# A Comparison of Model Selection Approaches with Cross-validation

## Simple Cross-validation and Nested Cross-Validation

*DSTI S19 Artificial Neural Network Class Project*

DSTI S19 Cohort; Data Science Course; Motoharu DEI

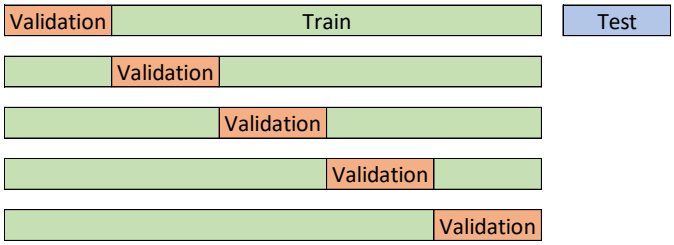
October 20th, 2019

## 1. Introduction

Cross-validation is a common approach to select the best performing model and parameters in every data science modeling. Its goal is to minimize training error while controlling the extent of overfitting [1]. The cross-validation is particularly prevailing when tuning the model's hyper parameters of a model [2].

When we are interested in selection of the best performing model along with choosing the best hyperparameters at the same time, two extensions can possible come into our consideration:

- ***CV Option 1***: to cross-validate the possible combinations of models and hyper parameters, and choose the best combination of model and hyperparameters together (*Figure 1*), and
- ***CV Option 2***: to cross-validate on two hierarchical layers (also known as '*nested cross-validation* [1]'): inner loop for selection of the best hyper parameter in a model and outer loop for selection of the best model (*Figure 2*).




Loop to select best combination of model and hyper parameters:

1. For training data in each split, train the model with each combination of model and hyper parameters.
2. Average the validation scores and choose the best combination of model and hyper parameters with the highest scores.
3. Retrain the selected model using the whole data.
4. Calculate the test score on the test data.

*Figure 1: CV Option 1 – Selecting the best combination of model and hyperparameters in single cross-validation.*

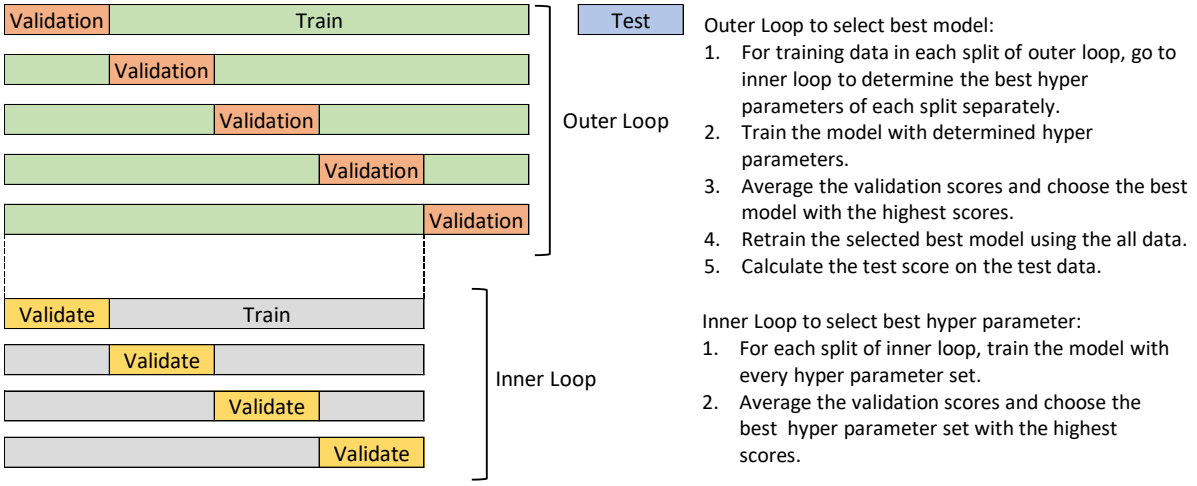



*Figure 2: CV Option 2 – Selecting the best hyperparameters and model in two-layered cross-validation ('nested cross-validation').*

In this project, we are going to see how these two approaches behave differently in model and hyperparameter selection to the same data set. To test this, I used a data set from Kaggle Dataset, "FIFA 19 complete player dataset" (see Appendix-2).

We will also look at how the selections are stable with 10 randomly shuffled data (without replacement) by repeating the same workflow of CV Option 1 and CV Option 2 above.

Any analysis done is based in this report on Python code (see version information in Appendix-1) and is open sourced in a Jupyter Notebook on author's Github page here: https://github.com/daydreamersjp/DataScienceTechInstitute/blob/master/Artificial-Neural-Network/DSTI%20ANN%20Project%20FIFA19%20Data.ipynb.

## 2. Method

I used the "FIFA 19 complete player dataset" (see Appendix-2) with 17,907 valid data and 31 selected explanatory variables (see Appendix-3, Appendix-4), to construct models with hyperparameter tuning.

For preprocessing of the variables:

- The target variable ("market value of each player") was transformed by logarithm in base 10, which made the variable distribution symmetric and bell curve-like. See histogram in Appendix-4.
- The explanatory variables were scaled by Min-Max to range of [0, 1] based on training data in each cross-validation fold, and the same transformation formula was applied to the validation data or test data. The selection of Min-Max is appropriate enough as no outstanding outlier was observed. See histogram in Appendix-4.
- I produced additional explanatory variables by adding polynomial features up to the second degree of the original variables (square and interaction). This uplifted the prediction power from the level of R-squared=70% to 90% in ordinary linear regression.

After these preprocessing, the number of all explanatory variables used in the model was 992.

*Table 1* summarizes the models and hyperparameters explored. I selected the candidate models from each one in *linear regression family* ('Linear Regression with LASSO'), *decision tree family* ('lightGBM'), *neural network family* ('Multi-Layer Perceptron with relu'), and *margin maximization family* ('Support Vector Regression'). I chose two important hyperparameters and two possible values in each of them (except linear regression with LASSO where the model inherently has only one hyperparameter.) I ran the grid search cross-validation when choosing the best hyperparameters.

*Table 1: Summary of the model and hyperparameters options explored in this project.*

| Model<br>*(Python package name used)* | Hyperparameters<br>*(parameter name in Python function)* | Parameter Values |
|---|---|---|
| Linear Regression with LASSO<br>*(scikit-learn)* | Constant that multiplies the L1 term.<br>*(alpha)* | [0.0001, 0.01] |
| lightGBM<br>*(lightgbm)* | Minimal number of data in one leaf.<br>*(min_data_in_leaf)* | [100, 200] |
| | Max number of leaves in one tree.<br>*(num_leaves)* | [40, 80] |
| Multi Layer Perceptron with ReLU<br>*(scikit-learn)*<br>` | Whether to use early stopping.<br>*(early_stopping)* | [True, False] |
| | Number of neurons in each hidden layer. Having two numbers means there are two layers.<br>*(hidden_layer_sizes)* | [(100), (50, 50)] |
| Support Vector Regression<br>*(scikit-learn)* | Penalty parameter of the error term.<br>*(C)* | [1.0, 2.0] |
| | Epsilon in the epsilon-SVR model.<br>*(epsilon)* | [0.1, 0.2] |

Since the focus of this project was a comparison of two cross-validation approaches, not to generate an accurate prediction model, it is worth noting the selection of the models, hyperparameters, and parameter options were not necessarily complete, and choosing another options may result in different numeric outputs, although the key statement in this report are expected to be still applicable.

The 20% of the data set was held out for a test set to check the score of the finally selected model and parameters. For any cross-validations used in CV Option 1 and CV Option 2 (inner loop and outer loop), I used the number of folds = 5.

Every model evaluation was based on R-squared on the validation fold or test data.

Finally, to test the stability of the model selection and parameter, I randomly shuffled the data set without replacement and repeated the same model train and evaluation routines 10 times. Data shuffle was without replacement because the repetition of the same data rows in with-replacement shuffle would possibly cause data leakage[3] and distortion of the results. In this study, I only reran the lightGBM and MLP models because they were the only two closely competing highly.

## 3. Results

### 3.1 Results of CV Option 1: Selecting the best set of models and hyperparameters in single CV

Table 2 below shows all the results of the combinations of models and hyperparameters from CV Option 1. Multi-layer perceptron with early_stopping=True and hidden_layer_sizes=(50,50) was selected with the best CV average score 0.935 and the test data score 0.932, seemingly with less possibility in overfitting. The score was fairly close to the one of the best lightGBM model (0.934), and was within ± standard deviation. Therefore, we can say that these two models had similar predictive power on this data set.

*Table 2: Results with CV Option 1 (Total run time was 11.70 minutes.)*

| Model (Hyperparameters) | Run Time (sec) | CV Score Average | CV Score SD | Test Score |
|---|---|---|---|---|
| LASSO(alpha=0.0001) | 14.40 | 0.907 | 0.003 | |
| LASSO(alpha=0.01) | 3.41 | 0.706 | 0.007 | |
| lightGBM(min_data_in_leaf=100, num_leaves=40) | 25.37 | 0.934 | 0.003 | |
| lightGBM(min_data_in_leaf=100, num_leaves=80) | 33.33 | 0.934 | 0.004 | |
| lightGBM(min_data_in_leaf=200, num_leaves=40) | 19.88 | 0.931 | 0.004 | |
| lightGBM(min_data_in_leaf=200, num_leaves=80) | 20.90 | 0.931 | 0.003 | |
| MLP(early_stopping=True, hidden_layer_sizes=(100)) | 45.28 | 0.930 | 0.004 | |
| **MLP(early_stopping=True, hidden_layer_sizes=(50,50))** | **27.05** | **0.935** | **0.002** | **0.932** |
| MLP(early_stopping=False, hidden_layer_sizes=(100)) | 47.41 | 0.930 | 0.005 | |
| MLP(early_stopping=False, hidden_layer_sizes=(50,50)) | 22.93 | 0.917 | 0.018 | |
| SVR(C=1.0, epsilon=0.1) | 148.08 | 0.847 | 0.009 | |
| SVR(C=1.0, epsilon=0.2) | 81.32 | 0.840 | 0.008 | |
| SVR(C=2.0, epsilon=0.1) | 139.17 | 0.870 | 0.008 | |
| SVR(C=2.0, epsilon=0.2) | 73.29 | 0.862 | 0.008 | |

*Figure 3* below is the scatter plot comparing the *true log10(target value)* and *predicted log10(target value)* to each row of test data based on the final multi-layer perceptron. The predicted values are close to the true values in almost any level of value.

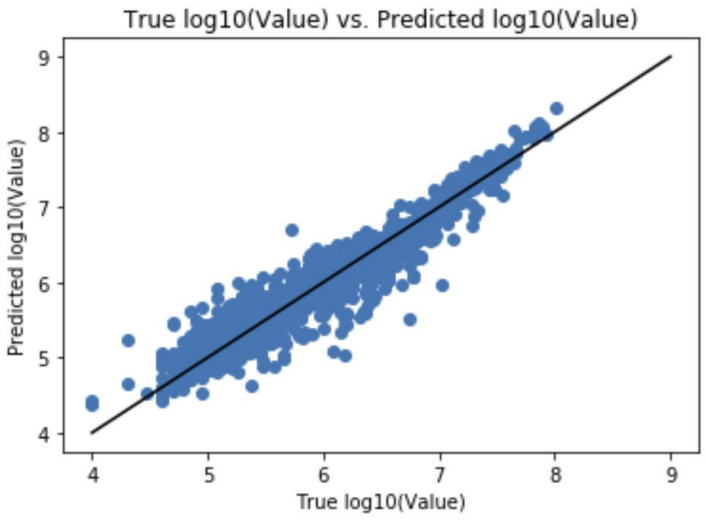



*Figure 3: Scatter plot of true target value (log10 transformed) and predicted target value (log10 transformed) in test data with best mode in CV Option 1 (and commonly in CV Option 2 as stated below.)*

## 3.2 Results of CV Option 2: With nested CV, selecting the best hyperparameters in inner-loop CV and the best model in outer-loop CV

*Table 3* shows all the results obtained using the CV option 2, a nested cross-validation approach. Again, Multi-layer perceptron was selected with a small margin to lightGBM within the score standard deviation. Multi-layer perceptron was the only model which had different selection in the result of inner-CV, with three separate choices, (False, (50,50)), (True, (50,50)), and (True, (100)).

As a final model, I reran the multi-layer perceptron on single cross-validation to select the hyperparameters and calculated the score on test data. Since the cross-validation set up is the same as the CV Option 1, the selection of the hyper parameters and the test score were the same as well, which were early_stopping=True and hidden_layer_sizes=(50,50), and 0.932.

Total run time 39.70 minutes was close to 4 times long of CV Option 1, 11.70 minutes, which was not surprising since I ran the 5 cross-validations in each of outer loop folds while CV Option 1 ran only once in each fold.

*Table 3: Results with CV Option 2 (Total run time was 39.70 minutes.)*

| Model | Run Time (sec) | Outer-CV Score Average | Outer-CV Score SD | Best Hyperparams in Inner-CV | Final Rerun Model and Params and Test Score |
|---|---|---|---|---|---|
| LASSO | 57.11 | 0.907 | 0.003 | *alpha:*<br>#1: alpha=0.0001<br>#2: alpha=0.0001<br>#3: alpha=0.0001<br>#4: alpha=0.0001<br>#5: alpha=0.0001 | |
| lightGBM | 310.28 | 0.934 | 0.003 | *(min_data_in_leaf, num_leaves):*<br>#1 (100, 40)<br>#2 (100, 40)<br>#3 (100, 40)<br>#4 (100, 40)<br>#5 (100, 40) | |
| **MLP** | **611.49** | **0.935** | **0.001** | ***(early_stopping, hidden_layer_sizes):***<br>**#1: (False, (50, 50))**<br>**#2: (True, (50, 50))**<br>**#3: (True, (50, 50))**<br>**#4: (True, (50, 50))**<br>**#5: (True, (100))** | ***(early_stopping, hidden_layer_sizes)***<br>***=* (True, (50, 50))**<br><br>**Test Score: 0.932** |
| SVR | 1,403.43 | 0.870 | 0.008 | *(C, epsilon):*<br>#1: (2.0, 0.1)<br>#2: (2.0, 0.1)<br>#3: (2.0, 0.1)<br>#4: (2.0, 0.1)<br>#5: (2.0, 0.1) | |

### 3.3 Results of the Same CV Runs Using Shuffled Data Set without Replacement

Table 4 shows all the results of the runs with shuffled data set without replacement. In 5 shuffles out of 10, both CV options chose the same model and hyperparameters as the best final model, while about other 5 shuffles, Option 1 preferred multi-layer perceptron and Option 2 preferred lightGBM. There were no outstanding differences in CV score averages and CV score standard deviations in both CV options. In 4 cases out of 5 which preferred the different model, the test scores were a little better in CV Option 1 than in CV Option 2, but not far enough beyond the level of standard deviation.

*Table 4: Results by 10 shuffled data set (Total run time was around 2 hours in CV Option 1 and 6 to 7 hours in CV Option 2.)*

| | CV Option 1 | | | | CV Option 2 | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Shuffle # | Best Model and Best Hyperparameters | CV Score Average | CV Score SD | Test Score | Best Model | CV Score Average | CV Score SD | Best Hyperparameters in Final Rerun | Test Score |
| 1 | MLP(early_stopping=False, hidden_layer_sizes=(50,50)) | 0.935 | 0.003 | 0.935 | MLP | 0.934 | 0.001 | early_stopping=False, hidden_layer_sizes=(50,50) | 0.935 |
| 2 | lightGBM(min_data_in_leaf=100, num_leaves=80) | 0.933 | 0.003 | 0.943 | lightGBM | 0.933 | 0.004 | min_data_in_leaf=100, num_leaves=80 | 0.943 |
| 3 | lightGBM(min_data_in_leaf=100,num_leaves=40) | 0.934 | 0.003 | 0.935 | lightGBM | 0.934 | 0.003 | min_data_in_leaf=100, num_leaves=40 | 0.935 |
| 4 | MLP(early_stopping=True, hidden_layer_sizes=(50,50)) | 0.935 | 0.001 | 0.940 | lightGBM | 0.935 | 0.002 | min_data_in_leaf=100, num_leaves=80 | 0.936 |
| 5 | MLP(early_stopping=True, hidden_layer_sizes=(100)) | 0.935 | 0.005 | 0.939 | lightGBM | 0.935 | 0.004 | min_data_in_leaf=100, num_leaves=40 | 0.934 |
| 6 | MLP(early_stopping=True, hidden_layer_sizes=(50,50)) | 0.936 | 0.003 | 0.938 | lightGBM | 0.934 | 0.002 | min_data_in_leaf=100, num_leaves=80 | 0.938 |
| 7 | ightGBM(min_data_in_leaf=100,num_leaves=80) | 0.935 | 0.001 | 0.934 | lightGBM | 0.935 | 0.001 | min_data_in_leaf=100, num_leaves=80 | 0.934 |
| 8 | MLP(early_stopping=True, hidden_layer_sizes=(50,50)) | 0.937 | 0.003 | 0.930 | lightGBM | 0.936 | 0.004 | min_data_in_leaf=100, num_leaves=40 | 0.929 |
| 9 | MLP(early_stopping=True, hidden_layer_sizes=(50,50)) | 0.936 | 0.003 | 0.932 | lightGBM | 0.935 | 0.003 | min_data_in_leaf=100, num_leaves=80 | 0.937 |
| 10 | MLP(early_stopping=True, hidden_layer_sizes=(50,50)) | 0.933 | 0.004 | 0.941 | MLP | 0.933 | 0.004 | early_stopping=True, hidden_layer_sizes=(50,50) | 0.941 |

## 4. Discussion

There was no outstanding advantage found about either of CV Option 1 or CV Option 2 in my study in terms of validation and test score. Two models, lightGBM and multi-layer perceptron, performed well enough closely in any CV validation scores in any shuffles, and I ended up with selecting either model by chance within possible variation of scores. One conceptual difference is that CV Option 2 selects the best performing model without specifying the hyperparameters, while CV Option 1 selects the best combination of model and hyperparameters at once. Therefore, Option 2 is expected to choose a model performing averagely well if new data has similar distribution to the training data, provided their hyperparameters are tuned again on the new data (although it was not evident in my study.) Drawback of this is that the final model in CV Option 2 must find the best

hyperparameters with additional cross-validation process when it is applied to a new training data, which requires additional computation time.

Another apparent disadvantage of CV Option 2 was that it took approximately four times longer to compute, which was just as presumed in advance because there were 5 cross-validation evaluations in each of outer loop fold while CV Option 1 ran only once in each fold. This difference may pose a significant discouragement to CV Option 2 particularly when the model space or hyperparameter space are large.

In coding practice, CV Option 2 has more to evaluate and compare, which leads to more operations and possible confusion about the data pipelining.

I used grid search CV, which searches every combination of model and hyperparameters. When the number of hyperparameters are more than 3, it is more common to use randomized search CV instead of grid search because the number of possible searches gets too many to try all. Naïve randomized search CV tests parameters selected randomly in the probability determined by the size of search space. Therefore, if a model has a smaller number of hyperparameters, in CV Option 1 their combinations of model and hyperparameters will be selected fewer times in randomized search CV even when that model is potentially the most performant if its hyperparameters are searched enough. When it could be a case, it will be more appropriate to stratify the search by model instead of doing it at complete random. This issue will not happen in CV Option 2 by nature because it will compare the models in outer loop and treats every model equally.

## 5. Conclusion

We compared two cross-validation approaches to select the best model and hyperparameters:

- to select a combination of model and hyperparameters at once in a single cross-validation; and
- to select hyperparameters and model in inner loop and outer loop of nested cross-validation.

Although the latter CV approach takes a lot longer to compute and needs careful pipelining, no apparent advantage in model scores were observed in my study. When doing randomized search CV, the former CV approach may imbalance the number of searches in specific models with fewer hyperparameters. Then, stratified search by model to balance the number of searches should be more appropriate.

## Reference


1. Cawley, Gavin C.; Talbot, Nicola L. C. (2010). "On Over-fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation". *Journal of Machine Learning Research: 2079–2107*.
2. Neale C (January 5th, 2019). “Hyper-Parameter Tuning and Model Selection, Like a Movie Star” https://towardsdatascience.com/hyper-parameter-tuning-and-model-selection-like-a-movie-star-a884b8ee8d68. *Towards Data Science*.
3. Kaufman, Rosset, Perlich (2011). “Leakage in Data Mining: Formulation, Detection, and Avoidance”. *KDD’11, August 21 – 24, 2011*.

# Appendix

## Appendix-1 Version Information

Here is the summary of version information about key components in the analysis.

*Table A-1: Key version information used in the project.*

| Packages | Version |
|---|---|
| Python | 3.6.8 |
| pandas | 0.25.1 |
| numpy | 1.16.0 |
| matplotlib | 3.0.2 |
| scikit-learn | 0.20.2 |
| lightgbm | 2.2.1 |

## Appendix-2 Data Source

The data was obtained from Kaggle Dataset. The "FIFA 19 complete player dataset" was prepared by Karen Gadiya at December 21st, 2018, and one of the datasets earning the most votes at the date of this report. The data comes from a football video game "FIFA 19", therefore, the data is entirely generated in the game, not produced in real world, although it reflects the actual players' attributes (name, nationality, etc.) and their actual specifications as football players. It has 18,207 rows and 88 columns, one row represents one football player, one column represents a spec of the player as well as an attribute of them such as age, nationality, position etc.

See the link to the data (https://www.kaggle.com/karangadiya/fifa19) for further information.

## Appendix-3 Data Screening Water Flow

The data size used in the analysis is 17,907, screened from the original data size of 18,207 based on data quality. Below table represents how the size was determined.

*Table A-2: Data waterfall from the original data to the data used in the analysis.*

| Data Size | Reason of Drop |
|---|---|
| 18,207 | NA (original data size) |
| 18,159 | Missing in player's skill columns |
| 17,907 | Missing in player's market Values |

Since the data values are something generated in the game as detailed in Appendix-2, the data is already clean and as many as 98% out of the original data rows are already complete enough for the analysis.

## Appendix-4 Details of Data Columns

The data used in the analysis has one column for the target variable and 31 columns for the explanatory variables. I chose the player's market value as a target variable which ranges from 10 thousand euro to 119 million euro, and other numeric variables as explanatory variables which represent each player's specifications. Although it may underpin some marginal accuracy to add categorical variables, I preferred the numeric variables because it would cause less additional data preprocessing and having higher accuracy is not necessarily my focus in this project.

Here is the summary of the data columns.

*Table A-3: Description of explanatory variables and target variable used.*

| Variable Names | Description | Data Type | Mean | SD | Min | Max |
|---|---|---|---|---|---|---|
| Target variable to predict: | | | | | | |
| Value (Unit: Euro) | Player's market value | Integer | 2,450 K | 5,633 K | 10 K | 118.5 M |
| Explanatory variables: | | | | | | |
| Age | Player's age | Integer | 25.10 | 4.66 | 16 | 45 |
| SkillMoves | Rating about player's skill in moves | Integer | 2.36 | 0.76 | 1 | 5 |
| Crossing | Extent of player's skill in crossing | Float | 49.75 | 18.35 | 5.0 | 93.0 |
| Finishing | Extent of player's skill in finishing | | 45.59 | 19.51 | 2.0 | 95.0 |
| HeadingAccuracy | Extent of player's accuracy in heading | | 52.30 | 17.36 | 4.0 | 94.0 |
| ShortPassing | Extent of player's skill in short passing | | 58.72 | 14.67 | 7.0 | 93.0 |
| Volleys | Extent of player's skill in volleys | | 42.94 | 17.69 | 4.0 | 90.0 |
| Dribbling | Extent of player's skill in dribbling | | 55.42 | 18.90 | 4.0 | 97.0 |
| Curve | Extent of player's skill in curve | | 47.22 | 18.38 | 6.0 | 94.0 |
| FKAccuracy | Extent of player's accurary in free kick | | 42.88 | 17.48 | 3.0 | 94.0 |
| LongPassing | Extent of player's skill in long passing | | 52.73 | 15.31 | 9.0 | 93.0 |
| BallControl | Extent of player's skill in ball control | | 58.42 | 16.66 | 5.0 | 96.0 |

| | | | | | | |
|---|---|---|---|---|---|---|
| Acceleration | Extent of player's skill in acceleration | | 64.62 | 14.93 | 12.0 | 97.0 |
| SprintSpeed | Extent of player's skill in sprint speed | | 64.74 | 14.65 | 12.0 | 96.0 |
| Agility | Extent of player's skill in agility | | 63.54 | 14.76 | 14.0 | 96.0 |
| Reactions | Extent of player's skill in reactions | | 61.82 | 9.02 | 21.0 | 96.0 |
| Balance | Extent of player's skill in balance | | 63.97 | 14.15 | 16.0 | 96.0 |
| ShotPower | Extent of player's skill in shot power | | 55.49 | 17.21 | 2.0 | 95.0 |
| Jumping | Extent of player's skill in jumping | | 65.12 | 11.83 | 15.0 | 95.0 |
| Stamina | Extent of player's skill in stamina | | 63.22 | 15.88 | 12.0 | 96.0 |
| Strength | Extent of player's skill in strength | | 65.33 | 12.55 | 17.0 | 97.0 |
| LongShots | Extent of player's skill in long shots | | 47.13 | 19.25 | 3.0 | 94.0 |
| Aggression | Extent of player's skill in aggression | | 55.88 | 17.35 | 11.0 | 95.0 |
| Interceptions | Extent of player's skill in interceptions | | 46.69 | 20.69 | 3.0 | 92.0 |
| Positioning | Extent of player's skill in positioning | | 50.00 | 19.52 | 2.0 | 95.0 |
| Vision | Extent of player's skill in vision | | 53.45 | 14.11 | 10.0 | 94.0 |
| Penalties | Extent of player's skill in penalties | | 48.55 | 15.69 | 5.0 | 92.0 |
| Composure | Extent of player's skill in composure | | 58.65 | 11.42 | 3.0 | 96.0 |
| Marking | Extent of player's skill in marking | | 47.26 | 19.87 | 3.0 | 94.0 |
| StandingTackle | Extent of player's skill in standing tackle | | 47.68 | 21.65 | 2.0 | 93.0 |
| SlidingTackle | Extent of player's skill in sliding tackle | | 45.64 | 21.27 | 3.0 | 91.0 |

And their histograms are in Figure A-1 below. Note the variable “Value” (player’s market value, a target variable) was already transformed by log10 in this histogram.

*Figure A-1: Histogram of all variables used in the analysis.*