Concrete Strength Prediction with R: A Machine Learning Workflow for Compressive Strength Regression

End-to-End Modeling with workflow_set and Visual Insights via patchwork

Damiano Pincolini

2024-11-23

Table of contents

1. PREFACE **2**
1.1. Project goal . . . 2
1.2. Loading packages . . . 2
1.3. Data loading and content analysis . . . 2
1.4. Dataset analysis . . . 3
1.4.1. Dataset structure, datatype and quality analysis . . . 3
1.4.2. Data featuring . . . 5
1.5. Data partitioning . . . 6

2. EXPLORATIVE DATA ANALYSIS **6**
2.1 Feature analysis . . . 6
2.1.1. Target analysis . . . 6
2.1.2. Univariate predictors analysis . . . 7
2.1.3. Multivariate predictors analysis . . . 9
2.2. Conclusions . . . 9

3. TRAINING AND TESTING ML MODELS WITH WORKFLOW_SETS PACKAGE 10
3.1. Training . . . 11
3.1.1. Setting cross validation and metrics selection . . . 11
3.1.2. Preprocessing recipes . . . 11
3.1.3. Model Specifications . . . 11
3.1.4. Workflow Sets . . . 12
3.1.5. Tuning and selection best performing workflow and hyperparameters . . . 12

3.2. Testing 13
3.2.1. Performance on the test set 13
3.2.2. Residuals analysis 15

CONCLUSIONS **16**
1. About the impact of dataset preprocessing on models' performance 16
2. About the effectiveness of the packages used throughout this work 16

1. PREFACE

1.1. Project goal

The overall goal of this project is to find out an efficient regression model trained and tested on a dataset available on Kaggle.com (https://www.kaggle.com/datasets/prathamtripathi/regression-with-neural-networking). Beside this, there are a couple of more specific aims to be set:

1. evaluate the impact of applying preprocessing steps either to every variable or to a limited group of them,
2. optimizing data visualization in order to increase the overall readability of the document.

As far as point 1 is concerned, the main focus is on usage of the workflow_set package, while for third point patchwork and kableExtra packages are involved.

To keep the whole process fast and fluent, I will not use a wide set of engines since the main focus is on how to manage preprocessing and to improved data visualization.

1.2. Loading packages

I avoid recalling the script that loads packages; I just point out the choiche of the tidymodels' ecosistem along with kableExtra and patchwork.

1.3. Data loading and content analysis

After downloading the .csv file from kaggle.com (see link above) and storing, I have saved it into R environment as "DataOrigin" dataset.

1.4. Dataset analysis

1.4.1. Dataset structure, datatype and quality analysis

With a help from SmartEDA and kableExtra packages, it's possible to produce two tables to have a bird's eye view on DataOrigin dataset.

Descriptions	Value
Sample size (nrow)	1030
No. of variables (ncol)	9
No. of numeric/interger variables	9
No. of factor variables	0
No. of text variables	0
No. of logical variables	0
No. of identifier variables	0
No. of date variables	0
No. of zero variance variables (uniform)	0
%. of variables having complete cases	100% (9)
%. of variables having >0% and <50% missing cases	0% (0)
%. of variables having >=50% and <90% missing cases	0% (0)
%. of variables having >=90% missing cases	0% (0)

The dataset content appears clean: there are eight predictors and a target variable (CompressiveStrenght), all featuresare numerical and there's no missing value. It doesn't seem necessary to trasform any datatype from numeric to factor.

Index	Variable_Name	Variable_Type	Sample_n	Missing_Count	Per_of_Missing	No_of_distinct_values
1	Cement	numeric	1030	0	0	278
2	BlastFurnaceSlag	numeric	1030	0	0	185
3	FlyAsh	numeric	1030	0	0	156
4	Water	numeric	1030	0	0	195
5	Superplasticizer	numeric	1030	0	0	111
6	CoarseAggregate	numeric	1030	0	0	284
7	FineAggregate	numeric	1030	0	0	302
8	Days	numeric	1030	0	0	14
9	CompressiveStrength	numeric	1030	0	0	845

Also from this view the dataset seems very "smooth".

I want to have a fast glance to the main statistics in order to understand if there could be some weird value. I'll simply use the basic summary() command.

The ranges between minimun and maximum values are generally pretty visible considering that (a part from days and Strenght) the measure is the same (kg/m3).

	variable	min	max	mean	median	quartile1	quartile3	SD	IQR
2	BlastFurnaceSlag	0.00	359.4	73.90	22.00	0.00	142.95	86.28	142.95
1	Cement	102.00	540.0	281.17	272.90	192.38	350.00	104.51	157.62
6	CoarseAggregate	801.00	1145.0	972.92	968.00	932.00	1029.40	77.75	97.40
9	CompressiveStrength	2.33	82.6	35.82	34.44	23.71	46.14	16.71	22.43
8	Days	1.00	365.0	45.66	28.00	7.00	56.00	63.17	49.00
7	FineAggregate	594.00	992.6	773.58	779.50	730.95	824.00	80.18	93.05
3	FlyAsh	0.00	200.1	54.19	0.00	0.00	118.30	64.00	118.30
5	Superplasticizer	0.00	32.2	6.20	6.40	0.00	10.20	5.97	10.20
4	Water	121.80	247.0	181.57	185.00	164.90	192.00	21.35	27.10

It could be interesting considering days not only in terms of single numbers of days, but also in terms of value ranges since it is probably more relevant the difference between the concrete of 0-to-5 days and 10-to-15 days, rather than the difference between a concrete made 3 or 5 days ago. Specifically, it seems that a meaningful set of ranges is the following:

- 1 day: to assess the initial behavior of the concrete.
- 7 days: to get a preliminary indication of its strength.
- 28 days: this is the standard for measuring the final compressive strength of concrete. Many concrete mixes reach about 70-80% of their total strength by this period.
- Beyond 28 days (e.g., 90 days, 180 days, or even 1 year): this is considered the long-term. During this period, supplementary cementitious materials like blast furnace slag or fly ash can continue reacting (through pozzolanic processes), further improving the compressive strength.

Furthermore, there's a combination of two concret's "ingredients" that is specifically useful: Water/Cement Ratio. This is the most critical parameter for compressive strength. A lower ratio (less water) increases compressive strength because it reduces the internal porosity of the concrete. However, if the ratio is too low, it can lead to compaction difficulties. So, compressive strength is inversely proportional to the w/c ratio. There are empirical formulas (such as Abrams' law) that relate compressive strength to the water/cement ratio. Note that the unique values of Days columns are 14, which is quite a low number for a thousand and more istances. It's quite fair to detect a clear pattern of measurement in terms of days from concrete preparation.

Days	count
1	2
3	134
7	126
14	62
28	425
56	91
90	54
91	22
100	52
120	3
180	26
270	13
360	6
365	14

There are two measurement for both 90 e 91 days: I'll keep them together into the "90 days" value. The 120 days value has only three instances. I assume they can be represented by 100 days value. The same goes for 360 and 365 days. I'll merge the two cases since, especially the first one, has got so little cases (only 6). It could be useful to break down compressive strength into the following catogories:

1. Poor Strength: < 10 MPa: Concrete with compressive strength in this range is considered very low quality and would typically be unsuitable for most structural applications.
2. Fair Strength: 10 MPa - 20 MPa: Concrete in this range is considered low-strength and would be used for non-critical applications, like sidewalks or low-load structures.
3. Good Strength: 20 MPa - 35 MPa: This is a common range for general-purpose concrete used in residential buildings and standard construction projects.
4. Very Good/Structural Strength: 35 MPa - 50 MPa: Concrete in this range is used for more demanding structures such as larger buildings or industrial facilities.
5. High/Excellent Strength: > 50 MPa: High-strength concrete, often used in high-rise buildings, bridges, and infrastructure projects requiring superior durability.

1.4.2. Data featuring

For further steps, I want to create the following variables:

1. dayRange (categorical) that represents the evolution of strenght based on the numbers of day of concrete preparation.

2. waterCementRatio (numerical).
3. StrengthCategory (categorical).

1.5. Data partitioning

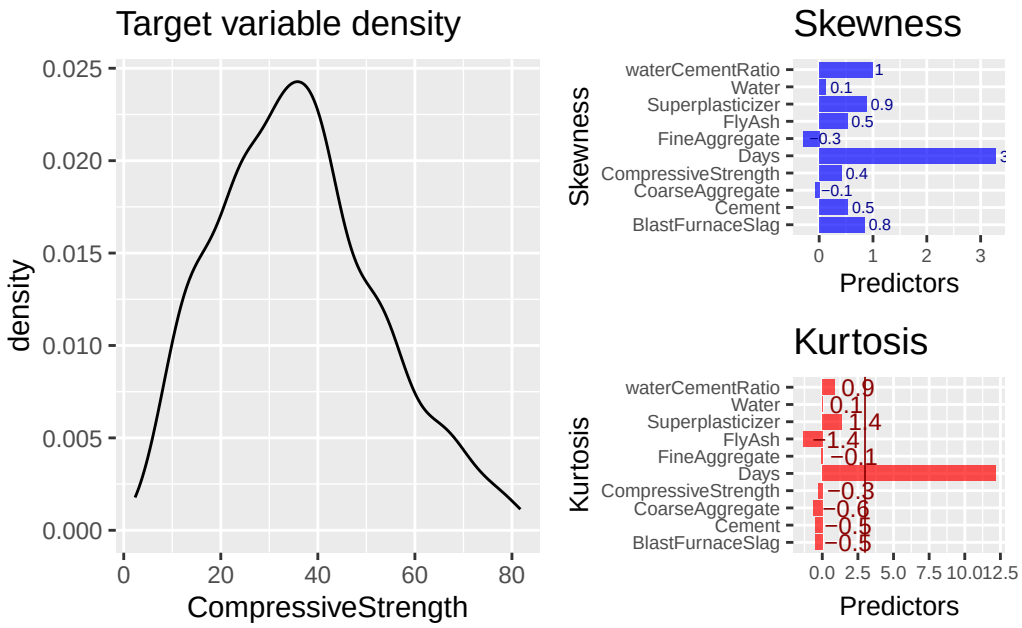
For data partitioning a classical 80/20 proportion between train and test datasets has been choosen to create the training and the test set (DataTrain and DataTest respectively).

2. EXPLORATIVE DATA ANALYSIS

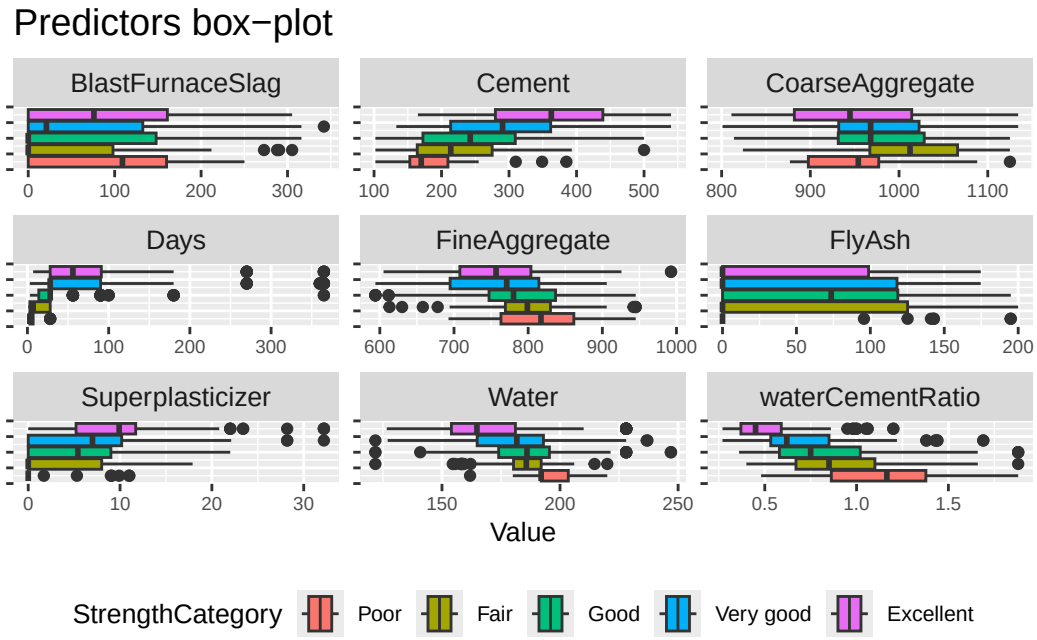
2.1 Feature analysis

2.1.1. Target analysis

First of all, I want to see the distribution of the target values. Again, SmartEDA package provides with avery useful command: ExpNumStat() that returns a set of statistics that, after a little manipulation, can be visualized in a functional way throught the patchwork package.



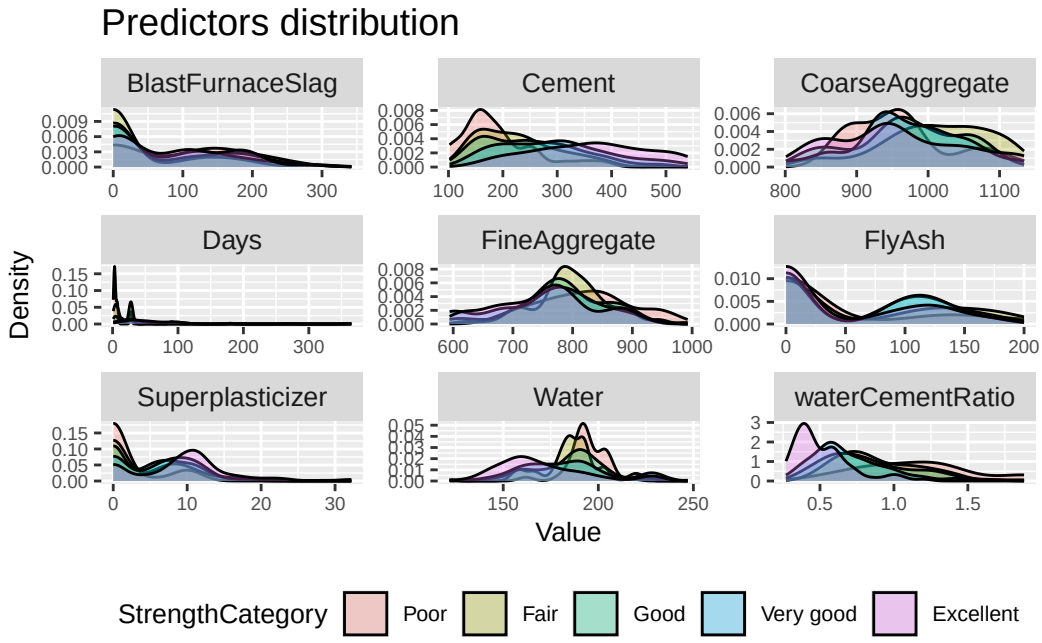
Days feature appears pretty skewed and unbalanced.



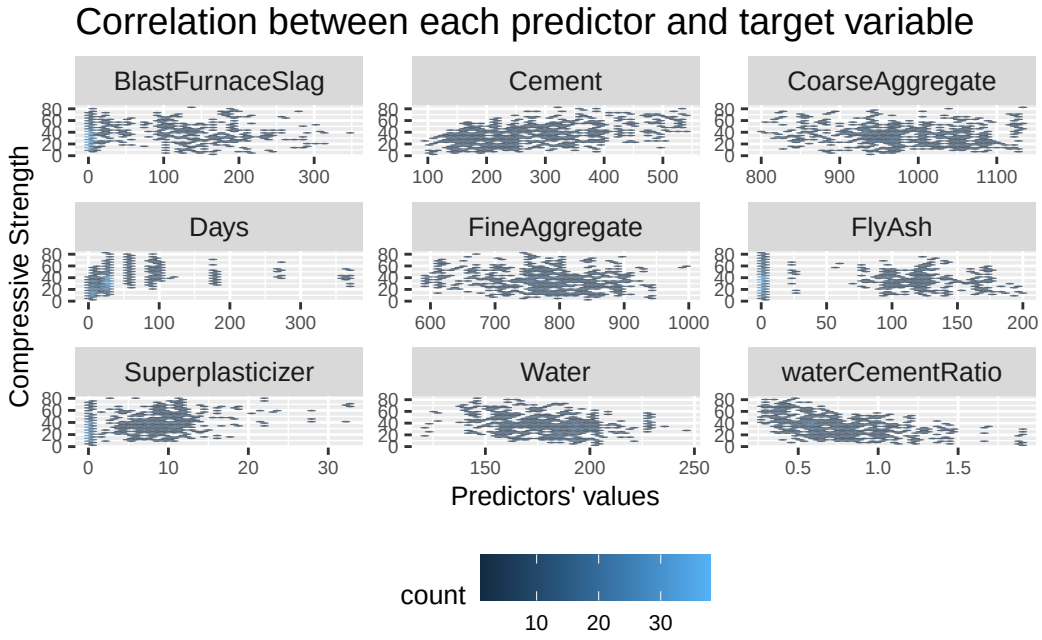
Box-plots show the presence of outliers, but the dataset content backs the idea that each dataset's instance is referred to a different building purpose for which that specific concrete composition has been defined: this should advice not to handle any "extreme" value.

2.1.2. Univariate predictors analysis

In order to have a bird's eye view of the different behaviour of how target variable "replies" to each single predictor, I prefer a wrapped collection of each feature's plot rather than a sequence of single plots to improve overall readability and ease comparisons. To do that, I will reframe with pivot_longer() command the DataTrain tibble bringing the 20 feature into one (longer) column that is going to include every single variable. I'll aggregate predictors according to target value, but I won't use the continuous output "CompressiveStength" (too many values would lead to too many plots compromising readability), but the new-created categorical "strenghtCategory".



Is there a kind of visible relationship between each predictor and the target? In order to try to make plots more readable, geom_hex() has been used instead of geom_point().

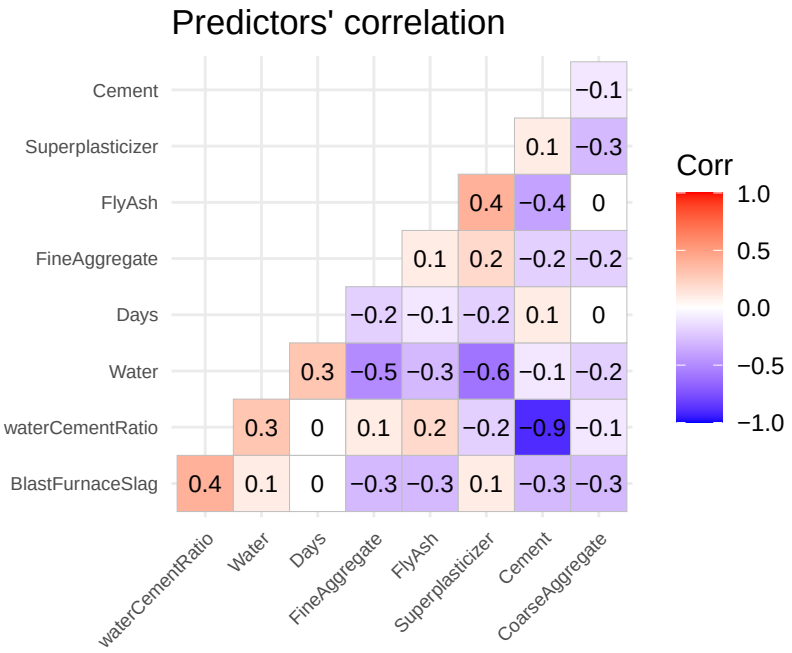


At a first glance I can actually observe that water, cement and (consequentially) water/cement

ratio show a somehow clearly visibile linear relation with target values, while other point-plots show undistinctive clouds.

2.1.3. Multivariate predictors analysis

Finally, are predictors correlated with each other? Are all of them really useful during the next ML models' training phase? Is it possibile/necessary to select some feature rather than using all of them, even if the number of feature is, all in all, quite limited? Visualizing the existing correlations is the next step.



There's only a robustly high value in correlation (waterCementRatio vs Cement). Quite suriprisingly the same high correlation is not evident between this ratio and it other component(water).

2.2. Conclusions

1. Insight about skewness:

- high skewness for one predictor (Days) (Fisher's Gamma Index > 3),
- five predictors (waterCementRation, Superplaticize, Blast, Cement, FlyAsh) with a moderate skewness (values between 0.5 and 1),
- the remaining three features show a low-level of skewness.

A transformation to handle this shape distribution is to be evaluated. Specifically, it will be interesting to try a selective transformation only on extremely skewed variables.

2. Insight about kurtosis:

- Days feature is strongly leptokurtic (around 12),
- all remaining features have a kurtosis index below 2 (from 1.4 to -1.4) which testify a significant platykurtic distribution.

3. Outliers do exist, but it can be assumed that cases with very high or low values are explainable by the final scope for which that concrete blend has been calculated.

4. Only two predictors are correlated between themselves. Cement and (the new created) waterCementRatio.

3. TRAINING AND TESTING ML MODELS WITH WORKFLOW_SETS PACKAGE

Some features have a non-normal distribution visible via skewness and/or kurtosis. In order to get a performing ML model, an effective pre-processing path is to be selected. Specifically, I'm interested in checking if a general predictors' transformation works better or worst than a targeted feature transformation. During preprocessing phase, it'll be convenient to handle:

- the skewness of some or all predictors,
- the kurtosis of some or all predictors,
- the skewness and the kurtosis of target variable.

Above mentioned skewness and kurtosis is supposed to be reduced with a power transformation (Yeo-Johnson) that is applied to either all predictors, or a group of them; in both cases, target variable will be either transformed or kept with its original scale.

To keep the process simple, I'll pick only a couple of models that both seem appropriate to this case and are supposed to be sensitive to any of the above mentioned "issues" referred to dataset features:

- XGBoost (sensible to outliers),
- linear regression (sensible to skewness).

Regarding to metric, the need for transforming variables (possibly including output as well) involves to use only metrics not expressed in the target variable's scale in order to keep results comparable. To this purpose, R-squared seems to fit the bill. It expresses the proportion of the variance in the response variable returned by a model that can be explained by the predictor variables. Its values ranges from 0 (worst predictive model) to 1 (best predictive model) despite the scale of features used.

3.1. Training

3.1.1. Setting cross validation and metrics selection

As far as cross validation is concerned, the dataset small dimension implies a limited number of folders (5 in this case).

3.1.2. Preprocessing recipes

1. The first recipe is as basic as possible. It only removes two categorical variable I'm not interested in using that I've created for explorative puroposes. Here we basically have no preprocessing.

2. The second recipe comes in two version (applied both to all features and to a group of them). Specifically, as far as skewness is concerned, I want to transform: Days, waterCementRation, Superplasticizer, Blast, Cement and FlyAsh. Regarding to kurtosis issue, I'd only need to work on Days beacuse all other predictors have a platykurtic shape which indicates lighter tails and a flatter peak (thus fewer outliers). Of course, the step to remove dayRange and StrengthCategory is confirmed.

3. A third group of recipe takes into account the transformation of the target variable along with all (or some of) preditors in order to see if models' predictive capability improves.

3.1.3. Model Specifications

There are several of ML models out there. The aim of this project is not to test each of them and find the best, but evaluating how feature transformation impact on the overall performance. To keep the whole process simple, a couple of engines may be enough:

1. linear regression,

2. XgBoost.

3.1.4. Workflow Sets

Now, I use the powerful workflow_set{} package to combine every recipe with every model, obtaining the following ten combinations:

1. No preprocessing + Linear regression.
2. No preprocessing + XgBoost.
3. Yeo-Johnson transformation applied to all predictors + Linear regression.
4. Yeo-Johnson transformation applied to all predictors + XgBoost.
5. Yeo-Johnson transformation applied to a group of (chosen) predictors + Linear regression.
6. Yeo-Johnson transformation applied to a group of (chosen) predictors + XgBoost.
7. Yeo-Johnson transformation applied to all predictors and target feature + Linear regression.
8. Yeo-Johnson transformation applied to all predictors and target feature + XgBoost.
9. Yeo-Johnson transformation applied to a group of (chosen) predictors and target feature + Linear regression.
10. Yeo-Johnson transformation applied to a group of (chosen) predictors and target feature + XgBoost.

3.1.5. Tuning and selection best performing workflow and hyperparameters

To choose the best performing workflow, it's necessary to extract the results (expressed in terms of rsq) for every workflow that has been trained.

wflow_id	.metric	CvAvgScore	Rank
YjAny_target_Xgb	rsq	0.9175763	1
YjAll_target_Xgb	rsq	0.9175689	2
YjAny_Xgb	rsq	0.9130881	3
YjAll_Xgb	rsq	0.9130526	4
NoPrep_Xgb	rsq	0.9127228	5
YjAny_target_LR	rsq	0.8149525	6
YjAll_target_LR	rsq	0.8146873	7
YjAny_LR	rsq	0.8003771	8
YjAll_LR	rsq	0.7998732	9
NoPrep_LR	rsq	0.6136640	10

According to rsq metric we can observe that XgBoost brings definitely better performance than linear regression regardless of preprocessing steps. Specifically, XGBoost model's perfomances are pretty steady and show a very limited range: from 0.9127 to 0.9175, while linear regression rsq varies from 0.6136 to 0.8149. This confirms that XGBoost is less sensitive to dataset issues like outliers, skewness etc and thus to preprocessing activities. On the other hand, skewness and/or kurtosis have stonger impact on linear regression model and this requires a significant handling before running models. Nevertheless, even after such a preprocessing session, the "plain" XGBoost model still performs better than the most intensively preprocessed workflow with LR model.

Let's pick the best hyperparameters that have been used during model training (5 combinations have been set in the workflow mapping).

trees	tree_depth	.config
1801	2	Preprocessor1_Model1

3.2. Testing

3.2.1. Performance on the test set

At this stage, the best workflow with the optimal hyperparameters have to be used on the test set (through last_fit() command) to understand which performace they lead to.

.metric	.estimator	.estimate	.config
rsq	standard	0.9600353	Preprocessor1_Model1

An rsq equal to 0.96 states that the chosen model is able to expalin the 96% of variance of the test dataset. Quite suriprisingly, this result outperform the rsq of training phase where 0.91 rsq was reached.

As previuosly said, the model selection has been based on rsq due to the tranformations that in some cases have been applied to the target variable.

It may be interesting measuring other metrics (notably RMSE and MAE) on the test dataset transformed according to the preprocess recipe that has been choosen (Yeo-Jonhson applied to both some predictors and target feature) so to have a more complete view of the model's prections ability.

To to that, we need to prep the recipe and apply to test dataset and then extract RMSE and MAE which we are interested to investigate.

.metric	.estimator	.estimate	.config
rmse	standard	0.799695	Preprocessor1_Model1

.metric	.estimator	.estimate	.config
mae	standard	0.5255302	Preprocessor1_Model1

To evaluate this metrics, it is necessary to know the content of the test dataset preprocessed with the Yeo-Johnson transformation applied to a group of predictors (Days, waterCementRatio, Superplasticizer, BlastFurnaceSlag, Cement and FlyAsh) and target.

After, transforming test set, the values of the target value (CompressiveStrength) changes significantly.

Test_dataset	Target_variable_mean
Original	35.80
Preprocessed	10.31

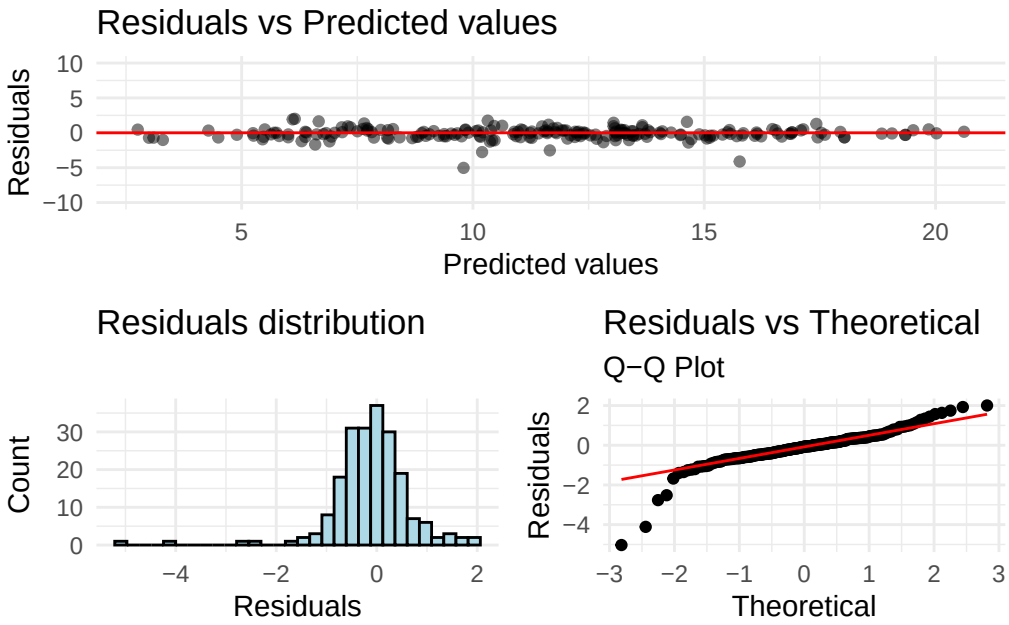
The rsq (0.96) explain the 96% of variance. It's a solid point of start. Since this is even better than the training rsq level (0.91), it's useful to chech for other metrics (RSME and MAE) which are calculated on the test set only after preprocessing it with the same recipe applied to the training set.

To sum up:

1. The mean of the target variable in the preprocessed test set is 10.31.
2. With a 0.96 of rsq the model seems to be able to well explain the variability of the target variable.
3. The RMSE equal to 0.79 espresses a value less than 10% of the target avarage (7.66%).
4. A Mean Absolute Error (MAE) lower than the RMSE highlights that errors are, in avarage, pretty limited. RMSE bigger than MAE points out there are some big errors that affect more significantly the first metric rather than the latter. The RMSE/MAE ratio is about 1.52 and suggest that error distribution is non perfectly uniform but still pretty regular. A level of this ratio below 2 testifies that outliers are not "having major impact"ruling" the model. The difference between RMSE and MAE may be caused by some errors bigger than others but it doesn't seem such a compromising issue. An analysis of residual seems appropriate anyway.

All in all, it seems that the model generalizes well from the training data to the test data, avoiding both overfitting and underfitting.

3.2.2. Residuals analysis



Based on the residuals analysis, it is possible to point out the following insights:

1. Residuals vs Predicted Values plot checks for patterns in residuals to assess whether the regression assumptions (such as linearity and homoscedasticity) hold. The residuals appear to be scattered randomly around the red horizontal line at zero. This indicates that there is no obvious systematic pattern, suggesting that the model captures the underlying structure in the data well. If the residuals formed a clear pattern (e.g., a curve or funnel shape), it would indicate issues like non-linearity or heteroscedasticity.

2. Residuals Distribution histogram assesses the normality of residuals. The residuals show a roughly symmetric and unimodal distribution centered around zero, which is close to the normal distribution assumption. However, there may be slight deviations from normality (e.g., some outliers or slight skew), but overall, the distribution looks reasonable for most regression tasks.

3. Q-Q Plot compares the quantiles of the residuals to a theoretical normal distribution. The points mostly align with the red diagonal line, which indicates that the residuals approximately follow a normal distribution. Some deviations are visible at the tails (extreme values), suggesting potential outliers or heavy tails in the residual distribution.

The model seems to perform well: residuals are randomly distributed (no pattern), suggesting linearity and independence and appear approximately normally distributed, with slight deviations at the tails.

These plots suggest that the model is appropriate for the data, though further investigation into potential outliers or extreme values in the residuals might be warranted if they influence model performance.

CONCLUSIONS

This project has intended to explore, in the context of a regression problem, the impact of different preprocessing recipe on different models' performances and to use, when possibile, tools for optimize the visualization of plots and table so to make the final document more readable.

1. About the impact of dataset preprocessing on models' performance

As a matter of fact, XgBoost has performed much better than linear regression. Specifically, some insight has been detected:

1. XgBoost model has not benefited from major gains by adopting preprocessing steps, while linear regression models' performances have changed significantly when different proprocessing recipes have been introduced.

2. Target variable transformation has caused slight but visible better scores (third decimal digit).

3. Transforming all predictors rather than only a subset has not resulted in noticeable differences in the rsq scores.

4. Rsq has been used to evaluate models' performance since this score is expressed in a standard e neutral scale. Once the model has been selected, other metrics (RMSE and MAE) have been used along with rsq to obtain more pieces of information and, thus, a more complete review of results.

5. Combining scores and residuals analysis seems to be a good practice to better read model's performance.

2. About the effectiveness of the packages used throughout this work

1. workflow_set confirms itself as a very useful package to mix recipes and models and to select the best combination.
2. EDA may gain some benefits from the more efficient plot exposure offerered by patchwork package. The opportunity to order in one or more rows a group of plots allows to create a sequence that help creating an effective storytelling.