AIA Recommendation

Data Science Case Study

Linh V Nguyen

Agenda

1. Problem and Data Overview
2. Feature Engineer
3. Modeling and Metrics
4. Insight Analysis
5. Summary and Conclusions

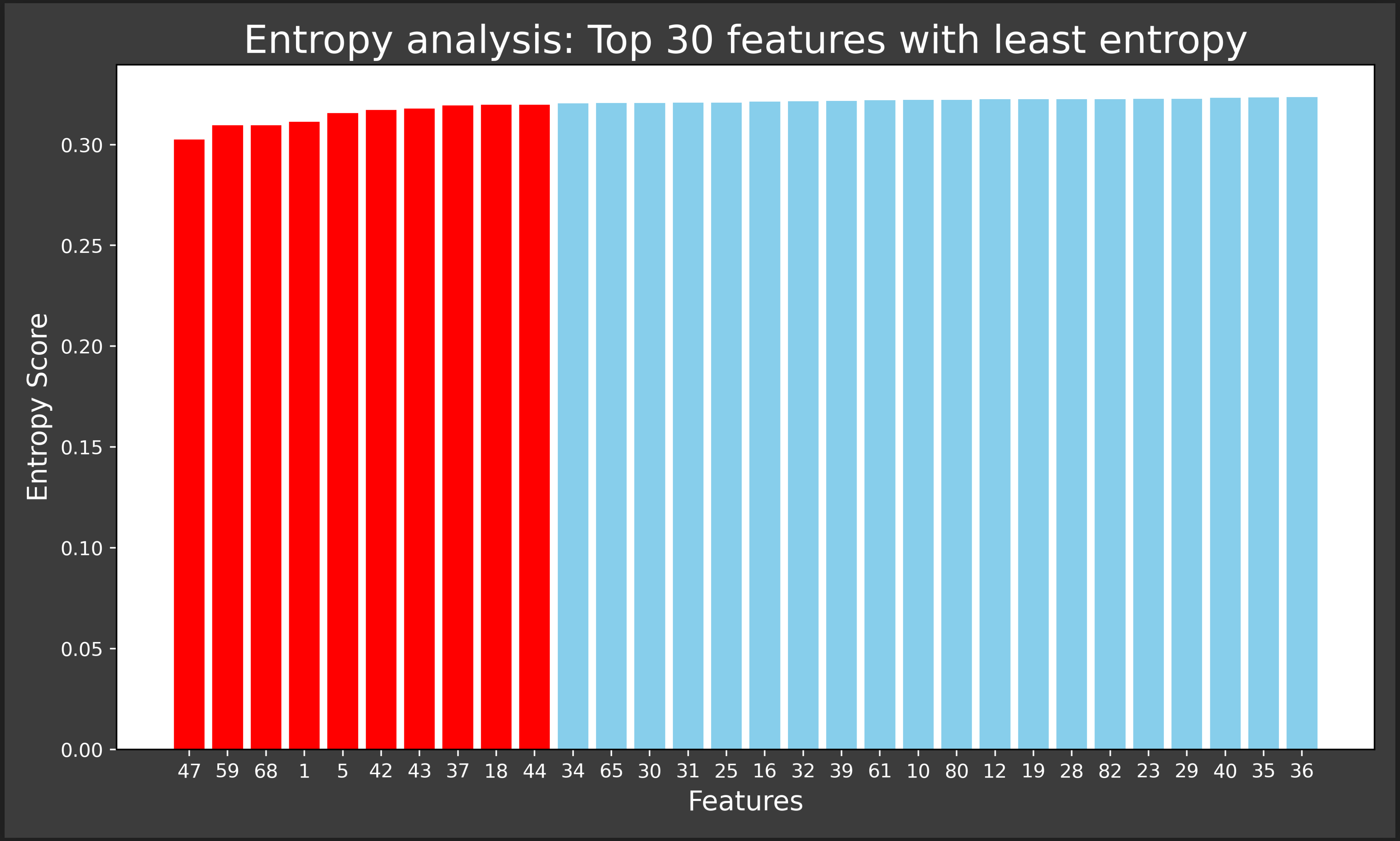
Problem and Data Overview

1. Recommended a list of 800 users with highest probability to purchase AIA policy
2. Imbalance Binary Classification
3. Data: already encoded, and processed
4. Columns/Variables: Categorical
5. No null/missing values
6. High dimension, small samples: 85 x 5822 (possibly curse of dimensionality)

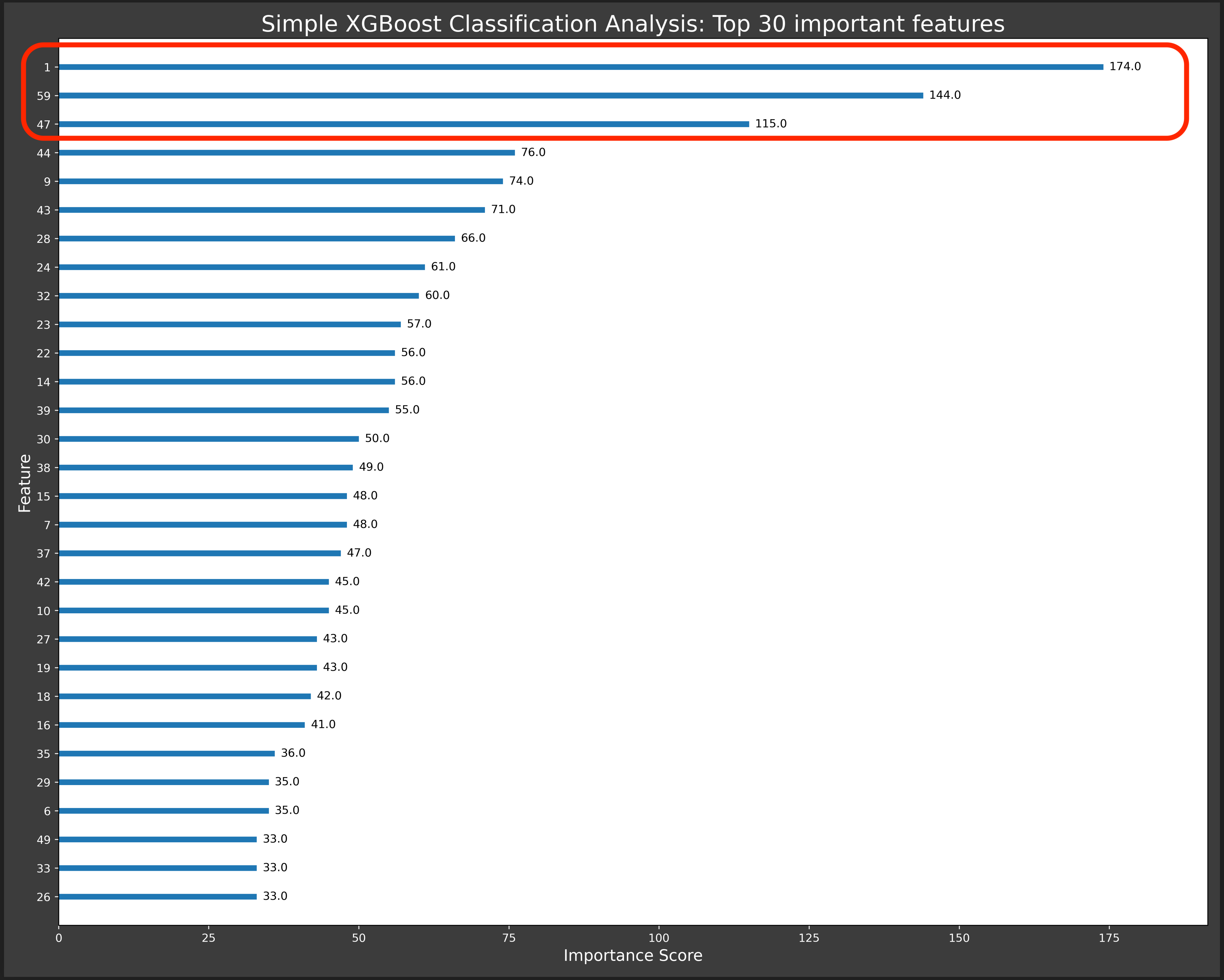
Feature Engineer - Correlation Check

- Some pair of features strongly correlated (> 0.7), including:
- + Customer subtype and main type (1 & 5)
- + Household size and household with children
- - Rented house and Home owners (30 & 31)
- - National Health Services and Private Health Insurance (35 & 36)
- + Product ownership: Contribution to a product and corresponding amount (43-85)
- Some features are possibly mutually exclusive/inclusive
- Reduce 85 features to a small subset for modeling

Feature Engineer - Selection

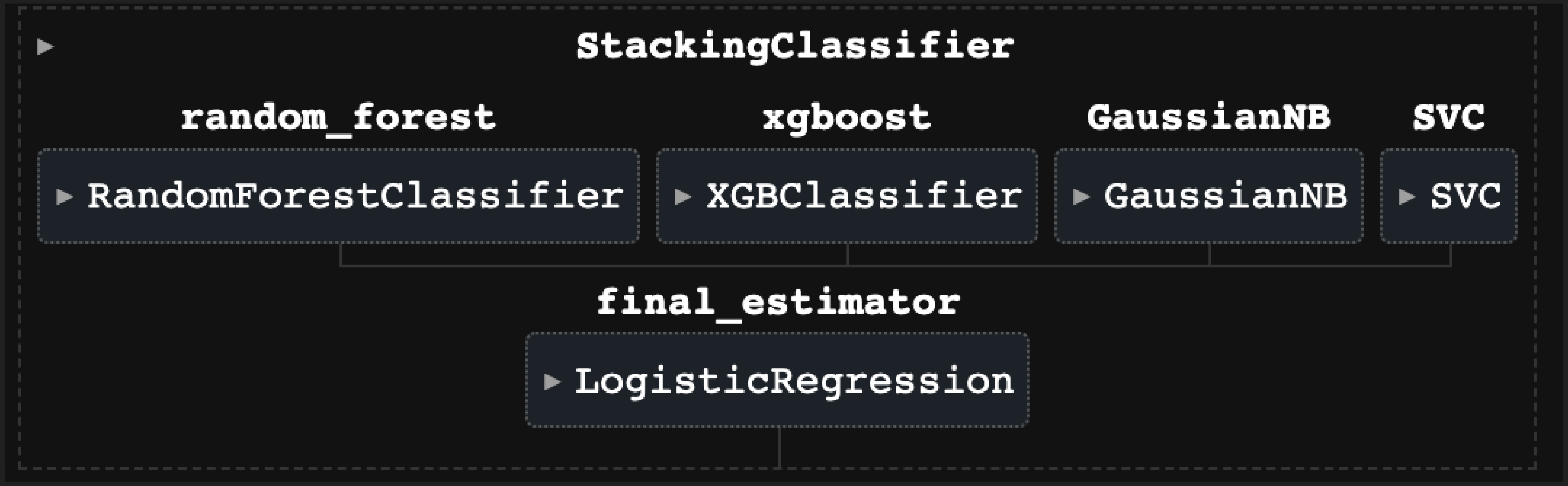


Feature Engineer - Selection

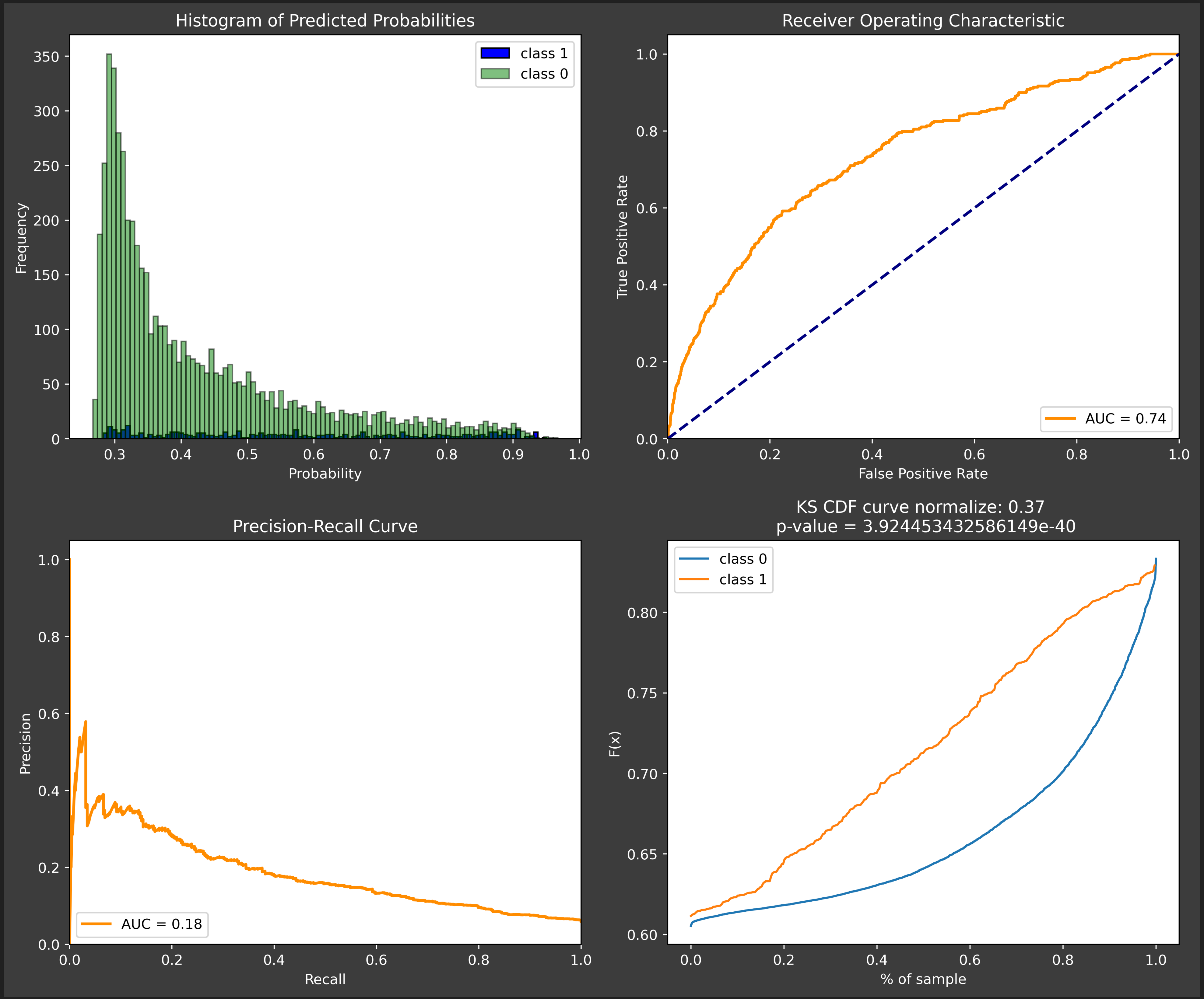


Modeling and Metrics

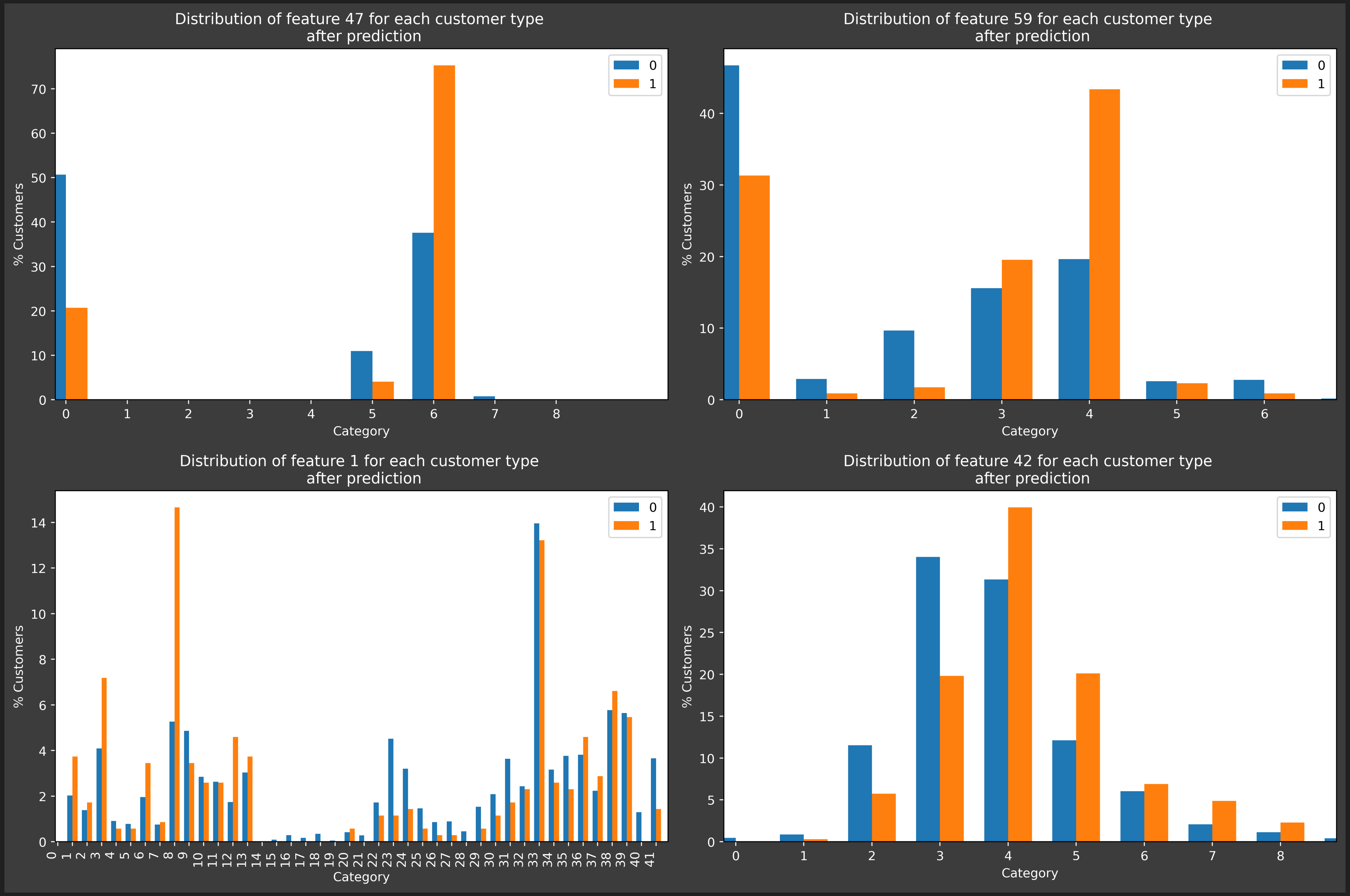
- Training with 5-fold cross validation
- Features: 47, 1, 59, 42, 43, 37, 18, 44
- Class Weights 1/0: 16/1
- Metrics: KS-score, ROC-AUC, ROC-Precision Recall
- Stacking Classication



Modeling and Metrics



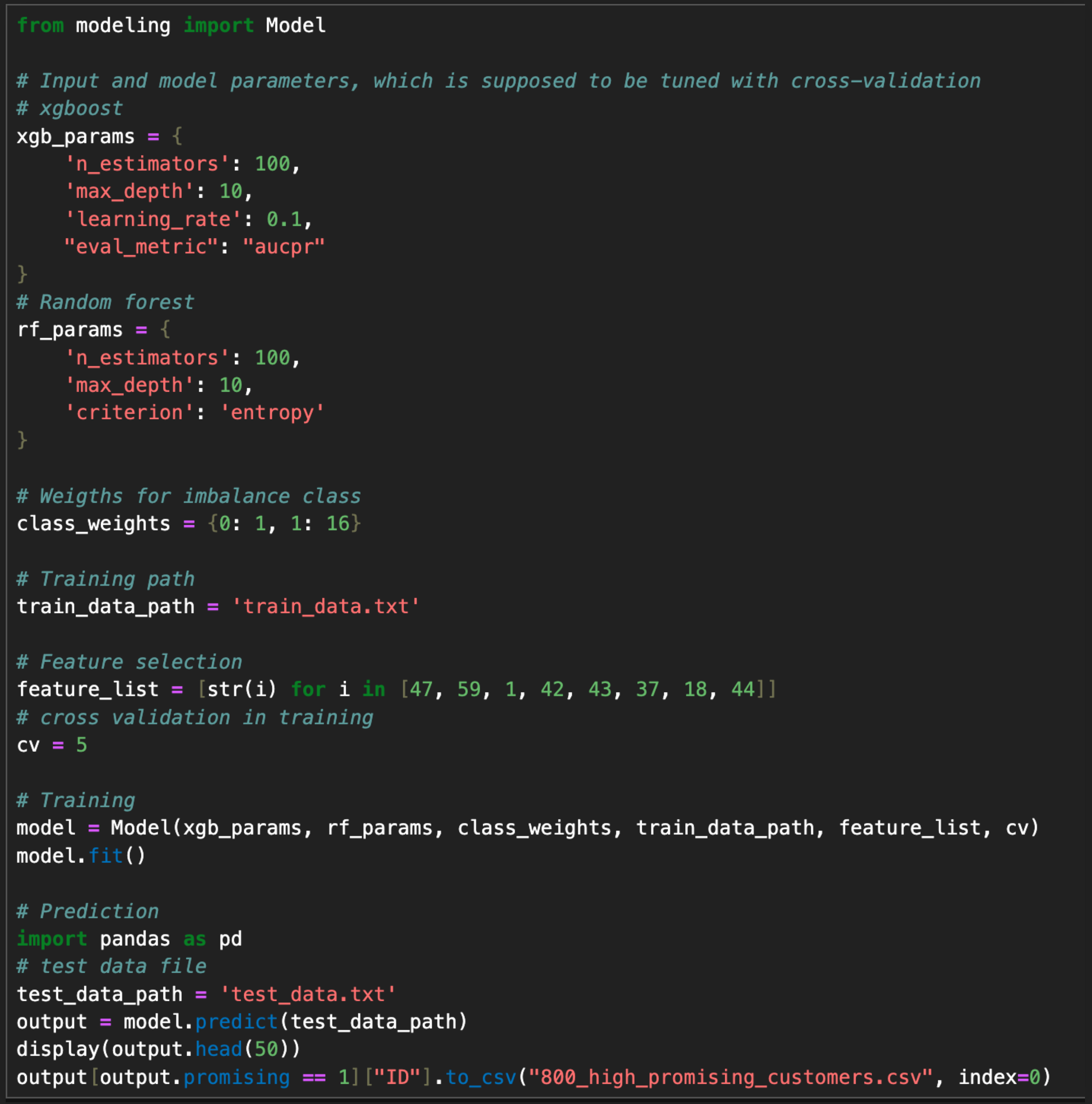
Insight Analysis



Summary & Conclusions

- Assuming higher category means higher level
- Customers who purchase AIA tends to be upper-middle class, higher income, affluent and have high level of contribution and number of car/fire policies

Appendix - How to run the model (in Run_code_here.ipynb)



```python
from modeling import Model

# Input and model parameters, which is supposed to be tuned with cross-validation
# xgboost
xgb_params = {
    'n_estimators': 100,
    'max_depth': 10,
    'learning_rate': 0.1,
    "eval_metric": "aucpr"
}
# Random forest
rf_params = {
    'n_estimators': 100,
    'max_depth': 10,
    'criterion': 'entropy'
}

# Weigths for imbalance class
class_weights = {0: 1, 1: 16}

# Training path
train_data_path = 'train_data.txt'

# Feature selection
feature_list = [str(i) for i in [47, 59, 1, 42, 43, 37, 18, 44]]
# cross validation in training
cv = 5

# Training
model = Model(xgb_params, rf_params, class_weights, train_data_path, feature_list, cv)
model.fit()

# Prediction
import pandas as pd
# test data file
test_data_path = 'test_data.txt'
output = model.predict(test_data_path)
display(output.head(50))
output[output.promising == 1]["ID"].to_csv("800_high_promising_customers.csv", index=0)
```