Bank Customer Segmentation

Presented by: Samuel Sim Wei Xuan

Business Problem Overview

The bank aims to increase its customer base and revenue by identifying potential affluent customers within its Existing To Bank (ETB) segment. By upgrading these customers from normal to affluent status, the bank can offer tailored products and services to enhance satisfaction and drive revenue growth.

Target Segment

Focus on the Existing To Bank (ETB) customers, particularly those currently classified as normal but with the potential to become affluent.

Data Utilization

Analyze a comprehensive dataset featuring various customer attributes and a binary label indicating 'affluent' or 'normal' status to identify potential candidates for upgrade.

Expected Result

The bank is poised to effectively identify and target hidden affluent customers within the ETB segment, leading to increased revenue, customer satisfaction, and a stronger market presence.

Machine Learning Problem Definition (1/3)

Objectives

Customer Segmentation:

Utilize supervised machine learning (classification) to identify potential affluent customers within ETB segment.

Identification of Upgrade Candidates:

Predict whether a customer should be classified as "affluent" or "normal" based on their data profile, focusing on maximizing recall for the affluent class. False positives would actually be the hidden affluent customers to target.

Data Overview

Features:

Customer data from the provided dataset, including demographic information, transaction history, account balances, etc.

Labels Usage:

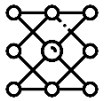
Binary labels indicating whether each customer is affluent or normal, used for training and evaluating the classification model.

Machine Learning Problem Definition (2/3)

Model Development

Preprocessing and Feature Engineering:

Clean and preprocess data to ensure it's suitable for classification, including handling missing values, encoding categorical variables, etc.



Model Selection:

Choose appropriate classification algorithms (e.g., XGBoost, Random Forest) to predict the target variable effectively.

Model Evaluation:

Given the best model choice, tune the model with a focus on maximizing recall for the affluent class.

Business Application

Targeted Marketing:

Utilize predictions from the classification model to target potential affluent customers with tailored marketing campaigns and product offerings.

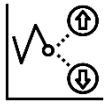
Segment Analysis:

Analyze the characteristics and behaviors of predicted affluent customers to refine marketing strategies and enhance customer engagement.

Machine Learning Problem Definition (3/3)

Rationale for Classification over Clustering

While clustering can provide valuable insights into grouping customers based on similarities in their data profiles, the decision to opt for classification instead was driven by several key factors:



Granular Predictions:

Classification provides individual predictions for each customer, enabling precise targeting and personalized strategies.

Interpretability:

Classification models offer feature importance metrics, helping understand the factors driving segmentation.

Threshold Control:

With classification, the bank can set thresholds to predict affluent customers, aligning with strategic goals.

Model Evaluation:

Clear evaluation metrics like recall allow for measuring model effectiveness and refining approaches.

Strategy Development:

Insights from classification aid in developing targeted marketing and product strategies.

Technical Details (1/2)

1. Folder Structure

Kedro Structure: Easier for MLE to deploy with an actual **Kedro** project

- ***maybank/conf/base:*** yaml configurations.
- ***maybank/conf/local:*** Empty as no secret credentials
- ***maybank/data:*** Contains all the saved data (raw, processed) and EDA html output
- ***maybank/docs:*** Contains the sphinx documentation for the src folder
- ***maybank/notebooks/main_notebook.ipynb:*** Contains the data processing, EDA and modeling work
- ***maybank/src/pipelines/main_pipeline.py:*** Runs the entire pipeline from end-to-end before inference
- ***maybank/notebooks/analysis.ipynb:*** Contains the inferencing and analysis of results
- ***maybank/src:*** Contains all the scripts that the notebook imports
- ***maybank/tests:*** Empty at the moment, it is for pytest integration

Technical Details (2/2)

2. Dependencies Management

- Used **Poetry** for managing project dependencies
- Provides a reliable and efficient tool for dependency management

3. Linting

- Implemented linting with **ruff** and **flake8** to ensure code consistency and quality, handled by **Poetry**

4. Documentation

- Set up **sphinx** documentation
- To see **sphinx** configurations, look under ***docs/config.py***
- To view the documentation, look under ***docs/html/index.html*** to view the entire interactive HTML documentation.

Suggested Deployment (MLOPs) (1/9)

Overall Steps:

1. Develop with **Kedro** and **Poetry**
2. Integrate **MLflow** for experiment tracking and workflow
3. Build a **Docker** file to package the entire project
4. Set up CI/CD pipelines with **Jenkins**
5. Deploy the project to **OpenShift**
6. Create a **Django REST API** project to expose my deployed project endpoint
7. Integrate backend calls to a front-end for the bank users to quickly drop a dataset of ETB Customers for quick inference
8. Set up **OpenTelemetry** and integrate it with a UI such as **Kibana** for logging and tracing of deployed model inference API calls

Suggested Deployment (MLOPs) (2/9)

Step 1:

1. Develop with **Kedro** and **Poetry**
    - Organize this machine learning project using **Kedro** for project structuring and workflow management
    - Use **Poetry** for dependency management, ensuring consistent environments across different machines

Suggested Deployment (MLOPs)(3/9)

Step 2:

Incorporate **MLflow** into this **Kedro** project for experiment tracking, model versioning, and workflow management

Suggested Deployment (MLOPs)(4/9)

Step 3:

- Create a **Dockerfile** to package this **Kedro** project, **MLflow**, and other dependencies into a containerized environment

Suggested Deployment (MLOPs) (5/9)

Step 4:

- Write a **Jenkinsfile** defining the CI/CD pipeline for this project
- Include stages for checking out the source code, building the **Docker** image, running tests and linting, pushing the image to a container registry, and deploying to **OpenShift**

Suggested Deployment (MLOPs)(6/9)

Step 5:

- Set up an **OpenShift** cluster and create a project for deploying this machine learning project.
- Prepare a deployment configuration file (deployment.yaml) describing how to deploy this dockerized project in **OpenShift**
- Apply the deployment configuration to the **OpenShift** project

Suggested Deployment (MLOPs)(7/9)

Step 6:

- Develop a **Django REST API** project to expose endpoints for the deployed machine learning model
- Implement endpoints for inference, allowing users to send data for predictions

Suggested Deployment (MLOPs)(8/9)

Step 7:

- Develop a front-end interface for bank users to interact with the **Django REST API**
- Allow users to upload datasets of ETB customers for quick inference using the deployed machine learning model

Suggested Deployment (MLOPs)(9/9)

Step 8:

- Implement **OpenTelemetry** for logging and tracing of deployed model inference API calls
- Integrate **OpenTelemetry** with a UI tool like **Kibana** to visualize and analyze logs and traces, enabling efficient monitoring and debugging

Additional Considerations

- The decision not to use **PySpark** for a small dataset but planning for its deployment later is a strategic choice balancing current efficiency with future scalability
- There will just be a need to refactor the **Pandas** codes to **PySpark**, which is relatively simple

EDA (Using Sweetviz) (1/1)

1. **Comprehensive Insights:**
    - Offers diverse visualizations for thorough data exploration
    - From histograms to association analysis, it covers various aspects of dataset examination
2. **Time Efficiency:**
    - Automates EDA processes, saving valuable time
    - Provides quick insights into data distribution and associations (correlation etc.)
3. **Quality Assessment:**
    - Detects missing values and zeros, aiding in data quality assessment

Check ***maybank/data/raw/EDA.html*** for the interactive EDA

Missing Values Imputation (1/4)

Imputation Strategy Analysis

**Inside the notebook has deeper analysis with codes*

1. **Understanding Metadata:**
    - Delve into metadata to comprehend the definition of each feature.
    - Facilitates informed assumptions for handling missing values effectively.
2. **Tailored Imputation Strategies:**

    Customize imputation based on the characteristics and significance of each feature.
    - Ensures a more effective data treatment approach.

Missing Values Imputation (2/4)

No.	Variable	Missing %	Imputation Strategy	Explanation
1	C_ID	0%	No missing values	-
2	C_AGE	0%	No missing values	-
3	C_EDU	58%	Constant Imputation: "Not Provided"	Since it is the customer education, missing values are imputed with "Not Provided", cannot use "Others" as it is already specified.
4	C_HSE	66%	Constant Imputation: "Not Provided"	Since it is the customer house type, missing values are imputed with "Not Provided".
5	PC	<1%	Zero Imputation	Distribution is not skewed and only <1% missing. PC is postal code, hence does not make sense to use mean/mode imputation. Hence use zero imputation instead.
6	INCM_TYP	45%	Mode Imputation: 2.0	Drop at 1, 7, and 8 bins. Impute with 2, assuming no income would be already be recorded under either 1 or 8 depending on which is the lowest income type bin.
7	gn_occ	1%	Constant Imputation: "Not Provided"	Since it is occupation, missing values are imputed with "Not Provided", cannot use "Others" as it is already specified.
8	NUM_PRD	0%	No missing values	-
9	CASATD_CNT	38%	Zero Imputation: 0	Impute with 0, indicating no CASA or TD accounts, verified with below CASA and TD analysis.
10	MTHCASA	41%	Zero Imputation: 0	Impute with 0, indicating no CASA account.
11	MAXCASA	41%	Zero Imputation: 0	Impute with 0, indicating no CASA account.
12	MINCASA	41%	Zero Imputation: 0	Impute with 0, indicating no CASA account.

Missing Values Imputation (3/4)

No.	Variable	Missing %	Imputation Strategy	Explanation
13	OWN_CASA (NEW)	-	New Binary Column: 1 for owned, 0 for not owned	Created a new binary column to differentiate not owning a CASA and an empty CASA account, since MTH/MAX/MINCASA already have 0s, which implies an empty CASA account.
14	DRvCR	55%	Zero Imputation: 0	Impute with 0, indicating either zero debit or absence of credit (0 division).
15	MTHTD	-	Zero Imputation: 0	Impute with 0, indicating no TD account
16	MAXTD	-	Zero Imputation: 0	Impute with 0, indicating no TD account
17	OWN_TD (NEW)	-	New Binary Column: 1 for owned, 0 for not owned	Created a new binary column, upon below CASA and TD analysis, it can be shown that indeed missing values in CASATD_CNT implies 0 CASA and TD accounts.
18	Asset_value	0%	No missing values	-
19	HL_tag	96%	Constant Imputation: 0	Supposed to be either 1 or 0 according to metadata, and only 1 exists at the moment. Impute with 0.
20	AL_tag	92%	Constant Imputation: 0	Supposed to be either 1 or 0 according to metadata, and only 1 exists at the moment. Impute with 0.
21	pur_price_avg	92%	Zero Imputation: 0	Impute with 0, no 0s, hence should be safe assumption that the property purchase price is 0. Also _avg could imply average, hence if no property owned, then 0 division.
22	UT_AVE	96%	Zero Imputation: 0	Impute with 0, indicating no UT transaction, no 0 exists and verified with below Unit Trust Analysis.
23	MAXUT	96%	Zero Imputation: 0	Impute with 0, indicating no UT transaction, no 0 exists and verified with below Unit Trust Analysis.
24	N_FUNDS	96%	Zero Imputation: 0	Impute with 0, indicating no funds owned, no 0 exists and verified with Unit Trust Analysis.

Missing Values Imputation (4/4)

No.	Variable	Missing %	Imputation Strategy	Explanation
25	MAX_MTH_TRN_AMT	82%	Zero Imputation: 0	Impute with 0, verified with below Credit Card TRN Analysis, follow imputation of ANN_N_TRX.
26	MIN_MTH_TRN_AMT	82%	Zero Imputation: 0	Impute with 0, verified with below Credit Card TRN Analysis, follow imputation of ANN_N_TRX.
27	AVG_TRN_AMT	82%	Zero Imputation: 0	Impute with 0, verified with below Credit Card TRN Analysis, follow imputation of ANN_N_TRX.
28	ANN_TRN_AMT	82%	Zero Imputation: 0	Impute with 0, verified with below Credit Card TRN Analysis, follow imputation of ANN_N_TRX.
29	ANN_N_TRX	82%	Zero Imputation: 0	Impute with 0. 0 does not exist hence 82% either no credit transaction or no credit card owned.
30	CC_AVE	74%	Zero Imputation: 0	Impute with 0, indicating no credit card owned in the past, hence safe to impute with 0.
31	CC_LMT	28%	Zero Imputation: 0	Impute with 0. indicating no credit card owned at the moment, hence safe to impute with 0. Assumption still valid since 28% within the above 82% of rows 25-29.
32	OWN_CC (NEW)	-	New Binary Column: 1 for owned, 0 for not owned	Created new binary column, to distinguish 0 credit limit due to bank assigning 0 limit and not owning a credit card hence 0 limit due to imputation.
32	OWN_PREV_CC (NEW)	-	New Binary Column: 1 for owned, 0 for not owned	Created new binary column, to distinguish 0 CC_AVE due to 0 CC_AVE and not owning a credit card in past hence 0 limit due to imputation.

Data Cleaning (1/1)

1. Impute missing data based on above analysis
2. Missing data mostly imputed with 0, hence data quality might not be the best which directly affects model performance
    - Future steps: Rebuild everything but with other forms of imputation
3. Optimize efficiency by converting float64 to float32

Feature Engineering (1/2)

Utilizing tree-based models like XGBoost or Random Forests: No need to standardize or normalize features.

Reasoning:

1. **Invariance to Monotonic Transformations:**
    - Tree-based models base decisions on feature comparisons, making them insensitive to monotonic transformations like normalization or standardization.
2. **Natural Handling of Different Scales:**
    - These models partition the feature space based on relative feature values, not their absolute magnitude, making them robust to varying scales among features.

Furthermore, XGBoost and Random Forests are robust machine learning models that are less sensitive to outliers, unlike clustering models like SVMs that are sensitive to boundary spaces or regression-based models.

Feature Engineering (2/2)

For categorical variables like occupation:

Utilize **Stratified K-Fold Target Encoding** on the specified categorical columns.

1. **<u>Prevent Data Leakage</u>**
    - To prevent data leakage and ensure each fold is representative of the whole dataset given our imbalanced dataset.
2. **<u>Tracking Purpose:</u>**
    - Track of all the encoders used for target encoding.
    - Under the assumption that future observations is from a similar distribution to given training set.

Data Sampling (1/2)

Combine under-sampling and over-sampling strategy to ensure a more balanced training fold

Under-sampling methods:

Method	Advantages	Disadvantages
ClusterCentroids	Preserves information while reducing majority class.	May not accurately capture underlying distribution.
CondensedNearestNeighbour	Selects subset representing majority class.	May inadvertently remove informative instances.
EditedNearestNeighbours	Removes noisy majority class instances.	Sensitive to noise, may not fully address imbalance.
RandomUnderSampler	Simple and fast.	May discard useful instances, effectiveness in solving imbalance may vary.

Choice: RandomUnderSampler, due to large majority class just use a straight forward simple one

Data Sampling (2/2)

Over-sampling methods:

Method	Advantages	Disadvantages
RandomOverSampler	Simple and effective.	May lead to overfitting, reduction in diversity.
SMOTENC	Handles both numerical and categorical features.	Requires parameter tuning, can be computationally intensive.
ADASYN	Focuses on low-density regions.	Sensitive to noise, requires careful parameter adjustment.
RandomOverSampler	Simple and effective.	May lead to overfitting, reduction in diversity.

Choice: SMOTENC,

- ADASYN appears preferable due to its capability to generate synthetic samples in regions where the classifier is likely to make errors but it treats variables as continuous scale
- SMOTENC is tailored for datasets with a mix of categorical and continuous features, incorporating the nature of categorical features during synthetic sample generation

Modeling (1/2)

Comparing 2 common classification approaches:

Evaluated on average recall (focus on ensuring I classify all the affluent) using (k=10) k-fold cross-validation to determine the best approach to proceed with:

1. **Random Forest**
    - n_estimators: 100
2. **XGBoost Classifier**
    - objective: binary: logistic
    - learning_rate: 0.001
    - n_estimators: 100

Modeling (2/2)

Comparing 2 common classification approaches:

Evaluated on average recall (focus on ensuring I classify all the affluent) using (k=10) k-fold cross-validation to determine the best approach to proceed with:

1. **Random Forest**
    - n_estimators: 100
2. **XGBoost Classifier**
    - objective: binary: logistic
    - learning_rate: 0.001
    - n_estimators: 100

Results:

Random Forest - Avg Precision: 0.58
Random Forest - Avg Recall: 0.63

XGBoost - Avg Precision: 0.48
XGBoost - Avg Recall: 0.78

XGBoost Tuning (1/3)

- **Utilizing XGBoost for classification**
  - Focus on maximizing average recall score, particularly crucial for identifying affluent classes effectively.
- **Employing Optuna, an optimization library**
  - Automatically fine-tune hyperparameters for the XGBoost classifier
  - Optuna performs an iterative search over specified hyperparameter ranges
  - Maximizes recall score via k-fold cross-validation
- **Leveraging SelectFromModel from scikit-learn**
  - To select the most important features for model training, integrated within the cross-validation
  - Aids in reducing complexity and improving interpretability by retaining only the most relevant features based on importance weights
  - Commented out due to the small number of features

XGBoost (Optuna Parameters) (2/3)

Due to time constraints (worked on the project for a day), I ran quickly over **50** trials instead of 1000s.

With our over-under resampler and k=10 fold validation

Parameter	Description	Range
objective	Objective function for XGBoost	binary:logistic
tree_method	Tree construction algorithm	gpu_hist
gpu_id	GPU device ID	0
n_estimators	Number of trees	1000 – **3000**
max_depth	Maximum depth of a tree	5 – 20
learning_rate	Learning rate	0.001 - 0.1 (log scale)
subsample	Subsample ratio of the training instances	0.6 - 1.0
colsample_bytree	Subsample ratio of columns when constructing each tree	0.6 - 1.0
gamma	Minimum loss reduction required to make a further partition on a leaf node	0.0 - 5.0
min_child_weight	Minimum sum of instance weight (hessian) needed in a child	1 - 20

XGBoost (Optuna Parameters) (3/3)

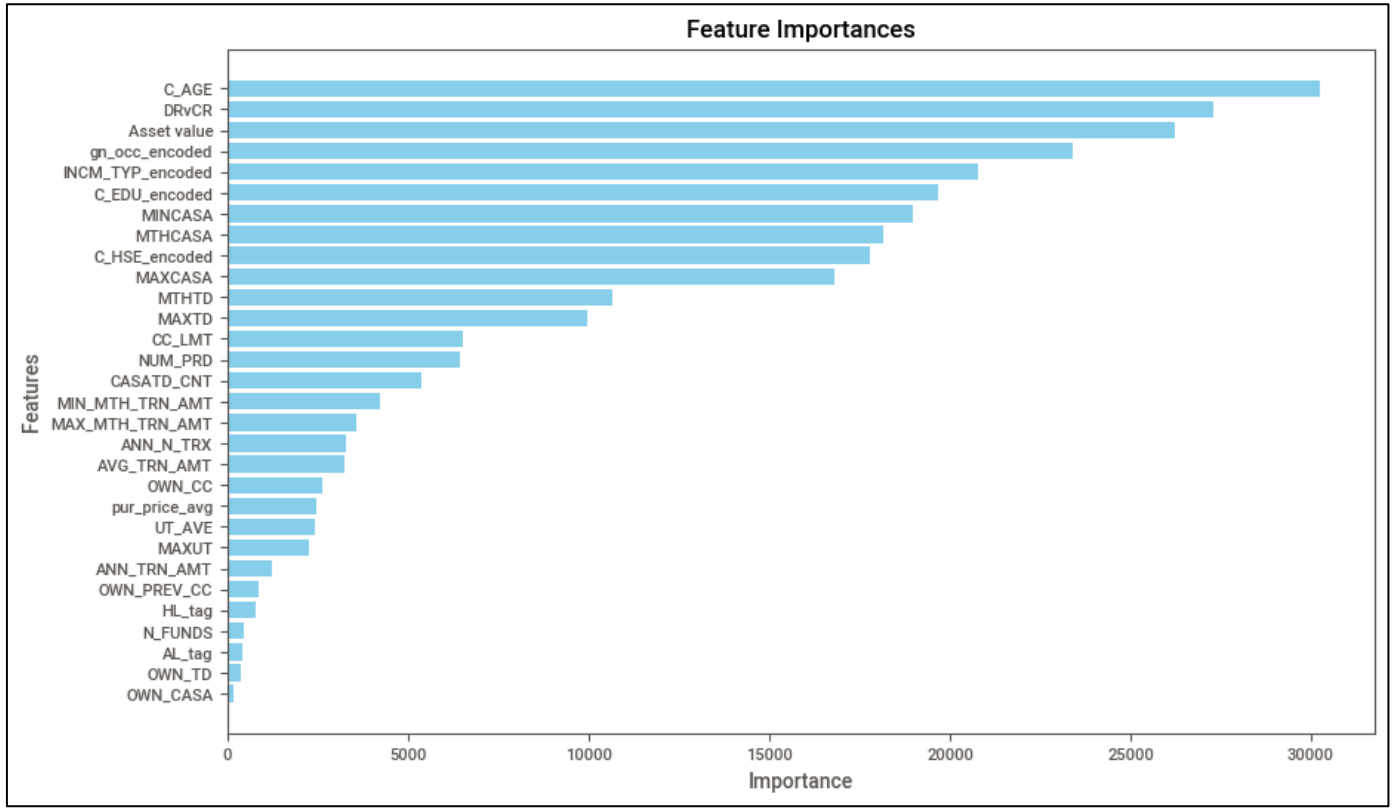
Best Hyper-parameters (average recall of **0.56**) to do **Inference:**

Note the average recall is “generalized”, as over-under sampling as applied to each training fold during k-fold validation, next slide will show the recall on overall ETB dataset

Parameter	**Description**	**Range**
objective	Objective function for XGBoost	binary:logistic
tree_method	Tree construction algorithm	gpu_hist
gpu_id	GPU device ID	0
n_estimators	Number of trees	2284
max_depth	Maximum depth of a tree	3
learning_rate	Learning rate	0.002192928185705617
subsample	Subsample ratio of the training instances	0.965188332889952
colsample_bytree	Subsample ratio of columns when constructing each tree	0.7215667495401594
gamma	Minimum loss reduction required to make a further partition on a leaf node	2.322789657995237
min_child_weight	Minimum sum of instance weight (hessian) needed in a child	1

Analysis (Feature Importance) (1/2)

- Here is a visualization of the ranking of the features according to their importance
- Hence strategies towards these features should be prioritized:
  - C_AGE
  - DVcCR
  - Asset value
  - gn_occ_encoded
  - INCM_TYP_encoded
  - C_EDU_encoded

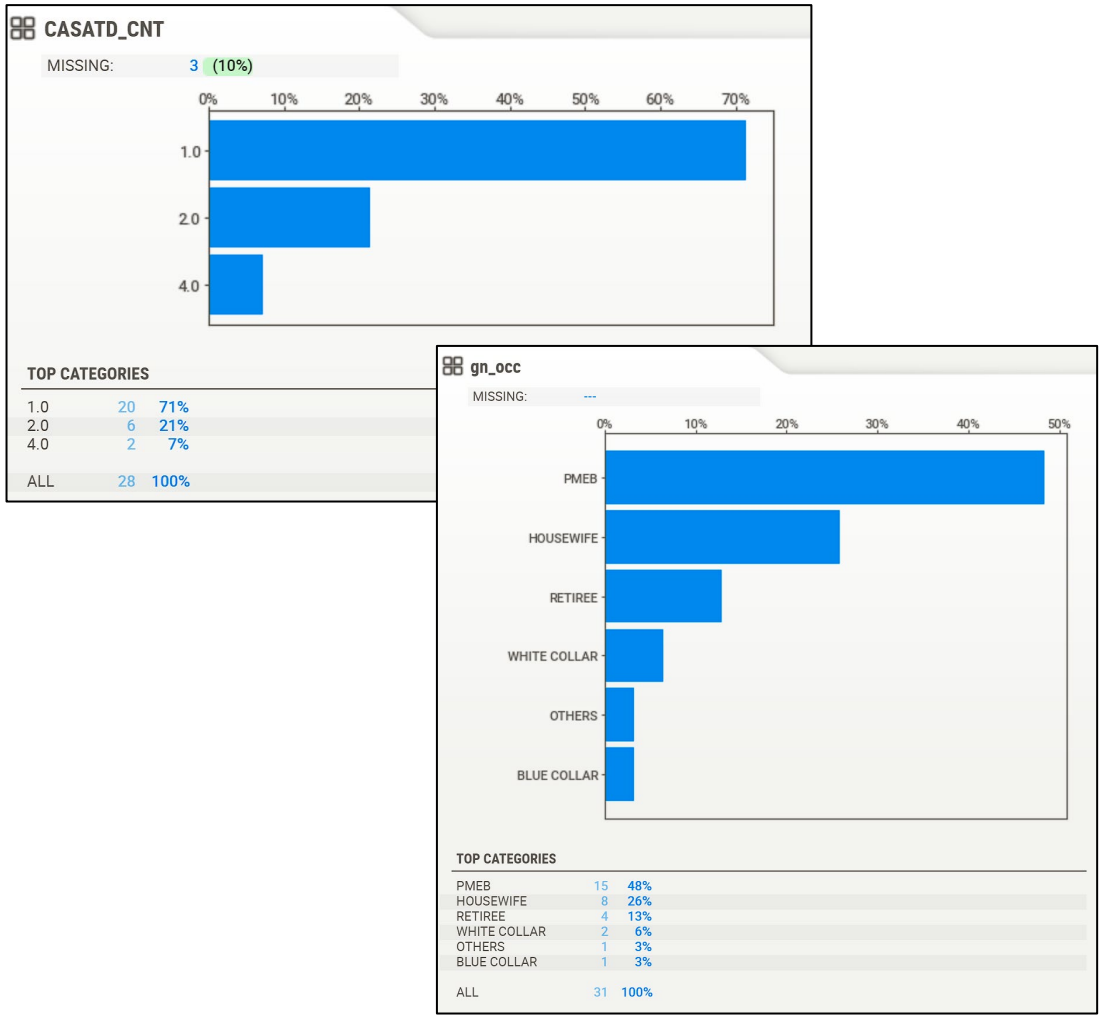


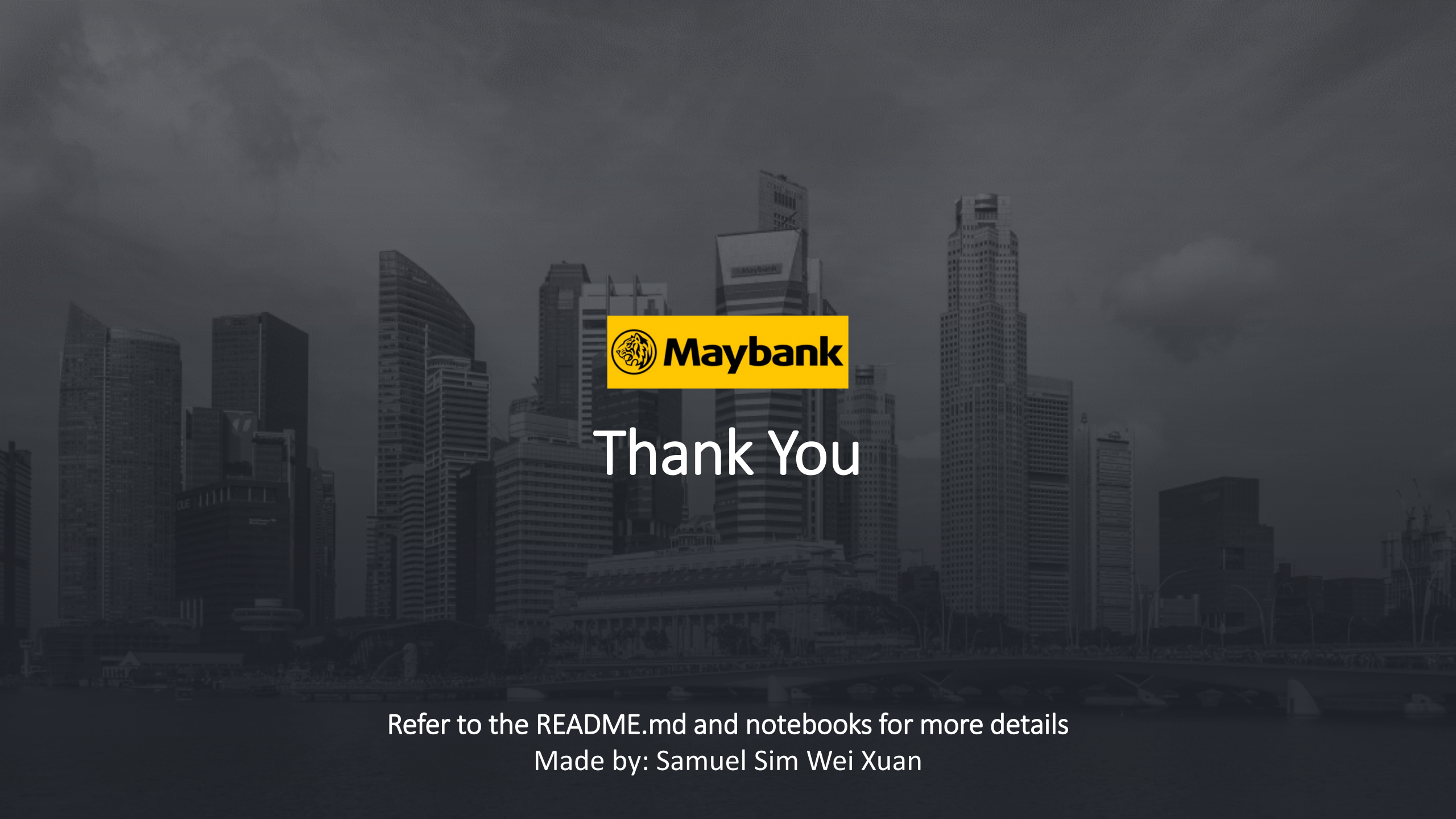
Analysis (Inference) (2/3)

- Recall on the entire provided dataset:
  - **9716** out of the **10,926** affluent customers classified correctly
- Among the **55,157** normal customers
  - **31** has been classified as affluent customers, these are our potential affluent customers
- Run EDA (Sweetviz) again on just these **31** potential affluent customers
- Clearly my results are not yet satisfactory, with only a recall of **89%**
  - If more time given, I would certainly increase the number of trials for Optuna
  - Redo the imputation with mean/mode which might drastically improve performace, but need consult the domain experts of the dataset
  - Tune the resampling process, as my model is over generalizing or too much data is lost from resampling
  - Adjust the folds for k fold validation

Analysis (Inference) (3/3)

- EDA (Sweetviz) again on just these **31** potential affluent customers
- **Points Noted:**
  - For example we know the occupation of these customers or their CASA and TD counts, and determine strategies for these customers
  - However need to sit down with domain experts to understand the data better and the insights that can be drawn from the EDA sweetviz results





Thank You

Refer to the README.md and notebooks for more details

Made by: Samuel Sim Wei Xuan