In this report, I detail the development and performance of an ensemble model designed to address a binary classification problem set by a recent competition. The challenge was to predict the presence or absence of any of three specific medical conditions in individuals, based solely on over fifty anonymized health characteristics. This task aims to streamline the diagnostic process, preserving patient privacy while accurately assessing health risks.

The ensemble comprises seven advanced tree-based models: one CatBoost, three XGBoosts each with unique parameters, one LightGBM, and two TabPFNs (Tabular Parameterized Feedforward Networks), also with distinct parameters. These models were meticulously calibrated to minimize balanced log loss, the competition's primary evaluation metric, achieving a noteworthy score of 0.39256 on the final hidden dataset. Efforts to mitigate training data class imbalance included sample weighting and oversampling techniques.

Data Preprocessing Overview: The preparation of my dataset was a critical step in ensuring the effectiveness of the ensemble model. This process involved several key operations to manage missing values, scale numerical features, and transform the data to better suit my modeling needs. Specifically, I:

- Merged additional data from "greeks.csv", focusing on the 'Epsilon' variable, to enrich our dataset.
- Applied logarithmic transformations to most numerical columns to normalize their distributions, a strategy particularly effective for handling skewed data.
- Created a new feature to flag missing values in the 'BQ' column, an innovative approach to leverage missingness as a predictive signal.

Cross-Validation in Tuning and Training: A cross-validation strategy was employed during both the hyperparameter optimization and model training phases. Using Optuna for hyperparameter tuning, I leveraged cross-validation to ensure the selected parameters would perform consistently well across different subsets of the data.

Hyperparameter optimization was conducted using Optuna, focusing on selecting the most impactful features. Analysis revealed that optimal performance was attained using a subset of features, specifically with the third XGBoost model and LightGBM. The features to use in the models with dimensionality reduction were chosen by considering their impact on the model’s balanced log loss score. The code for these procedures is organized across three files: Optuna_Main.py for tuning, Send_Main.py for model training and submission file generation, and icr_lib.py, which houses shared functions.

The training dataset featured 617 entries, some with missing values, notably in columns labeled 'EL' and 'BQ'. Initial experiments with simple linear regression proved inadequate, prompting exploration of oversampling and undersampling strategies to address the pronounced class imbalance; the dataset was predominantly composed of class 0 (no condition) instances.

Despite considering more complex methods like SMOTE (Synthetic Minority Over-sampling Technique) and cost-sensitive training, I opted against these due to the advanced models' inherent capabilities and the potential for overfitting. Nonetheless, selective oversampling of the ensemble's components yielded slight improvements in public test set performance.

Below are the best parameters found during the Optuna study followed by explanation of what do they mean.

params = {'dimensionality_reduction_model1_index': 4, 'dimensionality_reduction_model2_index': 3,

'catboost_iterations': 119, 'catboost_learning_rate': 0.2373478627191737, 'catboost_depth': 4,

'catboost_l2_leaf_reg': 1.3063393283179154, 'xgboost_n_estimators': 165, 'xgboost_max_depth': 3,

'xgboost_subsample': 0.8119794123653089, 'xgboost_colsample_bytree': 0.9438318190982824, 'lgbm_max_depth': 5, 'lgbm_learning_rate': 0.2785788274064489, 'lgbm_num_leaves': 120, 'lgbm_colsample_bytree': 0.9729778631903917, 'xgboost2_n_estimators': 191, 'xgboost2_max_depth': 7, 'xgboost2_subsample': 0.7330530954570363, 'xgboost2_colsample_bytree': 0.6394804662463566, 'xgboost3_n_estimators': 119, 'xgboost3_max_depth': 8, 'xgboost3_subsample': 0.9671996924315035, 'xgboost3_colsample_bytree': 0.6744071411442031, 'AB': True, 'AF': False, 'AH': True, 'AM': False, 'AR': False, 'AX': False, 'AY': False, 'AZ': False, 'BC': True, 'BD ': False, 'BN': False, 'BP': False, 'BQ': True, 'BR': False, 'BZ': False, 'CB': False, 'CC': True, 'CD ': False, 'CF': True, 'CH': True, 'CL': False, 'CR': False, 'CS': False, 'CU': False, 'CW ': False, 'DA': False, 'DE': False, 'DF': False, 'DH': False, 'DI': False, 'DL': True, 'DN': True, 'DU': True, 'DV': True, 'DY': True, 'EB': True, 'EE': True, 'EG': True, 'EH': True, 'EJ': False, 'EL': True, 'EP': True, 'EU': True, 'FC': False, 'FD ': True, 'FE': False, 'FI': False, 'FL': False, 'FR': False, 'FS': True, 'GB': True, 'GE': True, 'GF': False, 'GH': False, 'GI': True, 'GL': True, 'Epsilon': False, 'BQ_is_nan': False, 'oversampling_catboost': False, 'oversampling_xgboost': True, 'oversampling_xgboost2': False, 'oversampling_xgboost3': True, 'oversampling_lgbm': False}

1. **dimensionality_reduction_model1_index and dimensionality_reduction_model1_index-** I have decided to choose (with Optuna) the two models out of five on which I am performing the dimensionality reduction. These models get only a subset of columns (both models the same subset). The subset of columns to use was also determined during the Optuna study. The best performance was found with choosing the xgboost3 and lgbm as the models on which the reduction is performed.

2. '**AB**': True, '**AF**': False, '**AH**': True, '**AM**': False, '**AR**': False, '**AX**': False, '**AY**': False, '**AZ**': False, '**BC**': True, '**BD** ': False, '**BN**': False, '**BP**': False, '**BQ**': True, '**BR**': False, '**BZ**': False, '**CB**': False, '**CC**': True, '**CD** ': False, '**CF**': True, '**CH**': True, '**CL**': False, '**CR**': False, '**CS**': False, '**CU**': False, '**CW** ': False, '**DA**': False, '**DE**': False, '**DF**': False, '**DH**': False, '**DI**': False, '**DL**': True, '**DN**': True, '**DU**': True, '**DV**': True, '**DY**': True, '**EB**': True, '**EE**': True, '**EG**': True, '**EH**': True, '**EJ**': False, '**EL**': True, '**EP**': True, '**EU**': True, '**FC**': False, '**FD** ': True, '**FE**': False, '**FI**': False, '**FL**': False, '**FR**': False, '**FS**': True, '**GB**': True, '**GE**': True, '**GF**': False, '**GH**': False, '**GI**': True, '**GL**': True

   -This set of flags indicate which columns were used and which weren't in the two sub-models on which the dimensionality reduction was performed.

3. **'Epsilon'**: False- The flag says wether the extra Epsilon column which contains the date when the sample was obtained was used or not.

4. '**BQ_is_nan'**: False- Says wether the BQ_is_nan column created during the data preprocessing was used or not.

5. '**oversampling_catboost'**: False, '**oversampling_xgboost'**: True, **'oversampling_xgboost2**': False, '**oversampling_xgboost3'**: True, **'oversampling_lgbm**': False

   - This set of flags says on which models the oversampling was performed.

6. **catboost_iterations**- determines the number of trees built as part of the catboost sub-classifier.

7. **catboost_depth**- determines the depth of the trees being built as part of the catboost sub-classifier.

8. **catboost_l2_leaf_reg**- coefficient for the L2 regularization term in the cost function of the catboost sub-model. This parameter helps in controlling overfitting by penalizing complex models. A higher value of l2_leaf_reg results in stronger regularization, leading to simpler models which can generalize better on unseen data. The allowed values for this parameter are any positive numbers, with a default value of 3.0.

9. **xgboost_n_estimators**- specifies the number of boosting rounds, essentially the count of sequential models or trees to be built for the sxboost sub-model. It's crucial for model performance, with higher values potentially improving training accuracy but increasing the risk of overfitting.
   xgboost2_n_estimators- the same, but on the second xgboost classifier.

10. **xgboost_max_depth**- is used to control the maximum depth of the decision trees that are created as part of the xgboost sub-model.
    xgboost2_max_depth, xgboost3_max_depth- the same, but on the second and third xgboost classifiers.

11. **xgboost_subsample**- refers to the fraction of the training data to be randomly sampled for building each tree. It's a way to control overfitting. It takes a value between 0 and 1. A value of 1 means that the algorithm will use all training data to build each tree. Using a subset of the data for building each tree, it prevents the model from learning too much from the training data.
    xgboost2_subsample, xgboost3_subsample- the same, but on the second and third xgboost classifiers.

12. **xgboost_colsample_bytree**- specifies the fraction of features (columns) to be randomly sampled for each tree in the xgboost sub-model before growing it. This is part of the model's effort to prevent overfitting and add more randomness to the model-building process. By selecting a subset of all features for each tree, it ensures that the trees in the model are not all using the same top features and thus can capture more diverse patterns in the data.
    xgboost2_colsample_bytree, xgboost3_colsample_bytree- the same, but on the second and third xgboost classifiers.

13. **lgbm_max_depth**- is used to control the maximum depth of the decision trees that are created as part of the lgbm sub-model.

14. **lgbm_learning_rate**- learning rate of the lgbm sub-model.

15. **lgbm_num_leaves**- specifies the maximum number of leaves (terminal nodes) that can be formed in any tree of the lgbm sub-model. It defines how detailed the model can be in terms of making splits to accommodate the nuances in the data.

16. **lgbm_colsample_bytree**- specifies the fraction of features (columns) to be randomly sampled for each tree in the lgbm sub-model before growing it.