linear relationships between features and target variables by using different kernel functions.
- SVMs are effective in high-dimensional spaces and can perform well when the data is not linearly separable.
- The higher accuracy of 90.37% suggests that the SVM might capture more complex patterns in the data.

Performance Comparison:

- The Support Vector Machine outperforms Logistic Regression in terms of accuracy, indicating that SVM captures more intricate relationships in the data.
- It's important to consider the nature of the dataset. If the dataset has non-linear patterns, SVM may be more suitable. If the dataset is predominantly linear, Logistic Regression may be sufficient and more interpretable.

Considerations:

- Precision, recall, and F1 score are useful metrics to evaluate model performance, especially when dealing with imbalanced datasets.
- It's crucial to assess the potential overfitting or underfitting of models using techniques like cross-validation.
- The choice of the model should also consider factors like interpretability, computational efficiency, and ease of implementation.

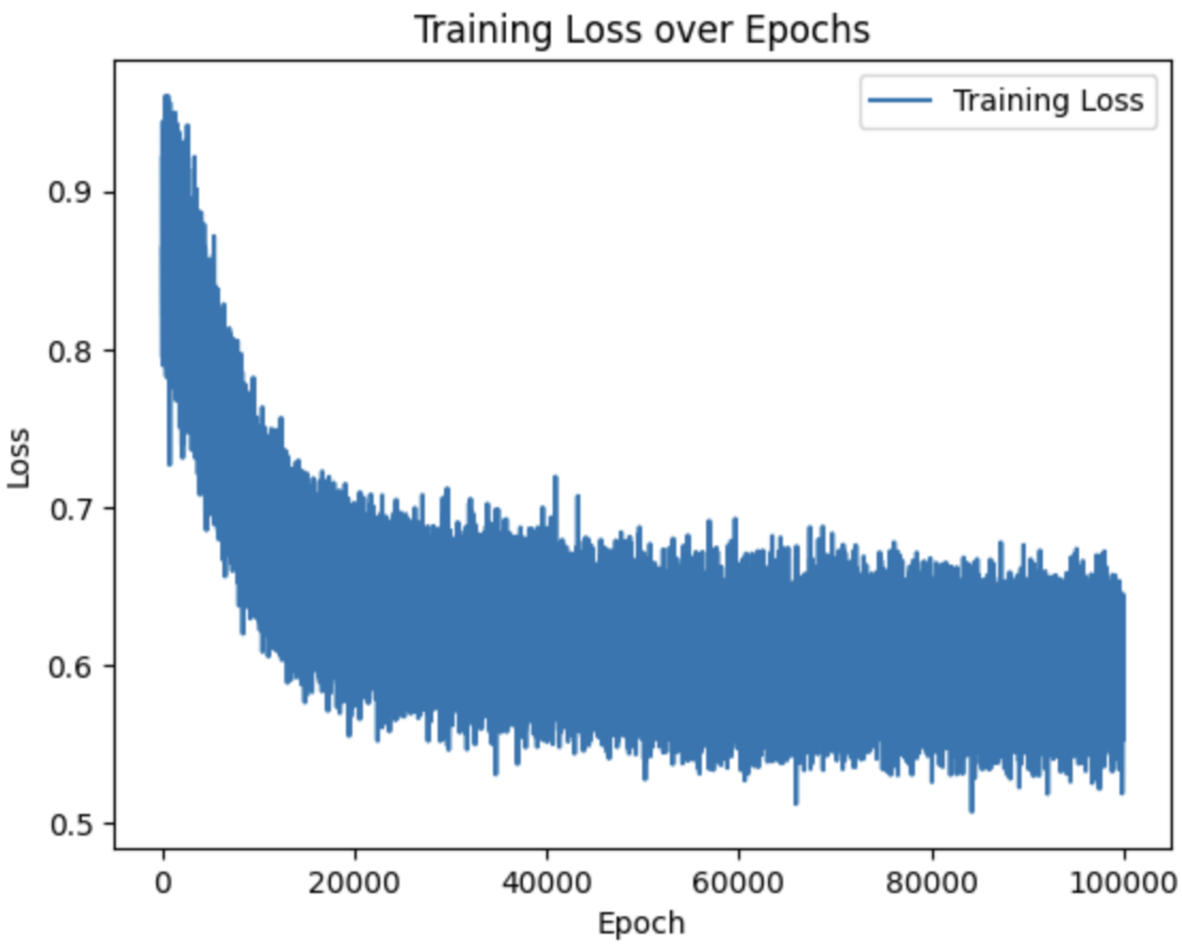
In summary, based on the provided accuracy values, the Support Vector Machine appears to perform better for this dataset.

Plot Results:

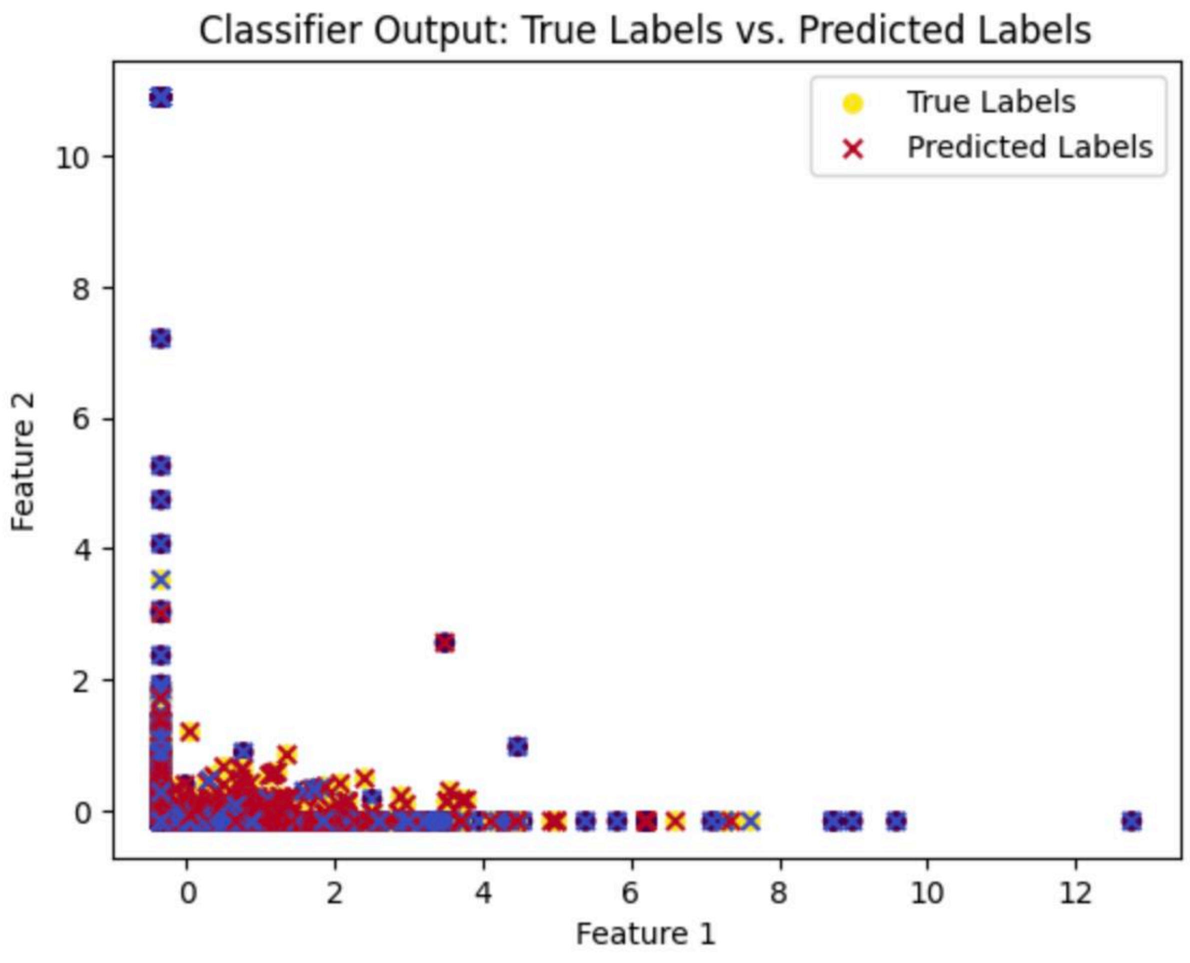
- **Logistic Regression:**

The Logistic Regression classifier model achieved 88.45% accuracy over 1,00,000 epochs of training.

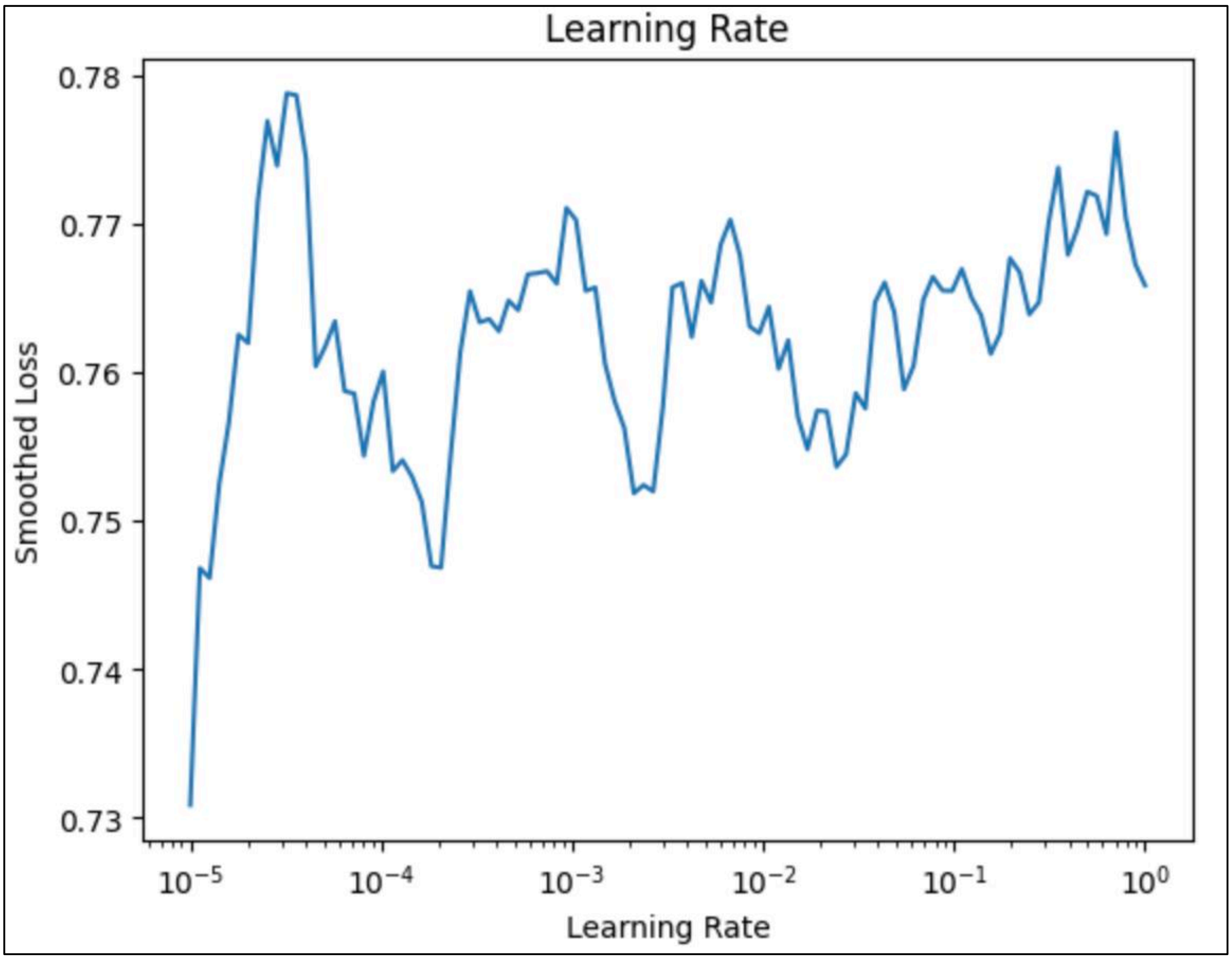
```
Accuracy: 0.8810105949470253
```



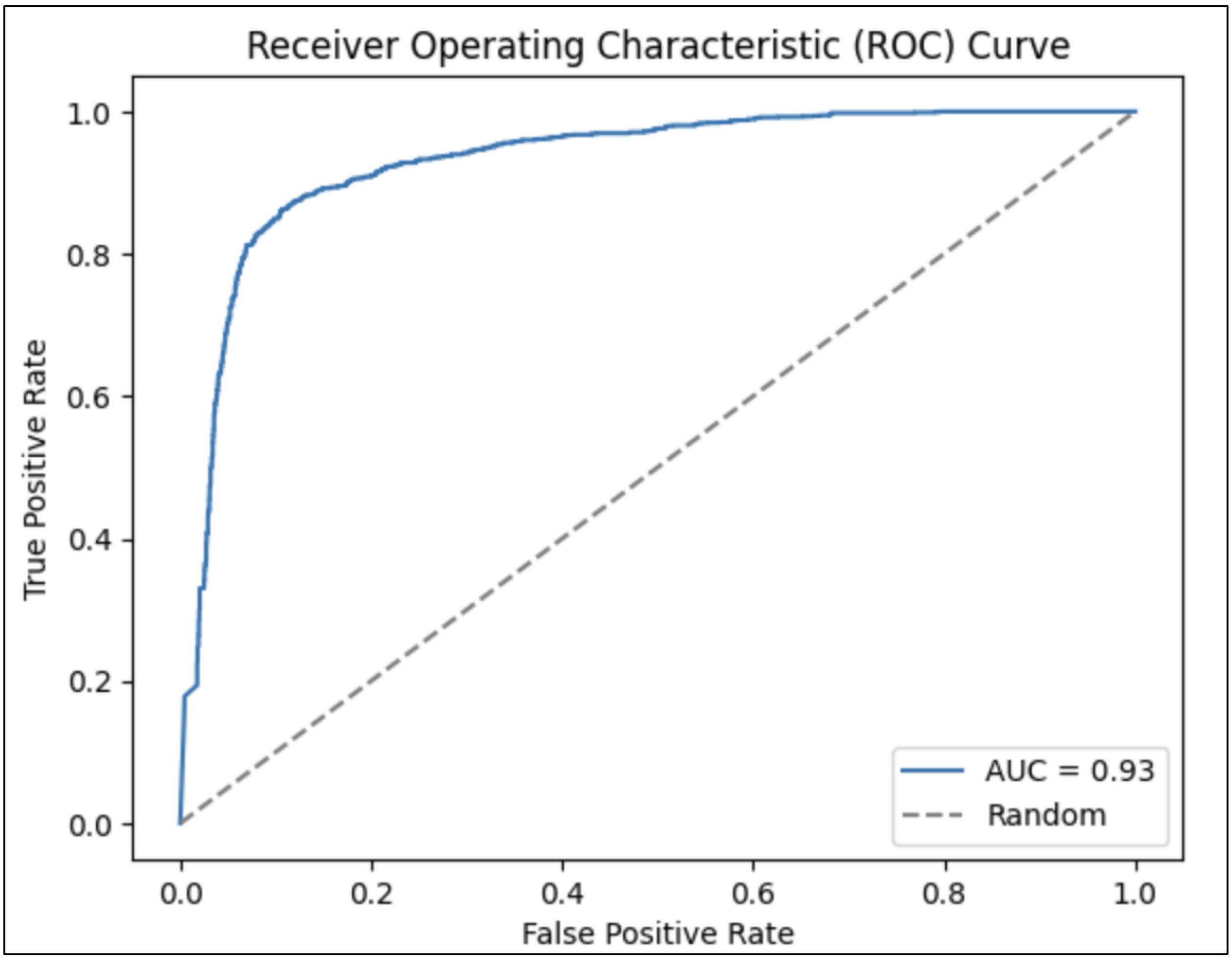
- The Logistic Regression Classifier plotting the True Labels vs Predicted Labels



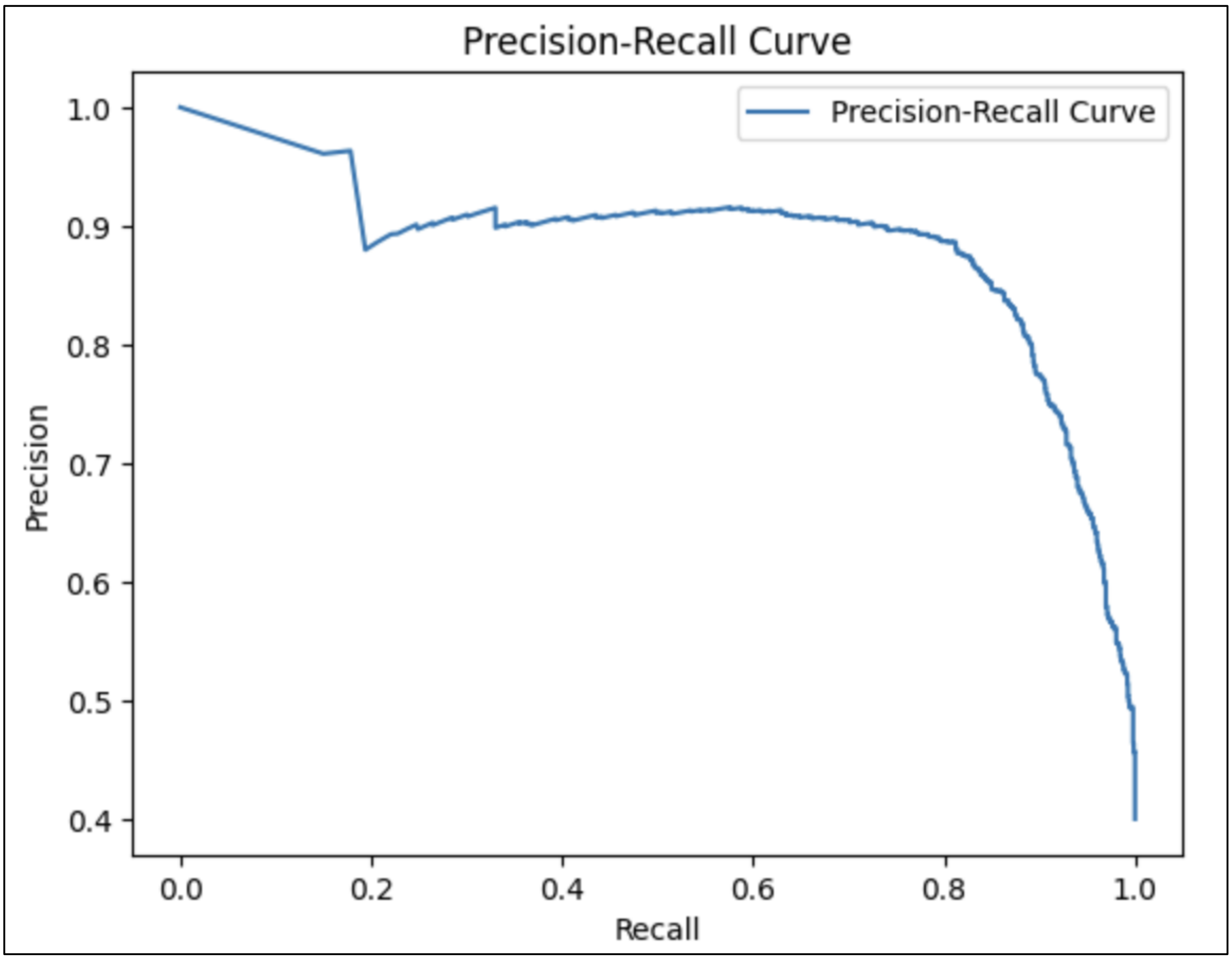
- The Logistic Regression Learning Rate



- ROC Curve:

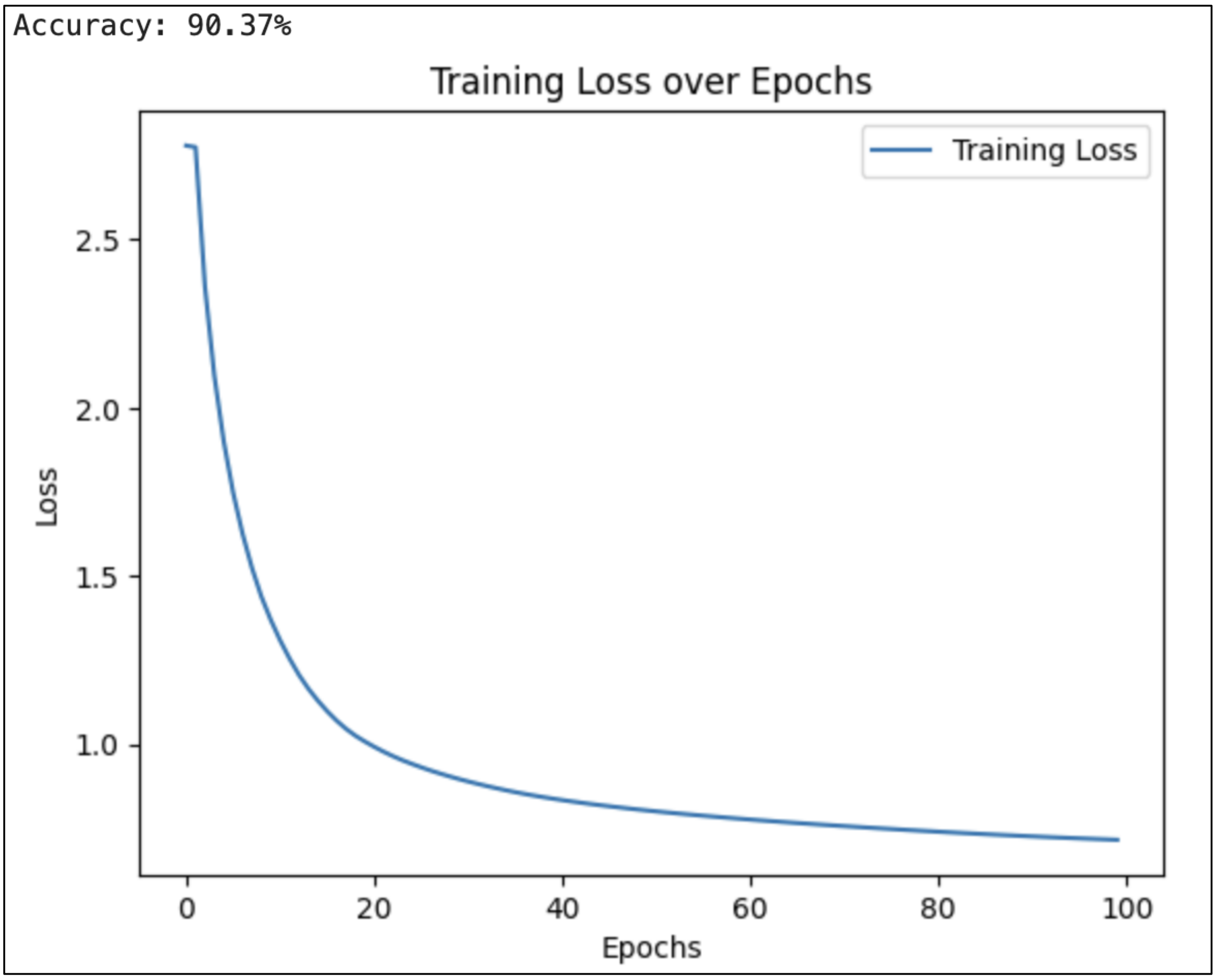

Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
AUC = 0.93
Random

- Precision and Recall Curve



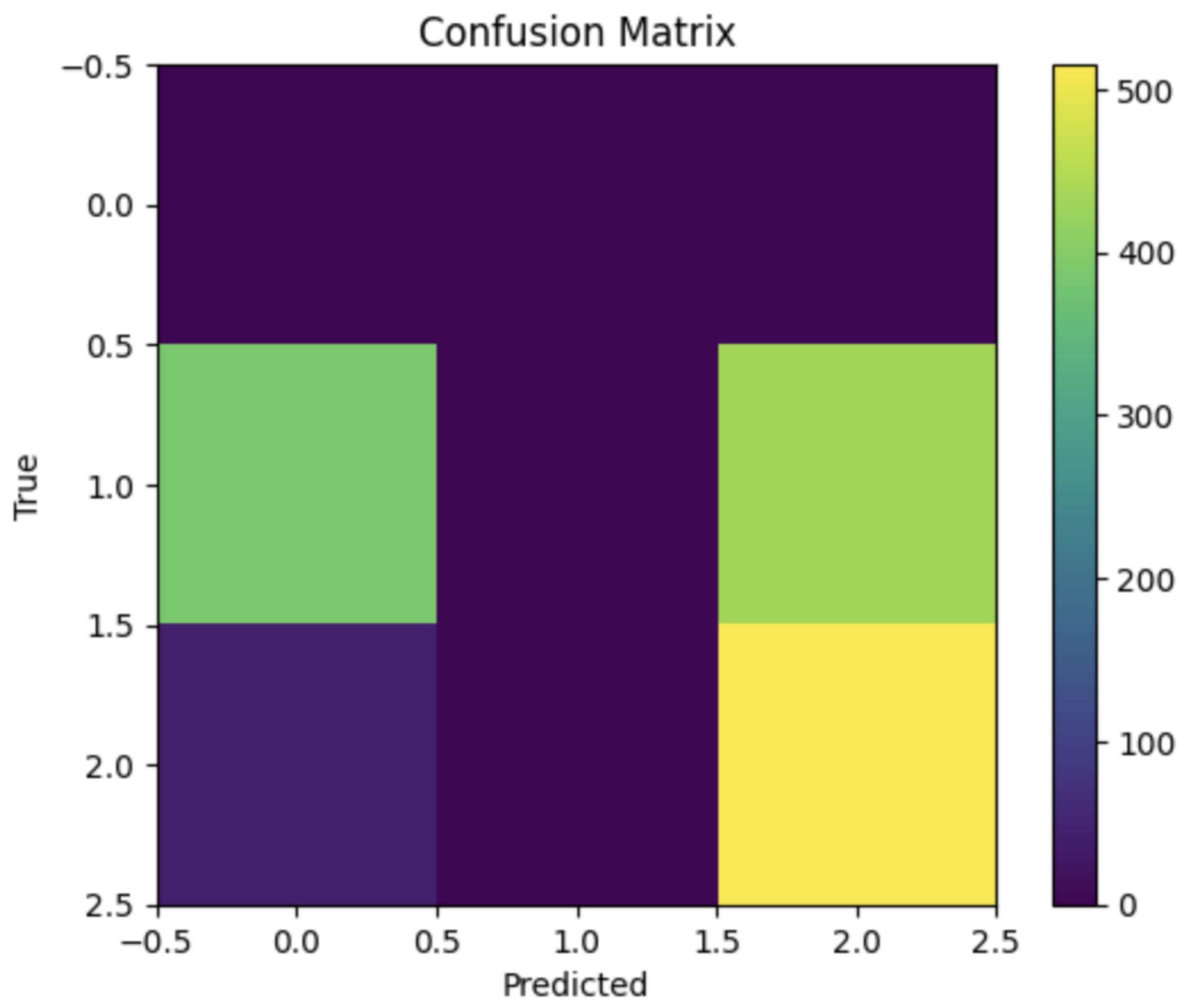
- **Support Vector Machine:**

The Support Vector Machine classifier achieved a 90.37% accuracy over 100 epochs of training data with a batch size of 50.

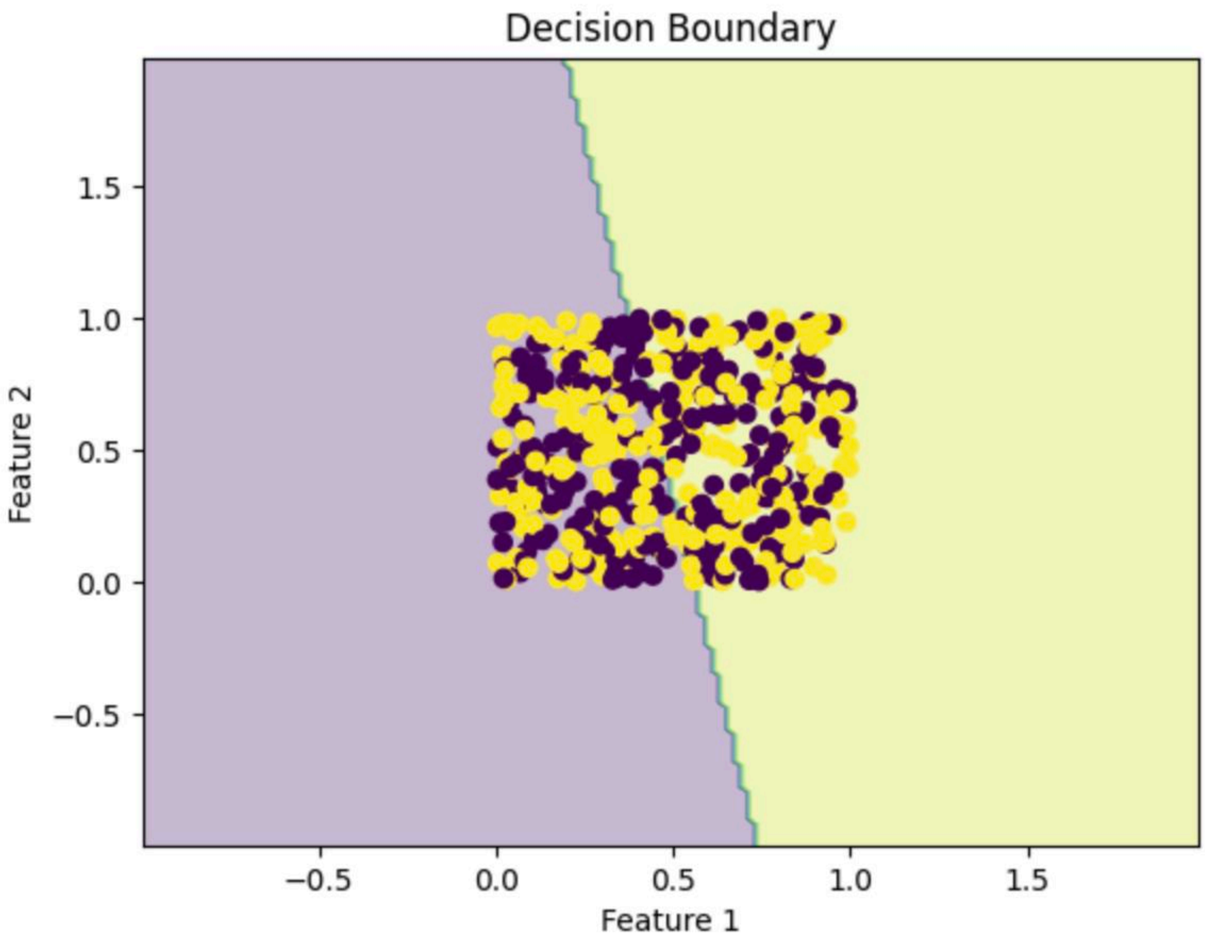


```
Accuracy: 90.37%
```

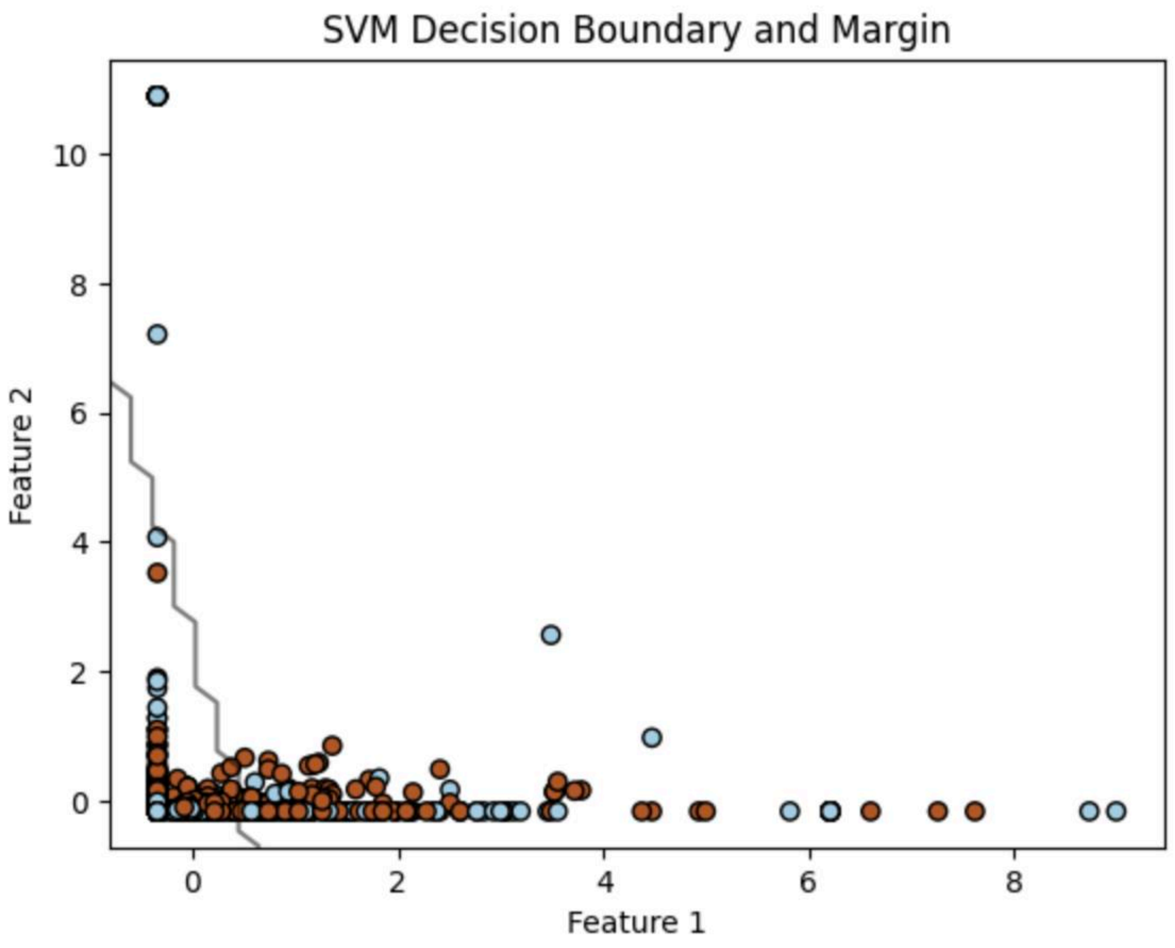
- Confusion Matrix



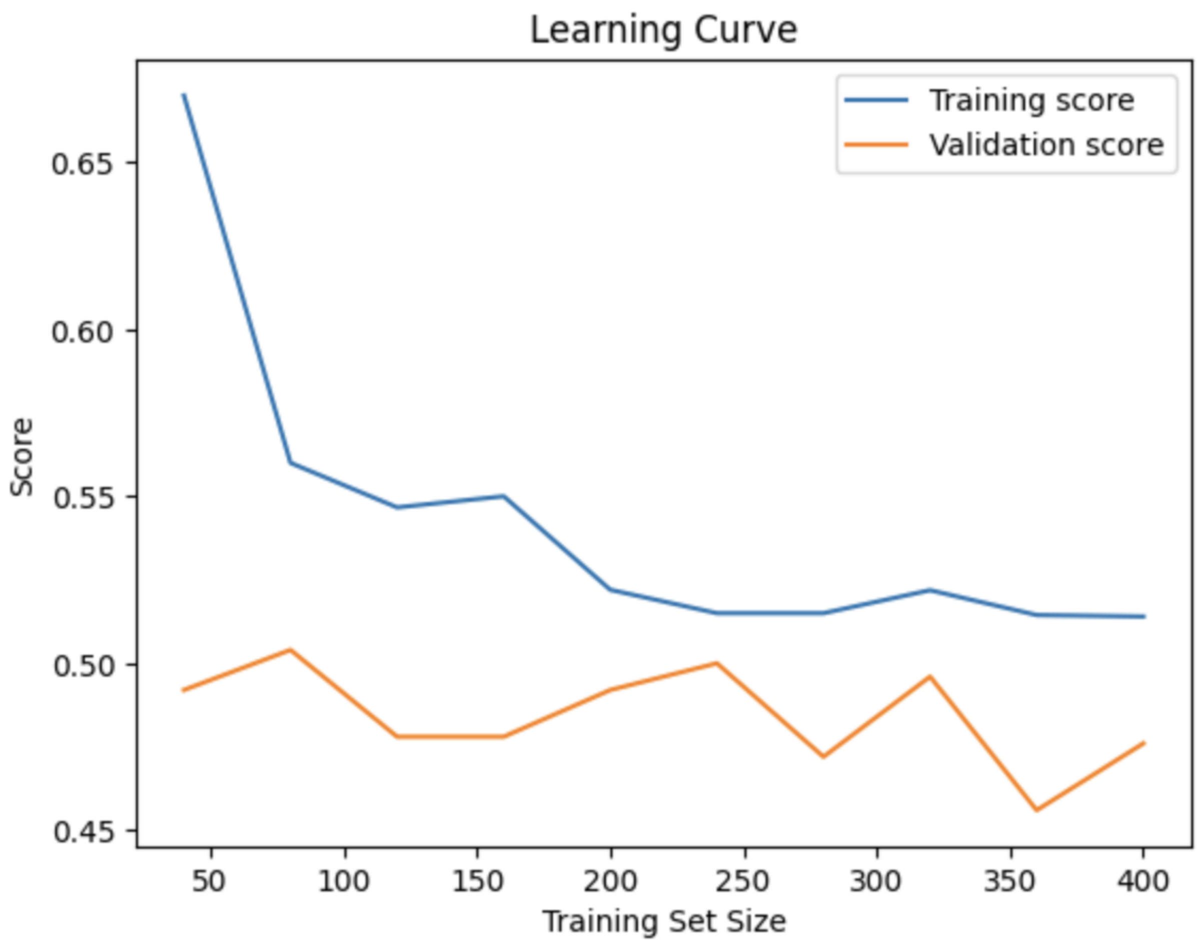
- Decision Boundary



- SVM Decision Boundary and Margin



- The Support Vector Machine Learning Curve


Learning Curve
Training score
Validation score
Score
Training Set Size