**Machine Learning Project Report**

*By Dennis Bettels*

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

ML course (654AA), Academic Year:2019/20

Date:

Type of project: B

Abstract

An extensive analysis of ML tools available in scikit-learn applied to a variety of different models such as Neural Networks and K-Nearest Neighbours, along with a fine tuning of hyperparameters to best fit the models to the Students Dataset[[1]](#footnote-1) , a dataset predicting the final grades obtained by students in the subject of Mathematics based on 32 parameters. Additionally, a classification is run on Monk’s Problems Dataset to ascertain the quality of the models, and a regression is run on the official ML-CUP dataset using the best model.

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. Various tools within the Scikit-learn library were used to create multiple models and compare their performances. The models used during this project were as follows: Support Vector Models, K-Nearest Neighbour and Decision Trees. These models were trained on 2 datasets: one a multivariate classification predicting the final maths grade of a student based on a variety of factors and another on MONKS’s problem dataset, a binary classification problem. 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 show high improvements when calibrated properly.

The Support Vector Regressor turned out to be too slow with the ‘poly’ and ‘linear’ kernel to include those two in a Gridsearch and complete the task in a reasonable amount of time. A random search was therefore performed to address this issue, hoping it would find the best possible hyperparameters. This put it at a disadvantage when compared to the KNR, which was capable of undergoing a complete gridsearch.

Method

The methodology used for the project was as follows: the KNN, SVM and Decision Trees were all made with their respective Scikit-learn packages, as well as the regressors which were tested (SVR and KNR).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', ‘poly’,’linear’)  Specifies the type of kernel to be used with the algorithm  Note:poly and linear not used for regressor | Regularization parameter C | Gamma coefficient  Λ  Kernel coefficient for rbf, poly and sigmoid kernel types | Decision Function Shape  OVO, OVR  One vs rest (ovr) decision function or one v one. Ignored in binary classification | Shrinking Heuristic  True,False  Whether to use the shrinking heuristic | Tol  Tolerance for stopping criterion | Coef0  Independent term in kernel function. Only used in poly and sigmoid |
| KNN | n-neighbours  number of neighbours to be used | Weight Function  ('uniform', 'distance')  Weight function. Either uniform weights will be used or weight points by inverse of their distance | Algorithm  ('ball\_tree', 'kd\_tree', 'brute')  Type of algorithm to compute nearest neighbours | Leaf size  Leaf size passed to ball tree and kd\_tree | Power Parameter  P | Metric  (minkowski, Euclidean, manhattan, chebyshev)  Distance metric to calculate the tree |  |
| DTC | Criterion  ['gini', 'entropy']  The function to measure the quality of a split. | Splitter  ['best','random']  The strategy used to choose the split at each node | Max depth  [None,10,100,1000]  The maximum depth of the tree | Min samples split  [0.1,0.2,0.3,  0.4,0.5,0.7,  0.9,2,  10,50,100]  The minimum number of samples required to split an internal node | Min samples leaf  [1,5,10,20,30]  The minimum number of samples required to be at a leaf node | Min\_weight\_  fraction\_leaf  [0,0.1,0.2,0.3,  0.4,0.5]  The minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node | Max features  [None, 'auto',  'sqrt','log2']  No. of features to consider when searching for best split |
|  | Max leaf nodes  [None,5,10,100]  Grows a tree in ‘best first’ fashion with maximum number of leaf nodes | Min impurity  Decrease  [0,0.5,1.0] | Class weight  [None, 'balanced']  Weights associated with classes | Random state  [1,2,3,4,5]  Fixes the randomness of estimator. |  |  |  |

Table 1: hyperparameters of the models

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 Students dataset and MONK dataset.

The gridsearch was performed by running every available value where fixed text values were required. For numeric values, orders of magnitude were used at first and then fine tuned over multiple gridsearches. One issue this exhaustive approach created is that dead-weight parameters (such as coef0 for rbf) were iterated over when they had no bearing on the final result, which increased computational time(in some cases significantly). This also had an effect on the randomized search.

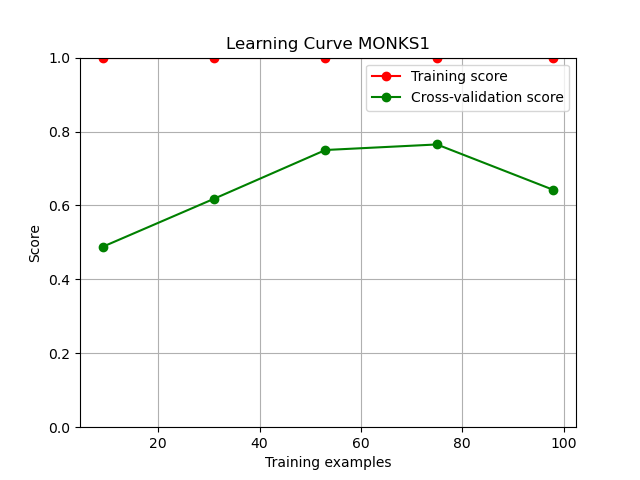
Various graphs were plotted in **Section 3** covering the effect that certain hyperparameters had on the final accuracy score of the model.

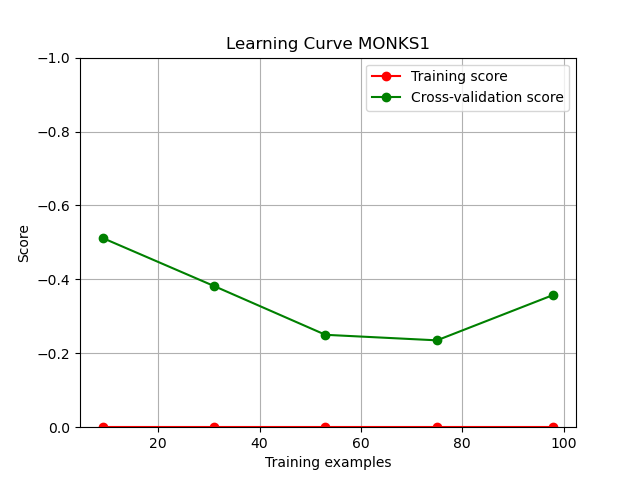
Throughout all models, the popular cross-validation value of 5 was used as it is

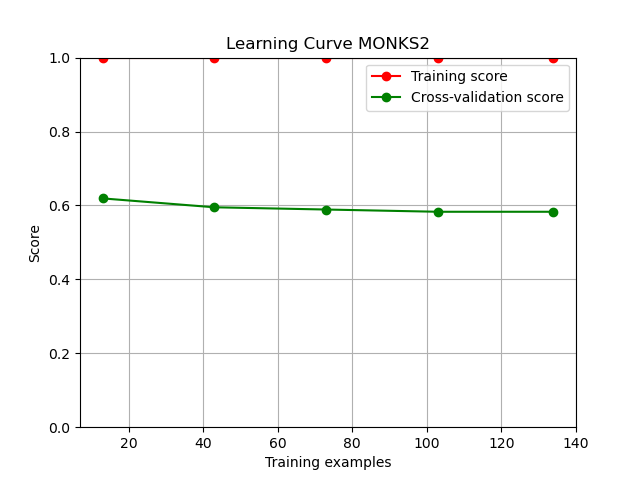
**3. Experiments**

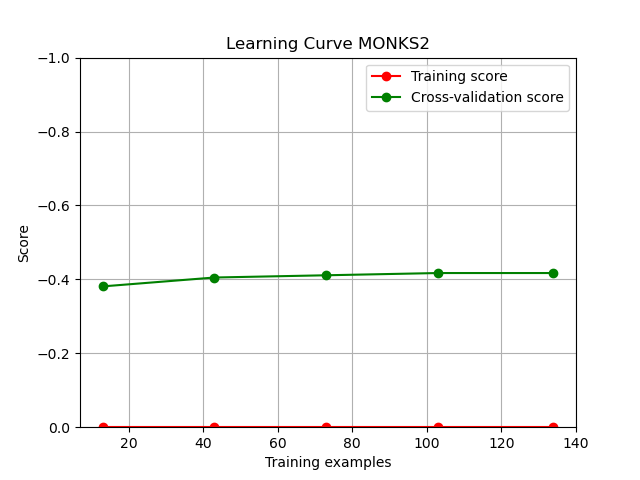
|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Hyperparameters** | **MSE** | **Accuracy(TR/TS)** |
| Monk 1 (SVM) | {'C': 35, 'coef0': 0, 'decision\_function\_shape': 'ovo', 'gamma':’auto’, 'kernel': 'poly', 'shrinking': True, 'tol':0.001} | TR: 0.0  TS: 0.07889 | TR: 100%  TS: 92.11% |
| Monk 1 (KNN) | {'algorithm': 'brute', 'leaf\_size': 7, 'metric': 'chebyshev', 'n\_neighbors': 28, 'p': 1, 'weights': 'distance'} | TR: 0.0  TS: 0.19954 | TR:100%  TS:80.05% |
| Monk 1 (DTC) | Monks1 - (splitter= 'best', random\_state= 3, min\_weight\_fraction\_leaf= 0.1, min\_samples\_split= 2, min\_samples\_leaf= 10, min\_impurity\_decrease= 0, max\_leaf\_nodes= 5, max\_features= None, max\_depth= None, criterion= 'gini', class\_weight= None) | TR:0.14634  TS:0.25058 | TR:85.37%  TS:74.94 |
| Monk 2 (SVM) | {'C': 1000, 'coef0': 0, 'decision\_function\_shape': 'ovo', 'gamma': 'auto', 'kernel': 'rbf', 'shrinking': True, 'tol': 0.0001} | TR:0.0  TS: 0.17865 | TR:100%  TS:85.15% |
| Monk 2 (KNN) | {'algorithm': 'brute', 'leaf\_size': 30, 'metric': ' mahalanobis ', 'n\_neighbors': 1, 'p': 1, 'weights': 'uniform'} | TR: 0.0  TS: 0.17865 | TR:100%  TS:82.13% |
| Monk 2 (DTC) | (splitter= 'random', random\_state= 3, min\_weight\_fraction\_leaf= 0, min\_samples\_split= 2, min\_samples\_leaf= 1, min\_impurity\_decrease= 0, max\_leaf\_nodes= None, max\_features= None, max\_depth= 10, criterion= 'gini', class\_weight= 'balanced') | TR:0.0  TS:0.09049 | TR:100%  TS:90.95% |
| Monk 3 (SVM) | {'C': 10, 'coef0': 1, 'decision\_function\_shape': 'ovo', 'gamma': 'auto', 'kernel': 'poly', 'shrinking': True, 'tol': 0.0001,’class\_weight’:’balanced’} | TR: 0.00826  TS: 0.05568 | TR:99.17%  TS:94.43% |
| Monk 3 (KNN) | {'algorithm': 'auto', 'leaf\_size': 60, 'metric': 'manhattan', 'n\_neighbors': 30, 'p': 2, 'weights': 'distance'} | TR: 0.0  TS: 0.07657 | TR:100%  TS:92.34% |
| Monks 3 (DTC) | (splitter= 'random', random\_state= 1, min\_weight\_fraction\_leaf= 0, min\_samples\_split= 0.1, min\_samples\_leaf= 1, min\_impurity\_decrease= 0, max\_leaf\_nodes= None, max\_features= None, max\_depth= None, criterion= 'gini', class\_weight= None) | TR:0.04959  TS:0.0 | TR:95.04% TS:100% |
| Students (SVM) | (class\_weight=None,  decision\_function\_shape='ovo',  gamma='scale',  kernel='poly',  shrinking=True,C=1, coef0=1, tol=0.0001) | TR: 0.82595  TS: 2.46835 | TR: 80.38%  TS:41.77% |
| Students (KNN) | (weights='uniform',metric='manhattan',  algorithm='kd\_tree',  p=1,leaf\_size=6,n\_neighbors=37) | TR: 0.0  TS: 0.58228 | TR: 100%  TS: 41.77% |
| Students(DTC) | (splitter='random',  min\_impurity\_decrease=0,  max\_features=None,  criterion='gini',  class\_weight=None,  min\_weight\_fraction\_leaf=0,  min\_samples\_split=0.1,  max\_leaf\_nodes=13,  max\_depth=191,min\_samples\_leaf=4) | TR:2.33544  TS:2.54430 | TR:42.41%  TS:27.85% |

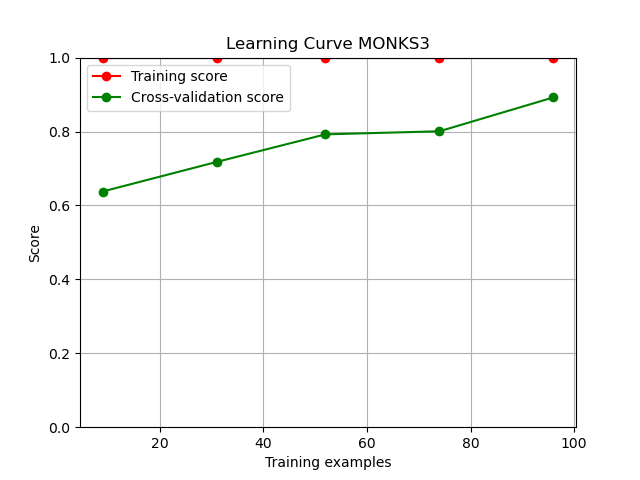
Table 2: best hyperparameters of various models

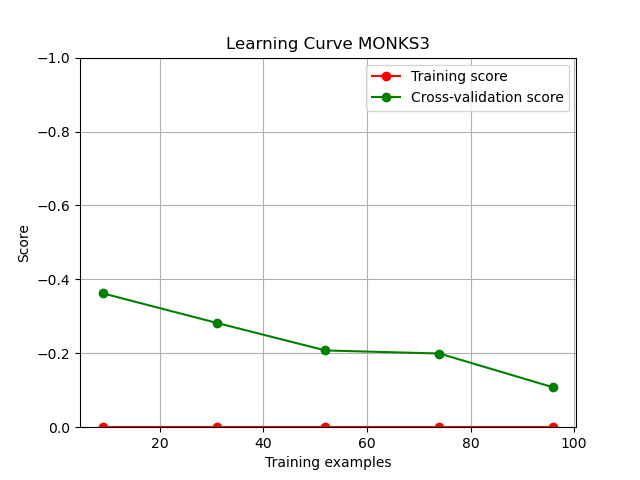
**Figure 1.** Plot of the learning rate of KNN on MONK’s 1 (left accuracy right error)

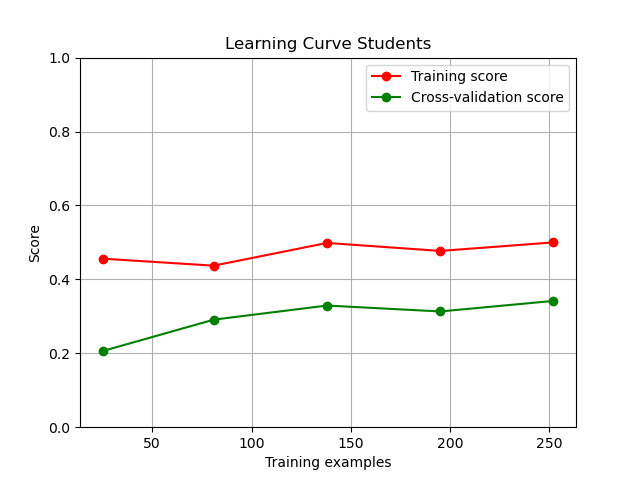
****

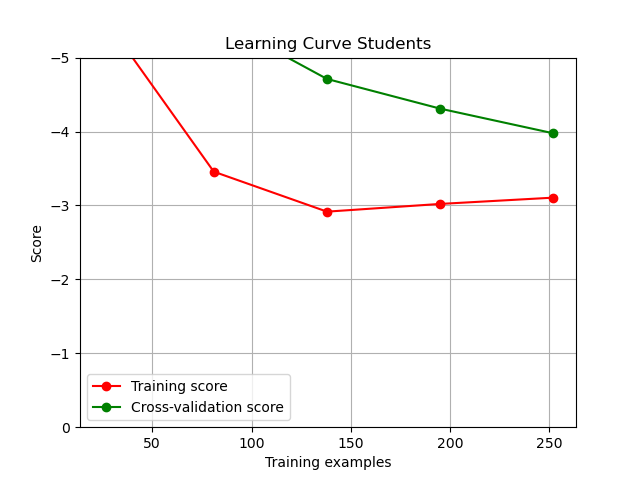
**Figure 2.**  Plot of the learning rate of KNN on MONK’s 2

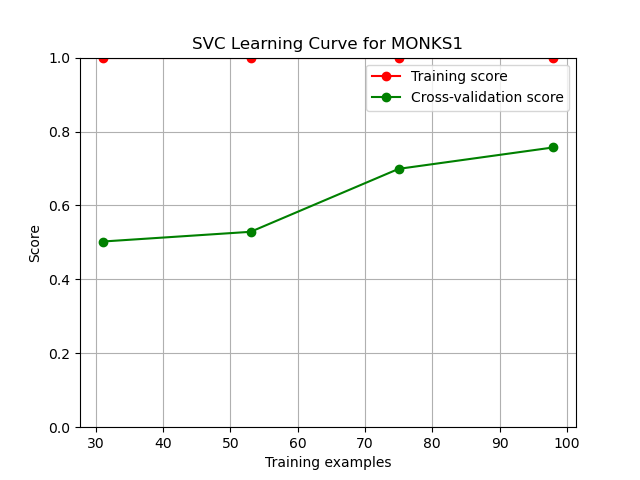


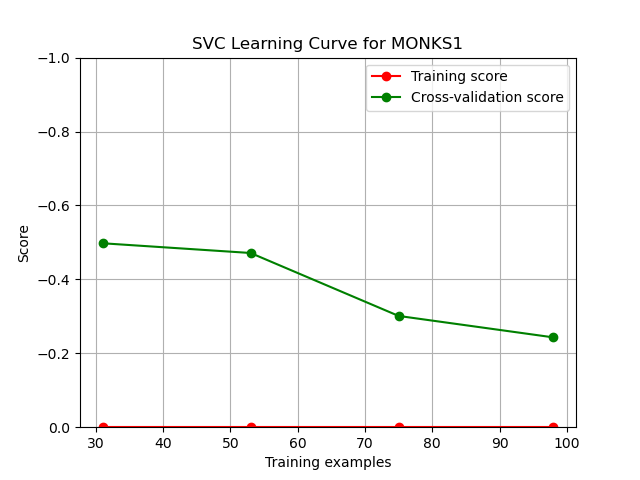
**Figure 3.**  Plot of the learning rate of KNN on MONK’s 3

****

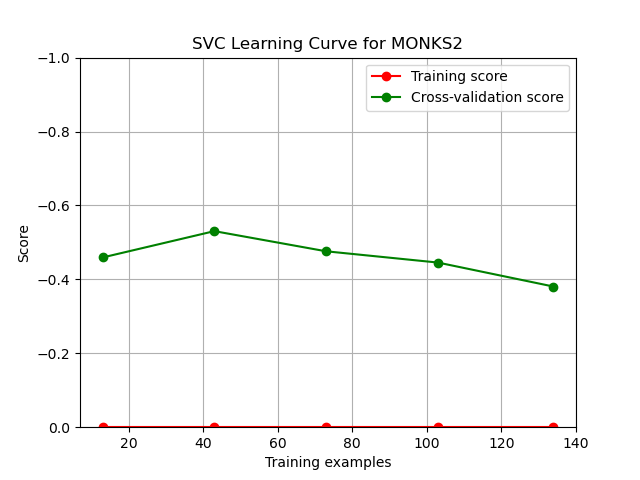
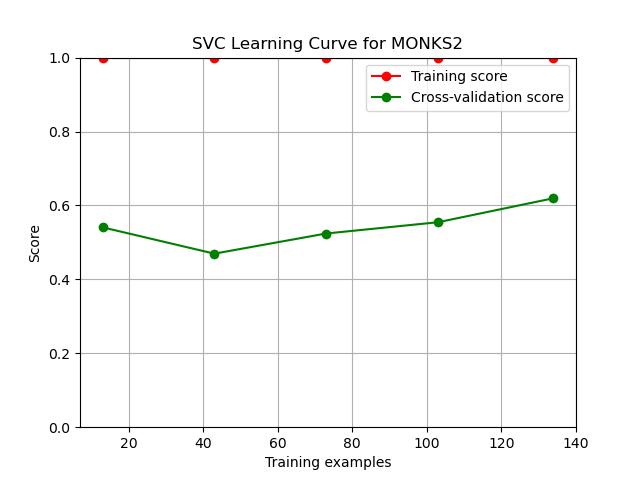
**Figure 4.**  Plot of the learning rate of KNN on Student Dataset

****

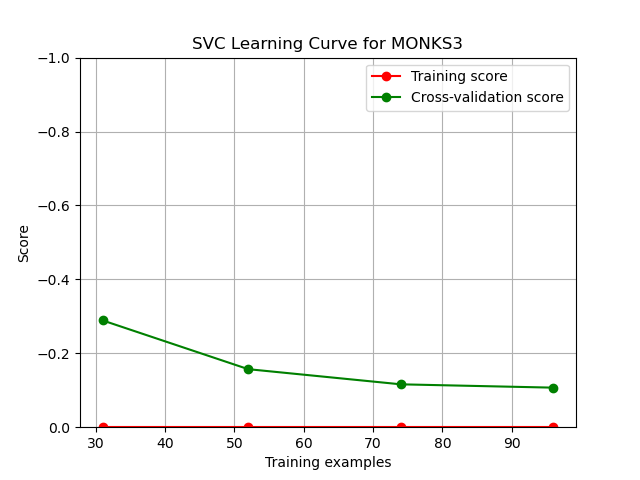
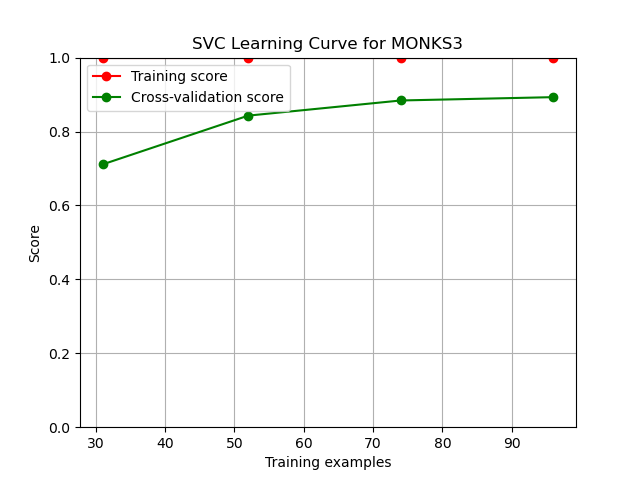
**Figure 5.** Plot of the learning rate of SVM on MONK’s 1

****

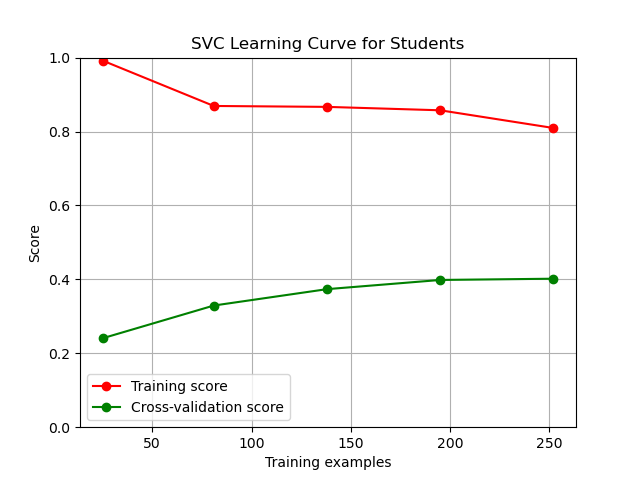
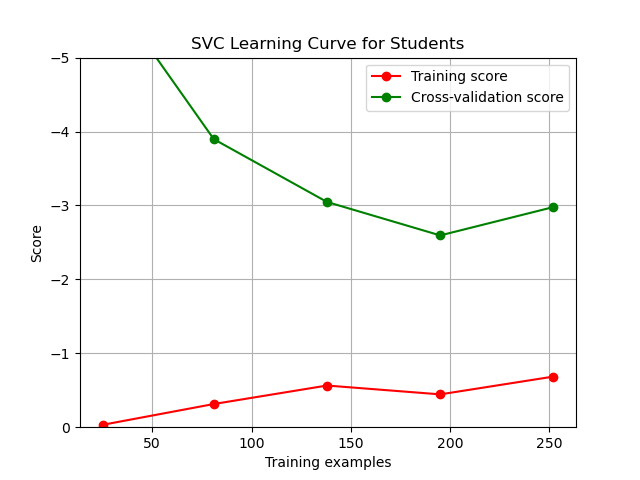
**Figure 6.** Plot of the learning rate of SVM on MONK’s 2

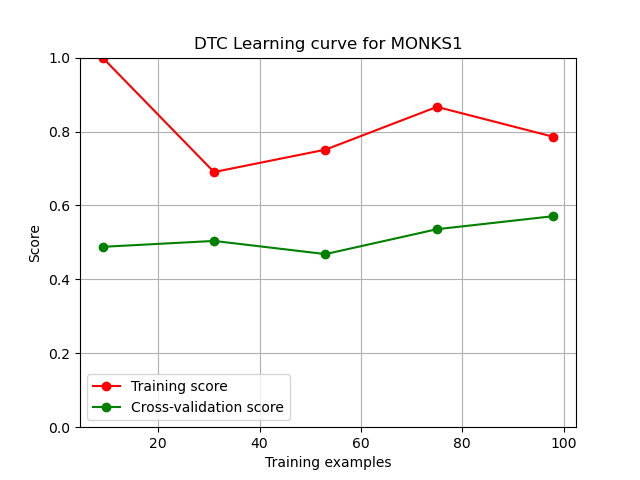
****

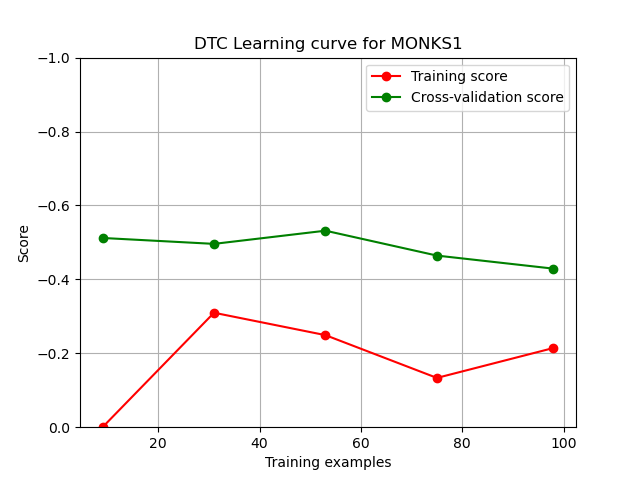
**Figure 7.** Plot of the learning rate of SVM on MONK’s 3



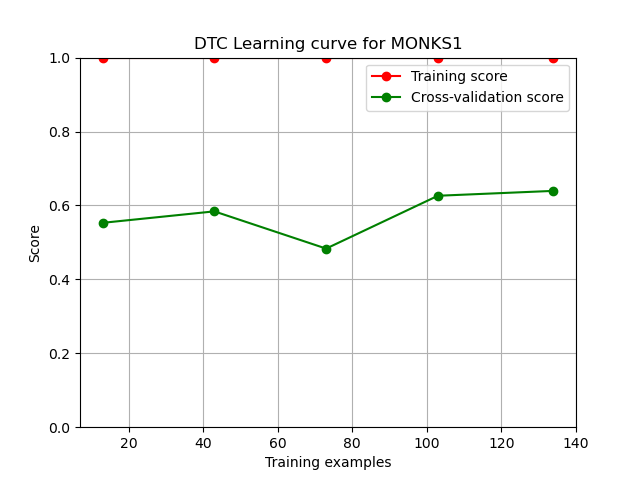
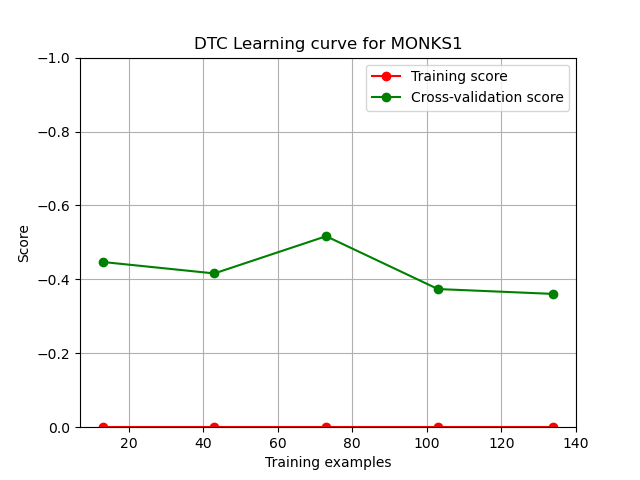
**Figure 8.** Plot of the learning rate of SVM on the Students’ Dataset



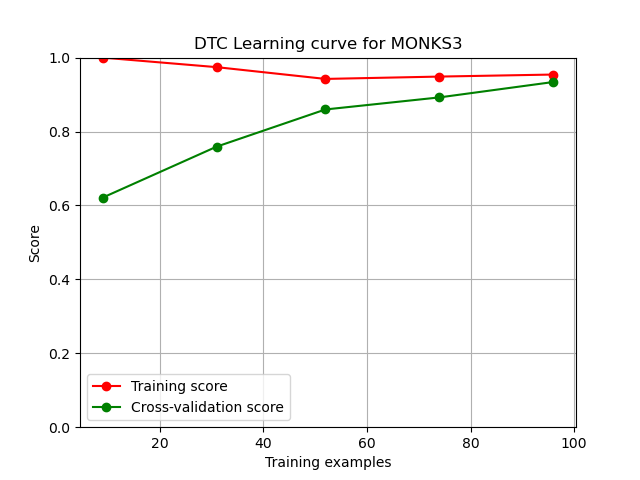
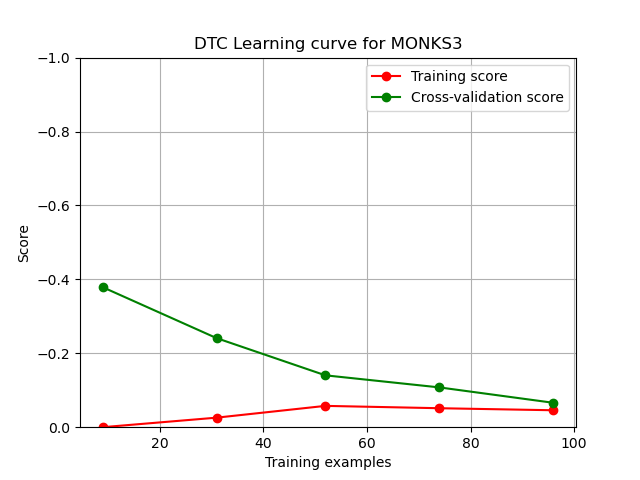
**Figure 9.** Plot of the learning rate of DTC on MONKS1



**Figure 10.** Plot of the learning rate of DTC on MONKS2



**Figure 11.** Plot of the learning rate of DTC on MONKS3



**CUP Results**

**Support Vector Regression**

The SVR was run similarly to the method used in the previous section. A simple gridsearch was run using the same parameters and methodology, with the difference that SVR does not support multiple outputs, and therefore the hyperparameters were tuned differently for the two columns (referred to as column x and column y). The testing for this model was obtained by using 20% of the training dataset as a test set. Considering the obtained learning curves (see Fig. 12-15) indicated an improvement with more data, the final result is hopefully even better . The measures used for evaluating the models were sklearn’s explained\_variance\_score (E.V.S) function and mean\_squared\_error (M.S.E) function. E.V.S ended up being the final metric used to determine which model would qualify for the MLCup, with a value closest to 1 being the best. The values for the gridsearch/randomsearch were as follows:

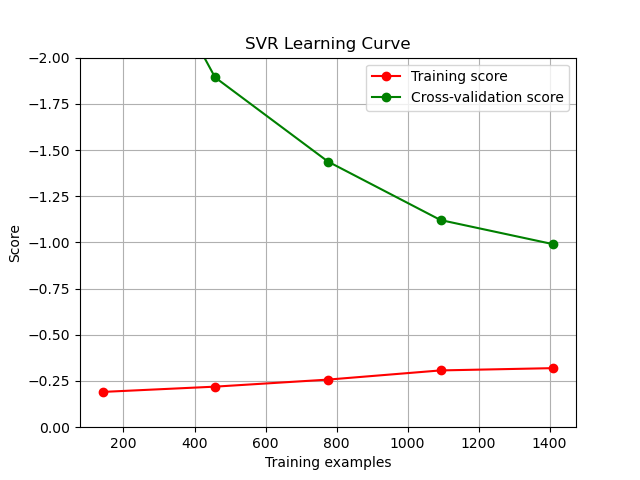
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| SVR | Kernel  ('rbf','sigmoid', ‘poly’,’linear’)  Specifies the type of kernel to be used with the algorithm  Note:poly and linear not used for regressor | Regularization parameter C | Gamma coefficient  Λ  Kernel coefficient for rbf, poly and sigmoid kernel types | Shrinking Heuristic  True,False  Whether to use the shrinking heuristic | Tol  Tolerance for stopping criterion | Coef0  Independent term in kernel function. Only used in poly and sigmoid |
| KNR | n-neighbours  number of neighbours to be used | Weight Function  ('uniform', 'distance')  Weight function. Either uniform weights will be used or weight points by inverse of their distance | Algorithm  ('ball\_tree', 'kd\_tree', 'brute')  Type of algorithm to compute nearest neighbours | Leaf size  Leaf size passed to ball tree and kd\_tree | Power Parameter  P | Metric  (minkowski, Euclidean, manhattan, chebyshev)  Distance metric to calculate the tree |

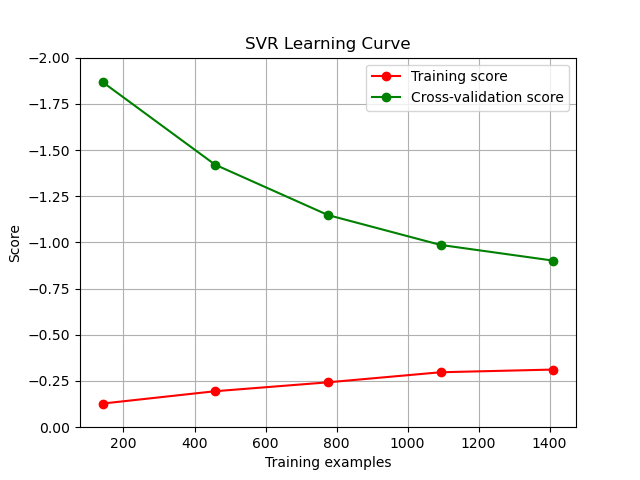
Table 3 – hyperparameters used for the regression gridsearch

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task** | **Hyperparameters** | **E.V.S** | **M.S.E** | **M.E.E** |
| SVR(x) | (shrinking=True, kernel='rbf',C=15,  epsilon=0.1,  gamma=0.098,  tol=0.07) | TR: 0.99159  TS: 0.98843 | TR: 0.54753  TS: 0.74782 | TR-0.63855  TS-0.93437 |
| SVR(y) | (coef0=0,kernel='rbf',shrinking=True,  C=8,tol=0.003,  epsilon=0.29,gamma=0.23) | TR- 0.98171  TS- 0.95306 | TR- 0.32629  TS- 0.80650 | (see above) |
| KNR(x) | (algorithm='auto',  metric='manhattan',  weights='distance',  n\_neighbors=15,p=2) | TR- 1.0  TS- 0.99167 | TR- 0.0  TS- 0.54222 | TR-0.0  TS-0.68146 |
| KNR(y) | (algorithm='auto', metric='manhattan', weights='distance',p=2,n\_neighbors=7) | TR- 1.0  TS- 0.96499 | TR- 0.0  TS- 0.60517 | (see above) |

Table 4- best hyperparameters found for the regressors

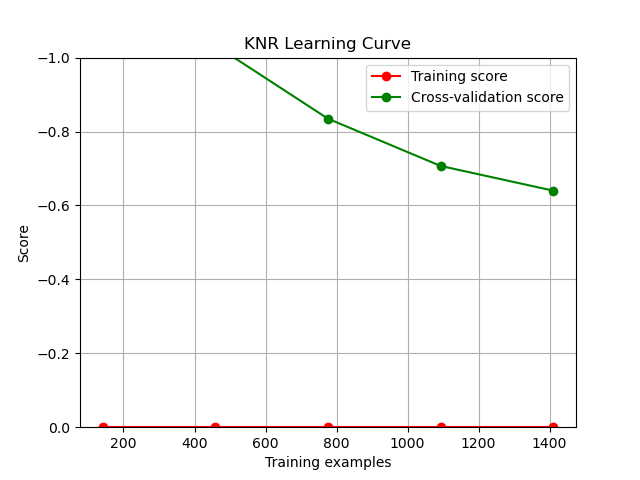
Note: The random search was run with 10000 iterations. Despite how reduced it was compared to gridsearch, this still took approximately 4 minutes to run.

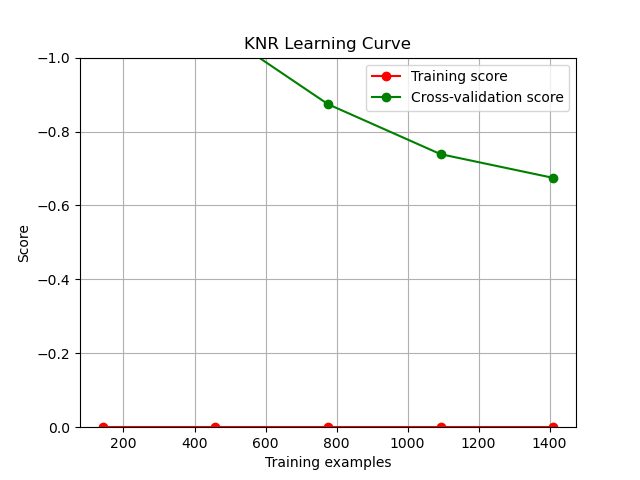
**Figure 12.** Learning Curve of the SVR for column x and y



**K-Nearest Neighbours Regression**

**Figure 13.** Learning Curve of the KNR for column x and y





These obtained results indicate that KNR is our best model to perform the regression, and therefore used for the final prediction of column 1 and 2 respectively of the ML-CUP dataset

**Conclusion**

Over the course of this research, various models and tools were utilized to fit the Student Results dataset. Our models were trained to correctly predict whether, based on certain criteria, what the final score obtained by a student would be based on 32 variables. We found out that the most efficient models we had at our disposal were both the SVM and the KNN, with a test accuracy of 41.77%. The decision tree performed very poorly compared to these previous two. This is potentially due to Decision Tree Classifier suffering notoriously from overfitting. Additionally, as shown in Figures 20-22(see appendix), randomness can play a large factor (±10%) for the final accuracy of the model, so it is a possibility that the seeds used for the randomness were simply unlucky.

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 K-Nearest Neighbours Regression on the ML Cup dataset, resulting in a final Explained Variance Score of 0.99 and 0.96 for columns x and y respectively. After calculating the Mean Euclidean Error (the official metric for the MLCUP), a training M.E.E of 0.0 and testing M.E.E of 0.68146 was obtained.

**ACKNOWLEDGEMENTS**

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

**References**

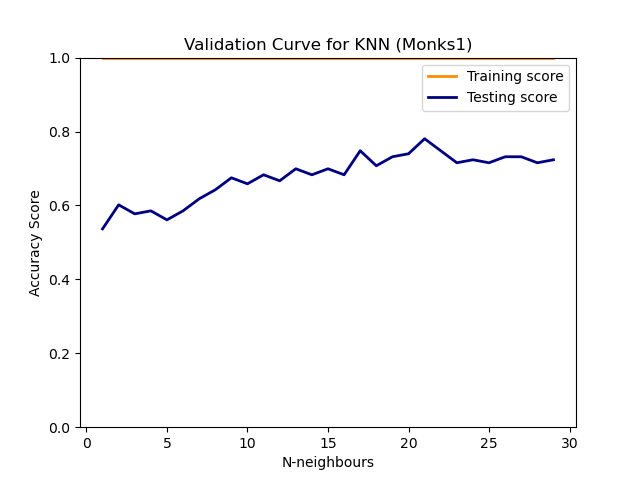
[1]: A. Micheli: Lecture slides for Machine Learning 2018/2019

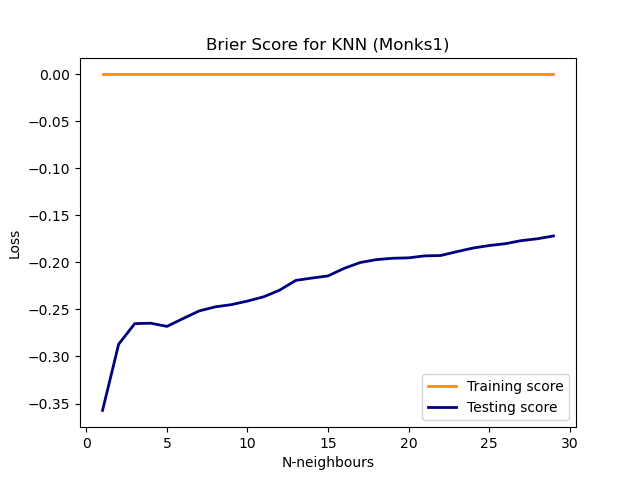
[2]: <https://www.andreagrandi.it/2018/04/14/machine-learning-pima-indians-diabetes/>

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

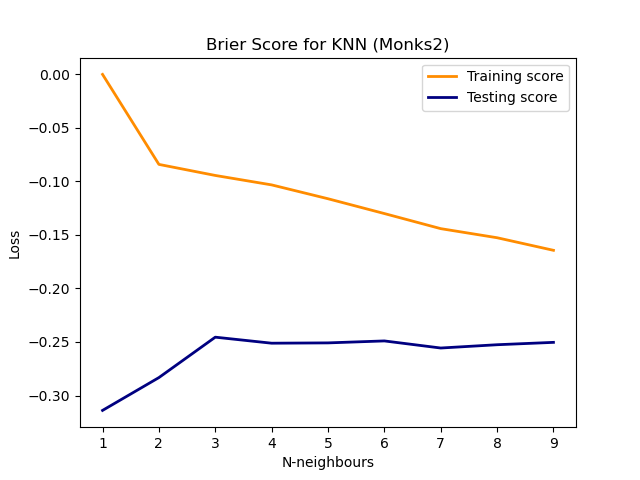
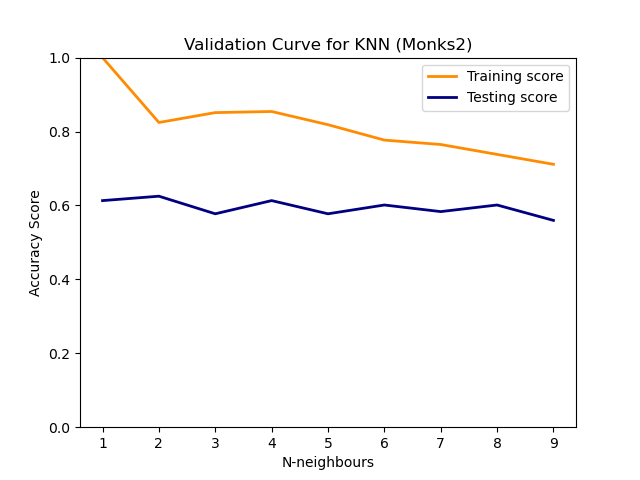
[4]: Scikit-Learn: <https://scikit-learn.org/stable/user_guide.html>

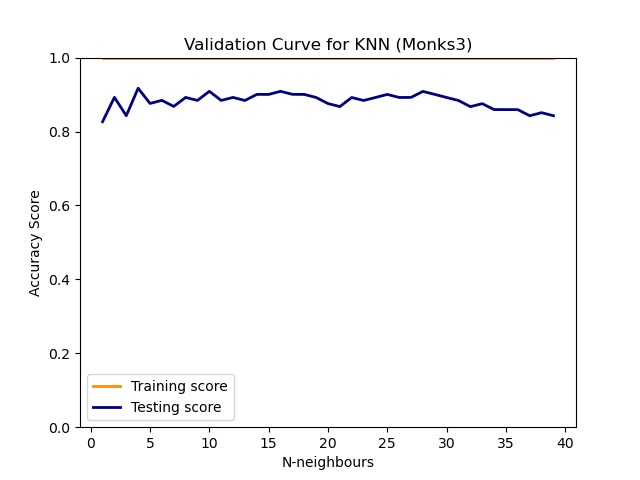
**Appendix**

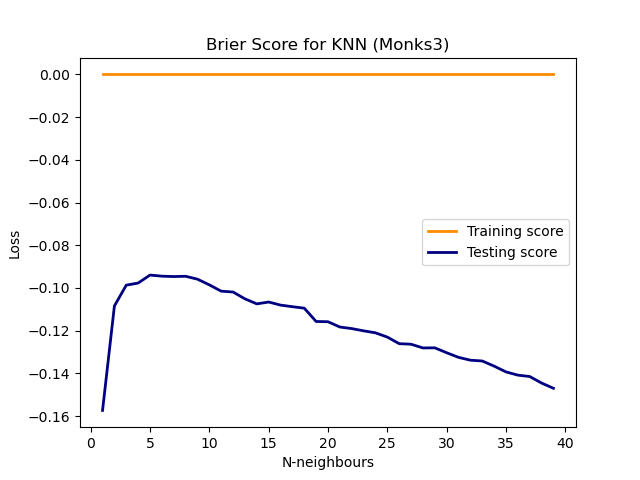
**Figure 14.** Plot of accuracy and Loss of the Monk’s 1 on the KNN



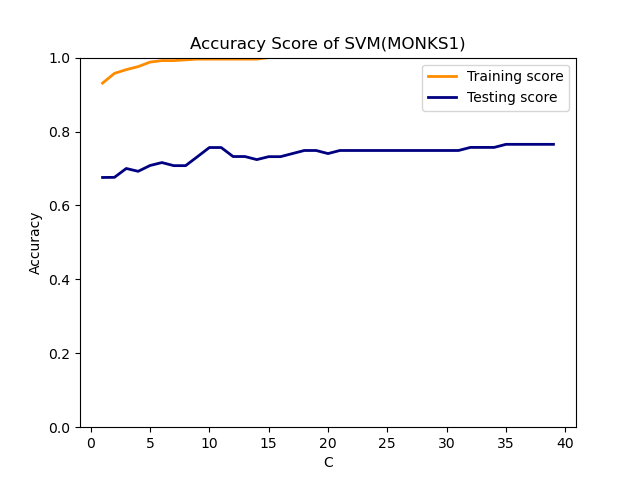
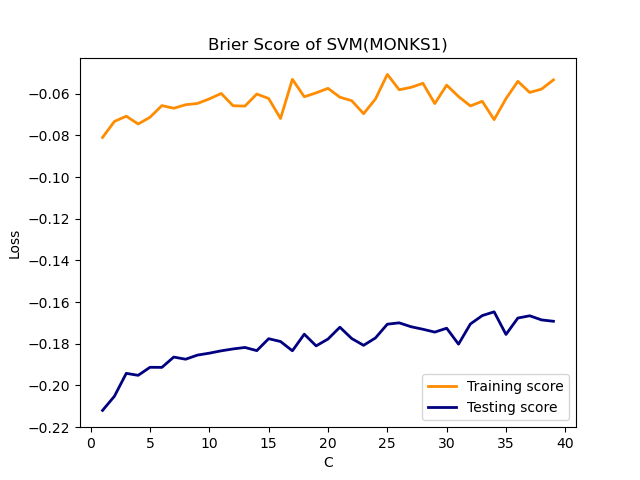
**Figure 15.**  Plot of accuracy and Loss of the Monk’s 2 on the KNN

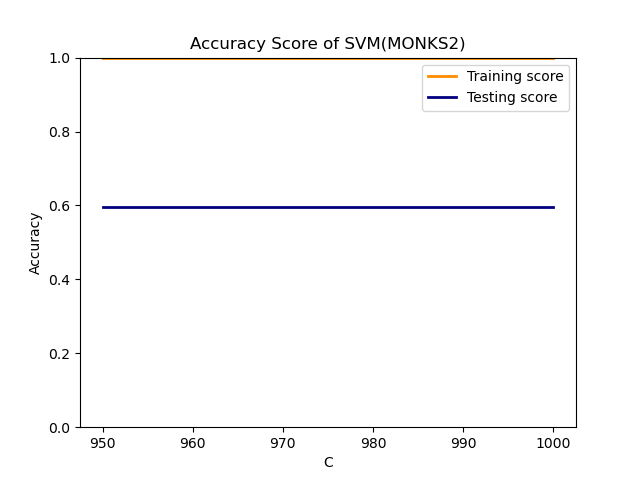


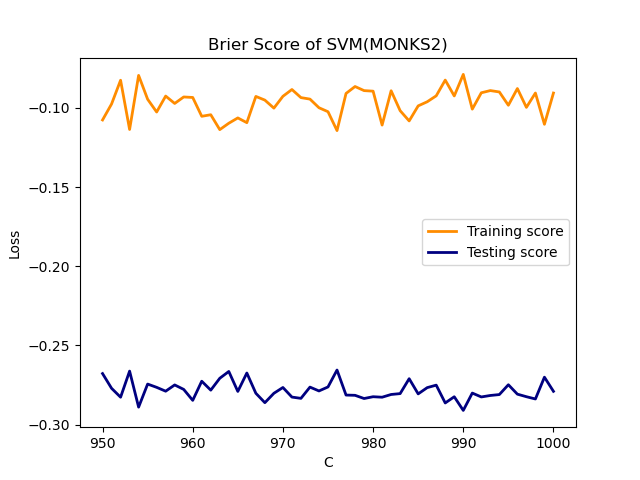
**Figure 16.** Plot of accuracy and Loss of the Monk’s 3 on the KNN

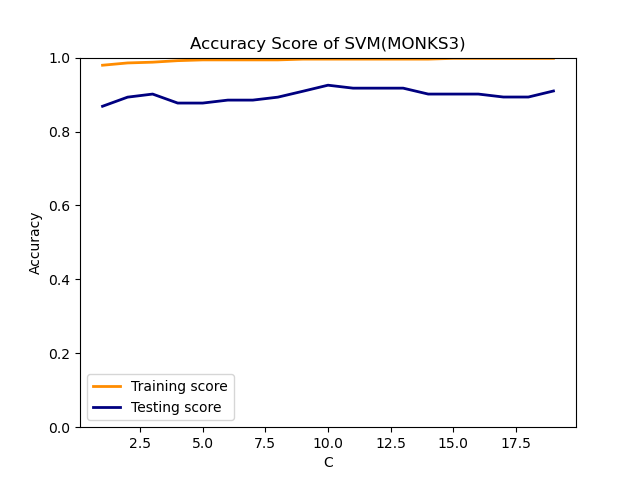
****

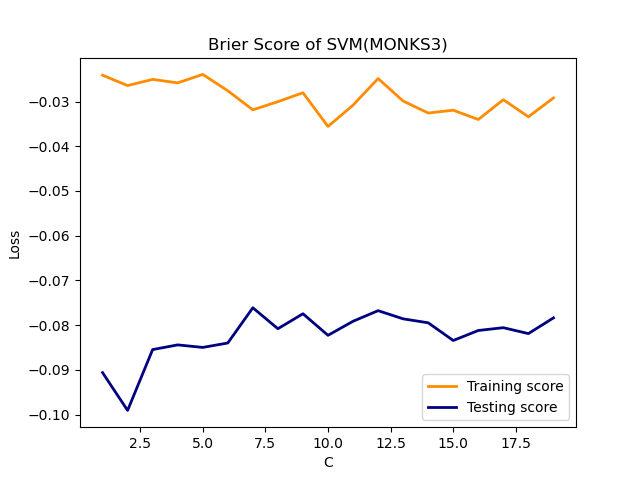
**Figure 17.** Plot of accuracy and Loss of the Monk’s 1 on the SVM

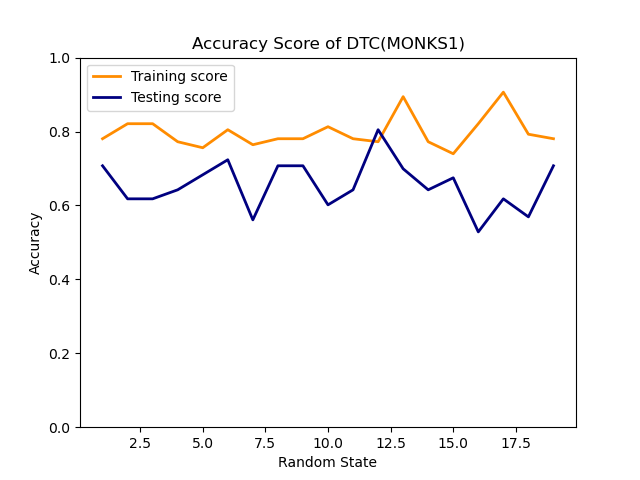
****

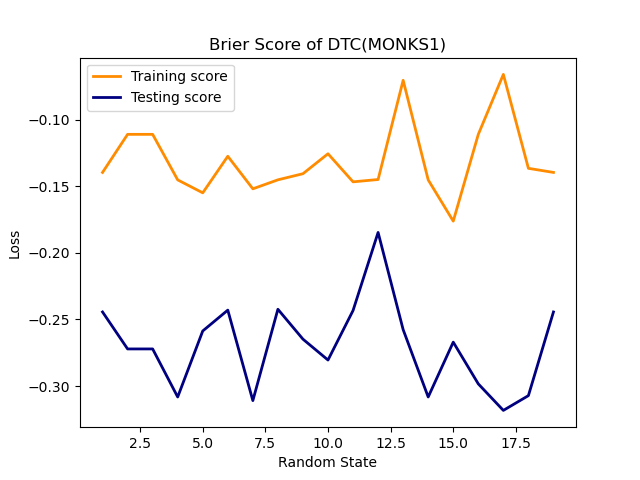
**Figure 18.** Plot of accuracy and Loss of the Monk’s 2 on the SVM

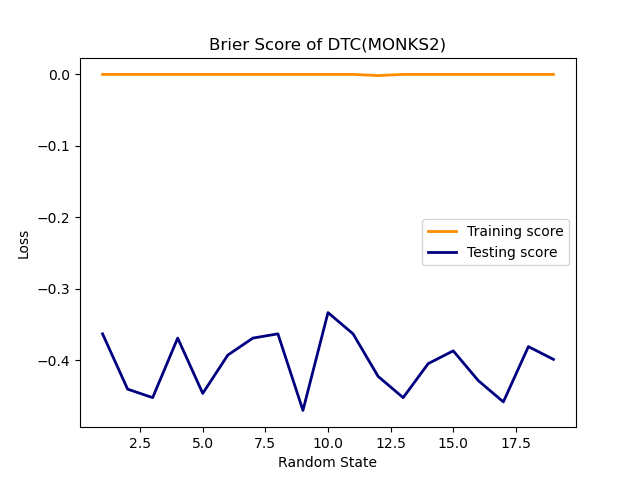


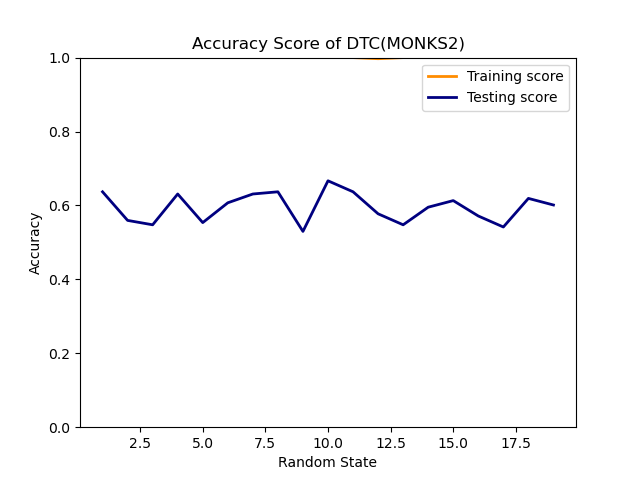
**Figure 19.** Plot of accuracy and Loss of the Monk’s 3 on the SVM



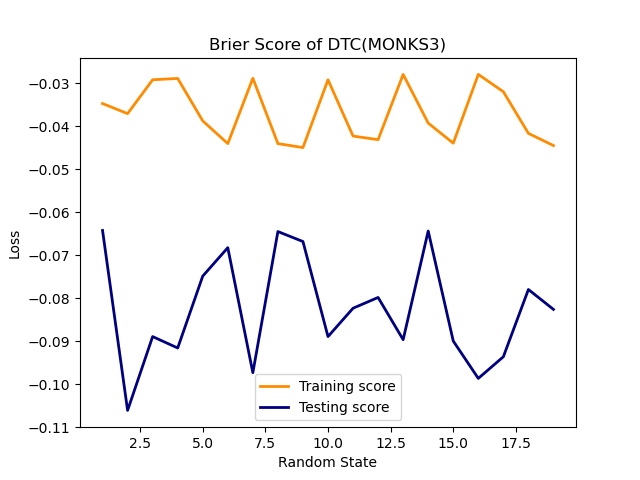
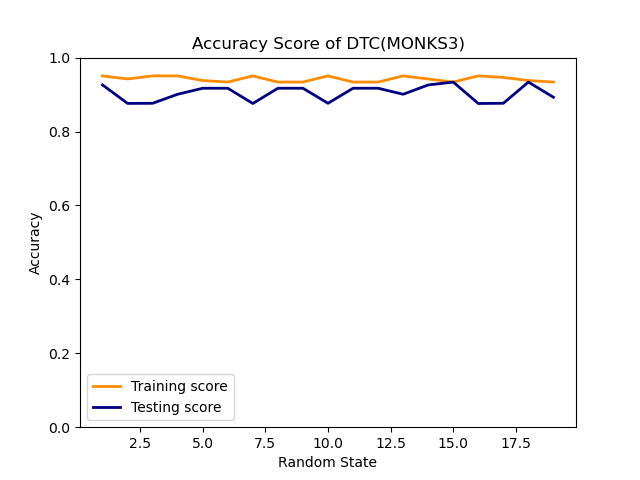
**Figure 20** Plot of the effect of randomness on the accuracy and Loss of the Monk’s 1 on the Decision Tree

****

**Figure 21** Plot of the effect of randomness on the accuracy and Loss of the Monk’s 1 on the Decision Tree

****

**Figure 22** Plot of the effect of randomness on the accuracy and Loss of the Monk’s 1 on the Decision Tree

****

1. https://archive.ics.uci.edu/ml/datasets/Student+Performance [↑](#footnote-ref-1)