CS-UY 4563: Machine Learning

Final project Written Report

Image Classification of Plant Seedlings

April 26, 2023

Professor Linda M Sellie

Usaid Malik

Introduction

In this project, a data set was obtained from the Department of Electrical and Computer Engineering at Aarhus University. The data set was of plant seedlings. The data set consisted of over 3,000 images of 12 species of plants at different growth stages. The images were in RGB with a resolution of 10 pixels per mm.

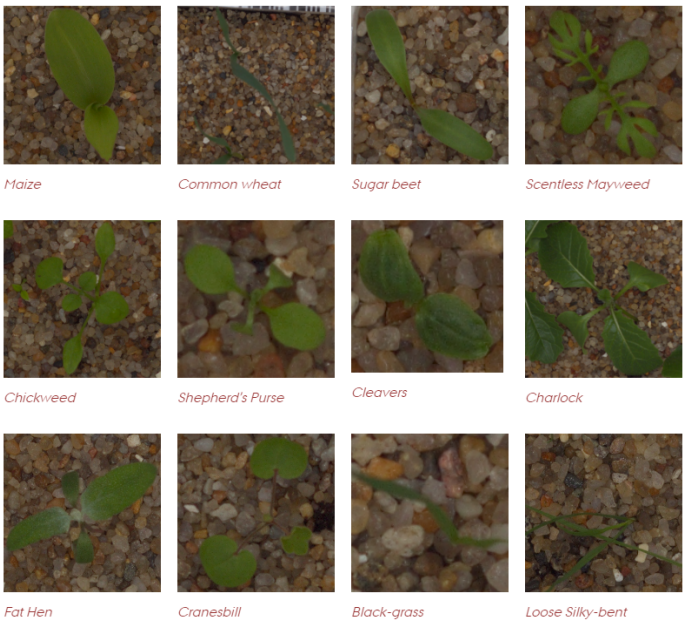


Figure 1: The Plant Species and Images

The data received was originally of the plants in their surrounding habitats. The university had a data set that was segmented, meaning the environment was removed and replaced with black pixels and only the plant remained. This data set was trained so as to reduce irreducible errors from the outside world.

Figure 2: (Left) Unsegmented Maize sample. (Right) The same Maize sample segmented

The data set was then transferred into the program and stored in a NumPy array. The NumPy arrays contained the RGB values for an image and another NumPy array stored the corresponding label for the plants from 0 to 11, for the 12 species of plants. The segmented data set was then split randomly into a training and a testing set with 80% of the data used for training and 20% for testing. Different models were trained and compared to determine the best model for the data set. Logistic regression, support vector machines, and neural networks were trained to be able to classify the 12 species of plants. The models were run under varying conditions under the same randomly split data. Examples of these conditions are varying regularization strengths, different transformations of the features, and different image resolutions.

Preprocessing

The data was preprocessed before training. Initially, the images within the set were non-uniform. Different images had different resolutions and as such made it difficult to train a model as some images would have more parameters than other images and vice versa.

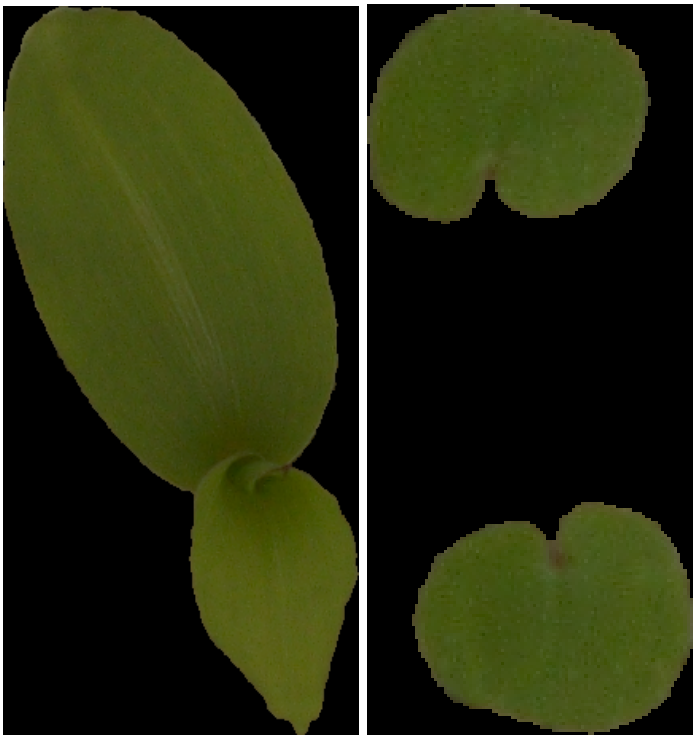


Figure 3: an example of two images from the data set with different resolutions.

To combat this issue, each image had its resolution changed to a square aspect ratio. Initially, the resolution chosen was 36 x 36. This resolution was arbitrarily chosen, however, it was based somewhat on the grounds that it was small enough that the model wouldn't take too much time to train and large enough such that enough features were present to be able to train the model.

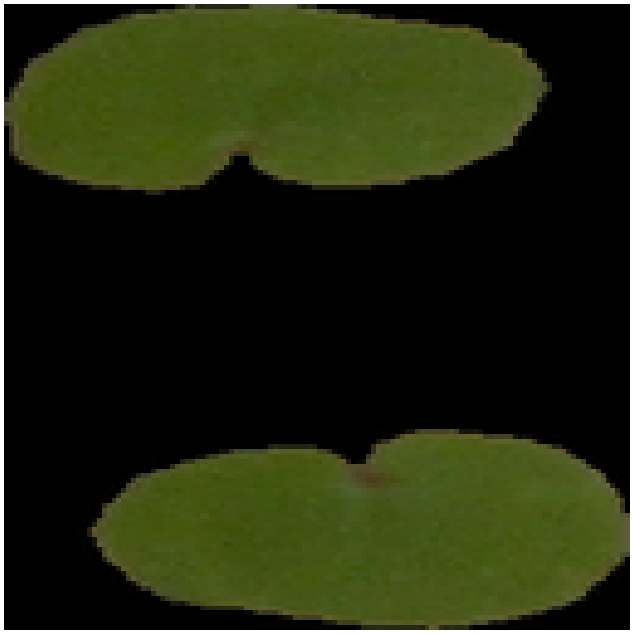


Figure 4: an example of a training example after rescaling

The data set was slightly unbalanced initially with there being less of the species Shepherd's Purse, Maize, and Common Wheat Compared to the other species.

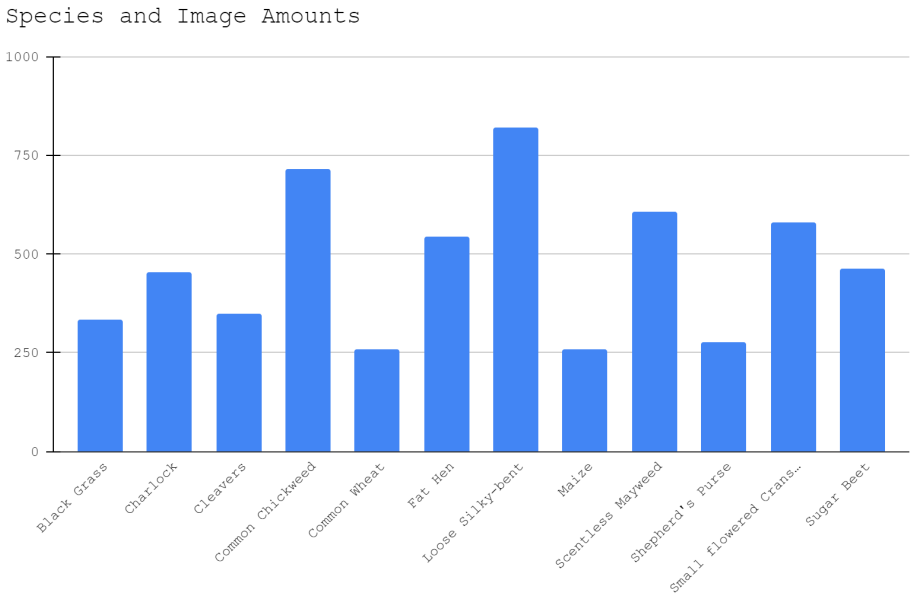


Figure 5: The initial distribution of data

As shown by the chart the data was not completely symmetric in the sense that all species had the same image amounts. As such another data set was created which undersampled all species and included 250 images each for each species.

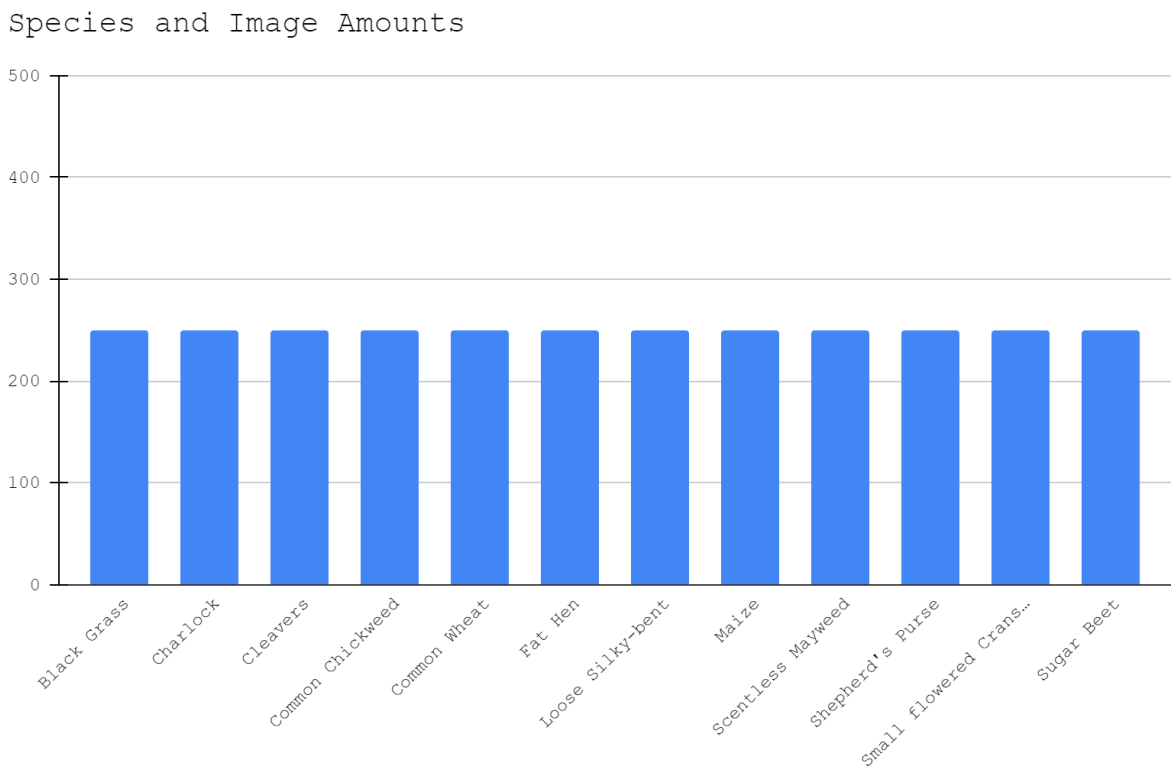


Figure 6: All data after undersampling

Then to observe the effects of undersampling and not undersampling some models were trained on both data sets.

Afterwards, as the data was not in the program and had to be inserted into the notebook, the image files were read and placed into a NumPy array. In the program, the images were converted to their RGB values using the Pillows Python library. The array of RGB values was

then flattened and placed into a NumPy vector with the correct corresponding label. The labels ranged from 0 to 11 and corresponded to each of the different species in the set. Each image was then stacked into the feature vector with the features being the pixel RGB values. The images were also rescaled to 36 x 36, 50 x 50, and 128 x 128 pixels and these different pixel-size images were trained on the various models.

Finally, the feature vector was scaled using the StandardScaler() object from the sklearn library that zero-centered all the data and put the pixel values between zero and one.

Logistic regression

Logistic regression was one of the models used. The implementation of logistic regression was used from the sklearn python library. The model was initially trained on the data without undersampling and then was trained on the data set with undersampling to observe the difference in accuracies between the two models.

The accuracies were then plotted against the iterations to see the effect on the training accuracy depending on the iterations of the model. The iterations were implemented by changing the max_iter parameter of the model provided by sklearn.

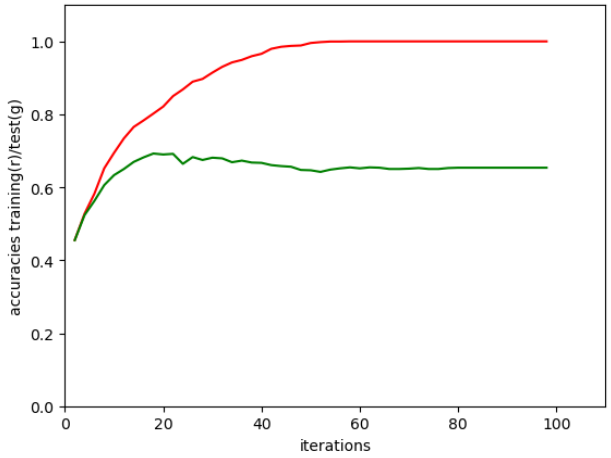


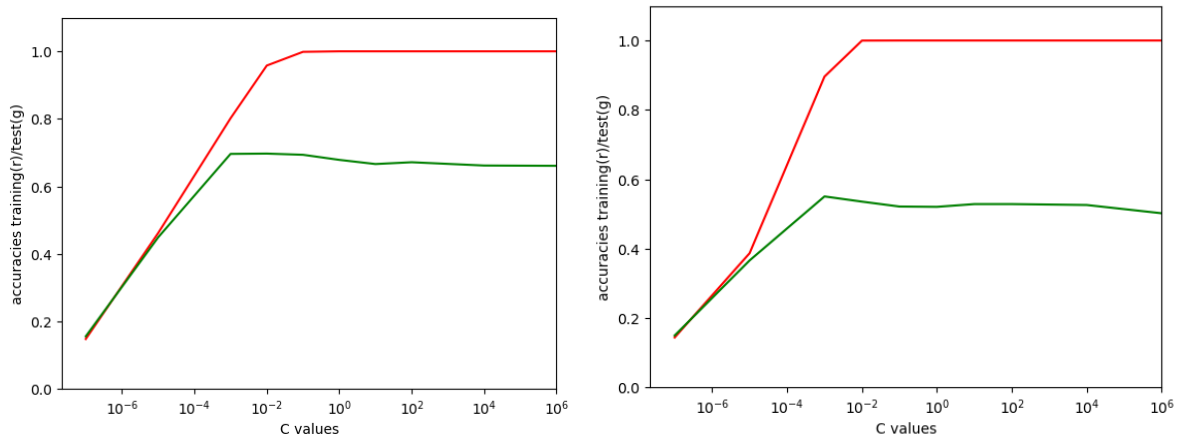
Figure 7: The graph of the accuracies against iterations

From the results of the iterations versus accuracy graph an iteration amount of 80 was chosen for the proceeding models as 80 iterations were the minimum amount of iterations that gave a similar accuracy as the iterations above 80.

The model was then initially trained on the unchanged data set with all features scaled via the StandardScaler(). Then the model was trained with different combinations of L2 regularization strengths, feature transforms, and image resolutions. The graphs of the accuracies on the training data and test data are shown below. The graphs are separated by their image resolution.

The accuracy is plotted against C, the inverse of the regularization strength. The C values chosen were C = [0.0000001, 0.00001, 0.001, 0.01, 0.1, 1, 10, 100, 10000, and 1000000]. The c-axis is on a logarithmic scale. The training accuracy is shown in red and the test accuracy is shown in green.

Resolution 36 x 36 Pixels



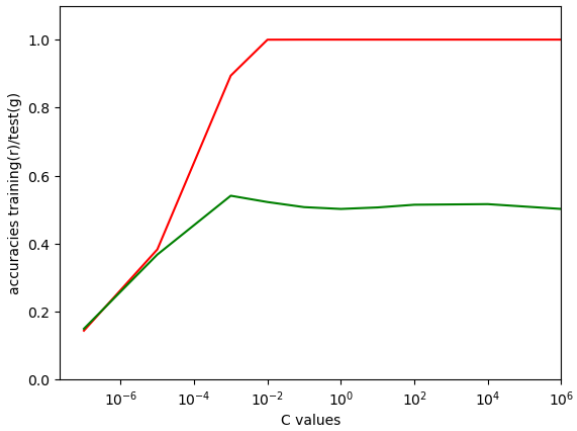
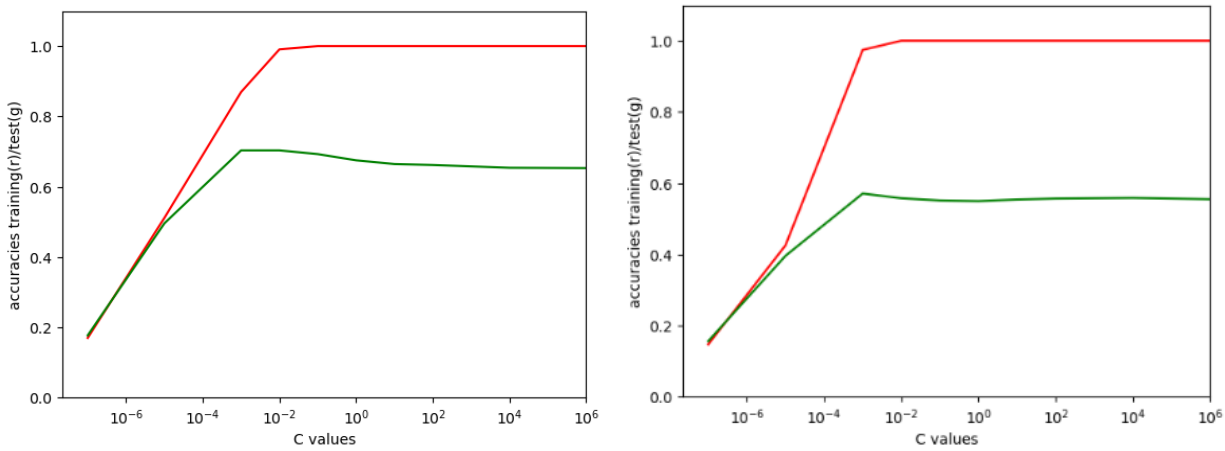


Figure 8: Left: No Feature Transformation | Right: Squared Feature Transformation | Bottom: Cubic Feature Transformation

Resolution 50 x 50 Pixels



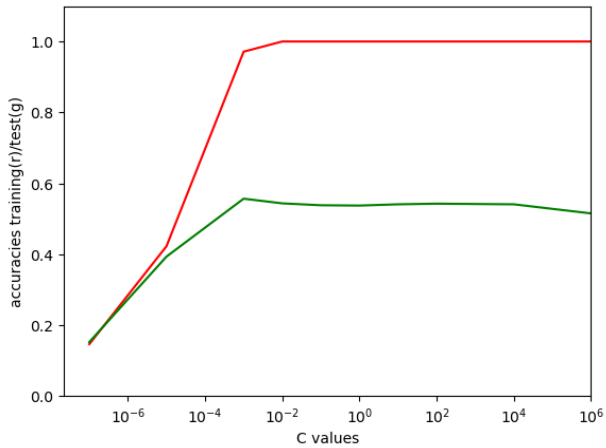


Figure 9: Left: No transformation | Right: Squared Feature Transformation | Bottom: Cubic Feature Transformation

Resolution 128 x 128 Pixels

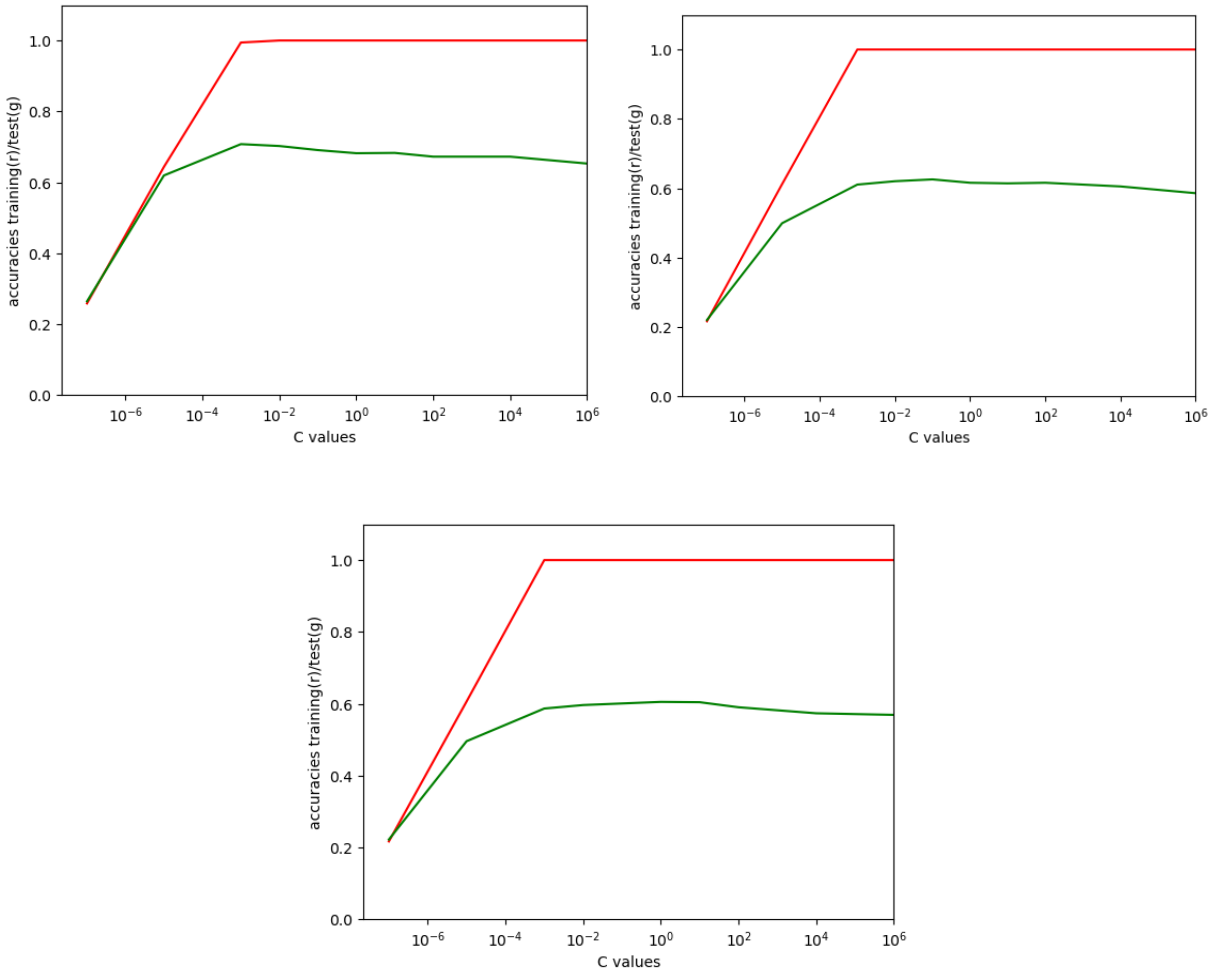


Figure 10: Left: No transformation | Right: Squared Feature Transformation | Bottom: Cubic Feature Transformation

Afterwards the confusion matrix was created for the highest accuracy model in the data set that did not have undersampling. The highest accuracy model was the model with no feature transforms, with 128 x 128-pixel sizes and an inverse regularization strength of 1e-5.

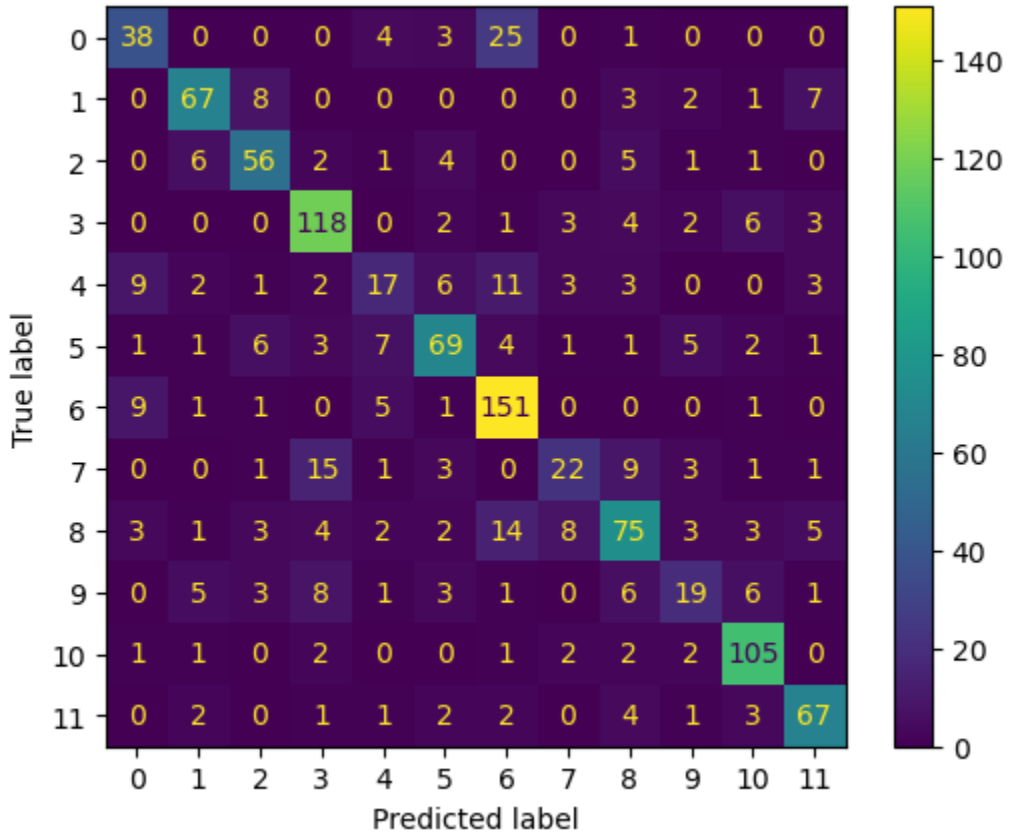


Figure 11: Confusion matrix for the highest performing model

After observing the confusion matrix it was apparent the model was giving inaccurate results as the more present data sets dominated the correct predictions. As such the logistic regression models were trained on the undersampled data set only with L2 and L1 regularization.

Next, the model was trained using the same combinations as above as well as L2 regularization, however, once again this time the model was trained on the data set that was undersampled to observe the effects of undersampling on the model's accuracy.

(Undersampled) Resolution 36 x 36 Pixels

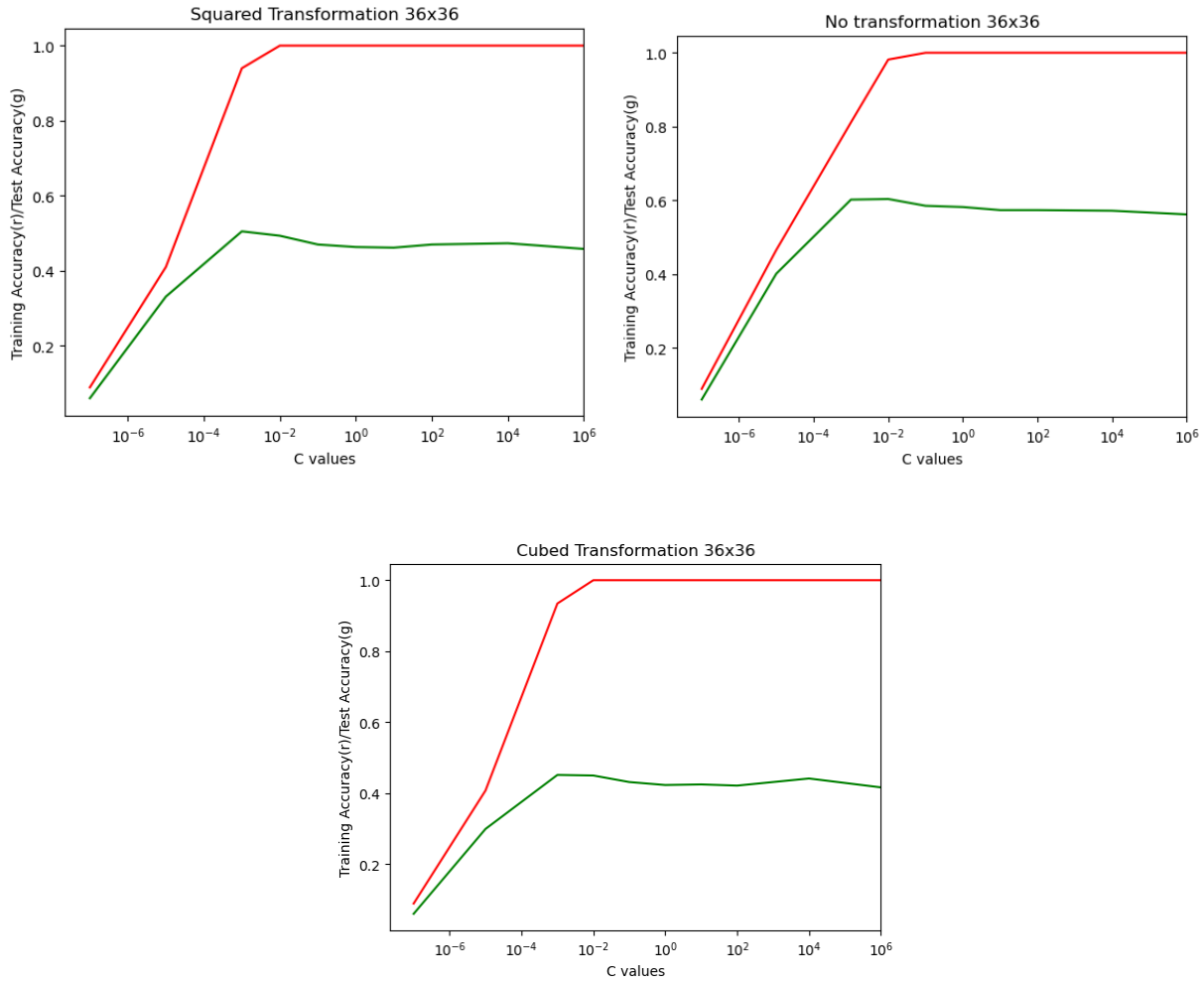


Figure 12: trained model's accuracies on undersampled data with 36 x 36-pixel resolutions

(Undersampled) Resolution 50 x 50 Pixels

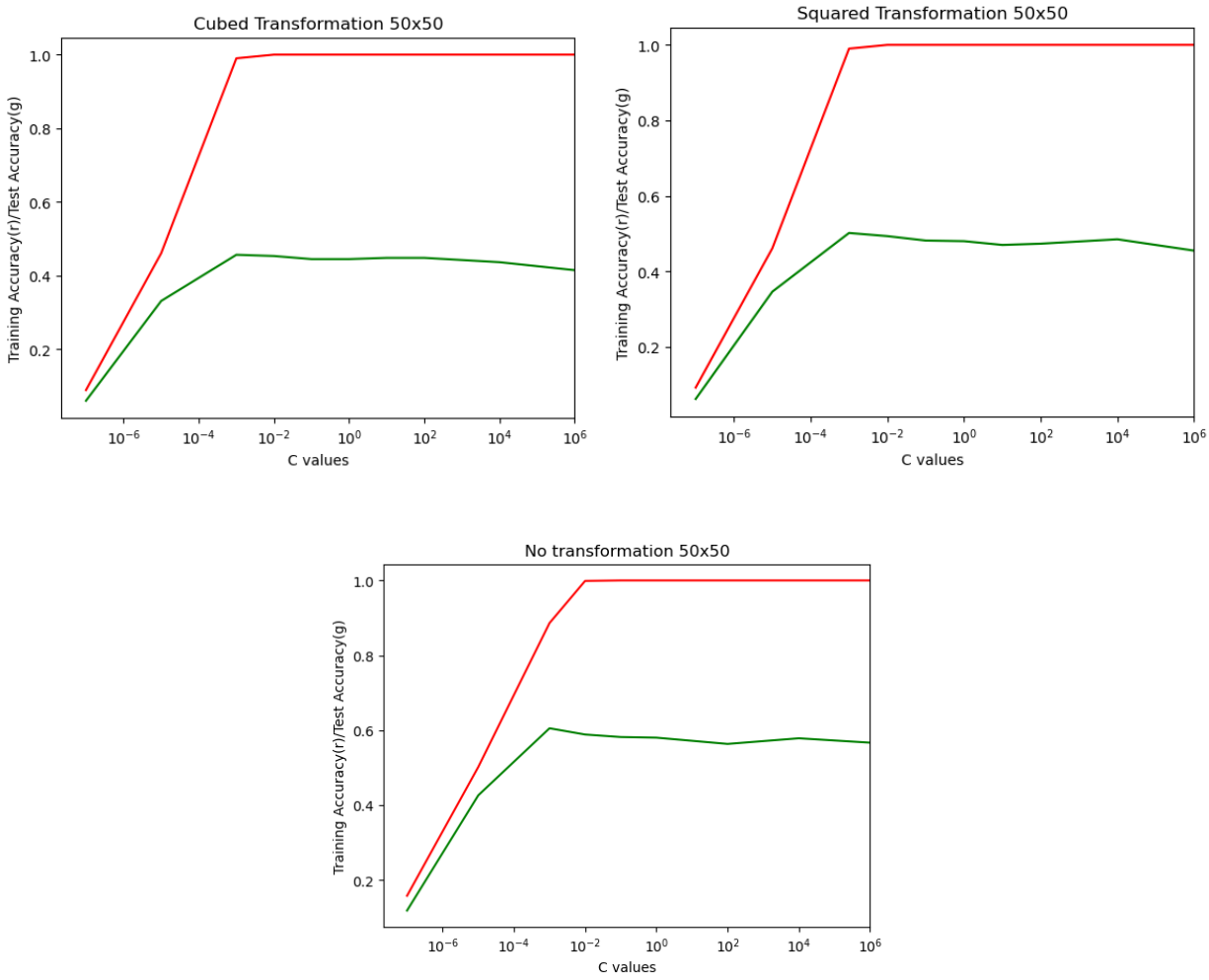


Figure 13: trained model's accuracies on undersampled data with 50 x 50-pixel resolutions

(Undersampled) Resolution 128 x 128 Pixels

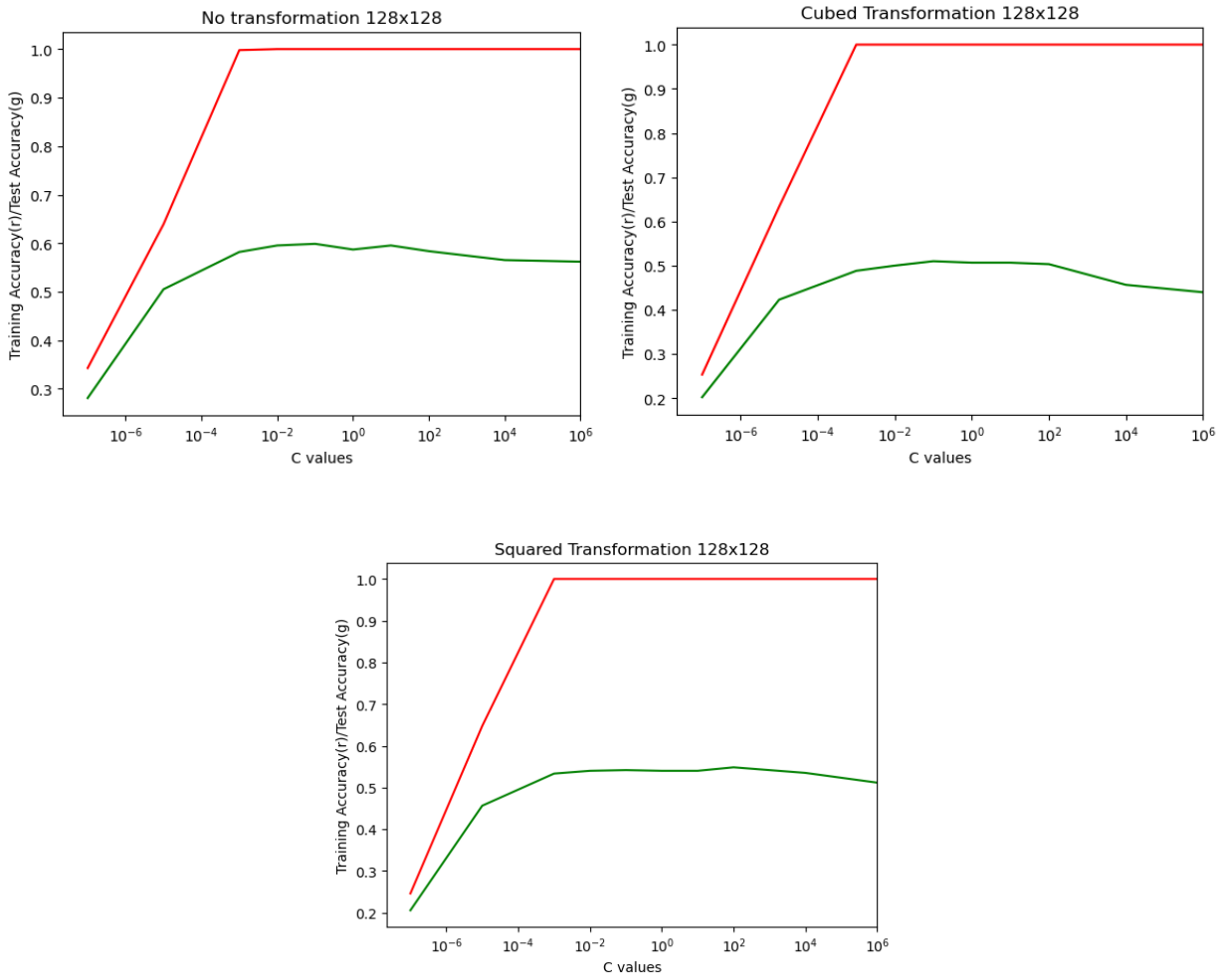


Figure 14: trained model's accuracies on undersampled data with 128 x 128-pixel resolutions

Next L1 regularization was implemented on the model only on the undersampled data set using the same combination of feature transforms and image resolutions.

L1 Regularization Resolution 36 x 36 Pixels

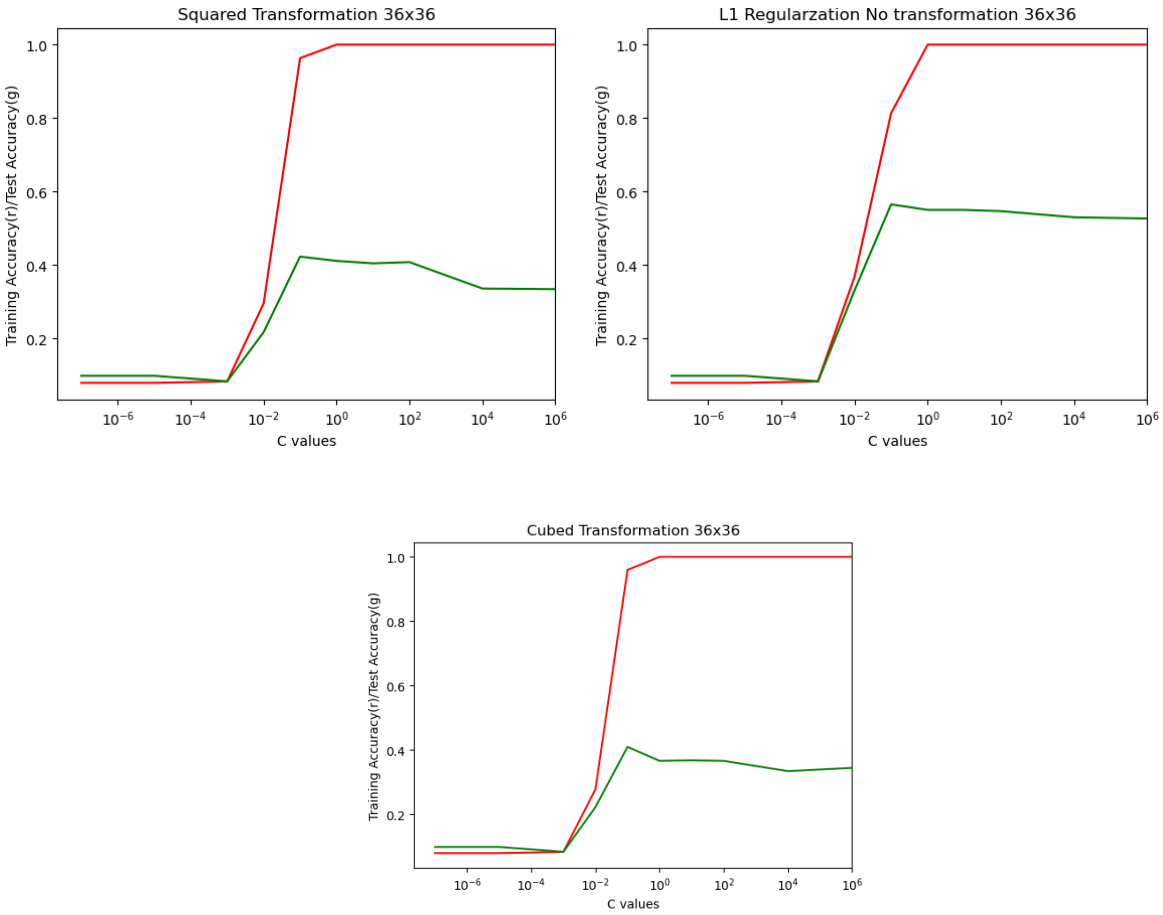


Figure 15: different feature transforms effects on trained model's accuracies on undersampled data with 36 x 36-pixel resolutions and L1 regularization

L1 Regularization Resolution 50 x 50 Pixels

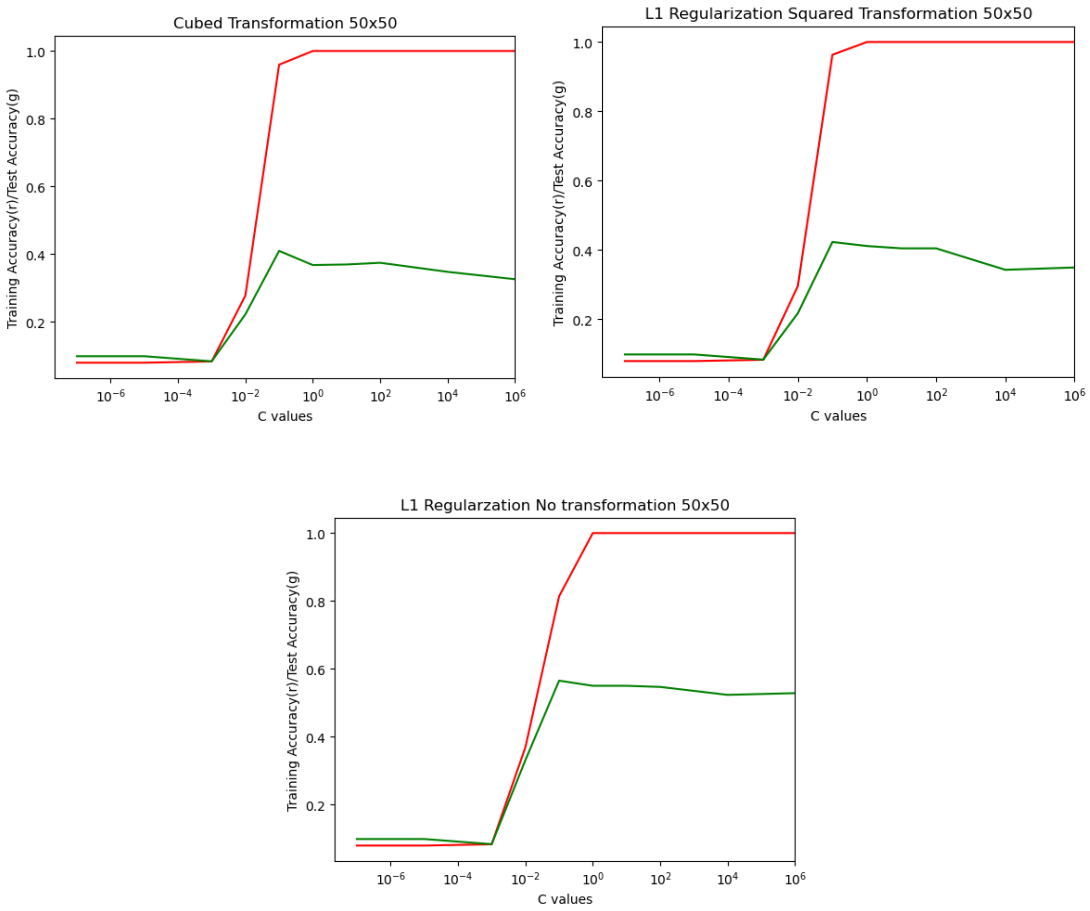


Figure 16: different feature transforms effects on trained model's accuracies on undersampled data with 50 x 50-pixel resolutions and L1 regularization

L1 Regularization Resolution 128 x 128 Pixels

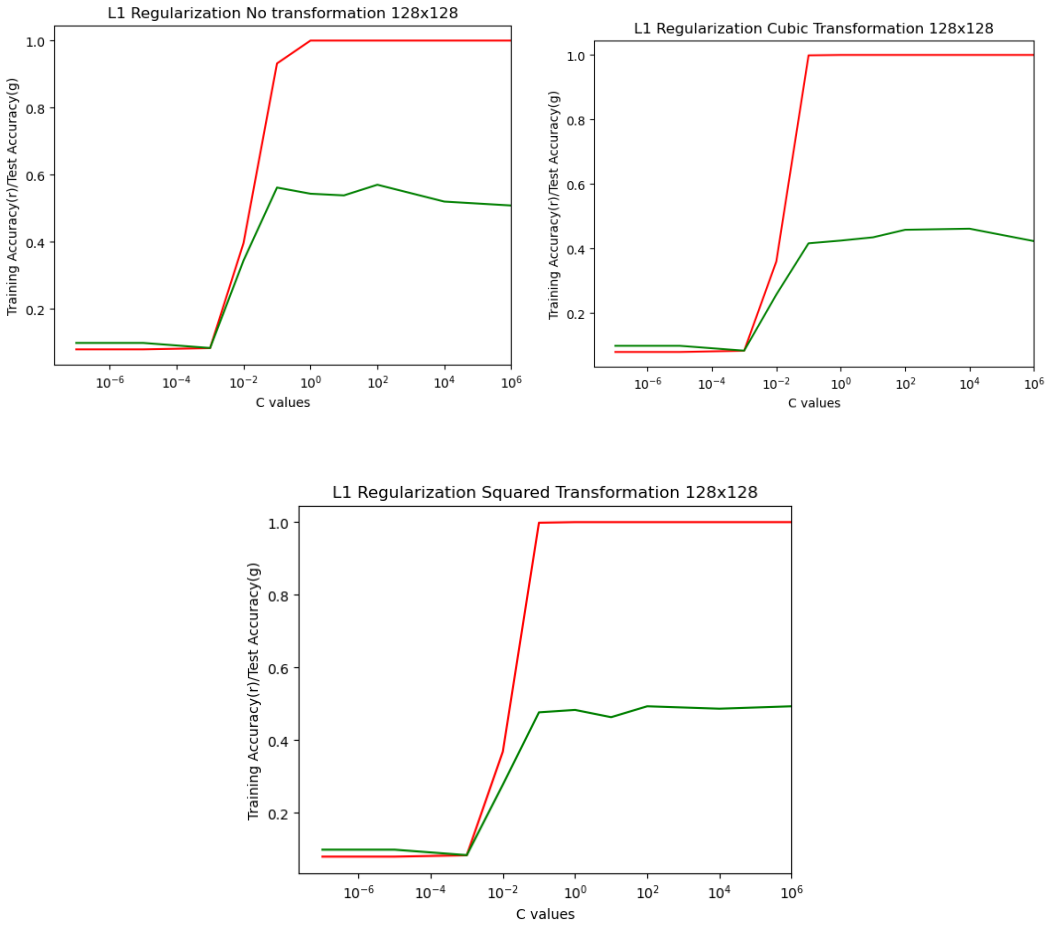


Figure 17: different feature transforms effects on trained model's accuracies on undersampled data with 128 x 128-pixel resolutions and L1 regularization

Afterwards the confusion matrix was plotted for the highest accuracy model from the undersampled data set from the L2 and L1 regularization sets. The highest accuracy model proved to be with L2 regularization, an inverse regularization strength of 0.001, no feature transforms, and on 50 x 50-pixel images.

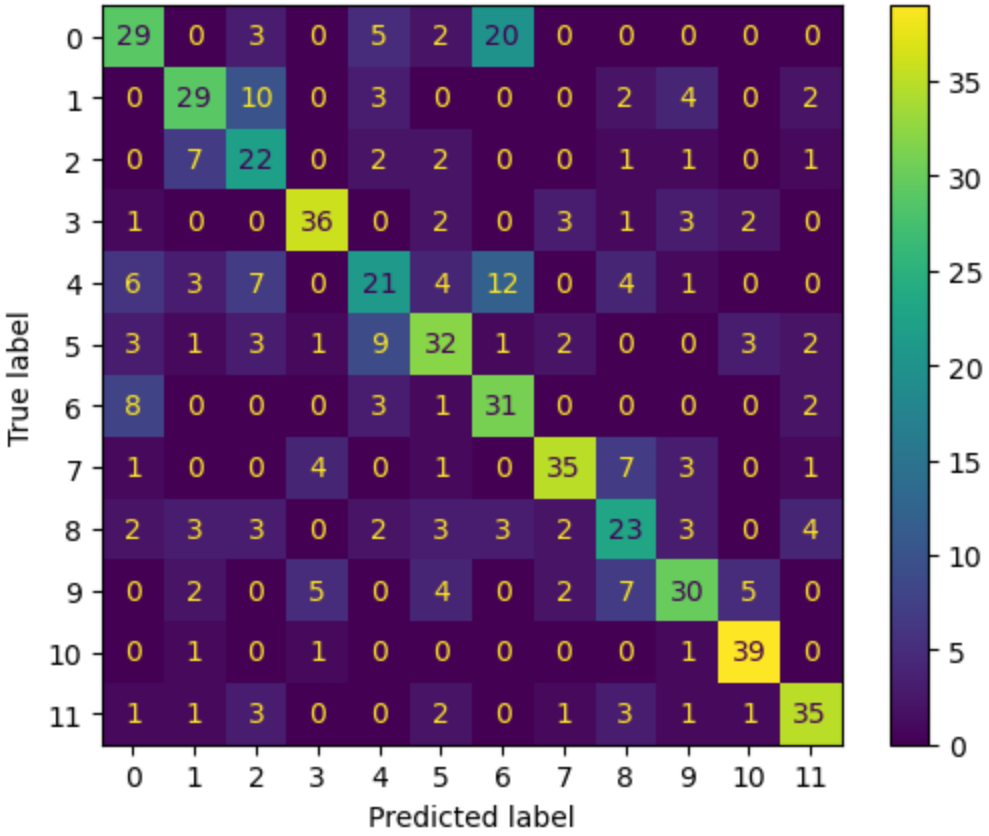


Figure 18: Confusion Matrix for the best-trained model on the undersampled data

The averaged precision, recall, and f score, for this model determined by the sklearn library were 0.6083, 0.602, 0.6008.

Neural Networks

Neural networks were another model used. The neural network was implemented using the MLP function in the sklearn neural network library. The neural network was tested on a multitude of different configurations. The alpha values for the regularization strength were [0.0000001, 0.00001, 0.001, 0.01, 0.1, 1, 10, 100, 10000, and 1000000]. The model was then

tested on 36 x 36-pixel size images as they resulted in the least computational time. The Neural network was tested using a logistic activation function, a RELU activation function, and a tanh activation function. The model was tested using 100 hidden layers, 100, 50, hidden layers, 100, 50, 30 hidden layers, and 200, 100, 50, and 30 hidden layers. The results of the accuracy were then plotted against the regularization strength for the different models and layers combinations. The model was also tested on the unaltered data and the undersampled data.

Unaltered Data:

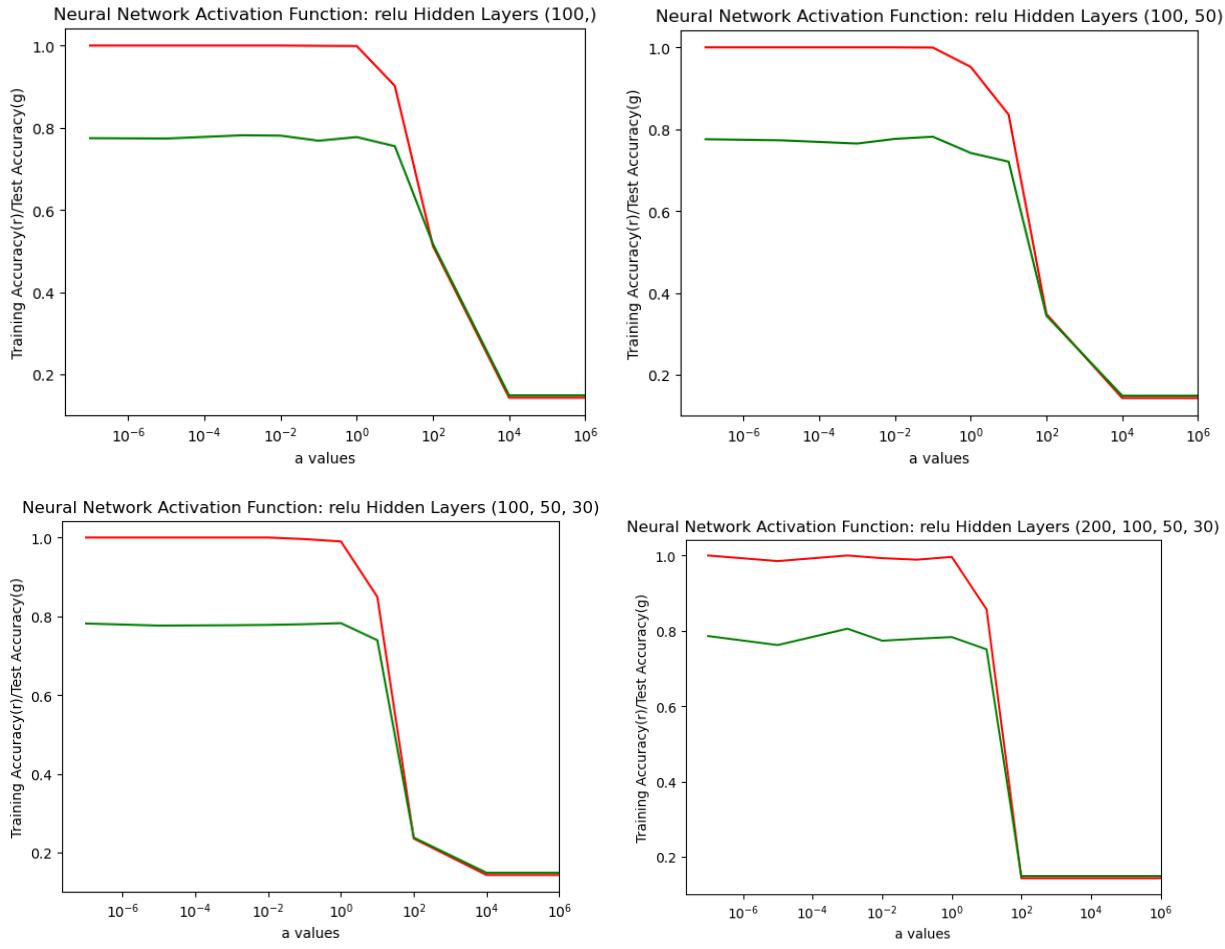


Figure 19: Graphs of Accuracies vs Regularization Strengths for a RELU activation function with hidden layers given in graph labels

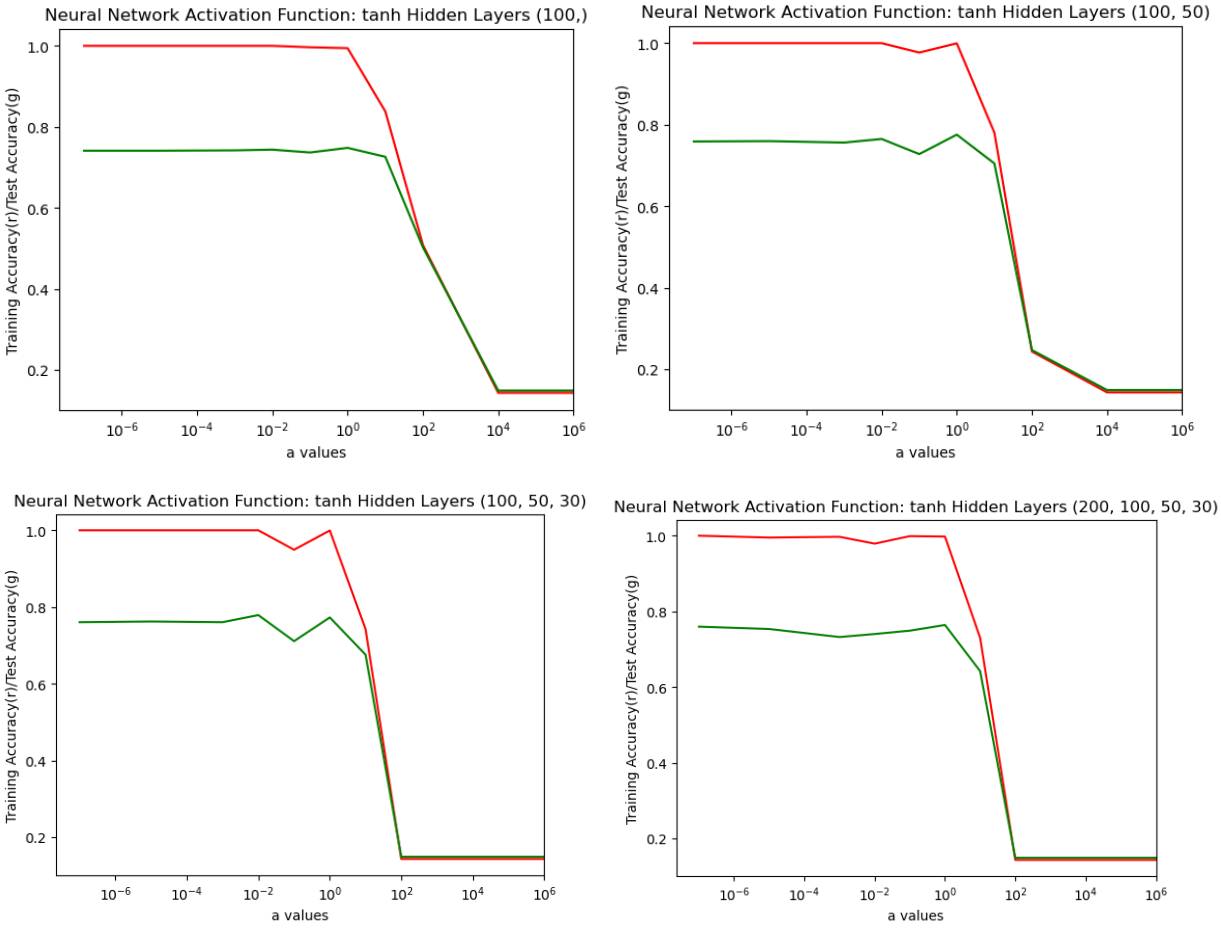
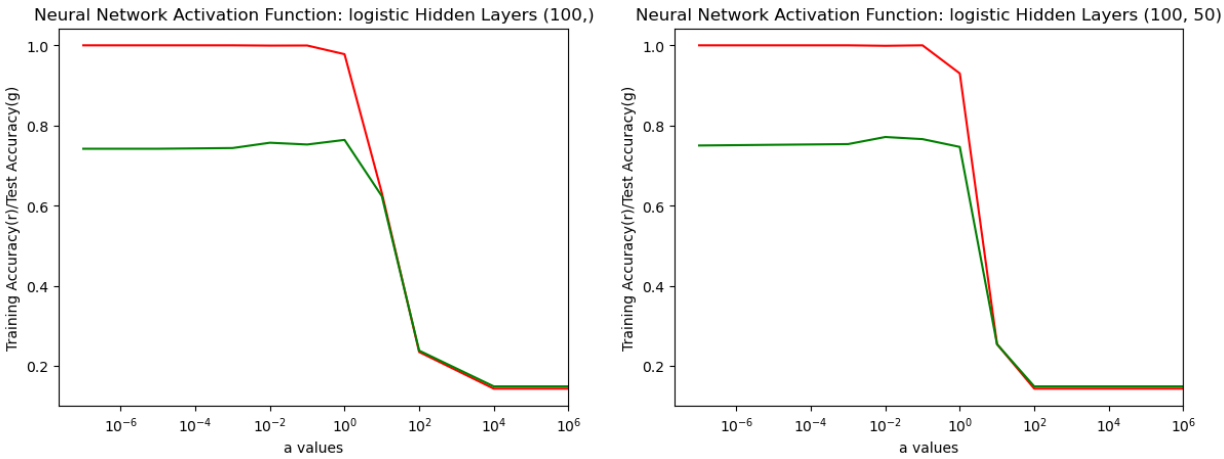


Figure 20: Graphs of Accuracies vs Regularization Strengths for a tanh activation function with hidden layers given in graph labels



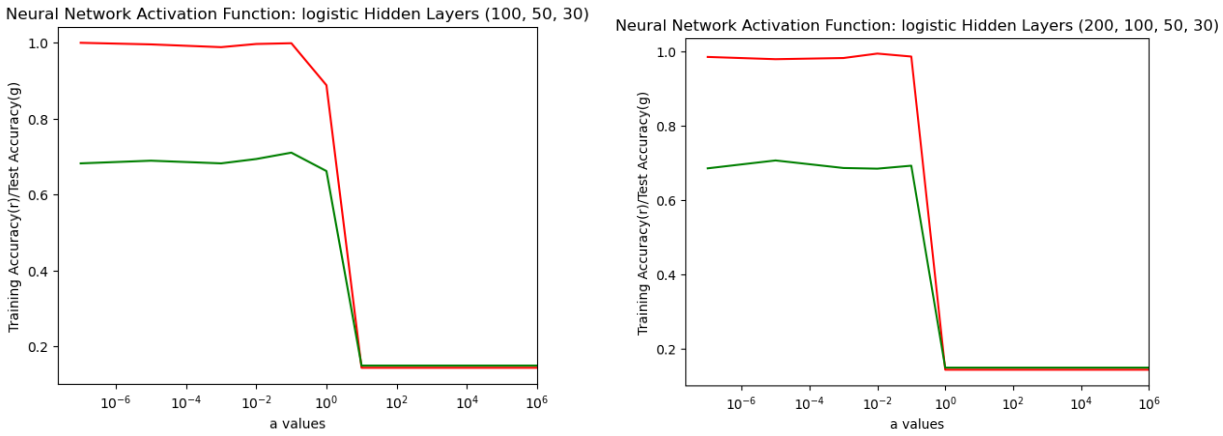


Figure 21: Graphs of Accuracies vs Regularization Strengths for a Logistic activation function with hidden layers given in graph labels

Next the confusion matrix for the best performing neural network model was created. The best model was the RELU activation function with 200,100,50,30 hidden layers with a regularization strength of 0.001.

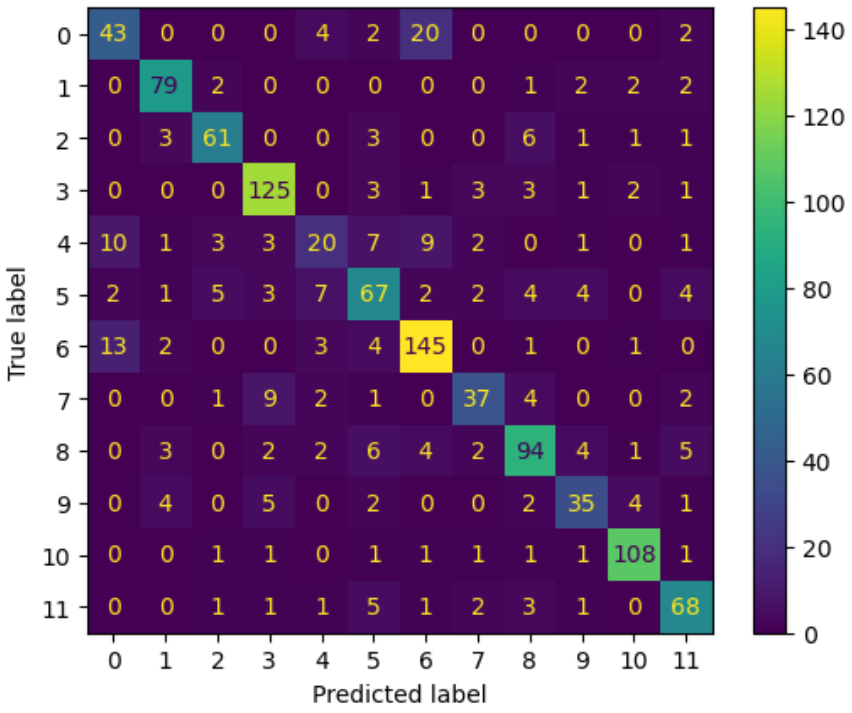


Figure 22: Confusion matrix of the best-performing model

Once again as the confusion matrix pointed out the issues with training on the full data the neural network was retrained this time on the undersampled data using the same combinations of neural networks as before.

Undersampled Data:

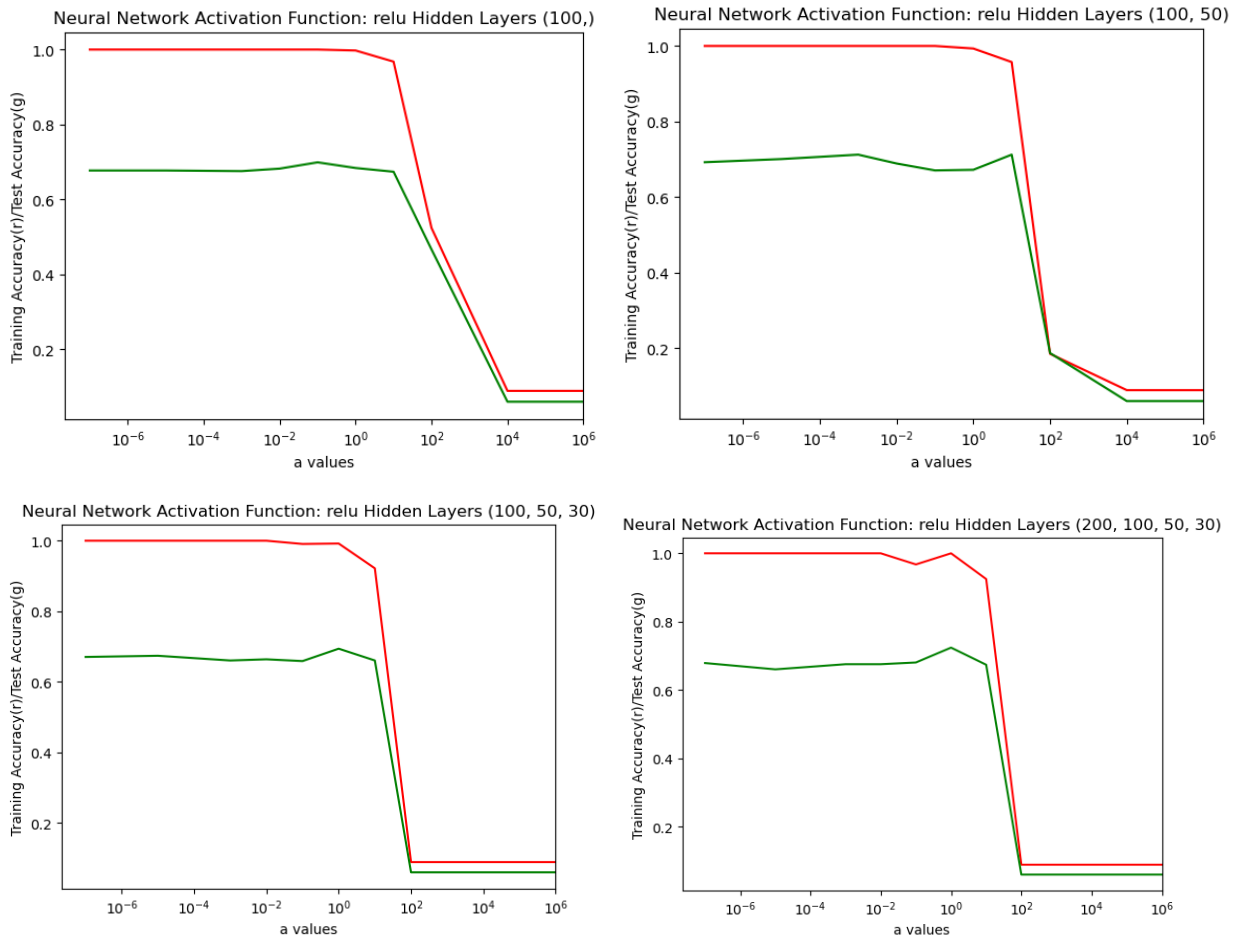


Figure 23: Graphs of Accuracies vs Regularization Strengths for the undersampled data and a RELU activation function with hidden layers given in graph labels

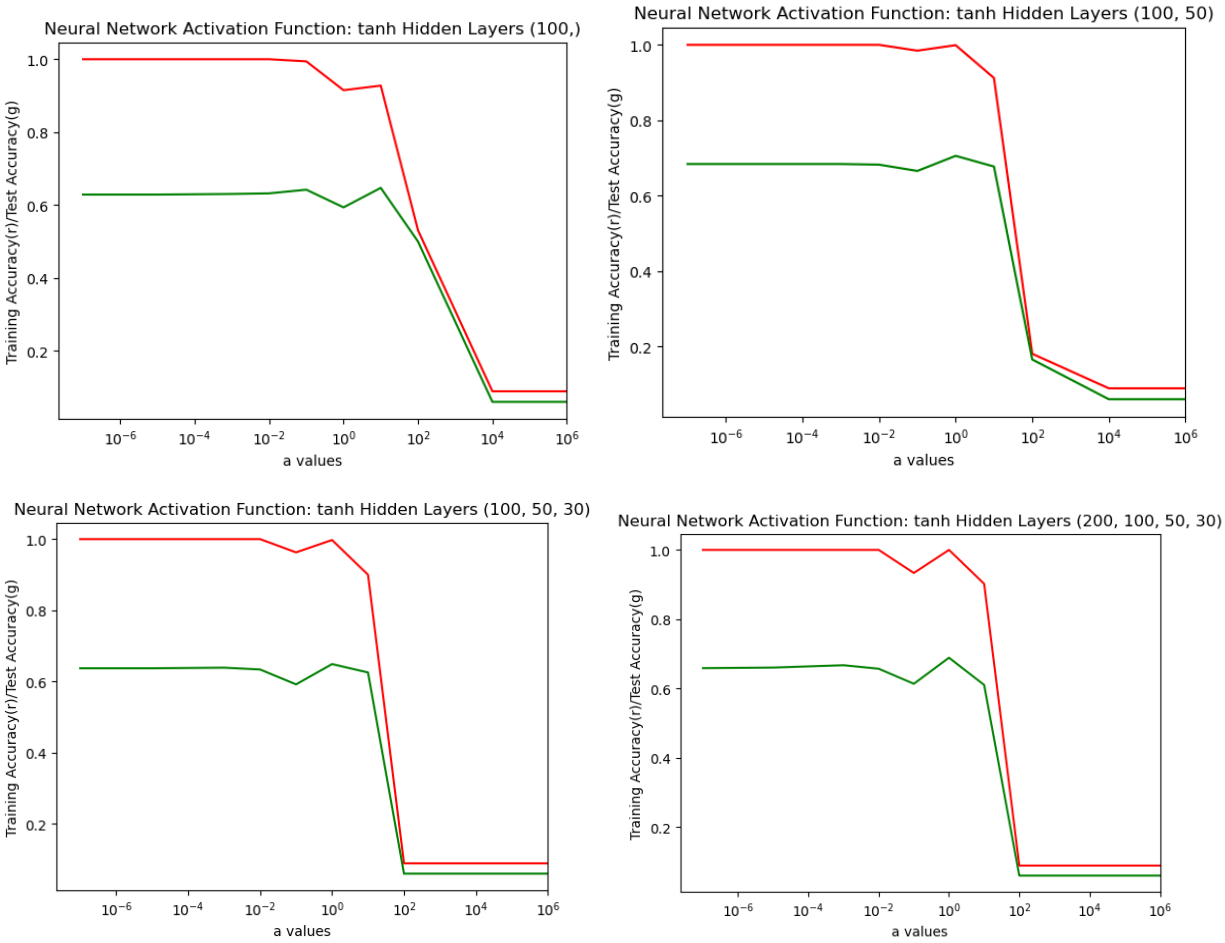
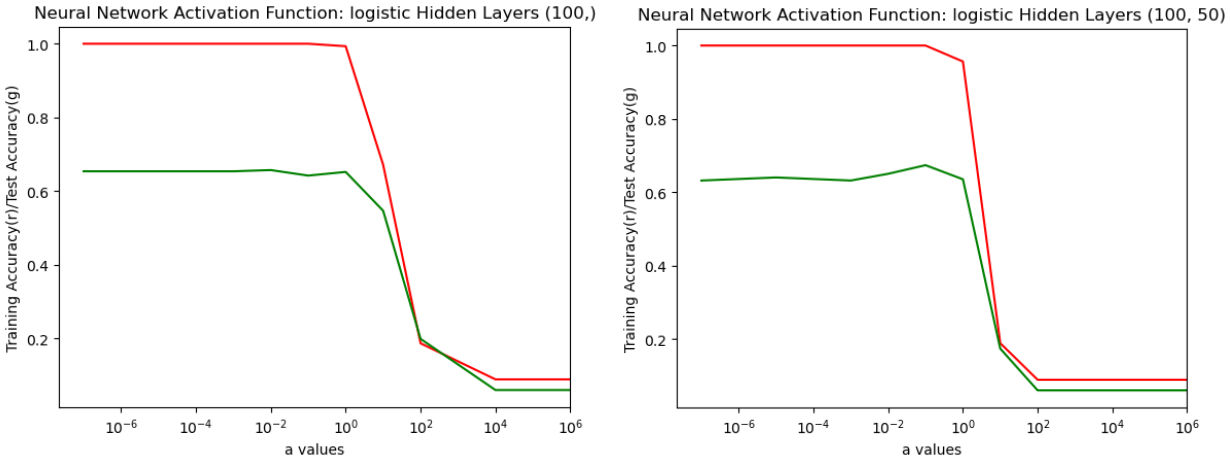


Figure 24: Graphs of Accuracies vs Regularization Strengths for the undersampled data and a tanh activation function with hidden layers given in graph labels



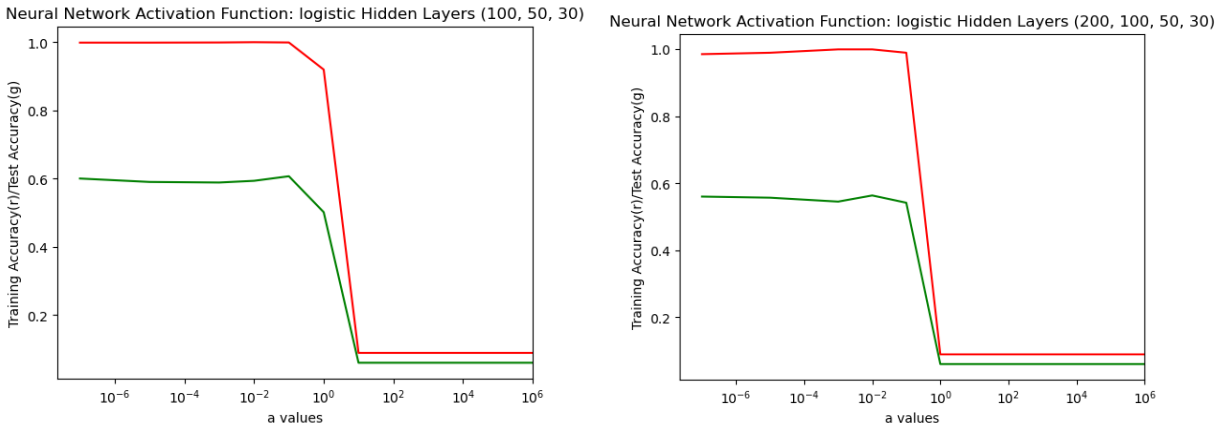


Figure 25: Graphs of Accuracies vs Regularization Strengths for the undersampled data and a Logistic activation function with hidden layers given in graph labels

The best model was found and the confusion matrix for the best model was plotted for the undersampled data. The best model proved to be the model using a RU activation function, 200,100,50,30 hidden layers, and a regularization strength of 1.

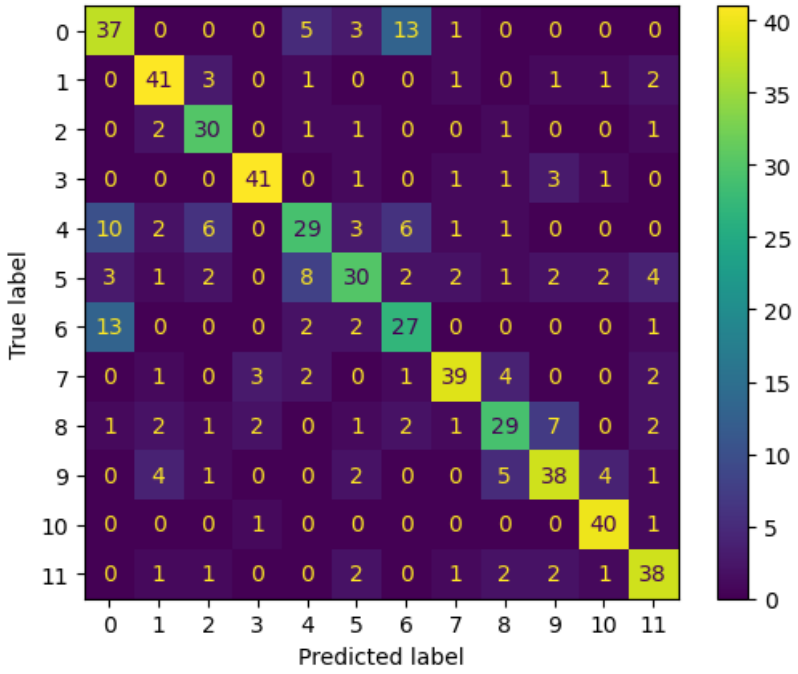


Figure 26: Confusion matrix for the best model on the undersampled data

The average precision, recall, and fscore for this model determined by the sklearn library were 0.702, 0.6906, 0.6874 respectively.

SVM

Support vector machines (SVM) was another model used. The model was trained utilizing sklearn's implementation for the SVM. This model was trained on different combinations of kernels and regularization strengths. The model was not trained on different feature transforms and pixel values as the previous results hinted that it was better to use no feature transforms. The pixel value chosen was 36 x 36 as it resulted in the least computational time and didn't result in substantial accuracy differences if training on the higher pixelated ratios. The kernel functions used were the linear, radial-basis function and polynomial degree 3. The inverse values for the regularization strength were [0.0000001, 0.00001, 0.001, 0.01, 0.1, 1, 10, 100, 10000, and 1000000].

The best hyperplanes found in each of the respective models and configurations yielded the test accuracy. The test accuracy and training accuracy was then plotted against the inverse regularization strengths as well as the different kernel functions.

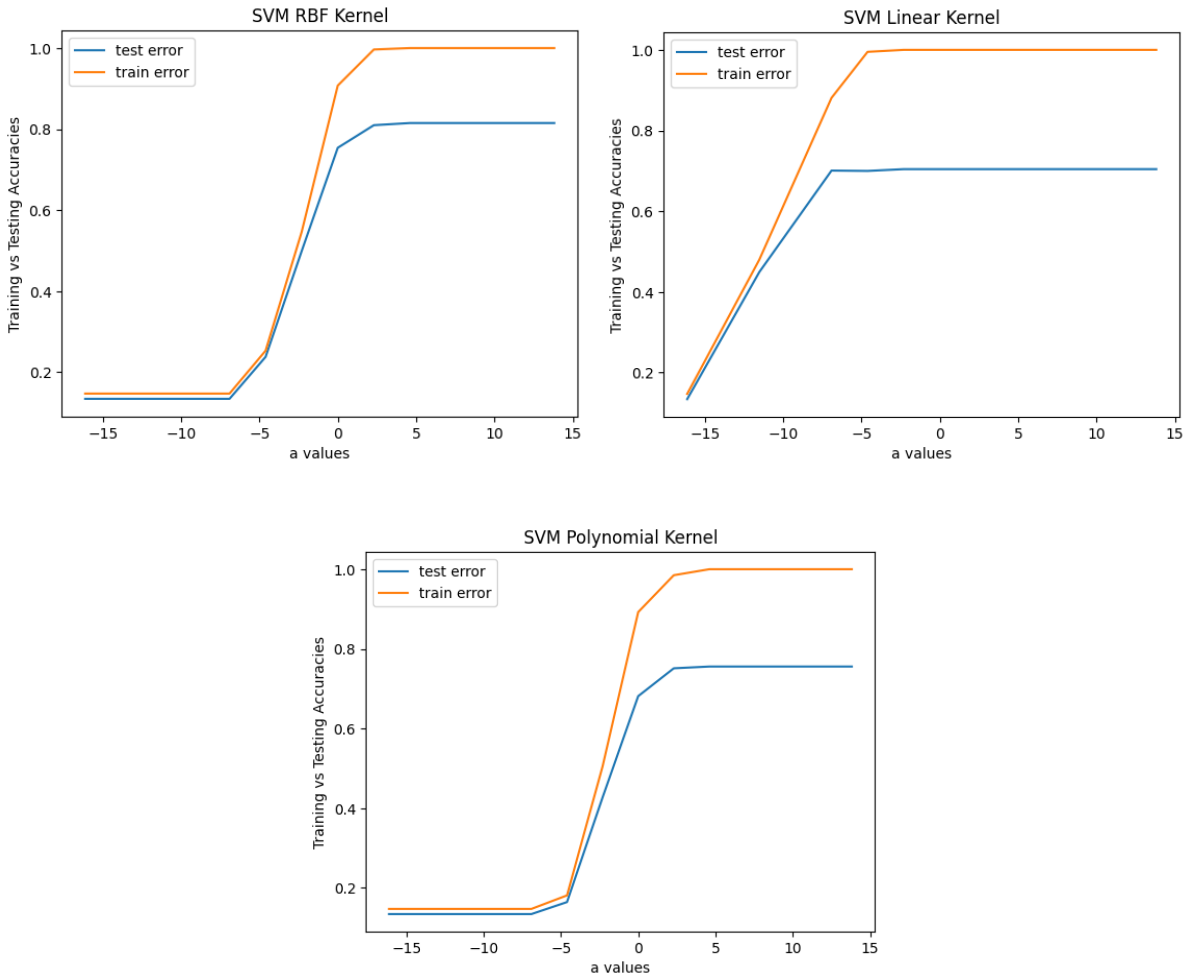


Figure 25: Graphs of Accuracies vs Regularization strengths for different kernels.
(Top left: Linear;Top right:RBF, Bottom: Polynomial Degree 3)

The best models across the kernels were found to be, 70.44%, 81.50%, 75.58% , for the Linear kernel,RBF kernel and Polynomial kernel, with inverse regularization strengths of 1, 1e+6,100 respectively.

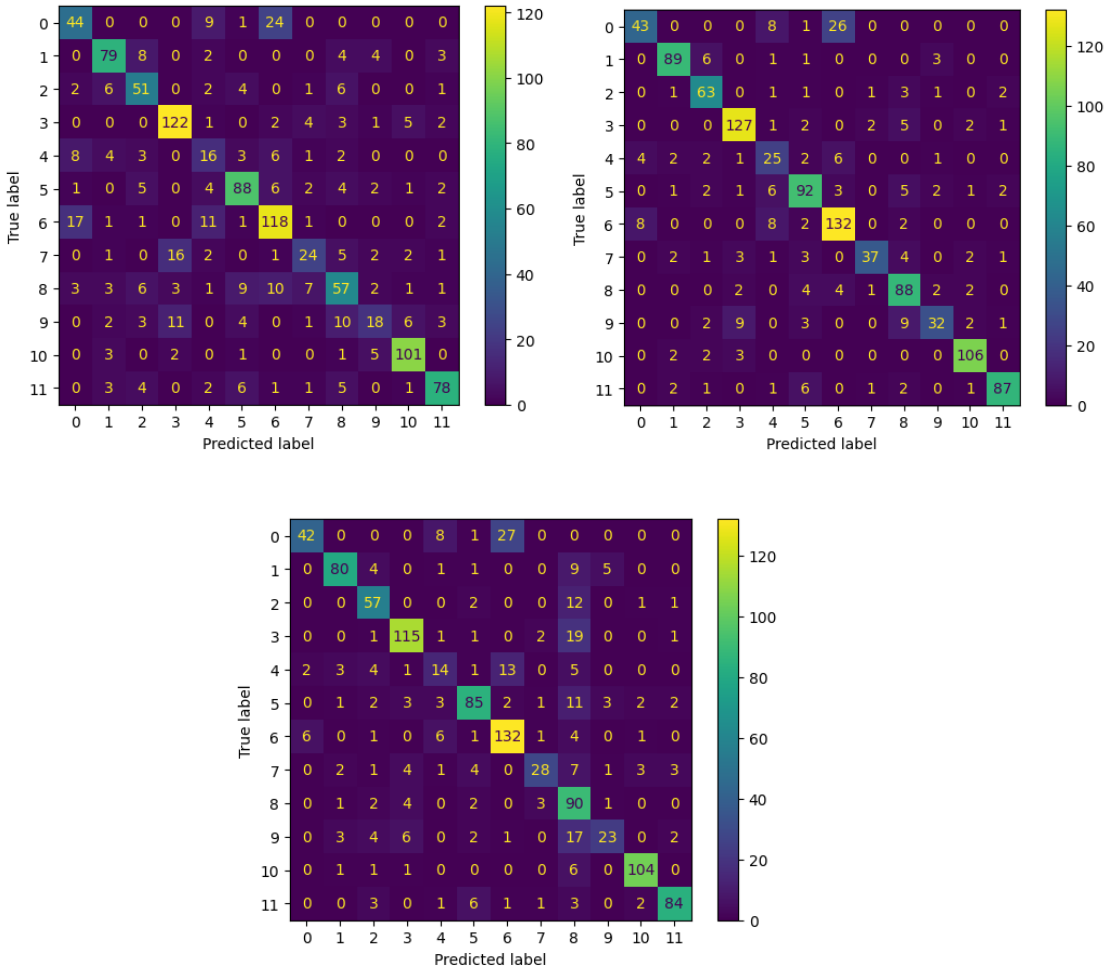


Figure 26: Confusion Matrix for different kernels

(Top left: Linear;Top right:RBF, Bottom: Polynomial Degree 3)

Results

The results of all the different models were plotted down and noted. The results were then compiled into tables.

Table 1: Results for Logistic Regression models vs inverse regularization strengths for L2 regularization

	C Values	1e-7	1e-5	0.001	0.01	0.1	1	10	100	1e+4	1e+6

No Feature transform 36 x 36Pixels	Training	0.1476	0.4615	0.8016	0.9578	0.9984	1	1	1	1	1
	Test	0.1555	0.4488	0.6961	0.697	0.6935	0.6784	0.6714	0.6714	0.6617	0.6608
Squared Feature Transform 36 x 36Pixels	Training	0.78	1	1	1	1	1	1	1	1	1
	Test	0.4814	0.5053	0.5088	0.5	0.4938	0.485	0.4717	0.4682	0.4664	0.4673
Cubic Feature Transform 36 x 36 Pixels	Training	0.0795	0.0795	0.0833	0.2778	0.9594	1	1	1	1	1
	Test	0.0987	0.0987	0.0836	0.2224	0.4097	0.3662	0.3679	0.3662	0.3344	0.3445
No Feature transform 50 x 50 Pixels	Training	0.0795	0.0795	0.0833	0.369	0.8138	1	1	1	1	1
	Test	0.0987	0.0987	0.0836	0.3311	0.5652	0.5502	0.5502	0.5468	0.5234	0.5284
Squared Feature Transform 50 x 50 Pixels	Training	0.1473	0.4252	0.9742	1	1	1	1	1	1	1
	Test	0.1564	0.3958	0.5707	0.5574	0.5512	0.5495	0.5539	0.5565	0.5583	0.5548
Cubic Feature Transform 50 x 50 Pixels	Training	0.146	0.4224	0.9713	1	1	1	1	1	1	1
	Test	0.1519	0.3931	0.5564	0.5433	0.538	0.5371	0.5406	0.5424	0.5406	0.515
No Feature transform 128 x 128 Pixels	Training	0.2582	0.6435	0.9943	1	1	1	1	1	1	1
	Test	0.2641	0.6193	0.7076	0.7023	0.6908	0.682	0.6829	0.6723	0.6723	0.6528
Squared Feature Transform 128 x 128 Pixels	Training	0.2163	0.6119	1	1	1	1	1	1	1	1
	Test	0.22	0.4991	0.6104	0.6201	0.6254	0.6157	0.614	0.6157	0.6051	0.5857
Cubic Feature Transform 128 x 128 Pixels	Training	0.2165	0.6061	1	1	1	1	1	1	1	1
	Test	0.2217	0.4956	0.5866	0.5963	0.6007	0.6051	0.6042	0.5901	0.5733	0.5689
Undersampled No Feature transform 36 x 36Pixels	Training	0.0891	0.4657	0.8113	0.9816	1	1	1	1	1	1
	Test	0.0602	0.4013	0.602	0.6037	0.5852	0.5819	0.5736	0.5736	0.5719	0.5619
Undersampled Data Squared Feature Transform 36 x 36Pixels	Training	0.0891	0.4075	0.9343	1	1	1	1	1	1	1
	Test	0.0602	0.2993	0.4515	0.4498	0.4314	0.4231	0.4247	0.4214	0.4415	0.4164

Undersampled Data Cubic Feature Transform 36 x 36 Pixels	Training	0.0891	0.41	0.9397	1	1	1	1	1	1	1
	Test	0.0602	0.3311	0.505	0.4933	0.4699	0.4632	0.4615	0.4699	0.4732	0.4582
Undersampled Data No Feature transform 50 x 50 Pixels	Training	0.1582	0.502	0.8862	0.9987	1	1	1	1	1	1
	Test	0.1187	0.4264	0.6054	0.5886	0.5819	0.5803	0.5719	0.5635	0.5786	0.5669
Undersampled Data Squared Feature Transform 50 x 50 Pixels	Training	0.0921	0.4611	0.99	1	1	1	1	1	1	1
	Test	0.0619	0.3462	0.5017	0.4933	0.4816	0.4799	0.4699	0.4732	0.4849	0.4548
Undersampled Data Cubic Feature Transform 50 x 50 Pixels	Training	0.0891	0.4607	0.99	1	1	1	1	1	1	1
	Test	0.0602	0.3311	0.4565	0.4532	0.4448	0.4448	0.4482	0.4482	0.4365	0.4147
Undersampled Data No Feature transform 128 x 128 Pixels	Training	0.3427	0.6389	0.9979	1	1	1	1	1	1	1
	Test	0.2809	0.505	0.5819	0.5953	0.5987	0.587	0.5953	0.5836	0.5652	0.5619
Undersampled Data Squared Feature Transform 128 x 128Pixels	Training	0.246	0.6477	1	1	1	1	1	1	1	1
	Test	0.2057	0.4565	0.5334	0.5401	0.5418	0.5401	0.5401	0.5485	0.5351	0.5117
Undersampled Data Cubic Feature Transform 128 x 128 Pixels	Training	0.2536	0.6335	1	1	1	1	1	1	1	1
	Test	0.2023	0.4231	0.4883	0.5	0.51	0.5067	0.5067	0.5033	0.4565	0.4398

Table 2: Results for Logistic Regression models vs inverse regularization strengths for L1 regularization

	C Values	1e-7	1e-5	0.001	0.01	0.1	1	10	100	1e+4	1e+6
Undersampled Data Squared Feature Transform 36 x 36Pixels	Training	0.0795	0.0795	0.0833	0.3686	0.8134	1	1	1	1	1
	Test	0.0987	0.0987	0.0836	0.3311	0.5652	0.5502	0.5502	0.5468	0.5301	0.5268
Undersampled Data Cubic Feature Transform	Training	0.0795	0.0795	0.0833	0.2958	0.9632	1	1	1	1	1

36 x 36 Pixels											
	Test	0.0987	0.0987	0.0836	0.2174	0.4231	0.4114	0.404	0.408	0.3361	0.3344
Undersampled Data Cubic Feature transform 50 x 50 Pixels	Training	0.0795	0.0795	0.0833	0.2778	0.9594	1	1	1	1	1
	Test	0.0987	0.0987	0.0836	0.2224	0.4097	0.3679	0.3696	0.3746	0.3478	0.3261
Undersampled Data No Feature Transform 50 x 50 Pixels	Training	0.0795	0.0795	0.0833	0.2958	0.9632	1	1	1	1	1
	Test	0.0987	0.0987	0.0836	0.2174	0.4231	0.4114	0.4047	0.4047	0.3428	0.3495
Undersampled Data Squared Feature Transform 50 x 50 Pixels	Training	0.0891	0.4607	0.99	1	1	1	1	1	1	1
	Test	0.0602	0.3311	0.4565	0.4532	0.4448	0.4448	0.4482	0.4482	0.4365	0.4147
Undersampled Data No Feature transform 128 x 128 Pixels	Training	0.0795	0.0795	0.0833	0.3967	0.9318	1	1	1	1	1
	Test	0.0987	0.0987	0.0836	0.3445	0.5619	0.5435	0.5385	0.5702	0.5201	0.5084
Undersampled Data Squared Feature Transform 128 x 128Pixels	Training	0.0795	0.0795	0.0833	0.3686	0.9983	1	1	1	1	1
	Test	0.0987	0.0987	0.0836	0.2776	0.4766	0.4833	0.4632	0.4933	0.4866	0.4933
Undersampled Data Cubic Feature Transform 128 x 128 Pixels	Training	0.0795	0.0795	0.833	0.3603	0.9987	1	1	1	1	1
	Test	0.0987	0.0987	0.0836	0.2575	0.4164	0.4247	0.4348	0.4582	0.4615	0.4231

Table 3: Results for Neural Network models vs regularization strengths for L2 regularization

US stands for training on the undersampled data, S stands for training on all the data and the hidden layers are given in the form of 200, 100, 50, 30

	Alpha Values	1e-7	1e-5	0.001	0.01	0.1	1	10	100	1e+4	1e+6
US RELU, 100	Training	1	1	1	1	1	0.9975	0.9678	0.5238	0.0891	0.0891
	Test	0.6773	0.6773	0.6756	0.6823	0.6833	0.6839	0.6739	0.4666	0.0602	0.0602
US RELU 100, 50	Training	1	1	1	1	1	0.9933	0.9573	0.1854	0.0891	0.0891

	Test	0.6923	0.7007	0.7124	0.689	0.6706	0.6722	0.7124	0.1873	0.0602	0.0602
US RELU, 100, 50, 30	Training	1	1	1	1	0.9908	0.9921	0.9218	0.0891	0.0891	0.0891
	Test	0.6706	0.6739	0.6605	0.6639	0.6589	0.694	0.6605	0.0602	0.0602	0.0602
US RELU, 200, 100, 50, 30	Training	1	1	1	1	0.9674	1	0.9247	0.0891	0.0891	0.0891
	Test	0.6789	0.6605	0.6756	0.6756	0.6806	0.7241	0.6739	0.0602	0.0602	0.0602
US tanh, 100	Training	1	1	1	1	0.9941	0.9151	0.9276	0.531	0.0891	0.0891
	Test	0.6288	0.6288	0.6304	0.6321	0.6421	0.5936	0.6472	0.5	0.0602	0.0602
US tanH 100, 50	Training	1	1	1	1	0.9845	0.9992	0.9121	0.1808	0.0891	0.0891
	Test	0.6839	0.6839	0.6839	0.6823	0.6656	0.7057	0.6773	0.1656	0.0602	0.0602
US tanh 100, 50, 30	Training	1	1	1	1	0.9628	0.9975	0.9	0.0891	0.0891	0.0891
	Test	0.6371	0.6371	0.6388	0.6338	0.592	0.6488	0.6254	0.0602	0.0602	0.0602
US tanh 200, 100, 50, 30	Training	1	1	1	1	0.9335	1	0.9021	0.0891	0.0891	0.0891
	Test	0.6589	0.6605	0.6672	0.6572	0.6137	0.689	0.6104	0.0602	0.0602	0.0602
US Logistic 100	Training	1	1	1	1	1	0.9933	0.6715	0.187	0.0891	0.0891
	Test	0.6538	0.6538	0.6538	0.6572	0.6421	0.6522	0.5468	0.199	0.0602	0.0602
US Logistic 100, 50	Training	1	1	1	1	1	0.9565	0.1887	0.0891	0.0891	0.0891
	Test	0.6321	0.6404	0.6321	0.6505	0.6739	0.6355	0.1739	0.0602	0.0602	0.0602
US Logistic 100,50,30	Training	0.9987	0.9987	0.9992	1	0.9992	0.9197	0.0891	0.0891	0.0891	0.0891
	Test	0.6003	0.5903	0.5886	0.5936	0.607	0.5017	0.0602	0.0602	0.0602	0.0602
US Logistic 200,100,50,30	Training	0.9854	0.9895	0.9996	0.9996	0.9895	0.0891	0.0891	0.0891	0.0891	0.0891
	Test	0.5602	0.5569	0.5452	0.5635	0.5418	0.0602	0.0602	0.0602	0.0602	0.0602
S Relu, 100	Training	1	1	1	1	0.9997	0.9989	0.9019	0.5123	0.1436	0.1436

	Test	0.7747	0.7739	0.7818	0.7801	0.7686	0.7774	0.7553	0.5177	0.1492	0.1492
S RELU 100,50	Training	1	1	1	1	0.9996	..9523	0.8359	0.3486	0.1436	0.1436
	Test	0.7756	0.773	0.765	0.7765	0.7818	0.742	0.7208	0.3445	0.1493	0.1493
S Relu 100,50,30	Training	1	1	1	1	0.996	0.9901	0.8493	0.2361	0.1436	0.1436
	Test	0.7818	0.7765	0.7774	0.7783	0.78	0.7827	0.7394	0.2385	0.1493	0.1493
S RELU 200,100,50,30	Training	1	0.9852	1	0.9929	0.989	0.9962	0.8571	0.1436	0.1436	0.1436
	Test	0.7862	0.7624	0.8056	0.7739	0.7792	0.7836	0.7509	0.1493	0.1493	0.1493
S tanh 100	Training	1	1	1	1	0.9965	0.9943	0.8383	0.5083	0.1436	0.1436
	Test	0.7412	0.7412	0.742	0.7438	0.7367	0.7482	0.7261	0.5027	0.1493	0.1493
S tanh 100, 50	Training	1	1	1	1	0.9768	0.9996	0.7806	0.2439	0.1436	0.1436
	Test	0.7588	0.7597	0.7562	0.765	0.7279	0.7756	0.7049	0.2473	0.1493	0.1493
S tanh 100,50,30	Training	1	1	1	1	0.9494	0.9996	0.7422	0.1436	0.1436	0.1436
	Test	0.7606	0.7624	0.7606	0.7792	0.7111	0.773	0.6758	0.1493	0.1493	0.1493
S tanh 200,100,50,30	Training	1	0.9951	0.9971	0.979	0.9989	0.9978	0.7301	0.1436	0.1436	0.1436
	Test	0.7597	0.7535	0.7323	0.7403	0.7491	0.7641	0.6422	0.1493	0.1493	0.1493
S logistic 100	Training	1	1	1	0.9993	0.9996	0.9779	0.6318	0.235	0.1436	0.1436
	Test	0.742	0.742	0.7438	0.7571	0.7527	0.7641	0.6237	0.2385	0.1493	0.1493
S Logistic 100, 50	Training	0.9998	0.9998	0.9998	0.9987	0.9998	0.9298	0.2547	0.1436	0.1436	0.1436
	Test	0.75	0.7518	0.7535	0.7712	0.7659	0.7465	0.2544	0.1493	0.1493	0.1493
S Logistic 100, 50, 30	Training	0.9996	0.9954	0.9881	0.9965	0.9984	0.8878	0.1436	0.1436	0.1436	0.1436
	Test	0.682	0.689	0.682	0.6935	0.7102	0.6617	0.1493	0.1493	0.1493	0.1493
S Logistic 200, 100 50, 30	Training	0.9852	0.9792	0.9823	0.9943	0.9863	0.1436	0.1436	0.1436	0.1436	0.1436
	Test	0.6855	0.7067	0.6864	0.6846	0.6926	0.1493	0.1493	0.1493	0.1493	0.1493

Table 4: Results for SVM vs L2 inverse regularization strengths

	Alpha Values	1e-7	1e-5	0.001	0.01	0.1	1	10	100	1e+4	1e+6
Linear	Training	0.1475	0.4803	0.8811	0.9951	1	1	1	1	1	1
	Test	0.1345	0.4496	0.7009	0.7	0.7044	0.7044	0.7044	0.7044	0.7044	0.7044
RBF	Training	0.1475	0.1475	0.1475	0.2533	0.5465	0.9074	0.9965	1	1	1
	Test	0.1345	0.1345	0.1345	0.2381	0.4991	0.7540	0.8097	0.8097	0.8150	0.8150
Poly	Training	0.1475	0.1475	0.1475	0.1814	0.5053	0.8924	0.9849	1	1	1
	Test	0.1345	0.1345	0.1345	0.1646	0.4274	0.6815	0.7513	0.7556	0.7556	0.7556

Conclusion

The data presented interesting features within it and the models being trained presented interesting conclusions as well. For the logistic regression model, it was interesting to see how the type of data trained led to different results.

Without undersampling the data accuracy presented itself to be much higher than with undersampling. However, this was quickly disproved as accurate when looking at the confusion matrix which hinted at the accuracy being overinflated by the amount of correctly classified results from the species present in larger amounts. Since the data set was unbalanced it was more likely that the model would train on more of the species present in higher amounts and test on those same species as well. As such, when the testing phase came much of the lower amount samples were exacerbated and the accuracy was instead propped up by the samples present in larger amounts which were seen by the confusion matrix when many correctly classified examples were also the same examples that had more samples in them. After undersampling the results had gone down but not too much showing how undersampling can lead to a more accurate

outcome in terms of training on different data but lead to a lower accuracy overall which may be resolved by having more training examples in the set.

Another interesting thing to note was that the data did not come in all the same aspect ratios. All the data was presented differently and as such had to be resized so that all the images had the same amount of features and so that the model could be trained properly. Different pixel values lead to different results however not too varied in their execution. In this model, especially the changes in pixel values for no feature transforms seemed to be positively correlated with a larger accuracy regardless of the regularization value used. However, this trend did not seem to carry over to any other feature transformations. It was expected that the model would overfit on the 128 x 128 images as they have many more features compared to the other image dimensions however it was found that the 128 x 128 images actually resulted in the highest accuracy. This can possibly be attributed to the model being able to actually learn the data better and not underfit the data which may have been the case with the lower-resolution images.

In terms of regularization when working with the logistic regression model L2 regularization seemed to provide the best result across the models. L1 regularization consistently leads to lower accuracies than L2 regularization regardless of which feature transforms or models were used, which shows the nature of the differences between L1 regularization and L2 regularization. Mainly that one takes weights down to 0 (L1) whereas the other does not. This can mean that some features which were necessary for the execution of the correct prediction in the model actually were canceled out by the L1 regularization and kept by the L2 regularization which led to the higher accuracy of the L2 regularization. Furthermore, the graph of the L1

regularization dropped off significantly quicker than did the graph of the L2 regularization which went down in accuracy steadier.

Regarding the feature transformations for the logistic regression model consistently regardless of regularization or regularization value squared and cubic transformations performed noticeably worse than no feature transforms. This hints at the model overfitting to the data when using the squared and cubic feature transformation which was instead mitigated by not feature transforming at all which led to less overfitting compared to the other two feature transforms. Furthermore the precision and recall of this model suggest that the model had an about average rate of identifying true positives when compared to the other best models such as the neural networks.

Neural networks lead to interesting conclusions as well. The different activation functions and hidden layer combinations lead to different accuracy results. Similar to the logistic regression model, training with the neural networks on the data that was unchanged led to a higher accuracy than what was actually the case. The data presented itself as having inaccuracies as it had too many of one sample and not enough of another and as such the data resulted in a very high accuracy of 83.3% for the same reasons as the logistic regression model compared to the undersampled data having a more reasonable 72.43% accuracy when testing on the undersampled data. With regards to regularization, only L2 regularization was used for neural networks and only 36 x 36-pixel resolution images were used since they lead to the least computational time and as was seen in logistic regression they didn't lead to drastic differences in accuracy across different pixel dimensions. Instead, the differences in training were the results of different hidden layers being used and different activation functions. The RELU activation function provided the best results across regularisations and others although it wasn't drastically

different from the tanh activation function. The logistic activation function provided a noticeably worse result when compared to the other two activation functions when within the same amount of hidden layers.

Regarding the regularization, it was interesting to see how as the regularization strength increased past a certain threshold the accuracy dropped down much quicker than expected and it was a trend that continued throughout all the models for the same regularization value between 100 and 10,000 depending on the model. It was interesting that those models with fewer hidden layers were not as affected by a 100 regularization strength compared to those models with more hidden layers highlighting how the models with more hidden layers are more prone to overfitting and as such the regularization strength has a stronger effect on them depending. The neural networks best model furthermore also had a high precision hinting at the models ability to classify the positive points correctly and also had a similarly high recall.

Although the SVM model was not trained on other feature transforms and pixel values due to the insights gained from training on the previous two models, mainly that the data did not respond well to the higher feature transforms and that the data did not have significant differences when using 36 x 36, 50x50, or 128 x 128 pixel values, the model was still trained on the different kernel functions in relation to the best combination of the previous three transforms and the results were interesting to note. The RBF kernel with less regularization.

As the regularization strengths decreased for almost all models the testing accuracy increased, indicative of the models not being affected by regularization as much as one may believe. Furthermore the linear kernel seemed to stabilize with its accuracies at an inverse regularization value of 0.001 whereas the RBF kernel took a weaker regularization strength with an inverse regularization strength of 10 to start its stable accuracy period. Furthermore the

weaker regularization values indicate that the model is not necessarily overfitting and that the data is not necessarily noisy as is the case as the seedlings were placed on a black background.

As the RBF kernel perfmored better than the linear and polynomial kernel by a substantial margin it gives the idea that the underlying data is not necessarily linearly separable and that the data is intertwined with each other as should be the case with the image classification problem.

These results could be improved by adding more data and training different types of models. The neural network models only contained training on data of 36 x 36 pixels and say if 128 x 128 pixels were used it may have been possible that a stronger accuracy result may have been achieved. If the data was given as squares to begin with it would have been easier to not have to resize the data, and instead of stretching the plants would remain as they are and the black pixels would fill the sides. This may have resulted in larger accuracy as the plants would be less varied when trained in the model. Furthermore, the SVM model could have been tested using different feature transforms however it was opted not too due to the effects they had from the previous two models, but trying these different feature transforms as well as trying the SVM function on different pixel values could have lead to a higher accuracy and a deeper intuition about the underlying data and the patterns present within it. Overall it's possible adding more hidden layers and having more features from a greater pixel resolution would lead to a higher accuracy from neural networks as that lead to the highest accuracy overall throughout the models.

Works Cited

https://vision.eng.au.dk/plant-seedlings-dataset/

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.ConfusionMatrixDisplay.html#sklearn.metrics.ConfusionMatrixDisplay

https://python-pillow.org/

https://matplotlib.org/stable/index.html

Sample report and slides MeiNa Xie, Winnie Zheng