CREDIT CARD CLIENT STATUS PREDICTION

Yangyin Ke

Brown university

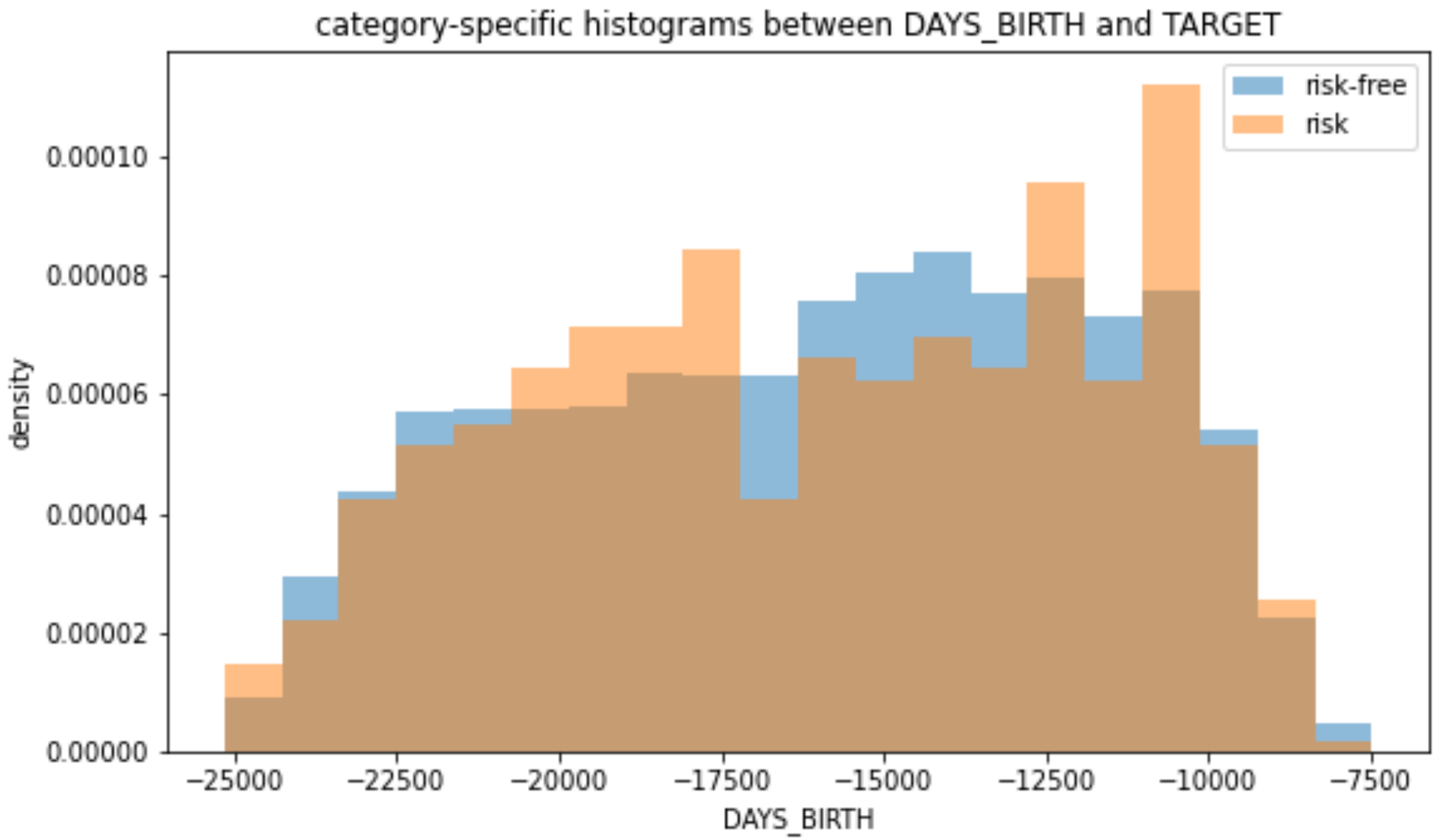
Github: https://github.com/yangyinke/Credit-Card-Client-Status-Prediction

INTRODUCTION

- Problem description:
  - Predict whether clients are risk users or risk-free users based on their personal information
  - Classification problem: Risk User (overdue > 60 days) vs. Risk-free User (pay off or overdue < 60 days)
- Importance:
  - Use in credit card application approval
- Data source:
  - Kaggle: Credit Card Dataset for Machine Learning
  - *Link: https://www.kaggle.com/rikdifos/credit-card-approval-prediction*

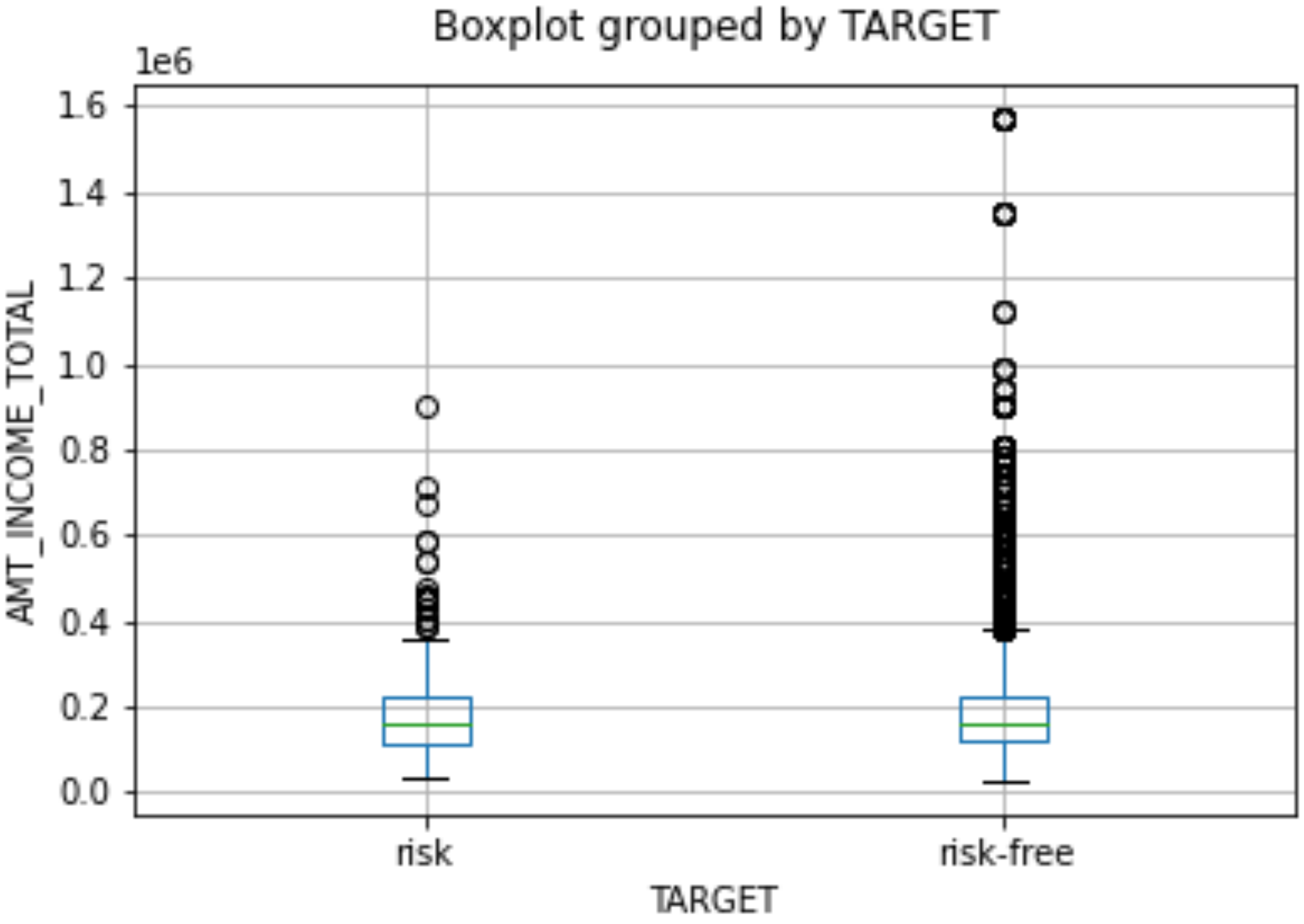
EDA I
DAYS OF BIRTH
&
RISK STATUS

- Similar distribution for different risk status
- Weak correlation



EDA II
ANNUAL INCOME
&
RISK STATUS

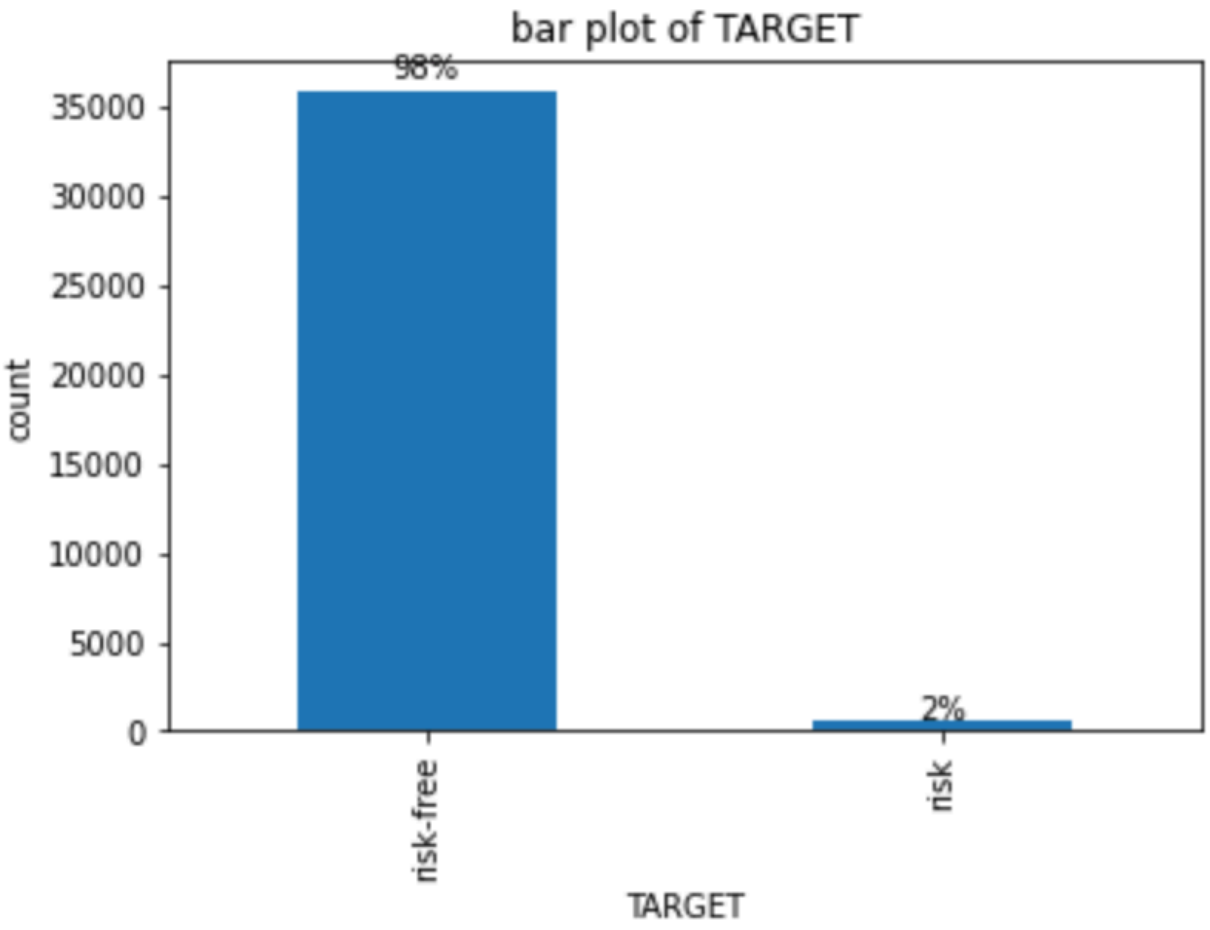
- Risk-free user gain slightly more annually than risk user
- More outliers in risk-free user
- People with higher income is more likely to be risk-free



EDA III DISTRIBUTION OF RISK STATUS

Imbalanced dataset

- 98% Risk-free User
- 2% Risk User
- Stratified splitting



GENERAL PIPELINE

Data splitting

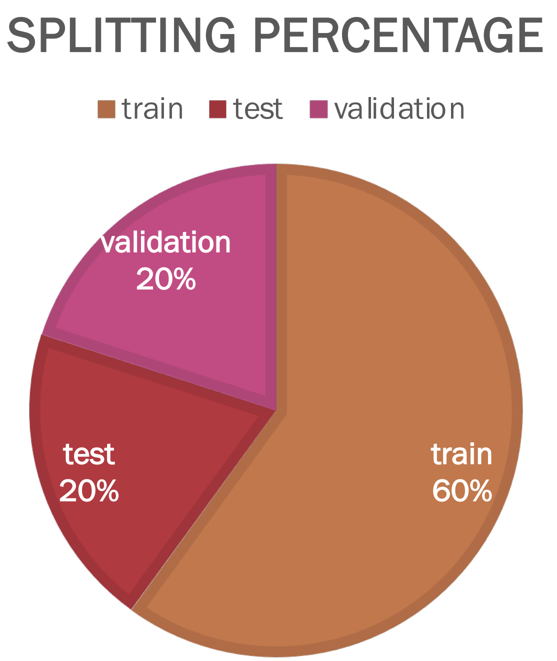
Data preprocessing

Loop through hyperparameter combinations (GridSearchCV)
=> best model at certain random state

Calculate and record test score

DATA SPLITTING

- Percentage:
  - train: 60%
  - test: 20%
  - validation: 20%
- Stratified
  - Original dataset is IID and imbalanced (98% Risk-Free User vs. 2% Risk User)



DATA PREPROCESSING

- Preprocessors:
  - **OneHotEncoder**: not clearly ordered categorical variables (e.g. gender)
  - **OrdinalEncoder**: clearly ordered categorical variables (e.g. education level)
  - **StandardScaler**: continuous variables without boundaries (e.g. annual income)
- Missing Value:
  - Type_of_occupation: 30%
  - Treat missing values as another category
- Labels:
  - Risk-free & Risk => 0 & 1

MACHINE LEARNING ALGORITHMS TUNNING

Logistic Regression

- C (regularization strength): 10, 100, 1000

Random Forest

- max_depth (maximum depth of the tree) : 40, 60, 80
- max_features (maximum fraction of features considered at each split): 0.3, 0.5, 0.7
- n_estimators (number of trees in the forest): 50, 60, 70.

XGBoost

- n_estimators (number of trees used): 1000, 10000.

K Nearest Neighbors

- n_neighbors (number of neighbors to use): 1, 3
- Weights (weight function): ‘uniform’, ‘distance’

TEST SCORES (F1_SCORE)

Baseline: 0.66667

Logistic Regression: 0.68415

- 6.25016 standard deviations above the baseline

Random Forest: 0.98798

- 327.29757 standard deviations above the baseline

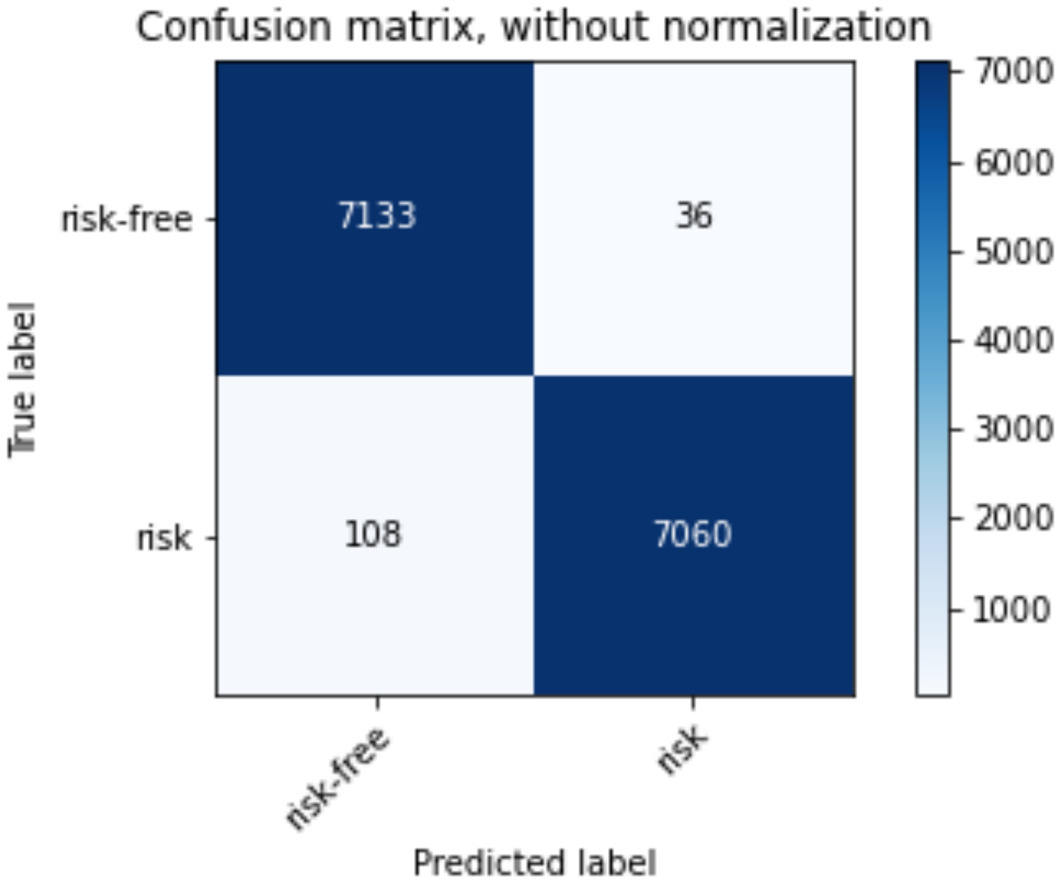
XGBoost: 0.99020

- 394.5923 standard deviations above the baseline

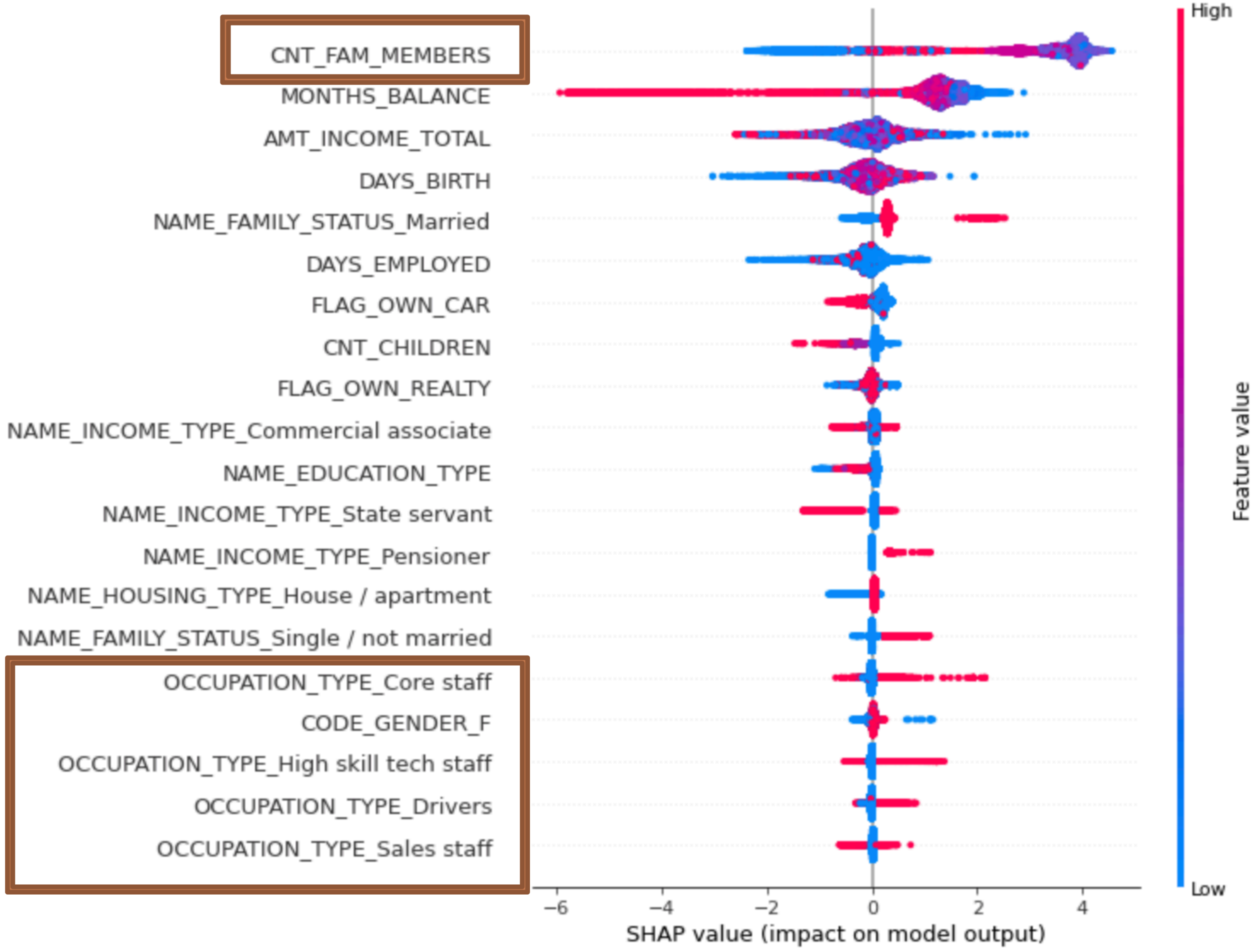
K Nearest Neighbors: 0.96886

- 281.6093 standard deviations above the baseline

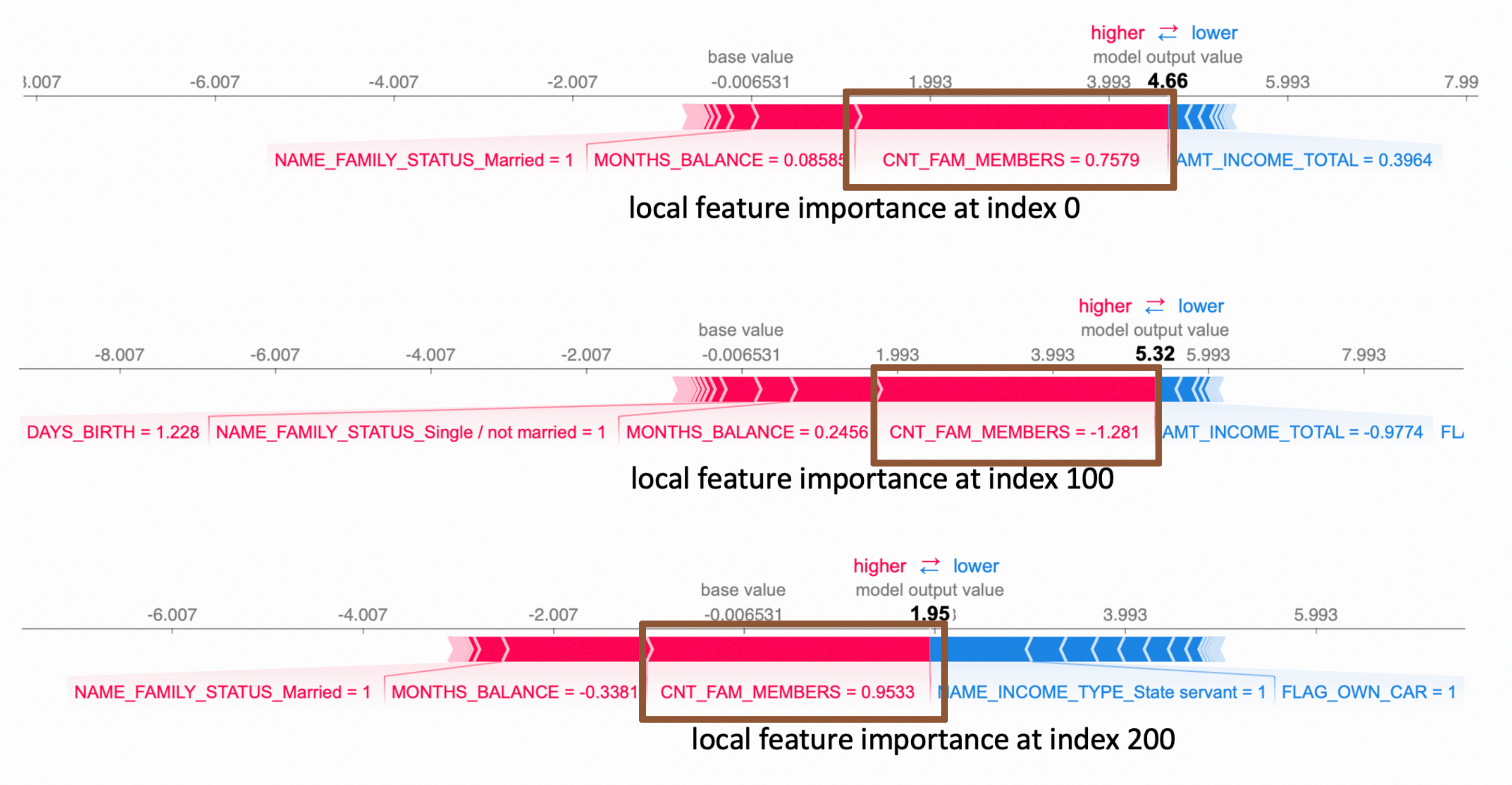
CONFUSION MATRIX



GLOBAL FEATURE IMPORTANCE



LOCAL FEATURE IMPORTANCE



Least important: occupation_type (30% missing)

OUTLOOK

Weak Spot

- Have problem handling imbalanced incoming data

Improvement

- Training on an imbalanced dataset with some other smart ways
- Try reduced_feature_xgboost

Thank
you