CS 4564 Introduction to Machine Learning

Who’s That Pokémon?:

Pokemon Image Classification Final Project

Melissa Jacob

Jonathan Lin

Project Submission Date: 4/28/2021

**Objective**

The objective of this project is to use machine learning methods learned in class to correctly classify Pokemons. This project will use machine learning methods such as logistic regression, support vector machines, and neural networks to determine the best model for classifying Pokemons. The dataset contains 150 different Pokemons with almost 7000 samples spread across the 150 different Pokemons. The goal will be to use the methods learned in class to correctly classify each Pokemon correctly.

Dataset: <https://www.kaggle.com/lantian773030/pokemonclassification>

**Problem Statement**

With eight generations of Pokemons out there, many people start to forget the first generation of Pokemons and their names. We want to be able to input an image of one of the 150 first generation Pokemons and be able to know what Pokemon that is.

**Data Preprocessing**

Originally we wanted to use all 7000 samples, 150 different categories, and with higher resolution, however, due to machine limitations we ended up scaling down the amount of data we use so we can properly complete the project.

We ended up using 14 distinct Pokemons and about 650 samples. We converted the image to HSV with 40 by 40 resolution which gave us 4800 features per image. The image was then put into a numpy array and flattened. All image data was split into training and testing sets using a stratified split to ensure that there were samples from all targets in each set. Finally, the sets were put through a standard scaler.

The targets did not all have the same number of samples, meaning our dataset was unbalanced. To counter this, all models used were passed the parameter class\_weight=’balanced’ to automatically adjust and balance the weights.

**Logistic Regression**

The first model we used to test was logistic regression. We utilized the scikit-learn logistic regression class to assist with the model. We also reused code from homework 4 to assist with the logistic regression model. For logistic regression we tested the model with no penalty, L1 penalty, and L2 penalty. For L1 and L2 penalty it was tested with C values between 0.0001, 0.001, 0.01, 0.1, 1, 10, 100. Different image transformations were also tested, images with no transformation, vertical flip, rotation, and brighten were tested.

Each of these models have a higher testing error than training error, indicating that they have low variance, but high bias.

Overall, these transformations did not appear to have a significant effect on the accuracy or precision, as they all leveled out to a training accuracy of 100% and testing accuracy of 73% with weighted precision of 77%. So the best method is L2 regularization with c=0.001.

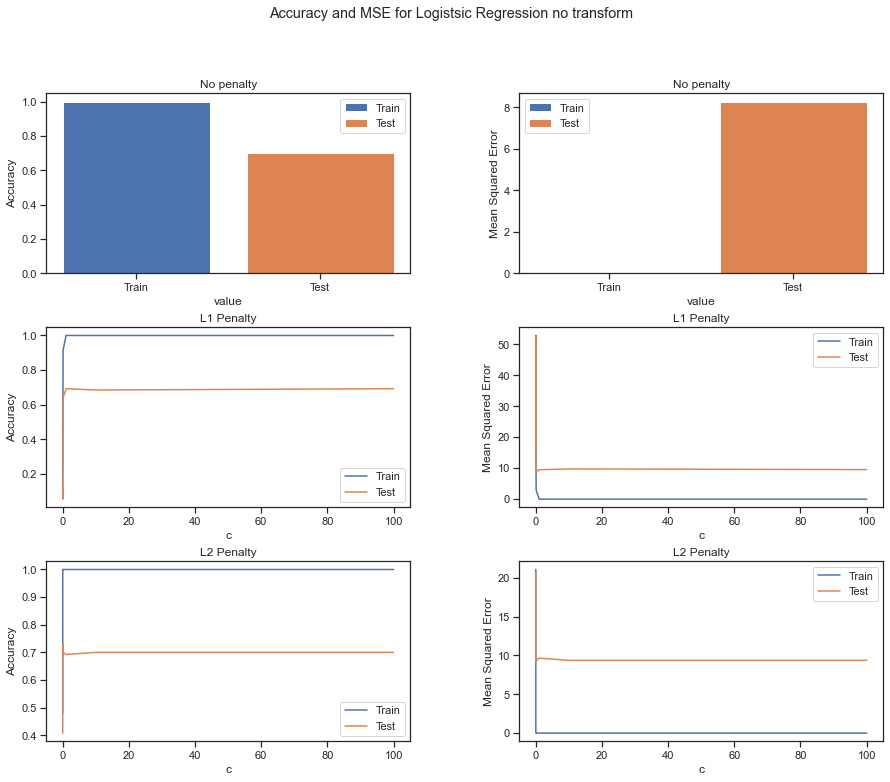
**No Transformation**

The first test was untransformed images. The table below shows the accuracy, weighted average precision, and weighted average recall on both test and training sets, each tested with no penalty, L1 regularization and L2 regularization. For no regularization, the training accuracy was 100% while the testing accuracy was 70%. For L1 regularization, c value of 100 gave the best training accuracy of 100% and testing accuracy of 73% with weighted precision of 73%. For L2 regularization, c value of 0.001 gave the best training accuracy of 100% and testing accuracy of 73% with weighted precision of 77%.

The best overall result for the untransformed image was L2 regularization with c=0.001 which had training accuracy and weighted precision of 100% and a testing accuracy of 73% with weighted precision 77%.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| No transform | Training | | | Testing | | |
|  | Accuracy | Weighted Average Precision | Weighted Average recall | Accuracy | Weighted Average Precision | Weighted Average recall |
| No regularization | 1.00 | 1.00 | 1.00 | 0.70 | 0.72 | 0.70 |
| L1. c=0.00001 | 0.08 | 0.01 | 0.08 | 0.08 | 0.01 | 0.08 |
| L1, c=0.0001 | 0.09 | 0.01 | 0.09 | 0.09 | 0.01 | 0.08 |
| L1, c=0.001 | 0.09 | 0.01 | 0.09 | 0.10 | 0.01 | 0.10 |
| L1, c=0.01 | 0.06 | 0.00 | 0.06 | 0.05 | 0.00 | 0.05 |
| L1, c=0.1 | 0.91 | 0.92 | 0.91 | 0.64 | 0.68 | 0.64 |
| L1, c=1 | 1.00 | 1.00 | 1.00 | 0.69 | 0.69 | 0.69 |
| L1, c=10 | 1.00 | 1.00 | 1.00 | 0.68 | 0.69 | 0.68 |
| L1, c=100 | 1.00 | 1.00 | 1.00 | 0.69 | 0.73 | 0.69 |
| L2. c=0.00001 | 0.48 | 0.66 | 0.48 | 0.41 | 0.59 | 0.41 |
| L2, c=0.0001 | 0.79 | 0.82 | 0.79 | 0.62 | 0.67 | 0.62 |
| L2, c=0.001 | 0.96 | 0.96 | 0.96 | 0.73 | 0.77 | 0.73 |
| L2, c=0.01 | 1.00 | 1.00 | 1.00 | 0.70 | 0.73 | 0.70 |
| L2, c=0.1 | 1.00 | 1.00 | 1.00 | 0.70 | 0.73 | 0.70 |
| L2, c=1 | 1.00 | 1.00 | 1.00 | 0.69 | 0.72 | 0.69 |
| L2, c=10 | 1.00 | 1.00 | 1.00 | 0.70 | 0.73 | 0.70 |
| L2, c=100 | 1.00 | 1.00 | 1.00 | 0.70 | 0.73 | 0.70 |

The graphs below demonstrate the accuracy and mean square error of logistic regression with comparison to no penalty, L1 penalty, and L2 penalty. It can be seen that as C value increased, both L1 and L2 regularization increased in both training accuracy and testing accuracy with L1 regularization reaching optimum model at c = 100 and L2 regularization at c = 0.001. With an increase in c value, the model overfits and does not see an increase in testing accuracy.

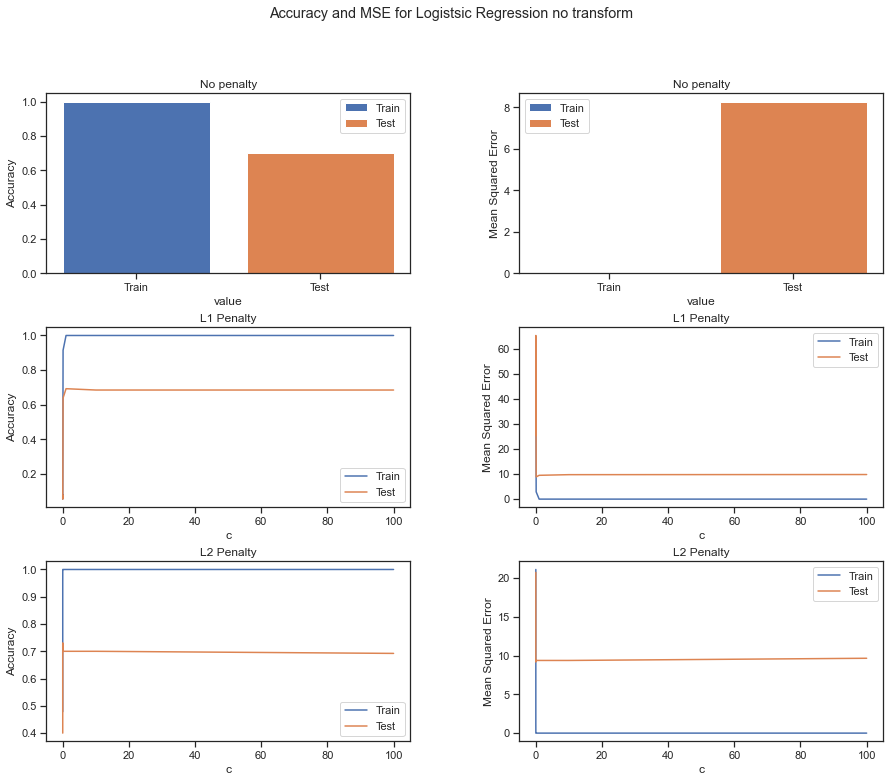


**Horizontal Flip**

The next test was on images flipped around the y axis. The table below shows the accuracy, weighted average precision, and weighted average recall on both test and training sets, each tested with no penalty, L1 regularization and L2 regularization. The best overall result for the vertically flipped image was L2 regularization with c=0.001 which had training accuracy and weighted precision of 100% and a testing accuracy of 73% with weighted precision 77%.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Vertical Flip | Training | | | Testing | | |
|  | Accuracy | Weighted Average Precision | Weighted Average recall | Accuracy | Weighted Average Precision | Weighted Average recall |
| No regularization | 1.00 | 1.00 | 1.00 | 0.70 | 0.72 | 0.70 |
| L1. c=0.00001 | 0.07 | 0.01 | 0.07 | 0.07 | 0.00 | 0.07 |
| L1, c=0.0001 | 0.05 | 0.00 | 0.05 | 0.05 | 0.00 | 0.05 |
| L1, c=0.001 | 0.08 | 0.01 | 0.08 | 0.08 | 0.01 | 0.08 |
| L1, c=0.01 | 0.06 | 0.00 | 0.06 | 0.06 | 0.00 | 0.06 |
| L1, c=0.1 | 0.91 | 0.92 | 0.91 | 0.64 | 0.68 | 0.64 |
| L1, c=1 | 1.00 | 1.00 | 1.00 | 0.69 | 0.69 | 0.69 |
| L1, c=10 | 1.00 | 1.00 | 1.00 | 0.68 | 0.69 | 0.68 |
| L1, c=100 | 1.00 | 1.00 | 1.00 | 0.68 | 0.72 | 0.68 |
| L2. c=0.00001 | 0.48 | 0.65 | 0.48 | 0.40 | 0.58 | 0.40 |
| L2, c=0.0001 | 0.79 | 0.82 | 0.79 | 0.62 | 0.67 | 0.62 |
| L2, c=0.001 | 0.96 | 0.96 | 0.96 | 0.73 | 0.77 | 0.73 |
| L2, c=0.01 | 1.00 | 1.00 | 1.00 | 0.70 | 0.3 | 0.70 |
| L2, c=0.1 | 1.00 | 1.00 | 1.00 | 0.70 | 0.73 | 0.70 |
| L2, c=1 | 1.00 | 1.00 | 1.00 | 0.70 | 0.73 | 0.70 |
| L2, c=10 | 1.00 | 1.00 | 1.00 | 0.70 | 0.73 | 0.70 |
| L2, c=100 | 1.00 | 1.00 | 1.00 | 0.69 | 0.72 | 0.69 |

The graphs below demonstrate the accuracy and mean square error of logistic regression with comparison to no penalty, L1 penalty, and L2 penalty. It can be seen that as C value increased, both L1 and L2 regularization increased in both training accuracy and testing accuracy with L1 regularization reaching optimum model at c = 100 and L2 regularization at c = 0.001. With an increase in c value, the model overfits and does not see an increase in testing accuracy.

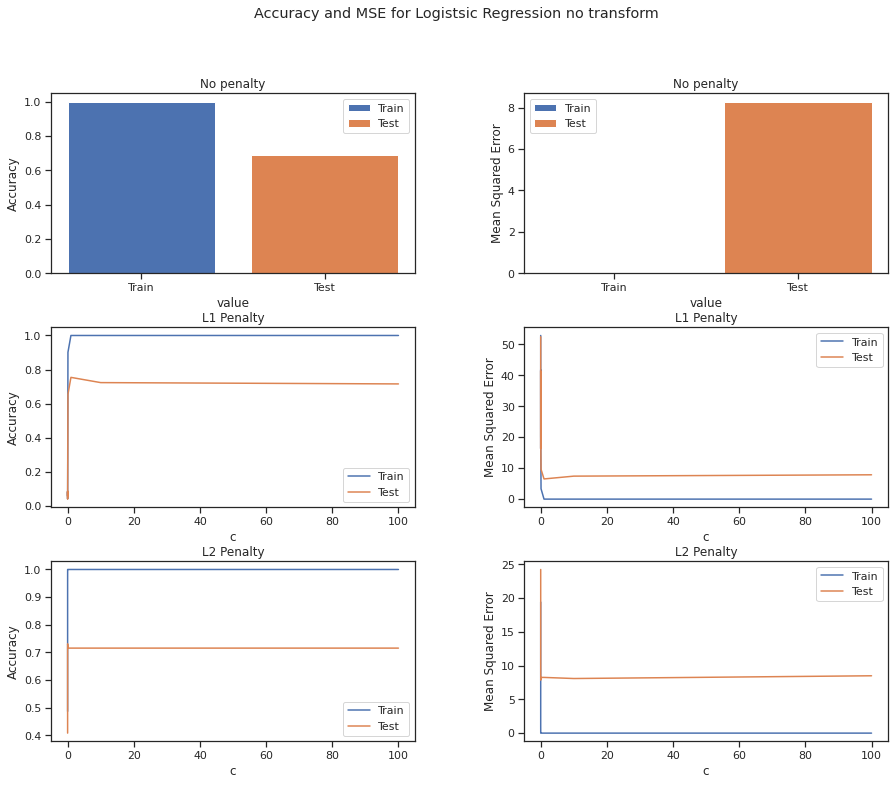


**15 Degree Rotation**

The next test was on images rotated by 15 degrees. The table below shows the accuracy, weighted average precision, and weighted average recall on both test and training sets, each tested with no penalty, L1 regularization and L2 regularization. The best overall result for the vertically flipped image was L2 regularization with c=0.001 which had training accuracy 100% with weighted precision of 96% and a testing accuracy of 73% with weighted precision 77%.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 15 degree rotate | Training | | | Testing | | |
|  | Accuracy | Weighted Average Precision | Weighted Average recall | Accuracy | Weighted Average Precision | Weighted Average recall |
| No regularization | 1.00 | 1.00 | 1.00 | 0.67 | 0.69 | 0.67 |
| L1. c=0.00001 | 0.07 | 0.00 | 0.07 | 0.07 | 0.00 | 0.07 |
| L1, c=0.0001 | 0.04 | 0.00 | 0.04 | 0.04 | 0.00 | 0.04 |
| L1, c=0.001 | 0.08 | 0.01 | 0.08 | 0.08 | 0.01 | 0.08 |
| L1, c=0.01 | 0.08 | 0.01 | 0.08 | 0.08 | 0.01 | 0.08 |
| L1, c=0.1 | 0.91 | 0.92 | 0.91 | 0.64 | 0.68 | 0.64 |
| L1, c=1 | 1.00 | 1.00 | 1.00 | 0.69 | 0.69 | 0.69 |
| L1, c=10 | 1.00 | 1.00 | 1.00 | 0.68 | 0.69 | 0.68 |
| L1, c=100 | 1.00 | 1.00 | 1.00 | 0.68 | 0.72 | 0.68 |
| L2. c=0.00001 | 0.48 | 0.65 | 0.48 | 0.40 | 0.58 | 0.40 |
| L2, c=0.0001 | 0.79 | 0.82 | 0.79 | 0.62 | 0.67 | 0.62 |
| L2, c=0.001 | 0.96 | 0.96 | 0.96 | 0.73 | 0.77 | 0.73 |
| L2, c=0.01 | 1.00 | 1.00 | 1.00 | 0.70 | 0.3 | 0.70 |
| L2, c=0.1 | 1.00 | 1.00 | 1.00 | 0.70 | 0.73 | 0.70 |
| L2, c=1 | 1.00 | 1.00 | 1.00 | 0.70 | 0.73 | 0.70 |
| L2, c=10 | 1.00 | 1.00 | 1.00 | 0.70 | 0.73 | 0.70 |
| L2, c=100 | 1.00 | 1.00 | 1.00 | 0.69 | 0.72 | 0.69 |

The graphs below demonstrate the accuracy and mean square error of logistic regression with comparison to no penalty, L1 penalty, and L2 penalty. It can be seen that as C value increased, both L1 and L2 regularization increased in both training accuracy and testing accuracy with L1 regularization reaching optimum model at c = 100 and L2 regularization at c = 0.001. With an increase in c value, the model overfits and does not see an increase in testing accuracy.

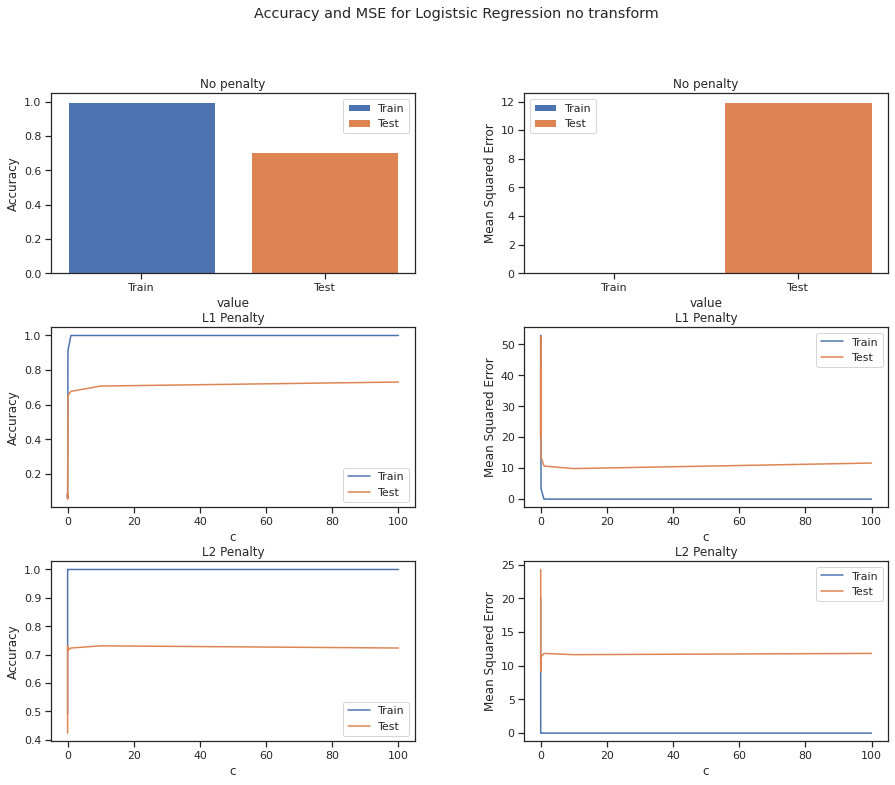


**1.5x Brightened**

The final test was on 1.5x brightened images. The table below shows the accuracy, weighted average precision, and weighted average recall on both test and training sets, each tested with no penalty, L1 regularization and L2 regularization. The best overall result for the brightened image was L2 regularization with c=0.01 which had training accuracy 100% withweighted precision of 100% and a testing accuracy of 73% with weighted precision 75%.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Brighten Image | Training | | | Testing | | |
|  | Accuracy | Weighted Average Precision | Weighted Average recall | Accuracy | Weighted Average Precision | Weighted Average recall |
| No regularization | 1.00 | 1.00 | 1.00 | 0.66 | 0.66 | 0.66 |
| L1. c=0.00001 | 0.07 | 0.01 | 0.07 | 0.07 | 0.00 | 0.07 |
| L1, c=0.0001 | 0.08 | 0.1 | 0.08 | 0.05 | 0.01 | 0.08 |
| L1, c=0.001 | 0.07 | 0.00 | 0.07 | 0.07 | 0.00 | 0.07 |
| L1, c=0.01 | 0.09 | 0.01 | 0.09 | 0.08 | 0.01 | 0.08 |
| L1, c=0.1 | 0.91 | 0.92 | 0.91 | 0.65 | 0.65 | 0.65 |
| L1, c=1 | 1.00 | 1.00 | 1.00 | 0.65 | 0.64 | 0.65 |
| L1, c=10 | 1.00 | 1.00 | 1.00 | 0.70 | 0.71 | 0.70 |
| L1, c=100 | 1.00 | 1.00 | 1.00 | 0.72 | 0.74 | 0.72 |
| L2. c=0.00001 | 0.49 | 0.68 | 0.49 | 0.39 | 0.51 | 0.39 |
| L2, c=0.0001 | 0.76 | 0.79 | 0.76 | 0.60 | 0.70 | 0.60 |
| L2, c=0.001 | 0.97 | 0.97 | 0.97 | 0.68 | 0.70 | 0.68 |
| L2, c=0.01 | 1.00 | 1.00 | 1.00 | 0.72 | 0.73 | 0.72 |
| L2, c=0.1 | 1.00 | 1.00 | 1.00 | 0.73 | 0.75 | 0.73 |
| L2, c=1 | 1.00 | 1.00 | 1.00 | 0.72 | 0.73 | 0.72 |
| L2, c=10 | 1.00 | 1.00 | 1.00 | 0.72 | 0.73 | 0.72 |
| L2, c=100 | 1.00 | 1.00 | 1.00 | 0.72 | 0.73 | 0.72 |

The graphs below demonstrate the accuracy and mean square error of logistic regression with comparison to no penalty, L1 penalty, and L2 penalty. It can be seen that as C value increased, both L1 and L2 regularization increased in both training accuracy and testing accuracy with L1 regularization reaching optimum model at c = 100 and L2 regularization at c = 10. With an increase in c value, the model overfits and does not see an increase in testing accuracy.



**SVM**

The second model we used was SVM. Similar to logistic regression we also utilized the scikit-learn SVM class and methods to assist with our model. We also reused code from homework 5 to further assist our coding. We used three different kernels: linear, polynomial, and rbf kernel. With each kernel, different c values were used ranging from 0.00001 to 100.

Each of these models have a higher testing error than training error, indicating that they have low variance, but high bias.

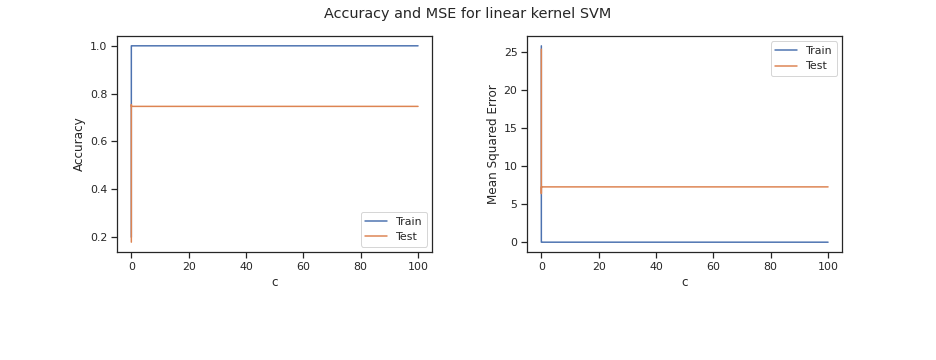
Overall, the best kernel used for SVM was the rbf kernel with c =10 which had a training accuracy of 100% and a testing accuracy of 75% with precision of 77%. Linear kernel with c = 0.001 was a close second with a slightly lower testing precision of 76%. The polynomial kernel was the worst of the three with a significantly lower accuracy of 63%, but similar precision of 74%.

**Linear Kernel**

For the linear kernel, the best c value for the model was reached at 0.001, where it reached 100% training accuracy and 75% testing accuracy. The model seems to have overfitting, however, other c values provided worse performance, for example the 0.001 gave a 100% training accuracy with only a 75% testing accuracy. The testing accuracy did not increase after c value of 0.001.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Training | | | Testing | | |
|  | Accuracy | Weighted Average Precision | Weighted Average recall | Accuracy | Weighted Average Precision | Weighted Average recall |
| c=0.00001 | 0.19 | 0.23 | 0.19 | 0.21 | 0.24 | 0.21 |
| c=0.0001 | 0.82 | 0.83 | 0.82 | 0.65 | 0.69 | 0.65 |
| c=0.001 | 1.00 | 1.00 | 1.00 | 0.75 | 0.76 | 0.75 |
| c=0.01 | 1.00 | 1.00 | 1.00 | 0.72 | 0.74 | 0.72 |
| c=0.1 | 1.00 | 1.00 | 1.00 | 0.72 | 0.74 | 0.72 |
| c=1 | 1.00 | 1.00 | 1.00 | 0.72 | 0.74 | 0.72 |
| c=10 | 1.00 | 1.00 | 1.00 | 0.72 | 0.74 | 0.72 |
| c=100 | 1.00 | 1.00 | 1.00 | 0.72 | 0.74 | 0.72 |

The graphs below demonstrate the accuracy and mean square error. It can be seen that as C value increased, both training accuracy and testing accuracy increased. With the highest accuracy of 75% occurring at c=0.001. With an increase in c value, the model overfits and does not see an increase in testing accuracy.

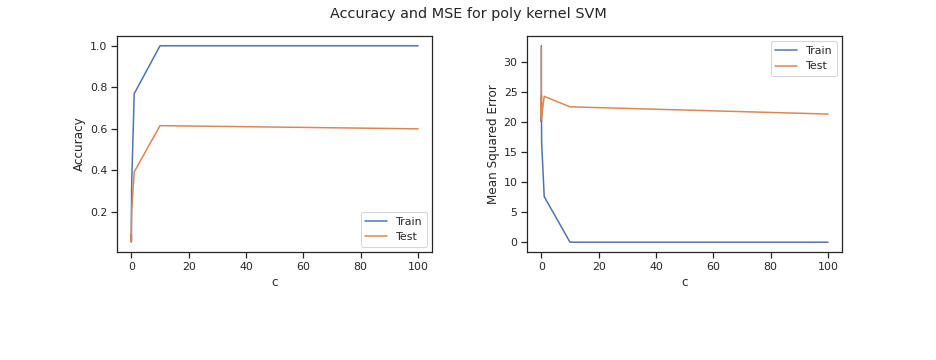


**Polynomial Kernel, degree=3**

For the polynomial kernel, the best c value for the model was reached at 10, where it reached 100% training accuracy and 63% testing accuracy. The model seems to have overfitting, however, other c values provided worse performance, for example c=1 gave a 76% training accuracy with a 40% testing accuracy. The testing accuracy decreased after c=10.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Polynomial degree=3 | Training | | | Testing | | |
|  | Accuracy | Weighted Average Precision | Weighted Average recall | Accuracy | Weighted Average Precision | Weighted Average recall |
| c=0.00001 | 0.09 | 0.01 | 0.09 | 0.09 | 0.01 | 0.09 |
| c=0.0001 | 0.06 | 0.00 | 0.06 | 0.05 | 0.00 | 0.05 |
| c=0.001 | 0.08 | 0.09 | 0.03 | 0.05 | 0.04 | 0.05 |
| c=0.01 | 0.13 | 0.11 | 0.13 | 0.09 | 0.05 | 0.09 |
| c=0.1 | 0.29 | 0.78 | 0.29 | 0.18 | 0.37 | 0.18 |
| c=1 | 0.76 | 0.92 | 0.75 | 0.40 | 0.62 | 0.40 |
| c=10 | 1.00 | 1.00 | 1.00 | 0.63 | 0.74 | 0.63 |
| c=100 | 1.00 | 1.00 | 1.00 | 0.59 | 0.74 | 0.59 |

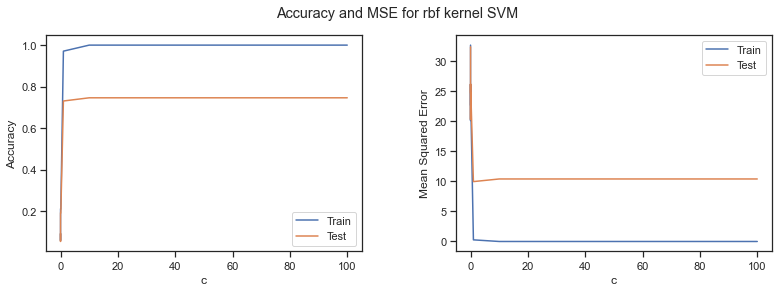
The graphs below demonstrate the accuracy and mean square error. It can be seen that as C value increased, both training accuracy and testing accuracy increased. With the highest accuracy of 63% occurring at c=10. With an increase in c value, the model overfits and does not see an increase in testing accuracy.

****

**RBF Kernel**

For the rbf kernel, the best c value for the model was reached at 10, where it reached 100% training accuracy and 78% testing accuracy. The model seems to have overfitting, however, the model with c = 1 gave 98% training accuracy and 74% testing accuracy, which could be a model just as good as c = 10. Other c values below 1 gave worse performance compared and c values above 10 did not improve the model.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| RBF | Training | | | Testing | | |
|  | Accuracy | Weighted Average Precision | Weighted Average recall | Accuracy | Weighted Average Precision | Weighted Average recall |
| c=0.00001 | 0.09 | 0.01 | 0.09 | 0.09 | 0.01 | 0.09 |
| c=0.0001 | 0.06 | 0.00 | 0.06 | 0.05 | 0.00 | 0.05 |
| c=0.001 | 0.13 | 0.04 | 0.13 | 0.10 | 0.02 | 0.10 |
| c=0.01 | 0.21 | 0.05 | 0.21 | 0.19 | 0.05 | 0.19 |
| c=0.1 | 0.21 | 0.05 | 0.21 | 0.19 | 0.05 | 0.19 |
| c=1 | 0.97 | 0.97 | 0.97 | 0.73 | 0.75 | 0.73 |
| c=10 | 1.00 | 1.00 | 1.00 | 0.75 | 0.77 | 0.75 |
| c=100 | 1.00 | 1.00 | 1.00 | 0.75 | 0.77 | 0.75 |

The graphs below demonstrate the accuracy and mean square error. It can be seen that as C value increased, both training accuracy and testing accuracy increased. With the highest accuracy of 75% occurring at c=10 and C=100. ****

**Neural Network**

Our last training method was neural networks. We utilized scikit-learn as well as homework 6 to assist us in our model building and training. For neural networks we used three different activation functions, with the three being: logistic/sigmoid, relu, and tanh activation. We also tested the model with different nn structures, with different numbers of neurons in the hidden layers as well as the number of hidden layers. The two nn structures that we tested were (4800, 210, 14) and (4800, 85, 324, 14) The input layer and output layer remained the same with 4800 as input and 14 as output. The alpha values ranged from 0.00001 to 100 for training the model.

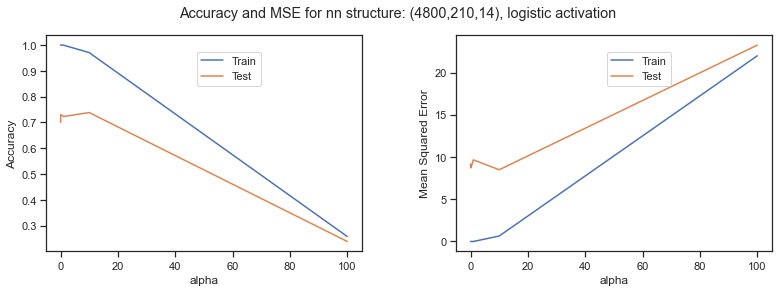
Each model begins with a higher testing error than training error, indicating that they have low variance but high bias. But as the value of c increases, they both have similar high errors, indicating that the model has high variance and low bias and has become overfit.

Overall, the best model was relu activation with an nn structure of (4800, 85, 324, 14) and alpha value of 0.1. This gave 100% training accuracy and 76% testing accuracy with 78% precision.

**Logistic activation**

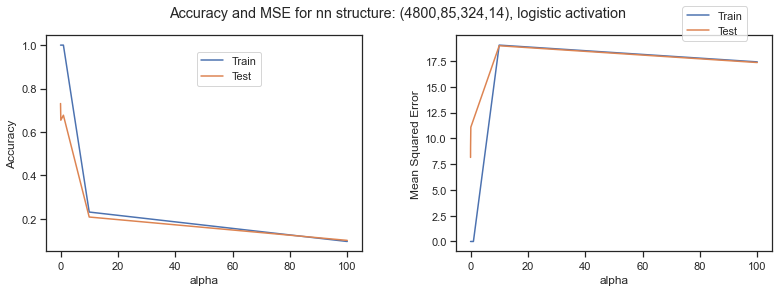
For the neural network using logistic/sigmoid activation with nn structure of (4800, 210, 14), the alpha value of 10 gave the best training and testing accuracy. The training accuracy of 97% and testing accuracy of 79% were the best compared to other alpha values. Other alpha values provided 100% training accuracy, however, the testing accuracy were lower. This shows that the alpha value of 10 provided the best fitting model while other models had overfitting.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| (4800,210,14)  logistic | Training | | | Testing | | |
|  | Accuracy | Weighted Average Precision | Weighted Average recall | Accuracy | Weighted Average Precision | Weighted Average recall |
| a=0 | 1.00 | 1.00 | 1.00 | 0.70 | 0.72 | 0.70 |
| a=0.00001 | 1.00 | 1.00 | 1.00 | 0.70 | 0.72 | 0.70 |
| a=0.0001 | 1.00 | 1.00 | 1.00 | 0.70 | 0.72 | 0.70 |
| a=0.001 | 1.00 | 1.00 | 1.00 | 0.70 | 0.72 | 0.70 |
| a=0.01 | 1.00 | 1.00 | 1.00 | 0.70 | 0.72 | 0.70 |
| a=0.1 | 1.00 | 1.00 | 1.00 | 0.73 | 0.74 | 0.73 |
| a=1 | 1.00 | 1.00 | 1.00 | 0.72 | 0.73 | 0.72 |
| a=10 | 0.97 | 0.97 | 0.97 | 0.74 | 0.74 | 0.74 |
| a=100 | 0.026 | 0.09 | 0.26 | 0.24 | 0.08 | 0.24 |



For the nn structure of (4800, 85, 324, 14), the alpha value of 0, 0,01, 0.001, and 0.0001 all gave the same training and testing accuracy value of 100% training accuracy and 69% testing accuracy, as alpha increased, the training and testing accuracy both decreased. It can be seen that the model is overfitted.

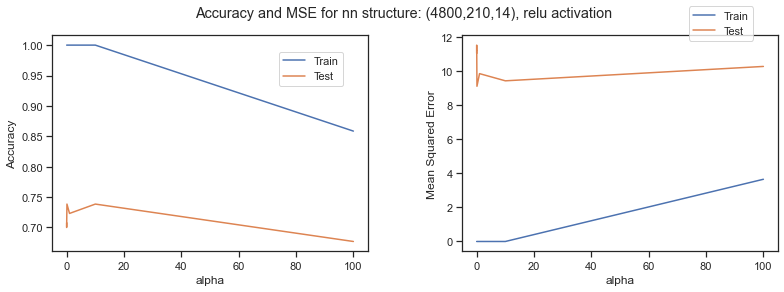
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| (4800,85,324,14)  logistic | Training | | | Testing | | |
|  | Accuracy | Weighted Average Precision | Weighted Average recall | Accuracy | Weighted Average Precision | Weighted Average recall |
| a=0 | 1.00 | 1.00 | 1.00 | 0.73 | 0.75 | 0.73 |
| a=0.00001 | 1.00 | 1.00 | 1.00 | 0.73 | 0.75 | 0.73 |
| a=0.0001 | 1.00 | 1.00 | 1.00 | 0.73 | 0.75 | 0.73 |
| a=0.001 | 1.00 | 1.00 | 1.00 | 0.73 | 0.75 | 0.73 |
| a=0.01 | 1.00 | 1.00 | 1.00 | 0.72 | 0.74 | 0.72 |
| a=0.1 | 1.00 | 1.00 | 1.00 | 0.65 | 0.68 | 0.65 |
| a=1 | 1.00 | 1.00 | 1.00 | 0.68 | 0.69 | 0.68 |
| a=10 | 0.23 | 0.07 | 0.23 | 0.21 | 0.06 | 0.21 |
| a=100 | 0.09 | 0.01 | 0.09 | 0.10 | 0.01 | 0.10 |



**ReLU Activation**

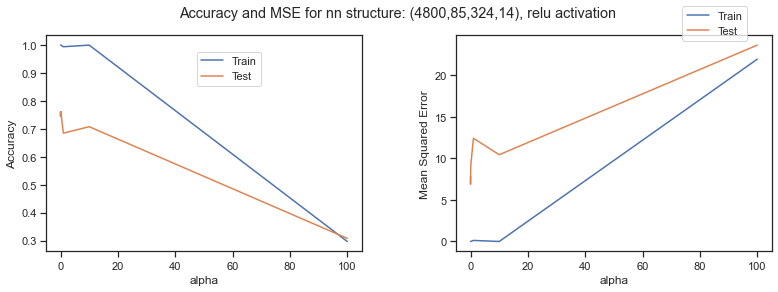
For the neural network using relu activation with nn structure of (4800, 210, 14), the alpha value of 0.01 gave the best training and testing accuracy. The training accuracy of 100% and testing accuracy of 76% were the best compared to other alpha values. Other alpha values also provided 100% training accuracy and the testing accuracy was close to 76%. This demonstrated that there is overfitting in the model.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| (4800,210,14)  relu | Training | | | Testing | | |
|  | Accuracy | Weighted Average Precision | Weighted Average recall | Accuracy | Weighted Average Precision | Weighted Average recall |
| a=0 | 1.00 | 1.00 | 1.00 | 0.71 | 0.72 | 0.71 |
| a=0.00001 | 1.00 | 1.00 | 1.00 | 0.71 | 0.72 | 0.71 |
| a=0.0001 | 1.00 | 1.00 | 1.00 | 0.70 | 0.71 | 0.70 |
| a=0.001 | 1.00 | 1.00 | 1.00 | 0.70 | 0.71 | 0.70 |
| a=0.01 | 1.00 | 1.00 | 1.00 | 0.71 | 0.72 | 0.71 |
| a=0.1 | 1.00 | 1.00 | 1.00 | 0.74 | 0.76 | 0.74 |
| a=1 | 1.00 | 1.00 | 1.00 | 0.72 | 0.74 | 0.72 |
| a=10 | 1.00 | 1.00 | 1.00 | 0.74 | 0.76 | 0.74 |
| a=100 | 1.00 | 1.00 | 1.00 | 0.68 | 0.68 | 0.68 |



For the nn structure of (4800, 85, 324, 14), the alpha value of 0.1 gave the best training and testing accuracy. The training accuracy of 100% and testing accuracy of gave the same training and testing accuracy of 75% were the best compared to other alpha values. Other alpha values also provided 100% training accuracy and similar testing accuracy. This demonstrates that the model is overfitted.

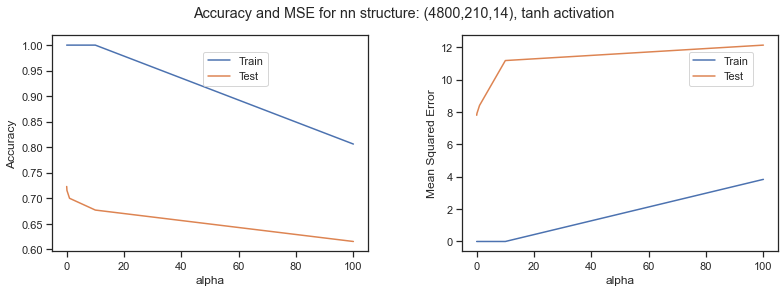
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| (4800,85,324,14)  relu | Training | | | Testing | | |
|  | Accuracy | Weighted Average Precision | Weighted Average recall | Accuracy | Weighted Average Precision | Weighted Average recall |
| a=0 | 1.00 | 1.00 | 1.00 | 0.75 | 0.74 | 0.75 |
| a=0.00001 | 1.00 | 1.00 | 1.00 | 0.75 | 0.74 | 0.75 |
| a=0.0001 | 1.00 | 1.00 | 1.00 | 0.75 | 0.74 | 0.75 |
| a=0.001 | 1.00 | 1.00 | 1.00 | 0.76 | 0.76 | 0.76 |
| a=0.01 | 1.00 | 1.00 | 1.00 | 0.75 | 0.76 | 0.75 |
| a=0.1 | 1.00 | 1.00 | 1.00 | 0.76 | 0.78 | 0.76 |
| a=1 | 0.99 | 0.99 | 0.99 | 0.68 | 0.69 | 0.68 |
| a=10 | 1.00 | 1.00 | 1.00 | 0.71 | 0.73 | 0.71 |
| a=100 | 0.30 | 0.12 | 0.30 | 0.31 | 0.13 | 0.31 |



**Tanh Activation**

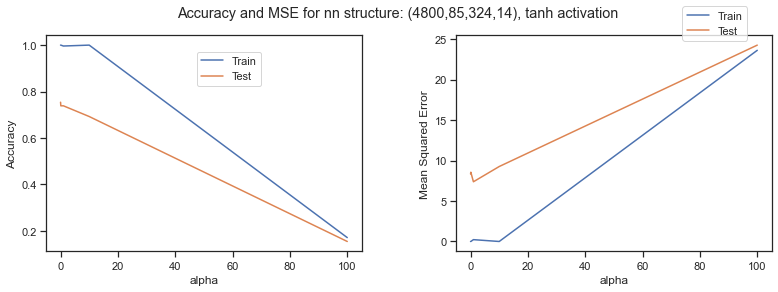
For the neural network using tanh activation with nn structure of (4800, 210, 14), the alpha values from 0.1 best training and testing accuracy. The training accuracy was 100% while the testing accuracy was 76%. The other alpha values provided the same training accuracy values but worse testing accuracy compared to alpha value of 0.1. It can be stated that there is overfitting with the model.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| (4800,210,14)  tanh | Training | | | Testing | | |
|  | Accuracy | Weighted Average Precision | Weighted Average recall | Accuracy | Weighted Average Precision | Weighted Average recall |
| a=0 | 1.00 | 1.00 | 1.00 | 0.73 | 0.74 | 0.72 |
| a=0.00001 | 1.00 | 1.00 | 1.00 | 0.73 | 0.74 | 0.72 |
| a=0.0001 | 1.00 | 1.00 | 1.00 | 0.73 | 0.74 | 0.72 |
| a=0.001 | 1.00 | 1.00 | 1.00 | 0.73 | 0.74 | 0.72 |
| a=0.01 | 1.00 | 1.00 | 1.00 | 0.73 | 0.74 | 0.72 |
| a=0.1 | 1.00 | 1.00 | 1.00 | 0.76 | 0.73 | 0.72 |
| a=1 | 1.00 | 1.00 | 1.00 | 0.70 | 0.73 | 0.70 |
| a=10 | 1.00 | 1.00 | 1.00 | 0.68 | 0.71 | 0.68 |
| a=100 | 0.81 | 0.83 | 0.81 | 0.62 | 0.64 | 0.62 |



For the nn structure of (4800, 85, 324, 14), the alpha values from 0.00001 to 0.1 all gave the same training and testing accuracy and it was the best. The training accuracy was 100% while the testing accuracy was 74%. The rest of the alpha values had worse testing accuracy compared to the range of alpha values provided above.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| (4800,85,324,14)  tanh | Training | | | Testing | | |
|  | Accuracy | Weighted Average Precision | Weighted Average recall | Accuracy | Weighted Average Precision | Weighted Average recall |
| a=0 | 1.00 | 1.00 | 1.00 | 0.75 | 0.77 | 0.75 |
| a=0.00001 | 1.00 | 1.00 | 1.00 | 0.75 | 0.77 | 0.75 |
| a=0.0001 | 1.00 | 1.00 | 1.00 | 0.75 | 0.77 | 0.75 |
| a=0.001 | 1.00 | 1.00 | 1.00 | 0.75 | 0.77 | 0.75 |
| a=0.01 | 1.00 | 1.00 | 1.00 | 0.75 | 0.77 | 0.75 |
| a=0.1 | 1.00 | 1.00 | 1.00 | 0.74 | 0.76 | 0.74 |
| a=1 | 0.99 | 0.99 | 0.99 | 0.74 | 0.76 | 0.74 |
| a=10 | 1.00 | 1.00 | 1.00 | 0.69 | 0.70 | 0.69 |
| a=100 | 0.17 | 0.06 | 0.17 | 0.15 | 0.05 | 0.15 |



**Result**

The best overall model based on training and testing accuracy was: neural networks with logistic/sigmoid activation, nn structure of (4800, 240, 14), and alpha value of 10. The model had 97% training accuracy with 79% testing accuracy which was the highest compared to all other models.

The best model for each method was:

For logistic regression:

* No transformation, L2 regularization with c = 0.001 (λ=1000)
* Result: 73% testing accuracy with 77% weighted average precision

For SVM:

* RBF kernel with c =10 (λ=0.1)
* Result: 75% testing accuracy with 77% weighted average precision

For neural network:

* ReLU activation, NN structure (4800, 85, 324, 14), alpha = 0.1.
* Result: 76% testing accuracy with 78% weighted average precision.

The neural network was the best overall.

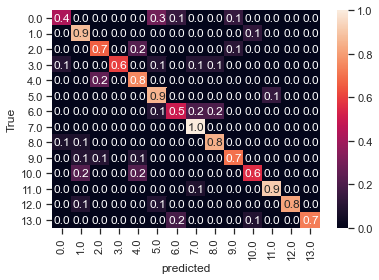
**Conclusion**

In conclusion, the best method for classifying Pokemons from the given dataset was using neural networks with one hidden layer and using logistic/sigmoid activation function with alpha value of 10. Other machine learning methods and models gave really close testing accuracy, however, the neural networking testing accuracy of 79% was best of them all. Most of the testing accuracies with 100% training accuracy were around 70-75%.

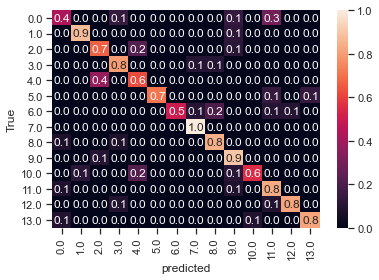
Because of the difference between training and testing accuracy in all models, there were concerns of incorrectly classifying to only one class, but the heatmap for the best prediction method does not show this. The heatmaps on the following page show that only classes 0, 6, and 10 were being misclassified a significant amount of the time. It’s possible that these pokemon were not distinct enough from each other to be properly classified with the image resolution and number of training samples we used.

The original goal of classifying all 150 first generation Pokemon with 7000 samples was unable to be completed due to the fact that our machines did not have enough space to run such a model. We also believed that if we were able to have higher resolution for the images, which would give it more features to train, the testing accuracy would certainly increase. We would’ve also trained the neural networks with more iterations and more neurons in the hidden layers. With our given machine it would simply take way too long to run the model. Overall, working on this project was a eye opening experience as we had to make multiple tweaks with our data and methods along the way to ensure we can get reasonable results with our model. We also realized that doing image recognition is a lot harder than expected to get a really good testing accuracy. For future works, we would want to use convoluted neural networks to see if that will help with the testing accuracy since we saw multiple image classification using CNN.

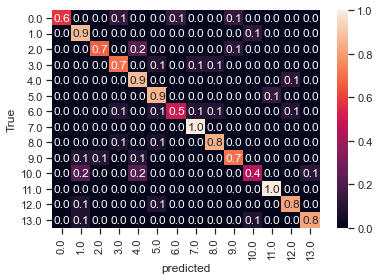
Heatmap for best logistic regression:



Heatmap for best RBF kernel:



Heatmap for best Neural Network:



Citations

Pokemon Dataset

<https://www.kaggle.com/lantian773030/pokemonclassification>

Image Transformations:

<https://towardsdatascience.com/augmentation-for-image-classification-24ffcbc38833>

Heatmap:

<https://medium.com/swlh/logistic-regression-for-image-classification-e15d0ae59ce9>

Programming homework 5 was referenced for the SVM section.