*Introduction*

In our analysis, we investigated the Boston Housing Prices and California Housing datasets using regression techniques, and the Iris Flower and Olivetti Faces datasets using classification techniques. We created linear regression, polynomial regression, and neural network models to predict the Boston and California Housing prices. We created logistic regression, support vector classification (SVC), and neural networks to predict iris flower classes and identify which of forty faces was being displayed of the Olivetti faces.

*Small Regression: Boston Housing*

1. Dataset

The Boston Housing dataset comes with 13 attributes such

as crime rate, distance from employment centers, and average number of rooms per residence. The target variable is the median value of the homes in that block for which the data was gathered. The target value is measured in $1000s and ranges from 5 to 50. As the target values are continuous this is a regression problem. Early analysis of the data showed high correlation between the average number of rooms and the percentage of the population that is lower status with the median values. Other attributes showed less correlation, but none were low enough that we considered removing them.

1. Linear Regression

Our first attempt to make a predictive model came in the

form of the simple linear regression model. For this model, and all subsequent models for this dataset, we tracked the mean squared error (mse) and r-squared (r2) score as a means of evaluation. We split the data into training and testing data and trained the model as expected. The results were a model that put up a mse of 33.45 and r2 score of 0.59. The model seemed to do a fairly good job predicting the value of houses in the $15000 - $35000 range but struggled heavily on the most expensive houses.

1. Polynomial Regression

With the linear regression model lacking

somewhat, we decided to try using higher-order polynomial regression models. This was done by generating modified data that includes new attributes that represent the attributes up to a specified power. First, we tried using a model that included degree 2 (squared) data. This model outperformed the linear regression, putting up a mse of 25.22 ad r2 score of 0.69. We also tried a degree 3 model, however it performed poorly, putting up a mse of 1050370.42 and r2 score of -12898.31. Out of curiosity, we tested the degree 3 model on its own training data where it put up a mse of 0.0000048 showing it was clearly overfitting.

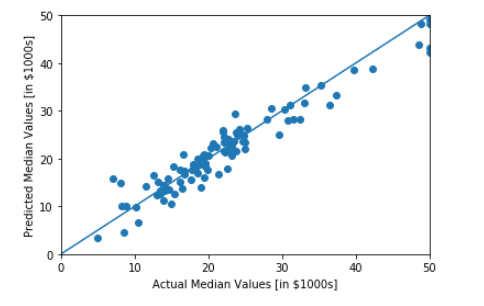
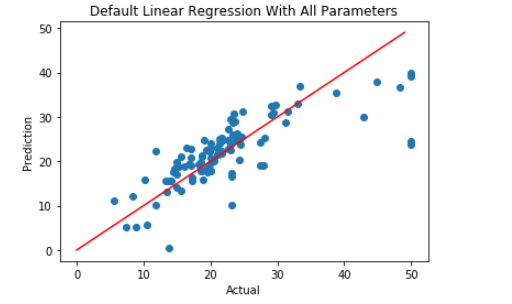
1. Neural Networks

The last type of model we attempted to create was a

neural network model. We tried many different setups in terms of number of hidden layers and neurons per layer; even messing with the activation functions and optimizer on a few equations. In the end, we found that a model with 2 hidden layers, both using the ReLU activation function with 50 and 2 neurons, respectively, worked quite well. The model managed a mse of 14.43 and r2 score of 0.86, outdoing the regression models from earlier. However, we felt there was still room for improvement so we tried two different adjustments: data normalization and early stopping. Data normalization noticeably helped the model’s score. Our new model (now with 64 and 2 neurons in the hidden layers), having been trained on normalized data, was able to post a respectable MSE of 7.76 and r2 score of 0.92, which was the best we had seen so far. Unfortunately, early-stopping did not improve the scores of the neural networks, regardless of whether normalized data was used.

1. Analysis

We saw MSE scores of 33.45, 25.22 and 7.76 for the linear regression, degree 2 polynomial regression, and best neural network respectively. While all the models were fairly-accurate predicting housing values for houses around the $15000-$35000 range (where the majority of the data points resided) there was significant improvement seen on edge cases with the more complex models, especially on the most expensive houses in the dataset. This is well depicted by looking at the prediction vs. actual graphs of our linear regression (*Fig. 1*) and neural network (*Fig. 2*) models.



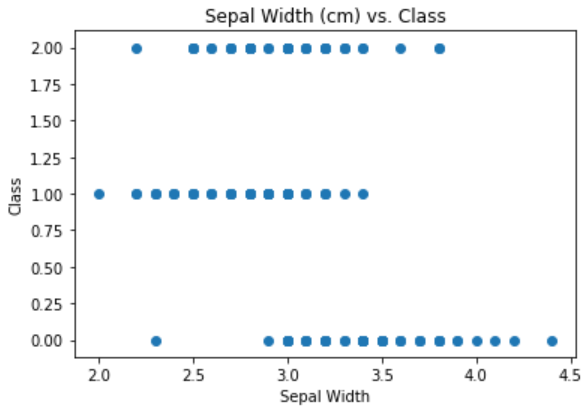
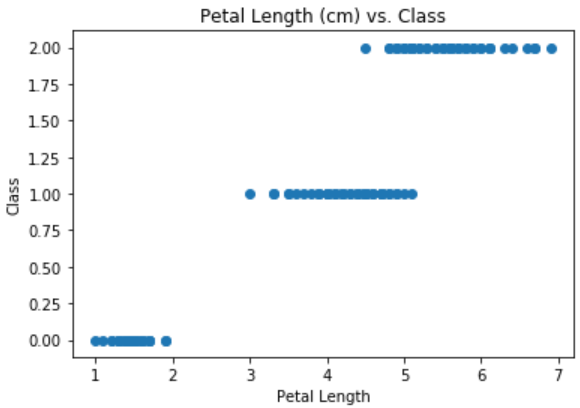
*Fig. 1 Fig. 2*

Overall, we are happy with what the final neural network model was able to accomplish as a MSE of 7.76 feels quite low given the range of the data.

*Small Classification: Iris*

1. Dataset

The problem for this data set was classifying iris flowers into three classes: Setosa, Versicolour, and Virginica. We were given 150 data points to work with, and we classified these data points based on the data sets’ four attributes: petal length, petal width, sepal length, and sepal width. From Figure 3 shown below we can see that the data is already somewhat clustered based on the classes and petal length. In contrast, sepal width is not as correlated with classification as the other attributes but it still contributes to the learning of the models.



*Fig. 3. Petal Length and Sepal Width*

1. Logistic Regression

The first model we used to classify the data was a logistic regression model. After training the model based on the sample data we achieved an accuracy score of 1.0 and F1 score of 1.0. A perfect score seemed too good to be true but we believe a simple logistic regression was able to get a perfect score so easily because the data was already correlated pretty heavily. After running the cross-validation on the same data the accuracy and F1 score were both lowered to 0.973. Although not perfect, this was still a very good score that we were happy with.

1. Support Vector Classification

The second model we tried out was a SVC model. Initially after training the model on the testing data, we found the SVC also achieved an accuracy and F1 score of 1.0. We knew from looking at the data that the simple models would do pretty well, but getting a perfect score on both the logistic regression and SVC was surprising. After running the cross-validation for this model the accuracy and F1 score we got was 0.980. This is slightly better than the logistic regression, so we were happy with the result.

1. Neural Network

After seeing the high accuracy scores from the logistic regression and SVM, we decided to test the data using a neural network. With such simple data we expected the neural network to perform very well. After training and testing the model, we ended up with a cross-validation accuracy and F1 score of 0.966. We ran the neural network many times with different configurations to try to get a better score, but this is the average result we got.

1. Analysis

For each of the machine learning models, we were happy with the resulting accuracy score. Accounting for cross-validation, the accuracy scores were: 0.973 for the logistic regression, 0.980 for the support vector classifier, and 0.966 for the neural network. The neural network performing worse than the simpler models was initially surprising, but after looking at the data it seemed like the simpler models had an easier time classifying the data since the data was already clustered.

*Large Regression: California Housing*

1. Dataset

For the large dataset for the regression models, we used the California Housing dataset, which is used to predict the values for housing in California. The target range for the data was between 0.15 and 0.5, and contains eight attributes. The two attributes with the highest correlation to the house value were median income, which increased as the value increased, and the block population, which increased the lower the value was.

1. Linear Regression

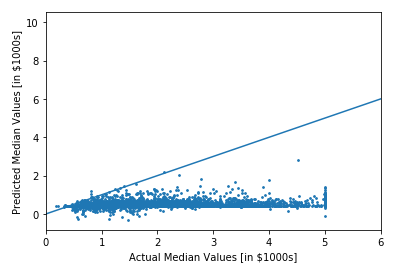
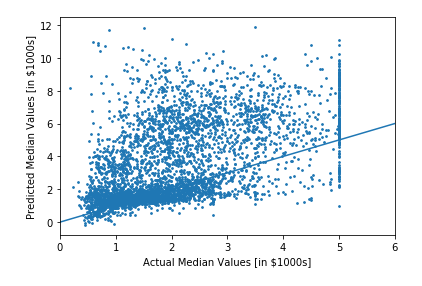
To start, we created a linear regression model of the data, splitting the data into 80% training and 20% testing. After training, the r2 score ended up being a 0.59, with a cross-validation score of around 0.51 and a RMSE score of around 0.74. While the error score is low, considering how low the cross-validation and r2 scores were, it was clear that the error score could be lower for this dataset.

1. Polynomial Regression

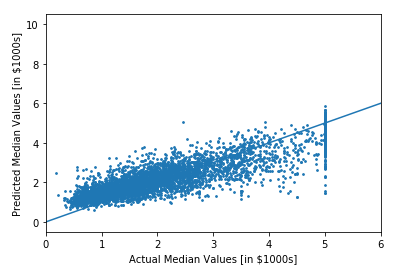
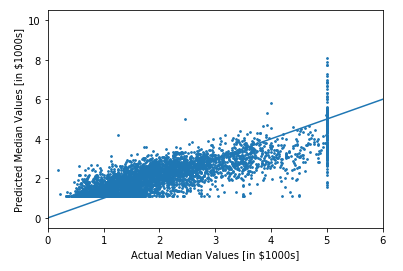
We then decided to model the data using polynomial regression. With a polynomial regression model of degree 2, the r2 score increased to 0.68, but the model began to overfit the data. The cross-validation score, at around -0.43, supported this conclusion. However, this suspicion of overfitting was confirmed when we created a polynomial regression model of degree 3, which returned a cross-validation score of -23,6704.02. So while the accuracy improved as the degree of the polynomial regression increased, the models were not useful since they overfit the data.

1. Neural Network

Finally, we used a neural network to model the dataset. We first started by using normalized data, hoping for the MSE to be lower than 0.53, which was the error we found for the linear regression model. However, this score ended up being much higher than we hoped at 6.44. In order to get the number of epochs correct, an early stopping model was done for the normalized data. This performed much worse than the non-early stopping model, with an MSE of 21.42.

*Fig. 4. Non-Early Stopping model (left) and Early Stopping Model (right)*

We then created a neural network model using the non-normalized data to see if it would perform better. Once again, we hoped for an MSE lower than 0.53, and were ultimately successful with an MSE of 0.49. To see if we could achieve an even lower error, and decided to run an early stopping model with the same non-normalized data. This model ended up with an error of 0.43, which is marginally better than the prior model.

*Fig. 5. Non-Early Stopping model (left) and Early Stopping Model (right)*

1. Analysis

Out of the simpler models used, the linear regression model performed the worst with an MSE of 0.59. However, even though the polynomial regression models were more accurate on the training data, they overfit the data, leading the linear regression model to be the best out of the simpler models. Regarding the neural networks, the first models done with normalized data performed very poorly, the error scores being too high with an error score of 21.42 with early stopping. When we moved onto non-normalized data, the models performed much better, the early stopping model using non-normalized data performing the best. This model, with the error score of 0.43, had the best results out of all the models used on the dataset.

*Large Classification: Olivetti Faces*

1. Dataset

The Olivetti Faces dataset contains a set of 400 images of faces taken between April 1992 and April 1994 at AT&T Laboratories Cambridge. There are 40 distinct subjects, and 10 photos of each subject. The images are stored in grayscale pixel values, with a value of 1 indicating a white pixel and a value of 0 indicating a black pixel. The target for this dataset is an integer 0-39, indicating the identity of the subject in the photo

1. Logistic Regression

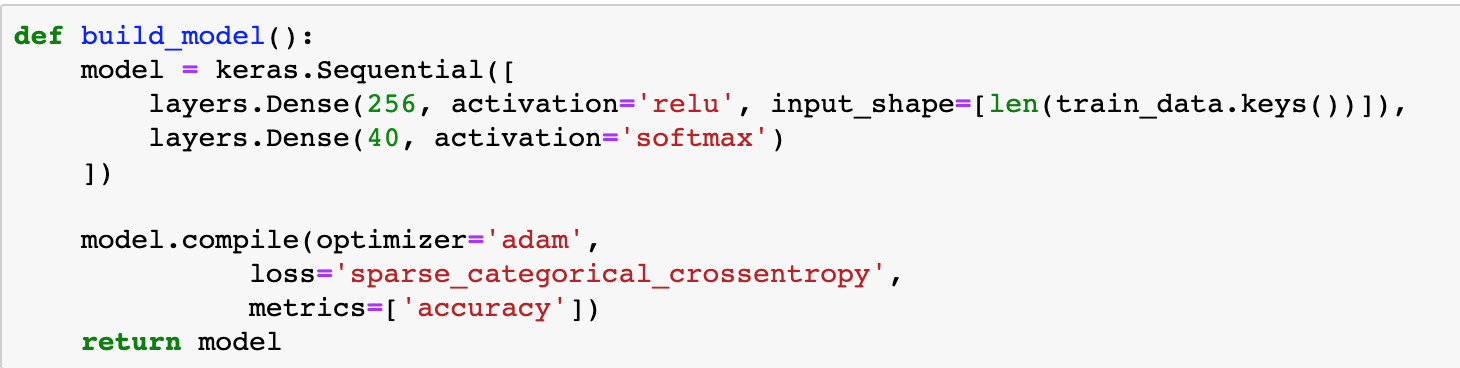
We first used logistic regression to model the data. Following training, we achieved an accuracy score of .963 and F1 score of .952 for the model and a cross-validation accuracy score of .972 and F1 score of .970. Although this model proved to be highly accurate on the dataset, we hoped to achieve an even higher level of accuracy with a different model.

1. Support Vector Classifier (SVC)

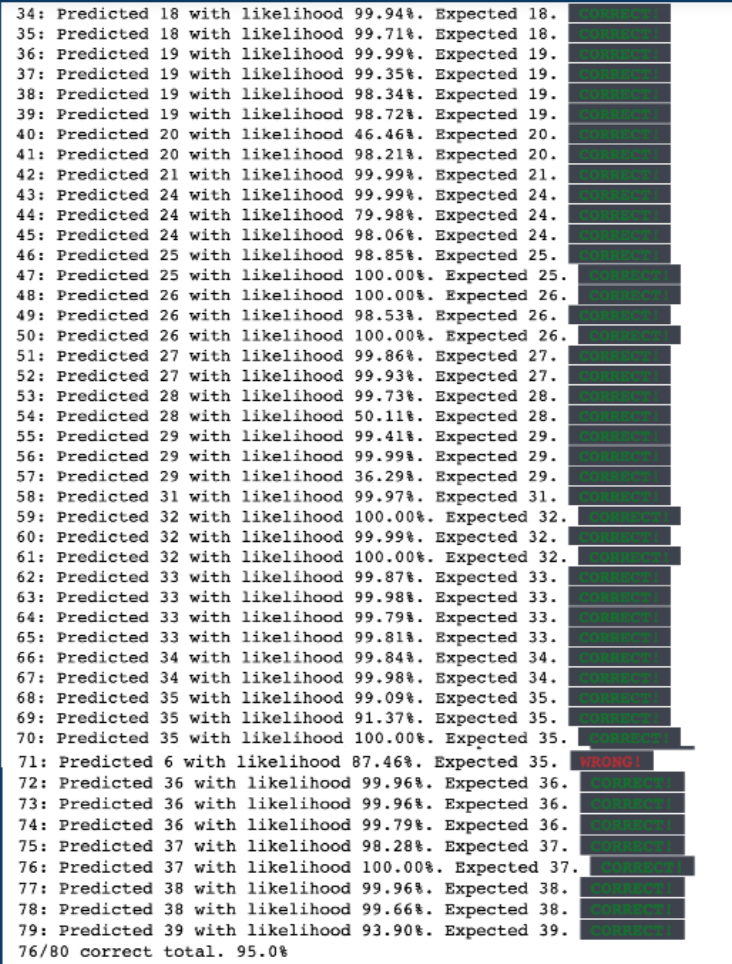
The SVC model did not prove to be a more effective model for this dataset than the previously-used logistic regression model. The SVC model’s accuracy was .825 and F1 score was .865, while the cross-validation scores were more similar to that of the logistic regression with accuracy .943 and F1 score of .939. From these scores, it was evident that the SVC model was not the proper model to use on this dataset.

1. Neural Network

We created a neural network model of this dataset with one hidden layer. The hidden layer was created with 256 neurons and an ReLU activation function. The output layer had 40 neurons (since there are 40 possible outputs) and a Softmax activation function The end-result of the neural network with the dataset was a cross-validation accuracy of .953 and F1 score of .945.



*Fig. 6. Neural network modeling the Olivetti Faces dataset*



*Fig. 7. predicted vs expected for neural network*

1. Analysis

We concluded that the Olivetti Faces dataset was best modeled using a logistic regression model. This conclusion was drawn from comparing the cross-validation scores across the three models we created for the data.

*Future Works*

With more time we would create a convolutional neural network for the Olivetti Faces dataset. The convolutional neural network model is more optimized for image classification than a normal neural network, and therefore we could hopefully attain a higher degree of accuracy than what was achieved with our neural network model. We would also attempt to further improve the neural network for the Iris dataset, since it did not outperform the logistic regression or SVC models. Another strategy we would test out is the use of normalized data on the simpler models for the California and Boston Housing datasets. We tested the neural networks for those datasets using normalized data, but we neglected to do the same with the simpler models.