Problem: Given various performance statistics from a new player, can we fabricate a model to determine their overall grading from 0-100 (‘ova’).

> Original shape (17125, 107)

I dropped columns that were numerically redundant (stats that already were counted at each “grouping”, including the total stats) and would not affect the target from a categorical standpoint (name, id, wage, etc.). I believe that other attributes like value/wages from the perspective of the PROBLEM do not affect the target, more so the physical attributes of the players themselves.

> New shape (17125, 25)

Used info() to show both dtypes and null count

Next to work on changing some fields to their correct type

age = categorical/object

hits = categorical is correct but the field needs cleaning

change values ending in K to actual numbers

then group values by 0-99,100-249, 250-500,500-999,1000+

Then to fill NA’s in a/w and d/w (might drop the records)

Only 0.5% of each column were NA’s. Used ffill to replace/distribute this into the categories for each

Check for multicollinearity, observations:

‘bov’ and target ‘ova’ are very similar.

‘def’ and ‘defending’ are very similar.

‘dri’ and ‘mentality’ had many instances of high collinearity with other columns, dropped both.

> New shape (17125, 20)

Check distribution of the numerical columns

Define columns related to numerical columns

Anomalies: Attacking, 2 peaks. Skill, 2 peaks. Movement, right skewed. Defending, 2 peaks. Goalkeeping, high concentration on low values. Base Stats, random

Data transformation: Tried various transformation techniques bu log trans seemed to work best in most cases

Goalkeeping – log trans.

Sho – log trans.

Defending – tried sq.rt and boxcox transformations but log trans made the most significant difference in making the distribution more regular.

Dropped original columns for Goalkeeping, Sho and Defending

Categoricals:

- Height : used string functions to replace and split the inches and foot indicators, and then multiplied the feet by 12 and added inches to make all records into inches. Then, grouped them together to make less categories

- Weight: mapped a function to remove “lbs” from records to then map grouped categorical values into a new column “weight\_group”

- Age: Early in my code I identified that this feature, though numerical in nature, is categorical so I converted the column to string. This helped when I was checking for distributions of numercials, etc. as I was focusing on dtypes that were np.number. However, when it came time to group them, I needed to convert the values back to int in my function, then create and map the age group to a new column “age\_group” based on their value.

- Charts helped to show which categories needed to be adjusted to make a more even distribution.

- Barplots helped to illustrate the relationship between categorical columns and the target ‘ova’.

Encoded the categorical data using pd.get\_dummies but when I went to do the linear regression, I received an error that the column titles were not of the same dtype so I changed the method to use the OneHotEncoder.

I ran 3 different types of scalers (MinMax, Standard and Normalizer) to try to see what would give me the best result.

Results: MinMax, Standard and Normalizer)

A table with numbers and symbols

Description automatically generated