Introduction to Machine Learning project: $\mathbb{X}$ food popularity

Alessio Gaia, Carpenè Sara, Fantuzzi Giulio, Valentinis Alessio[*]

[*] problem statement, solution design, solution development, data gathering, writing

Course of AY 2023-2024 - Data Science and Artificial Intelligence

1 Problem statement

This project aims to design and develop a Machine Learning system to predict *how popular* a tweet about food will be. Let $X = \{x \mid x \text{ is a tweet about food}\}$ be the set of tweets of interest. Since $x \in X$ cannot be processed directly by the machine, we first need to apply a pre-processing function $f_{\text{pre-proc}} : X \to X'$ into a machine-interpretable space X' (see section 2.2). Then, we seek to learn a model $m \in M$ from an $f'_{\text{learn}} : P^*(X' \times Y) \to M$ and use it into a function $f'_{\text{predict}} : X' \times M \to Y$ to predict the output variable y for an input tweet.

Constructing such an f'_{learn} would be too complex for humans, due to the intricate nature of the problem and the significant human resources required to handle a multitude of observations, each with lots of details. Moreover, the complexity of the solution dictates that the model $m \in M$ will not be straightforward, so relying on computational methods is the most practical approach for the prediction phase. A Machine Learning solution is hence well-suited for the task at hand. More precisely, we will opt for a regression approach and, considering the nature of both X and Y, supervised learning techniques emerge as the most fitting choice.

2 Data

2.1 Web scraping

As no dataset was provided, we performed our analysis on data gathered using the Python library `ntscraper` [1]. This library bypasses the limit to 1500 tweet downloads per month imposed by $\mathbb{X}$'s official API, which could make our dataset insufficiently large (small size) and not representative (poor coverage). Thanks to this library, we successfully collected a total of 4075 English-language tweets containing either the word or the hashtag *'food'*.

2.2 Pre-processing

While the input variable is inherently digital, it is not directly processable by machines. Therefore, a pre-processing phase involving feature engineering is a key step in the design of our ML system. The library `ntscraper` used for data retrieval facilitates the process, as it not only captures the text of the tweets, but also extracts features including likes, comments and retweets count, quotes, retweet status, external links, pictures, videos and gifs. The library provides also functionalities to get user details, such as profile image, bio, website and total of tweet, media, followers and following count. In a subsequent data preparation phase, a vast majority of the variables was converted into a more suitable format (e.g., from *string* to *bool*), and some custom functions has been implemented to get additional information from texts, such as the number of hashtags and emojis in each tweet. Finally, after some basic manipulations to texts (conversion to lowercase, removal of punctuation, stemming, removal of stop words), we performed a *tf-idf* vectorization in order to penalize frequently occurring words while highlighting less common and more informative ones, contributing to the model's focus on meaningful features.

Regarding the response variable, we opted for a slightly different version of the *engagement rate* [4, 2], a metric that measures the overall interactions received by a tweet, divided by the total number of user's followers. Formally:

$$y^{(i)} = \ln\left(\frac{x^{(i)}_{\text{likes}} + x^{(i)}_{\text{retweets}} + x^{(i)}_{\text{comments}} + 1}{x^{(i)}_{\text{followers}} + 1}\right) \tag{1}$$

The log transformation was applied to mitigate the skewness of the engagement, while 1 was added both to numerator and denominator as a way to extend its applicability to any tweet (avoiding also the risk to obtain infinite values).

Note: once computed the engagement rate, the features involved in (1) were dropped from the dataset, in order to avoid the estimation of any trivial model.

3 Assessment and performance indexes

To assess the degree to which our solutions solved the regression problem, we focused both on their *effectiveness* and *efficiency*. Effectiveness was measured by evaluating the *RMSE* (the lower, the better), while efficiency was measured in terms of CPU time (in seconds), during both learning and prediction phase. To reach a more general and robust result, the entire assessment procedure was based on a 10 fold cross-validation.

4 Proposed solution

We implemented different supervised learning techniques, namely: Regression Tree, Random Forest, Support Vector Regressor (SVR) and kNN. To assess their performances, we referred to the Dummy Regressor as a comparison baseline.

4.1 Hyperparameter tuning

For each learning technique, we conducted a hyperparameter tuning phase through grid search, based on RMSE and performed within a 10-fold CV.

Regression Tree

The optimal value for n_{min} was found to be $n_{\text{min}} = 100$ after tuning across the grid $\{1, 10, 15, 25, 30, 50, 100, 150, 200, 500\}$.

Support Vector Regressor (SVR)

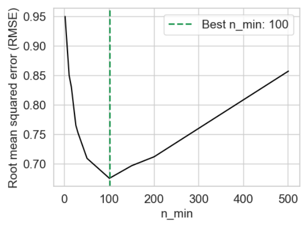
Linear: the optimal value for the regularization parameter c was found to be $c = 0.005$ after tuning across the grid $\{0.001, 0.003, 0.005, 0.01, 0.03, 0.05\}$;

Poly: c was tuned across the grid $\{0.001, 0.005, 0.03, 0.05, 0.1, 0.5, 1\}$, while the degree of the polynomial was tuned across the grid $\{2, 3, 4, 5, 6, 7\}$.
The optimal combination resulted $(c = 0.1, d = 2)$;

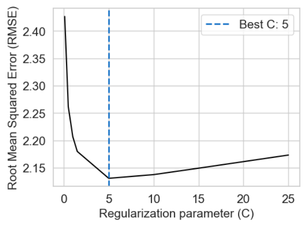
Rbf: c was tuned across the grid $\{0.1, 0.5, 1, 1.5, 5, 10, 15, 25\}$, while the parameter γ was tuned just across the default values provided by *scikit-learn* [6].
The optimal combination resulted $(c = 5, \gamma = \text{'scale'})$.

k-Nearest Neighbor (kNN)

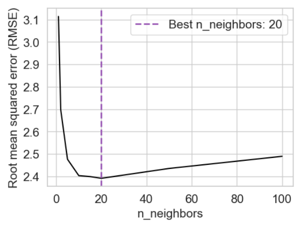
The optimal value for the parameter $n_{\text{neighbors}}$ was found to be $n_{\text{neighbors}} = 20$ after tuning across the grid $\{1, 2, 5, 10, 15, 20, 50, 100\}$.



(a) *Regression Tree*

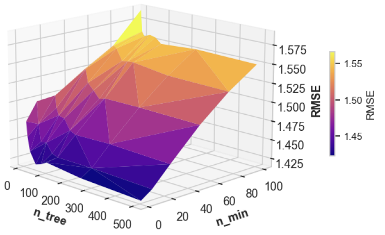


(b) *SVR (rbf)*



(c) *kNN*

Random Forest (RF)
Despite RF having good default values ($n_{\text{tree}} = 500$, $n_{\text{min}} = 1$ and $n_{\text{vars}} = \lceil \frac{p}{3} \rceil$), we still tuned n_{tree} and n_{min} across the grids $\{10, 15, 25, 50, 75, 100, 250, 500\}$ and $\{1, 10, 25, 50, 75, 100\}$, respectively. The optimal pair resulted $(n_{\text{tree}} = 500, n_{\text{min}} = 1)$.



5 Results and discussion

Results of our computational experiments are illustrated in table 1 and fig. 2. The Dummy regressor, though highly efficient, demonstrates limited predictive capability and, as expected, it resulted in the worst learning technique (highest

Table 1: *Performance metrics for the analyzed learning techniques*

	Effectiveness (RMSE)		**Efficiency (Time in seconds)**			
Models	μ	σ	$learn_\mu$	$learn_\sigma$	$pred_\mu$	$pred_\sigma$
Dummy Regressor	2.549105	0.089576	0.001508	0.001075	0.000287	0.000083
SVR (kernel=*linear*)	2.226802	0.081544	1.528836	0.007922	0.156785	0.000370
SVR (kernel=*poly*)	2.414326	0.077510	1.497258	0.003428	0.159443	0.000456
SVR (kernel=*rbf*)	2.092663	0.076907	1.650402	0.010091	0.275482	0.002714
kNN ($n_{\text{neighbors}} = 20$)	2.360611	0.067055	0.005344	0.001519	0.012011	0.002559
Regression Tree ($n_{\text{min}} = 100$)	1.631314	0.058482	0.026548	0.005597	0.001133	0.000201
RF ($n_{\text{tree}} = 500; n_{\text{min}} = 1$)	1.378885	0.054963	1.052611	0.012792	0.029593	0.004383
RF ($n_{\text{tree}} = 100; n_{\text{min}} = 1$)	1.381328	0.057168	0.225316	0.008992	0.007599	0.000237
RF ($n_{\text{tree}} = 100; n_{\text{min}} = 25$)	1.408994	0.051836	0.159062	0.005354	0.007049	0.000198

RMSE). Surprisingly, despite its greater inner complexity, SVR ranks among the least effective and efficient techniques, requiring the longest time for both learning and prediction. In contrast, Regression Tree shows improved effectiveness, with an RMSE approximately 36% better than the Dummy regressor and 22% better than SVR with a Gaussian kernel (the most effective SVR configuration). While not as efficient as the Dummy regressor, it balances predictive capability with computational efficiency. As for kNN, it falls between Regression Tree and SVR in terms of effectiveness, and is efficient in learning but slower in prediction (approximately 26 times slower than Tree).

Random Forest with $n_{\text{tree}} = 500$ and $n_{\text{min}} = 1$ outperforms Regression Trees and boasts the lowest *RMSE*. However, its moderately efficiency comes at the cost of relatively slower predictions. Exploring relaxed hyperparameter values reveals a subtle interplay between predictive accuracy and computational efficiency. For instance, with and $n_{\text{tree}} = 100$ and $n_{\text{min}} = 25$ the *RMSE* increases by only 2.14%, while the gain in terms of time performances is remarkable: learning and prediction are as much as 84.89% and 76.18% faster, respectively.

In conclusion, to showcase the *'Wisdom of the trees'* principle, Random Forest emerges as the most effective technique analyzed according to the assessment guidelines and performance indexes proposed in section 3.

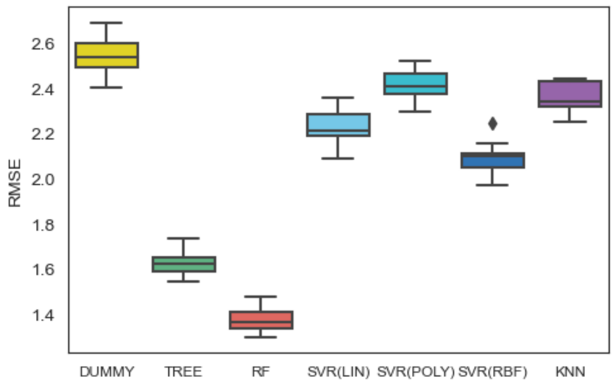


Figure 2: *Boxplot of learning techniques effectiveness (with optimal hyperparameters)*

References

[1] Lorenzo Bocchi. `ntscraper`: Scrape from Twitter using Nitter instances. https://github.com/bocchilorenzo/ntscraper.

[2] Werner Geyser. Twitter money calculator – how much are your tweets worth? https://influencermarketinghub.com/twitter-money-calculator/.

[3] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 6. Springer, 2013.

[4] Sam Lauron. What is a good engagement rate on twitter? https://www.rivaliq.com/blog/good-engagement-rate-twitter/.

[5] E Medvet. *Introduction to Machine Learning*. 2023.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.