For the entirety of this report and the methods described the data that was used had already been divided into a training set and a testing set. However, before any training could be done the data was pre-processed.

The data used for this report was handpicked from the full dataset (used only those data points that corresponded to class 5 and 7). Additionally, each data point in the training set was divided by 255 to have a range of inputs between [0,1]. Lastly, to ensure that training and testing does not take a lot of time, only 6000 input samples were picked from a total of 12000 and 1000 testing points were picked from a total of 2000.

Part 1: Logistic Regression

In this method, the l2 penalty function was used to train and test the data. The main objective from this model was to understand how changing the regularization parameter affects the training and testing accuracy of the model. To get a better understanding of underfitting or overfitting the regularization parameter was picked from a range (logarithmically spaced grid). The staring value for C_0 was 0.05 and 1.5 for alpha. The regularization parameter was calculated in this fashion: $reg = alpha^n * C_0$, where n = {1,…,10}. This allowed for a range of regularization parameters and produced results that were worth mentioning.

Now that the regularization parameters had been picked, the model could be trained, and the training and testing accuracies could be reported and plotted. This experiment was run for a total of 10 times where the regularization parameter changed in each iteration. Table 1 below shows some of the statistics on the accuracies for the training and testing set.

	Min	**Max**	**Average**
Training	Accuracy: 0.964444 Regularization: 0.050	Accuracy: 0.980000 Regularization: 1.922168	Accuracy: 0.97283
Testing	Accuracy: 0.954000 Regularization: 1.922168	Accuracy: 0.966000 Regularization: 0.168750	Accuracy: 0.9606

The trend and relationship between the accuracy and regularization parameter can be visualized better with a graph. Figure 1 shows the graph of how the accuracies changes as the regularization parameter increases on a logarithmically spaced grid.

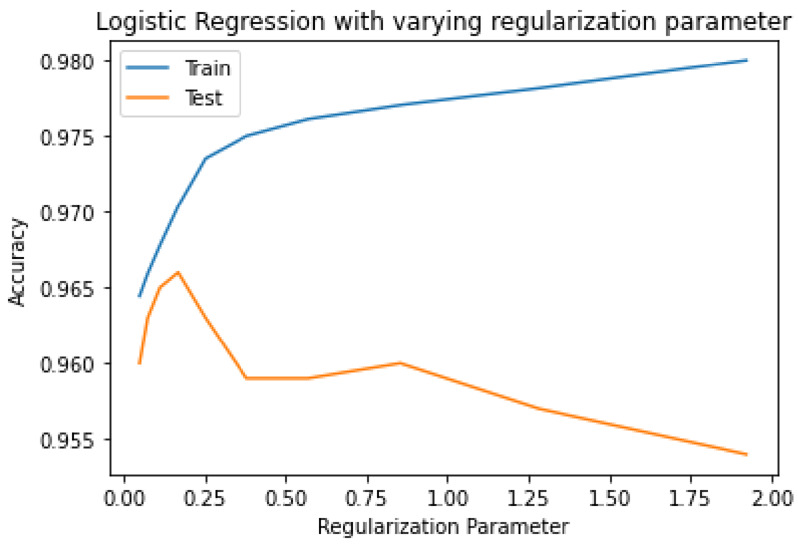

Figure 1: Training and Testing Accuracy of Logistic Regression

From the table 1 and figure 1, we can deduce certain features about logistic regression over a logarithmically spaced for the regularization parameter. Just by scanning over the table it can be noticed that the training set performs better as the regularization parameter increases while in the case of testing set it is the opposite; the accuracy decreases as the regularization parameter increases. Additionally, the training accuracy is better than the testing accuracy on average. Furthermore, the figure aids in strengthening the observations from the table, it is evident that the accuracy for the training set is a lot better as the regularization parameter increases.

Based on the formula for logistic regression it is known that as the regularization parameter increases, the model is less regularized. In other words, as the regularization parameter increases the model tends to overfit the data. The trend of overfitting can be seen from figure 1, the model performs far better on the training data than it does on the testing data.

All in all, it can be concluded that as the regularization parameter increases the training accuracy of the model increases while the testing accuracy decreases. Although, it is still safe to say that the model is accurate as the testing accuracy is 96% on average. Lastly, if the values for the regularization parameter were kept very small, we may have seen the case where the model underfits the data; the model performs better on the testing set than on the training set.

Part 2: Linear Support Vector Machines (SVM)

In this method, the SVM used had a linear kernel function and was used on the training and testing set from part 1. The objective of this method was the same as part 1; to find and observe the change in accuracies as the regularization parameter changed over a logarithmically spaced grid. The procedure for selecting this logarithmically spaced grid was the same as mentioned above, however, in this case the starting value for C_0 was 0.01 while alpha stayed the same at 1.5. The model was again trained for a total of 10 times as done in part 1 and the training and testing accuracies were recorded. Table 1 below shows some of the statistics on the accuracies for the training and testing set.

	Min	**Max**	**Average**
Training	Accuracy: 0.955500 Regularization: 0.010	Accuracy: 0.976500 Regularization: 0.384434	Accuracy: 0.967416
Testing	Accuracy: 0.952000 Regularization: 0.010	Accuracy: 0.965000 Regularization: 0.075938	Accuracy: 0.9588

The trend and relationship between the accuracy and regularization parameter can be visualized better with a graph. Figure 2 shows the graph of how the accuracies changes as the regularization parameter increases on a logarithmically spaced grid.

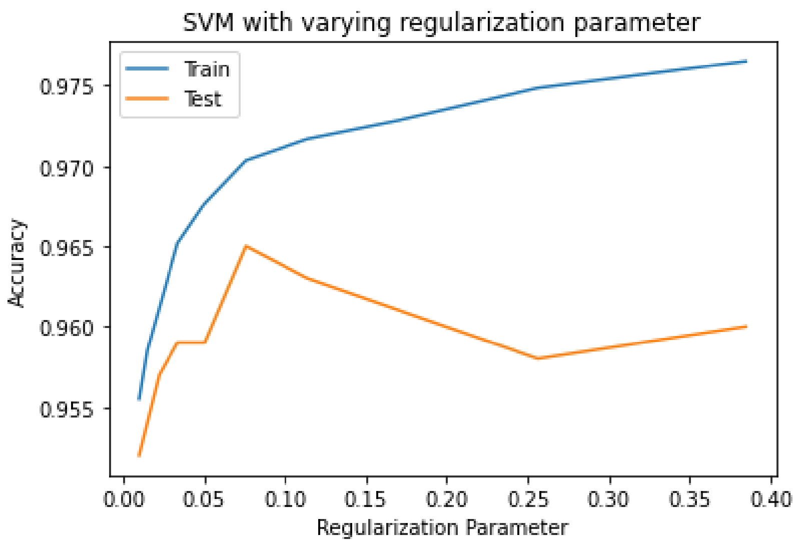

Figure 2: Training and Testing Accuracy of Linear SVM

From the table and figure 2, we can make certain assumptions about the model. From a quick glance at the table, it is noticed that the training set performs better as the regularization parameter increases while in the case of testing set it fluctuated; the accuracy and the regularization parameter did not have an inverse relation, nor did they have a 1-1 relation. A similar trend was seen in Part 1 with logistic regression. However, for a linear SVM the overall testing accuracy increased over time (while fluctuating). Lastly, on average, the model performed better on the training set than the testing set.

Furthermore, figure 2 allows us to visualize the trend between the two accuracies better. We can see that this model also overfits the dataset, although, it is important to note that the accuracies are very close to each other for a few values of the regularization parameter, after which they begin to diverge; unlike what was seen in logistic regression. It should be known that the logarithmic grid was different for this model so the comparison between the two models might not be as useful.

All in all, it can be concluded that as the regularization parameter increases the training accuracy of the model increases while the testing accuracy fluctuates despite having an increase overall. It is noteworthy that the two accuracies are very close to each other for certain values of the regularization parameter. The model has an average accuracy of 95.8% on the testing set so it can still be considered an accurate model. Lastly, underfitting could have been noticed within the same dataset if the regularization parameter had wider range of values.

Part 3: K-Fold Cross Validation

In this section the objective was to implement K-Fold cross validation on the training data to find the optimal regularization parameter for logistic regression and linear SVM. The ideal size of fold is somewhere between 5-10, hence, for this experiment the fold size was 10. Before the methods could be compared the training inputs and training class labels were split into 10 folds.

The K-Fold method requires any jth fold to be the validation set which is used to compare the accuracy of the model after training on the j-1 remaining folds. To introduce a bit a generalization, the validation set was picked randomly. Once the validation set was picked, the remaining folds needed to be concatenated to make it easier for training purposes.

Part 3.1: Logistic Regression

Now that the data has been split into the correct folds and validation set, the logistic regression model is ready to be trained. It is important to note that the logarithmic scale for the regularization parameter was the same as part 1 (C_0 = 0.05 and alpha = 1.5). The training and testing accuracies were recorded and can be seen in the table below.

	Min	**Max**	**Average**
Training	Accuracy: 0.961481 Regularization: 0.050	Accuracy: 0.980370 Regularization: 1.922168	Accuracy: 0.97257
Testing	Accuracy: 0.94833 Regularization: 1.281445	Accuracy: 0.958333 Regularization: 0.050	Accuracy: 0.9535

The table above shows some statistics of the accuracies when the model has been trained with j-1 folds and tested on the validation set. Figure 3 below allows us to visualize this trend. Important to note here that the trend observed in part 1 can be seen here as well. The model overfits the dataset (performs better on the training data than on the validation set).

In this method, the optimal regularization parameter was needed to be found, therefore the way this was picked was by grabbing the maximum accuracy on the validation set after the model was trained and finding the corresponding regularization parameter that yielded that accuracy.

Once the “optimal” regularization parameter was picked (C=0.050), the original dataset was trained and tested with this parameter. The aim for this was to observe if the test errors changed from before and if there was a significant change. The accuracy for training with the optimal parameter was 0.962667 and for testing it was 0.96000. Comparing these values with the average training and testing accuracies from part 1 and with the average training and testing accuracies using the K-fold method we can conclude that there is not much of a difference between the testing accuracies.

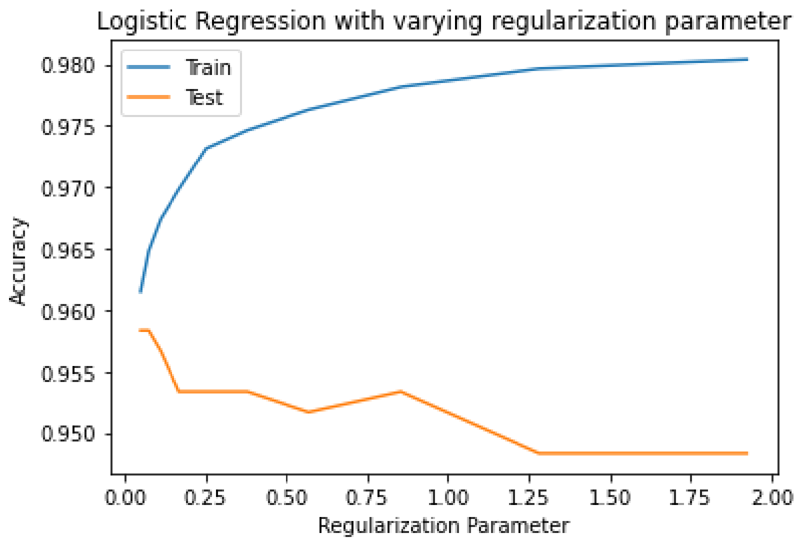

Figure 3: Training and Validation Accuracy with K-Fold and Logistic Regression

Part 3.2: Linear SVM

The procedure mentioned in section 3.2 was repeated but this time the model was a linear support vector machine. The table below shows some statistics of the training and testing accuracies when the model has been trained with j-1 folds and tested on the validation set and figure 4 allows us to visualize the trend. The log scale for the regularization parameter in this section was the same as the one used in part 2 ($C_0 = 0.01$ and alpha = 1.5).

	Min	**Max**	**Average**
Training	Accuracy: 0.954630 Regularization: 0.010	Accuracy: 0.978704 Regularization: 0.384434	Accuracy: 0.96798
Testing	Accuracy: 0.946667 Regularization: 0.010	Accuracy: 0.956667 Regularization: 0.050625	Accuracy: 0.949666

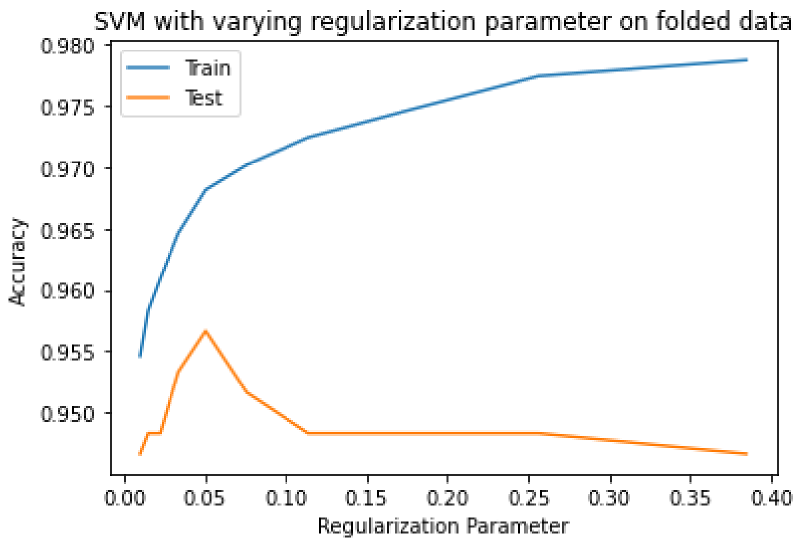

Figure 4: Training and Testing Accuracy using K-Fold and Linear SVM

The trend that was noticed in part 2 is also noticed here with the model trained on the k-folded data and tested using the validation set. The model overfits the dataset (performs better on the training data than on the validation set). However, in this case the overall testing accuracy decreased as the regularization parameter increases despite having a similar fluctuation as seen in part 2.

In this method, the optimal regularization parameter needed to be found, therefore the way this was picked was by grabbing the maximum accuracy on the validation set after the model was trained and finding the corresponding regularization parameter that yielded that accuracy.

Once the "optimal" regularization parameter was picked (C=0.050625), the original dataset was trained and tested with this parameter. The aim for this was to observe if the test errors changed from before and if there was a significant change. The accuracy for training with the optimal parameter was 0.967667 and for testing it was 0.959000. Comparing these values with the average training and testing accuracies from part 2 and with the average training and testing accuracies using the K-fold method we can conclude that there is not much of a difference between the testing accuracies.

All in all, the training and testing accuracies of the two methods after using the K-Fold Cross Validation method turned out to be the same. It was surprising that despite the log scaled being different for the two methods the optimal regularization parameter turned out to be approximately 0.05 for both the methods. This resulted in the two methods learning the full dataset with a similar accuracy. The table below helps in visualizing the training and testing accuracies for both the methods after the K-Fold Cross Validation.

	Training	**Testing**
Logistic Regression	0.962667	0.960000
Linear SVM	0.967667	0.959000

Part 4: Kernelized SVM

In this section the model that was used was a kernelized SVM with the Gaussian kernel. This method is essentially the same as part 2 and part 3.2 however, because the kernel is Gaussian it has an additional parameter of "scale" or "gamma". The objective of this method was to find the optimal pair of gamma and regularization parameter for each value of gamma that would yield the highest training and testing accuracies. To do this, the K-Fold method was employed again to find the optimal pairs and then the full dataset was trained and evaluated.

Before the model could be trained, a logarithmically spaced grid for gamma values (10 values in total) was computed (like what was done for the regularization parameter). Some pseudocode will assist in understanding how the algorithm worked.

```
For gamma in gamma_list:
    Optimal_pairs = {}
    Get the validation set and the remaining folds for training
    For regularization in regularization_list:
        Train_model()
    Optimal_pairs[max_pair] = max_accuracy
```

In the pseudocode above, the program would pair each gamma value with every regularization parameter and then train the model. After the model has been fitted, the testing accuracy (on the validation set) was appended to a dictionary with its corresponding key (gamma and regularization pair). When all the pairs had been computed, the pair with the highest accuracy for each gamma value was appended to another list. Lastly for this list of all pairs with the max accuracy, the model was re-trained on the full dataset and the training and testing accuracies were recorded. Figure 5 below shows how the training and testing accuracies changed with respect to gamma.

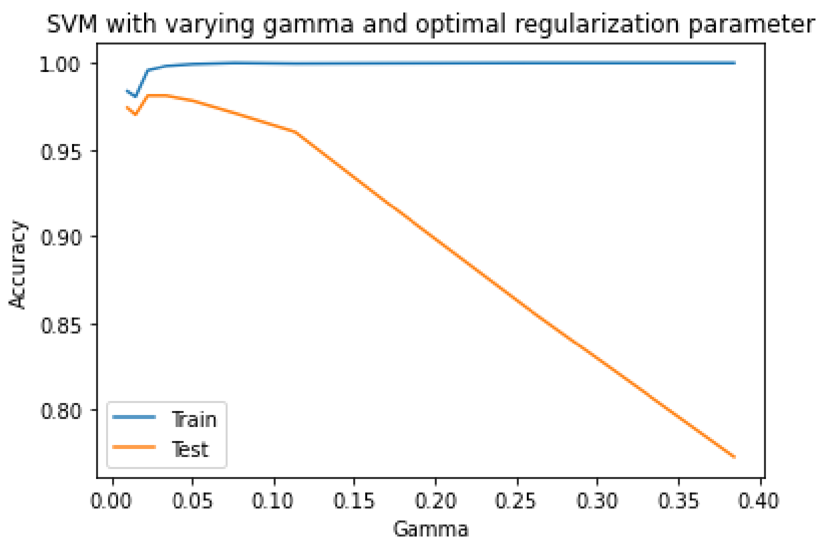

Figure 5: Training and Testing Accuracy of SVM with gamma

From the graph it can be noticed that the training accuracy for this was consistent and stayed at approximately 1.00 (except for a few values for gamma in the beginning). However, the testing accuracy had an increase for a single value of gamma after which it began to decline. Comparing this model with the linear SVM (with cross-validation, figure 4) we can conclude that this model overfits the data a lot more than the linear SVM does. The training accuracy for the linear SVM also increased as the regularization parameter increased however, this was not as consistent as the Gaussian Kernel SVM. With respect to the testing accuracy, the linear SVM performed a lot better, it has a slight increase in the accuracy after which it slowly began to decline. Whereas, in the case of the Gaussian Kernel SVM the testing accuracy had a very steep decline and went below 0.9 (which was not seen in the linear SVM).

All in all, it can be concluded that the Gaussian Kernel SVM performed far better on the training set than the linear SVM but the opposite was seen for the testing set. Therefore, it is safe to say that the Gaussian Kernel SVM overfits the data a lot more than the Linear SVM does. However, different conclusions could have been made if the values of gamma and/or regularization parameter were picked from a different logarithmic scale.