Telling a Good Yarn: Predicting a Yarn's Success with Ravelry Data and Machine Learning

Alexander Hall

Department of Computing and Digital Media, Robert Gordon University, Aberdeen,UK

***Abstract*—Various machine learning methods were applied to the entire yarn database from Ravelry.com in an attempt to identify attributes that influence the average user-rating for a given yarn. The dataset was used to train Support Vector Machines, Deep Neural Networks and Random Forests; with various parameters. The Random Forest method outperformed the other two methods. Attributes relating to the yarn producer (for example, number of yarns previously produced) and physical dimensions of the yarn (grams, gauge etc) were found to be more important than the physical composition of the yarn when predicting user rating. The time a yarn has been present on the Ravelry database was found to be the most significant predictor for a yarn's rating.**

I. Introduction

RAVELRY is the world's most popular online network for fibre artists. It is used by businesses and individuals worldwide to exhibit and advertise knitting and crocheting products and patterns. Since it was founded in 2007, it has grown to include sizeable databases of a variety of knitting products which can be rated by any of the 6 million users, with the 'yarn' database being of interest to this study. For clarity, in the context of knitting and crocheting, a 'yarn' is composed of raw animal, vegetable or synthetic fibers; often in a mixture. These are interlocked to create a continuous length, colloquially known as a 'ball of wool'. Any user on Ravelry is able to rate a yarn, and each yarn on the database subsequently has an average rating. As it is notoriously difficult to predict the success of a yarn based on it's physical characteristics, a number of machine learning methods were applied in an attempt to identify the attributes which lead to a successful yarn.

A. Related Work

Both Support Vector Machines (SVMs) and Neural Networks (NNs) have seen prior use in predicting the quality of a yarn [1], [2] and improving yarn characteristics [3], [4]. However, these applications are focused on the industrial process of large-scale yarn production. To date, no studies have been identified which apply modern data-mining methods to predict the commerical success of yarn products; with Ravelry being the largest publically available information source for such products. An extensive literature search failed to identify any academic studies of the Ravelry databases, making this study the first known attempt to predict yarn rating through the use of machine learning.

B. Ethical, Legal and Social Considerations

The dataset was obtained through use of a free Ravelry API key which was obtained directly from Ravelry administrators via a public message board. As such, no new data has been released into the public domain as a result of this study. Usage of the data falls within the Ravelry API terms of use [5]. Furthermore, the co-founders of Ravelry were contacted to obtain permission for the use of the data, which is publically available. As the yarns which are rated on Ravelry are commercial products, any relationships found between yarn characteristics and ratings could potentially be used for commercial gain by fiber companies. As the individual companies which made up the yarn dataset could, if identified, potentially be affected by the results, any references to company names or identities have been removed from the dataset.

II. Data Exploration, Preparation and Dimensionality Reduction

The raw dataset required significant processing prior to analysis. The raw dataframe build directly from the data downloaded from the Ravelry API contained 109556 instances of 25 attributes of type list, integer, float, string and boolean. Processing the dataset required two steps. First, the data for all attributes was coerced to type numeric in order to simplify application of machine learning methods. Secondly, attributes which were considered redundant or not relevant to the analysis were removed. Coercing all attributes to numeric type was carried out first as this gave an expanded set of attributes which could then be selectively removed. The attributes which were of type 'list' were converted to multiple numeric attributes, after expanding out the relevant lists. This resulted in a dataset of 91 attributes. Those of type 'boolean' were converted to numeric (TRUE=1, FALSE=0).

The second step of data processing was then carried out by removing attributes considered redundant or outside of the scope of this investigation. All text-only attributes were removed (yarn description, company name etc). All attributes related to hook or needle size were removed since these are dependent on the yarn weight and therefore carried no additional information. There were several attributes for yarn weight since the raw dataset included the yarn weight in several different units of measurement. Therefore, all attributes relating to yarn weight were removed with the exception of 'yarn_weight_ply'.

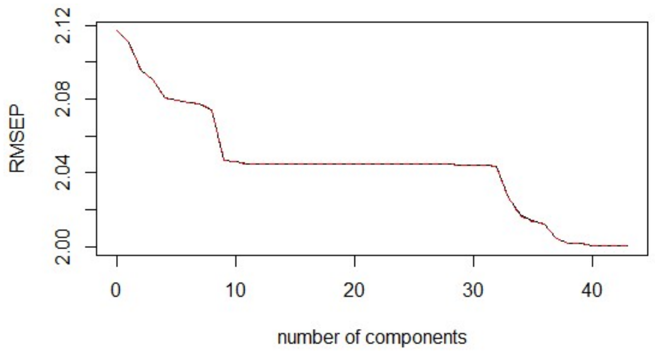
The 'id' attribute is simply a numeric identifier for each yarn. It was decided to leave this present in the dataset since the id is assigned sequentially as new yarns are added to the database. The 'id' attribute therefore gives an indication of the relative age of yarn, with older yarns having smaller ids. Finally, three new attributes were created from the attributes which recorded the percentage of different fiber types in each yarn. These new attributes named 'vegetable', 'animal' and 'synthetic' recorded the percentage of vegetable, animal and synthetic fiber in each yarn. These give a more general overview of each yarn's composition and it was believed that including them may improve the quality of any correlations. After processing, the dataset consisted of 43 independent variables, along with the dependant 'rating_average' variable (henceforth referred to as 'yarn rating').

57504 of the dataset entries contained NA values. It was decided to omit these entries from analysis due to the already large dataset size and to simplify analysis. In addition 17192 instances had a yarn rating of zero. These instances represented yarns which had not yet been rated. These were also removed from the dataset as these instances formed a discontinuity in the raw data – zero star ratings are not possible on Ravelry so with these instances included, yarn ratings could take the values 1.0-5.0 OR 0.0. Removing the non-rated instances ensured the data was continuous and therefore suited to a regression model. A total of 34860 instances remained for analysis.

A. Principal Component Regression

In an attempt to reduce dimensionality of the dataset further, Principal Component Regression (PCR) was carried out to identify any colinear independant variables. As the number of variables used for PCR increased, the Root Mean Squared Error of Prediction (RMSEP) monotonically decreased (see figure 1), indicating all the remaining independant variables accounted for some degree of variability in the data. It was therefore decided to use all remaining attributes in the dataset for subsequent analysis.

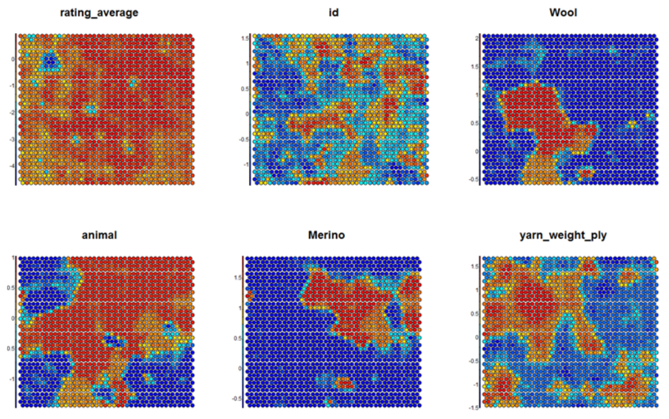
Fig. 1: Validation plot for PCR carried out on the yarn dataset



B. Self Organising Map

Due to the inability of PCR to identify a clear set of important variables, it was decided to construct a self-organising map (SOM) to assist in visualising any potential relationships between variables and yarn rating. The Kohonen package was used in R for this purpose [6]. Through trial and error, a grid size of 40X40, with a hexagonal topography was chosen to ensure instances were reasonably well distributed amongst nodes in the map. Separate heatplots for each individual variable were created from the output of the SOM algorithm. Each heatplot provided a visual representation of the distribution of its respective variable throughout the map. The positions of each instance from the dataset remained constant between heatmaps, which were coloured by the 'density' of the respective variables [6]. Correlations between variables can be qualitatively identified by studying the heatplots for similar or differing regions across two plots. No immediately obvious correlations were identified but some more subtle potential relationships are demonstrated in figure 2.

Fig. 2: Heatplots for individual variables, as generated by a Self Organising Map



The inability to identify any clear correlations in the SOM heatplots further suggests that no single variable, or simple linear combination of independent variables has a significant influence on the yarn rating. As a result, more complex methods of analysis were necessary to identify possible non-linear relationships.

III. Experiments

All data acquisition, processing and analysis were performed using R. In order to reduce computation time, the data analysis was performed with virtual Linux machines using the Amazon EC2 cloud computing service. In all cases a single '4.8x Large' instance (36 core Intel Xeon E5-2666 v3) [7] was used, running RStudio server. Prior to using EC2, attempts were made to train models on subsets of the data (with 10% of the dataset used for training), however such models gave inconsistent results; therefore the entire dataset was used for model training and testing going forward. In order to effectively utilise the multi-core architecture, parallelisation of the code was necessary. In the case of the SVM and random forest models, parallelisation was carried out on the parameter grid search and the cross validation process. In order to implement this, the 'doParallel' library was used [8]. The deeplearning function in H2O, as used for the neural network, is implicitly parallel through the

Hogwild! scheme [9] so no further parallelisation of the code was required for NN models. Parallelisation came at the cost of reproducibility during parameter tuning since dynamic task allocation prevents subsequent runs of each program executing identically [15]. However, parallelisation can be disabled and each of the best models may be re-created using the parameters found by each respective experiment.

Each method was limited to approximately 4 hours of computation time to allow a meaningful comparison between the methods. In all cases, some experimentation was carried out on a local machine with small subsets of the dataset to determine rough ranges for tuning parameters prior to training the full dataset on EC2.

5 fold cross validation was used throughout to evaluate each model. This was chosen as a compromise between evaluation quality and computation time. As is common for regression models, the Root Mean Squared Error (RMSE) was used as the evaluation metric for each model. Prior to computing any predictive models, the RMSE between all actual yarn ratings and the mean yarn rating was calculated, giving a RMSE of 0.675. As this was a trivial computation, this was used as a benchmark for model performance, with any RMSE below this considered an improvement.

A. Support Vector Machine

In order to implement a SVM, the e1071 library was used [10]. The cost and gamma parameters were tuned on the following grid:
Cost: 1 - 1e7
Gamma: 1e-5 - 1
The tuning range was so large because experiments on a subset of the overall dataset on a local machine failed to find a more localised search space over which the model performed well. Some parameter combinations would cause excessive runtime, therefore limiting the ability to search for an optimal range. This led to a total of 40 hours being necessary to execute the full experiment on EC2; however, on completion it was found that the best-performing individual model had required less than an hour to run, suggesting a better tuned search grid would require a similar running time to the subsequent methods.

B. Random Forest

By fitting numerous decision trees to the data and taking a consensus, Random Forests (RFs) often perform well with noisy data [11]. Previous studies have shown that extensive parameter tuning is often unnecessary to obtain good results [12]. Indeed, the parameters which can be meaningfully tuned are limited to the 'mtry', ' nodesize' and 'ntree'. These parameters represent the number of variables selected for each tree, the number of instances in the smallest node before terminating tree generation and the number of trees generated respectively [13]. The randomForest package [13] was used to train Random Forest regression models on the dataset and tune the parameters. The mtry and nodesize parameters were tuned on the following grid:
Mtry: 10 - 40
Nodesize: 1 - 7
Experimentation with a subset of the data indicated that these parameter ranges gave the best results. It also indicated that tuning the 'ntree' parameter made little difference to overall performance, hence the use of the default value only in the final experiment. Models were trained and evaluated on the full dataset using EC2, which required approximately 3 hours of computation time.

C. Deep Neural Network

To produce a predictive neural network, the 'deepLearning' function from the H2O package was used [14], [15]. This allowed the training of numerous networks with different parameter values. Due to the relatively large range of parameters available for tuning, a random search was carried out rather than a grid search. Parameters were randomly allocated for each candidate model from the following ranges:

Number of hidden layers: 1-4.
This was determined through experimentation on subsets of the overall dataset. Networks with more than 4 hidden layers performed poorly while requiring excessive computation time.

Number of nodes per hidden layer:

TABLE I: search space for number of hidden nodes per layer

Hidden Layer	1	2	3	4
Number of Nodes	10-100	10-100	10-80	5-30

Activation function: Rectifier, Rectifier With Dropout.
The rectifier type activation function is typical for regression models. The 'rectifier with dropout' activation function was also tried[16]

Epochs: 10.
Chosen as a compromise between computation speed and accuracy based on experimentation on subset of overall dataset.

Input dropout ratio: 0-0.1
This range provided reduced RMSE for given computation time during experimentation on subset of overall dataset.

L1: 1e-5 - 1e-2
This range provided reduced RMSE for given computation time during experimentation on subset of overall dataset.

L2: 1e-5 - 1e-2
This range provided reduced RMSE for given computation time during experimentation on subset of overall dataset.

Distribution: Gaussian
Other distributions were tried during the experimentation phase, however, the best performing models invariably used

a Gaussian distribution.

Adaptive learning rate: TRUE.
This provided reduced RMSE for given computation time during experimentation on subset of overall dataset.

Following experimentation on a small subset of the data to determine the above ranges for tuning parameters, the full training run was carried out on EC2. Computation time was limited to 4 hours, which resulted in the generation of 848 candidate models. This is significantly higher than the number of models generated by the SVM and Random Forest algorithms, however the benefit is offset by the larger parameter search space.

IV. Results

The performance for the best model obtained by each method was as follows:

TABLE II: Performance for each method

Prediction method	**RMSE**	**Parameters of best model**
Benchmark,	0.675	NA

SVM	0.582	cost=100
		gamma=1e-5

Random Forest	0.551	mtry=10
		nodesize=7

Deep Neural Network	0.582	hidden layers=3
		hidden nodes=(98,40,43)
		activation function = Rectifier
		rate=0.005
		l1=2.2e-4
		l2=1.4e-3

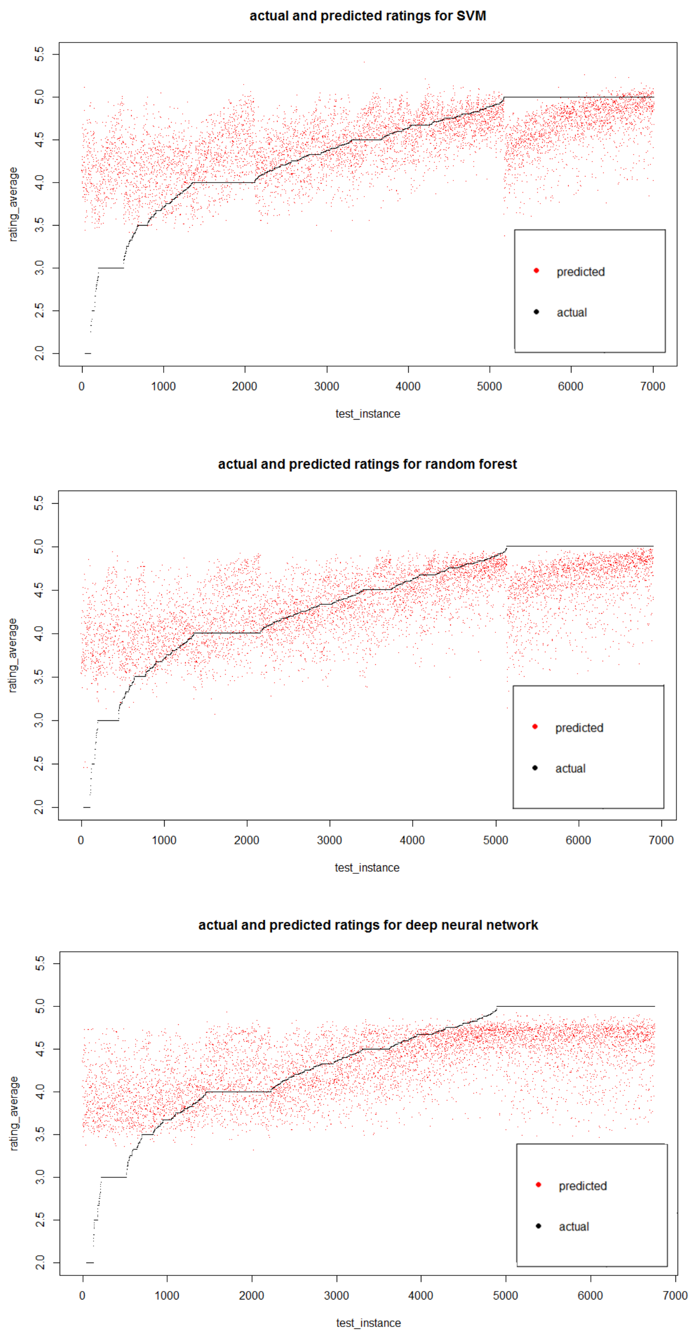
The Random Forest model was the best predictor for yarn rating, giving a RMSE of 0.551. It is notable that the neural network and SVM gave similar results, whereas the Random Forest model performed significantly better. Plotting the predicted and actual ratings for a single test set (obtained from the cross validation fold allocations) gives an indication of the strengths of each model (figure 3.)

As each experiment involved the generation of multiple candidate models, the performance of the non-winning candidate models can be plotted to assess each algorithm's sensitivity to parameter tuning.

Within the parameter ranges tested, the SVM produced many models with extremely poor performance (RMSE greater than 1), suggesting a narrower parameter range should have been searched. The Random Forest models outperformed the best SVM and Neural Network models even when sub-optimal parameters were used. This is illustrated in Figure 4.

The e1071 implementation for SVMs does not include the ability to view variables by importance so only the NN and RF outputs may be compared for important variables.The plots in figure 5 show the relative variable importance for each method.

Fig. 3: Actual and predicted results for all instances from the testing set of one validation fold



V. Discussion and Conclusions

The yarn ratings on Ravelry are provided by a range of artists and hobbyists, all with different demands for a yarn. What one user considers a good yarn (for example low cost but low quality) may be considered poor by another. In addition, since many yarns are produced by other hobbyists, there is a tendency for ratings to be artificially high [27]. These factors made prediction of a yarn's rating difficult and manifest as noise in the data.

Fig. 4: Performance of all models for full parameter search space

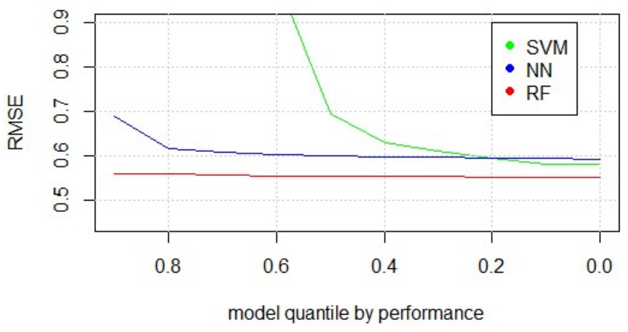
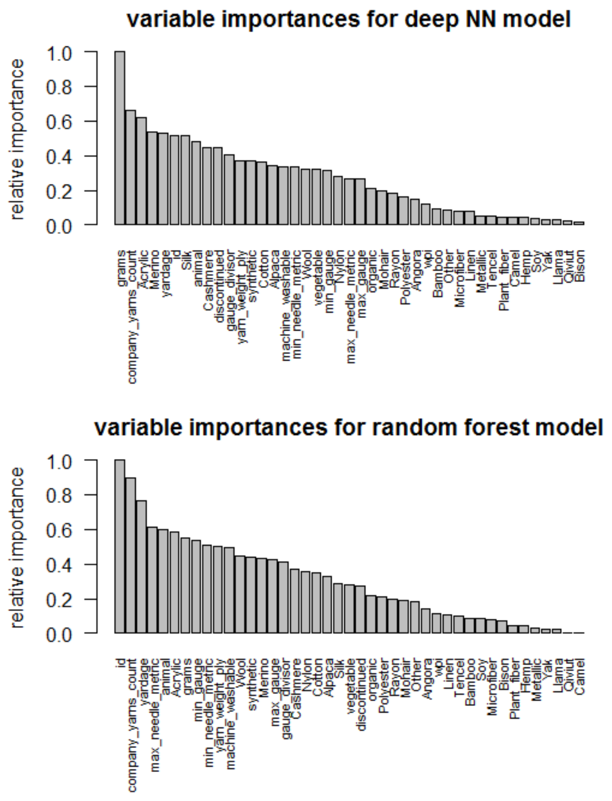


Fig. 5: Relative Variable Importances for NN and RF Methods



A. Model Performance

The Random Forest method significantly outperformed a tuned Neural Network and SVM. Even without parameter tuning, a Random Forest would have outperformed the best performing models from the other two methods. Therefore, it can be said that for the given parameter search space and computational time, a Random Forest algorithm provides the best means to predict the rating for a given yarn. This does not necessarily mean it is the best of the three methods since a more extensive or optimised parameter search may allow either of the other two methods to exhibit an improvement in performance [22]

Inspection of the plots in figure 3 suggests that the SVM and RF models had the tendency to overfit on the ‘id’ attribute (for these plots, the instances were ordered by ’id’ where the yarn ratings were equal - hence the tendency for predicted ratings to increase with instance number when yarn ratings are equal). The similarity in prediction pattern between the SVM and RF suggests a more optimised parameter search for the SVM method may have yielded better results. The NN models did not exhibit this behaviour, although this may simply be because the NN assigned the most importance to the ’grams’ attribute which may have been overfitted to a similar degree. The NN method suffered from limited accuracy due to a tendency to constrain predictions to a relativley limited range, where both the SVM and particularly the RF method yielded less constrained predictions.

Although the Random Forest method gave the best accuracy, the reduction in RMSE over simply predicting the mean rating value for each yarn was only 0.124, from 0.675. As discussed, noise in the data will always impose a limit on the performance of predictive models for this dataset. As this is the first known study of this dataset, any significant improvement over guessing the mean was considered a success; however there is little doubt that other models could provide improved performance and the minimum practically achievable RMSE remains an open challenge.

B. Variable Importance

Variable importance can only be compared between the NN and RF methods andis illustrated in figure 5. In both cases, properties relating to the yarn product itself, rather than its actual composition accounted for the majority of the varience in ratings. For the RF model, three variables are immediately apparent as having high importance, these are the yarn id - which may be used as a proxy for time the yarn has been on the database; ’company yarn count’ - which quantifies the number of other yarns the producing company already has on the database; and yardage - which is the overall length of yarn. None of these variables relate to a yarn’s actual composition which indicates that Ravelry users are more likely to consider the nature of the producing company and the physical dimensions of the product, rather than the composition of the yarn itself. After these attributes, importance appears to fall off in a roughly linear manner. The most important variable which relates to yarn composition is the ’animal’ variable (5th most important variable), which quantifies the amount of animal fibers in the yarn, this is followed by the ’acryllic’ variable (6th most important variable). This suggests that although animal fibers are popular, cheap acryllic yarns also attract good ratings.

Although the NN model gave poorer predictions on the yarn rating, it is interesting to note the variable importances used. Again, three variables are immediately apparent as being more important than others: ’grams’, ’company yarns count’ and ’Acryllic’. Subsequent importance dropoff is roughly linear, in keeping with the pattern observed from the RF. However, the NN model, assigned higher relative importance to the most important single variable than the RF model. Perhaps of more interest are the number of attributes relating to yarn fiber composition amongst the most important variables. Although product dimensions and company characteristics were of the most importance, the NN model assigned more importance to fiber composition than the RF model; with 'acryllic', 'merino', 'silk', 'animal' and 'cashmere' content all being in the top ten most important variables. With the exception of the 'acryllic' attribute, this suggests that after considering product dimensions and company characteristics, higher quality but more expensive yarns are generally favoured if the NN model is used for prediction. This difference also indicates that the NN model considered a fundementally different set of characteristics to be of importance to yarn rating, and therefore would merit further study in order to optimise this method for yarn rating prediction.

VI. Future Work

While the models produced by this study were successful in identifying trends between yarn characteristics and user rating, any predictions made from these models are highly prone to error; and no single variable, or combination of variables, can be said to have an overriding influence on the rating. Indeed, the noisy nature of the yarn data implies a ceiling on the accuracy of any model, no matter how well trained it is. However, the author acknowledges that improvements on the model accuracies presented here are certainly possible. To obtain improved results while keeping computation time/cost constant, the following may be investigated in the future:

A. *Improved computational efficiency*

In all cases, models were trained and tested using parallel code running on 36 virtual CPU cores. The first step to improve efficiency would be to optimise the parallelisation. For example, for the SVM models, a single thread could delay the entire training run by occupying a single CPU core for several hours. If the SVM algorithm itself was parallelised, rather than just the parameter tuning and cross validation, a speedup could be achieved. In addition, it is recommended to use holdout validation rather than 5 fold cross validation during the parameter tuning stage since for future studies as this would allow 5 times as many parameters to be searched for the same computation time, albeit with reduced confidence in the model accuracies. Cross validation may then be used to validate the final model, therefore providing a higher confidence for accuracy.

An additional approach would be to use GPU processing [17]. This would allow several thousand threads to be run simultaneously, therefore sampling from a much larger parameter grid and evaluation of a greater number of candidate models. GPU processing was not used for this study as libraries for GPU-optimised code are only available for limited applications in R currently. The deepLearning function in H2O may be run on a GPU through the deepwater package [23], however this functionality was not used in order to allow for a fair comparison between models. While it is certainly possible to manually create GPU-optimised functions in R, a better approach currently would be to use a dedicated package, such as TensorFlow or MXNet [18], [19].

B. *Parameter optimisation*

During generation of candidate neural network models, it was apparent that the method of randomly searching the parameter space was highly inefficient. Rather than a random or grid search (as used for the SVM and Random Forest models), some form of optimisation algorithm could potentially be used to select the best parameters. A genetic algorithm or simulated annealing are possible candidates and could be used in conjunction with GPU processing to ensure enough candidate models were generated to make the optimisation worthwhile [20], [21]. This would allow for optimisation of parameters which were not tuned in this study.

C. *Alternative models*

This study focused on three distinct algorithms. Variations of these methods, or different methods entirely may give improved results. In the case of SVMs, different kernel methods may be used to the radial basis function applied here [24]. Numerous varieties of neural networks are available in addition to the deep backpropogation network used in this study, which have been shown to exhibit excellent performance for certain problem types[26]. Alternatives ensemble learning methods to the Random Forest model include Boosting, Bagging and Stacking[24], [25]. Other, completely different, machine-learning methods are available which may out-perform the Random Forest. Unsupervised clustering methods, for example, may give further insight into which variables are the most important for determining yarn rating [25].

D. *Data processing*

The full dataset was not used for this study – approximately 52% was omitted due to NA values. This decision was made as a trade off between computation time and model accuracy. Future studies may, however, benefit from assigning values to these NA entries in order to make use of the full dataset. This would require more comprehensive data processing since many of the rows containing NA values are invalid entries to the database. Similarly, a number of text-based attributes were removed from the dataset. These contained information such as a written description of the yarn, and information on the supplying company. Consideration for these attributes could provide improved results but would require alternative methods of analysis.

References

[1] L. Hao, Q. Guan-xiong, L. Xiao-jiu, C. Ling and W. Yuxiu, Quality grade recognition of knitted yarns by support vector machines, *2010 International Conference on Computing, Control and Industrial Engineering*, Wuhan, 2010, pp. 49-51

[2] Z. Bo, Prediction of end breakage rates of cotton yarn in ring spinning processing by applying neural network approach and regression analysis theory, *2012 International Conference on System Science and Engineering (ICSSE)*, Dalian, Liaoning, 2012, pp. 555-558.

[3] R. Chattopadhyay, 4 - Artificial neural networks in yarn property modeling, In Woodhead Publishing Series in Textiles, edited by A. Majumdar, Woodhead Publishing, 2011, Pages 105-125, *Soft Computing in Textile Engineering*, ISBN 9781845696634, http://dx.doi.org/10.1533/9780857090812.2.105.

[4] A. Basu, 6 - Yarn engineering using an artificial neural network, In Woodhead Publishing Series in Textiles, Woodhead Publishing, 2011, Pages 147-158, *Soft Computing in Textile Engineering*, ISBN 9781845696634, http://dx.doi.org/10.1533/9780857090812.2.147.

[5] C. Forbes, Ravelry Application Developer and API License Agreement [online], Retrieved from http://www.ravelry.com/wiki/pages/Legal%20:%20Application%20Developer%20and%20API%20License%20Agreement, (accessed 24/04/2017)

[6] R.Wehrens, J.Kruisselbrink, 2017, Package 'Kohonen' [online], Retrieved from https://cran.r-project.org/web/packages/kohonen/kohonen.pdf (accessed 17/04/2017)

[7] Amazon, 2017, Amazon EC2 Instance Types [online], Retrieved from https://aws.amazon.com/ec2/instance-types/, (accessed 24/02/2017)

[8] R. Calaway, S. Weston, D. Tenenbaum, Revolution Analytics, 2015, Package 'doParallel' [online], Retrieved from https://cran.r-project.org/web/packages/doParallel/doParallel.pdf (accessed 19/04/2017)

[9] Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 693–701, 2011.

[10] D. Meyer, E. Dimitriadou, K. Hornik, A. Weingessel, F. Leisch, C. Chang, C. Lin, 2017, Package 'e1071' [online], Retrieved from http://cran.rproject.org/web/packages/e1071/index.html. (accessed 22.04.17).

[11] A. Folleco, T. M. Khoshgoftaar, J. Van Hulse and L. Bullard, Software quality modeling: The impact of class noise on the random forest classifier, *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, Hong Kong, 2008, pp. 3853-3859.

[12] M. Prior and T. Windeatt, Parameter Tuning using the Out-of-Bootstrap Generalisation Error Estimate for Stochastic Discrimination and Random Forests, *18th International Conference on Pattern Recognition (ICPR'06)*, Hong Kong, 2006, pp. 498-501.

[13] L. Brieman, L. Cutler, A. Liaw, M. Wiener, 2015, Package 'randomForest' [online], Retrieved from https://cran.r-project.org/web/packages/randomForest/randomForest.pdf, (accessed 16/04/2017)

[14] The H2O.ai team, 2017, 'Package h2o' [online], Retrieved from https://cran.r-project.org/web/packages/h2o/h2o.pdf, (accessed 23/04/2017)

[15] A. Candel, J. Lanford, E. Ledell, V. Parmar, A.Arora, Deep Learning With h2o [online], Third Edition, H2O.ai.inc, Mountain View, USA, Available at https://h2o-release.s3.amazonaws.com/h2o/rel-slater/9/docs-website/h2o-docs/booklets/DeepLearning_Vignette.pdf

[16] C.-C. Jay Kuo, Understanding convolutional neural networks with a mathematical model, *Journal of Visual Communication and Image Representation*, Volume 41, November 2016, Pages 406-413

[17] M. K. Zhou, F. Yin and C. L. Liu, GPU-Based Fast Training of Discriminative Learning Quadratic Discriminant Function for Handwritten Chinese Character Recognition, *2013 12th International Conference on Document Analysis and Recognition*, Washington, DC, 2013, pp. 842-846.

[18] Abadi et al, 2015, TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems, arXiv:1603.04467

[19] Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. arXiv:1512.01274, 2015a.

[20] E. Elyan. and M.M.Gaber, 2017. A genetic algorithm approach to optimising random forests applied to class engineered data. *Information sciences* [online], 384, pages 220-234. Available from: https://dx.doi.org/10.1016/j.ins.2016.08.007

[21] A. M. Coroiu, Tuning model parameters through a Genetic Algorithm approach, *2016 IEEE 12th International Conference on Intelligent Computer Communication and Processing (ICCP)*, Cluj-Napoca, 2016, pp. 135-140.

[22] G. James, D. Witten, T. Hastie, R. Tibshirani, 2013, *An Introduction to Statistical Learning*, New York, Springer.

[23] A. Candel, Deep Learning in H2O using Native GPU Backends, 2017, Github repository, available at https://github.com/h2oai/deepwater

[24] G. Williams, 2011, *Data Mining With Rattle and R*, New York, Springer

[25] A. Witten, E. Frank, M. Hall, 2011, *Data Mining: Practical Machine Learning Tools and Techniques*, Third Edition, Burlington, Elsevier

[26] L. Deng, G. Hinton and B. Kingsbury, New types of deep neural network learning for speech recognition and related applications: an overview, *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, Vancouver, BC, 2013, pp. 8599-8603.

[27] J. Forbes, 2016, Ravelry's Wiki : A Community Edited Guide [online], Retrieved from http://www.ravelry.com/wiki/pages/HomePage (accessed 15/04/2017)