CSE574
Project-4

Classification and Regression

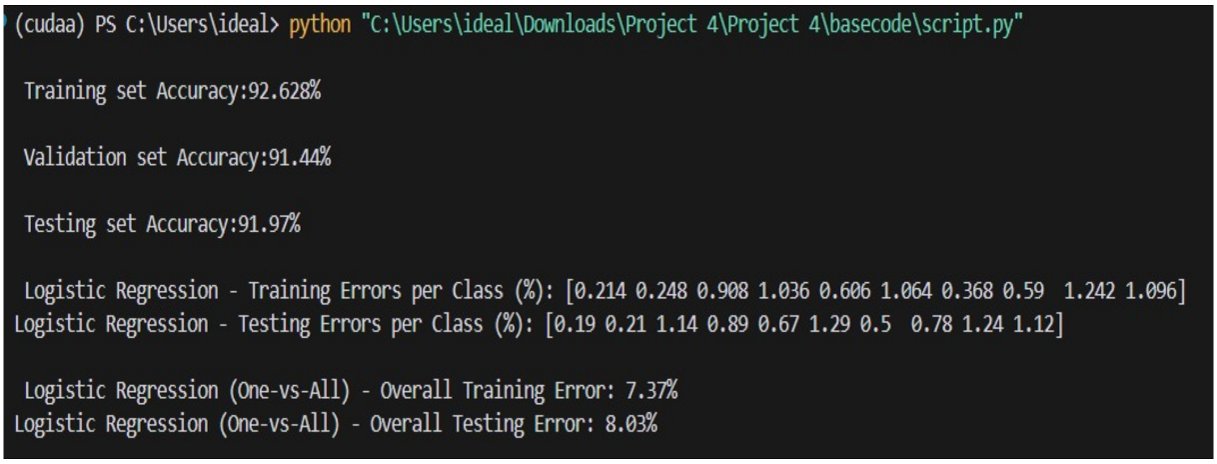
Amulya Reddy Datla
UB Person number: 50560100
amulyare@buffalo.edu

Vignesh Thalur Jayachandrakumar
UB Person number: 50559900
vignesht@buffalo.edu

1 Logistic Regression

1.1 In your report, you should train the logistic regressor using the given data X (Preprocessed feature vectors of MNIST data) with labels y. Record the total error with respect to each category in both training data and test data. And discuss the results in your report and explain why there is a difference between training error and test error

Logistic Regression to classify hand-written digit images into correct corresponding labels is implemented and the concerned results are obtained. The below image consists of accuracies and recorded errors per class.



```
(cudaa) PS C:\Users\ideal> python "C:\Users\ideal\Downloads\Project 4\Project 4\basecode\script.py"

 Training set Accuracy:92.628%

 Validation set Accuracy:91.44%

 Testing set Accuracy:91.97%

 Logistic Regression - Training Errors per Class (%): [0.214 0.248 0.908 1.036 0.606 1.064 0.368 0.59  1.242 1.096]
Logistic Regression - Testing Errors per Class (%): [0.19 0.21 1.14 0.89 0.67 1.29 0.5  0.78 1.24 1.12]

 Logistic Regression (One-vs-All) - Overall Training Error: 7.37%
Logistic Regression (One-vs-All) - Overall Testing Error: 8.03%
```

Figure 1: Logistic Regression Output

1.1.1 Analysis of the results

Training the logistic regressor using the given data X Preprocessed feature vectors of MNIST data with labels y is performed. The performance analysis of logistic regression reveals that the classifier achieved training set accuracy of 92.628%, validation set accuracy of 91.44% and Testing set Accuracy of 91.97%. The error analysis provides insight into the performance of the model across digit classes from zero to nine during training and testing. For training, the errors recorded for each class are as shown in the above screenshot. These results indicate significant variations in error rates, with higher errors observed for certain digits like eight in both training and testing, suggesting these classes pose greater challenges for the model and may require further attention for improved performance.

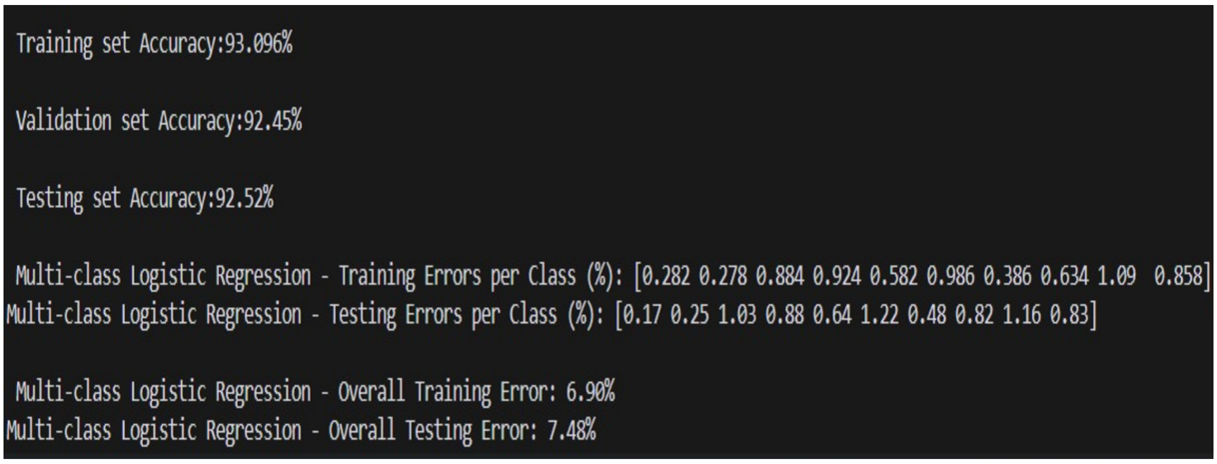
1.1.2 Difference between Training Error and Test Error

Total testing error is significantly higher than the total training error for both logistic regression. There is a small difference between training and testing errors suggesting that the model is not over-fitting significantly to the training data. The model parameters are optimized to minimize the error on the training set, resulting in a lower training error and the testing error was slightly higher due to generalization challenges. A small difference in errors 7.37% for training set and 8.03% for testing indicates that the model generalizes well, but variability in the test set causes the testing error to be slightly higher. Class imbalance, such as fewer samples of specific digits also affects the performance of the model, making it perform better on more represented classes and worse on underrepresented ones. The complexity of digit features, particularly for ambiguous or overlapping digits, further contributes to higher testing errors for certain classes. Finally, noise introduced by handwriting variability across individuals in the test set adds additional challenge, leading to increased errors that were not accounted for in the training set.

2 Multi-class Logistic Regression

2.1 In your report, you should train the logistic regressor using the given data X(Preprocessed feature vectors of MNIST data) with labels y. Record the total error with respect to each category in both training data and test data. And discuss the results in your report and explain why there is a difference between training error and test error. Compare the performance difference between multi-class strategy with one-vs-all strategy

Multi-Class Logistic Regression is implemented on the MNIST dataset and the concerned results are obtained. The below image consists of accuracies and recorded errors per class.



```
Training set Accuracy:93.096%

Validation set Accuracy:92.45%

Testing set Accuracy:92.52%

Multi-class Logistic Regression - Training Errors per Class (%): [0.282 0.278 0.884 0.924 0.582 0.986 0.386 0.634 1.09  0.858]
Multi-class Logistic Regression - Testing Errors per Class (%): [0.17 0.25 1.03 0.88 0.64 1.22 0.48 0.82 1.16 0.83]

Multi-class Logistic Regression - Overall Training Error: 6.90%
Multi-class Logistic Regression - Overall Testing Error: 7.48%
```

Figure 2: Multi Class Logistic Regression Output

2.1.1 Analysis of the results

Training the multi class logistic regressor using the given data X Preprocessed feature vectors of MNIST data with labels y is performed. The performance analysis of logistic regression reveals that the classifier achieved training set accuracy of 93.096%, validation set accuracy of 92.45% and Testing set Accuracy of 92.52%. The model shows varying performance across different digit classes. The error analysis provides insight into the performance of the model across digit classes from zero to nine during training and testing. For training, the errors recorded for each class are as shown in the above screenshot. These results indicate significant variations in error rates, with higher errors observed for certain digits in both training and testing.

2.1.2 Difference between Training Error and Test Error

The training error rates range from 0.278% to 1.09%, while testing errors range from 0.17% to 1.22%, indicating slightly higher variability in the test set compared to the training set.This pattern is typical in machine learning models and indicates good generalization without severe over-fitting. The slight difference between the total training error 6.90% and testing error 7.48% highlights the good generalization capability of the model. This small gap of 0.58% indicates that the model has avoided significant over-fitting, maintained consistent performance across different data splits, and successfully learned generalizable patterns from the training data. Per-class error analysis shows that certain digits are more challenging to classify due to variability in how they are written, similar features between digit pairs, and the complexity in distinguishing number shapes. This multi-class approach demonstrates robust performance, using a single classifier to handle all 10 classes efficiently.

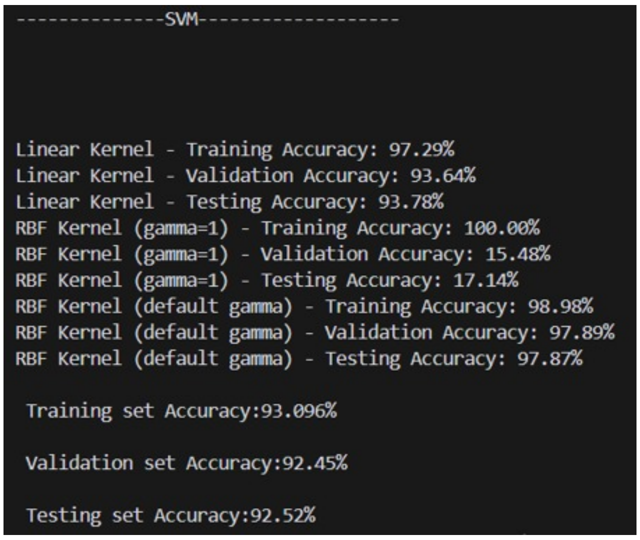
2.1.3 Difference between multi-class strategy with one-vs-all strategy

Metric	Multi-Class	One vs All
Training Accuracy	93.096%	92.628%
Testing Accuracy	92.52%	91.97%
Validation Accuracy	92.45%	91.44%
Training Error	6.90%	7.37%
Testing Error	7.48%	8.03%

The above table shows the comparison between the multi-class strategy and the one-vs-all strategy reveals distinct differences in performance across various metrics. The multi-class strategy achieved 93.096% training accuracy, 92.52% testing accuracy and 92.45% validation accuracy slightly outperforming the one-vs-all strategy having 92.628%, 91.97% and 91.44%. The multi-class strategy had a training error of 6.90%, which is slightly better than the one-vs-all strategy having 7.37%. The multi-class strategy also had a lower testing error of 7.48% compared to 8.03% for the one-vs-all strategy. These results highlight the superior performance of multi-class strategy across all measured metrics, indicating better generalization and lower over-fitting compared to the one-vs-all strategy. Multi-Class Logistic Regression is simpler to implement as it uses a single model instead of multiple classifiers, resulting in slightly better overall accuracy and avoiding inconsistencies. However, it has limitations, such as higher computational cost and susceptibility to error propagation.

3 Support Vector Machines

3.1 In your report, you should train the SVM using the given data X(Preprocessed feature vectors of MNIST data) with labels y. And discuss the performance differences between linear kernel and radial basis, different gamma setting.



```
--------------SVM-------------------



Linear Kernel - Training Accuracy: 97.29%
Linear Kernel - Validation Accuracy: 93.64%
Linear Kernel - Testing Accuracy: 93.78%
RBF Kernel (gamma=1) - Training Accuracy: 100.00%
RBF Kernel (gamma=1) - Validation Accuracy: 15.48%
RBF Kernel (gamma=1) - Testing Accuracy: 17.14%
RBF Kernel (default gamma) - Training Accuracy: 98.98%
RBF Kernel (default gamma) - Validation Accuracy: 97.89%
RBF Kernel (default gamma) - Testing Accuracy: 97.87%

 Training set Accuracy:93.096%

 Validation set Accuracy:92.45%

 Testing set Accuracy:92.52%
```

Figure 3: Support Vector Machines Output

The above image highlights the comparison between the performance of Support Vector Machine using linear and Radial Basis Function kernels with different gamma settings. The Linear kernel demonstrates strong, consistent performance across all datasets with a training accuracy of 97.29%, validation accuracy of 93.64%, and testing accuracy of 93.78%. In contrast, the RBF kernel with gamma=1 shows extremely poor performance, with a training accuracy of 100.00% but significantly low validation accuracy of 15.48% and testing accuracy of 17.14%, indicating severe overfitting. On the other hand, the RBF kernel with the default gamma setting excels with a training accuracy of 98.98%, validation accuracy of 97.89%, and testing accuracy of 97.87%, demonstrating the best balance between training and generalization performance. This comparison highlights the critical importance of selecting the appropriate kernel and tuning hyper-parameters in SVM models for optimal performance.

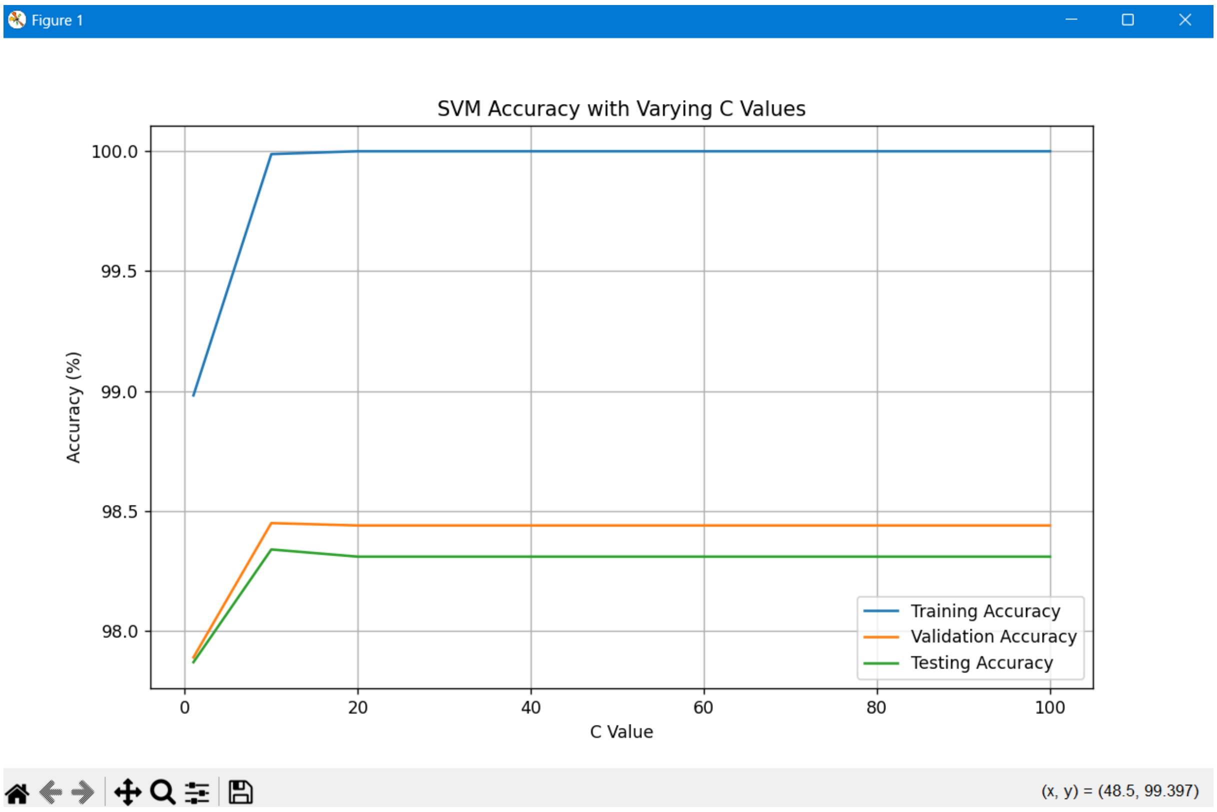


Figure 4: Support Vector Machines Plot

This plot illustrates the impact of varying the C parameter which is regularization strength on SVM performance across different accuracy metrics. It shows three distinct accuracy curves. Training accuracy rapidly increased from around 99% to 100% as C rises up to twenty, then plateaus. Validation accuracy stabilizes around 98.4%, and testing accuracy stabilizes at approximately 98.3%. In the initial range Of C from between zero and twenty, there is a steep improvement across all metrics. Beyond the plateau region C value is greater than twenty, all metrics remain relatively constant with minimal improvement. The small gap between training and validation or testing accuracies indicates minimal over-fitting, demonstrating optimal model behavior around C value of twenty.