# Week agenda for the team

|  |  |
| --- | --- |
| Day 1 | Problem understanding, familiarization with data, insights into solution |
| Day 2 | Feature engineering , insights into modeling strategy, discussion at the end of day |
| Day 3 | Model building, discussion at the end of day |
| Day 4 | Model building, discussion at the end of day, summarizing and reporting |
| Day 5 | Final steps, presentation preparation and reporting |

# Users

Website: <https://old2025.ai-notebook.ktu.edu/hub/login2> ar <https://old2025.ai-notebook.ktu.edu/login2>

* altusr10 Udi7faigh92 Karolina
* altusr11 bei9eGeemo2 Kamilija
* altusr12 thood5Saep2 Callum
* altusr13 Lahpeibei12 Madhav
* altusr14 hiuc7voDie2
* altusr15 yuephohJ4a2

# Problem description

Goal - to generate several combinations of offers (plan + possibly VAS + price) for the client.

# Data explanation

* xls file with variables description

# Possible model building strategies for discussion

1. Ranking Plans by Suitability (Learning to Rank (LightGBM/XGBoost ranker))

To score/rank candidate plans based on how suitable they are for the user.

More: in each row of a training data set a **relevance score** (user-plan) should be assigned a relevance score based on how well the plan fits their behavior (usage, needs) and possible business goals (upsell opportunity, churn reduction). It could be from 3-highly relevant to 0-poor match (here historical upgrades/downgrades could be used as well; Alternatively , we could use clustering technique). For example,

A screenshot of a screen

AI-generated content may be incorrect.

where: user\_id: Unique user identifier, current\_plan\_id: What plan the user has now (ending soon), plan\_id: Candidate plan we're evaluating for upsell, avg\_watch\_hrs\_day: Daily viewing time, genre\_pref\_\*: Genre usage ratios (e.g., 60% sports), internet\_speed\_avg: Measured average internet speed, income\_bracket: Income classification (Low, Medium, High), plan\_price: Monthly cost of the candidate plan, plan\_quality: Video resolution tier (SD, HD, 4K), includes\_sports/kids: Binary flags for plan features, is\_upgrade: Whether the plan is higher-tier than the current one, price\_diff: Difference in price vs. current plan, …, promotions: Could be discounts, trials for few months, relevance\_score: How suitable the plan is for the user (label)

Then, to build a model , which ranks all possible plans for each user. For this purpose, LightGBM ranker, XGBoost ranker, LambdaMART could be used

A close-up of words

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

Šitas variantas turėtų tiktų ir naujiems planams įvesti.

1. Customer Segmentation (Clustering)

We create many features, but maybe with some expanding time window to include changes in customer behaviour. When clusters are created, we could apply clustering .

A screenshot of a computer

AI-generated content may be incorrect.

Also, could be the case, that after the clusters are created , each of them could be clustered again to assign the price level.

Technniques to be applied: typical clustering techniques (k-means, hierarchical, dbscan).

Alternatively, content-based filtering could be used to match user profile to plan features (unsupervised similarity). But in this case, for introduction of new plans/prices it wouldn’t work straightforward.

Also, autoencoders / deep embeddings could be used to learn compressed latent representations of users and plans, then compute similarity. These representations could be used in clustering and also in learn-to-rank

1. Multi-class supervised

Optionally, we could create a relevance score using some rule-based scoring logic

|  |
| --- |
| def compute\_relevance(user, plan):  score = 0  # 1. Usage-based relevance  if user['avg\_watch\_hours\_per\_day'] >= 4 and plan['plan\_quality'] == '4K':  score += 1  # 2. Genre match  if plan['includes\_sports'] and user['genre\_pref\_sports'] > 0.4:  score += 1  if plan['includes\_kids'] and user['genre\_pref\_kids'] > 0.3:  score += 1  # 3. Internet fit  if plan['plan\_quality'] == '4K' and user['internet\_speed\_avg'] < 25:  score -= 1 # not suitable  # 4. Price/Income compatibility  if plan['plan\_price'] > 25 and user['income\_bracket'] == 'Low':  score -= 1 # too expensive  # 5. Upgrade bonus  if plan['tier'] > user['current\_tier']:  score += 1  # Clamp score between 0 and 3  return max(0, min(score, 3)) |

A screenshot of a computer

AI-generated content may be incorrect.

It could be also a weighted scoring example

|  |
| --- |
| relevance = (0.4 \* content\_match) + (0.3 \* price\_fit) + (0.3 \* usage\_match) |