## Hackathon Project Documentation

### Introduction

In this project, the primary goal was to develop a machine learning model to predict the presence of HIV based on various demographic and medical features. This document details the methodology, variables used, model selection, hyperparameter tuning, and evaluation metrics. The target variable (hiv) is a binary integer, indicating whether an individual is HIV positive (1) or not (0). The independent variables include demographic features such as gender, region, race, age, various diagnosis codes, procedure codes, and National Drug Codes (NDCs).

### Dependent Variable

The dependent variable in this study is:

* **hiv**: A binary integer variable indicating the presence (1) or absence (0) of HIV.

### Independent Variables

The independent variables used in this project include a range of demographic and medical features:

#### Demographic Variables:

* + gender\_F, gender\_None, gender\_M, gender\_U, gender\_: Gender indicators.
  + region\_Midwest, region\_Other, region\_South, region\_West, region\_Northeast: Geographic region indicators.
  + race\_None, race\_Other, race\_White, race\_Black, race\_Hispanic, race\_Asian: Race indicators.
  + age: Age of the individual (decimal).

#### Medical Diagnosis Codes:

* + new\_diagnosis\_code\_K219, new\_diagnosis\_code\_E559, new\_diagnosis\_code\_Z79899, new\_diagnosis\_code\_E119, new\_diagnosis\_code\_Z20822, new\_diagnosis\_code\_Z0000, new\_diagnosis\_code\_E785, new\_diagnosis\_code\_F419, new\_diagnosis\_code\_I10, new\_diagnosis\_code\_Z23: Indicators for various medical diagnoses.
  + Combined Code: new\_diagnosis\_code\_others: Includes all other diagnosis codes not in the top 10, based on percentage prevalence.

#### Procedure Codes:

* + new\_procedure\_code\_99213, new\_procedure\_code\_80061, new\_procedure\_code\_36415, new\_procedure\_code\_99214, new\_procedure\_code\_83036, new\_procedure\_code\_99284, new\_procedure\_code\_80053, new\_procedure\_code\_99203, new\_procedure\_code\_85025, new\_procedure\_code\_84443: Indicators for various medical procedures.
  + Combined Code: new\_procedure\_code\_others: Includes all other procedure codes not in the top 10, based on percentage prevalence

#### NDC Codes (National Drug Codes):

* + new\_ndc\_code\_59310057922, new\_ndc\_code\_00054327099, new\_ndc\_code\_57237000511, new\_ndc\_code\_68180012202, new\_ndc\_code\_69452015120, new\_ndc\_code\_00781261305, new\_ndc\_code\_65862042005, new\_ndc\_code\_66993001968, new\_ndc\_code\_00173068220, new\_ndc\_code\_60505082901: Indicators for various prescribed drugs.
  + Combined Code: new\_ndc\_code\_others: Includes all other NDC codes not in the top 10, based on percentage prevalence.

Note: Before creating the new\_diagnosis\_code\_others, new\_procedure\_code\_others, and new\_ndc\_code\_others variables, the percentage prevalence of each code was carefully analysed. Only the top 10 most prevalent codes were kept as individual variables, while all other less frequent codes were combined into the respective "others" categories. This approach helped reduce the dimensionality of the dataset while preserving the most informative features.

### D. Methodology

#### 1. Data Preparation

The dataset was pre-processed to ensure that all categorical variables were appropriately encoded using One – hot encoding, and missing values were handled. The independent variables were separated from the dependent variable.

#### 2. Addressing Class Imbalance with SMOTE

The original dataset was imbalanced, with a significantly higher number of non-HIV cases compared to HIV cases. To address this imbalance, **Synthetic Minority Over-sampling Technique (SMOTE)** was applied:

SMOTE was applied to create synthetic samples of the minority class (HIV positive), ensuring that the dataset was balanced before model training.

Mathematically, SMOTE works by selecting a minority class sample and introducing synthetic points along the line segments joining its k-nearest neighbours. This process can be defined as:

New Sample=Original Sample+Random Factor×(Neighbor Sample−Original Sample)

#### 3. Train-Test Split

The dataset was split into training and testing sets in a 70:30 ratio to evaluate the model's performance on unseen data.

#### 4. Model Selection

Two machine learning models were selected for this task:

* **Random Forest Classifier**
* **XGBoost Classifier**

#### 5. Hyperparameter Tuning

Hyperparameters for both models were tuned using **GridSearchCV**. The goal was to find the best combination of hyperparameters that maximized the F1 score and ROC-AUC. The following hyperparameters were tuned:

* **Random Forest:**

Random Forest is an ensemble learning method that operates by constructing multiple decision trees during training and outputting the mode of the classes for classification tasks.

**Best Hyperparameters**

* + max\_depth: **None**: This allows the trees to grow until all leaves are pure, which helps capture complex patterns but risks overfitting.
  + max\_features: **auto**: Uses the square root of the number of features at each split, balancing between speed and accuracy.
  + min\_samples\_leaf: **1**: Ensures that leaf nodes have at least one sample, allowing for the capture of all nuances in the data.
  + min\_samples\_split: **5**: Nodes with fewer than 5 samples are not split, preventing overfitting.
  + n\_estimators: **300**: A large number of trees reduces the variance of predictions and improves the model's robustness.
* **XGBoost:**

XGBoost is a gradient boosting technique that builds models sequentially, where each new model corrects errors made by the previous ones.

**Best Hyperparameters**:

* + colsample\_bytree**: 0.8**: Uses 80% of features to grow each tree, preventing overfitting and enhancing model generalization.
  + learning\_rate: **0.1**: Controls the contribution of each tree, balancing learning speed and accuracy.
  + max\_depth: **10**: Limits the depth of each tree to prevent overfitting while capturing complex patterns.
  + n\_estimators: **300**: The number of trees in the ensemble.
  + subsample: **0.9**: Uses **90%** of the data at each iteration, helping to prevent overfitting.

The best hyperparameters were selected iteratively through GridSearchCV with cross-validation.

#### 6. Model Training

The models were trained using the best hyperparameters obtained from GridSearchCV. The training process involved fitting the models on the training data and evaluating them on the test data.

#### 7. Model Evaluation

The models were evaluated on both training and test sets using various metrics, including accuracy, precision, recall, F1 score, and ROC AUC score.

* **Random Forest Evaluation**

1. Train Set Metrics:
   * F1 Score: 0.9092
   * ROC AUC Score: 0.9627
2. Test Set Metrics:
   * F1 Score: 0.7376
   * ROC AUC Score: 0.8043

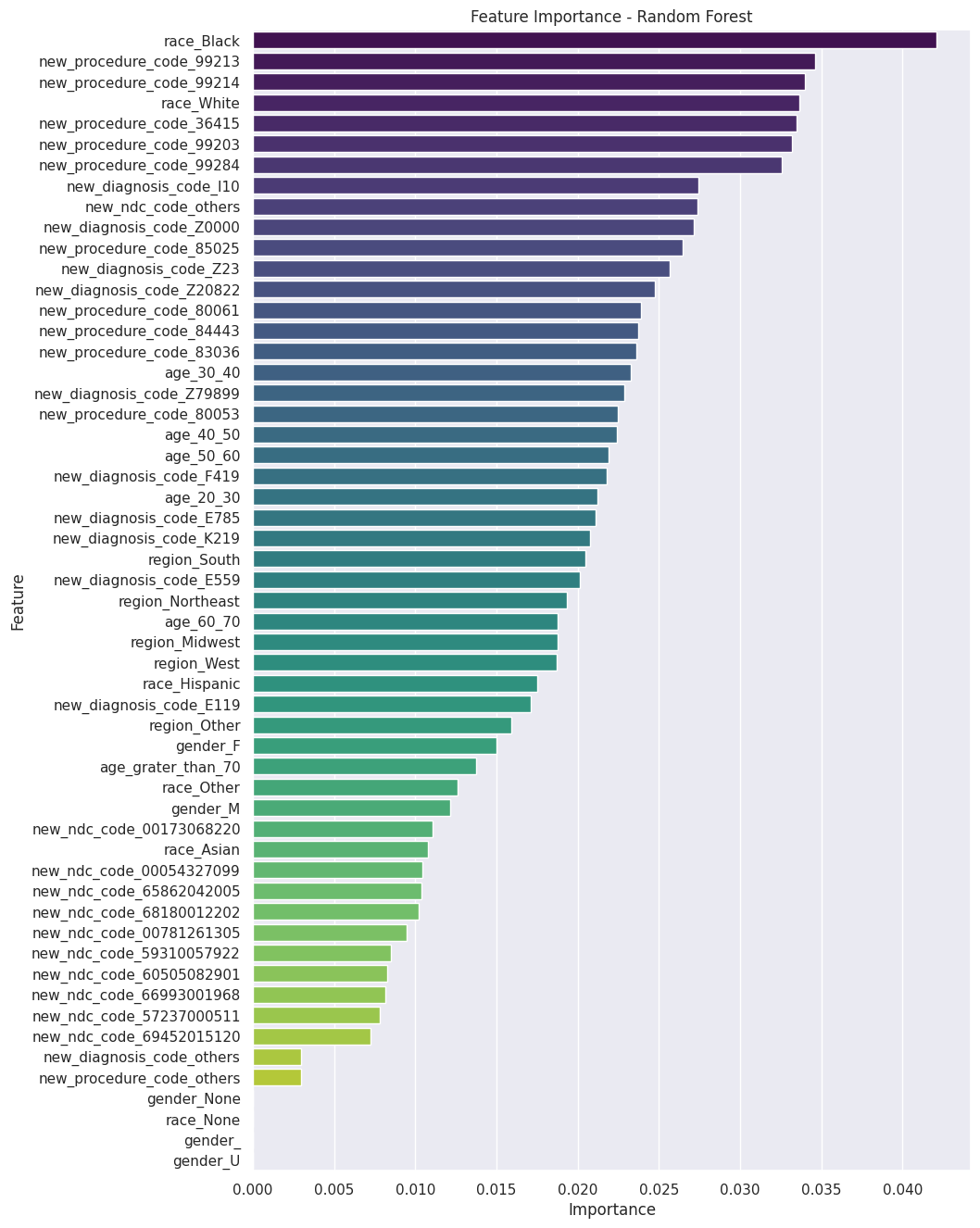
* **XG Boost Evaluation**

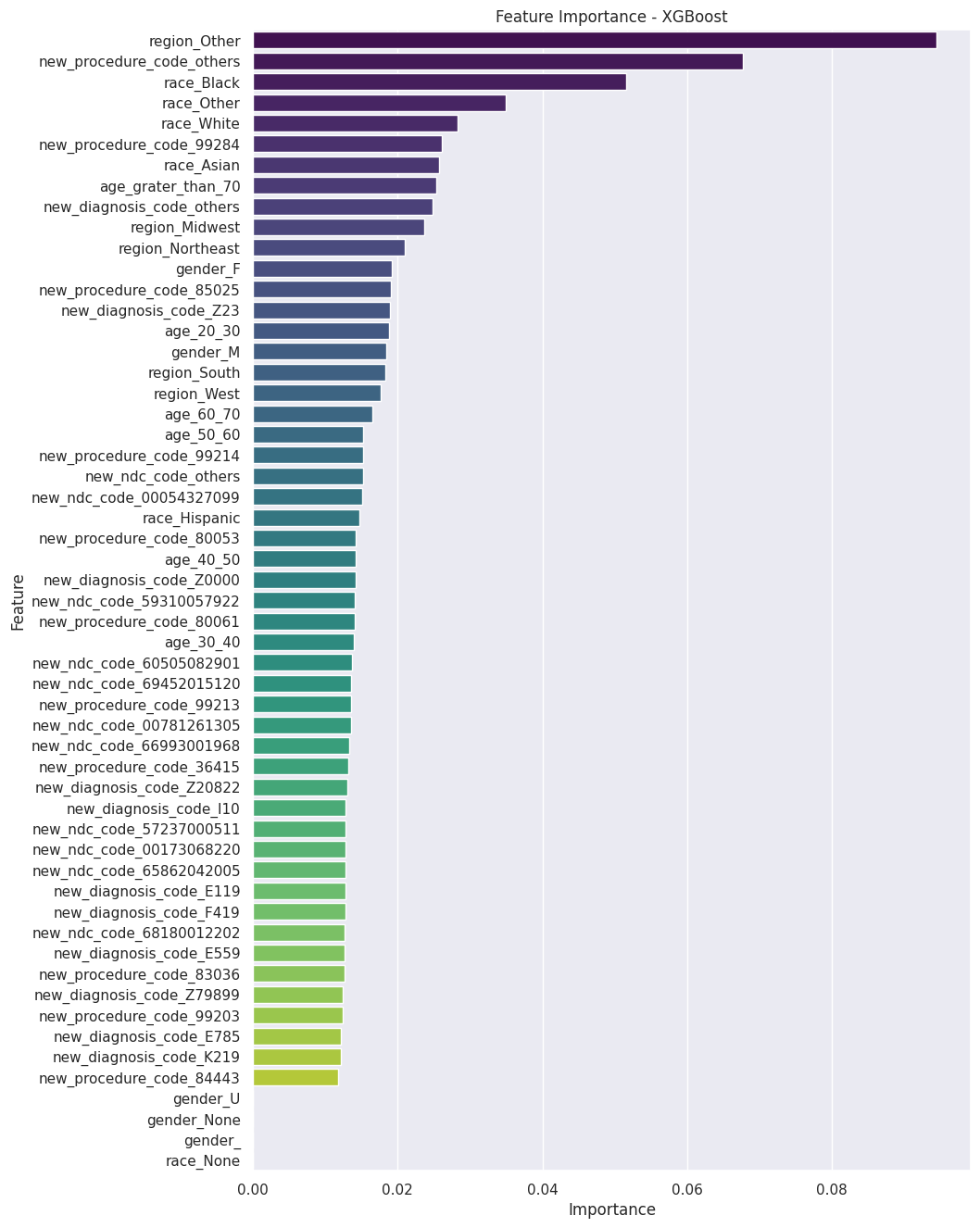
1. Train Set Metrics:
   * F1 Score: 0.8274
   * ROC AUC Score: 0.9121
2. Test Set Metrics:
   * F1 Score: 0.7155
   * ROC AUC Score: 0.7872

The XGBoost model, while slightly less accurate than the Random Forest on the training set, generalizes better to the test set. Thus concluding Random forest fits best to the data and better prediction accuracy is obtained on unseen data.

#### 8. Feature Importance

The importance of each feature was calculated and visualized for both models. This helps in understanding which variables are most influential in predicting the presence of HIV.





#### 9. Results Visualization

ROC curves were plotted for both models to visually assess their performance. Feature importances were also visualized to highlight the most critical features.

10. Prediction on Holdout Set

Based on the random forest model the predictions were done on the holdout set and submitted the final scoring table : “**scoring\_dataset\_hackathon\_a\_2024\_08\_30\_10\_46\_47**”

## Conclusion

This project successfully implemented Random Forest and XGBoost models to predict HIV status. Through the use of SMOTE, the dataset's class imbalance was effectively addressed. Hyperparameter tuning was essential in optimizing model performance, with the best parameters identified through an iterative process.

The results demonstrate that while both models perform well, careful consideration must be given to model complexity to prevent overfitting, as seen in the discrepancy between training and test set metrics. Further steps could involve experimenting with regularization techniques, feature engineering, or ensemble methods to improve generalization.

## Future Work

To enhance the model's performance and reduce overfitting, future work could explore:

* **Advanced Feature Selection**: Using techniques like recursive feature elimination (RFE) to reduce the feature space.
* **Regularization Techniques**: Applying L1 or L2 regularization to control model complexity.
* **Ensemble Methods**: Combining multiple models using techniques like stacking or bagging to improve prediction accuracy.