• Module Code: CSMML16
• Assignment report title: Comparison of Traditional and Deep Learning Classification Methods Using Python Libraries.
• Student Number (e.g. 25098635): 24008603
• Date (when the work completed):
• Actual hrs spent for the assignment: 40hrs
• Assignment evaluation (3 key points):

Comparison of Traditional and Deep Learning Classification Methods Using Python Libraries.

Taylor, Rhian
Computer Science dept.
University of Reading
Reading, England
r.c.taylor@pgr.reading.ac.uk

***Abstract*— Traditional and deep learning machine learning models were compared in terms of ease of implementation, parameter tuning, performance and generalisability using python libraries such as scikit-learn and KERAS. Accuracy and loss were measured to understand model performance on both training and validation data sets. Regularisation methods such as cross validation and early stopping were applied to increase generalisability and avoid overfitting. Contrasting traditional methods and variations of the deep learning model were included with the aim to generally compare traditional and deep learning.**

Keywords—machine learning, deep learning, classification, python

I. Introduction

The task consisted of comparing a traditional and deep learning (DL) machine learning (ML) approach on a supervised learning problem: classify predictions on the ATLAS dataset. To compare the machine learning methods the hyperparameters and number of layers were explored to highlight differences in how the approaches work.

A traditional machine learning approach, in this instance, is considered to not include deep learning. This paper explored the K Nearest Neighbour (KNN) and Support Vector Machine (SVM) as traditional approaches. Both are classification methods rather than regression. SVM algorithms look to maximise the minimum distance between data point and the margin of the decision boundary and are particularly suited to binary classification problems [1] whereas KNN is a non-parametric method that classifies new data based on the closest data points to it [2].

L. Deng and D. Yu propose 5 variations of a definition of deep learning [3]. The key points presented in these definitions can be summarised to define deep learning as a complex machine learning technique, composed of layers and levels of abstraction that feed into each other, to learn representations. The KERAS framework was used to build a layered model as it provides a simple way to implement deep learning whilst retaining control over parameters such as number of layers, loss function, activation function etc.

The data provided was the ATLAS dataset which consists of outputs of a simulation of the ATLAS detector in CERN. This was created for a machine learning task to develop methods for detecting Higgs Bosons. The dataset consisted of background and signal data and their associated measurements. Any events with missing data were removed for this exercise. The sample had 1000 records and 30 dimensions that would be taken into account to perform classification. Classes are represented as 0 or 1 in the dataset and there are 5312 and 4688 instances of each respectively, which is balanced enough for this exercise. To train the models the data was split 90:10 for training and validation to decrease the risk of overfitting the models.

II. Methods

A. Traditional - K Nearest Neighbours (KNN)

KNN is a classification model that works by classifying data points within a certain radius to the central point of the class to that class. The size of the circle is dependent on K as it has to encompass the closest datapoints of that quantity creating a boundary for that class. KNN is known for being relatively simple and easy to implement however a disadvantage is that it can be computationally expensive as it requires the entire training set to be stored [4].

To determine the optimal number of K the *grid_search* method in the scikit-learn library [5] was utilised to determine the best value for K in the rang 1:30. Additionally the *cross_val_score* method [6] was used to show the models accuracy on new data across 10 trials to avoid overfitting. The models performance, in terms of accuracy, was recorded when re-applied to the training data (re-substitution method). Additionally, accuracy was recorded when applied to unseen validation data (hold-out method) for comparison against the other models.

B. Traditional – Support Vector Machine (SVM)

SVMs are decision machines, best suited to binary classification problems, that can be tailored via several hyperparameters. An advantage of SVMs is that their parameters correspond to the convex optimization problem so that each local solution can be considered a global optimum [4]. On the other hand, a disadvantage of SVMs is that if the classes overlap linear separation can cause poor model generalization. To counter this a misclassification penalty is usually introduced that is applied to data points on the wrong side of the margin. The data was scaled for the SVM model to improve the algorithm speed.

Two SVM models were measured in this task, known henceforth as pre-tuning and post-tuning, one with generic hyper-parameters and one where grid_search was applied to determine the most suitable hyper-parameters. The pre-tuning model parameters were set to *gamma = 'auto'* and *kernel = 'linear'* whilst the post-tuning model parameters were *C = 10*, *gamma= 0.01*, and *kernel= 'rbf'*.

Gamma –	This parameter defines the influence of a single training data point. The lower the gamma value the further the distance.
Kernel –	This parameter defines the kernel function.
C –	This parameter corresponds to the trade-off between correct classification of training data and maximising the margin. A larger C value results in the acceptance of a smaller margin.

As mentioned above, the two kernels used were linear and radial basis function (RBF). Generally, kernels are measures of similarity between *x* and *x'* that can be defined as[2]:

$$k(x, x') \geq 0$$

A linear kernel is the dot product of vectors *x* and *x'* and are best applied when the classes are linearly separable.

$$K_{linear}(x, x') = x^T x'$$

By comparison the popular RBF kernel decreases with distance, has a range of 0-1 and has a feature space that can handle infinite dimensions. The RBF kernel is dependent on the radial distance from a central point[4] and can be defined as:

$$K_{RBF}(x, x') = \exp\left(-\frac{1}{2\sigma^2} \parallel x - x'^{\parallel^2}\right)$$

Where σ^2 defines the bandwidth of the kernel, the larger it is the slower the decay.

Cross Validation was also applied to the SVM model as well as the re-substitution and hold-out methods to better compare the accuracies of the models.

C. Deep Learning

The deep learning (DL) model was built using the KERAS library[7]. This allowed a model to be created layer by layer and for the model parameters to be easily customised. The approach taken consisted of building a model, assessing the performance, adding a layer and repeating the assessment. This was done for a model with 0-3 hidden layers. To be considered deep learning the model must consist of multiple hidden layers. Therefore, the approach taken allowed the comparison of single layer networks and deep learning in addition to the comparison with the traditional ML techniques.

Each variation of the DL model utilised the same split of training and validation data as the traditional techniques and used the same parameters with the only difference being the addition of a hidden layer and the number of units it consisted of.

The loss function utilised in the DL model was *binary cross-entropy* which is the default loss function for binary classification problems. The loss is determined by measuring the distance of a prediction from the true value and taking an average of the class-wide errors.

ADAM was used as the optimization function in the DL model. Introduced in 2015, ADAM is an efficient, straightforward algorithm that is suited to large datasets or data with high dimensionality. The method is based on stochastic optimization and aims to increase efficiency where the parameter space has many dimensions. The name was derived from adaptive moment estimation and it has advantages that include the unnecessity of a stationary objective, the ability to work with sparse gradients and that the hyperparameter stepsize approximately binds its stepsizes[8].

Two activation functions are utilised in the DL Model. For every layer except the last the *RELU* activation function is used, whereas the *sigmoid* activation function is used in the output layer.

RELU stands for rectified linear unit and is the activation function utilised in the majority of the DL models layers. This is the most commonly used activation function and is defined as [9]:

$$h(a) = \max(0, a)$$

The activation function used in the output layer, otherwise known as the output function, of each variation of the DL model is the *sigmoid* function which is best suited to binary problems and is therefore better suited to this task than the *softmax* output function. The *sigmoid* output function is defined as:

$$\hat{y} = p(y = 1|x) = \sigma(a)$$

Each variation of the DL model trains in a maximum of 30 epochs, with the application of early stopping. Early stopping was used as a regularisation method on the DL model to minimise overfitting. Whilst iteratively training the DL model, early stopping will force a halt when the validation score shows that the model is learning the training data exactly and, therefore, will not generalise well.

The number of units in each layer was dictated by the number of dimensions in the data. The convention of the input layer having a number of units that is more than or equal to the number of dimensions in the data and each layer subsequentially has less units until the final output layer has one was used in each variation of the DL model.

III. Results

The traditional ML methods, KNN and SVM, and the effects of tuning hyperparameters on these models was considered by comparing their accuracies pre and post tuning. Prior to tuning the models, the KNN outperformed the SVM and both were recording accuracies in the lower regions of 70%.

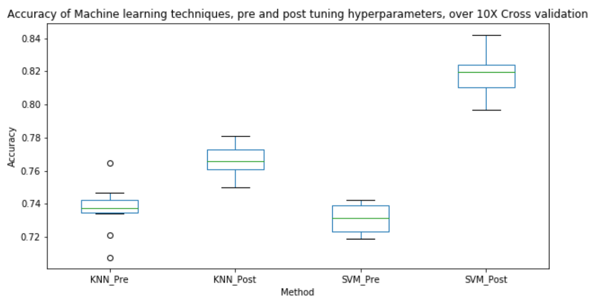


Figure 1 Box plot comparing accuracies of traditional ML methods pre and post tuning.

After tuning the hyperparameters both models saw an improvement, with the SVM accuracy overtaking KNN and averaging around 82% as shown in Figure 1 and Figure 2.

As early stopping was implemented on the DL model, accuracy and loss were recorded at each training epoch for each variation of the model (0-3 hidden layers) for both the training and validation data.

Over time, each variation of the model saw a general trend of increased accuracy and lower loss suggesting that with each iteration the model was learning and achieved good scores, above 80% on both training and test data. As seen in Figure 3 and Figure 4 the number of training epochs varies demonstrating the early stopping regularization and suggests the scores achieved by the DL models were not due to overfitting.

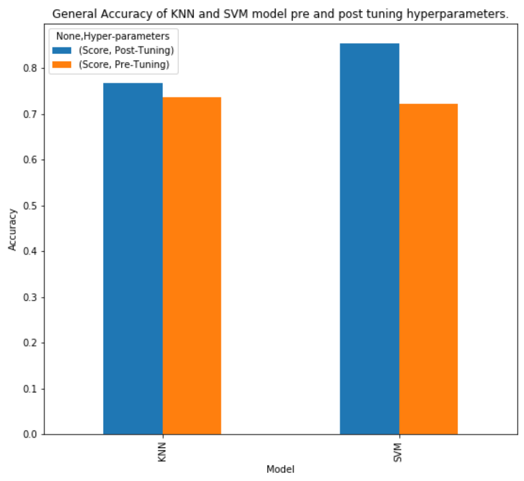


Figure 2 General accuracy of traditional methods pre and post tuning.

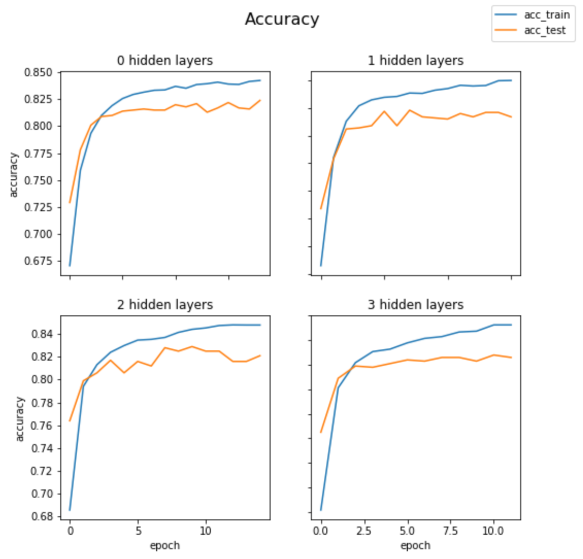


Figure 3 Accuracy of DL variations on train and test data over training epochs.

Each variation of the model performed better on the training data, as expected, and the figures show the learning curve was much smoother compared to the validation data. Despite this, the performance of the model when applied to the validation data showed that it was generalizable by achieving roughly 82% accuracy over the four variations.

When comparing the variations of the DL model, to understand the effect of the number of hidden layers, accuracy and loss were measured on both the training and validation data. Generally, the DL model with 3 hidden layers performed best as it achieved the highest accuracy on both the training and validation data, as shown in Figure 5.

On unseen data, as layers were added to the model (1-3 hidden layers) the accuracy increased linearly and the loss decreased in a similar fashion however, on the training

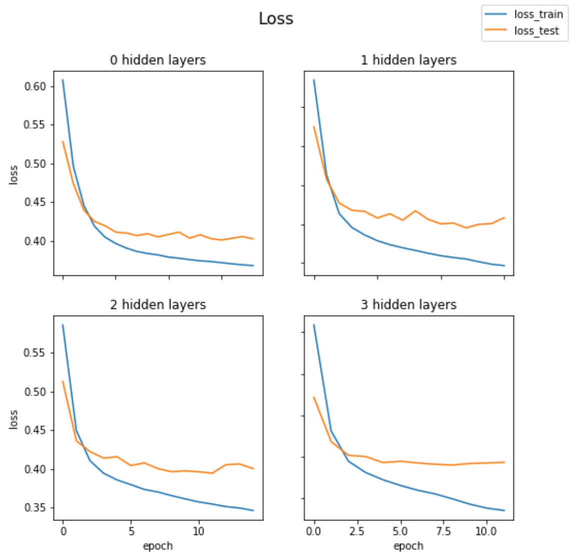


Figure 4 Loss of DL variations on train and test data over training epochs.

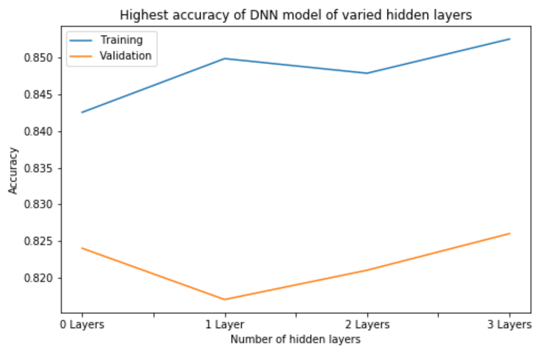


Figure 5 Accuracy of DL model with various number of hidden layers.

data the model with one hidden layer outperformed the model with 2 hidden layers.

When comparing the traditional and DL methods the accuracy of the models on both the training and validation data was considered. As stated, the DL model with 3 layers (DL-3) had the highest DL performance on both datasets. Every DL model outperformed the KNN approach, making it the worst performing model, as seen in Figure 7 and in more detail in Table 1.

On the training data set the SVM model performed the best with a slightly larger accuracy, 85.5%, compared to DL-3, 85.3%. However, on unseen data DL-3 performed slightly better than SVM with an accuracy of 82.6% compared to 81.1%.

Table 1 includes the accuracies outputted for each model where the accuracies for the DL are the highest achieved and the accuracies for the traditional models are the average accuracies from the cross validation performed. The performance of each model on both training and validation are included to give assist with the interpretation of the included figures where the accuracies are close.

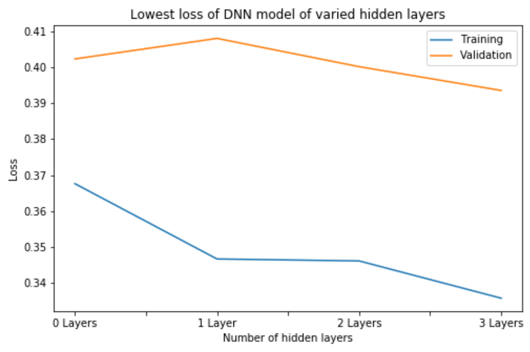


Figure 6 Loss of DL model with various number of hidden layers.

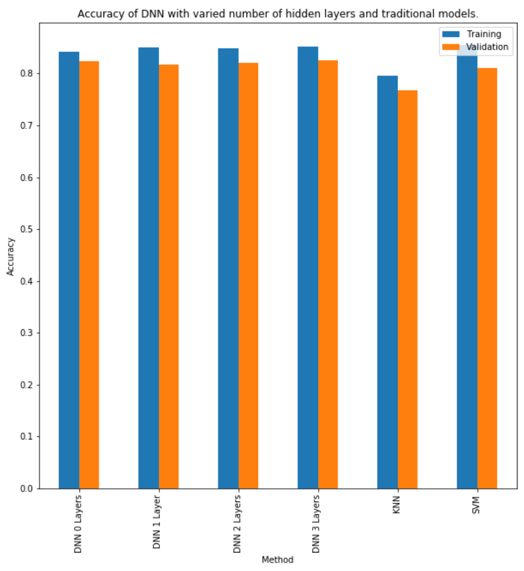
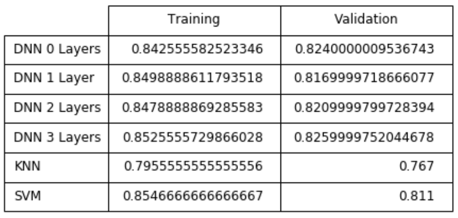


Figure 7 Grouped chart showing accuracy of each implemented model on training and validation data.

Table 1 Accuracies of each model on training and validation data.



	Training	Validation
DNN 0 Layers	0.842555582523346	0.8240000009536743
DNN 1 Layer	0.8498888611793518	0.8169999718666077
DNN 2 Layers	0.8478888869285583	0.8209999799728394
DNN 3 Layers	0.8525555729866028	0.8259999752044678
KNN	0.7955555555555556	0.767
SVM	0.8546666666666667	0.811

IV. Discussion

Two traditional ML methods were implemented, KNN and SVM, and of the two the SVM model performed the best and was considered the strongest competitor to the DL model from an accuracy perspective. However, the KNN was very simple to set-up as there was only one parameter to work with, K (the number of neighbours), and it ran very quickly. Furthermore, the KNN algorithm performed well on the raw data, there was no need to scale or normalise it. Considering this, the accuracy achieved by the KNN, 79.6% on training and 76.7% on validation, was very good. By contrast the SVM was slow, requiring the data to be scaled for it to finish in an acceptable time, and it also had several hyperparameters to consider which made finding optimal parameters a laborious process as there were several combinations to trial.

Four variations of a DL model were created to ascertain the impact of additional hidden layers and the results showed that the model with the most hidden layers, DL-3, performed the best on both test and validation data. Using the KERAS library to implement the DL model made it easy to vary the number of layers and explore different combinations of parameters for the model. The model was quick, and this was aided by the implantation of early stopping as the 30 epochs limit was not needed by any variation having reach their peak accuracy sooner.

The accuracy for each variation of the DL model was high, above 80% for both training and validation data, which raised the question whether DL was necessary for this task as the model with just an input and output layer performed sufficiently well.

When comparing DL-3 and SVM methods they had very similar accuracies with SVM performing best on seen data and DL-3 performing best on unseen data. Despite the differences in accuracy being small the model that performed best on unseen data, DL-3, should be considered the better of the two as it is more generalisable.

A. Limitations & Future Work

One limitation of the current work is that the original dataset was unbalanced. For this task the imbalance was tolerated as it was less than 10% of the total number of records however for future work it would be interesting to see whether such a small imbalance would affect the output of the various models.

The discussion in this work centred itself on the interpretation of model accuracy that did not consider measures such as precision, recall or F-measure that would have given additional insight into the performance of the model. Additionally, DL-3 and SVM were singled out as the top performing models, with DL-3 being considered the superior due to its performance on unseen data. However, no statistical test was performed to understand whether the difference in the SVM and DL-3 performance on unseen data was significantly different. Considering this, for future work, it is recommended that some test such as McNemar's test to understand the difference in the proportion of paired data, in this case the number of data points assigned to each class by the different models.

Finally, another consideration for future work would be to apply other regularisation methods to the DL model such as dropout where it is possible to use it in combination with cross validation increasing the consistency of the methodology as opposed to what was done in this study where the traditional methods shared a methodology and the DL variations shared a methodology regarding the generalisability of the models.

B. Conclusions

This task compared and contrasted traditional and DL machine learning methods, considering how each method worked, the model parameters, ease of set-up using python libraries and model accuracy.

Having applied KNN, SVM and DL to the ATLAS dataset DL-3 was considered the best in terms of accuracy and generalisability.

References

[1] S. G. Alonso *et al.*, "Data Mining Algorithms and Techniques in Mental Health: A Systematic Review," *Journal of Medical Systems*, vol. 42, no. 9. Springer New York LLC, 01-Sep-2018, doi: 10.1007/s10916-018-1018-2.

[2] T. Thorne, "3 – Classifiers and Support Vector Machines." pp. 1–29.

[3] L. Deng and D. Yu, "Deep learning: Methods and applications," *Found. Trends Signal Process.*, vol. 7, no. 3–4, pp. 197–387, 2013, doi: 10.1561/2000000039.

[4] C. M. Bishop, *Pattern Recognition and Machine Learning*. Cambridge: Springer New York LLC, 2006.

[5] Scikit Learn, "sklearn.model_selection.GridSearchCV." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html. [Accessed: 01-Mar-2020].

[6] Scikit Learn, "sklearn.model_selection.cross_val_score." [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_val_score.html. [Accessed: 01-Mar-2020].

[7] F. Branchaud-Charron, E. Bursztein, F. Rahman, G. de Marmiesse, H. Jin, and T. Lee, "Keras: The Python Deep Learning library." [Online]. Available: https://keras.io/. [Accessed: 01-Mar-2020].

[8] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–15, 2015.

[9] T. Thorne, "Neural Networks," pp. 1–44.