Wine vintage quality prediction

Deep Learning for perceptron

University of Tokyo

Date: 29/05/2025

Author: BOBANT Jean-Baptiste / PONCET Lucas / WALLIER Malek

Contents

Wine vintage quality prediction

Deep Learning for perceptron, University of Tokyo

Date: June 2025
Author: BOBANT Jean-Baptiste / PONCET Lucas / WALLIER Malek

Intro

Even for wine enthusiasts, having the ability to precisely determine the particularly good quality of a wine from its meteorological growth conditions and grape variety without waiting for tasting-based recommendations can be very challenging. That is what motivated us to leverage Deep Learning tools to assess this problem. Therefore, our objective is to combine efficiently the grape variety and growth's meteorological conditions of a wine to predict its quality without tasting it using trained Deep Learning models with consumer-based ratings and prior meteorological conditions.

Wine data collection

To build our wine database, we used the **Vivino** consumer-based application data. Moreover, we scrapped their open API in order to retrieve french red wines with vintages from 2010 to 2024. Vivino's API restricts scrapping to 80 pages so we had to roll consecutive scrapings with continuous ranges of price. Finally, we achieved scrapping 10k french red wines from vintages between 2010 and 2024, keeping for each wine :

Wine: wine's name (str)

Winery: wine's winery (str)

Year: wine's vintage from 2010 to 2024 (int)

Price: wine's current price on market (float)

Vintage_rating: wine's customer rating for this vintage between 0 and 5 (float)

Vintage_rating_count: wine's number of customer rating (int)

Region: wine's region (the scale of the region is very sparse, can go from "bordeaux" to "st-emilion-grand-cru") (str)

Cepages: wine's cepages present in its composition ordered by decreasing importance. (str)

Wine data preprocessing

Once our wine dataset created, we needed to figure out a way to label our data according to an outstanding wine quality. This label would be a balance of price and rating that represents the incredible potential of a wine thus labeled 1 and 0 either.
We started by normalizing the wine prices by year so as to avoid inflation effects on prices which would have an important impact on our quality indicator, penalizing older wines. To ensure robust comparison across wines, we compute a Wine Quality Indicator (WQI) combining adjusted rating and normalized price. First, we estimate a Bayesian average rating:

$$WR = \frac{v}{v+m} \cdot R + \frac{m}{v+m} \cdot C$$

where v is the number of ratings, R the average rating, m the median number of ratings, and C the mean rating. This smooths low-sample vintages toward the global average. We then normalize WR by dividing by 5:
$WRN = WR/5$.
Finally, we compute $WQI = WRN \times W_{\text{price}}$, rewarding high-rated, good-value wines relative to their vintage and region.

We then plotted distributions of WQI for combinations of regions and cepages to evaluate a threshold to apply in order to select outstanding quality wines. We decided to choose the **65th percentile as the threshold** for each cepages/region combination so as to decide whether a wine is of outstanding quality or not respectively 1 or 0 label.

Meteorological data collection

We sourced our weather observations from the open-data portal of Météo-France. Our workflow unfolded as follows:

1. **Select the relevant départements.** We first listed every French département that intersects a vineyard area included in this study.

2. **Combine the two historical files.** Météo-France provides daily climatological records in two archives, 1950–2023 and 2024–2025. After downloading both, we concatenated them into a single dataset so that each station's time series is continuous.

3. **Filter out incomplete stations.** Any station whose series contained too many missing values (threshold defined in the preprocessing notebook) was discarded to preserve data quality.

4. **Create one file per calendar year.** Instead of grouping all years together, we rebuilt the dataset so that each output file contains every selected département for one given year. This structure speeds up year-by-year analyses and model training.

5. **Restrict the time span.** Finally, we kept only observations from 2010 onward, as earlier data fall outside the analytical window of the project.

This cleaned and re-structured collection now forms the meteorological baseline for our subsequent modelling steps.

Meteorological data Preprocessing

Because grapevines in temperate Europe develop mainly from **April to September**, we restricted our climatic analysis to this six-month window. This choice is supported by three studies: [**? ? ?**].

Guided by the metrics proposed in these papers, we engineered the following agro-climatic variables:

GDD **Growing Degree Days** accumulated from April to September, a proxy for heat available to the vine.

$\mathbf{TM_{summer}}$ Mean daily **temperature** over the summer core (June–August).

$\mathbf{TX_{summer}}$ Mean daily **maximum temperature** for the same period.

$\mathbf{temp\_amp_{summer}}$ **Diurnal temperature range** (TX – TN) averaged over summer, indicating day–night amplitude.

hot_days Count of days with TX $\geq 35°C$, signalling potential heat stress.

$\mathbf{rainy\_days_{summer}}$ Number of days with precipitation ≥ 1 mm between June and August.

rain_June Total rainfall in June, relevant for flowering and berry set.

rain_SepOct Cumulative rainfall during September–October, affecting harvest conditions and botrytis risk.

$\mathbf{frost\_days_{Apr}}$ Days in April with TN $< 0°C$, capturing late-frost hazards to young shoots.

$\mathbf{avg\_TM_{Apr}}$ Average daily temperature in April, a cue for bud-break timing.

Together, these features constitute the climate signature fed to our learning algorithms, enabling them to relate intra-seasonal weather patterns to subsequent wine-quality outcomes.

Wine–Meteo Integration Pipeline

To combine each wine sample with representative climate data for its vintage, we implemented a spatial–temporal merging workflow:

Appellation Geocoding
Fuzzy-match Vivino AOC strings to reference polygons, reproject from EPSG:2154 to WGS84 and compute each appellation's centroid and deduplicate overlapping AOCs to yield a unique (region → latitude, longitude) lookup for approximately 10 000 wines.

Region–Year Grid Construction
Cross-join the set of centroids with years 2010–2024, producing a complete (region, year) table while preserving all region metadata for subsequent joins.

Nearest-Station Climate Attachment
Load cleaned, per-station annual weather (2010–2024) containing ten agro-climatic metrics. For each year, index station coordinates in a haversine-metric BallTree and query the closest station for every region centroid. Mark any region whose nearest station is > 40 km away as "unmatched", their weather will be NaN.

Final Wine–Weather Join and Curation
We parse and retain only the dominant *cépage* per wine (i.e. the first grape variety); convert vintage to `year`. We can then merge the Vivino table to the enriched (region, year) climate table, and drop rows lacking essential data (cépage, year, or core climate metrics).

Outcomes

Each wine record carries its price, dominant grape variety, appellation embedding, and a ten-dimensional climate signature for the growing season, for a clean merged dataset of $\sim$ 10 000 samples.

Context of the problem

We adopt a **temporal hold-out** design to **test** model robustness under contrasting vintage conditions: vintages 2018 (a "good" year for french wine) and 2021 (a notably "poor" year) compose our test set, while all other vintages serve as training data. This yields approximately **8000** training samples and **2000** test samples.
Each wine instance is encoded by:

- Agro-climatic sequence: 10-dimensional vector of climate variables from Météo-France station records, as specified in item
- Economic indicator: the vintage's price, normalized within each year to correct for inflationary and market dynamics.
- Varietal embedding: the principal cépage, represented as a learned categorical embedding.
- Geolocation embedding: the appellation ("region") and its linked Météo-France station ID, each encoded categorically.

Baseline models

We evaluate four classical baselines—Logistic Regression, Random Forests, XGBoost, and Histogram-based Gradient Boosting (HGB)—under a common preprocessing and optimisation pipeline. All models use a 5-fold cross-validated grid search over carefully selected hyper-parameter spaces, and their test-time performance is summarised below:

Model	**Accuracy**	**F_1-score**
XGBoost	0.7683	0.6510
Logistic Regression	0.7615	**0.7295**
Random Forest	0.7315	0.6895
HGBClassifier	0.7379	0.6903

Despite the more advanced ensembling capabilities of XGBoost and HGB, Logistic Regression achieves the highest F_1-score and ROC-AUC—suggesting that, given appropriate regularisation and class balancing, linear models remain strong contenders for this task. Overall, all models exceed 0.73 in accuracy, confirming the robustness of classical approaches before introducing deep learning architectures.

Multilayer Perceptron (MLP)

Our final tabular network follows the same design described in class, but adapted to our data and evaluation constraints:

- **Embedding layers.** All three categorical columns—`station`, `region`, and `cepages`—are mapped to dense vectors whose dimensionality is $\lfloor\sqrt{|\mathcal{V}_{\text{col}}|}/2\rfloor$ (with an UNK token for unseen values).
- **Dense backbone.** Two hidden blocks of size $128 \to 64$ with ReLU, batch-normalisation and a dropout of 0.1, followed by a linear output layer.
- **Loss re-weighting.** Class weights are inversely proportional to their frequency (pos/neg = 0.35) to curb the bias toward the majority class.
- **Optimisation.** Adam ($\eta = 1\times10^{-4}$) with L_2 regularisation (10^{-5}) and a maximum of 1500 epochs;
- **Threshold tuning.** A validation PR-curve is scanned and the smallest threshold achieving Recall $\geq 75\,\%$ and the greatest Precision is chosen; for the reported run this gives $\tau^\star = 0.63$.

Hyper-parameters.

Hidden layers	[128, 64]
Activation	ReLU
Batch-norm	False
Dropout	0.10
Learning rate	1×10^{-4}
Weight decay	1×10^{-5}
Batch size	512
Max / early stop	1500 / patience 10

Test performance (single best model, $\tau^\star = 0.63$).

Class	Precision	Recall	F_1
0 (negative)	0.92	0.82	0.87
1 (positive)	**0.72**	**0.87**	0.80
Accuracy		0.842	

The model therefore meets the business requirement of "high precision first": although the recall is slightly lower than 90 %, the precision remains above 70 %, resulting in an F_1 close to 0.80 and an overall accuracy of 0.84 on unseen data.

FT Transformer

Our final transformer-based model adapts the architecture described in [**?**] to wine classification, using stacked attention over feature tokens.

- **Embedding layers.** The three categorical variables (`station`, `region`, `cepages`) are mapped to trainable vectors. These are concatenated with learned projections of numeric features to form a token sequence per wine.
- **Feature tokenisation.** Numeric features are z-normalised and individually projected into d-dimensional vectors using a learned affine transform. A learnable CLS token is prepended to every sequence.
- **Encoder stack.** The model includes 6 Transformer encoder blocks, each with multi-head self-attention (8 heads), post-layer normalisation, dropout (0.3), and a two-layer feed-forward network with GELU activation and 6× expansion. Residual connections are applied both after the attention and feed-forward layers.
- **Loss re-weighting.** Class imbalance is addressed using weighted cross-entropy, with weights inversely proportional to class frequency (pos/neg = 0.35).
- **Optimisation.** The model is trained with AdamW ($\eta = 3{\times}10^{-4}$, weight decay $5{\times}10^{-4}$), using early stopping over 300 epochs.

Hyper-parameters.

d_{model}	128
Layers (depth)	6
Attention heads	8
Feed-forward	2-layer MLP, expansion ×6
Activation	GELU
Dropout rate	0.30
Normalisation	LayerNorm (post-residual)
Learning rate	3×10^{-4}
Weight decay	5×10^{-4}
Batch size	512
Max / early stop	300 / patience 10

Test performance (single best model, default threshold).

Class	Precision	Recall	F_1
0 (negative)	0.89	0.80	0.84
1 (positive)	**0.68**	**0.81**	0.74
Accuracy		0.801	

The FT-Transformer achieves robust generalisation, with a 0.80 accuracy and a balanced F_1 score which doesn't outperform the MLP performances even for a more complex architecture.

Conclusion

Deep learning model struggle on tabular data so it is not too surprising to see that MLP is the one that did the better job

Maybe try to implement tabpfn.

All the limits on the wine dataset, attribution of dataset, other website for scraping, seing the evolution of the wines using dated comments and ratings.

From a borader point of view, it would have been very interesting to further develop the study by analyzing the temporal evolution of ratings for a wine in order to being able to predict the quality peak of a wine from its growth wather conditions and grape variety.

Bibliography