CS-UY 4563: Machine Learning

Final Project Written Report

*Image Classification of 12 different seedings*

2:00 PM section

12/07/2022

Professor Linda N. Sellie

Jin Zhou

**Introduction**

For my project, the dataset used is plant seedling images provided by ‘<https://vision.eng.au.dk/plant-seedlings-dataset/>’. This project aims to find optimal models to correctly recognize the plant among the 12 types provided based on a seedling image using different methods.

The method utilized in this project are Logistic Regression, Simple Vector Machine(SVM), and Neural Network. Each approach will be run multiple times using different regularization techniques and feature transformations.

*(Two randomly selected original images)*

**Preprocessing**

The dataset used in this project is giving in separate folders each containing images of different plants. The images were first loaded as numerical features using Image function from the PIL library in combination with the numpy library with each image flattened into 49152 features (128\*128 pixels) after resizing them, and the label for each image in loaded into another array simultaneously in non-one hot fashion where each number from 0 to 11 represent one type of plant. Then, StandardScaler from the sklearn library is trained with the full feature matrix for future normalization.

*(Resized Imge)*



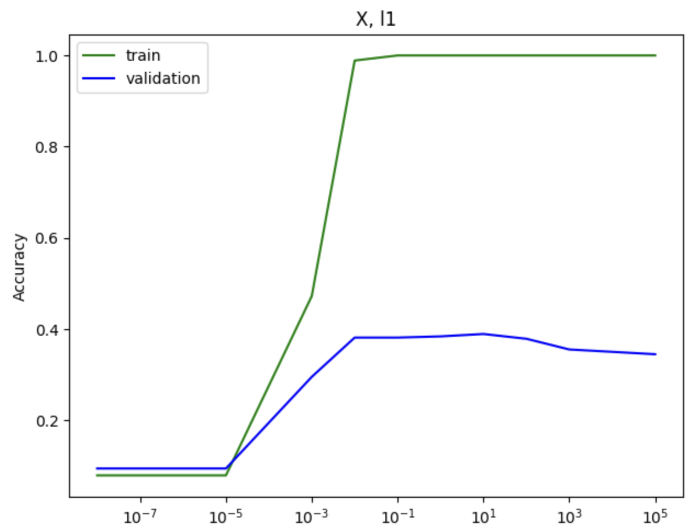
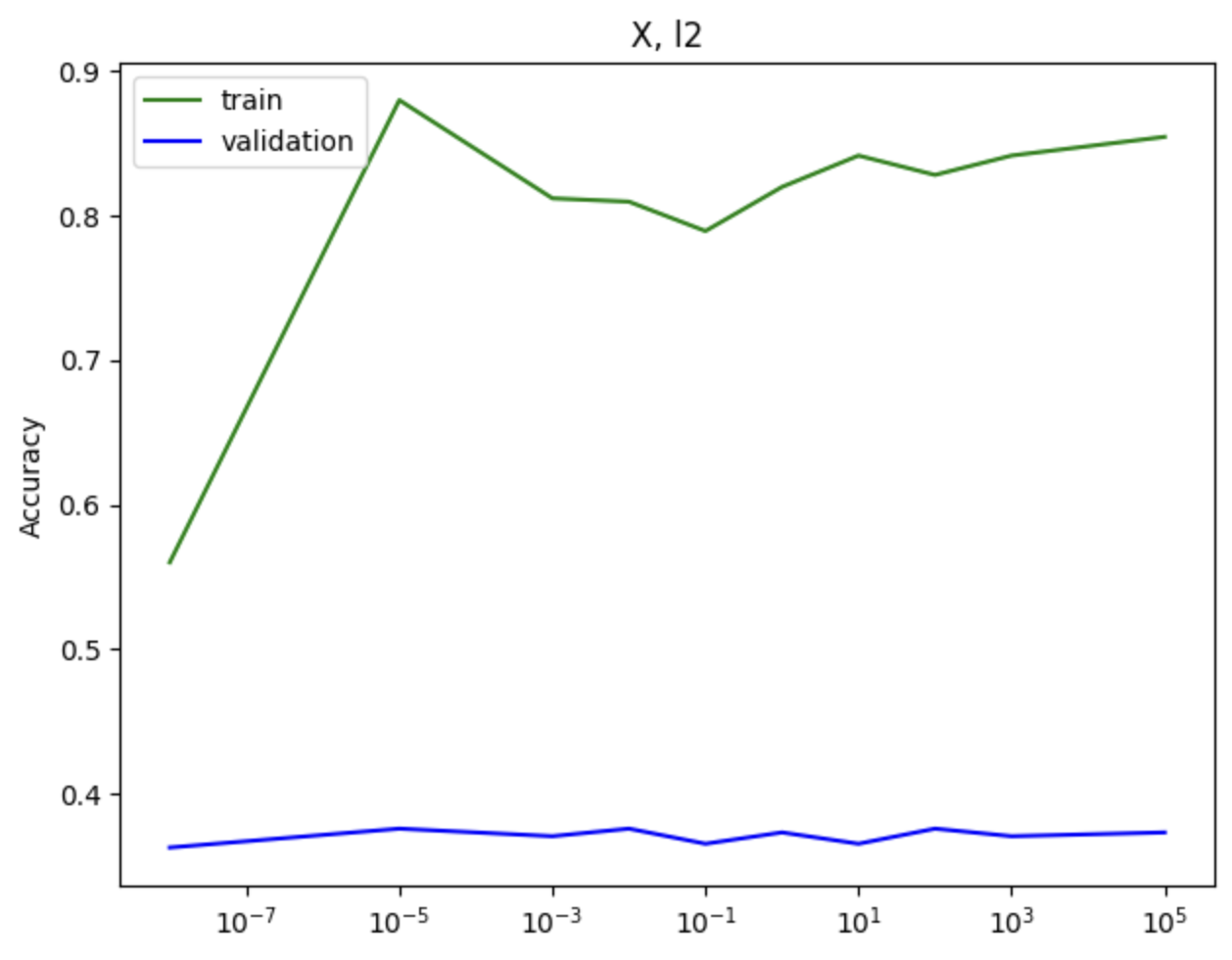
Lastly, the dataset is randomly separated into train, validation, and testing set each make up 72.25%, 12.75%, 15% of the entire dataset respectively using the train\_test\_split function from sklearn. The testing set here is only used to analyze the effect of the best model from each method at the end.

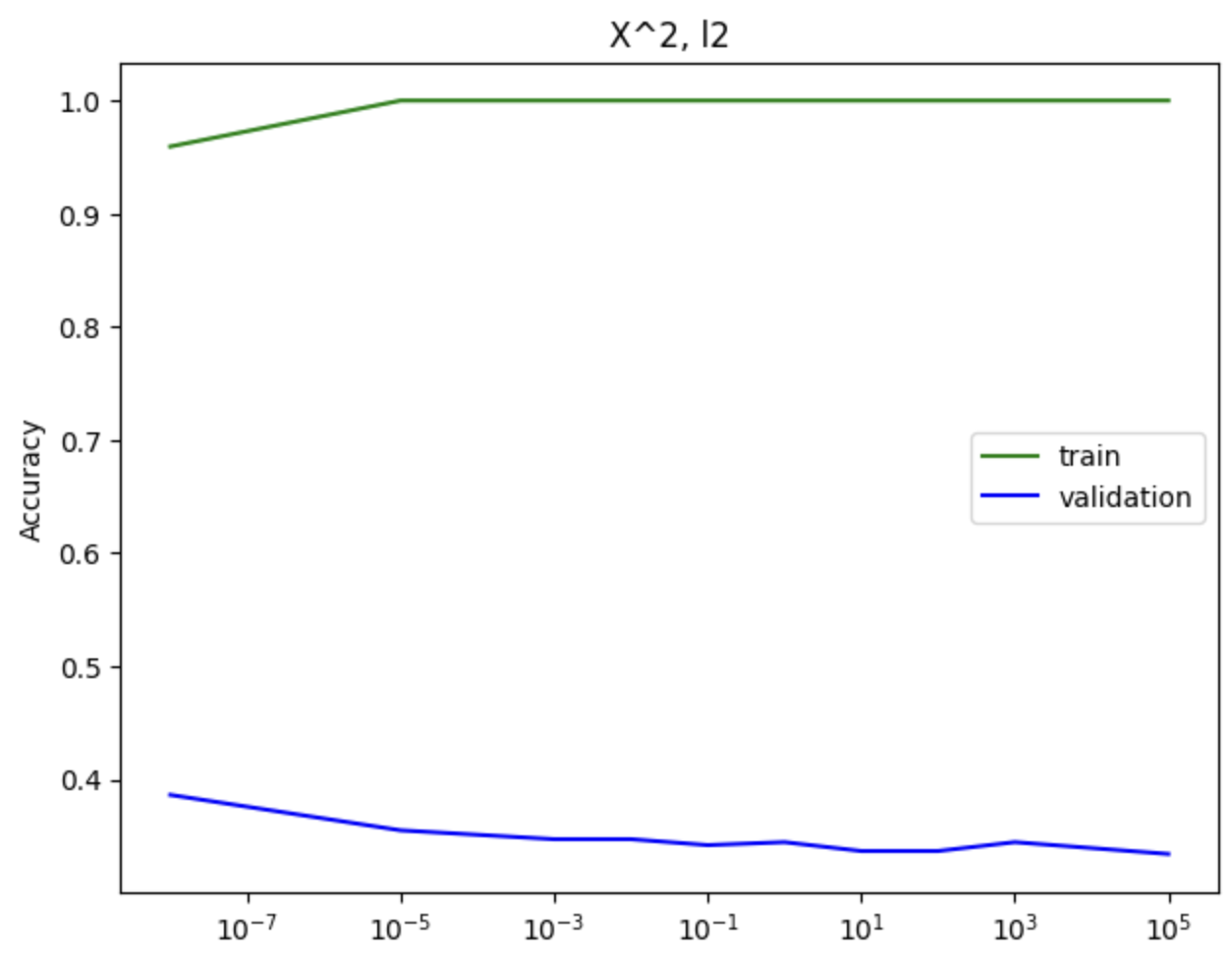
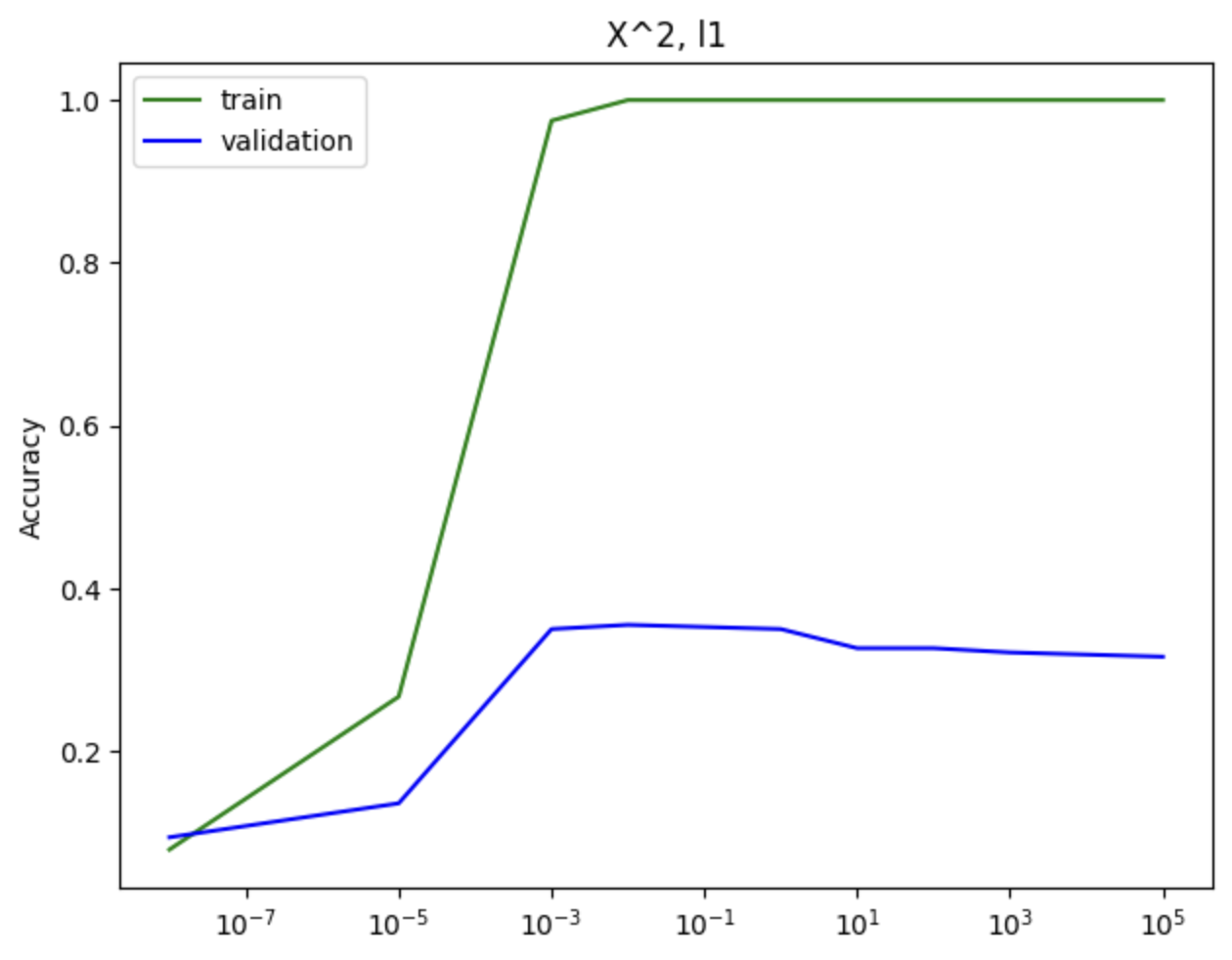
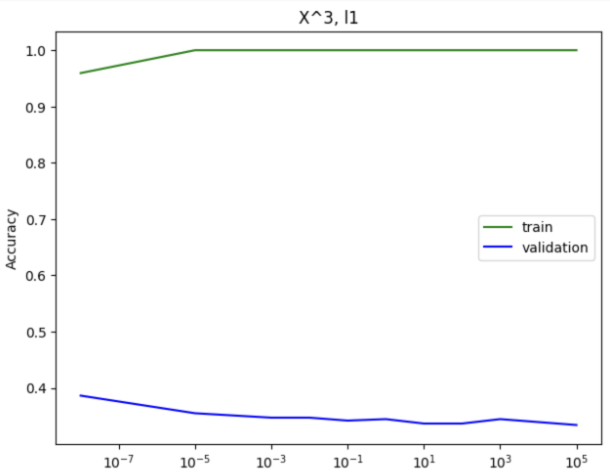
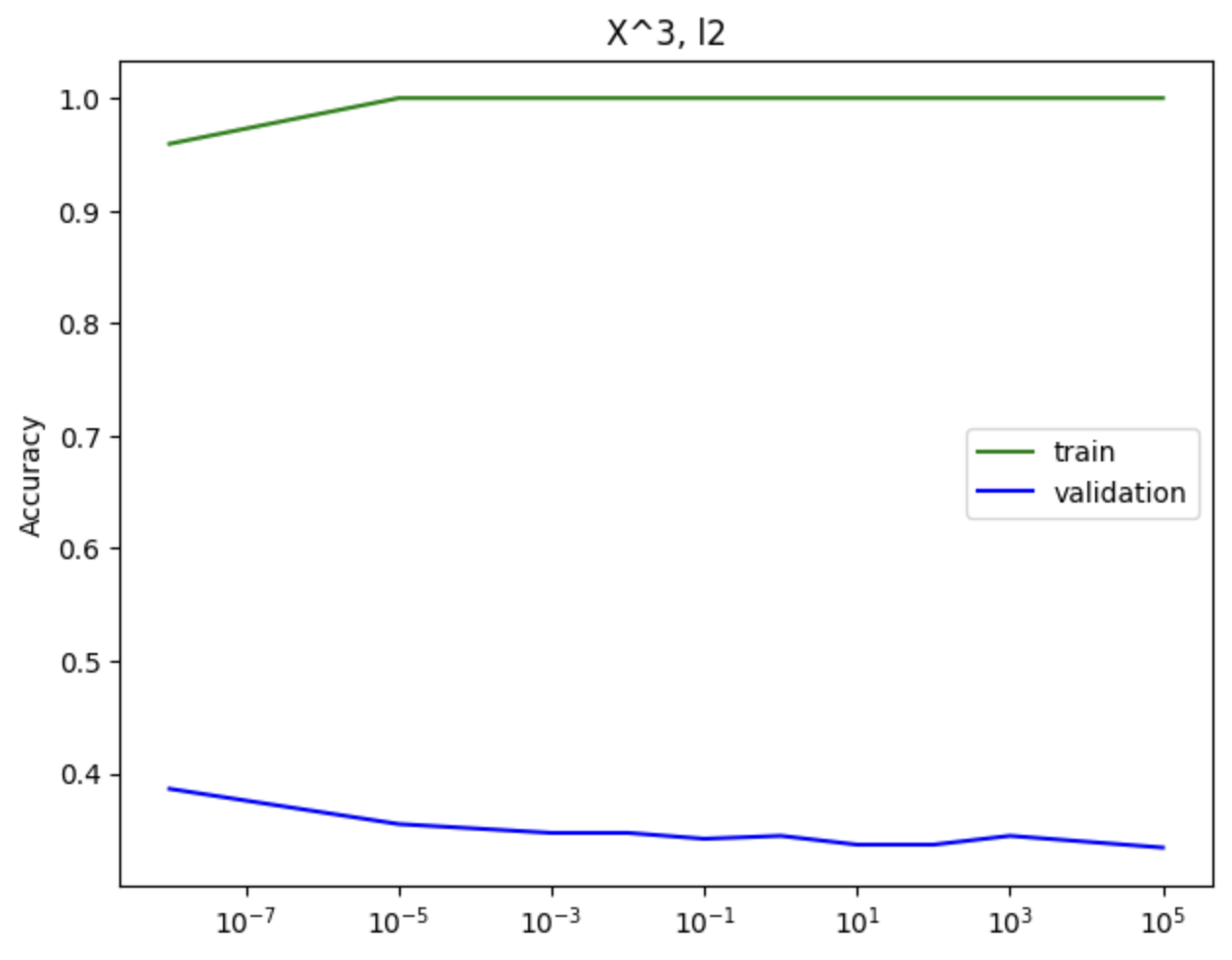
Before the data were loaded, I noticed that different amount of images are provided for each plant where a portion of them have twice as much of samples compared to others, which will potentially make model predictions distorted. Thus, the sample size for each plant is limited to 250 which results in 3000 total samples.

**1) Logistic regression**

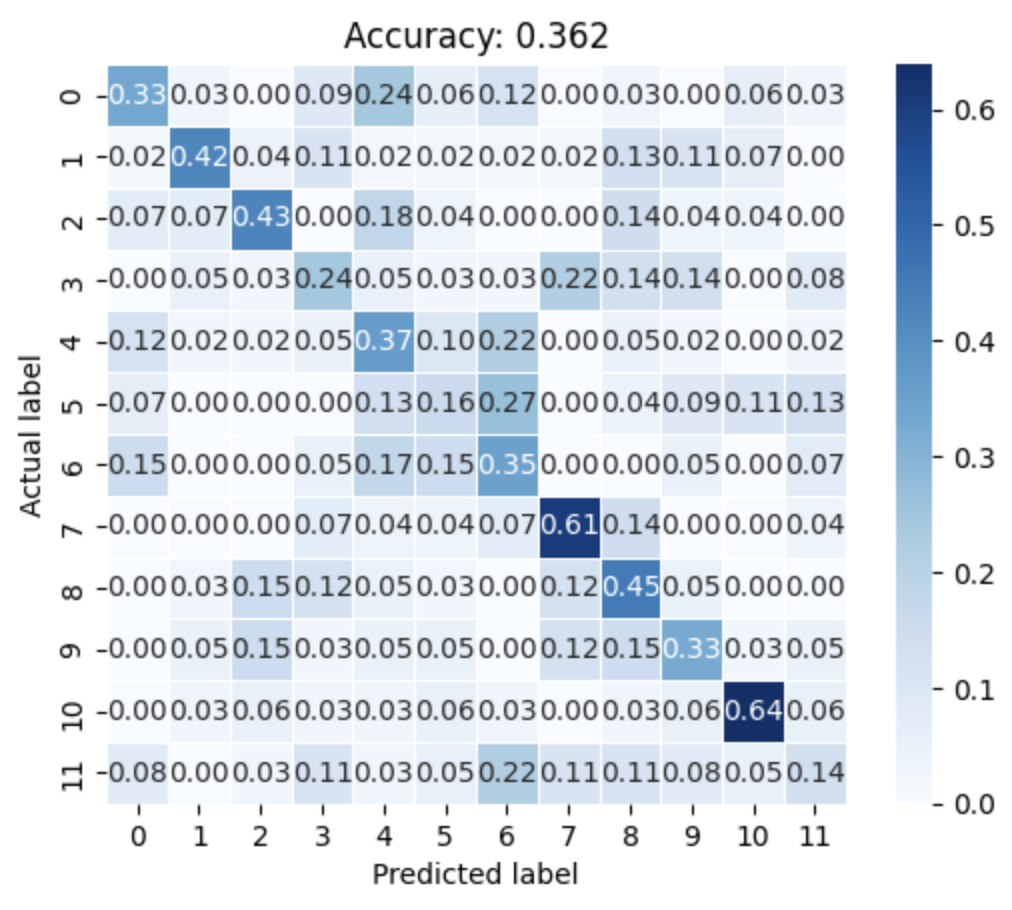
The first model I analyzed is logistic regression. For the training process I used LogisticRegression function from the sklearn library. Initially, the model is trained on the original dataset without regularizations , which resulted in training accuracy of 100% and validation accuracy of 40.1%.

Then the model is trained a max of 200 times on normal, squared, and cubed features with L1(LASSO) and L2(Ridge) regularizations, using C, inverse of the regularizing strength hyperparameter, of [0.00000001, 0.00001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 100000]. Lastly, the training and validation accuracy are plotted corresponding to their C with logarithmic x-axis for the C values. The plots are shown below.





Next, the model that produced highest validation accuracy with reasonably high training accuracy is selected to graph the following confusion matrix using the test data.

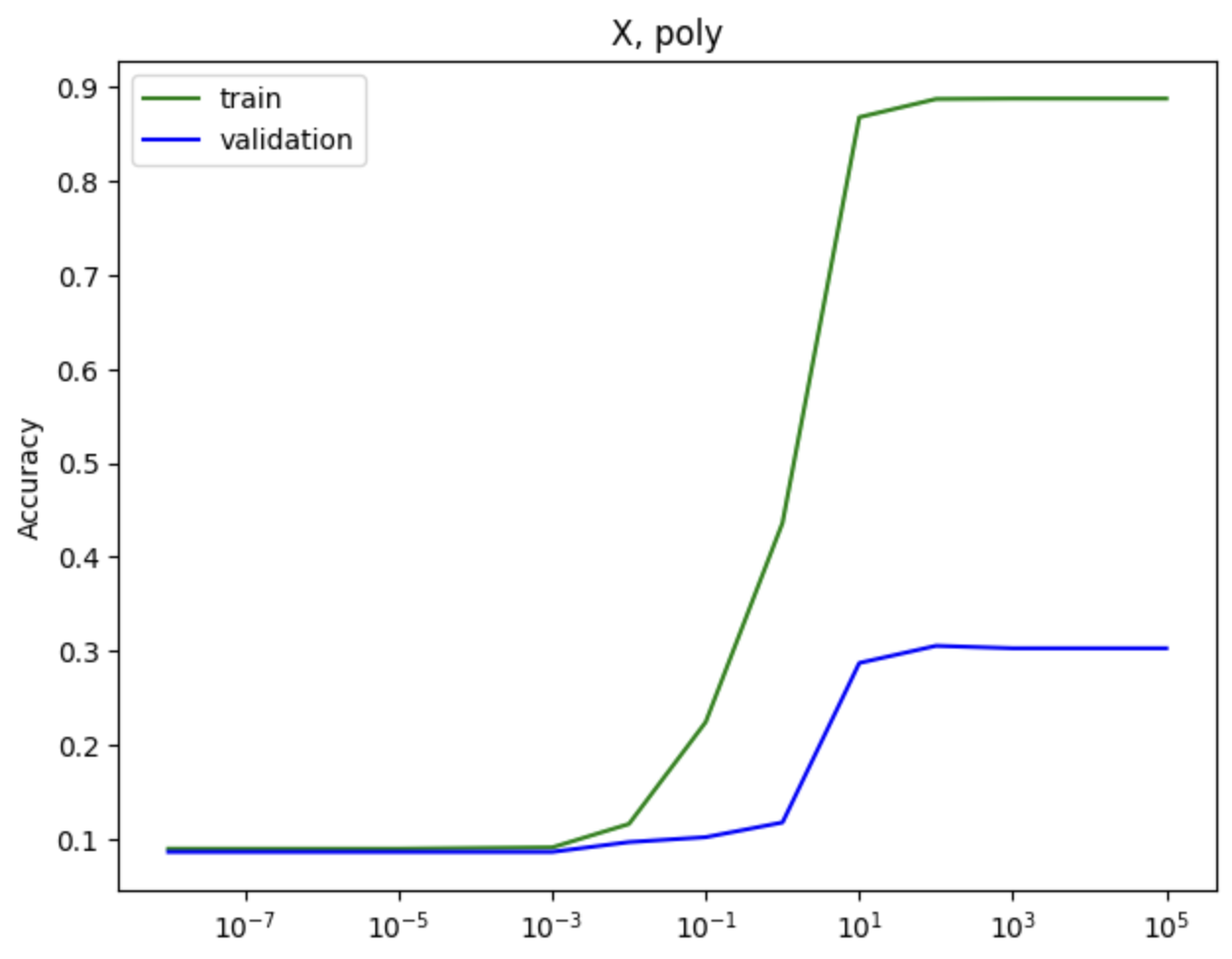
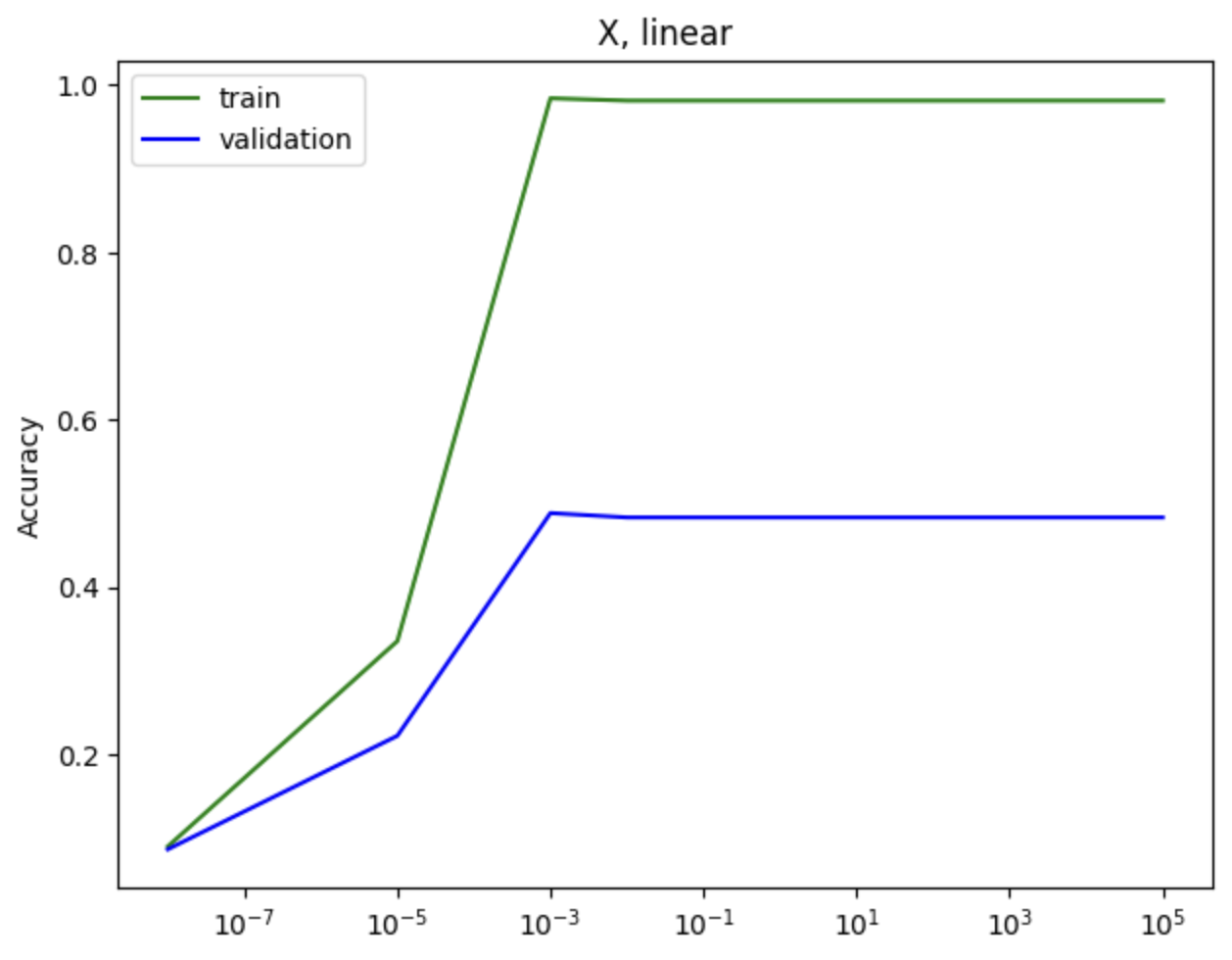
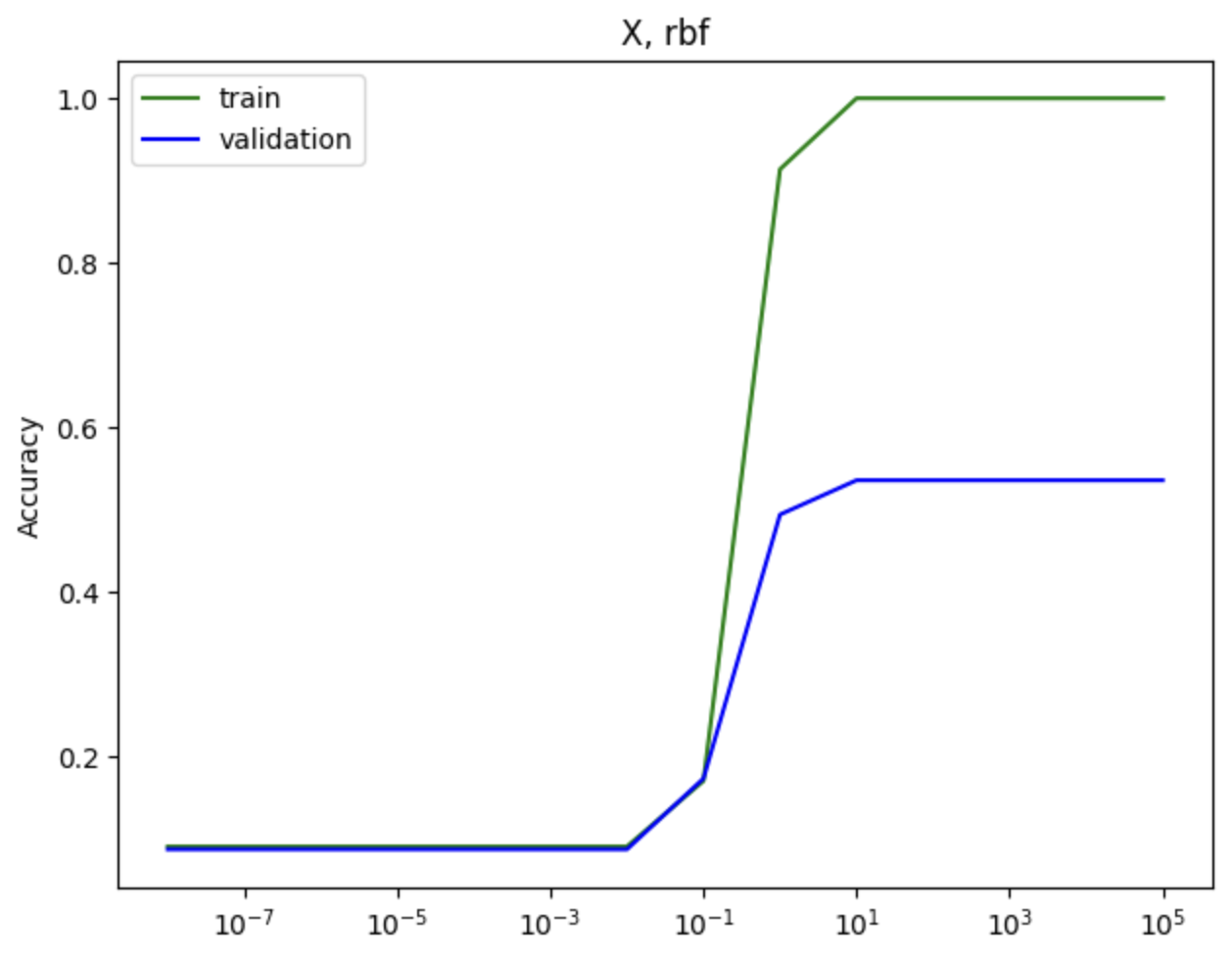
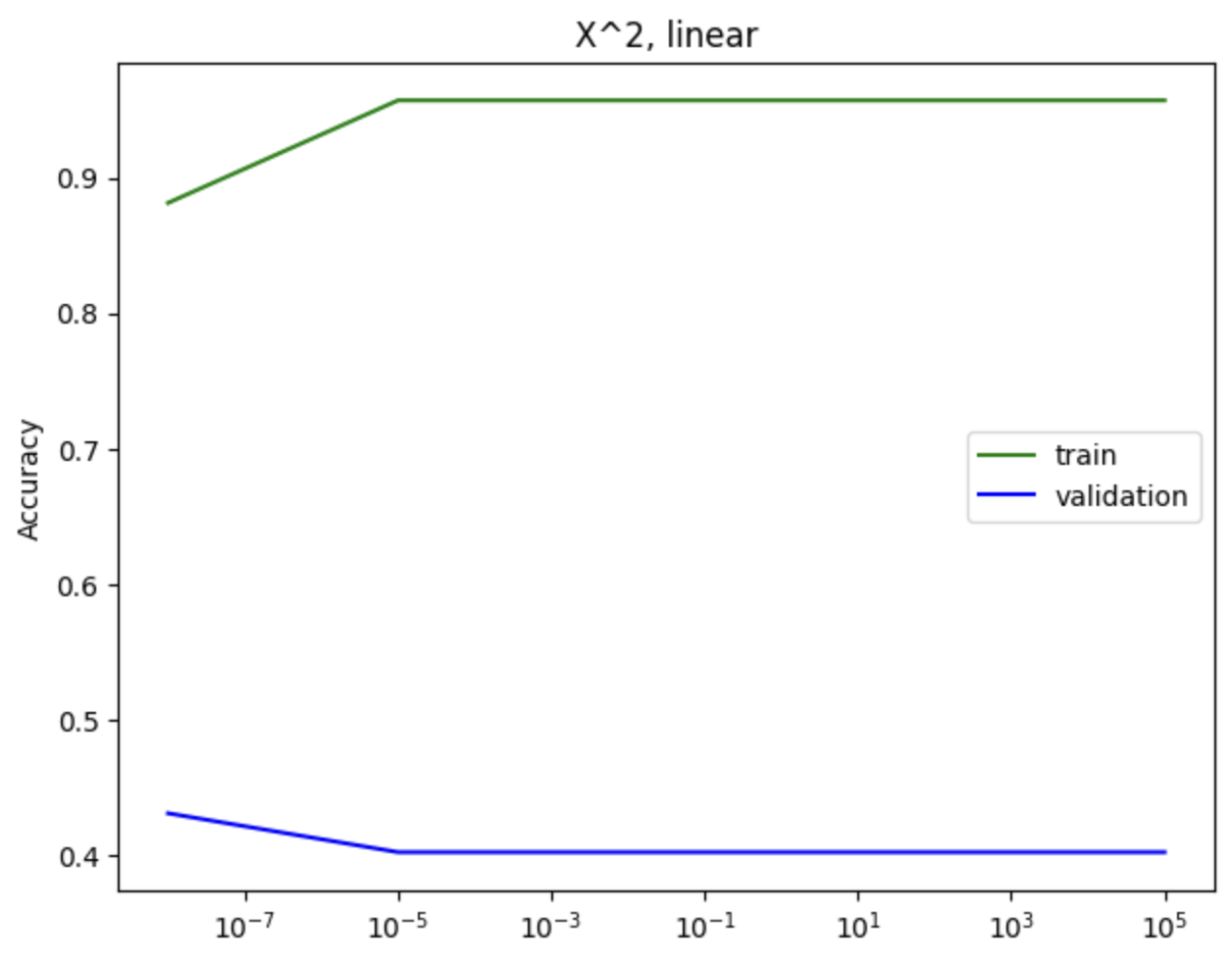
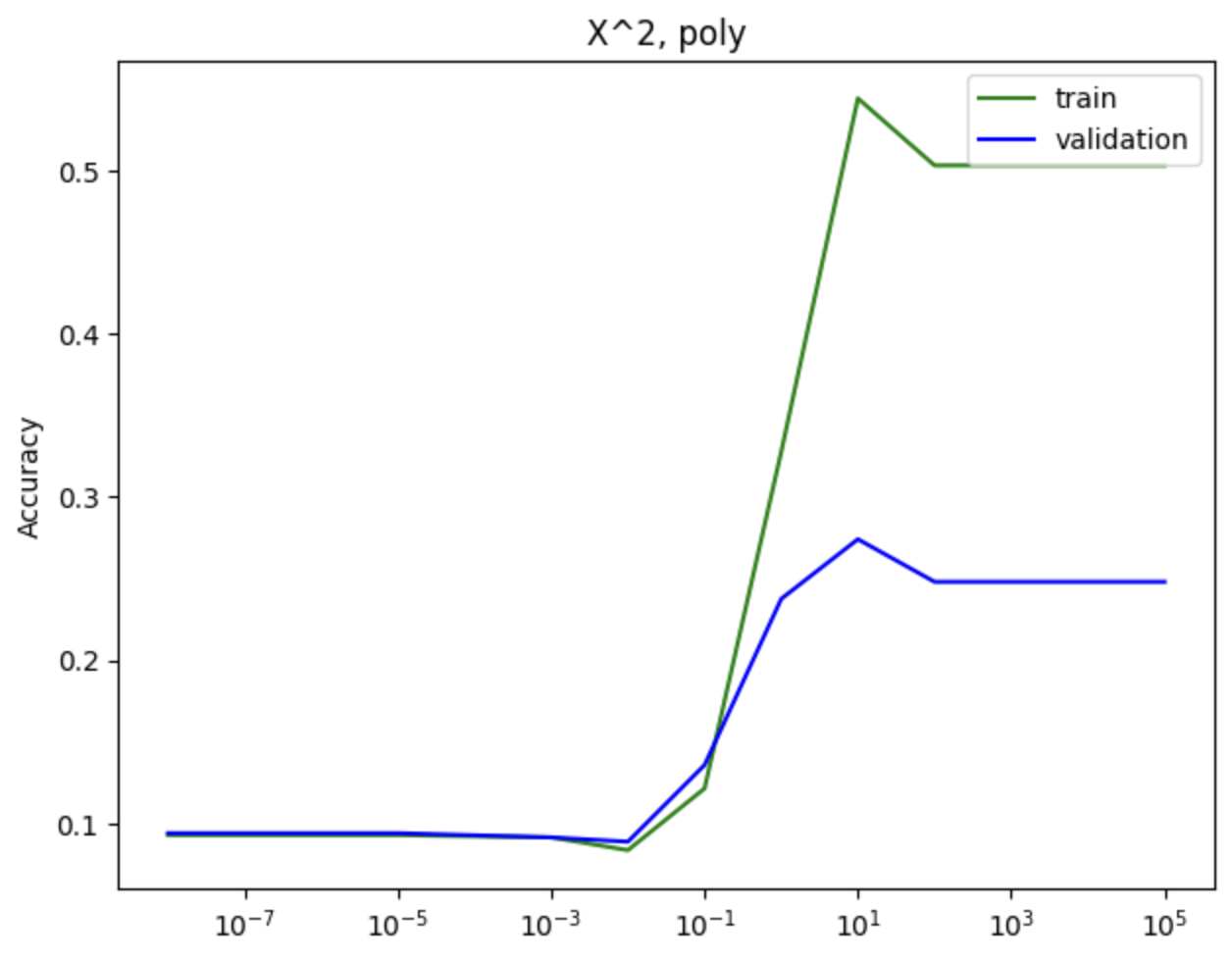
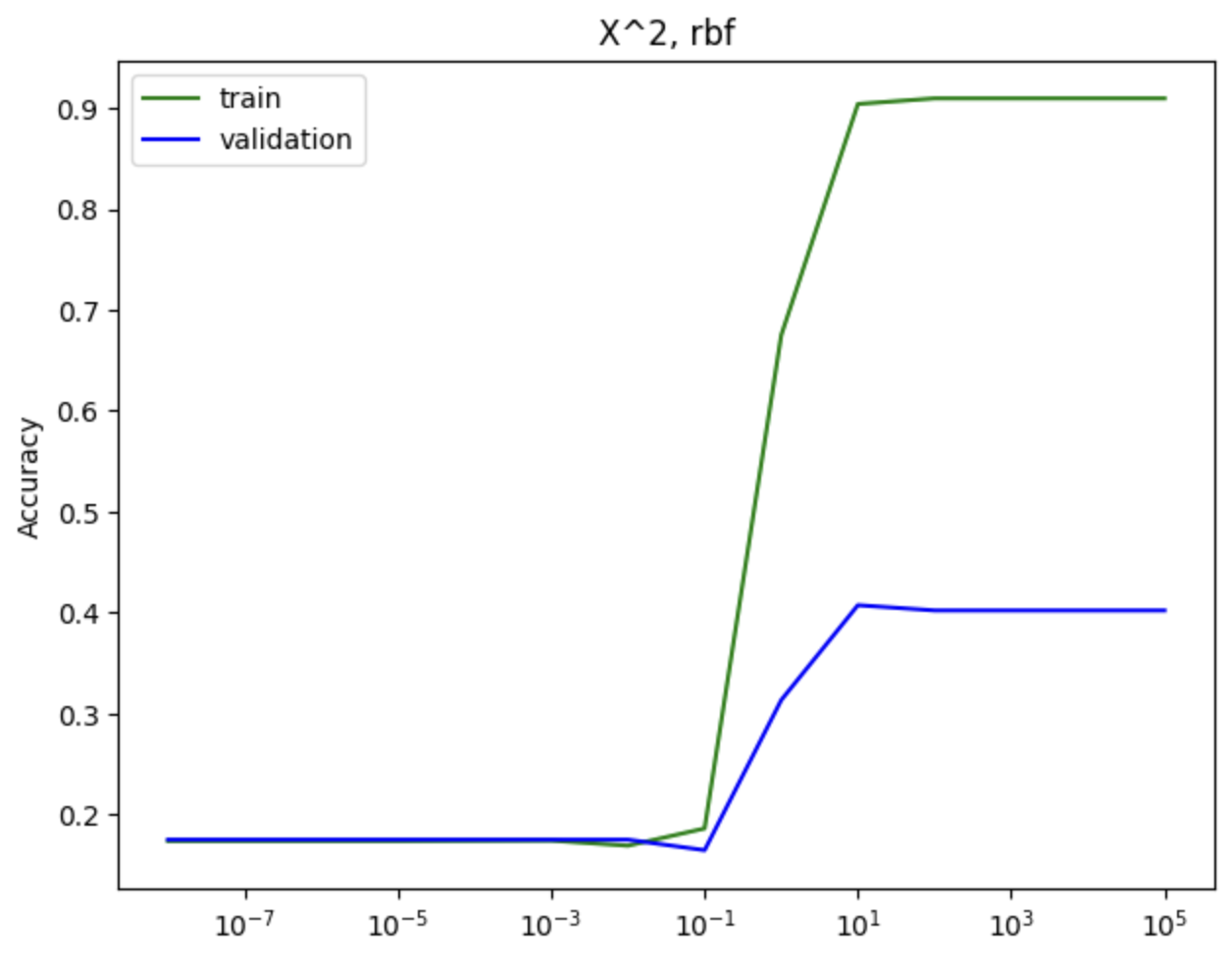
*(X, LASSO, C=10)*

**2) Support Vector Machine(SVM)**

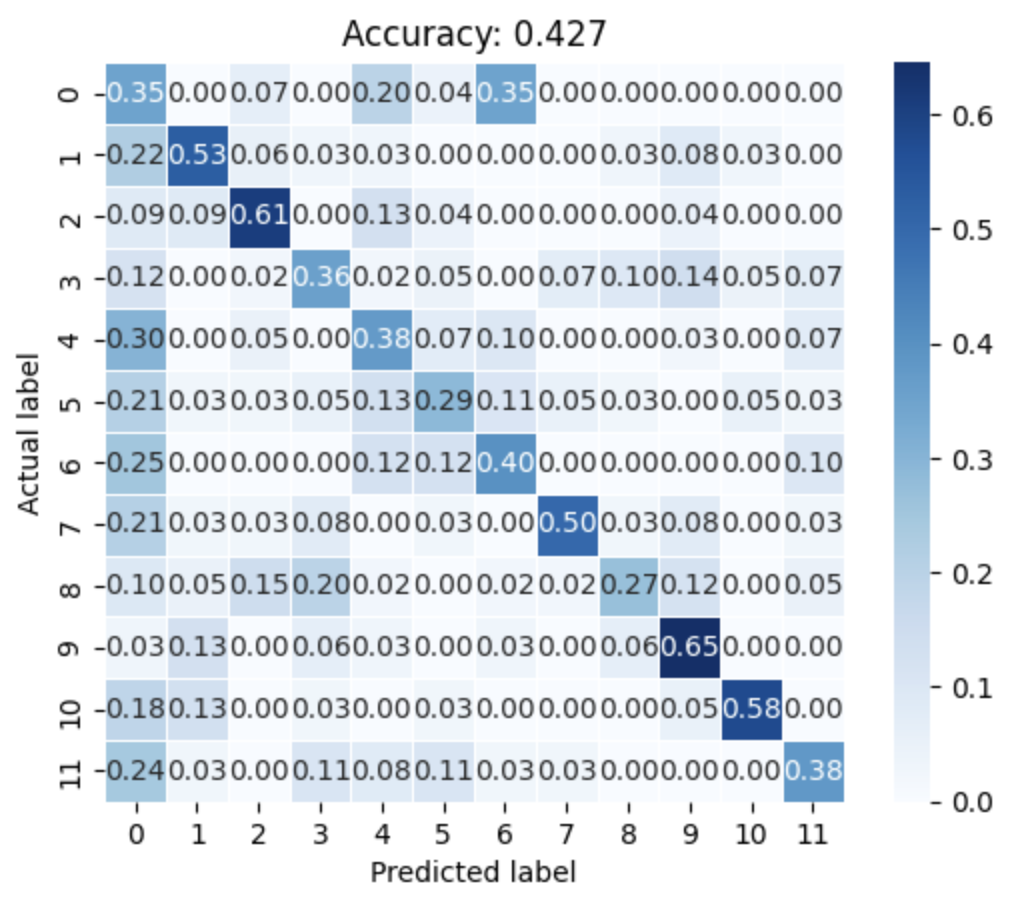
Next up, I decided to test out the SVM model. For model training, the SVC function form the sklearn library is used. For faster training, the feature space from now on for training is 12288(64\*64 pixels).

*(Image resized to 64\*64 pixels)*

For SVM, three kinds of kernel functions: linear, polynomial, and radial-basis function(rbf) are attempted. For each kernel function, a model is trained on original and squared features with the same set of hyperparameter C as mentioned before, using L22 regularization. Below are the results received:

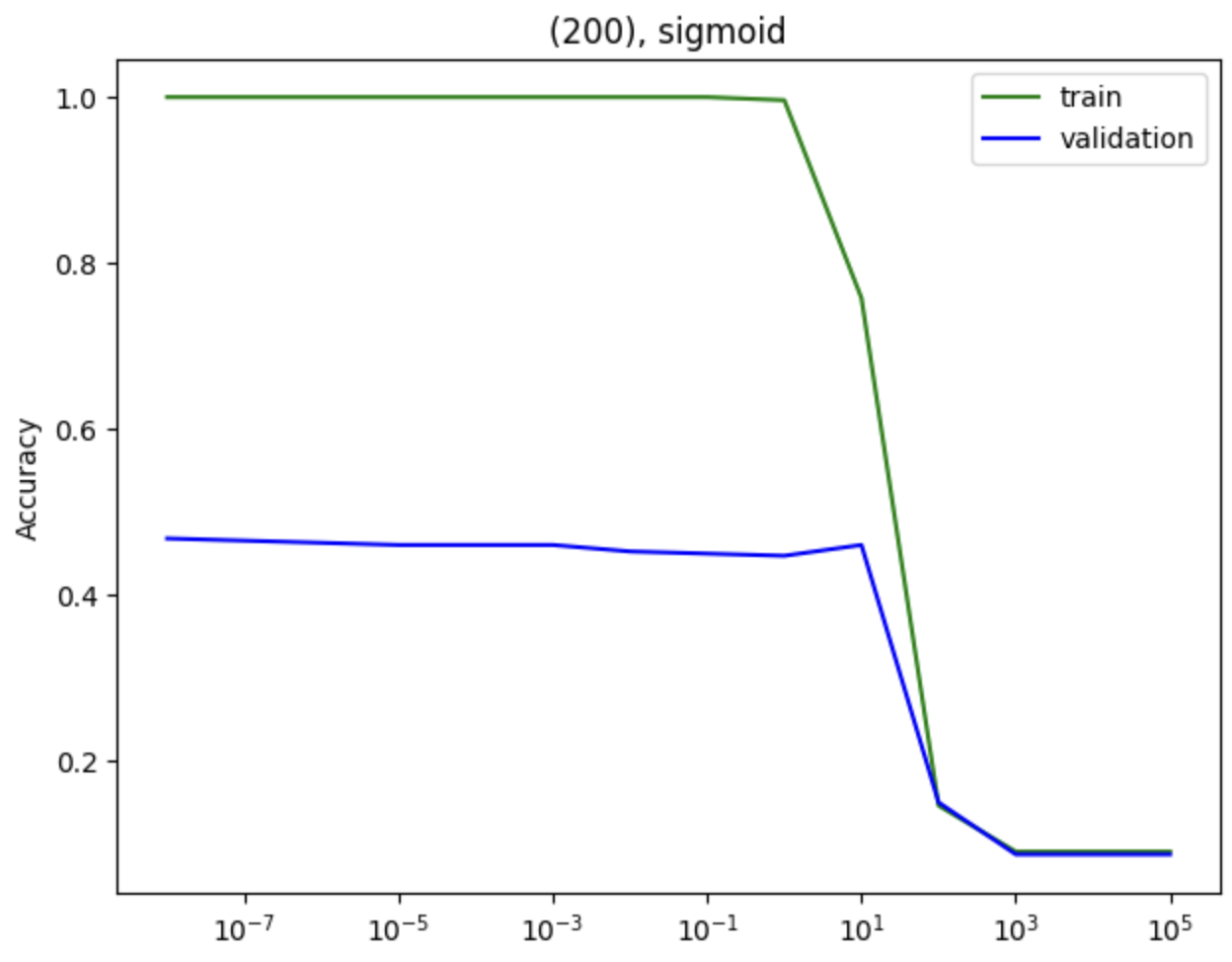


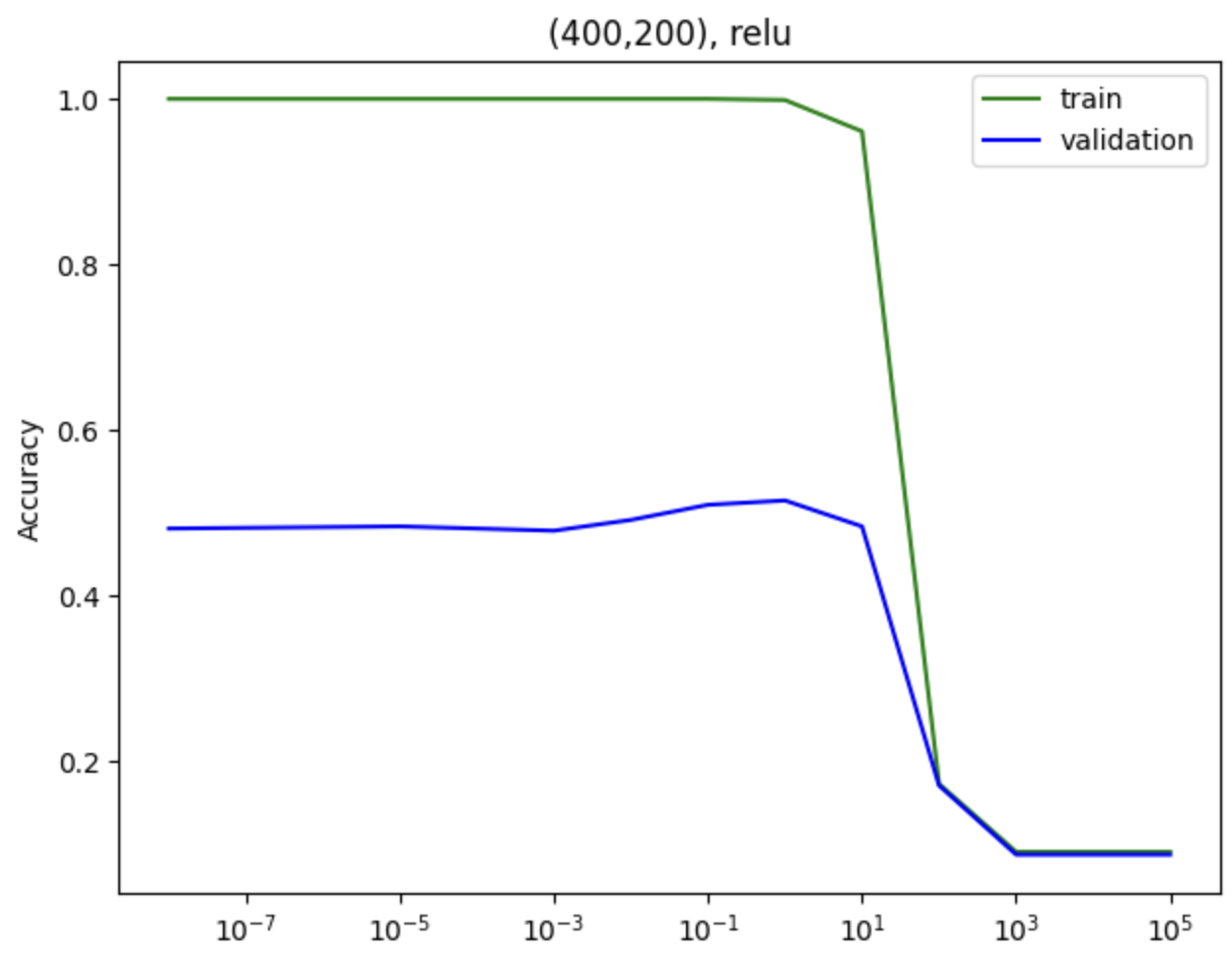
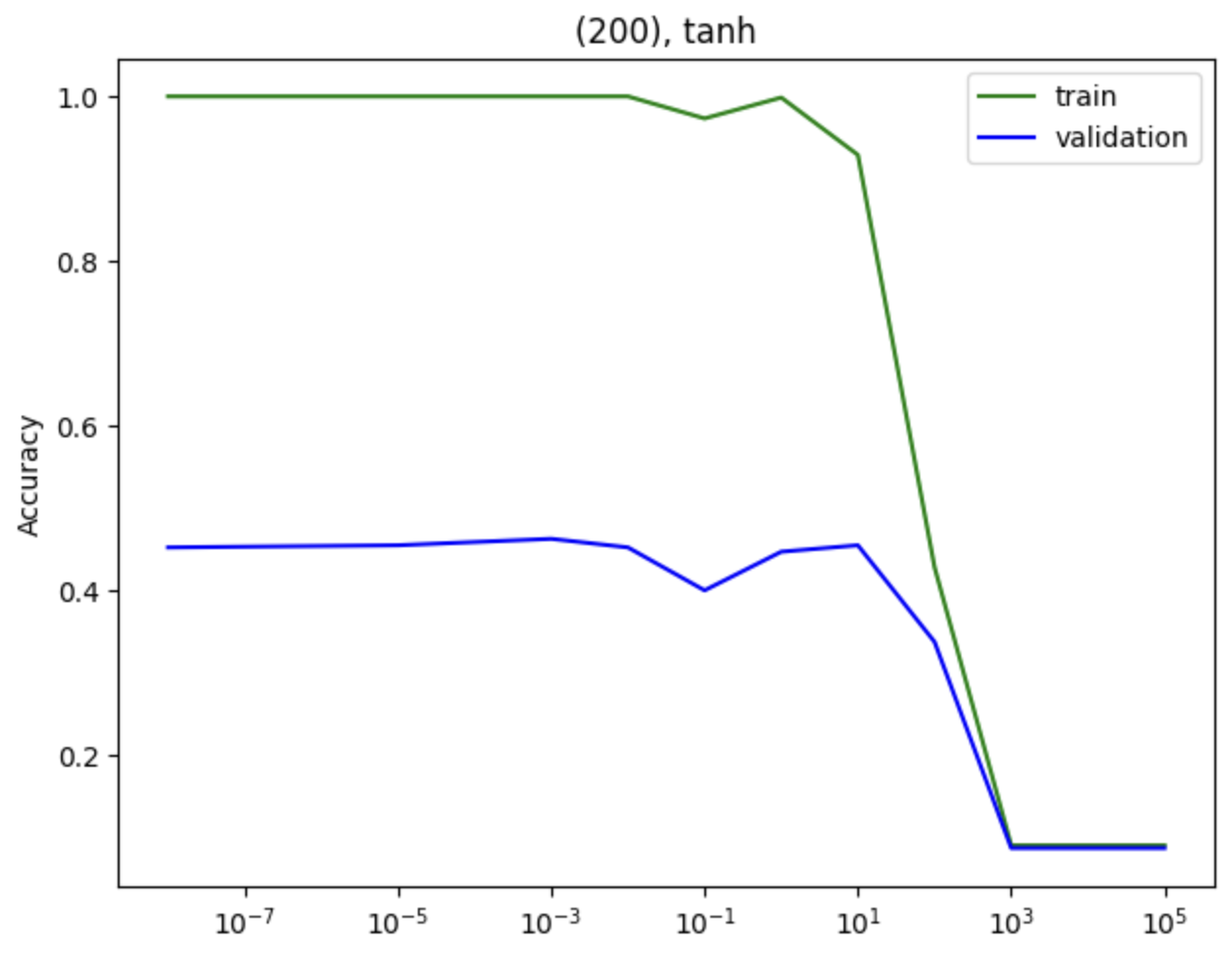
Then the confusion matrix for the best model among SVM models:

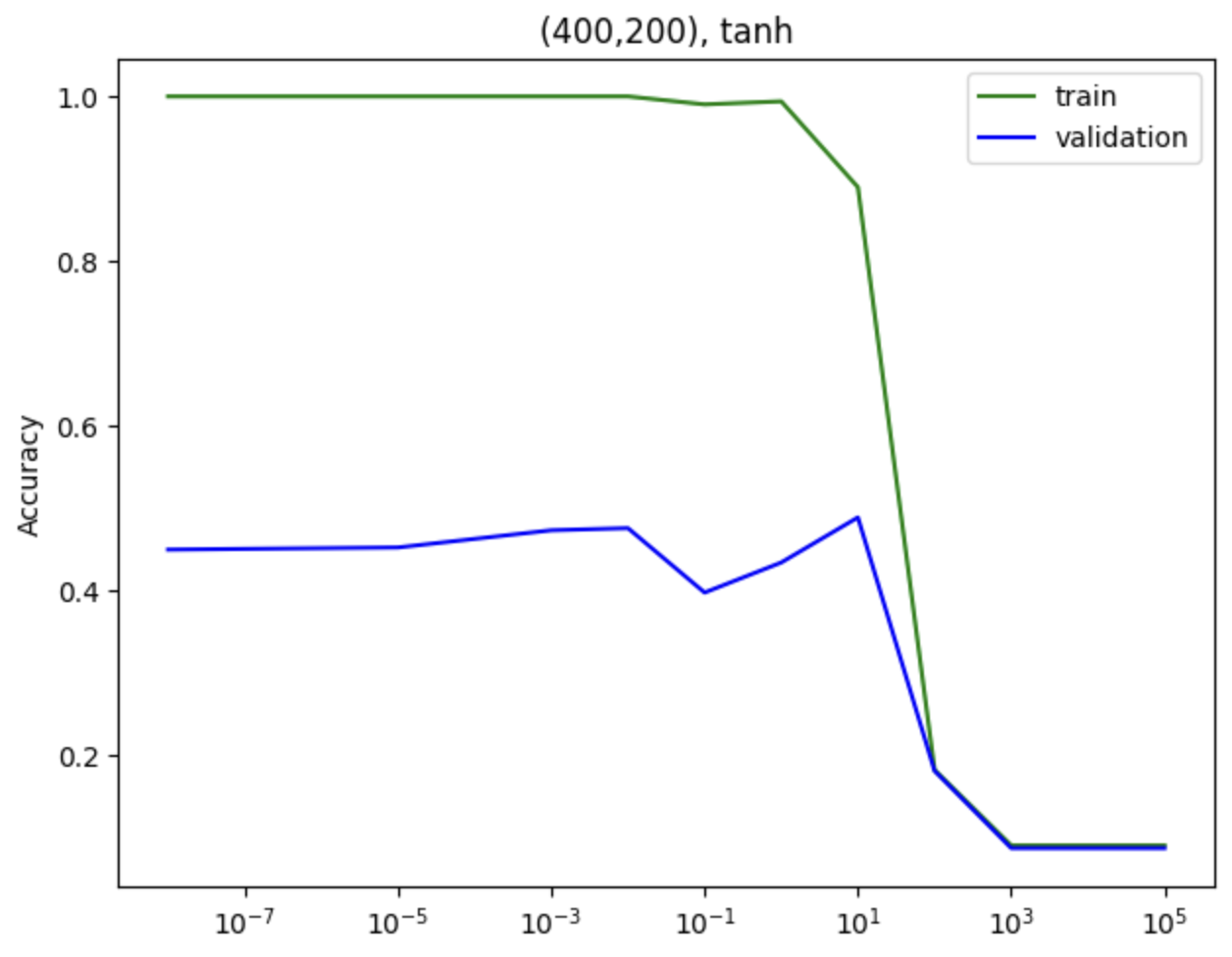
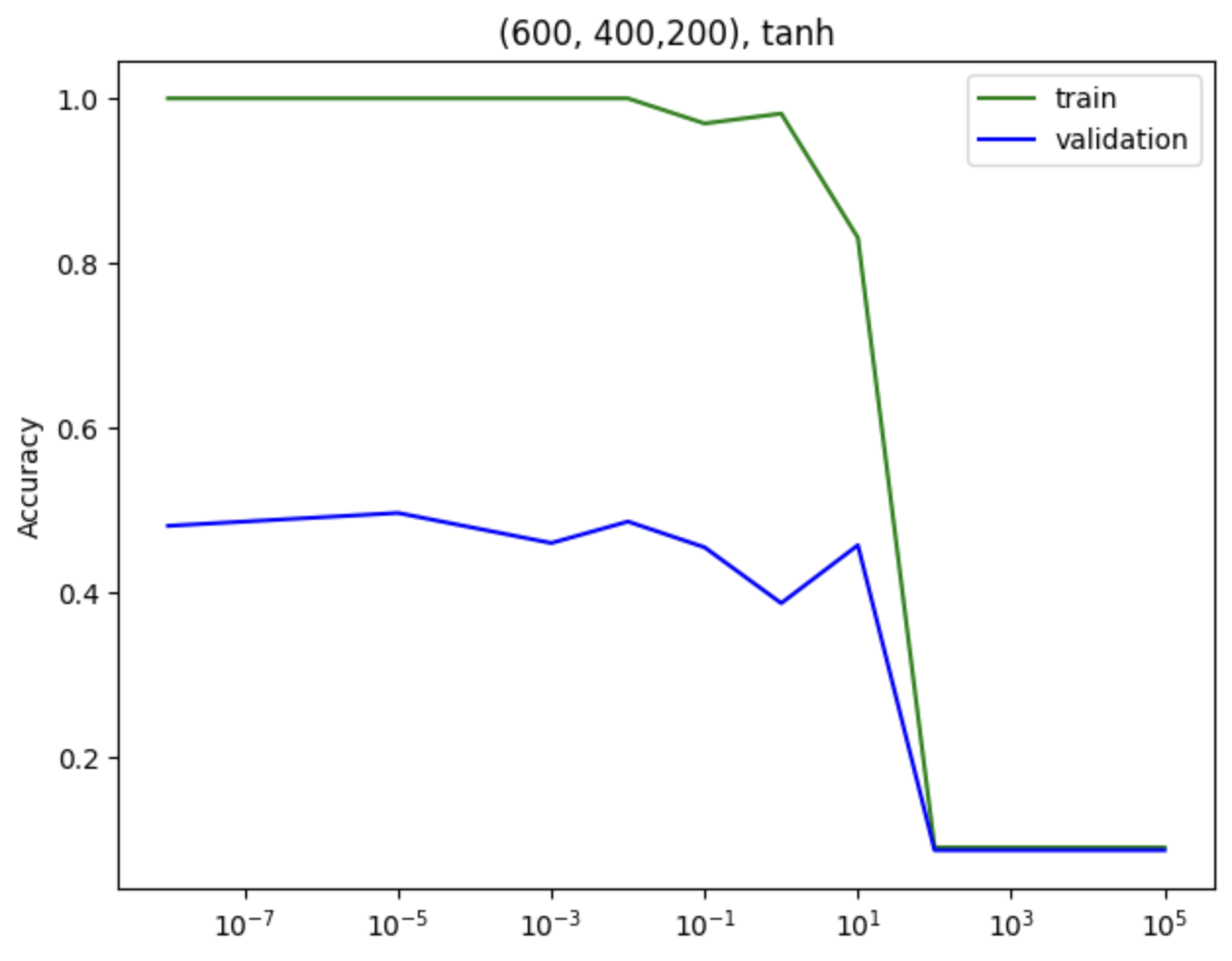
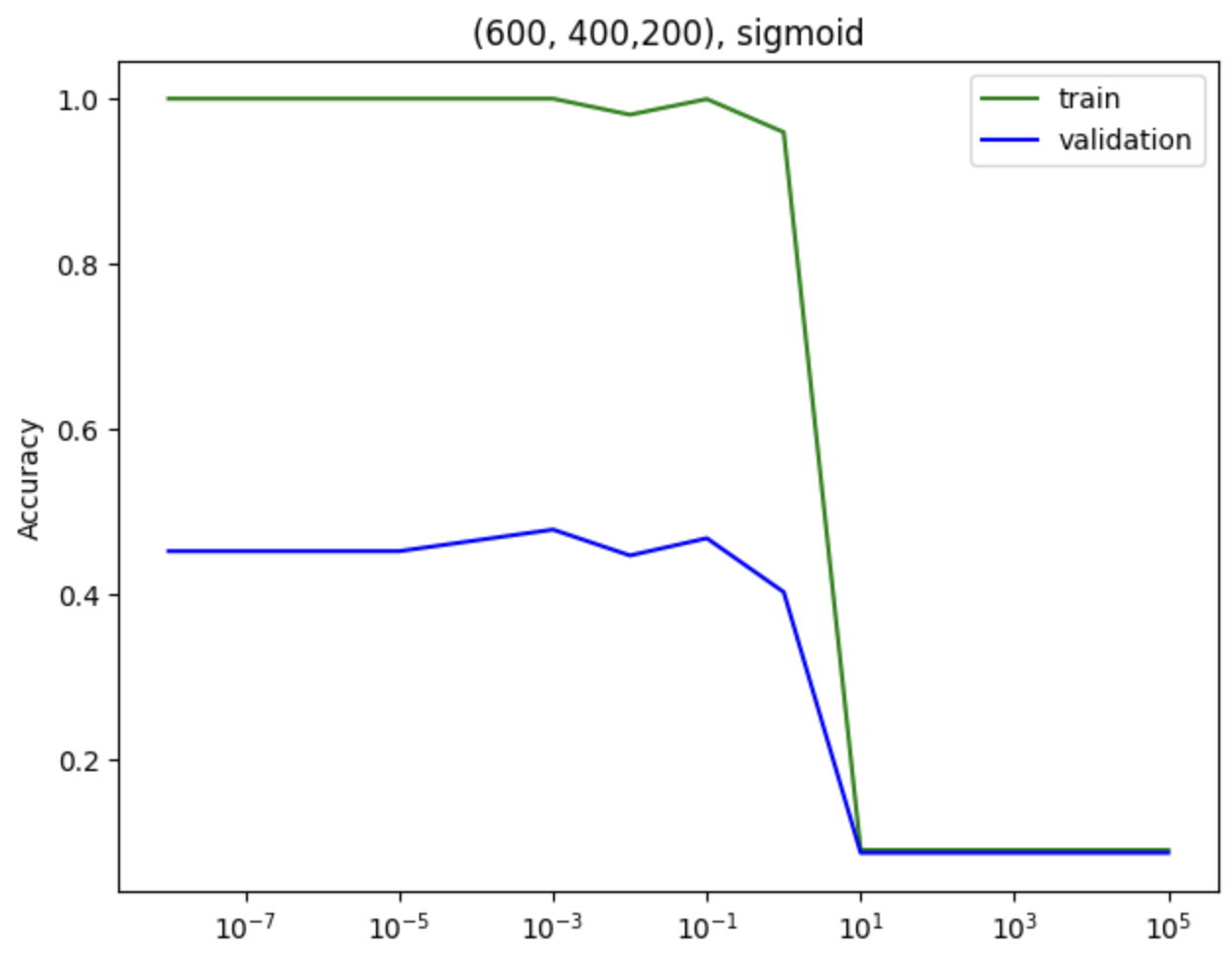
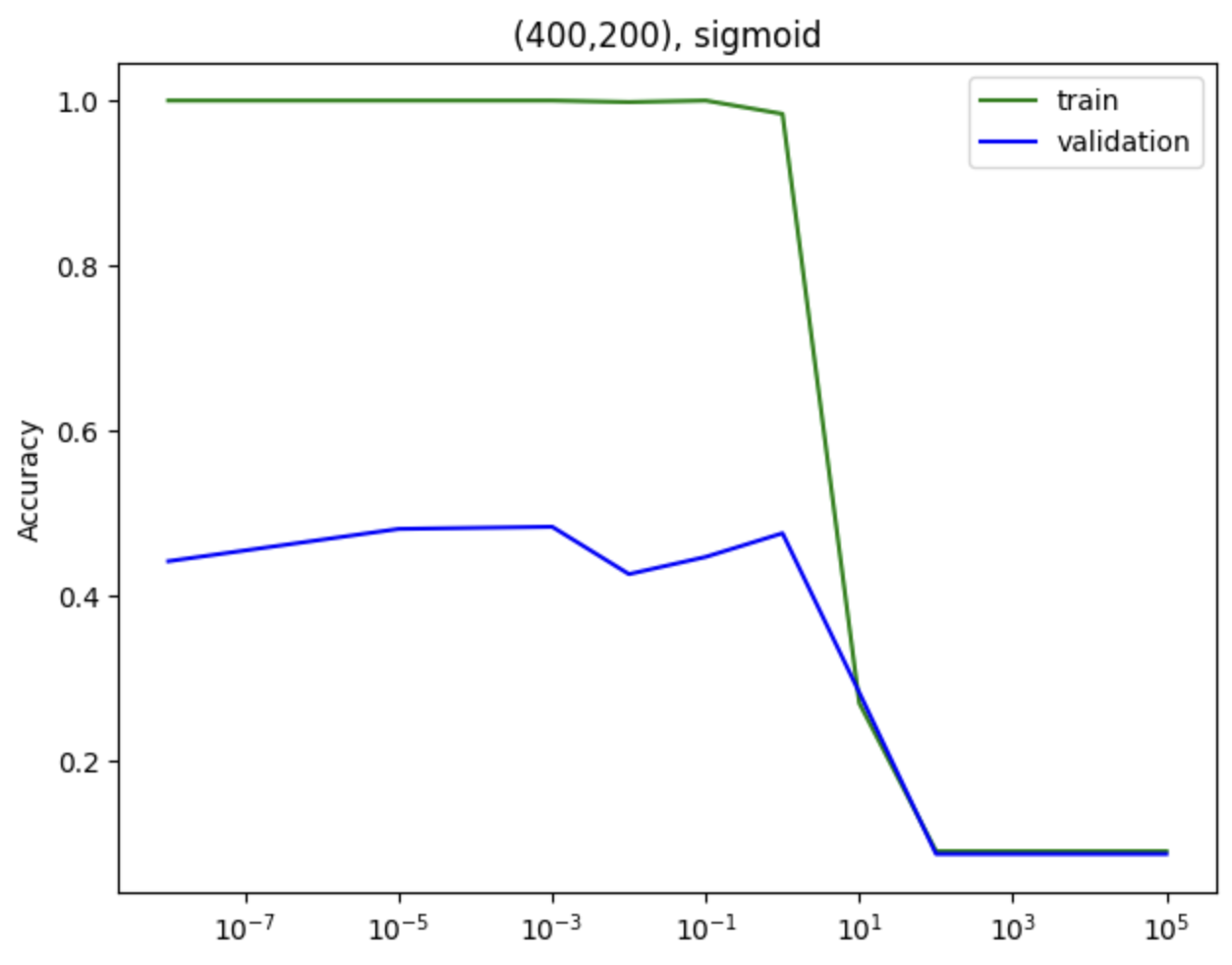
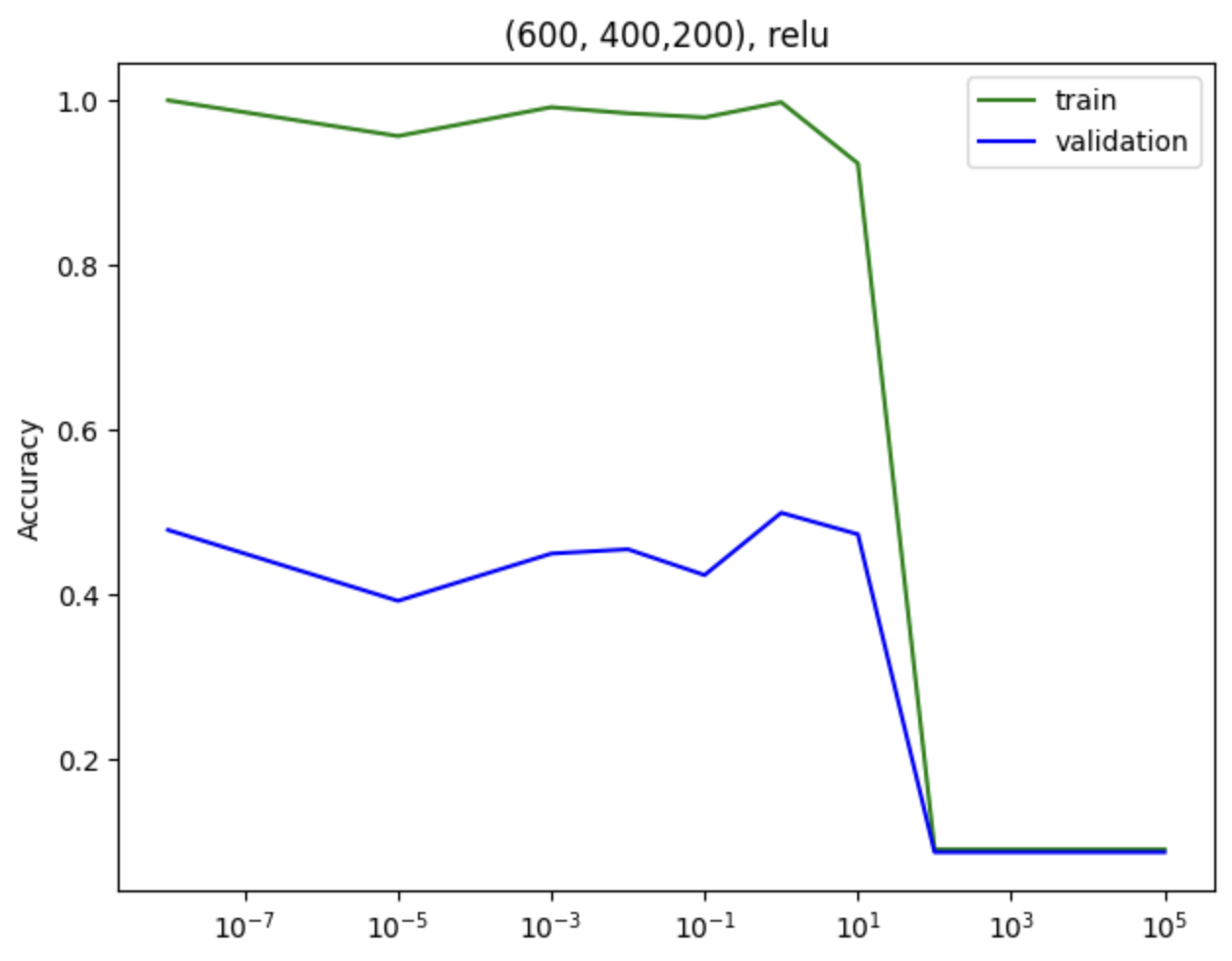
*(X, rbf, C=10)*

**3) Neural Network**

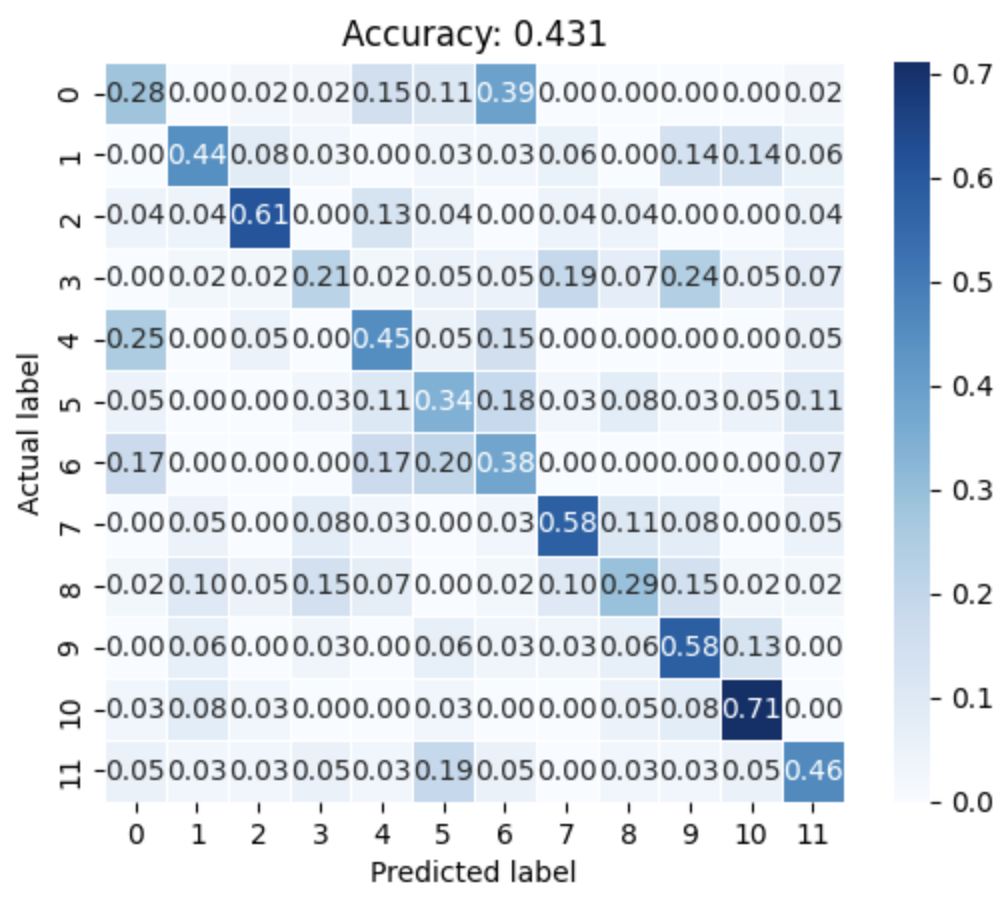
Lastly, neural network(NN) models are tested on the dataset. For training the models MLPClassifier function from sklearn library is used. Three activation functions are applied to the models: Rectified Linear Unit(ReLu), Logistic(sigmoid), and tanh. Additionally, three different NN hidden layer structures are used: (200), (400,200) and (600,400,200). For each on the combination above, each of the hyperparameter C(in this case called alpha, the normal regularizing strength) mentioned before are tested. The results are shown below:







Finally, confusion matrix on the testing dataset by the best model within ones mentioned above:

*((400,200), ReLu, alpha=1)*

**Results**

**Logistic Regression**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| alpha =>  Feature/Method↓ | 1e-8 | 1e-5 | 1e-3 | 1e-2 | 1e-1 | 1 | 1e1 | 1e2 | 1e3 | 1e5 |
| X  LASSO  train | 0.077 | 0.077 | 0.497 | 0.993 | 1 | 1 | 1 | 1 | 1 | 1 |
| X  LASSO  val | 0.094 | 0.094 | 0.290 | 0.446 | 0.405 | 0.405 | 0.402 | 0.402 | 0.368 | 0.371 |
| X  Ridge  train | 0.608 | 0.901 | 0.823 | 0.806 | 0.805 | 0.804 | 0.803 | 0.813 | 0.808 | 0.817 |
| X  Ridge  val | 0.417 | 0.407 | 0.418 | 0.407 | 0.404 | 0.399 | 0.394 | 0.407 | 0.399 | 0.402 |
| X2  LASSO  train | 0.077 | 0.271 | 0.974 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| X2  LASSO  val | 0.094 | 0.149 | 0.360 | 0.345 | 0.332 | 0.321 | 0.321 | 0.303 | 0.305 | 0.298 |
| X2  Ridge  train | 0.958 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| X2  Ridge  val | 0.389 | 0.355 | 0.347 | 0.337 | 0.326 | 0.332 | 0.332 | 0.337 | 0.321 | 0.324 |
| X3  LASSO  train | 0.077 | 0.982 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| X3  LASSO  val | 0.094 | 0.311 | 0.279 | 0.279 | 0.261 | 0.279 | 0.272 | 0.256 | 0.272 | 0.269 |
| X3  Ridge  train | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| X3  Ridge  val | 0.303 | 0.305 | 0.285 | 0.287 | 0.292 | 0.289 | 0.287 | 0.290 | 0.285 | 0.287 |

**SVM**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| alpha =>  Feature/Method↓ | 1e-8 | 1e-5 | 1e-3 | 1e-2 | 1e-1 | 1 | 1e1 | 1e2 | 1e3 | 1e5 |
| X  Linear  train | 0.090 | 0.335 | 0.984 | 0.982 | 0.982 | 0.982 | 0.982 | 0.982 | 0.982 | 0.982 |
| X  Linear  val | 0.086 | 0.222 | 0.488 | 0.483 | 0.483 | 0.483 | 0.483 | 0.483 | 0.483 | 0.483 |
| X  Poly  train | 0.090 | 0.090 | 0.091 | 0.116 | 0.224 | 0.436 | 0.878 | 0.887 | 0.888 | 0.888 |
| X  Linear  val | 0.086 | 0.086 | 0.086 | 0.097 | 0.102 | 0.117 | 0.287 | 0.305 | 0.303 | 0.303 |
| X  rbf  train | 0.090 | 0.090 | 0.090 | 0.090 | 0.169 | 0.091 | 1 | 1 | 1 | 1 |
| X  rbf  val | 0.086 | 0.086 | 0.086 | 0.086 | 0.172 | 0.493 | 0.535 | 0.535 | 0.535 | 0.535 |
| X2  Linear  train | 0.884 | 0.957 | 0.957 | 0.957 | 0.957 | 0.957 | 0.957 | 0.957 | 0.957 | 0.957 |
| X2  Linear  val | 0.394 | 0.379 | 0.379 | 0.379 | 0.379 | 0.379 | 0.379 | 0.379 | 0.379 | 0.379 |
| X2  Poly  train | 0.097 | 0.097 | 0.097 | 0.094 | 0.126 | 0.360 | 0.595 | 0.584 | 0.584 | 0.584 |
| X2  Poly  val | 0.107 | 0.107 | 0.104 | 0.110 | 0.128 | 0.240 | 0.298 | 0.300 | 0.300 | 0.300 |
| X2  rbf  train | 0.173 | 0.173 | 0.172 | 0.172 | 0.194 | 0.658 | 0.929 | 0.927 | 0.927 | 0.927 |
| X2  rbf  val | 0.178 | 0.178 | 0.180 | 0.172 | 0.183 | 0.345 | 0.452 | 0.449 | 0.449 | 0.449 |

**Neural Network**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| alpha =>  Feature/Method↓ | 1e-8 | 1e-5 | 1e-3 | 1e-2 | 1e-1 | 1 | 1e1 | 1e2 | 1e3 | 1e5 |
| (200)  ReLu  train | 1 | 1 | 1 | 0.992 | 1 | 0.996 | 0.980 | 0.518 | 0.090 | 0.090 |
| (200)  ReLu  val | 0.439 | 0.454 | 0.457 | 0.379 | 0.496 | 0.465 | 0.449 | 0.397 | 0.086 | 0.086 |
| (200)  Logistic  train | 1 | 1 | 1 | 1 | 1 | 0.996 | 0.758 | 0.145 | 0.090 | 0.090 |
| (200)  Logistic  val | 0.467 | 0.460 | 0.460 | 0.452 | 0.449 | 0.446 | 0.460 | 0.149 | 0.086 | 0.086 |
| (200)  tanh  train | 1 | 1 | 1 | 1 | 0.973 | 0.999 | 0.929 | 0.428 | 0.090 | 0.090 |
| (200)  tanh  val | 0.452 | 0.454 | 0.462 | 0.452 | 0.399 | 0.446 | 0.454 | 0.337 | 0.086 | 0.086 |
| (400,200)  ReLu  train | 1 | 1 | 1 | 1 | 1 | 0.999 | 0.961 | 0.172 | 0.090 | 0.090 |
| (400,200)  ReLu  val | 0.480 | 0.483 | 0.478 | 0.491 | 0.509 | 0.514 | 0.483 | 0.170 | 0.086 | 0.086 |
| (400,200)  Logistic  train | 1 | 1 | 1 | 0.998 | 1 | 0.984 | 0.269 | 0.090 | 0.090 | 0.090 |
| (400,200)  Logistic  val | 0.441 | 0.480 | 0.483 | 0.426 | 0.445 | 0.475 | 0.282 | 0.086 | 0.086 | 0.086 |
| (400,200)  tanh  train | 1 | 1 | 1 | 1 | 0.990 | 0.994 | 0.890 | 0.181 | 0.090 | 0.090 |
| (400,200)  tanh  val | 0.449 | 0.452 | 0.473 | 0.475 | 0.397 | 0.433 | 0.488 | 0.180 | 0.086 | 0.086 |
| (600,  400,200)  ReLu  train | 1 | 0.957 | 0.992 | 0.984 | 0.979 | 0.998 | 0.923 | 0.090 | 0.090 | 0.090 |
| (600,  400,200)  ReLu  val | 0.478 | 0.392 | 0.449 | 0.454 | 0.423 | 0.499 | 0.473 | 0.086 | 0.086 | 0.086 |
| (600,  400,200)  Logistic  train | 1 | 1 | 1 | 0.981 | 1.000 | 0.959 | 0.090 | 0.090 | 0.090 | 0.090 |
| (600,  400,200)  Logistic  val | 0.452 | 0.452 | 0.478 | 0.446 | 0.467 | 0.402 | 0.086 | 0.086 | 0.086 | 0.086 |
| (600,  400,200)  tanh  train | 1 | 1 | 1 | 1 | 0.970 | 0.982 | 0.831 | 0.090 | 0.090 | 0.090 |
| (600,  400,200)  tanh  val | 0.480 | 0.496 | 0.460 | 0.486 | 0.454 | 0.386 | 0.457 | 0.086 | 0.086 | 0.086 |

**Conclusion**

Amongst all the models, SVM and NN appears to function similarly well with NN winning by a small amoung. The best performing model overall is the Neural Network model with hidden layers (400, 200) using ReLu activation function and having regularizing hyperparameter alpha = 1. The reduction of feature size(i.e. Less clear pictures) applied to SVM and NN doesn’t appear to reduce accuracy of models since model accuracy increased for these two methods. For Logistic Regression, as the training accuracy increase, validation accuracy tend to start increasing. Until training accuracy reaches about 100%, validation accuracy tend to increase for a while as the regularizing parameter alpha(i.e 1/C) decreases, showing the models being underfit with too large to alpha, then the validation accuracy drops, meaning the occurrence of overfit. As shown in the results , the optimized model for each set up tends to end up between C=[0.1, 10]. Feature transformation did not produce better results since the best model is still trained with the original feature set.

As for SVM, no matter what kernel function applied, similar pattern as Logistic Regression happens for the mean time, but as alpha gets small enough, validation accuracy stays constant as 1/C continues to decrease. Same as Logistic regression, feature transformation does not results in better model accuracy, but it does not significantly reduce it either, except for the linear function case.

Lastly, for neural network, the validation accuracy are generally better compared to other two methods. The graph shows similar pattern to the ones of Logistic Regression, however, the plot is less smooth, existing more switches from increase to decrease and vice versa. But in general, using different activation functions does not affect the final validation accuracy of the model much, as optimal accuracy for each activation function being close to one another. The NN structures tested are proven helpful for estimating the plants by their images, yet, the optimal accuracy obtained is still merely 43.1%, meaning best structure has not been attempted. On top of that, the reduction in clarity of the images might have also caused models to have worse estimation accuracies.

Although 43.1% is not a very high accuracy, but some of the plants are extremely hard to distinguish even by eyes, thus, the result is very promising. There’s numerous ways to improve upon what was done in this project, one main thing being trying larger and better designed neural network structures. Also, with more time available, more features(clearer images) can be trained upon, and more data can be collected for better generalization of patterns.

**Works Cited**

Dataset for plants:

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

Documentation of image reading and resizing:

<https://pillow.readthedocs.io/en/stable/reference/Image.html>

Graphing confusion matrix:

<https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html>

Model training for Logsitic regression:

<https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html>

Model training for SVM:

<https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.htm>

Model training for Neural Network:

<https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html>