ChatGPT

Interpretable Categorical Classification

The data has only categorical features and a binary target (`y=yes/no`). In banking applications, **interpretability** is crucial. Simple "glass-box" models like logistic regression, decision trees, and Naïve Bayes are natural choices [1]. These models can output a probability of "yes" while allowing us to explain each prediction in terms of feature effects. For example, one survey notes that **logistic regression** and related linear models remain dominant in credit scoring *because of their inherent interpretability* [2]. Likewise, decision trees produce human-readable "if-then" rules, and Naïve Bayes clearly shows each feature's contribution to the class probability [1] [3].

Model Options

- **Logistic Regression (Generalized Linear Model):** This model predicts the log-odds of subscription as a linear combination of features, then converts to a probability. It directly provides calibrated probabilities and is widely used in finance [2]. Categorical features are typically one-hot encoded: for a feature with L categories, create $L-1$ binary dummies (one category is baseline) [4]. Each dummy's coefficient β gives the change in log-odds (or odds ratio) of "yes" when that category is present. In plain terms, a positive β means this category *increases* the probability of subscription, and a negative β *decreases* it. For example, if `job=management` has a high positive coefficient, we can say "management" strongly raises the chance of "yes," compared to the baseline job. Because logistic regression is linear in log-odds, we can **quote coefficients** or odds-ratios to justify any individual prediction. (Regularization such as L1 can be added to keep the model sparse if many dummies are used.)

- **Decision Tree:** A tree splits data by feature conditions (like `job=technician` or `education=tertiary`) and routes each client down a path to a leaf node with a predicted probability. These splits form explicit "if-then" rules that are very easy to read: e.g. *"If job=technician AND month=may AND no previous loan, then p(yes)=X"*. As Molnar explains, the tree structure **invites contrastive reasoning** – one can see "if this feature were different, the prediction would change" [5]. We can visualize or print the tree to show the exact split conditions. To keep the model interpretable, we would prune or limit depth so that each rule involves only a few features. Short trees (e.g. depth 3 or less) give very concise explanations: each prediction depends on at most 3 conditions. We can also compute feature importances (based on Gini or information gain) to rank which features drive the model. Overall, a decision tree's logic can be **traced** end-to-end, and each decision point can be cited when explaining an outcome [6] [5].

- **Naïve Bayes (Categorical):** Naïve Bayes computes $P(y=\text{yes}|\text{features}) \propto P(y=\text{yes}) \prod_i P(\text{feature}_i|\text{yes})$. We estimate each conditional probability $P(x_i = t \mid y)$ from the data (with smoothing). In practice, one can use Scikit-learn's `CategoricalNB` or compute frequencies manually. The independence assumption means each feature is treated separately, making the model **modular and transparent** [3]. For instance, Naïve Bayes might learn that P(job=`retired` | yes) is high while P(job=`retired` | no) is low; this directly tells us how being retired affects the

"yes" probability. To explain a prediction, we can show each factor's contribution: e.g., "Married clients multiply the baseline odds by *X*, tertiary education multiplies by *Y*, and together the chance of subscription is *Z*." Because each feature's likelihood is explicit, one can literally point to the learned tables of conditional probabilities. Naïve Bayes is very simple, and as Molnar notes, it is "very clear for each feature how much it contributes towards a certain class prediction" [3]. (Be sure to apply Laplace smoothing so unseen category/class combinations don't zero-out the probability.)

Feature Encoding & Preprocessing

- **Encoding Categoricals:** For logistic regression and naive Bayes, use one-hot (dummy) encoding or Scikit-learn's `CategoricalNB` (which requires ordinal encoding) [4] [7]. One-hot avoids implying any numeric order. Decision trees in many implementations (e.g. scikit-learn) can work with label-encoded categories, but one-hot is also fine; just interpret a split like "job_blue-collar = 1".
- **Handling Rare/Missing Values:** If categories like "Not Specified" are semantically missing, decide whether to treat them as a separate category or impute. Rare categories can be grouped (e.g. combine very infrequent jobs into "other") to stabilize probability estimates in Naïve Bayes.
- **Class Imbalance:** The target is skewed (many more "no" than "yes"). We can address this in model training by using class weights or by calibrating the decision threshold, but this doesn't reduce interpretability. In any case, we should evaluate metrics (AUC, recall on "yes" etc.) appropriate for an imbalanced binary task.

Explaining Predictions

All these models allow **clear explanations** for individual cases. In logistic regression and Naïve Bayes, we can decompose the prediction into per-feature contributions (e.g. feature weights or likelihood ratios) and say "because this person has X, Y, Z, their chance is low/high." In a decision tree, we explain by tracing the path of splits: each branch condition (e.g. `education=primary`) acts as a reason. For example, "This client's low subscription probability is because they have a low-education level and there was no previous successful campaign outcome, which are conditions that lead to mostly "no" predictions in the tree."

In summary, using one of these **interpretable classifiers** ensures that we can provide honest, feature-level reasons for any prediction [1] [5]. Logistic regression coefficients or Naïve Bayes likelihoods give us direct insight into how each category pushes the probability up or down, and decision-tree rules offer a straightforward "if-then" justification. This transparency meets the banking requirement that we **explain why a client has a low chance of subscription** in terms of their attributes.

Sources: Interpretable ML literature notes that *"linear models, decision trees and naive Bayes"* are inherently transparent choices [1]. The financial industry likewise emphasizes logistic regression for its clear coefficients [2]. Decision-tree rules and Naïve Bayes feature-wise probabilities are similarly easy to communicate [6] [3].

---

1 3 Interpretable Machine Learning
https://originalstatic.aminer.cn/misc/pdf/Molnar-interpretable-machine-learning_compressed.pdf

2 Enhancing ML Interpretability for Credit Scoring
https://arxiv.org/html/2509.11389v1

4 6 Linear Regression – Interpretable Machine Learning
https://christophm.github.io/interpretable-ml-book/limo.html

5 6 9 Decision Tree – Interpretable Machine Learning
https://christophm.github.io/interpretable-ml-book/tree.html

7 1.9. Naive Bayes — scikit-learn 1.7.2 documentation
https://scikit-learn.org/stable/modules/naive_bayes.html