Individual Assignment 2

CoinPilot user segmentation, modelling, and lightweight deployment

---

Company overview

CoinPilot is a fictional FinTech startup based in Singapore that provides digital savings and micro-investment services for young professionals. The company helps users achieve their financial goals by offering mobile-first tools for portfolio planning, automated investing, and responsible financial tracking. Having surpassed 100,000 active users across Southeast Asia, CoinPilot is experiencing rapid growth, backed by venture capital, with a lean product and engineering team.

Business issues to address

CoinPilot's leadership has identified several pressing issues they want to better understand using data:

1. **User segmentation challenge**
   The company has a rapidly growing and diverse user base. They want to uncover distinct behavioural and financial patterns across users, so they can tailor communication, improve engagement, and design personalised savings or investment plans.
2. **Premium conversion prediction**
   Only a fraction of users upgrades to the paid "premium" tier. CoinPilot needs to predict which users are most likely to convert, and what behavioural or financial traits are most strongly associated with conversion. This will inform both marketing strategy and product features.
3. **Portfolio and engagement patterns**
   Users spread their money across equity, bonds, cash, and crypto. CoinPilot leadership wants to understand whether portfolio allocations, app engagement, and financial habits contribute to premium conversion and retention.
4. **Operationalisation via lightweight deployment**
   CoinPilot's engineering team is lean. They need a simple, working proof-of-concept service that can take in a user profile and return a premium conversion prediction, with an interface for product managers to test inputs without writing code.

Assignment brief

0) Setup cell to install packages

At the top of your notebook, include:

```
pip install numpy pandas scikit-learn xgboost matplotlib fastapi "pydantic<2"
uvicorn streamlit joblib requests
```

1) Data preparation and sanity checks (use the dataset provided)

The dataset has been provided to you. It includes demographics, engagement, portfolio allocations, and a binary target `converted_premium`.

CoinPilot dataset: Data dictionary

Column	Description	Role
age	Age of the user in years (21–45)	Feature
tenure_months	Length of time (in months) the user has been with CoinPilot (1–48)	Feature
income_monthly	Monthly income in Singapore dollars	Feature
savings_rate	Proportion of monthly income saved (0–1)	Feature
risk_score	Risk tolerance score (0–100)	Feature
app_opens_7d	Number of app opens in the past 7 days	Feature
sessions_7d	Number of app sessions in the past 7 days	Feature
avg_session_min	Average session duration in minutes	Feature
alerts_opt_in	User opted in for alerts (1 = Yes, 0 = No)	Feature
auto_invest	User enabled automated investment (1 = Yes, 0 = No)	Feature
country	User's country: SG, MY, ID, PH, TH, VN	Feature
equity_pct	Portfolio allocation to equities (%)	Feature
bond_pct	Portfolio allocation to bonds (%)	Feature
cash_pct	Portfolio allocation to cash (%)	Feature
crypto_pct	Portfolio allocation to cryptocurrencies (%)	Feature
converted_premium	Whether the user upgraded to premium (1 = Premium, 0 = Free)	**Target**

Your tasks in this section:

- Load the dataset into a Pandas DataFrame.
- Display descriptive statistics and feature distributions.
- Calculate and display the premium conversion rate overall and by country.
- Produce at least one simple chart (e.g., bar chart of conversion rate by country).

2) Unsupervised user segmentation

2.1 PCA with scree plot

- Scale the numeric features with `StandardScaler`.
- Apply PCA and plot both the scree plot and cumulative explained variance.

- Choose a suitable `n_components` and justify briefly.

2.2 t-SNE visual exploration

- Apply t-SNE to the scaled features.
- Plot the 2-D embedding coloured by clusters or country.

2.3 k-means clustering

- Use elbow, silhouette, and Davies Bouldin index to decide on k.
- Fit k-means and profile the clusters using group means of key features.

Your tasks in this section:

- Carry out PCA and produce the scree plot.
- Run t-SNE and produce a labelled scatterplot.
- Apply k-means with your chosen k and profile the clusters.
- Provide short markdown commentary on patterns you observe.

3) Ensemble modelling for premium conversion

3.1 Train and test split

- Perform a stratified train-test split on `converted_premium`.

3.2 Random Forest with OOB

- Train a Random Forest with `oob_score=True`.
- Report OOB score, accuracy, precision, recall, and ROC AUC.
- Plot feature importances.

3.3 AdaBoost

- Train AdaBoost with a decision stump base learner.
- Report the same metrics as above.

3.4 XGBoost

- Train an `XGBClassifier` with sensible parameters.
- Report the same metrics as above.

Your tasks in this section:

- Train and evaluate all three ensemble models.
- Compare performance across accuracy, precision, recall, and ROC AUC.
- Include at least one feature importance visualisation.

- Add brief markdown comments on which features stand out as strong predictors.

4) Stacking and model comparison

- Build a `StackingClassifier` combining two of your ensemble models.
- Compare all models (Random Forest, AdaBoost, XGBoost, Stacking) in a summary table.
- Produce a simple comparison plot of at least one metric (e.g., ROC AUC).
- Provide a recommendation on which model is most appropriate for deployment.

Your tasks in this section:

- Implement a StackingClassifier with two base models.
- Generate a results table showing all models with the same set of metrics.
- Create a comparison visualisation.
- Justify your chosen “best” model using your results.

5) FastAPI prediction service and Streamlit client

5.1 Pipeline and persistence

- Build a pipeline including numeric scaling and one-hot encoding for country.
- Save the trained pipeline with `joblib`.

5.2 FastAPI app

- Implement three endpoints:
  - GET /health
  - GET /info
  - POST /predict that returns both class and probability
- Use Pydantic models for request and response schemas.

5.3 Streamlit client

- Build a short form to collect inputs.
- Send inputs to the FastAPI predict endpoint with requests.
- Display class and probability returned.
- Implement an offline fallback to load the local pipeline if API is unreachable.

Your tasks in this section:

- Create and save the scikit-learn pipeline.
- Write and test the FastAPI endpoints.
- Develop a Streamlit client with online (API) and offline (local) prediction paths.
- Demonstrate at least one working prediction end-to-end.

6) Findings and recommendations

Your tasks in this section:

- Summarise in 6–10 bullet points what your analysis uncovered:
    - Key user segments from clustering
    - Best-performing model for conversion prediction
    - Business-relevant features influencing premium conversion
    - Clear next steps for CoinPilot's product and marketing team
- Keep findings grounded in your outputs (no speculation).

Submission Guidelines

- **Due Date:** **28 Sep 2025 (Sunday), 11:59pm**
- **Submission Method:** Upload single Jupyter notebook file
- **File Naming:** `name_Assignment2.ipynb`
- **File Size:** Ensure notebook runs completely and all outputs are saved
- **Dependencies:** Include installation commands for all required packages in your notebook
- **Late Policy:** 10% deduction for every 24 hours late. No submission allowed beyond 48 hours late.

Grading Rubric

Total: 100 points
Each section is graded with the scale required.

A. Data preparation [5 points]

Criteria	Excellent (5 pts)	Good (4 pts)	Satisfactory (3 pts)	Needs Improvement (0–2 pts)
Data preparation and sanity checks	Loads dataset correctly, shows clear descriptive stats, conversion rates overall and by country, includes at least one correct visual	Minor omissions in stats or missing one required output	Basic load with limited checks or unclear outputs	Fails to load or missing most required checks

B. Unsupervised user segmentation [10 points]

Criteria	Excellent (9–10 pts)	Good (7–8 pts)	Satisfactory (5–6 pts)	Needs Improvement (0–4 pts)
Data preparation and sanity checks	Loads dataset correctly, shows clear descriptive stats, conversion rates overall and by country, includes at least one correct visual	Minor omissions in stats or missing one required output	Basic load with limited checks or unclear outputs	Fails to load or missing most required checks
PCA and scree	Correct scaling, scree and cumulative plots, sound choice of `n_components` with justification	Correct plots but weak justification	Plots present but unclear choice or setup errors	Missing or incorrect PCA
t-SNE plot	Clear 2-D scatter, readable legend, sensible perplexity	Present but minor readability issues	Present but confusing or poorly labelled	Missing or incorrect t-SNE
Cluster profiling	Uses elbow, silhouette, and Davies Bouldin; profiles clusters meaningfully	Uses at least two criteria, reasonable profiling	Basic attempt with minimal justification	Poor method choice or no profiling

C. Ensemble modelling for premium conversion [25 points]

Criteria	Excellent (23–25 pts)	Good (18–22 pts)	Satisfactory (13–17 pts)	Needs Improvement (0–12 pts)
Random Forest with OOB	Correct OOB usage, complete metrics (accuracy, precision, recall, ROC AUC), insightful feature importance	Minor reporting or parameter gaps	Runs but incomplete metrics	Incorrect setup or missing outputs
AdaBoost	Correct shallow base learner, complete metrics	Mostly correct but some gaps	Basic run with limited metrics	Incorrect or missing
XGBoost	Configured sensibly, complete metrics	Minor tuning or reporting gaps	Minimal metrics	Missing or incorrect
Commentary	Clear markdown notes on standout features and patterns	Notes included but light	Minimal commentary	Missing commentary

D. Stacking and model comparison [30 points]

Criteria	Excellent (28–30 pts)	Good (22–27 pts)	Satisfactory (15–21 pts)	Needs Improvement (0–14 pts)
StackingClassifier	Correctly built with two valid base models, clean fit and prediction, metrics reported	Works with small issues	Basic attempt with partial results	Missing or incorrect
Comparison table and plot	Neat summary table and clear visual comparison	Present but slightly messy	Minimal or hard to interpret	Missing
Choice and justification	Concise, metric-based justification aligned with results	Acceptable but light justification	Superficial reasoning	No reasoning or inconsistent with results

E. FastAPI service and Streamlit client [20 points]

Criteria	Excellent (19–20 pts)	Good (15–18 pts)	Satisfactory (10–14 pts)	Needs Improvement (0–9 pts)
Pipeline and persistence	Proper pipeline with scaling and encoding, saved and reloaded cleanly	Minor pipeline issues	Basic save/load with gaps	Missing or broken
FastAPI endpoints	All three endpoints (/health, /info, /predict) work with Pydantic models, response returns class and probability	Mostly correct with small schema issues	Only basic predict endpoint working	Missing or invalid
Streamlit client and fallback	Usable form, successful API call, offline fallback implemented	Works but with minor usability issues	Basic call without fallback	Missing or non-functional

F. Findings and recommendations [10 points]

Criteria	Excellent (9–10 pts)	Good (7–8 pts)	Satisfactory (5–6 pts)	Needs Improvement (0–4 pts)
Synthesis of insights	6–10 concise, grounded bullet points covering clusters, best model, feature drivers, and next steps	Covers most points but some lack depth	Few points, partially supported by outputs	Missing, speculative, or unsupported