Theory and Reflection

DTS304TC Coursework 1 Student ID: 1234560

---

1. Bagging vs Boosting

Bagging (Bootstrap Aggregating) trains B independent decision-tree base learners on bootstrapped samples drawn uniformly from the original dataset, then aggregates their predictions by majority vote for classification or averaging for regression. By making each tree learn on a different random subset of the data, bagging reduces variance through decorrelation: even if individual trees overfit, their errors partially cancel out when combined. Boosting, by contrast, trains base learners sequentially: each new learner is fitted to the residuals or misclassified instances from the current ensemble, with the effect that the ensemble reduces bias more aggressively. The key conceptual difference is that bagging treats all base learners as equally informative, while boosting adaptively reweights observations based on past mistakes.

My notebook implemented a controlled comparison of Random Forest (representing bagging) and XGBoost (representing boosting) under identical preprocessing and identical train/validation split. This design is crucial: any difference in results must then reflect the learning algorithm itself, not differences in data preparation or evaluation.

Table 1 summarises the key results drawn from `outputs/tables/personalised_improvement_summary.csv`. It shows model name, validation macro-F1, validation accuracy, the generalisation gap (train F1 minus val F1), per-class F1 scores, and training time. Four rows are shown: Baseline LR, Random Forest, untuned XGBoost, and Optuna-tuned XGBoost.

Table 1: Controlled supervised model comparison (identical pipeline and split).

Model	Val F1	Val Acc	Gap	High F1	Low F1	Std F1	Time(s)
Baseline LR	0.7238	0.7342	0.0146	0.7665	0.6490	0.7558	–
Random Forest	0.7708	0.7877	**0.2292**	0.7875	0.7095	0.8154	57.91
XGBoost	0.8144	0.8371	0.0155	0.8905	0.6944	0.8583	67.64
Tuned XGBoost	0.8520	0.8700	0.1219	0.9084	0.7620	0.8854	142.65

Gap = train_F1 − val_F1.

The results provide strong evidence for the theoretical predictions. Random Forest achieved a training macro-F1 of 1.0000 (perfect fit on the training set) but a validation macro-F1 of only 0.7708, yielding a generalisation gap of 0.2292. This extreme overfitting is also confirmed visually in the Random Forest confusion matrix produced in the notebook. XGBoost, by contrast, had a training macro-F1 of 0.8297 and a validation macro-F1 of 0.8144, giving a gap of only 0.0155. The difference in gaps is striking: RF's overfitting is roughly 15 times larger than XGBoost's.

The per-class F1 column in Table 1 reveals further structure. Before tuning, RF achieved a Low-class F1 of 0.7095, outperforming untuned XGBoost (0.6944) on the minority class—but this advantage disappears once XGBoost is tuned. After Optuna tuning, XGBoost's Low-class F1 rises to 0.7620, a gain of $+0.0676$ over its untuned state and substantially higher than RF's 0.7095. This demonstrates that boosting's sequential residual correction is better suited to learning the non-linear decision boundary between risk classes on this dataset. Bagging's variance-reduction mechanism cannot compensate for the bias that fully-grown trees impose on a mixed numerical and categorical feature space, which is why Random Forest underperforms here.

2. Hyperparameter Optimisation

I used Optuna with the TPE (Tree-structured Parzen Estimator) sampler for 30 trials, targeting maximisation of validation macro-F1. The search space covered nine XGBoost hyperparameters: n_estimators (100–500), max_depth (3–10), learning_rate (0.01–0.3, log-scale), min_child_weight (1–10), subsample (0.5–1.0), colsample_bytree (0.5–1.0), gamma (0–5), reg_alpha (10^{-4}–10, log-scale), and reg_lambda (10^{-4}–10, log-scale). The mixture of discrete and continuous parameters with multiple interactions makes a full grid search computationally prohibitive; TPE avoids exhaustive enumeration by modelling the density of good and bad trial configurations and directing subsequent searches toward promising regions of the parameter space.

Trial 22 produced the best validation macro-F1 of 0.8520, a gain of $+0.0376$ over the untuned XGBoost baseline

of 0.8144. The optimal configuration was: n_estimators= 276, max_depth= 9, learning_rate≈ 0.192, subsample≈ 0.707, colsample_bytree≈ 0.799, reg_lambda≈ 5.0, and gamma≈ 2.5. These values align with expectations: a moderate learning rate combined with large tree depth and many estimators allows the model to fit complex interactions, while subsample and colsample ratios around 0.7–0.8 provide regularisation. Figure 1 shows the Optuna parameter-importance plot, confirming that structural parameters and the learning rate dominated the optimisation.

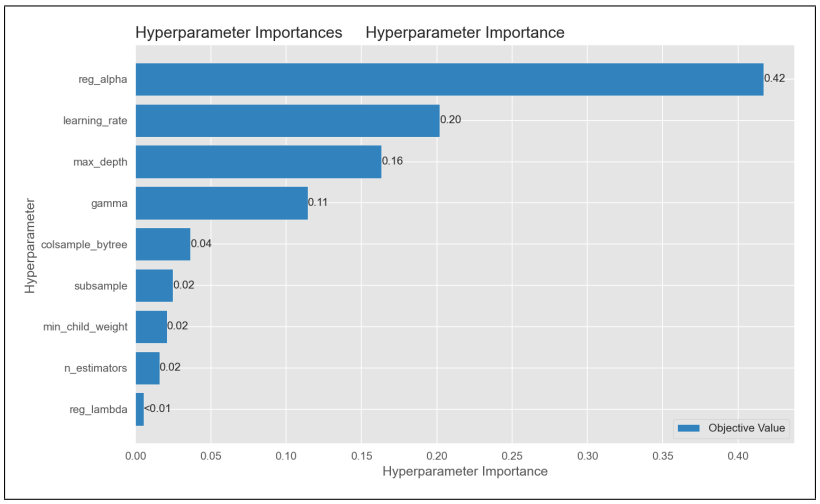

Figure 1: Optuna parameter importance. Larger bars indicate higher influence on validation macro-F1.

The per-class F1 changes in Table 1 deserve particular attention, because macro-F1 weights all three classes equally. Optuna's improvement in the `Low` class (minority) from 0.6944 to 0.7620 (+0.0676) is especially large, while the `High` class F1 increased from 0.8905 to 0.9084 (+0.0179) and the `Standard` class from 0.8583 to 0.8854 (+0.0271). This broad-based improvement across all three classes shows that TPE successfully optimised the class-balanced objective rather than overfitting to the majority class. The tuned model did not sacrifice performance on any single class to achieve higher overall metrics, which is exactly what the macro-F1 metric rewards.

3. K-Means vs GMM

K-Means assigns each sample x_i to the cluster $c_i \in \{1, \ldots, k\}$ whose centroid μ_c minimises the squared Euclidean distance $\|x_i - \mu_{c_i}\|^2$. This is a **hard assignment**: each sample belongs to exactly one cluster, with no notion of uncertainty or partial membership. GMM (Gaussian Mixture Model) takes a fundamentally different approach by modelling the data as a mixture of k multivariate Gaussian distributions: $p(x) = \sum_{j=1}^{k} \pi_j \mathcal{N}(x \mid \mu_j, \Sigma_j)$, where π_j are the mixing proportions. Each sample receives a posterior probability $p(c_j \mid x_i)$ for every component, enabling **soft assignment**: a sample can belong partially to multiple clusters. For insurance risk, where applicant profiles naturally overlap across risk bands rather than forming isolated groups, soft assignment is more aligned with the domain.

Table 2 reports the complete clustering results from `outputs/tables/clustering_comparison.csv`, covering k=2 through k=8. The columns are: k, K-Means inertia, K-Means silhouette score, GMM log-likelihood, GMM BIC, GMM AIC, and GMM silhouette score. K-Means silhouette scores remain low across the entire range, peaking at only 0.2015 at $k = 8$. This confirms that even the best K-Means configuration fails to find well-separated spherical clusters in this data. GMM achieves substantially higher silhouette scores: 0.4142 at $k = 2$ and 0.4015 at $k = 5$, which are roughly double the best K-Means values. At $k = 2$, the GMM silhouette of 0.4142 versus K-Means's 0.1740 is particularly revealing: it suggests that the two-cluster structure in this insurance dataset is inherently probabilistic (overlapping Gaussian components) rather than discrete (centroid-defined).

Table 2: Full clustering comparison across k=2 to k=8.

k	K-Means Inertia	K-Means Sil	GMM BIC	GMM AIC	GMM Sil
2	1,092,962	0.1740	−359,251	−362,062	**0.4142**
3	1,018,587	0.1732	−1,103,445	−1,107,666	0.2977
4	953,249	0.1808	−1,938,815	−1,944,446	0.3964
5	889,285	0.1964	−1,997,256	−2,004,298	0.4015
6	818,951	0.1768	−2,349,766	−2,358,217	0.2468
7	777,658	0.1971	−2,394,381	−2,404,243	0.3110
8	691,941	**0.2015**	−2,510,221	−2,521,493	0.1726

The GMM BIC column in Table 2 shows a monotonic decrease with larger k, which is expected since adding more components always allows a better fit to the training data. However, BIC also penalises model complexity, so the rate of decrease slows at larger k, suggesting diminishing returns. The K-Means inertia curve is gradual with no sharp elbow, indicating the absence of a natural cluster count—another sign that the data does not contain clearly separable spherical structures. Overall, the GMM's consistently higher silhouette scores across most values of k indicate that insurance applicants form probabilistic subtypes with soft boundaries. This validates the conceptual distinction between hard and soft assignment: GMM captures the overlapping nature of risk profiles that K-Means cannot represent. Importantly, neither clustering method is intended to replace the supervised classifier—they serve different objectives and the unsupervised analysis is purely exploratory.

4. Personalised Improvement Reflection

My compulsory category was **Category A: Data Quality and Missingness**. Before any modelling, I conducted an EDA that identified significant missing values in multiple columns. Five columns had particularly high missing rates: net_monthly_income_gbp, avg_payment_delay_days, monthly_investment_gbp, prior_debt_products, and account_tenure (at 30.6, 19.0, 21.1, 7.6, and 4.3 percent respectively). Rather than treating missing values as noise and simply applying median imputation, I added five binary missing-indicator features—one for each of these five columns—appending them to the feature set alongside median imputation. This is based on the hypothesis that the *pattern* of missingness itself may be informative: a missing income value might indicate financial instability or unemployment, which is a legitimate risk signal in insurance.

After adding the five missing indicators, validation macro-F1 rose from 0.8520 (the Optuna-tuned model) to 0.8529 (Category A XGBoost). The gain is modest ($+0.0009$) but meaningful, given that the tuned model was already strong and operating close to the performance ceiling implied by the feature space. More importantly, the gain confirms the hypothesis that missingness carries behavioural signal: in financial applications, missing income data does not occur at random and is therefore legitimately predictive. This also demonstrates an important methodological lesson: even small improvements should be interrogated to determine whether they reflect genuine signal or overfitting.

For my optional category, I implemented **Category D: Soft Voting Ensemble** by combining Random Forest and tuned XGBoost using soft voting (averaging predicted class probabilities). The ensemble achieved validation macro-F1 of 0.8510, which is below both the Category A model (0.8529) and the tuned XGBoost alone (0.8520). This outcome is instructive: it shows that model diversity alone is insufficient for ensemble improvement. The two base learners had very different prediction profiles—RF overfitted dramatically while XGBoost was well-calibrated—and combining them diluted the boosting model's advantage rather than complementing it. In practice, effective ensembles typically require base learners that are both individually strong and diverse in their error patterns. My final model selection was therefore the Category A XGBoost, chosen strictly on the basis of validation macro-F1 evidence.

A critical prerequisite for all modelling steps was data leakage control. Before any model training, I screened all available features using single-feature DecisionTree cross-validation. The feature `bureau_risk_index` achieved a single-feature macro-F1 of 0.9999—an extraordinarily high score that indicates near-perfect class separation. This immediately triggered the leakage detection threshold (set at 0.85), and the feature was removed before any further experimentation. This step is fundamental: without removing the leakage feature, all subsequent validation scores in Table 1 would be artificially inflated and every model comparison would be invalid. The leakage check also illustrates an important broader principle in applied machine learning: even when a feature appears to improve performance, it must be evaluated for its relationship to the target before being accepted.

5. AI Use Declaration

AI tools were used only in a limited support role throughout this coursework: they assisted with environment debugging (resolving package import and GPU configuration issues), and with LaTeX formatting to produce the final document. The experimental design, leakage detection decision, controlled model comparison, personalised improvement strategy, and all written interpretations of tables, figures, and metrics were derived from my own notebook results. No claims are made about hidden-test performance; the CSV file (`test_result_1234560.csv`) follows the required filename and column order from the assignment brief, generated solely for submission formatting.