**Machine Learning Project Report**

*By Dennis Bettels, Davide Varagnolo*

[*dbettels1@gmail.com*](mailto:dbettels1@gmail.com)*,* [*dvaragnolo94@gmail.com*](mailto:dvaragnolo94@gmail.com)

ML course (654AA), Academic Year:2018/19

Date: 13/07/2019

Type of project: B

Abstract

An extensive analysis of various ML tools such as keras and scikit-learn applied to a variety of different models such as Neural Networks, K-Nearest Neighbour and Support Vector Models, along with a fine tuning of hyperparameters using Gridsearch to best fit the models to a dataset describing diabetes occurrence rates in a group of Native Americans. Additionally, a classification is run on Monk’s Problems Dataset to ascertain the quality of the models, and a regression is run on the ‘ML-CUP18’ dataset for the competition.

Introduction

The purpose of this assignment was to evaluate a set of popular tools which are used to make Neural networks, and compare their efficacy in terms of precision, cross-validation score and loss. The tools which were examined for this research were Scikit-learn, Keras and Tensorflow. To evaluate these tools, we used them to create and manipulate a range of Neural Networks; Multi-Layered Perceptrons, Support Vector Models and K-Nearest Neighbour. These models were trained on 2 datasets: one describing the rate of diabetes in a group of native americans and another on the Monk’s problem dataset. The latter was done with the intention of testing how solid our machine learning algorithms were, as the Monk’s dataset is quite useful to determine classification networks quality and tends to give high results.

In order to complete the assignment, some assumptions had to be made. For the SVM, it was assumed that the set of optimal kernels to choose from for the dataset were RBF and sigmoid. This is because the other kernels where too computationally intensive for the hardware we had available. For the MLP, we had to keep only one hidden layer to avoid adding complexity to the model.

Method

The methodology used for the project was as follows: the MLP was made on keras which run on top of Tenserflow as well as Scikit-learn, and the KNN and SVM were made only with Scikit-learn. To import and carry out the pre-processing of the data both numpy and pandas were used across all models and tools. In order to the plot the graphs, matplotlib was used.

While all the models were created separately, the evaluation methods were similar. Once implemented, the hyperparameters were fine-tuned using gridsearch to return the most optimal results.

For each model, the hyperparameter which were tuned were as follows:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| SVM | Kernel  ('rbf','sigmoid') | Penalty Parameter C  (0.25, 0.5, 0.75, 1, 10, 100, 1000, 10000) | Gamma coefficient  Λ  (0.00001,0.0001, 0.001, 0.01, 0.1, 0.5, 1, 2, 3, 'auto') | Decision Function Shape  OVO, OVR | Shrinking Heuristic  True,False | Tol  0.0001,  0.0005,  0.001,  0.005,  0.01,  0.1 | Coef0  0,1,2,3 |
| KNN | n-neighbours  (1, 2, 5, 10, 50, 100) | Weight Function  ('uniform', 'distance') | Algorithm  ('auto', 'ball\_tree', 'kd\_tree', 'brute') | Leaf size  (10, 20, 30, 50, 100) | Power Parameter  P  (1,2,3,5,10) |  |  |
| MLP | Alpha  [0.025,0.05,0.1,0.2,0.4,0.7] | Hidden Layer Size  [8],[14],[16],[24],[28] | Maximum iteration  [64, 96, 112, 128, 160, 256] | Destination function  [‘relu’,  ’tanh’,  ’sigmoid’] | Momentum  [0.1,0.5,0.7,0.8,0.9] | Nesterov’s momentum  [True,False] | Learning rate initiation  [0.001, 0.01, 0.05, 0.1, 0.15,0.2 |
|  | Dropout  [0,0.2,0.5] | Decay  [0,0.01,  0.0001,  0.000005] |  |  |  |  |  |

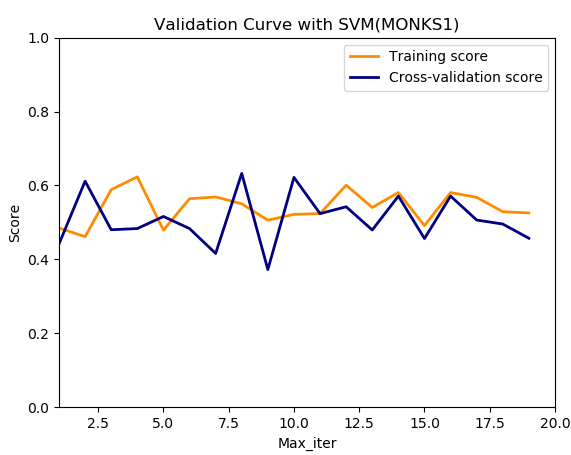
The primary evaluation method which was used was the final accuracy of the model and the mean squared error. This was done with both the Indians dataset and MONK dataset.

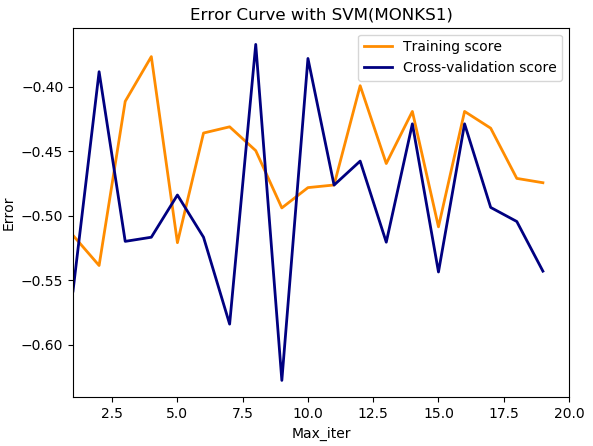
We noticed that our dataset was lacking several values for patients in the Indians dataset and replaced those with a value of 0. This adversely affected our final results, and as a consequence we resorted to pre-processing the data by removing all zero results and replacing them with the overall median of the data. This led to significant improvements, especially in the case of the MLP on the Indians dataset. Naturally, this preprocessing has no value on the MONK dataset in which the 0 are deliberate, therefore it was omitted for that dataset.

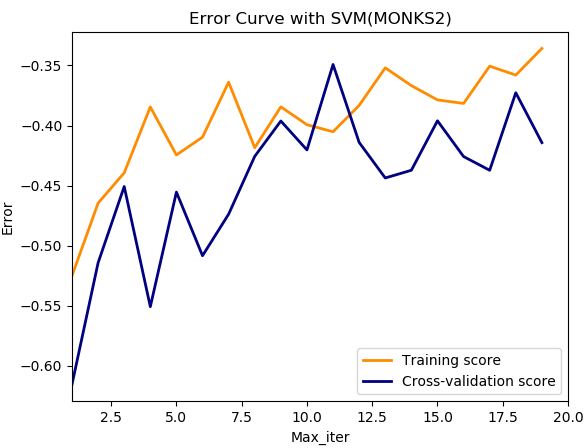
The gridsearch for the multilayer perceptron was performed in Scikit-learn. To test the best chosen hyperparameters, the classifier of Scikit-learn was used (MLPClassifier). The model was then translated into a sequential Keras model and we applied an ulterior hyperparameter fine-tuning not present on the Scikit-learn classifier, which was the dropout and decay. Both of these values were set to 0.0. Finally, to plot the results for the MLP the learning curve was utilised in Scikit-Learn.

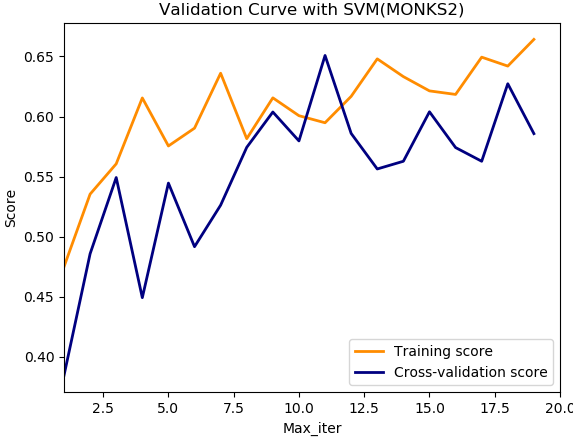
**3. Experiments**

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Hyperparameters** | **MSE** | **Accuracy(TR/TS)** |
| Monk 1(MLP) | activation = 'tanh'  learn\_rate = 0.2  neurons = 8  dropout = 0.0  momentum = 0.7  nesterov = True  epochs = 1024  batch\_size = 64  alpha= 0.001 | TR: 0.04251  TS: 0.05389 | TR: 100%  TS: 98.8% |
| Monk 1 (SVM) | {'C': 10000, 'coef0': 0, 'decision\_function\_shape': 'ovo', 'gamma': 0.001, 'kernel': 'rbf', 'shrinking': True, 'tol': 0.1} | TR: 0.14516  TS: 0.18056 | TR: 85%  TS: 82% |
| Monk 1 (KNN) | {'algorithm': 'auto', 'leaf\_size': 30, 'metric': 'euclidean', 'n\_neighbors': 6, 'p': 1, 'weights': 'uniform'} | TR: 0.12903  TS: 0.19444 | TR:87%  TS:81% |
| Monk 2(MLP) | activation = 'tanh'  learn\_rate = 0.2  neurons = 8  dropout = 0.0  momentum = 0.7  nesterov = True  epochs = 1024  batch\_size = 256  alpha= 0.001 | TR: 0.18411  TS: 0.20616 | TR: 76.9%  TS:71.3% |
| Monk 2 (SVM) | {'C': 1000, 'coef0': 0, 'decision\_function\_shape': 'ovo', 'gamma': 'auto', 'kernel': 'rbf', 'shrinking': True, 'tol': 0.0001} | TR:0.0  TS: 0.17824 | TR:100%  TS:82% |
| Monk 2 (KNN) | {'algorithm': 'auto', 'leaf\_size': 1, 'metric': 'euclidean', 'n\_neighbors': 4, 'p': 1, 'weights': 'uniform'} | TR: 0.26627  TS: 0.32639 | TR:73%  TS:76% |
| Monk 3(MLP) | activation = 'tanh'  learn\_rate = 0.2  neurons = 13  dropout = 0.0  momentum = 0.7  nesterov = True  epochs = 1024  batch\_size = 128  alpha= 0.001 | TR: 0.06792  TS: 0.08541 | TR:95%  TS:93% |
| Monk 3 (SVM) | {'C': 0.5, 'coef0': 0, 'decision\_function\_shape': 'ovo', 'gamma': 'auto', 'kernel': 'rbf', 'shrinking': True, 'tol': 0.1} | TR: 0.07377  TS: 0.0625 | TR:93%  TS:94% |
| Monk 3 (KNN) | {'algorithm': 'ball\_tree', 'leaf\_size': 2, 'metric': 'manhattan', 'n\_neighbors': 5, 'p': 1, 'weights': 'uniform'} | TR: 0.08197  TS: 0. 09259 | TR:92%  TS:91% |
| Indians (SVM) | {'C': 10, 'coef0': 0, 'decision\_function\_shape': 'ovo', 'gamma': 1e-05, 'kernel': 'rbf', 'shrinking': True, 'tol': 0.0001} | TR: 0.23127  TS: 0.17532 | TR: 78%  TS:72% |
| Indians(KNN) | {'algorithm': 'auto', 'leaf\_size': 1, 'metric': 'manhattan', 'n\_neighbors': 18, 'p': 1, 'weights': 'uniform'} | TR: 0.23836  TS: 0.22511 | TR: 76%  TS: 77% |
| Indians(MLP) | activation = 'tanh'  learn\_rate = 0.15  neurons = 16  dropout = 0.0  momentum = 0.7  nesterov = True  epochs = 512  batch\_size = 307  alpha = 0.001 | TR: 0.15554  TS: 0.16274 | TR:78%  TS:76% |

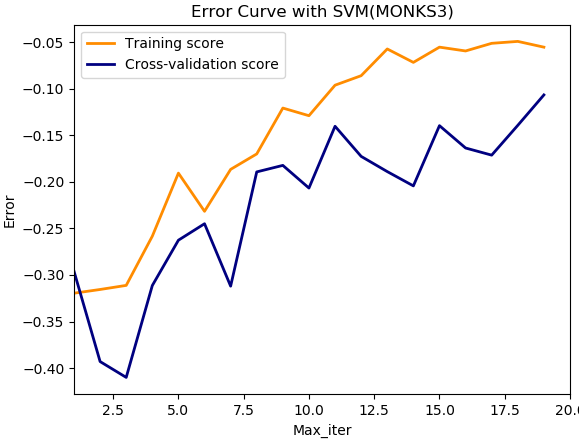
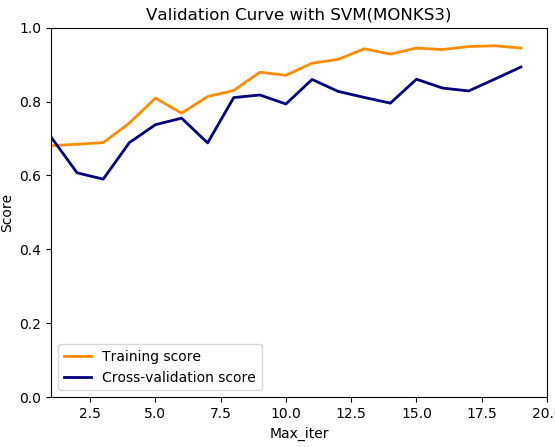
**Figure 1.** Plot of accuracy and MSE of the Monk’s 1 on the SVM



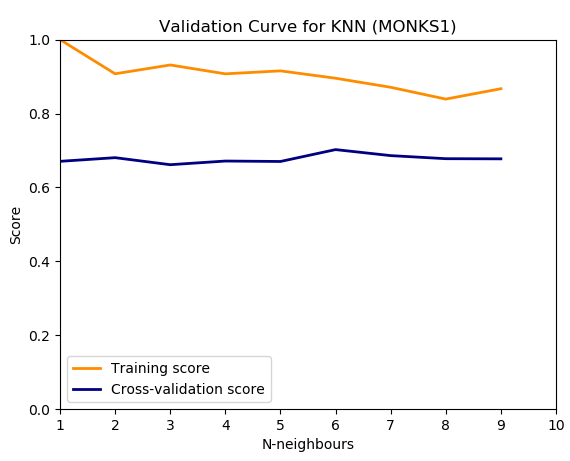
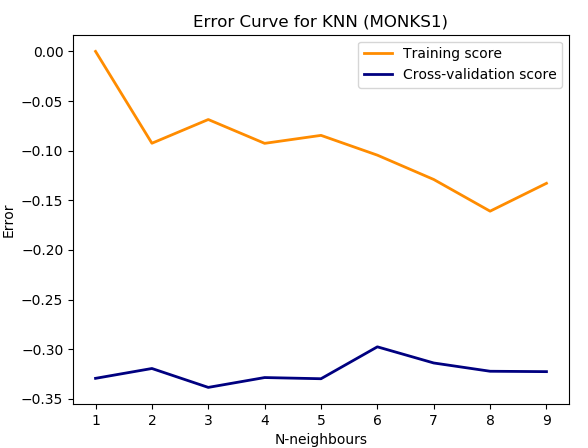
**Figure 2.**  Plot of accuracy and MSE of the Monk’s 2 on the SVM



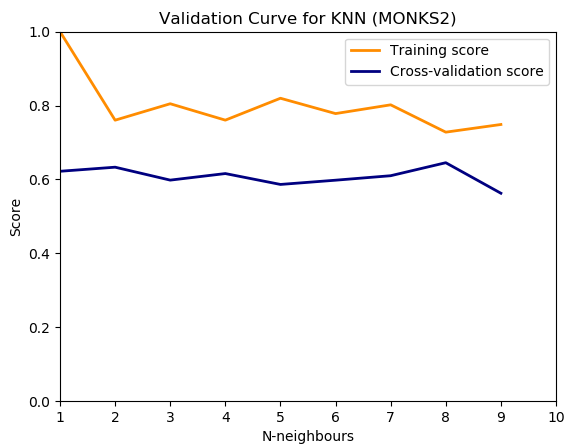
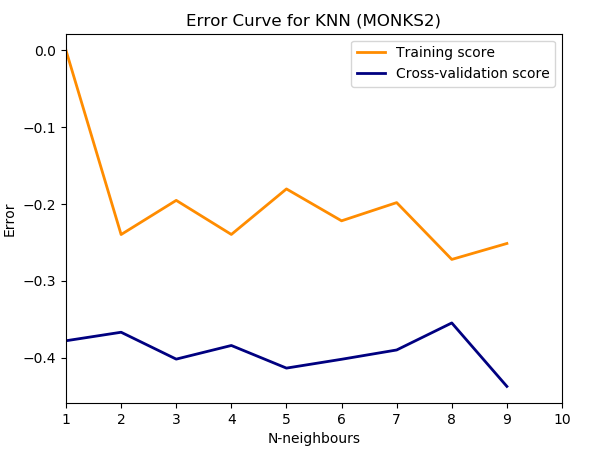
**Figure 3.** Plot of accuracy and MSE of the Monk’s 3 on the SVM

****

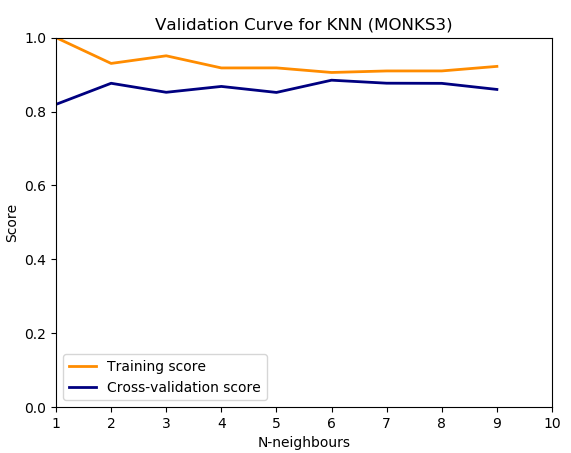
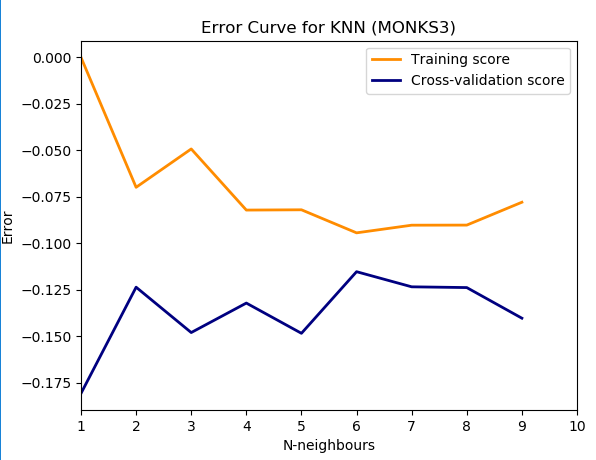
**Figure 4 .** Plot of accuracy and MSE of the Monk’s 1 on the KNN

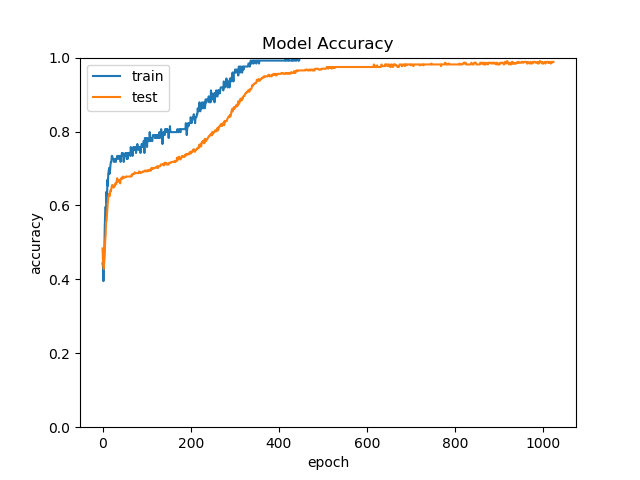
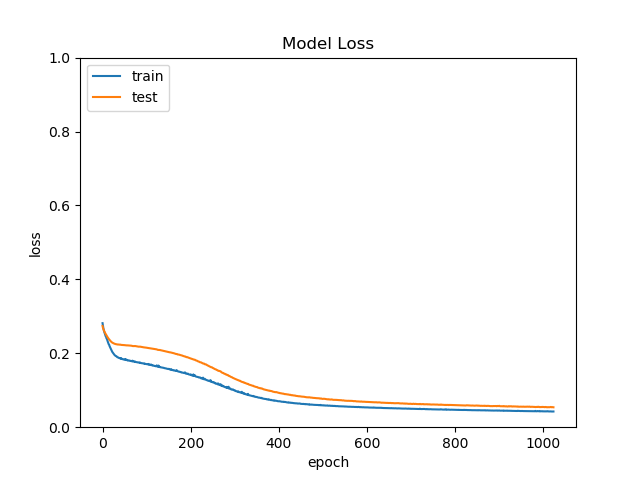
****

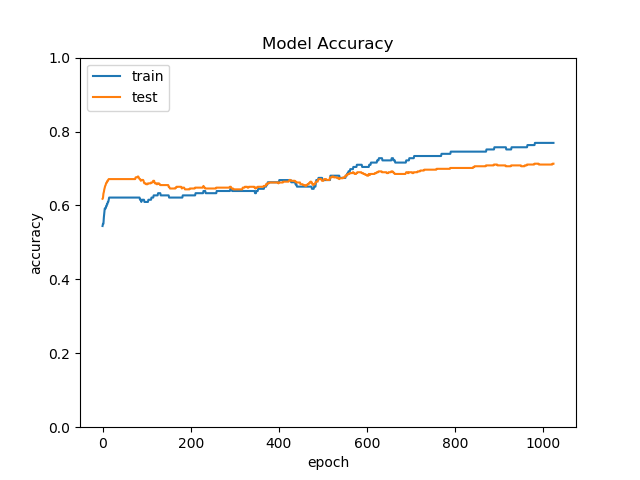
**Figure 5.** Plot of accuracy and MSE of the Monk’s 2 on the KNN

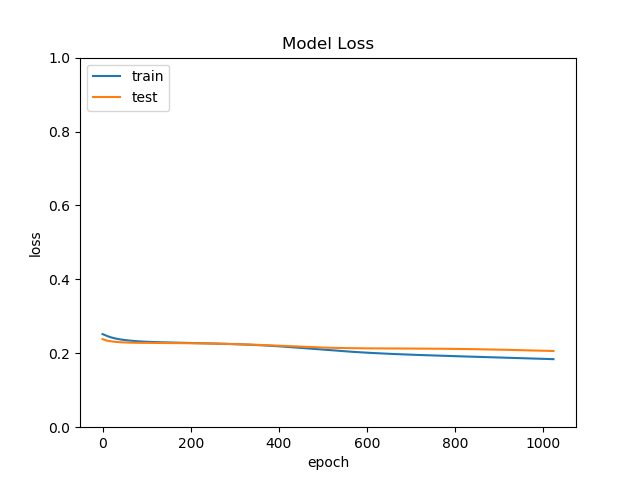


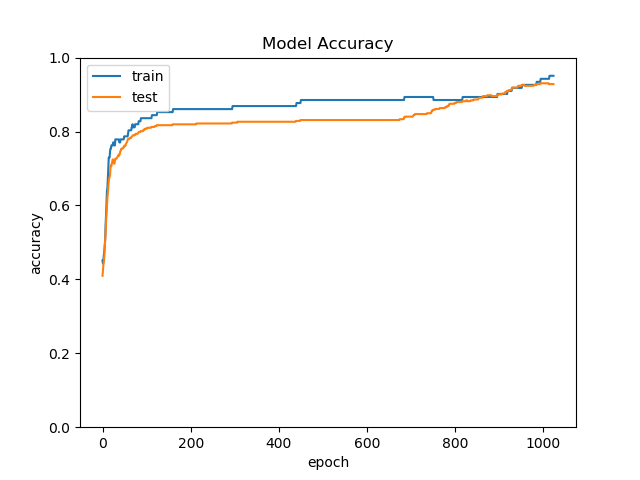
**Figure 6.** Plot of accuracy and MSE of the Monk’s 3 on the KNN

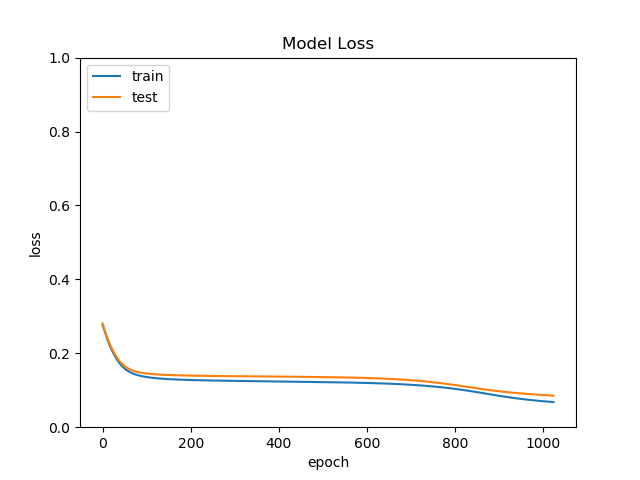


**Figure 7** Plot of accuracy and Loss of the Monk’s 1 on the MLP

**Figure 8** Plot of accuracy and Loss of the Monk’s 2 on the MLP

****

**Figure 9** Plot of accuracy and Loss of the Monk’s 3 on the MLP



**CUP Results**

Before, GridSearch (kfold=5) to choose the model:

First Run:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Hidden Layer Size | Alpha | Batch Size | Learning rate init | momentum |
| [8],[16],[24],[32] | [0.0, 0.00001, 0.001] | [32,128,512] | [0.01,0.1] | [0.7,0.9] |

The best hyperparameters that were found for this were alpha = 0.001, batch size = 32, hidden layer sizes = 512, learning rate init = 0.01 and momentum = 0.7

Second Run:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Hidden Layer Size | Alpha | Batch Size | Learning rate init | momentum |
| [14],[16],[18],[20] | [0.0005, 0.001, 0.0015] | [256,512,809] | [0.001,0.01] | [0.7] |

The best hyperparameters that were found for this were alpha = 0.0005, batch size = 809, hidden layer sizes = 14, learning rate init = 0.01 and momentum = 0.7

Third Run:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Hidden Layer Size | Alpha | Batch Size | Learning rate init | momentum |
| [14],[15],[16] | [0.0, 0.0001, 0.0005] | [809] | [0.01] | [0.7] |

The best hyperparameters that were found for this were alpha = 0.0001, batch size = 809, hidden layer sizes = 14, learning rate init = 0.01 and momentum = 0.7

The final value was calculated using the third and final model and resulted in a training loss of 2.9683 and a testing loss of 3.5121.

**Conclusion**

Over the course of our research, we attempted to use various models and tools to fit to the Indian dataset. This was a binary classification problem and our models were trained to correctly predict whether, based on certain criteria, a person would suffer from diabetes or not using data it had not seen before in a test set. We found out that the most efficient model we had at our disposal was the k-Nearest Neighbours Model(knn) implemented using Scikit learn, with a prediction accuracy of 77%. To test the general efficacy of our models, we also tried a series of tests on the MONK’s dataset, and concluded with testing the Multi-Layered Perceptron on the ML Cup dataset, resulting in a final testing loss of 3.5121. As we can see in the results, we had some issues with the MONK 2 dataset which tended to return lower results than predicted.

**ACKNOWLEDGEMENTS**

I/we agree to the disclosure and publication of my name, and of the results with preliminary and final ranking.

**References**

[1]: A. Micheli: Lecture slides for Machine Learning 2018/2019

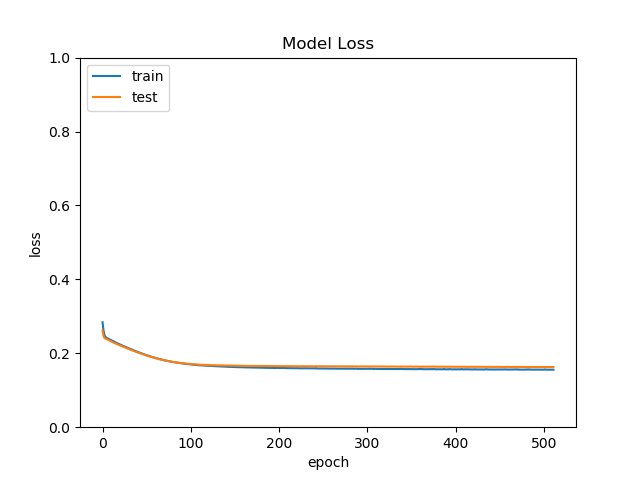
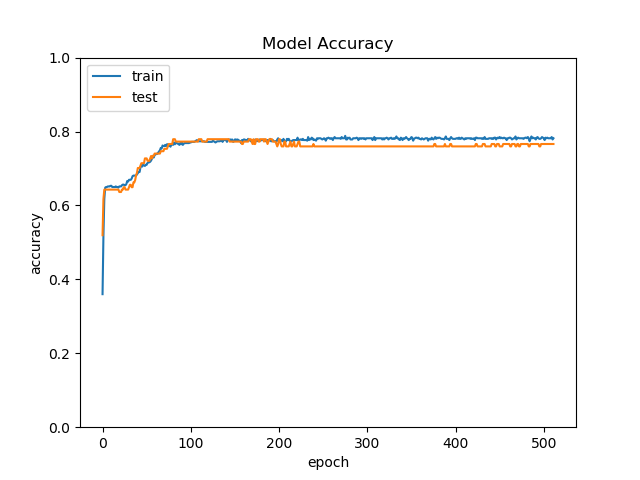
[2]: <https://www.andreagrandi.it/2018/04/14/machine-learning-pima-indians-diabetes/>

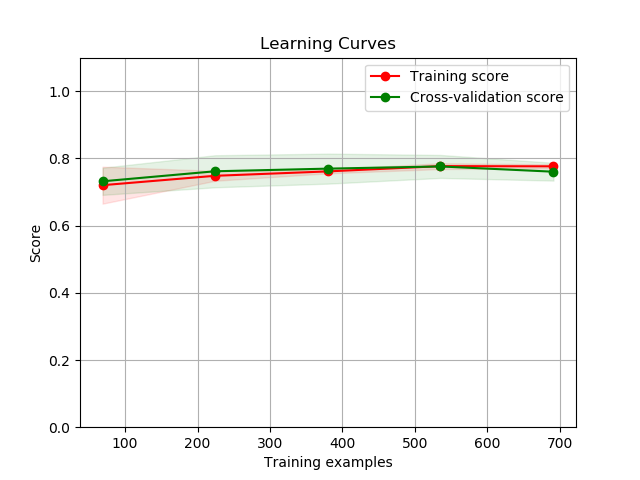
[3]: Keras: https://keras.io/

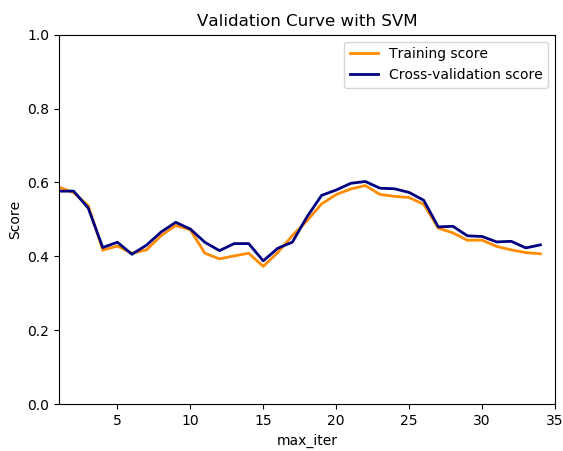
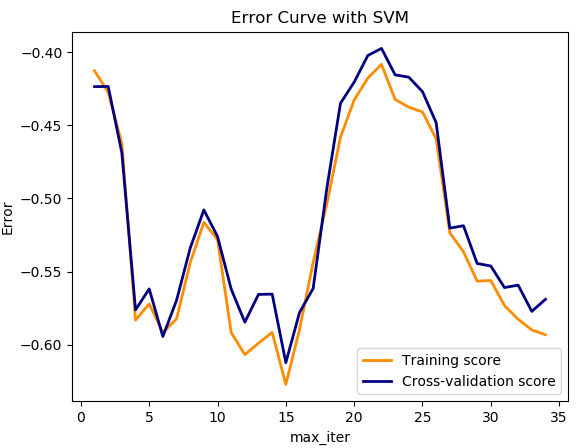
[4]: Scikit-Learn: https://scikit-learn.org/stable/user\_guide.html

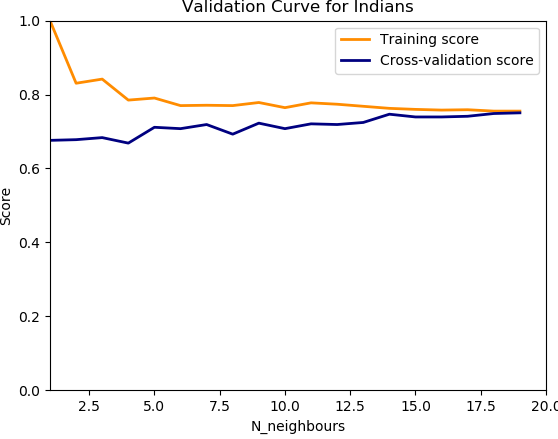
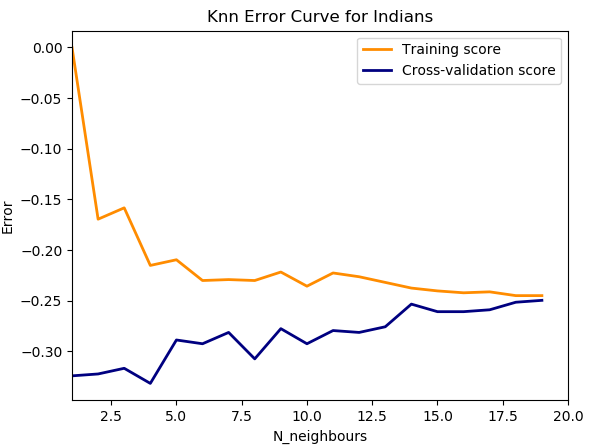
**Appendix**

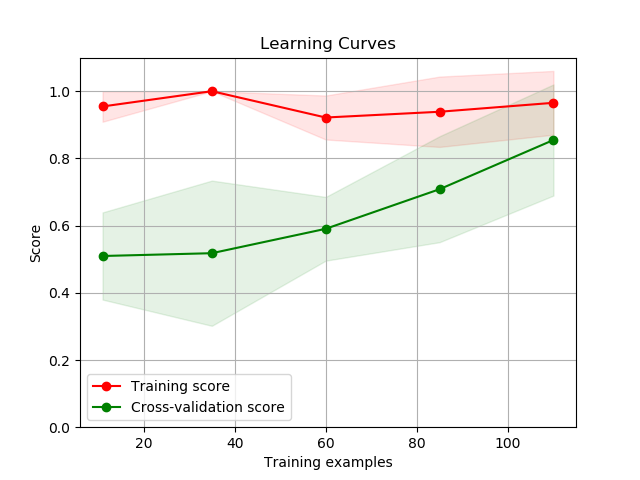
Figure 10. Accuracy and Error of Indians dataset on MLP with Keras

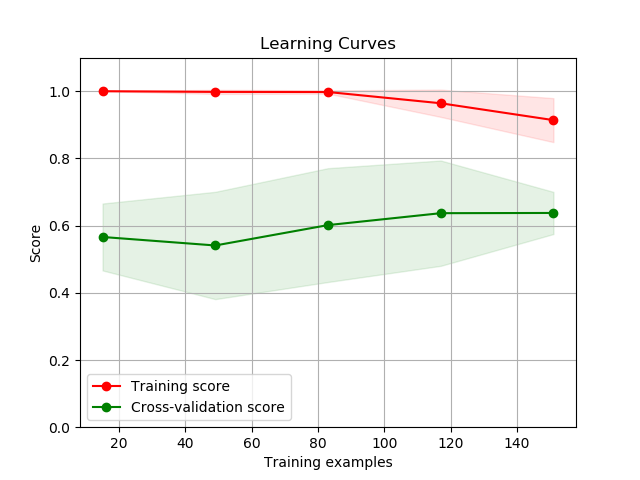
****

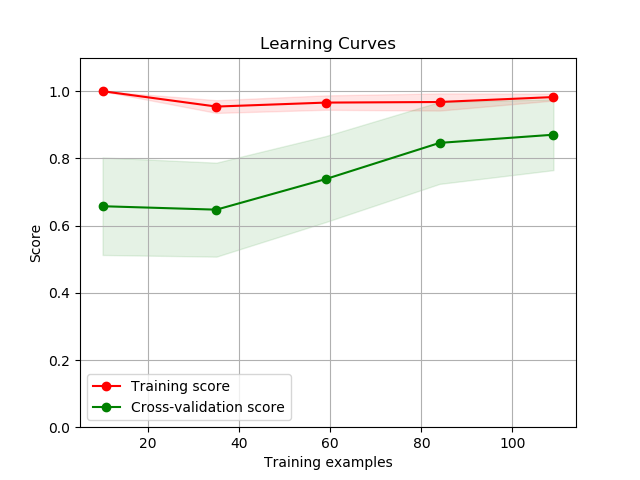
Figure 11 Accuracy and Error of Indians dataset on MLP with Scikit-learn

Figure 12. SVM on the Indians dataset with accuracy and Error(Scikit)

Figure 13. KNN on the Indians dataset with accuracy error (Scikit)

Figure 14 and 15. Learning Curve for Monks1 and 2 MLP(Scikit)



Figure 16. Learning Curve for Monks3 MLP(Scikit)