**Part 5:**

1. **Description of the experimental setup**

In this part of the assignment, we were asked to train a logistic regression model and a support vector machine model on a subset of the mnist dataset and to use cross-validation to evaluate the robustness of the models and compare their relative performance. Below is a summary of the key steps taken to accomplish these goals:

1. The full mnist dataset was uploaded and shuffled.
2. The dataset was then split into a training set of 5,000 images and a test set of 37,000 images. These subsets were also shuffled and broken into labels representing the hand-written digits and pixels representing the image of each label. The training and test subsets were used uniformly by all the models discussed below.
3. A logistic regression (LR) function with LASSO penalty was used to train the logistic regression model to classify the training set. We tested four LR models with four different values of “C” (the inverse regularization strength); namely, we used “C” in the range of **[0.01, 0.1, 1.0, 1.5]**. Larger values of “C” would give the model more freedom to decide on the weights of the model while lower values of “C” places more constraints on the model.In addition, we used the “saga” optimizer and 10,000 iterations for optimization purposes.
4. In addition to training the LR model on the full training set, we also utilized a 3-fold cross validation to test the robustness of the LR model. A 3-fold cross validation would train the model on two folds while using the remaining fold for evaluation. Thus a 3-fold cross validation allows us to produce three evaluations of the model’s accuracy and robustness. As shown below, we used the results of the 3-fold cross validation exercise to compare the accuracy of the classification models.
5. The LR model (alongside the support vector machines models discussed below) was evaluated used multiple criteria including: 1) the 3-fold cross validation accuracy measures, 2) overall accuracy of the model, 3) the precision of the model, 4) the recall of the model, 5) the f1 score of the model, and 6) the confusion matrix.
6. The final LR model was evaluated based on its accuracy and confusion matrix on the test dataset; namely the set of 37,000 images.
7. Similarly for the Support Vector Machines (SVM), we deployed two SVM models to train on the training set of the mnist dataset; namely, an SVM model with a Polynomial Kernel, and an SVM model with a Gaussian Kernel. Both Kernels attempt to find hyper-planes to separate the dataset on the transformed space (the space mapped to the kernel functions). The hyper-planes are chosen in such a way as to maximize the margin between the classes. The margin is the distance between the hyper-planes and the closest data points from each class. Effectively then, the hyper-planes are chosen in such a way as to minimize the classification error on the training dataset.
8. The Polynomial SVM uses two important hyper-parameters: the regularization parameter “C” and the number of degrees of the Polynomial function. We conducted a grid search to identify effective estimates of these two parameters. More specifically, we ran a grid search with the following hyper-parameter values: “C” = **[1, 3, 5, 10]** and Polynomial degrees of **[3, 5, 10]**. Thus, for each value of the Polynomial degrees, we produced four estimates of the model: one for each value of “C”. As stated above for the LR model, a 3-fold cross validation was used to evaluate the robustness of the classification results with the Polynomial SVM. It is important to note that polynomial functions with higher degrees will tend to produce a better fit for the training dataset, albeit with a much greater tendency to overfit.
9. Lastly for the Gaussian SVM, two hyper-parameters were fine-tuned; namely, the regularization parameter “C” and the Gamma parameter. The Gamma parameter determines the width of the distribution or equivalently how far the influence of a single training example could reach. Low values of gamma stretch the influence of each point of the distribution, while high values of gamma cause the radius of influence of each point to shrink. We trained a large set of hyper-parameter values in order to carefully fine-tune these two hyper-parameters. More specifically, we used the following grid search: “C” = **[0.01, 0.1, 0.5, 1, 3, 5, 7, 10]** and gamma values of **[0.001, 0.01, 0.05, 0.1, 1, 2, 4, 5, 10]**. Thus, for each value of “C”, we trained 9 models each with its own value for gamma. A 3-fold cross validation was used to evaluate the robustness of the classification results with the Gaussian SVM.
10. The LR and SVM models were trained on the same set of 5,000 training images and tested on the same set of 37,000 test images. We should note that only the selected models with the chosen fine-tuned hyper-parameters were used for testing based on the cross-validation results.
11. **List of Functions:**

The table below lists the primary functions used to test and evaluate the models:

|  |  |
| --- | --- |
| Algorithm | Functions |
| Logistic Regression with LASSO | LogisticRegression(penalty='l1',..) |
| Cross validation | cross\_val\_score() |
| Accuracy measures | confusion\_matrix(), precision\_score(), recall\_score(), and f1\_score |
| Polynomial Support Vector Machine | make\_pipeline(StandardScaler(), SVC(kernel="poly", degree, coef0, C)) |
| Gaussian Support Vector Machine | make\_pipeline(StandardScaler(), SVC(kernel="rbf", gamma, C)) |

It should be noted that we ran the grid search by looping through the hyper-parameters.

1. **Analysis:**
   1. The Logistic Regression Model:

Four versions of the logistic regression model were tested with varying values of the regularization parameter C. Below are the key results. We should note that the Python report contains additional measures generated such as the precision and recall of each model.

|  |  |  |
| --- | --- | --- |
| **Hyperparameter(s)** | **Accuracy on the training set** | **3-fold cross validation results** |
| C = 0.01 | 0.993 | [0.87042591 0.8830234 0.8847539] |
| **C = 0.10** | **1.00** | **[0.87462507 0.8818236 0.8835534]** |
| C = 1.0 | 1.00 | [0.87162567 0.8860228 0.8841536] |
| C = 1.5 | 1.00 | [0.87162567 0.8860228 0.8847539] |
| **Model Parameters: Penalty = ‘l1’, Solver = Saga, Iteration = 10000, Multinomial Class** | | |

As clearly indicated by the above table, the results are highly comparable across multiple values of “C”. Choosing the “C” value of 0.10, the table below shows the accuracy and confusion matrix produced by the logistic model on the test dataset of 37,000 images. The accuracy score and confusion matrix confirm the results of the 3-fold cross validation.

Overall Accuracy: 0.8738

Confusion Matrix:

**[3458 0 13 19 16 67 45 12 25 19]**

**[ 0 3940 25 36 7 11 6 13 66 14]**

**[ 40 58 3044 164 52 30 80 44 132 33]**

**[ 21 27 78 3275 11 182 22 46 109 57]**

**[ 9 11 51 7 3190 19 60 17 38 180]**

**[ 58 21 36 175 53 2626 69 28 190 56]**

**[ 30 10 59 4 79 52 3348 9 31 3]**

**[ 8 37 58 40 57 6 6 3477 24 209]**

**[ 20 75 95 206 41 174 43 17 2811 88]**

**[ 24 10 17 65 182 33 6 118 75 3162]**

* 1. The Support Vector Machine Models:

As stated above, we trained two SVM models; namely, the Polynomial-Kernel SVM (P-SVM) model and the Gaussian-Kernel SVM (G-SVM) model. Below we first produce the results of training the P-SVM:

|  |  |  |
| --- | --- | --- |
| **Hyperparameter(s)** | **Accuracy on the training set** | **3-fold cross validation results** |
| Degrees = 3, C = 1 | 0.9972 | [0.9286142 0.9466106 0.94837935] |
| Degrees = 3, C = 3 | 0.9996 | [0.9274145 0.9460108 0.94417767] |
| Degrees = 3, C = 5 | 1.000 | [0.9280144 0.9460108 0.94357743] |
| Degrees = 3, C = 10 | 1.000 | [0.9286142 0.9460108 0.94297719] |
|  | | |
| **Degrees = 5, C = 1** | **1.000** | **[0.9304139 0.9502099 0.94597839]** |
| Degrees = 5, C = 3 | 1.000 | [0.9304139 0.9496100 0.94597839] |
| Degrees = 5, C = 5 | 1.000 | [0.9304139 0.9496100 0.94597839] |
| Degrees = 5, C = 10 | 1.000 | [0.9304139 0.9496100 0.94597839] |
|  | | |
| Degrees = 10, C = 1 | 1.000 | [0.9280144 0.9466106 0.92977191] |
| Degrees = 10, C = 3 | 1.000 | [0.9280144 0.94661068 0.9297719] |
| Degrees = 10, C = 5 | 1.000 | [0.9280144 0.94661068 0.9297719] |
| Degrees = 10, C = 10 | 1.000 | [0.9280144 0.94661068 0.9297719] |

As stated earlier, the P-SVM model starts to show signs of over-fitting as the degree of the function increases. Based on the results of the 3-fold cross validation, we chose the P-SVM model with a degree parameter of 5 and a regularization parameter of 1. The chosen model was then tested on the test dataset of 37,000 images. The results are shown below:

Overall Accuracy: 0.94803

Confusion Matrix:

**[3596 0 7 2 10 21 18 2 14 4]**

**[ 0 4052 12 12 14 1 6 6 9 6]**

**[ 16 27 3453 51 43 4 16 23 39 5]**

**[ 8 14 46 3586 9 63 6 28 45 23]**

**[ 5 4 15 1 3462 5 12 3 5 70]**

**[ 16 19 9 80 46 3032 43 4 25 38]**

**[ 27 7 15 0 19 38 3509 1 9 0]**

**[ 2 28 44 13 56 8 0 3645 7 119]**

**[ 8 28 25 49 35 57 14 8 3318 28]**

**[ 11 5 20 26 102 12 1 63 28 3424]**

Similar analysis was conducted on the G-SVM. However, given the wide range used to fine-tune the hyper-parameters as stated above, we chose to only report the key subset of the trained G-SVM models. Below are the results:

|  |  |  |
| --- | --- | --- |
| **Hyperparameter(s)** | **Accuracy on the training set** | **3-fold cross validation results** |
| Gamma = 0.001, C = 0.01 | 0.3334 | [0.15656869 0.14877025 0.15606242] |
| Gamma = 0.001, C = 0.50 | 0.9482 | [0.89262148 0.91841632 0.89435774] |
| Gamma = 0.001, C = 3.00 | 0.9914 | [0.92321536 0.93881224 0.91836735] |
|  | | |
| Gamma = 0.01, C = 0.01 | 0.1132 | [0.11337732 0.11277744 0.11344538] |
| Gamma = 0.01, C = 0.50 | 0.9182 | [0.5794841 0.57168566 0.55762305] |
| Gamma = 0.01, C = 3.00 | 1.0000 | [0.73485303 0.72465507 0.70348139] |
|  | | |
| Gamma = 0.05, C = 0.01 | 0.1132 | [0.11337732 0.11277744 0.11344538] |
| Gamma = 0.05, C = 0.50 | 0.2070 | [0.17036593 0.17096581 0.16446579] |
| Gamma = 0.05, C = 3.00 | 1.0000 | [0.20215957 0.19556089 0.19387755] |
|  |  |  |
| **Gamma = 0.001, C = 5.00** | **0.9972** | **[0.92321536 0.94181164 0.92016807]** |

As can be seen from the table above, the G-SVM model starts to show signs of over-fitting as the parameters increase in value. Upon further fine-tuning which is included in the Python report, we chose a gamma value of 0.001 and a regularization parameter of 5. Below are the results of running the chosen model on the test dataset of 37,000 images:

Overall Accuracy: 0.9364

Confusion Matrix:

**[ 3563 0 27 4 7 21 30 2 17 3]**

**[ 0 4035 27 8 10 1 6 7 15 9]**

**[ 14 20 3456 69 26 5 21 24 36 6]**

**[ 8 14 97 3524 3 78 6 32 44 22]**

**[ 7 4 61 0 3395 6 17 11 4 77]**

**[ 12 18 30 82 38 3005 62 5 29 31]**

**[ 26 7 61 1 13 36 3472 2 7 0]**

**[ 2 26 144 16 33 3 0 3599 4 95]**

**[ 13 43 50 71 29 69 21 10 3233 31]**

**[ 10 7 52 33 90 13 1 98 21 3367]**

The results of the P-SVM and G-SVM models for the chosen hyper-parameters are comparable with a slight advantage for the P-SVM.

We, therefore, chose to compare the statistical difference between the classification accuracies of the P-SVM model with those of the logistic regression model.