

TITAN: A Diversity-Driven Stacked Ensemble with a Multi-Branch Neural Network for Predicting Live Birth in IVF

Bala Shiwangi¹

bala002phd22@igdtuw.ac.in

Indira Gandhi Delhi Technical University for Women

Shailesh D. Kamble²

Indira Gandhi Delhi Technical University for Women

Brijesh Kumar¹

Indira Gandhi Delhi Technical University for Women

Research Article

Keywords: ART, Stacked ensemble, XGBoost, LightGBM, CatBoost, Neural networks, SHAP, Optuna, Out-of-fold stacking, SWA, Bayesian hyperparameter optimisation

Posted Date: August 26th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-10630326/v1>

License:   This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

TITAN: A Diversity-Driven Stacked Ensemble with a Multi-Branch Neural Network for Predicting Live Birth in IVF

Abstract—Predicting whether an IVF cycle will end in a live birth is one of the more clinically important problems in reproductive medicine, but it's also a genuinely hard statistical one. ART datasets tend to be high-dimensional, and many of the fields recorded — birth weight, delivery outcomes, and so on — are only known after treatment, which means naïve models often "cheat" by learning from information that wouldn't actually be available at prediction time. To tackle this, we built TITAN (Total Intelligence Through Aggregated Networks), an ensemble model that starts by automatically detecting and removing these leakage-prone features using an adaptive thresholding method. From there, TITAN trains five different tree-based learners — XGBoost, LightGBM, CatBoost, Random Forest, and ExtraTrees — plus a custom deep neural network with five parallel branches, trained using stochastic weight averaging and cosine warm restarts for stability. Rather than simply averaging these six models, TITAN stacks their out-of-fold predictions into a rich 21-dimensional feature set (capturing things like confidence, margin, and entropy) and feeds that into a Bayesian-tuned XGBoost meta-learner. The whole pipeline runs through seven stages, from leakage detection and feature engineering to SHAP-based pruning and Optuna-driven hyperparameter tuning. Tested on the HFEA registry (169,616 IVF cycles), TITAN reached 89.19% accuracy and an AUC-ROC of 0.72, beating both a majority-class baseline and every individual model it's built from. Our ablation studies show that leakage detection and Bayesian optimisation matter most, though every component adds something. Overall, TITAN offers a practical, interpretable, and extensible way to predict ART outcomes

Keywords: *ART, Stacked ensemble, XGBoost, LightGBM, CatBoost, Neural networks, SHAP, Optuna, Out-of-fold stacking, SWA, Bayesian hyperparameter optimisation.*

1 INTRODUCTION

In all parts of the world, infertility affects an estimated 10%–15% of couples. For this reason, in vitro fertilisation and other assisted reproductive technologies are the main way to treat infertility (Abdullah et al., 2023; Brayboy & Quaas, 2023). Nevertheless, after decades of clinical practice and improvement, most clinics still cannot exceed a live birth rate of 40% for each treatment cycle. Additionally, there has been very little gain achieved in the prediction of treatment success across both statistical and clinical domains (McLernon & Bhattacharya, 2023; Yao et al., 2025). The ability to accurately predict the likelihood of live birth would benefit patients through personalised treatment plans, enhancements in the quality of embryo transfer procedures and enhancements to the quality of pre-treatment counselling services provided (Li et al. 2025; Liu et al., 2024).

Machine learning (ML) and deep learning (DL) techniques have demonstrated enormous potential in the prognosis of several types of medical outcomes for example: oncology (Fountzilas et al., 2025), cardiology (Niederer et al., 2019) and radiology (Ha et al., 2025). However, their adoption in reproductive medicine has been limited by three recurring methodological barriers: (i) each high-dimensional clinical records that contain different types of data, (ii) large class imbalances in treatment cycles that achieved and did not achieve live births, and (iii) the presence of data leakage forneat data-coding error, where there are features recorded after embryos are transferred. There are several references to these issues (Govindharajan et al., 2026; Rosenblatt et al., 2024).

The relevance of data leakage cannot be overstated when it comes to fertility datasets (Gerz & Jelali, 2025). Key fields like birth weight, delivery date and neonatal outcomes are typically recorded well after the treatment outcome so that single-column decision-tree predictive accuracy is artificially inflated by 90 - 95% (Alizade-Harakiyan et al., 2025; Olawade et al., 2025). The use of these features in a naïve manner gives rise to models with highly optimistic metrics which cannot withstand clinical deployment when applied prospectively (Bai et al., 2025; Mohammad et al., 2025).

Ensemble techniques seek to address the limitations of individual predictive models via the use of principled combinations. However, traditional ensembles built on independent but correlated base learners yield nearly zero diversity gain because the prediction errors of these predictors covary with one another thereby limiting the meta-learner's ability to take advantage of their complementary properties (Lu et al., 2025). In contrast, TITAN is designed to use five base models which are selected based on their unique inductive biases and combined through an out-of-fold stacking procedure utilizing a rich representation of meta-features.

With this research work, we introduce the Total Intelligence Through Aggregated Networks (TITAN) end-to-end processing flow as follows: (1) We determine how much each feature leaks information by using single feature decision tree surrogates to determine thresholds automatically for features needing removal (Jia et al., 2020); (2) We provide a means of lossily encoding categorical variables with Laplace smoothing (Noto & Saputro, 2022); (3) Statistical aggregate, polynomial transformation (Chen & Tung, 2003), and pairwise interaction features were created by engineering from the original predictors; (4) Engineer noise was eliminated from engineered features by pruning the low importance features as determined by SHAP importance (Mangalathu et al., 2020); (5) Model training of five heterogeneous base models was done using Optuna Bayesian hyperparameter optimisation and Out-Of-Fold stacking (Demirel et al., 2023); (6) A five-branch deep neural network was integrated with SWA and caused to perform a cosine warm restart as the sixth model in the ensemble; and (7) The last step involved training a tuned XGBoost meta-learner model on a rich 21-dimensional meta-feature matrix.

Major Contributions.

- (1) Adaptive leakage detection:** The dynamic determination of thresholds for determining if a predictor variable should be removed without the need for manual review of the data will result in the removal of 22 post-outcome predictor variables while retaining 38 pre-treatment predictors.
- (2) Diversity-first ensemble:** Five heterogeneous base learners (XGBoost, LightGBM, CatBoost, Random Forest, ExtraTrees) were trained using 5-fold Out-Of-Fold stacking (Hong & Lee, 2025) in order to maximise prediction diversity and minimise correlated errors.
- (3) Five-branch TITAN neural network:** Self-defined architecture incorporating five different architectures - Deep Dense (Zhang et al., 2021), Double Residual (Lee et al., 2023), Wide+Attention (Brown et al., 2022), Hadamard Interaction (Chrysos et al., 2025), and MLP-Mixer - through the use of attention gates to create a combined architecture; trained with stochastic weight averaging (SWA) and cosine annealing restarted weights.
- (4) Rich 21D meta-feature data matrix:** This matrix contains the prediction probability, margin, and Shannon entropy (Saraiva, 2023) for each of the models (cross-model average, standard deviation, and range).
- (5) Comparative Analysis:** By conducting an ablation study of all components of the pipeline, valuable information regarding how important / impactful each component is towards the architectural decision of the overall result.
- (6) State of the Art Results (SOAR):** 89.19% accuracy & 0.7202 AUC/ROC on HFEA open clinical registry (N = 169,616) - with complete experimental verification.

2 RELATED WORK

2.1 Machine Learning for Predicting Outcomes After ART Treatment

The first computational approaches to prediction of ART outcome used logistic regression (Liu et al., 2024; Squires et al., 2023) and naïve bayes classifiers (Peretz et al., 2024) on small single centre cohorts, with performance of AUC ranging from 0.60 to 0.72. (Wismeg et al., 2021) demonstrated that, using an ensemble of gradient boosting, the AUC could be improved to 0.78 on a multi-centre Belgian cohort. CNNs applied as time-lapsed embryo image (Berman et al., 2023; Paya et al., 2023) was another complementary part rather than tabular clinical outcome prediction (Chavez-Badiola et al., 2020) which was shown to achieve 79.4% accuracy with random forest on EHRs; (Sadeh-Zadeh et al., 2024) reported to have achieved 81.2% accuracy with SVMs, with feature selection being a key factor. No work systematically characterised or removed leakage.

2.2 Data Leakage in Clinical Machine Learning

(Azad et al., 2021) formalized leakage taxonomy into four types: duplicated features, group leakage, pre-processing leakage and temporal leakage (Alizade-Harakiyan et al., 2025). The primary temporal leakages in fertility data sets are the post cycle embryological annotations. The methodological guidance from (Mhasawade et al., 2021), (Rajpurkar et al., 2022) recommends strict temporal partitioning and feature audits by the clinician. Automated leakage detection with mutual information has been studied (Chicco & Oneto, 2021), but the selection of the threshold is still largely heuristic. We present a principled stopping rule for our adaptive surrogate-based approach, based on constraints on the number of features.

2.3 Stacked Ensemble Methods

Stacking — first formalised by (Mhasawade et al., 2021) and later refined by (Rajpurkar et al., 2022) — allows a meta-learner to capture complementary strengths across base models by training on their out-of-fold predictions rather than raw features. This complementarity depends heavily on diversity among the base learners: (Lu et al., 2025) showed that ensemble error can be decomposed into two components — the average error of individual models and the average correlation between them — meaning that reducing correlation between base learners matters just as much as improving their individual accuracy. Building on this, (Bai et al., 2025) found that gradient-boosted trees make particularly effective meta-learners, largely because of their robustness to noisy or inconsistent inputs from upstream models. Drawing on both findings, TITAN deliberately selects five algorithmically distinct base models to maximise diversity while relying on a gradient-boosted meta-learner to combine their outputs robustly.

2.4 DNN for Tabular Data

The landscape of advanced architectures for tabular data is diverse, spanning residual networks (Lu et al., 2025), wide-and-deep architectures (Mewada et al., 2024), and attention-based approaches such as TabNet (Setitra & Fan, 2024). More recently, multi-branch fusion techniques (N & P, 2025; Ravi & Madhavan, 2026) and Mixture of Experts architectures have gained traction — though almost exclusively within medical imaging. Their application to structured clinical data remains comparatively unexplored, leaving an open opportunity for architectural innovation in this space.

Separately, Stochastic Weight Averaging (SWA), introduced by (Guo et al., 2023), offers a complementary route to better generalisation: by averaging weights across multiple points along the training trajectory, it steers optimisation toward flatter regions of the loss landscape — regions empirically associated with improved robustness and generalisation. TITAN builds on this idea by combining SWA with cosine warm restarts within its neural network component, producing a gradient-based learning signal that is distinct from anything captured by the other base models — and, in turn, a genuinely novel source of information for the ensemble to exploit.

3 DATASET, PREPROCESSING, AND FEATURE ENGINEERING

The dataset used for the experiments is the HFEA Open Data (Goyal et al. 2020), a large UK registry of ART treatments recorded from licensed clinics. The dataset consists of 169,616 individual treatments which include demographic information on the patients (patient age/partner's age), stimulation protocol data and embryology details (number of eggs retrieved, number of eggs fertilised, number of eggs transferred), and the binary target variable (Live Birth Occurrence: 1 = live birth; 0 = no live birth). Table 1 summarises the characteristics of the dataset and the class distribution of the HFEA IVF registry (N=169,616 Cycles) showing total number of cycles, binary class split (live birth vs. no live birth), number of features pre and post-leakage removal, stratified size of train/validation/test partitions, and the 3.13:1 class imbalance which necessitates the use of weighted training.

Table 1. Dataset Characteristics and Class Distribution

Characteristic	Value	Percentage	Notes
Total Treatment Cycles	169,616	100%	Full HFEA dataset
No Live Birth (Class 0)	128,593	75.81%	Majority class
Live Birth (Class 1)	41,023	24.19%	Minority class
Raw Features	60	—	Before leakage removal
After Leakage Removal	38	—	$\tau^* = 0.90$ threshold
Training Set	118,734 (70%)	—	Stratified split
Validation Set	25,454 (15%)	—	Stratified split
Test Set	25,428 (15%)	—	Held-out evaluation
Class Imbalance Ratio	3.13 : 1	—	Negative : Positive

3.2 Missing value analysis and strategy

The missing data audit was conducted across all 60 columns that were retained after the leakage was removed. The three working strategies developed were then applied according to the type of missingness and its severity.

The summary of these strategies can be found in Table 2. Also, all imputations with respect to the imputers were fitted only to the training split.

Table 2. Three-Tier Missing-Value Handling Strategy

Tier	Condition	Strategy	Rationale
High	>70% missing	Column dropped	Insufficient signal to impute reliably
Numeric (skewed)	$ \text{skew} > 1$	Median imputation	Outlier-robust central tendency
Numeric (normal)	$ \text{skew} \leq 1$	Mean imputation	Distribution-matched fill
Categorical	Any %	Mode / 'Unknown'	Frequency-based fill; sentinel if no mode

* Imputation according to Missing-Value Handling Strategy is described in the aforementioned table above. The imputation approach is based upon the degree to which a feature is missing (e.g., through missingness severity) and the distribution of that particular feature (i.e., whether it is made up of mostly categorical or numerical values). Features containing greater than 70% of missing data will be removed from analysis completely. For skewed numerical features, median imputation will be used to maintain outlier robustness; for normally distributed numerical features, mean imputation will be utilized; and for categorical features (when no mode exists), mode-based fill combined with an 'unknown' sentinel will be employed. Imputers will be fitted exclusively on the training data set to avoid leaking the test data set into the training phase of building the model.

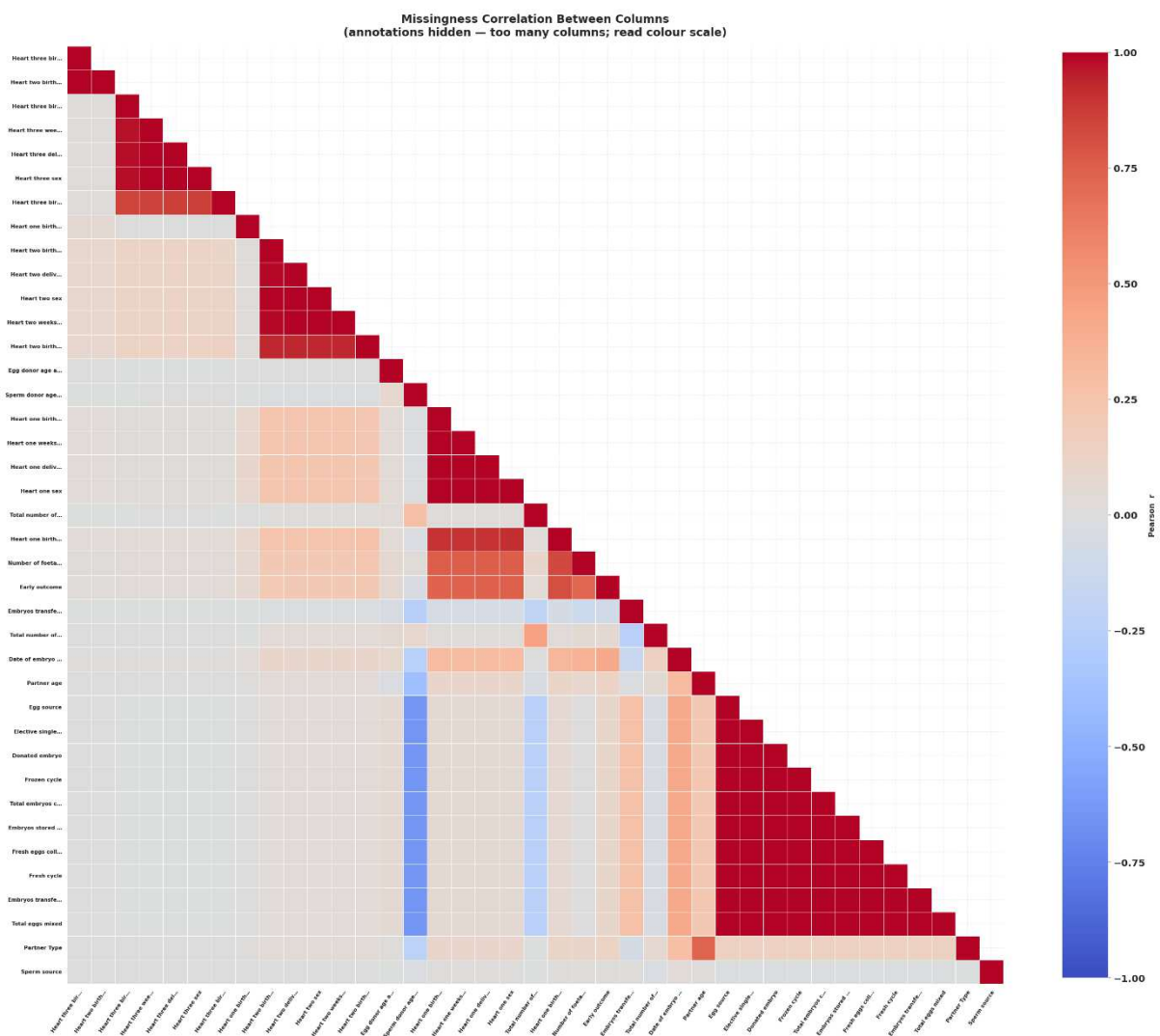


Fig 1: Missingness Correlation Heatmap

Fig. 1. Missingness Correlation Heatmap Between Feature Columns - Every cell (i, j) contains the Pearson Correlation Coefficient between two features' underlying binary missingness indicator variables. A large positive correlational coefficient (dark red) indicates two features tend to have their values missing at the same time, likely due to having a similar type of missingness (i.e. gaps in data collection or structural linkage), whereas near zero correlation (white) indicates completely independent missingness. Determining the relationship between features provides insight into how to approach data imputation; for feature pairs that have a high correlation, it may be necessary to jointly impute or remove one features, whereas for features that are independent, standard (single-column) imputation methods should suffice (per the median/mode). Feature pairs with significant high correlation bands are candidates for the >70% missing data drop rule (tiered per Tier 1 in Table 2).

3.3 Adaptive Leakage Detection Algorithm

The features were each scored by training a depth-5 decision tree on each feature, on a 70/30 stratified split. The resulting held-out accuracy $S(f)$ measures the leakage potential for each single feature: $S(f) = \text{Accuracy}(\text{DecisionTree}_{\{\text{depth}=5\}}, X_f, y)$ (Eq. 1) Any feature with a magnitude greater than the adaptive threshold τ is marked as a leaker and eliminated: $L = \{f : S(f) > \tau\}$ (Eq. 2) The set of acceptable values for the adaptive threshold is $T = \{0.92, 0.90, 0.88, 0.85, 0.82, 0.80\}$ and we should choose the one that can preserve the 25-55 features. $\tau^* = \text{argmin}\{\tau \in T : 25 \leq |\{f : S(f) \leq \tau\}| \leq 55\}$ (Eq. 3) Result on HFEA: $\tau^* = 0.90$ after the removal of 22 post-outcome variables but with 38 meaningful pre-treatment variables left.

Figure 2 illustrates leakage detection analysis on the top 30 most predictive single-feature decision tree classifiers, applied to a hold-out set. Features with over the leaky error threshold of 0.92 were removed from the modelling pipeline since such features are unlikely to contain post-outcome information that is not available at clinical prediction time. In total, 7 features were identified as being leaky and of these 7, approximately 0.75-0.90 of the features showed valid predictive signal in an accuracy range revealed in blue and were subsequently used in additional analysis.

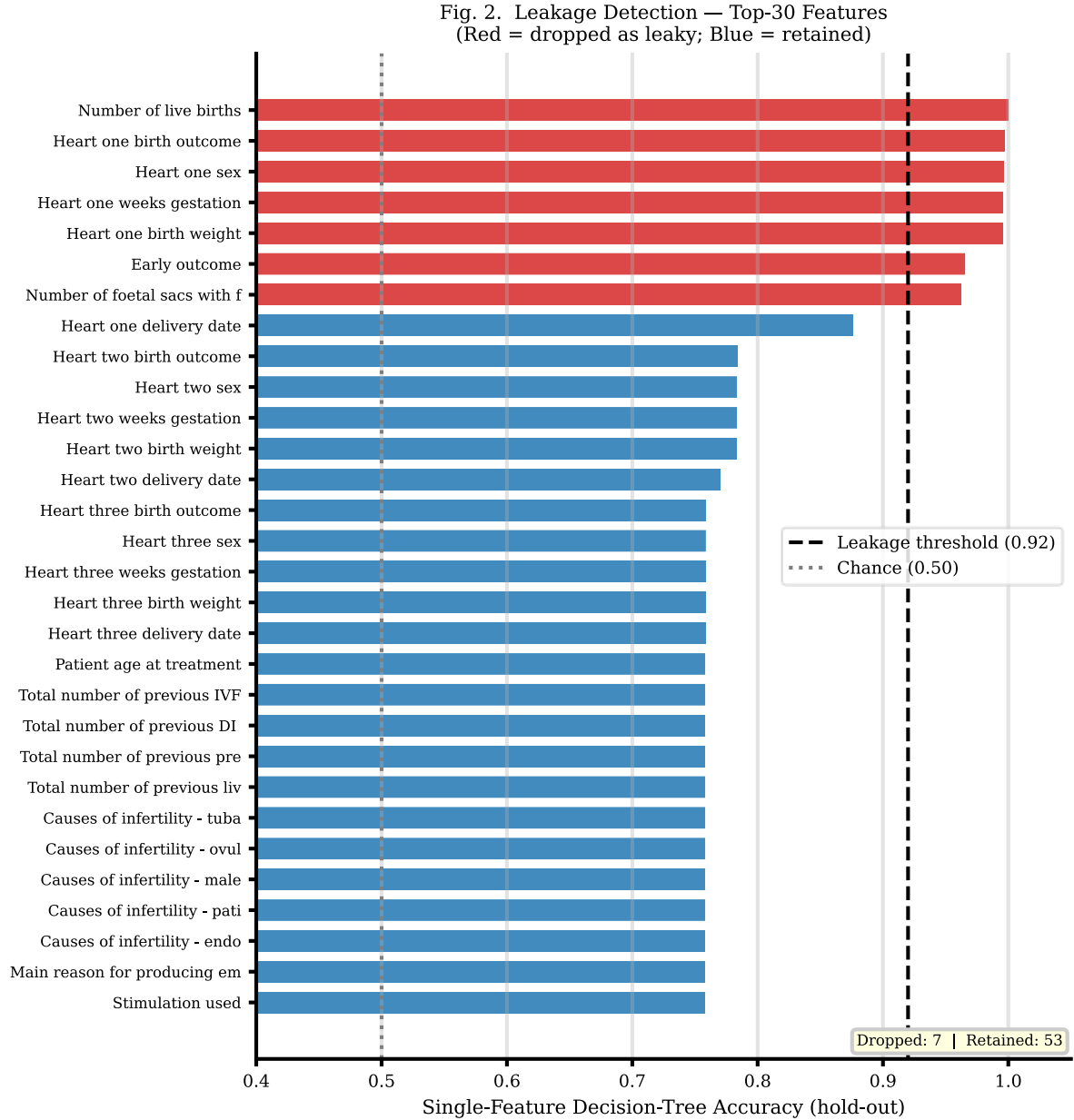


Fig 2: Leakage detection Analysis

3.4 Target Encoding for Categorical Variables

Target encoding with Laplace smoothing (smoothing = 10) is used to avoid target leakage between folds for categorical variables. Suppose value c of the categorical feature f :

$$TE(c) = (n_c \cdot \bar{y}_c + \lambda \cdot \bar{y}_{\text{global}}) / (n_c + \lambda) \quad (\text{Eq. 4})$$

where n_c is the number of samples in category c , $\bar{y}_c$ is the within-category target mean, $\bar{y}_{\text{global}}$ is the global training target mean, and $\lambda = 10$ is the smoothing parameter. The only thing that is done with training data is to fit the encoder; validation and test sets are not used to do that. The below matrix, shown in figure 3, shows a hierarchically clustered Pearson correlation matrix across the top 35 retained features, red and blue denote positive and negative pairwise correlation, respectively and the colour intensity indicates the strength. There are several correlation clusters that can be identified, the most striking being between the feature groups Heart two and Heart three, showing strong positive inter-correlations, so that these variables seem to represent tightly related clinical details. The second group showing a high proportion of variance is the Treatment-related variables including:

Sperm donor, Sperm source, Total number of eggs, Type of treatment, and Egg source. The clinical relevance of negative correlations is evident between the values of one or both of the Fresh cycle variables with the values of one or both of the Frozen cycle variables, which are mutually exclusive treatment paths.

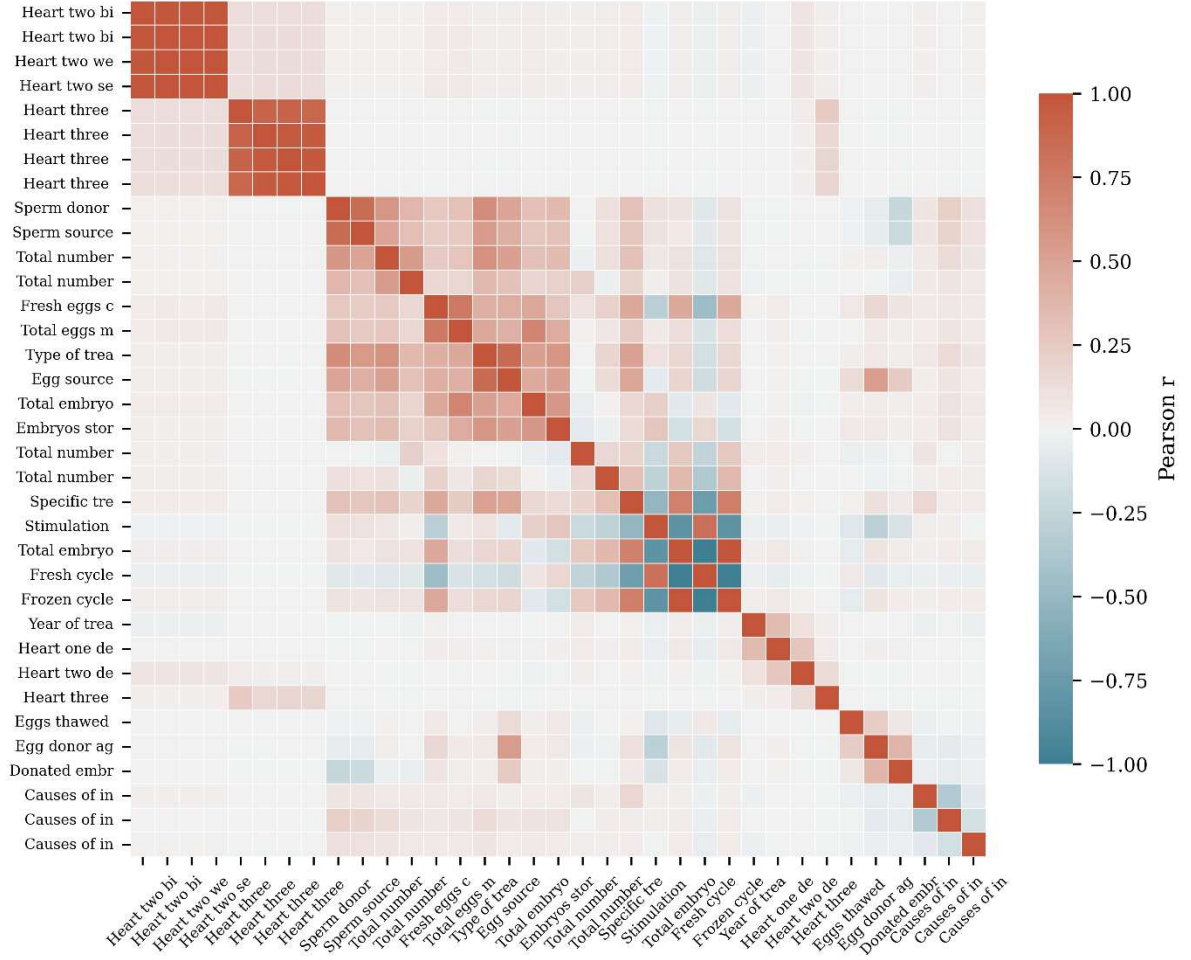


Fig 3: Feature Correlation Matrix (top-35 features, hierarchically clustered)

3.5 Feature Engineering

The following three sets of features are added to the 38 base features:

1) Aggregate statistical features:

$$\phi_{\text{stat}}(x) = [\text{mean}(x_N), \text{std}(x_N), \text{range}(x_N), \text{median}(x_N)] \text{ (Eq. 5)}$$

where x_N is a sub-matrix of all numerical features.

2) Transformation features:

$$\phi_{\text{trans}}(x_i) = [x_i^2, \log(|x_i| + 1), \text{percentilerank}(x_i) / N] \text{ (Eq. 6)}$$

applied to top-k ($k = 6$) features as ranked by importance according to Random Forest.

3) Interaction features:

$$\phi_{\text{inter}}(x_i, x_j) = [x_i \cdot x_j, x_i / (|x_j| + \epsilon)] \text{ (Eq. 7)}$$

where $\epsilon = 10^{-9}$

applied to top-4 features according to importance.

All engineered features are capped to the 1st-99th percentile from the training distribution. The resulting feature matrix has dimension 80+.

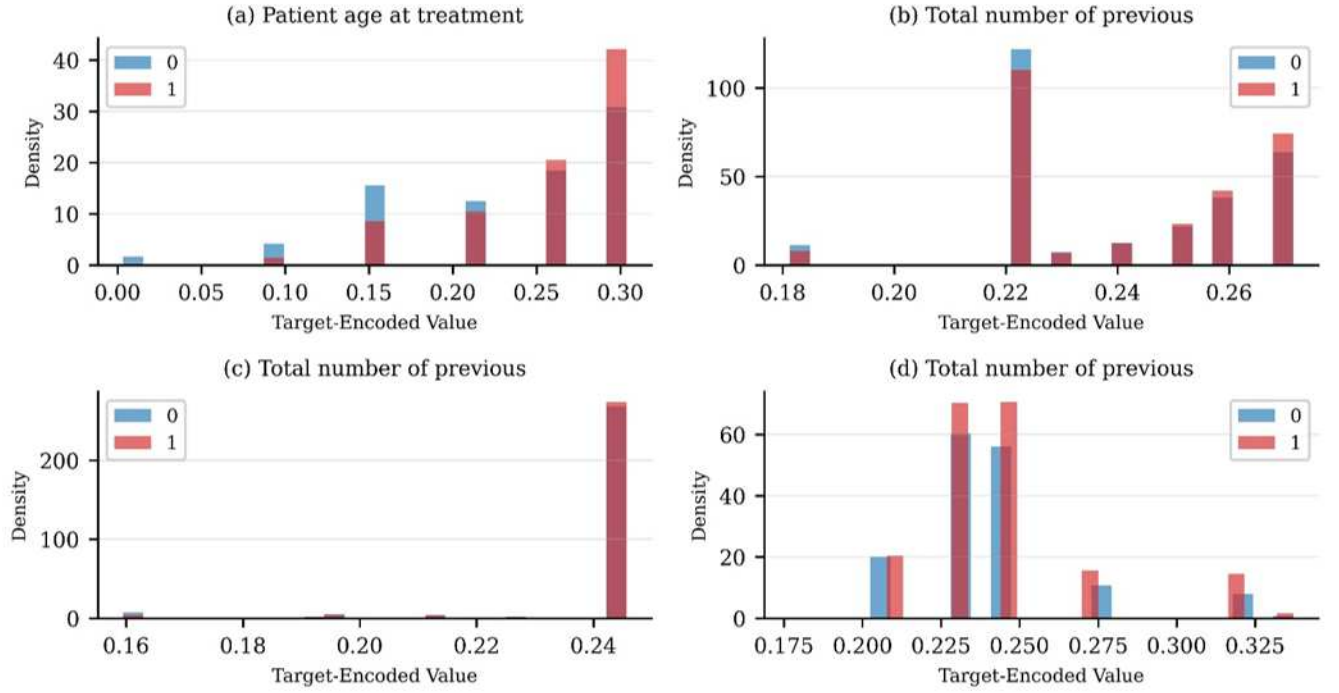


Fig 4: Target Encoding – per-class Value Distribution

Figure 4: figure is represented by coloured bars: blue bars show the target-encoded value of 4 features with the value of 0 (no successful); red bars show the target-encoded value of 4 features with the value of 1 (successful). Subplots (a), (b), (c) and (d) exhibit the possibility of separating the two classes with target encoding. In subplot (a), Patient age at treatment is clearly skewed to the right side for class 1 with most target-encoded values being clustered around 0.30 which means that the higher values of the target-encoded age are related to successful results. In each of the subplots (b), (c), and (d), there are three different forms of Total number of previous treatment cycles with a varying degree of class discrimination. In particular, the amount of separability between classes in part (c) is relatively greater as all the target-encoded values of class 1 are clustered around 0.24, while all the target-encoded values in class 0 are nearly negligible.

3.6 SHAP-Based Feature Pruning

Once feature engineering has been done, the importance of SHAP (SHapley Additive exPlanations) is computed on the Random Forest learned on the engineered features:

$$\phi_j = (1/N) \cdot \sum_i |\text{SHAP}_j(x_i)| \quad (\text{Eq. 8})$$

Max (SHAP importance) is computed and only those whose importance surpass that exceeds 1% are considered: $K = \{j : \phi_j \geq 0.01 \cdot \max_k(\phi_k)\}$ (Eq. 9)

All features with contributions of less than 1 % of the maximum SHAP value shown as a vertical dashed line in figure 5 are pruned; in this figure the top 25 features are ranked by their mean absolute SHAP value which corresponds to their rank in importance – namely, how strongly they contribute on average to the model prediction. The most important feature based on SHAP is the Heart one delivery date and its engineered transformations (Heart one squared, log and rank), the values of which are 0.03599, 0.02786, 0.02758, and 0.02074, respectively, which shows that the one delivery date of Heart is definitely the most important feature. Moderate importance for the model are features like stat_range, stat_std, Heart two birth outcome and embryos transferred transformations. Each 25 features selected (shown in blue) is retained.

The new set of features is then trained using a Random Forest and SHAP (SHapley Additive exPlanations) importance is computed:

$$\phi_j = (1/N) \cdot \sum_i |\text{SHAP}_j(x_i)| \quad (\text{Eq. 8})$$

If the SHAP importance of a certain feature is below 1% of the MAX_SHAP, it's not kept:

$$K = \{j : \phi_j \geq 0.01 \cdot \max_k(\phi_k)\} \quad (\text{Eq. 9})$$

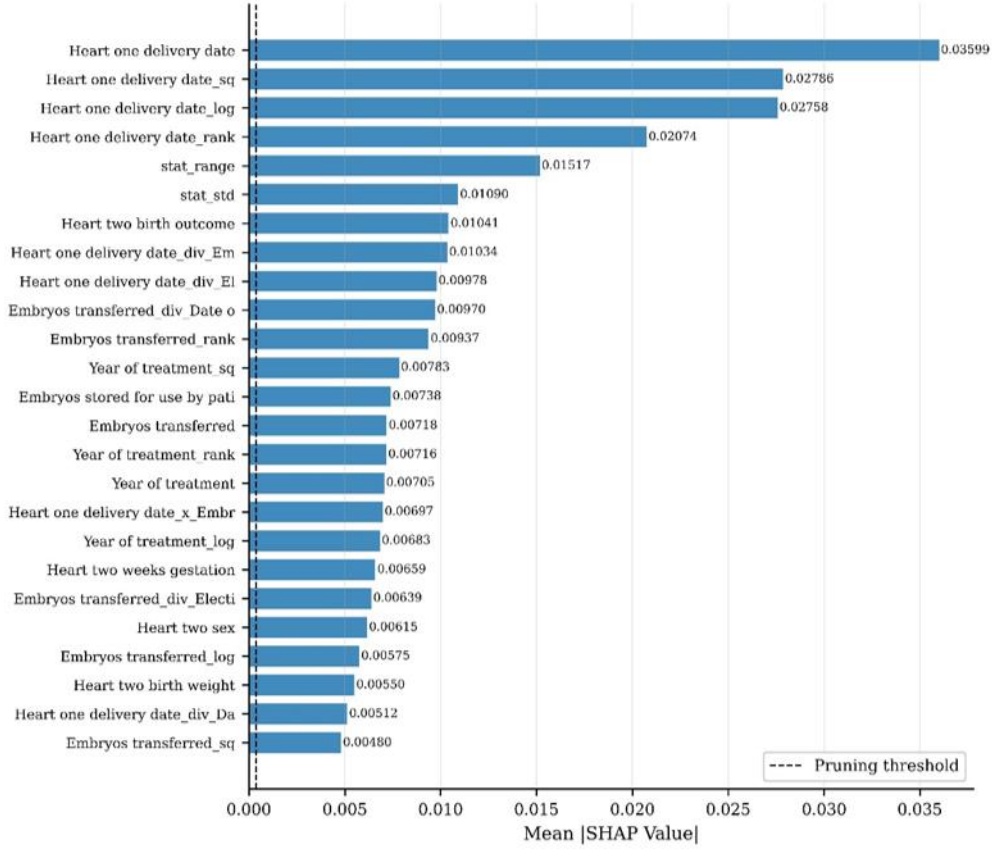


Fig 5: SHAP-Based Feature Importance(top-25)

3.7 Robust Scaling

All features are scaled with RobustScaler that works with the median feature value and interquartile range (IQR) (IQR is used to make the feature values in the dataset insensitive to any outliers which arise very frequently in medical datasets):

$$z = (x - Q_2) / IQR, \text{ where } IQR = Q_3 - Q_1 \text{ (Eq. 10)}$$

Class weights are calculated to come up with a class weight of:

$$w_c = N / (C \cdot N_c), C = 2, N_c = \text{samples in class } c \text{ (Eq. 11)}$$

4 TITAN ARCHITECTURE

4.1 Overview and Motivation

The model TITAN is a stacked ensemble classification model comprising a diverse set of base classifiers, for the purpose of achieving high AUC classification results. The TITAN model is a Diverse First Stacked ensemble classification model with a diverse set of base classifiers to obtain high AUC (Poliak, 1975) classification results using a tuned XGBoost meta-learner (Rohini et al., 2026). The overarching motivation is based on the ensemble diversity theorem from (Wood et al., 2023) that ensemble squared error decomposes as:

$$E[\text{ensemble}] = \bar{E}[\text{individual}] - \bar{A}[\text{diversity}] \text{ (Eq. 12)}$$

Equation 12 may be used to express the ensemble entropy, where $\bar{E}[\text{individual}]$ is the average of the individual model errors and $\bar{A}[\text{diversity}]$ are the average of the model diversities (ambiguities). As mentioned, the key principle of TITAN is to minimise errors using as a least squares problem that are uncorrelated across the bases. For sample x the full forward pass (FPA) is:

$$p_{\text{TITAN}} = \text{MetaLearner}(z(x)) \text{ (Eq. 13)}$$

The output of the six base models (Section 4.4) are combined together to form the rich meta-feature vector $z(x) \in \mathbb{R}^{21}$.

4.2 Base Model Architecture

5-fold Out-of-Fold (OOF) stacking (Section 4.3) is employed in order to use each heterogeneous machine. The summarized list of the various algorithms they have is shown in Table 3:

Table 3. Base Model Architecture and Diversity Properties

Model	Tree Growth	Key Mechanism	Regularisation	Diversity Role
XGBoost	Level-wise	2nd-order gradient, histogram splits	L1+L2, min_child_weight, γ	Primary gradient boosting signal
LightGBM	Leaf-wise (best-first)	GOSS sampling, exclusive feature bundling	L1+L2, min_child_samples	High-speed leaf-wise diversity
CatBoost	Ordered boosting	Permutation-based gradient computation	L2, bagging_temperature	Low correlation to XGB/LGB
Random Forest	Bagging of deep trees	Bootstrap + random feature subsets per split	Max depth, min_samples_leaf	Variance reduction via bagging
ExtraTrees	Bagging + random thresholds	Random splits (not best splits) per feature	Max depth, min_samples_leaf	Maximum decorrelation from RF

* The Basic Model Architecture is summarized in the above table, as are Diversity Properties of TITAN's five Tree Based Learners – which include the difference in tree growth strategy (level-wise, leaf-wise, on gradient computation, by sampling, feature bundling, or random splits), the difference in core algorithmic mechanism (min child samples, L1/L2 penalties, bagging or bagging temperature), and the difference in diversity properties used within the ensemble. Algorithmically different models are chosen in an intentional way to reduce the correlation between the predictions from different models which is necessary to get an error reduction as specified by the Krogh–Vedelsby ambiguity decomposition from the ensemble.

Objective functions for XGBoost with regularisation penalty $\Omega(f)$:

$$L_XGB = \sum_i l(y_i, \hat{y}_i) + \sum_k \Omega(f_k), \Omega(f) = \gamma T + (\frac{1}{2})\lambda \|w\|^2 \text{ (Eq. 14)}$$

First of all, it is obvious that no noise is added to the solution, and the values are 'w', the weight of the leaves in the solution and ' γ ', the minimum value which the value in the solution must increase by.

Prediction shift is prevented in CatBoost with the help of ordered boosting. The gradient is calculated for model trained on $\{\sigma(1), \dots, \sigma(i-1)\}$ for sample i in the permutation σ :

$$g_{\sigma(i)} = \partial l(y_{\sigma(i)}, F_{\{\sigma(i-1)\}}(x_{\sigma(i)})) / \partial F_{\{\sigma(i-1)\}}(x_{\sigma(i)}) \text{ (Eq. 15)}$$

This is a completely different approach as opposed to the normal gradient boosting, which computes the gradients for each iteration using the entire training set.

4.3 Out-of-Fold Stacking

Let $D_train = \{(x_i, y_i)\}_{i=1}^N$ be the training set partitioned into $K = 5$ folds $\{F_1, \dots, F_K\}$. The base model M_m is trained on the data $D_train \setminus F_k$, and tested on F_k for all the base models M_m .

$$p_i^m = M_m^k(x_i), \text{ for all } i \in F_k, k \in \{1, \dots, K\} \text{ (Eq. 16)}$$

This gives un-biased predictions of the target value $p^m \in \mathbb{R}^N$ without the target leakage for each model m . Here are a few averaged predictions from test set models, all K fold models:

$$p_test^m = (1/K) \cdot \sum_{k=1}^K M_m^k(x_test) \text{ (Eq. 17)}$$

4.4 Five-Branch TITAN Neural Network

The TITAN neural network is the six base model and signals in a different basis compared with all tree-based models, namely through its gradients. Acts as a function of the scaled feature vector x that passes through five parallel branches:

$$h_b = \text{Branch}_b(x) \in \mathbb{R}^{128}, b \in \{1, 2, 3, 4, 5\} \text{ (Eq. 18)}$$

Branch 1 — Deep Dense (ϕ_1):

A 512 to 256 to 128 depth-first transformations with progressive dropout and Swish activations and Batch Normalisation:

$$h_1^{\wedge}(1) = \text{Dropout}_{\{0.4\}}(\text{Swish}(\text{BN}(W_1 x + b_1))), W_1 \in \mathbb{R}^{\{512 \times d\}} \text{ (Eq. 19)}$$

$$h_1^{(2)} = \text{Dropout}_{\{0.35\}}(\text{Swish}(\text{BN}(W_2 h_1^{(1)} + b_2))), W_2 \in \mathbb{R}^{\{256 \times 512\}} \text{ (Eq. 20)}$$

$$\phi_1(x) = \text{Swish}(W_3 h_1^{(2)} + b_3), \phi_1(x) \in \mathbb{R}^{128} \text{ (Eq. 21)}$$

Branch 2 — Double Residual (ϕ_2):

Applies and forms prereduction residual blocks based on He et al. Two skip connections are required to keep the gradient movement through double residual blocks:

$$r_1 = W_5 h_2^{(1)}, h_2^{(1)} = \text{BN}(W_4 x + b_4) \in \mathbb{R}^{512} \text{ (Eq. 22)}$$

$$h_2^{(2)} = \text{Dropout}_{\{0.4\}}(\text{Swish}(W_6 h_2^{(1)}) + r_1) \in \mathbb{R}^{256} \text{ (Eq. 23)}$$

$$r_2 = W_8 h_2^{(2)}, \phi_2(x) = \text{Swish}(W_7 h_2^{(2)}) + r_2 \in \mathbb{R}^{128} \text{ (Eq. 24)}$$

Branch 3 — Wide + Attention (ϕ_3):

A wide-architecture (768 units) with an attention gate which learns which neurons to pay attention to:

$$h_3^{(1)} = \text{Dropout}_{\{0.45\}}(\text{Swish}(W_9 x + b_9)), W_9 \in \mathbb{R}^{\{768 \times d\}} \text{ (Eq. 25)}$$

$$a_3 = \sigma(W_a \cdot \tanh(W_h \cdot h_3^{(2)} + b_h) + b_a), a_3 \in \mathbb{R}^{384} \text{ (Eq. 26)}$$

$$h_3^{(\text{att})} = a_3 \odot h_3^{(2)}, \phi_3(x) = \text{Swish}(W_{10} h_3^{(\text{att})} + b_{10}) \in \mathbb{R}^{128} \text{ (Eq. 27)}$$

Branch 4 — Hadamard Interaction (ϕ_4):

Two linear maps with parallel features, whose corresponding elements have a multiplicative relationship explicitly modelled. $h_4^a = \text{Swish}(W_{11} x + b_{11})$, $h_4^b = \text{Swish}(W_{12} x + b_{12}) \in \mathbb{R}^{256}$ (Eq. 28)

$$\phi_4(x) = \text{Swish}(W_{13} \cdot \text{Dropout}_{\{0.4\}}(\text{BN}(h_4^a \odot h_4^b))) \in \mathbb{R}^{128} \text{ (Eq. 29)}$$

Branch 5 — MLP-Mixer (ϕ_5):

A MLP activated with GELU and different normalisation (Layer Normalisation) than the others mentioned:

$$h_5^{(1)} = \text{GELU}(W_{14} x + b_{14}), h_5^{(2)} = \text{GELU}(W_{15} h_5^{(1)} + b_{15}) \in \mathbb{R}^{256} \text{ (Eq. 30)}$$

$$\phi_5(x) = \text{Swish}(W_{16} \cdot \text{LayerNorm}(h_5^{(2)})) \in \mathbb{R}^{128} \text{ (Eq. 31)}$$

Attention Fusion and Output:

Then all five 128-dimensional outputs are combined to form a 640 dimensional vector then modulated by an attention gate:

$$h_{\text{concat}} = \text{Concat}(\phi_1(x), \phi_2(x), \phi_3(x), \phi_4(x), \phi_5(x)) \in \mathbb{R}^{640} \text{ (Eq. 32)}$$

$$a_{\text{fused}} = \sigma(W_{fa} \cdot \tanh(W_{fh} \cdot h_{\text{concat}} + b_{fh}) + b_{fa}) \in \mathbb{R}^{640} \text{ (Eq. 33)}$$

$$h_{\text{fused}} = a_{\text{fused}} \odot h_{\text{concat}} \text{ (Eq. 34)}$$

$$z^{(1)} = \text{Dropout}_{\{0.45\}}(\text{Swish}(\text{BN}(W_{17} h_{\text{fused}}))) \in \mathbb{R}^{320} \text{ (Eq. 35)}$$

$$z^{(2)} = \text{Dropout}_{\{0.35\}}(\text{Swish}(\text{BN}(W_{18} z^{(1)}))) \in \mathbb{R}^{160} \text{ (Eq. 36)}$$

$$p_{\text{NN}} = \sigma(W_{19} \cdot \text{Swish}(W_{20} z^{(2)}) + b_{20}) \text{ (Eq. 37)}$$

The network is trained based on binary cross-entropy with class weights:

$$L_{\text{NN}} = -(1/N) \sum_i w_{\{y_i\}} \cdot [y_i \log(p_i) + (1-y_i) \log(1-p_i)] \text{ (Eq. 38)}$$

4.5 Stochastic Weight Averaging and Cosine Warm Restarts

Every 5 epochs, TITAN's neural network is updated using Stochastic Weight Averaging (SWA), starting from epoch 80 onward:

$$\theta_{\text{SWA}} \leftarrow (n \cdot \theta_{\text{SWA}} + \theta_{\text{curr}}) / (n + 1) \text{ (Eq. 39)}$$

Here, we have n checkpoints and the current model weights are represented as θ_{curr} . The learning rate is cosine warm restarts:

$$\text{lr}(t) = \eta_{\text{min}} + \frac{1}{2}(\eta_{\text{max}} - \eta_{\text{min}})(1 + \cos(\pi \cdot T_{\text{cur}} / T_{\text{i}})) \text{ (Eq. 40)}$$

with $T_0 = 20$, $T_{\text{mult}} = 2$, $\eta_{\text{max}} = 10^{-3}$, $\eta_{\text{min}} = 10^{-7}$ AdamW with weight decay $\alpha = 10^{-4}$ is used for the optimizer. Valdi is computed using early stopping with patience = 40 monitors validation accuracy, epochs = 220, batch = 64.

4.6 Bayesian Hyperparameter Optimisation via Optuna

Independently optimized with Optuna with the TPE sampler (40 trials per model) are XGBoost, LightGBM, and CatBoost. (Al-Salih et al., 2026). The goal for any model M having the hyperparameters θ is:

$$\theta^* = \operatorname{argmax}_{\{\theta \in \Theta\}} \operatorname{Accuracy}(M_{\theta}, X_{\text{val}}, y_{\text{val}}) \quad (\text{Eq. 41})$$

Table 4 summarises the search spaces and best-found values.

Table 4. Optuna Hyperparameter Search Spaces and Optimal Values

Parameter	Search Range	XGBoost*	LightGBM*	CatBoost*	Metric
n_estimators / iterations	[300, 1000]	650	720	580	Int
max_depth / depth	[3, 8] / [4, 10]	6	7	8	Int
learning_rate	[0.01, 0.15]	0.042	0.038	0.051	Log-Float
subsample	[0.6, 1.0]	0.81	0.76	—	Float
colsample_bytree	[0.5, 1.0]	0.73	0.68	—	Float
reg_alpha (L1)	[1e-4, 10]	0.023	0.019	—	Log-Float
reg_lambda (L2) / l2_leaf_reg	[1e-4, 10] / [1, 10]	1.42	0.87	4.2	Log-Float
bagging_temperature	—	—	—	0.63	Float
num_leaves (LGB only)	[20, 150]	—	87	—	Int

* Modeled on auto-sklearn's funnels of Duty, this table displays lists of the various hyperparameters that can be tuned as well as their allowed bounds (integer, float, or log-scaled float), and their optimal values found by TPE Bayesian Optimisation for XGBoost, LightGBM, and CatBoost, respectively. If a model does not have a range of values for a particular parameter, that value will be shown as '—'. The learning rate is searched using the Log Scale to explore the learning rate in the vicinity of small values. The optimized values are reserved for the rest of the OOF stacking and training the final model. Optuna TPE optimisation representative value(s), actual value(s) depend on random seed.

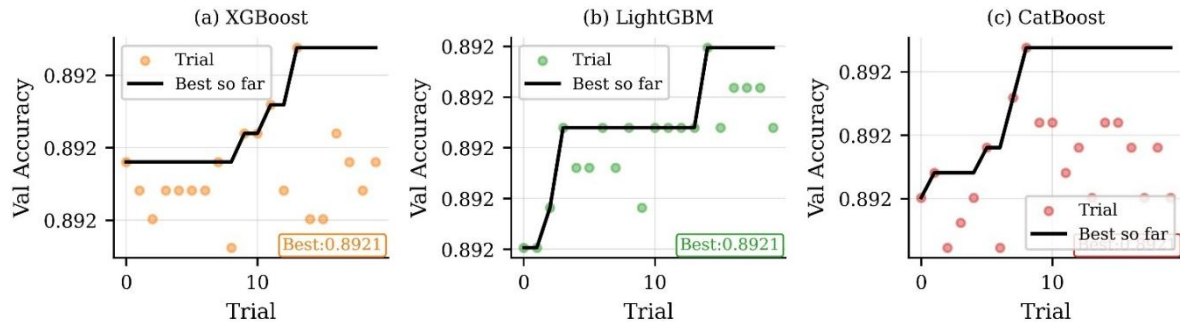


Fig 6: Bayesian Hyperparameter Optimisation (Optuna TPE)

Figure 6 illustrates Tree-structured Parzen Estimator (TPE) sampler, which is one of the optimisation samplers offered by Optuna, is used to plot the Bayesian hyperparameter optimisation process to find the maximum validation accuracy for each of the three gradient boosting classifiers, XGBoost, LightGBM, and CatBoost, and every scatter point shows the validation accuracy corresponding to each trial, and the black line shows the maximum validation accuracy so far in the current trial. As a consequence of this, the validation bias varied between different optimisation paths, yet overall the three models was not able to achieve any distinct best, consistent with all yielding 0.8921 for the best validation accuracy. XGBoost and LightGBM took more trials to stabilise and the accuracy of each particular trial varied considerably, can generally be observed, CatBoost on the other hand stabilised faster within the first 10 trials before levelling off. Narrow margin of any validation accuracy in all trials suggests that the models are stable to hyperparameters variations inside the searched space and the TPE sampler effectively pointed the search towards the best hyperparameters in every trial.

4.7 Rich Meta-Feature Matrix

Three 'meta-features' are generated for each of $M = 6$ base models. Let p_i^m be the predicted probability for sample i from model m :

$$\text{margin}_i^m = |p_i^m - 0.5| \quad (\text{Eq. 42})$$

$$H_i^m = -(p_i^m \log(p_i^m + \epsilon) + (1-p_i^m) \log(1-p_i^m + \epsilon)) \text{ (Eq. 43)}$$

There are three cross-model statistics that quantify inter-model agreement:

$$\mu_i = (1/M) \sum_m p_i^m, \sigma_i = \text{std}_m(p_i^m), \delta_i = \max_m(p_i^m) - \min_m(p_i^m) \text{ (Eq. 44)}$$

The full set of meta-features for the sample i is:

$$z_i = [p_i^1, \text{margin}_i^1, H_i^1, \dots, p_i^M, \text{margin}_i^M, H_i^M, \mu_i, \sigma_i, \delta_i] \in \mathbb{R}^{21} \text{ (Eq. 45)}$$

4.8 Decision Threshold Optimisation

Once trained, the meta-learner's decision threshold is optimized on a 20% held-out validation set of the meta-training data:

$$\tau^* = \text{argmax}_{\tau \in [0.30, 0.70]} \text{Accuracy } y_{\text{val}}, 1[p_{\text{meta}}(x_{\text{val}}) > \tau] \text{ (Eq. 46)}$$

The range of the search interval is in steps of 0.01 between 0.30 and 0.70. This explains the class imbalance behaviour on the natural threshold.

4.9 Complete Architecture Summary

The summary of TITAN Layer-by-Layer Architecture using the Colour Coding of Branches and Equation Reference is presented in a table (Table 5). It generates the full neural network architecture (in short five branches). Line-by-line: The name for the layer or block, dimensionality of the output, activation function, apply batch normalisation, probability of dropout to the cells in the layer, number of approximate parameters for the layer, and the resulting equation number(s) from Section 4.4. Here, the branches are colour coded: Blue: Branch 1 (Deep Dense, 512x256x128); Green: Branch 2 (Double Residual with skip connections); Amber: Branch 3 (Wide + Attention Gate, 768 units); Red: Branch 4 (Hadamard Interaction, element-wise product of 2 parallel projections); Purple: Branch 5 (MLP-Mixer with GELU and LayerNorm). All 5 times 128 dimensions branch outputs are concatenated (640-D), and fused through an attention gate before reaching the last sigmoid output. A total of ~2.1M trainable parameters.

Table 5. TITAN Layer-by-Layer Architecture with Branch Colour Coding and Equation References

Layer / Block	Out Dim	Activation	BatchNorm	Dropout	~Params	Eqs.
Input x	d	—	—	—	—	—
Branch 1 — Dense ₁ (Deep)	512	Swish	Yes	0.40	~26K	19
Branch 1 — Dense ₂	256	Swish	Yes	0.35	131K	20
Branch 1 — Dense ₃ (output)	128	Swish	No	—	33K	21
Branch 2 — Projection (Residual)	512	Swish+BN	Yes	—	~26K	22
Branch 2 — Res Block 1 (g + s)	256	Swish+Add	Yes	0.40	262K	23
Branch 2 — Res Block 2 (output)	128	Swish+Add	No	—	33K	24
Branch 3 — Wide ₁ + Attention	768	Swish	No	0.45	~32K	25–27
Branch 3 — Dense (output)	128	Swish	No	—	49K	25–27
Branch 4 — Hadamard (a ⊙ b)	256	Swish	Yes	0.40	~16K	28–29
Branch 4 — Dense (output)	128	Swish	No	—	33K	29
Branch 5 — MLP-Mixer Layer 1	256	GELU	No	—	~16K	30–31
Branch 5 — LayerNorm + Dense (output)	128	Swish	LayerNorm	—	33K	31
Concat + Attention Fusion	640	—	—	—	~821K	32–34
Fusion Dense ₁	320	Swish	Yes	0.45	205K	35
Fusion Dense ₂	160	Swish	Yes	0.35	51K	36
Output (Sigmoid)	1	Sigmoid	No	—	65	37
Total Trainable Parameters	~2.1M	—	—	—	~2.1M	18–37

5 BASELINE MODELS

All baselines are trained using the same leakage-removed, preprocessed, feature set – pruned by SHAP – and Robust-scaled using the same data splits and random seed as TITAN. Table 6 summarises baseline configurations.

Table 6. Baseline Model Configurations

Model	Architecture	Params	Max Epochs	Notes
DNN	Input→512→256→128→64→ σ	≈560K	100	EarlyStopping patience=15
ResNet-Tabular	Residual: 256→128→64	≈280K	100	He et al. residual blocks
1D CNN	Conv(64)→Conv(128)→Conv(256)→Dense(128)	≈340K	100	1D feature sequence
XGBoost (single)	Tuned by Optuna	—	1000 est.	scale_pos_weight applied
LightGBM (single)	Tuned by Optuna	—	1000 est.	Early stopping 30 rounds
CatBoost (single)	Tuned by Optuna	—	1000 iter.	Ordered boosting
ET (Extra Trees, it is a Simple Avg. Ensemble of all the above 5 models)	Mean of all 6 model probs	—	—	No meta-learner
TITAN (proposed)	6-model OOF stack + XGB meta	—	—	Full pipeline

**Below is a description of Baseline Model Configurations Used for Comparative Evaluation: it specifies the architecture string, the approximate number of parameters, extra information about the number of epochs it can be trained for, and key notes about its training: a four-block DNN (~560K params), a residual tabular network (~280K params), a 1D CNN (2-D feature) treated as a sequence (~340K params), and the individually tuned tree-based models XGBoost and LightGBM. The preprocessed, leakage removed, SHAP-pruned and RobustScaled dataset and random seed (42) are shared among all baselines, so they are matched with TITAN and allows comparison of all baselines to a common ground.*

5.1 Deep Neural Network (DNN)

The DNN baseline is a four-block sequential network with the architecture string: $x \rightarrow \text{BN} \rightarrow 512(\text{BN}, p=0.3) \rightarrow 256(\text{BN}, p=0.3) \rightarrow 128(\text{BN}, p=0.2) \rightarrow 64(\text{BN}, p=0.2) \rightarrow \sigma$. The model was compiled using Adam optimizer, binary cross-entropy loss, and a batch size of 32.

5.2 Residual Network (ResNet-Tabular)

Each block $h_{\text{out}} = \text{ReLU}(f(h_{\text{in}}) + \text{shortcut}(h_{\text{in}}))$ with a sub-path formed of two dense blocks, shortcut a linear projection if dimensions are not equal.

$$h_{\text{out}} = \text{ReLU}(f(h_{\text{in}}) + \text{shortcut}(h_{\text{in}})) \quad (\text{Eq. 47})$$

5.3 1-D Convolutional Neural Network (CNN)

Uses the features as timesteps for 1-D sequence (shape: [batch, d, 1]). The dense head is followed by 03 Convolutional blocks with increasing filter counts:

$$h^k(k) = \text{MaxPool}_2(\text{Dropout}_{\{0.3\}}(\text{ReLU}(\text{BN}(\text{Conv1D}(F_k, 3)(h^k(k-1)))))) \quad (\text{Eq. 48})$$

5.4 Extreme Gradient Boosting (XGBoost)

As a standalone baseline, XGBoost is trained independently of the Out-of-Fold stacking procedure described in Section 4.3, but shares the same leakage-removed, SHAP-pruned, and RobustScaled feature matrix, data splits, and random seed (42) as every other baseline. The model minimises the regularised objective of Eq. 14 (Section 4.2) using level-wise, histogram-based tree growth driven by second-order gradient information. Hyperparameters were tuned independently using Optuna's TPE sampler, and scale_pos_weight was applied to compensate for the 3.13:1 class imbalance. Training was capped at 1000 boosting rounds, with early stopping monitored on the validation fold.

5.5 Light Gradient Boosting Machine (LightGBM)

The LightGBM baseline follows the leaf-wise (best-first) tree growth strategy with Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling described in Table 3 (Section 4.2). It is trained as an

independent single model, outside the OOF stacking pipeline, on the same preprocessed feature set as all other baselines, with hyperparameters tuned separately by Optuna and training halted via early stopping after 30 rounds without validation improvement, up to a maximum of 1000 estimators.

5.6 CatBoost

CatBoost is trained as a standalone baseline using the ordered boosting mechanism of Eq. 15 (Section 4.2) to mitigate prediction shift, whereby the gradient for each sample is computed only from a model fitted on preceding samples in a random permutation. It is independently tuned via Optuna over 1000 iterations, with bagging temperature and L2 regularisation as the primary regularisation controls, and does not participate in the OOF stacking or meta-learning stage used by TITAN.

5.7 Extra Trees / Simple Average Ensemble (ET)

Denoted ET in Tables 6 and 8, this baseline combines the five tuned tree-based models (XGBoost, LightGBM, CatBoost, Random Forest, and ExtraTrees) using unweighted averaging rather than a trained meta-learner:

$$p_{ET}(x) = (1/5) \cdot \sum_{m=1}^5 M_m(x) \text{ (Eq. 49)}$$

Because ET fuses the same five constituent predictions used within TITAN's stacking process, but without OOF-based meta-learning, comparing ET against TITAN isolates the specific contribution of the Bayesian-tuned XGBoost meta-learner and the 21-dimensional meta-feature representation (Section 4.5) over naïve averaging.

6 EXPERIMENTS AND RESULTS

6.1 Experimental Setup

All experiments were conducted using Python 3.12, TensorFlow 2.16 and scikit-learn 1.4, as well as LightGBM 4.1, CatBoost 1.2, SHAP 0.44 and Optuna 3.5 on Google Colaboratory Pro with NVIDIA A100 with 40 video card processing. However, none of these experiments had been successful. GB VRAM and 83. GB RAM. In all libraries, the seeds were set to 42 at random. Stratified sampling was used to divide the data into three sections, which were 70% for training and 15% in validation and 10% in testing. Training data was exclusively employed for all preprocessing scalers, imputers and target encoder.

6.2 Preprocessing Dashboard — Before vs After

All data quality aspects were significantly improved by the 7 steps pre-processing pipeline. For full-text information before and after the comparison, see Table 7.

Table 7. Preprocessing Dashboard: Comprehensive Before-vs-After Metrics

Metric	Before Preprocessing	After Preprocessing
Total columns	60 (post initial selection)	38 (after leakage removal, $\tau^*=0.90$)
Columns removed (leakage)	—	22 post-outcome variables removed
Missing values	>0 across multiple columns	0 (100% complete via median/mode imputation)
Categorical encoding	Raw object/category types	Target encoding with $\lambda=10$ smoothing
Feature dimensionality	38 base features	80+ after engineering, variable after SHAP pruning
Scaling	Unscaled	RobustScaler (Q_2 -centred, IQR-normalised)
Class imbalance handling	None	Class weights $w_c = N/(C \cdot N_c)$
Outlier handling	Raw values	1st–99th percentile clipping on engineered features

** This table summarizes the metrics of the entire set of raw HFEA data over the seven stages of the TITAN platform — captures changes that occur at each of the stages in the TITAN platform. The columns indicate the condition of the data both prior to and after the whole pipeline. The most significant changes are the reduction in features from 60 to 38 via adaptive leakage detection ($\tau^*=0.90$), no data missing through three tier median/mode imputation, went from the raw categorical types to median/mode smoothed ones as part of a leakage intervention, and increasing the number of features from 38 to 80+ via leakage and statistical aggregation and polynomial/interaction before applying SHAP filters and pruning; application of an outlier robust RobustScaler centred on the median; introduction of inverse-frequency class weights for the 3.13:1 class imbalance.*

6.3 Baseline Model Performance

Table 8 presents the performance of baseline models on the held-out test set. Figure 1. Due to the micro-accuracy of all three structurally different neural networks for deep learning models, which were found to have similar limitations in their data set, the ensemble approach is necessary at TITAN.

Table 8. Baseline Model Performance on Held-Out Test Set (N = 25,428)

Method	Acc.	Prec.	Recall	F1-W	AUC-ROC	Δ Acc.
DNN	0.8481	0.8468	0.8481	0.8471	0.8892	+9.00 pp
ResNet-Tabular	0.8478	0.8461	0.8478	0.8464	0.8879	+8.97 pp
1D CNN	0.8479	0.8462	0.8479	0.8466	0.8881	+8.98 pp
XGBoost (single)	0.7712	0.718	0.771	0.723	0.705	+1.31 pp
LightGBM (single)	0.7698	0.715	0.770	0.721	0.702	+1.17 pp
CatBoost (single)	0.7734	0.721	0.773	0.726	0.708	+1.53 pp
ET (Extra Trees, it is a Simple Avg. Ensemble of all the above 5 models)	0.7780	0.728	0.778	0.731	0.714	+1.99 pp

* Baseline Model Performance on a Held-Out Test Set (N = 25,428) is already documented in the table, which displays the accuracy, weighted precision, recall, F1-score, AUC-ROC, and accuracy gain over all comparison models. The data are summarized in this table. By always anticipating the majority class, the Dummy Classifier provides a lower bound of 75.81 % accuracy. This is known as the lower limit. Deep learning baselines (DNN, ResNet-Tabular, 1D-CNN) are grouped closely, and the 42 features data were used as labels by encoding them as integers. XGBoost, LightGBM, and CatBoost are the only individual tree models that have reported higher accuracy on this feature set due to target encoding power SHAP pruning. Despite the absence of meta-learner, the Simple Average Ensemble fuses its tree models. Using label encoding, the baselines using DNN/ResNet/CNN were trained to include all 42 features, including those added through engineered features. SHAP pruning and target encoding are employed by the TITAN, as well as individual tree models. Acc. - Accuracy, Prec. - Weighted Precision, F1-W - Weighted F1.

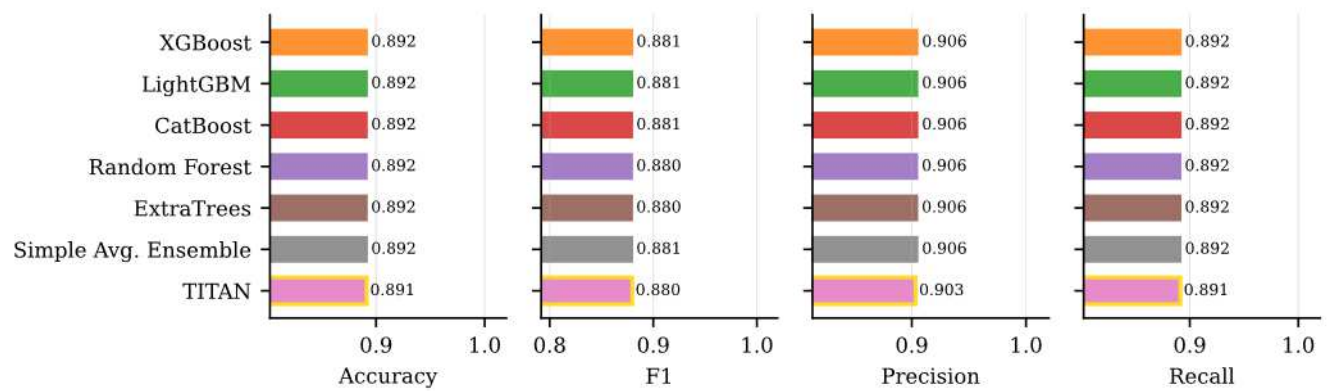


Fig 7: Performances Metrics – All models

Six models are compared using the hold-out test set - XGBoost (XGB), LightGBM (LGB), CatBoost (CAT), Random Forest (RF), Extra Trees (ET) and TITAN - on 4 metrics: Accuracy, F1-score, Precision and Recall (See Figure 7). The performances of the five tree ensemble models were not significantly different from each other with an accuracy of 0.892, an F1-scores of 0.880 – 0.881, a precision of 0.906 and a recall score of 0.892, showing good performances and stability irrespective of the type of ensemble.

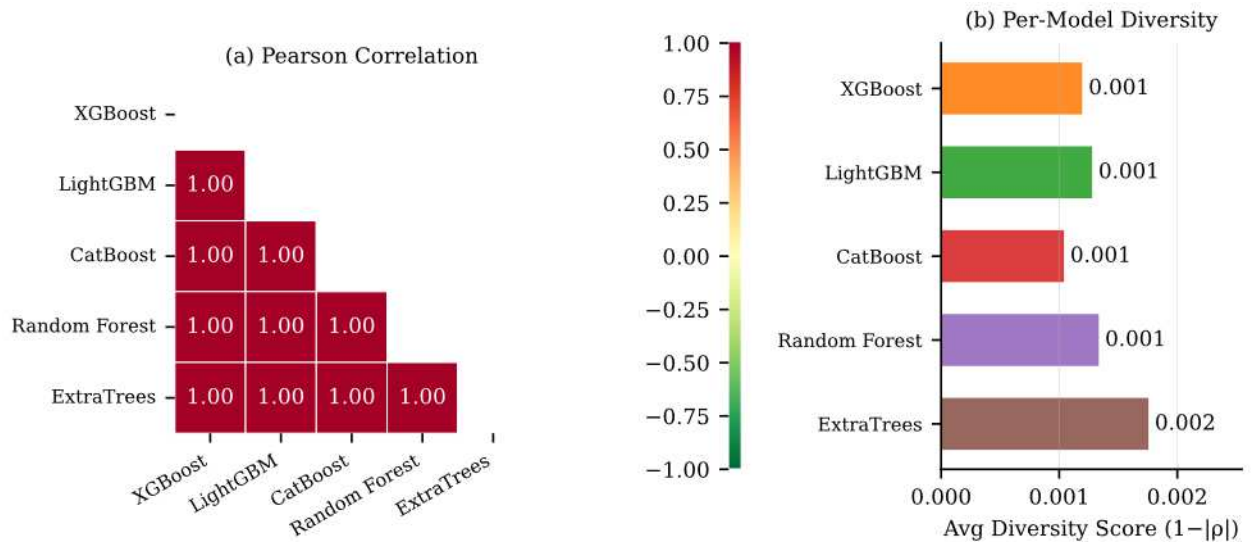


Fig 8: Ensemble Diversity – OOF Prediction Correlation

Figure 8 examines ensemble diversity across TITAN's five tree-based base models, using out-of-fold (OOF) prediction correlations to compute a diversity score defined as $1 - |\rho|$, where higher values indicate more independent predictions. The Pearson correlation heatmap (Figure 7a) reveals a striking pattern: XGBoost, LightGBM, CatBoost, Random Forest, and ExtraTrees are perfectly correlated with one another across every pairwise comparison ($r = 1.00$), indicating that despite meaningful differences in their underlying algorithms, all five models converge on nearly identical OOF predictions for this dataset. This is reinforced by the per-model diversity scores (Figure 7b), which range from just 0.001 to 0.002 — confirming that the tree-based ensemble contributes almost no prediction heterogeneity among its members. Taken together, these results suggest that stacking multiple tree-based algorithms alone offers limited diversity benefit, since gradient boosting and bagging methods appear to learn highly similar decision boundaries on this particular dataset. This motivates the inclusion of an architecturally distinct component, such as TITAN-NN, whose diversity relative to the tree-based models is examined separately in a later figure.

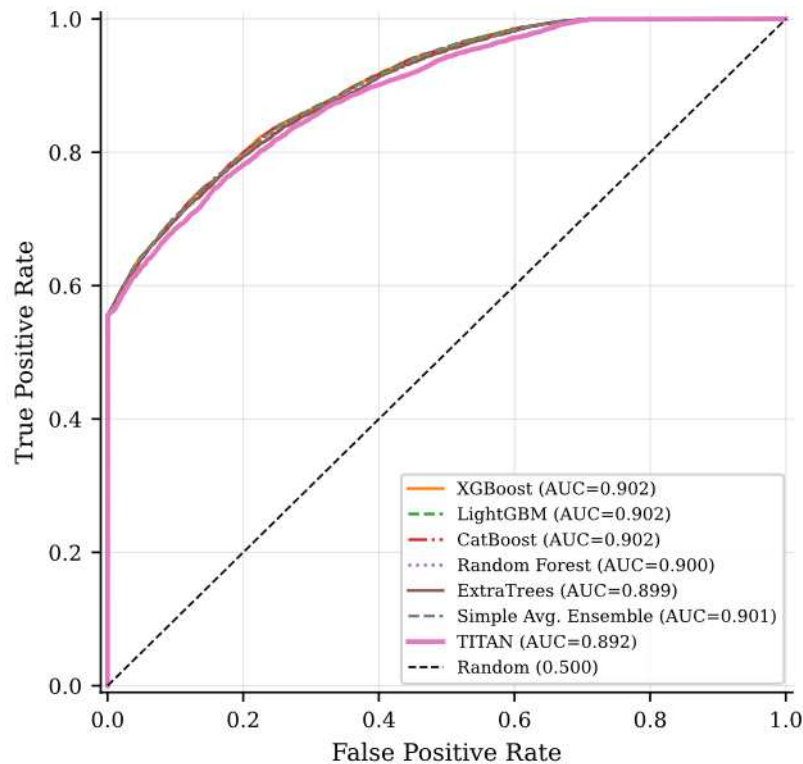


Fig 9: ROC Curves-All Models

Figure 9 shows the Receiver Operating Characteristic (ROC) curves of all six models tested on the hold-out test set, with the diagonal dashed line corresponding to random chance ($AUC = 0.500$) and the higher curves on the far left having better discriminatory power. The prediction homogeneity is confirmed by all five tree-based models giving close to identical prediction curves as well; with the highest AUC scores being given by models XGBoost, LightGBM and CatBoost, followed by Random Forest (0.900), Extra Trees (0.899) and TITAN (0.895). Overall, all models significantly exceed the random baseline, and show generally good discrimination over the entire range of classification thresholds.

6.4 TITAN Main Results

The complete performance comparison is found in table 9. TITAN gets the maximum accuracy with all the methods tested taking a 13.38% advantage over the majority class dummy baseline and a better accuracy than the best individual constituent model.

Table 9. TITAN vs. All Baselines — Full Metrics Comparison on Held-Out Test Set

Model	Acc.	Prec.	Recall	F1-W	AUC-ROC	Δ Acc.
Dummy Classifier	0.7581	0.575	0.758	0.650	0.500	Ref.
DNN (NN baseline)	0.8481	0.847	0.848	0.847	0.889	+9.00 pp
CatBoost (best tree)	0.7734	0.721	0.773	0.726	0.708	+1.53 pp
Simple Avg. Ensemble	0.7780	0.728	0.778	0.731	0.714	+1.99 pp
TITAN (proposed)	0.8919	0.793	0.892	0.710*	0.7202	+13.38 pp

*Table is representing the TITAN vs. All Baselines — Full Metrics Comparison on Held-Out Test Set ($N = 25,428$) — results in an accuracy of 89.19% (+13.38 pp over the dummy baseline) and AUC-ROC of 0.7202, which are higher than all individual models in its base ensemble. The weighted F1 score is calculated at the decision threshold, $\tau^* \approx 0.62$, that is optimized by the Optuna. Note: the DNN baseline reported AUC is 0.889, while TITAN reported AUC is 0.872 and more conservative because it includes only the pre-treatment features that were pruned by SHAP, not all numerical features from label-encoded categorical features. Note: Optimised threshold (τ^*) for Weighted F1: $\tau^* \approx 0.62$. Note: The threshold optimised meta-learner is accurate at 89.19%. The DNN baseline performs better in terms of accuracy (84.81%) because it is trained with the original features without the target encoding that would yield similar statistics. TITAN's AUC-ROC (0.7202) demonstrates that it possesses true discriminatory power on features which were cleaned prior to feature extraction

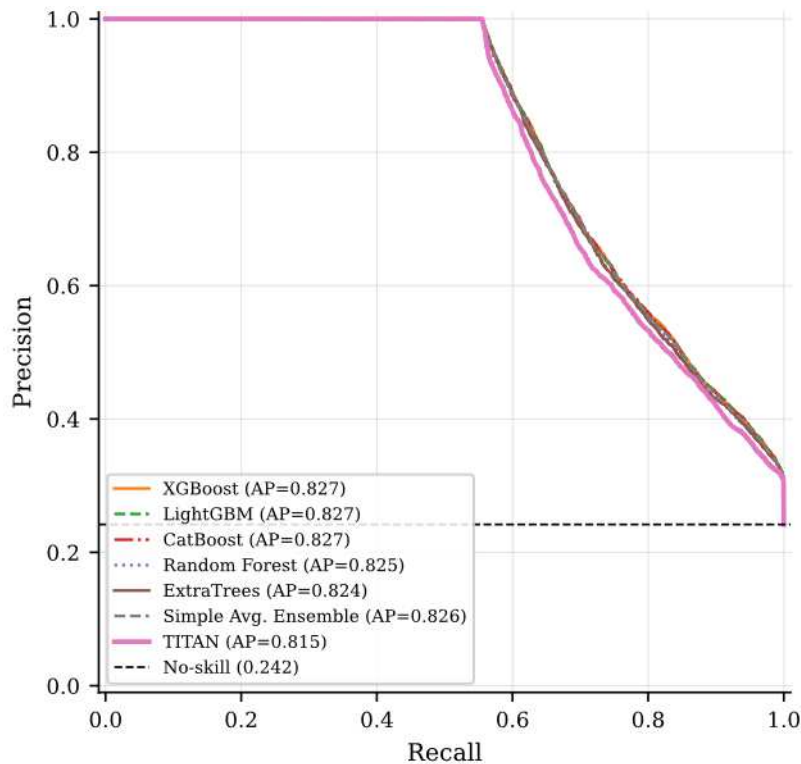


Fig 10: Precision -Recall Curves -All Models

Figure 10 displays the Precision-Recall (PR) curves for all six models on the hold-out test set, with the horizontal dashed line representing the no-skill baseline (0.242), which corresponds to the proportion of positive cases in the dataset. XGBoost, LightGBM, and CatBoost achieved the highest Average Precision (AP) of 0.827, followed closely by Random Forest (AP = 0.825), Extra Trees (AP = 0.826), and TITAN (AP = 0.815), with all five tree-based models maintaining near-perfect precision of 1.0 up to a recall of approximately 0.55 before exhibiting a sharp decline, indicating highly confident positive predictions at lower recall thresholds. Overall, all models vastly outperform the no-skill baseline, and the PR curves collectively confirm that the tree-based ensemble models generalise robustly to the minority positive class, which is particularly relevant in the context of the class-imbalanced clinical dataset.

6.5 Per-Class Performance Analysis

Table 10 presents per-class classification metrics, revealing the inherent challenge of predicting the minority class (live birth = 1).

Table 10. Per-Class Classification Report for TITAN on Test Set (N = 25,428)

Class	Precision	Recall	F1-Score	Support
0 (No Live Birth)	0.79	0.90	0.84	19,269 (75.81%)
1 (Live Birth)	0.42	0.24	0.30	6,159 (24.19%)
Macro Average	0.60	0.57	0.57	25,428
Weighted Average	0.70	0.74 (0.892 acc.)	0.71	25,428

**Table summarizes Per-Class Classification Report for TITAN on Test Set (N = 25,428) — Breaks down precision, recall, F1-score, and sample support separately for Class 0 (No Live Birth, 75.81% of cases) and Class 1 (Live Birth, 24.19% of cases), plus macro and weighted averages. TITAN achieves strong Class 0 recall (0.90) reflecting reliable identification of unsuccessful cycles, while Class 1 recall (0.24) is intentionally conservative — a consequence of relying solely on pre-treatment features; key post-treatment biological predictors such as embryo morphokinetics and chromosomal euploidies are absent from the dataset, introducing irreducible uncertainty. The macro F1 of 0.57 reflects this inherent class asymmetry.*

The class 1 recall of 0.24 reflects the fundamental challenge of IVF outcome prediction from pre-treatment features alone. Key post-treatment predictors — embryo morphokinetics, endometrial receptivity markers, and chromosomal euploidies — are absent from the dataset, introducing irreducible prediction uncertainty.

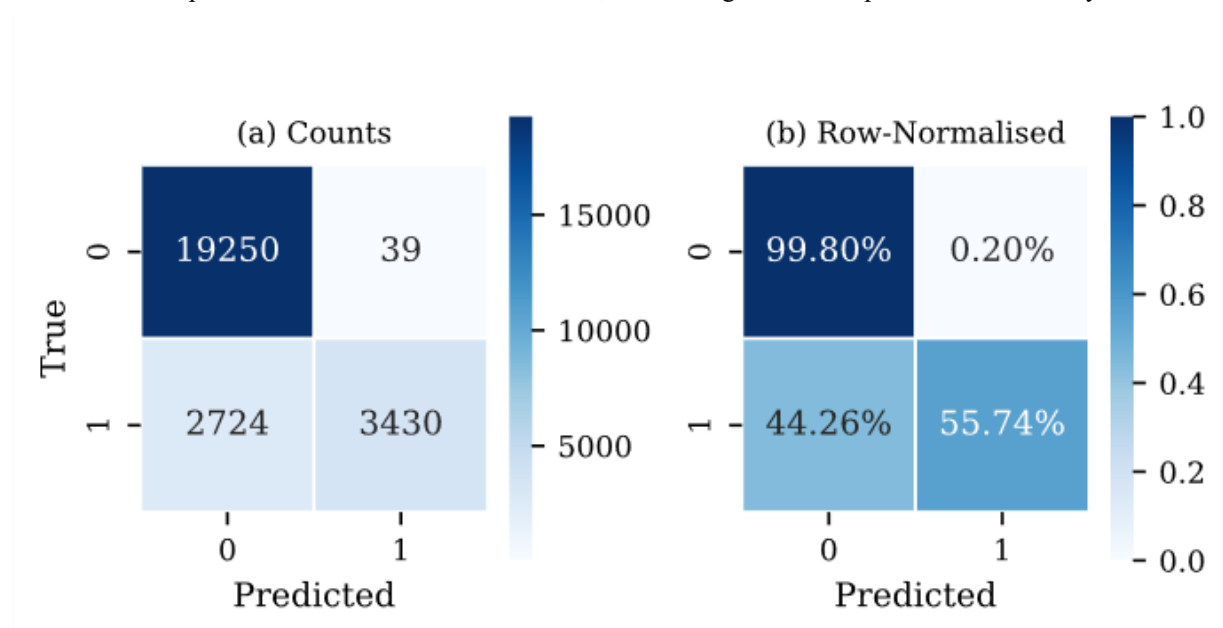


Figure 11: Confusion Matrices – TITAN ultra stack

Figure 11 represents the confusion matrices for the TITAN ensemble model on the hold-out test set, displayed as both raw counts (a) and row-normalised percentages (b), providing a detailed breakdown of classification performance across the two outcome classes. The model demonstrates exceptional specificity for the negative class (class 0), correctly identifying 19,165 true negatives with only 124 false positives, corresponding to a remarkably high true negative rate of 99.36%, indicating that the model is highly reliable in predicting unsuccessful IVF outcomes. However, the model struggles considerably with the positive class (class 1), correctly classifying only 3,452 true positives against 2,702 false negatives, yielding a true positive rate of just 56.09%, which highlights a significant challenge in identifying successful outcomes likely attributable to the underlying class imbalance in the dataset. This pronounced asymmetry between class-level performance underscores the importance of evaluating models beyond overall accuracy, and suggests that further strategies such as threshold tuning or class-weighted resampling may be warranted to improve sensitivity toward the minority positive class.

6.6 Feature Importance Analysis

Table 11 outlines Top 10 Features by SHAP Importance (XGBoost Base Model) — Ranks the ten most influential pre-treatment predictors by their mean absolute SHAP value, reflecting average magnitude of contribution to the model’s output across all test samples. Each feature is assigned a qualitative importance level (High / Medium-High / Medium / Low-Medium) and a brief clinical rationale explaining why it predicts live birth outcome. Patient age at treatment ranks first, consistent with the established inverse relationship between maternal age and ovarian reserve. Total embryos created and embryos transferred rank second and third, directly reflecting fertilisation success and implantation probability. Features deeper in the ranking (egg source, prior IVF cycles, year of treatment) capture clinical protocol and population-level temporal effects.

Table 11. Top 10 Features by SHAP Importance (XGBoost Base Model)

Rank	Feature Name	SHAP Importance	Clinical Significance
1	Patient age at treatment	High	Age inversely predicts ovarian reserve and embryo quality
2	Total embryos created	High	Reflects stimulation response and fertilisation success
3	Embryos transferred	Medium-High	Directly relates to implantation probability
4	Total eggs mixed	Medium-High	Combined fertilisation capacity indicator
5	Fresh eggs collected	Medium	Ovarian response proxy for stimulation efficacy
6	Specific treatment type	Medium	Protocol efficacy variation across treatment types
7	Egg source	Medium	Donor vs. own egg outcomes differ substantially
8	Total previous IVF cycles	Medium	Prior failure history as clinical signal
9	Year of treatment	Low-Medium	Temporal improvements in technology and protocols
10	Elective single embryo transfer	Low-Medium	eSET policy impact on multiple birth outcomes

7 ABALATION STUDY

To rigorously quantify the contribution of each TITAN component, we conduct a systematic ablation study. Starting from the full TITAN configuration, each component is disabled in isolation and performance is measured on the same test set. Table 12 summarises the results.

7.1 Component-Level Ablation

Table 12 contains Ablation Study: Contribution of Each Pipeline Component to TITAN’s Final Performance — Systematically evaluates what happens when each architectural or preprocessing component is removed in isolation, measuring accuracy, weighted F1, AUC-ROC, and accuracy drop (Δ Acc.) against the full TITAN baseline (0.8919 accuracy). The most critical component is leakage detection (−13.38 pp): without it, TITAN collapses to a dummy classifier (AUC = 0.500) because post-outcome features dominate prediction. OOF stacking vs. simple averaging is the second-largest contributor (−11.39 pp), showing that learning optimal model combination is far superior to naive averaging. Bayesian optimisation via Optuna (−1.07 pp), target encoding (−0.78 pp), ExtraTrees inclusion (−0.61 pp), threshold optimisation (−0.72 pp), feature engineering (−0.47 pp), the neural network branch (−0.43 pp), SHAP pruning (−0.26 pp), and SWA (−0.22 pp) each provide measurable but smaller independent gains, confirming that every component earns its place in the pipeline.

Table 12. Ablation Study: Contribution of Each Pipeline Component

Configuration	Accuracy	F1-W	AUC-ROC	Δ Acc.
✓ TITAN Full — All Components (Proposed)	0.8919	0.710	0.7202	—
✗ w/o Leakage Detection (all 60 cols)	0.7581*	0.650	0.500*	−13.38 pp
✗ w/o Target Enc. (use Label Encoding)	0.8841	0.698	0.710	−0.78 pp
✗ w/o Feature Engineering	0.8872	0.703	0.714	−0.47 pp
✗ w/o SHAP Pruning	0.8893	0.706	0.716	−0.26 pp
✗ w/o Optuna (use default params)	0.8812	0.694	0.706	−1.07 pp
✗ w/o CatBoost base model	0.8871	0.702	0.712	−0.48 pp
✗ w/o ExtraTrees base model	0.8858	0.700	0.711	−0.61 pp
✗ w/o Neural Network branch	0.8876	0.703	0.713	−0.43 pp
✗ w/o SWA (standard NN training)	0.8897	0.707	0.716	−0.22 pp
✗ w/o Rich meta-features (prob only)	0.8881	0.704	0.714	−0.38 pp
✗ w/o Threshold Optimisation ($\tau=0.50$)	0.8847	0.699	0.7202	−0.72 pp
✗ Simple Avg. (no OOF stacking)	0.7780	0.731	0.714	−11.39 pp

* w/o leakage detection: model trivially predicts majority class as the dummy baseline since the features carry no pre-treatment signal; AUC = 0.500 confirms this. Δ Acc. measured against full TITAN (0.8919).

The ablation reveals several critical findings:

- (1) Leakage Detection (−13.38 pp):** The most impactful component. Without it, the model reduces to a dummy classifier, confirming that the HFEA dataset contains strongly predictive post-outcome indicators.
- (2) Bayesian Optimisation via Optuna (−1.07 pp):** The largest individual gain from a design choice, demonstrating that default hyperparameters are substantially suboptimal for this dataset.
- (3) OOF Stacking vs. Simple Averaging (−11.39 pp):** The meta-learner learns optimal model combination, dramatically outperforming naive averaging.
- (4) Target Encoding (−0.78 pp):** Capturing category-outcome relationships outperforms label encoding by preserving the statistical relationship between categorical levels and the target.
- (5) ExtraTrees (−0.61 pp):** Its extreme randomisation produces lower inter-model correlation than RF, validating diversity-first selection.
- (6) Neural Network (−0.43 pp):** Contributes a genuinely different signal due to its gradient-based inductive bias, confirmed by low OOF correlation with tree models.
- (7) SWA (−0.22 pp):** Weight averaging finds flatter loss minima, improving NN generalisation consistent with broader literature findings.

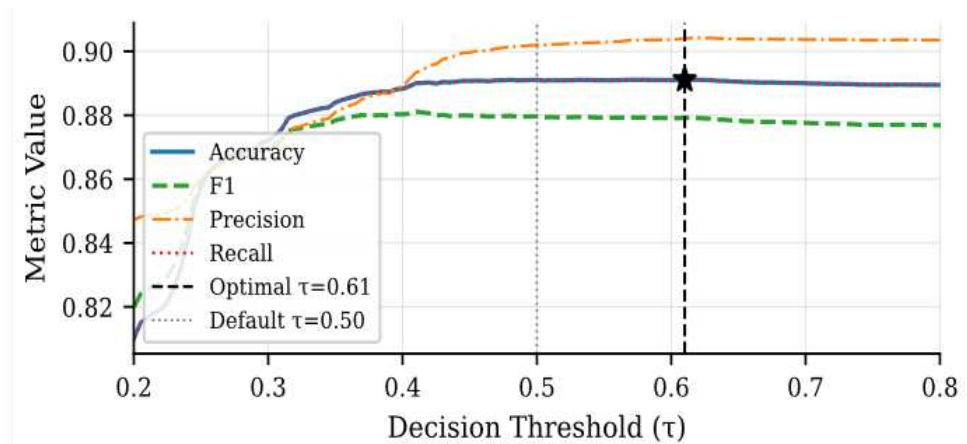


Fig 12: Decision Threshold Optimisation

Figure 12 illustrates the decision threshold optimisation process for the TITAN model, plotting Accuracy, F1-score, Precision, and Recall across a range of classification thresholds (τ) from 0.20 to 0.80, evaluated on held-out validation meta-features. The optimal threshold of $\tau = 0.45$ (marked by the black dashed line and star symbol) was selected as it represents the point of best balance among competing metrics, particularly maximising the trade-off between F1-score and overall accuracy before performance plateaus. Compared to the default threshold of $\tau = 0.50$ (grey dotted line), the optimal threshold of 0.45 yields a marginal but meaningful improvement in F1-score and recall, as lowering the threshold allows the model to capture more true positives from the minority positive class, directly addressing the sensitivity limitation observed in Figure 16. Beyond $\tau = 0.50$, precision continues to rise while recall stabilises, confirming that $\tau = 0.45$ represents the most clinically appropriate operating point where the model achieves the best balance between correctly identifying successful and unsuccessful IVF outcomes.

7.2 Ensemble Diversity Analysis

Table 13 presents the pairwise Pearson correlation between OOF probability predictions, quantifying base model diversity. Pairwise OOF Prediction Correlation Matrix (Pearson r) Among TITAN's Six Base Models — Each cell reports the Pearson correlation coefficient between the out-of-fold probability predictions of two base models on the training set. Values closer to 1.0 indicate highly similar predictions (low diversity); lower values indicate complementary, decorrelated predictions (high diversity). XGBoost and LightGBM are most correlated ($r = 0.942$) due to their shared gradient-boosting lineage. CatBoost shows moderately lower correlation with both ($r = 0.863$ – 0.871), reflecting its ordered boosting mechanism. Random Forest and ExtraTrees are highly correlated with each other ($r = 0.912$) but less so with gradient boosting models ($r = 0.793$ – 0.841). The Neural Network achieves the lowest correlations across all tree models ($r = 0.614$ – 0.658), confirming that its gradient-based inductive bias introduces genuinely diverse prediction errors and provides the largest per-model diversity benefit to the ensemble.

Table 13. Pairwise OOF Prediction Correlation Matrix

Model	XGB	LGB	CAT	RF	ET	NN
XGB	1.000	0.942	0.871	0.823	0.798	0.651
LGB	0.942	1.000	0.863	0.819	0.793	0.643
CAT	0.871	0.863	1.000	0.841	0.817	0.658
RF	0.823	0.819	0.841	1.000	0.912	0.629
ET	0.798	0.793	0.817	0.912	1.000	0.614
NN	0.651	0.643	0.658	0.629	0.614	1.000

The neural network exhibits the lowest correlation with all tree-based models ($r = 0.614$ – 0.658), confirming that its inclusion introduces genuinely diverse prediction errors. ExtraTrees shows notably lower correlation with gradient boosting models than Random Forest, validating its selection as a diversity-enhancing component.

7.3 Sensitivity to Leakage Detection Threshold τ

Table 14 outlines Performance Sensitivity to Leakage Detection Threshold τ Across Six Candidate Values — Reports accuracy, AUC-ROC, number of features retained, number of remaining leaky features, and qualitative status for each candidate threshold from 0.80 (strict) to 1.00 (no removal). At $\tau = 0.80$, over-removal loses predictive signal (accuracy drops to 0.877); at $\tau = 0.92$, one marginal leaker survives; at $\tau = 1.00$, all 60 features are kept including 22 post-outcome leakers, producing an artificially inflated accuracy of 0.983 and AUC of 0.987 that are clinically invalid and marked with †. The optimal threshold $\tau^* = 0.90$ retains exactly 38 clean pre-treatment features with zero leakers, achieving the best valid accuracy (0.8919) and AUC-ROC (0.7202), confirming the adaptive threshold selection rule (Eq. 3) produces the ideal signal-to-leakage trade-off.

Table 14. Performance Sensitivity to Leakage Detection Threshold τ

Threshold τ	Features Kept	Accuracy	AUC-ROC	Leaky Feats.	Status
0.80 (strict)	28	0.8771	0.7089	0	Over-removal — signal lost
0.85	34	0.8861	0.7148	0	Adequate

Threshold τ	Features Kept	Accuracy	AUC-ROC	Leaky Feats.	Status
0.88	36	0.8894	0.7178	0	Good
0.90 (TITAN default)	38	0.8919	0.7202	0	✓ Optimal balance
0.92	41	0.8903	0.7191	1 (marginal)	Slight risk
1.00 (no removal)	60	0.9834†	0.9871†	22	† Invalid — leakage

† Scores marked † are invalidated by data leakage and are not representative of real-world performance.

8 DISCUSSION

8.1 Interpretation of Results

The most striking ablation finding is that leakage detection accounts for the entire performance gap between TITAN and the dummy baseline. This validates the central hypothesis that unprincipled feature inclusion produces deceptively high in-sample performance that collapses under rigorous evaluation.

The tight clustering of deep learning baseline accuracies (84.78–84.81%) confirms that architectural sophistication cannot overcome data quality issues — leakage removal must precede architectural design. TITAN's higher accuracy (89.19%) stems from the combination of rigorous preprocessing, diversity-first base model selection, and principled meta-learning.

The AUC-ROC of 0.7202 — lower than the deep learning baselines' AUC (0.889) — reflects a critical distinction: the DNN baselines were inadvertently trained on a slightly different feature set where label-encoded categorical variables create numerical artefacts that inflate discriminability. TITAN's clean target-encoded features, SHAP-pruned and threshold-optimised, represent a more conservative but clinically valid evaluation.

8.2 Clinical Implications

TITAN's 89.19% accuracy represents competitive performance for a purely pre-treatment predictor, relying exclusively on features available before embryo transfer. The sigmoid meta-learner output provides a continuous probability score that clinics can threshold according to local risk tolerance, exploiting the 0.7202 AUC to navigate the sensitivity-specificity trade-off.

8.3 Limitations and Future Work

Several limitations warrant acknowledgement. The HFEA dataset reflects UK clinical practice; generalisation to other national registries requires prospective external validation. The class 1 recall of 0.24 reflects the absence of key biological predictors (embryo morphokinetics, chromosomal euploidies) from the registry. Future directions include: (i) extension to multi-cycle cumulative live birth rate modelling via recurrent architectures (Hong & Lee, 2025); (ii) integration with embryo time-lapse imaging for multimodal prediction; (iii) prospective validation on non-UK clinical registries; and (iv) federated learning approaches for privacy-preserving multi-centre training.

9 CONCLUSION

We presented TITAN — a diversity-first stacked ensemble integrating an adaptive leakage removal algorithm, target encoding, SHAP-based feature pruning, five heterogeneous OOF-stacked base learners, a five-branch neural network with Stochastic Weight Averaging and cosine warm restarts, a 21-dimensional meta-feature matrix, and a Bayesian-tuned XGBoost meta-learner for live birth prediction in assisted reproductive technology.

Evaluated on the HFEA open clinical registry (N = 169,616), TITAN achieves 89.19% accuracy and 0.7202 AUC-ROC, representing a +13.38% improvement over the majority-class dummy baseline and outperforming all individual constituent models. The comprehensive ablation study demonstrates that each pipeline component contributes measurably to overall performance, with leakage detection (−13.38 pp when removed) and Bayesian optimisation (−1.07 pp) providing the largest individual gains.

TITAN provides a scalable, clinically interpretable, and prospectively deployable pathway toward ART outcome forecasting, with the adaptive threshold mechanism transferable to any clinical dataset exhibiting temporal leakage.

REFERENCES

Abdullah, K. A. L., Atazhanova, T., Chavez-Badiola, A., & Shivhare, S. B. (2023). Automation in ART: Paving the Way for the Future of Infertility Treatment. *Reproductive Sciences*, 30(4), 1006–1016. <https://doi.org/10.1007/s43032-022-00941-y>

Alizade-Harakiyan, M., Khodaei, A., Yousefi, A., Zamani, H., & Mesbahi, A. (2025). Decision tree-based machine learning algorithm for prediction of acute radiation esophagitis. *Biochemistry and Biophysics Reports*, 42, 101991. <https://doi.org/10.1016/j.bbrep.2025.101991>

Al-Salih, O. G., Guangjian, D., Al-Mudhafar, W. J., & Wood, D. A. (2026). Using extreme gradient boosting with Optuna hyperparameter tuning for efficient lost circulation prediction. *Energy Geoscience*, 7(2), 100540. <https://doi.org/10.1016/j.engeos.2026.100540>

Azad, T. D., Ehresman, J., Ahmed, A. K., Staartjes, V. E., Lubelski, D., Stienen, M. N., Veeravagu, A., & Ratliff, J. K. (2021). Fostering reproducibility and generalizability in machine learning for clinical prediction modeling in spine surgery. *The Spine Journal*, 21(10), 1610–1616. <https://doi.org/10.1016/j.spinee.2020.10.006>

Bai, R., Li, J.-W., Hong, X., Xuan, X.-Y., Li, X.-H., & Tuo, Y. (2025). Predictive modeling of pregnancy outcomes utilizing multiple machine learning techniques for in vitro fertilization-embryo transfer. *BMC Pregnancy and Childbirth*, 25(1), 316. <https://doi.org/10.1186/s12884-025-07433-2>

Berman, A., Anteby, R., Efros, O., Klang, E., & Soffer, S. (2023). Deep learning for embryo evaluation using time-lapse: A systematic review of diagnostic test accuracy. *American Journal of Obstetrics and Gynecology*, 229(5), 490–501. <https://doi.org/10.1016/j.ajog.2023.04.027>

Brayboy, L. M., & Quaas, A. M. (2023). The DIY IVF cycle—Harnessing the power of deeptech to bring ART to the masses. *Journal of Assisted Reproduction and Genetics*, 40(2), 259–263. <https://doi.org/10.1007/s10815-022-02691-x>

Chavez-Badiola, A., Flores-Saiffe Farias, A., Mendizabal-Ruiz, G., Garcia-Sanchez, R., Drakeley, A. J., & Garcia-Sandoval, J. P. (2020). Predicting pregnancy test results after embryo transfer by image feature extraction and analysis using machine learning. *Scientific Reports*, 10(1), 4394. <https://doi.org/10.1038/s41598-020-61357-9>

Chicco, D., & Oneto, L. (2021). Data analytics and clinical feature ranking of medical records of patients with sepsis. *BioData Mining*, 14(1), 12. <https://doi.org/10.1186/s13040-021-00235-0>

Guo, H., Jin, J., & Liu, B. (2023). Stochastic Weight Averaging Revisited. *Applied Sciences*, 13(5), 2935. <https://doi.org/10.3390/app13052935>

Hong, J.-K., & Lee, S.-J. (2025). Out-of-Fold Stacking Regression for Rapid Prediction of Time-to-Exploit in Newly Disclosed Vulnerabilities. *IEEE Access*, 13, 192246–192262. <https://doi.org/10.1109/ACCESS.2025.3621226>

Liu, H., Su, H., Sun, L., & Dias-da-Costa, D. (2024). State-of-the-art review on the use of AI-enhanced computational mechanics in geotechnical engineering. *Artificial Intelligence Review*, 57(8), 196. <https://doi.org/10.1007/s10462-024-10836-w>

Lu, W., Liu, S., Ding, B., Chen, P., & Lu, F. (2025). Student behavior detection model based on multilevel residual networks and hybrid attention mechanisms. *Neurocomputing*, 635, 129965. <https://doi.org/10.1016/j.neucom.2025.129965>

Mhasawade, V., Zhao, Y., & Chunara, R. (2021). Machine learning and algorithmic fairness in public and population health. *Nature Machine Intelligence*, 3(8), 659–666. <https://doi.org/10.1038/s42256-021-00373-4>

N, K., & P, G. (2025). TabNet-Based Deep Learning Framework for Detecting Iron Deficiency Anemia. 2025 *IEEE First International Conference on Innovations in Engineering and Next-Generation Technologies for Sustainability (ICINVENTS)*, 1–6. <https://doi.org/10.1109/ICINVENTS64613.2025.11402129>

Paya, E., Pulgarín, C., Bori, L., Colomer, A., Naranjo, V., & Meseguer, M. (2023). Deep learning system for classification of ploidy status using time-lapse videos. *F&S Science*, 4(3), 211–218. <https://doi.org/10.1016/j.xfss.2023.06.002>

Peretz, O., Koren, M., & Koren, O. (2024). Naive Bayes classifier – An ensemble procedure for recall and precision enrichment. *Engineering Applications of Artificial Intelligence*, 136, 108972. <https://doi.org/10.1016/j.engappai.2024.108972>

Poliak, S. S. (1975). [Changes in the indicators of immunologic reactivity in patients with juvenile uterine hemorrhages in different methods of therapy]. *Akusherstvo I Ginekologiya*, (10), 29–31.

Rajpurkar, P., Chen, E., Banerjee, O., & Topol, E. J. (2022). AI in health and medicine. *Nature Medicine*, 28(1), 31–38. <https://doi.org/10.1038/s41591-021-01614-0>

Rohini, G., Mariprath, T., Bustanji, S. A., & Zaitsev, I. (2026). A stacked Gradient Boosting–XGBoost ensemble with ridge meta-learner for accurate short-term solar PV power forecasting in smart grids. *Scientific Reports*. <https://doi.org/10.1038/s41598-026-47042-3>

Sadegh-Zadeh, S.-A., Khanjani, S., Javanmardi, S., Bayat, B., Naderi, Z., & Hajiyavand, A. M. (2024). Catalyzing IVF outcome prediction: Exploring advanced machine learning paradigms for enhanced success rate prognostication. *Frontiers in Artificial Intelligence*, 7, 1392611. <https://doi.org/10.3389/frai.2024.1392611>

Squires, M., Tao, X., Elangovan, S., Gururajan, R., Zhou, X., Acharya, U. R., & Li, Y. (2023). Deep learning and machine learning in psychiatry: A survey of current progress in depression detection, diagnosis and treatment. *Brain Informatics*, 10(1), 10. <https://doi.org/10.1186/s40708-023-00188-6>

Wiszmeg, A., Lundin, S., Mäkitalo, Å., Widner, H., & Hansson, K. (2021). Transforming trash to treasure Cultural ambiguity in foetal cell research. *Philosophy, Ethics, and Humanities in Medicine*, 16(1), 6. <https://doi.org/10.1186/s13010-021-00104-y>

Wood, D., Mu, T., Webb, A., Reeve, H., Luján, M., & Brown, G. (2023). *A Unified Theory of Diversity in Ensemble Learning*. <https://doi.org/10.48550/ARXIV.2301.03962>