Analytics Cup 2023/2024

What do we cook today? - Developing a Recommender Model for Recipes generated by a Large Language Model.

The Challenge

In this year's Analytics Cup, you are part of the newly founded start-up *LLMeals*. The business model of this organization is to develop and leverage large language models to suggest new recipes to their customers based on a monthly fee. Within the first few months of the launch, your team observes that many customers cancel their subscriptions after the test period. To understand this quick turnover rate, your organization surveyed the quality of services.

The evaluation of this study revealed that the users perceive the suggested recipes' quality as very high. However, they do not fit the specific requirements of the users. For example, in some instances, the diet option of the recipe did not match the customer's diet. In other cases, the users were looking for breakfast recipes that could be prepared quickly but received dinner recipes where the preparation was elaborate and complex. To incorporate this feedback, you decided to develop a predictive model that classifies recipes into recipes that a user potentially likes or dislikes. Additionally, your team created a user interface where customers can enter information to filter relevant recipes. It provides options to determine the cooking time and macronutrients.

This new interface enabled your team to collect additional data about the users' search behavior. Since you wanted to integrate the feedback as quickly as possible, you did not incorporate the interface into your database pipeline so that you could extract the relevant data from different sources. Overall, four data sets can be leveraged to develop the model. The first data set is the "recipes.csv" file, which contains 75,604 recipes generated by the LLM and suggested to the users. The recipes can be uniquely identified by the entry in the column *RecipId*. Note we consider a meal vegan if it does not contain any animal products and vegetarian if it does not have any meat or fish. The file "reviews.csv" includes the results of a short survey displayed to the users after suggesting new recipes. They could evaluate the difficulty of the recipe in a range from 1 (easy) to 5 (very difficult) and whether they like the recipe. The file contains 140,195 reviews. In their user profiles, the customers could indicate their dietary preferences. An extract of this is shown in the "diet.csv" file, which contains 271,907 entries. Finally, the "requests.csv" table includes the users' search filter to find new recipes. This table depicts the newly developed interface mentioned before. In the future, it should serve as the input to the recipe LLM. Overall, you obtained 140,195 requests.

Your model will build the basis for generating additional training data for the recipe LLM so that the overall service quality will be improved.

YOUR TASK: Use these data sets to develop a model that can predict the outcome of the column *Like*, which indicates whether a customer likes a recipe (your model predicts *Like*=1) or dislikes it (your model predicts *Like*=0).

For reviews with a *TestSetId* and for which you are not given the *Like* (the "*private test set*"), you must make predictions that will form your submission. See the "submission_template.csv" for details about the required format. Your project will be evaluated and graded based on these predictions.

Evaluation

Your predictions will be evaluated based on the performance measure of *balanced accuracy* – the arithmetic mean of Sensitivity and Specificity – that your submission achieves on the private test set.

		Truth (an offer was accepted)	
Your prediction		YES	NO
	YES	True Positive	False Positive
	NO	False Negative	True Negative
		Sensitivity = True Positive Rate = $\frac{TP}{TP+FN}$	Specificity = True Negative Rate = $\frac{TN}{FP+TN}$
		Balanced Accuracy = BAC = $\frac{Sensitivity+Specificity}{2}$	

The Data

diet.csv

Column	Description
AuthorId	(String, primary key) A unique identifier for the customer.
Diet	(String) The diet option of a user.
Age	(Numeric) The age of the corresponding user.

recipes.csv

Column	Description
RecipeId	(Numeric, primary key) The unique identifier of a recipe.
Name	(String) The name of the recipe.
CookTime	(Numeric) The time it needs to cook the recipe.
PrepTime	(Numeric) The time it needs to prepare the recipe.
RecipeCategory	(String) A label indicating the type of the recipe.
RecipeIngredientQuantities	(String) A label that represents the number of ingredients in the recipe.
RecipeIngredientParts	(String) A label that contains the ingredients in the recipe.
Calories FatContent SaturatedFatContent CarbohydrateContent FiberContent SugarContent ProteinContent	(Numeric) The amount of Macronutrients in kcal/ g per 100g.
SodiumContent CholesterolContent	(Numeric) Nutrition facts in mg per 100g
RecipeServings	(Numeric) The number of servings that result from this recipe.
RecipeYield	(String) The number of servings that result from this recipe.

reviews.csv

Column	Description
AuthorId RecipeId	(String, foreign key) The unique identifier of a customer and a recipe.
Rating	(Numeric) A rating between 1 and 5 that the users could submit to evaluate the difficulty of the recipe.
Like	(Numeric) A label that indicates whether a user likes a recipe (=1) or not (=0).
TestSetId	(Numeric) The unique identifier in the test set.

requests.csv

Column	Description
AuthorId RecipeId	(String, Foreign keys) The unique identifiers of a user and a recipe.
Time	(Numeric) The duration a recipe should take at most (including the time reserved for the preparation and cooking).
HighCalories	(String) A flag that indicates whether the resulting meal/ beverage should have a high number of calories.
LowFat	(Numkleric) A flag that indicates whether the resulting meal/ beverage should include a low amount of fat.
HighProtein	(String) A flag indicating whether the resulting meal/ beverage has a high amount of proteins.
LowSugar	(String) A flag indicates whether the resulting meal/beverage has low sugar.
HighFiber	(Numeric) A flag indicates whether the resulting meal/beverage has high fiber.

Submission Rules

Submissions

A valid submission contains a csv-file containing predictions and a script that generates these predictions from your given data. Your submitted script **must be self-contained** and reproducible; more on that below. Your prediction file will be graded automatically and judged based on the performance measure of **balanced accuracy** it achieves on the test set. Your team can make **up to 10** valid submissions. Note that invalid submissions do not count to this limit. Only the **best valid submission** from your team will be evaluated for grading.

We have provided you with a sample submission file (with entirely random predictions), which you can use to check whether the format of your generated submission is correct.

Make sure that all submissions adhere to the following **naming scheme**. This ensures that you keep track of your files as a team, which is especially important if we invite you to a clarification meeting.

Prediction-File: *predictions_group_name_number.csv*
Script: *script_group_name_number.py*

Prohibitions

The following things are strictly prohibited and will result in disqualification:

- You may **NOT** hard-code predictions for any instances in the test set. All predictions must be based on your model output.
  This applies both to individual predictions (i.e., **forbidden**: prediction[TestSetId==201] = 1) as well as to fixed rules (**forbidden**: prediction[CUSTOMER==5] = 0).
  Note: Hard-coding **features** to be used in the model is generally allowed.
- You may **NOT** work together with other teams. If we find that you copied work or cooperated, both teams will be disqualified.
- You may **NOT** use fully automated machine learning approaches. For instance, using an already implemented cross-validation from sklearn is fine. Using packages that automate large parts of the pipeline, such as H2O is not allowed.

If you need clarification on whether something is allowed or not, please reach out to us or ask in the Moodle forum! In cases of ambiguities, we reserve final judgment on whether a given submission violates the rules above!

Reproducibility

All submissions must be **reproducible**, i.e., the submitted Python script must reproduce the same prediction file, even when run on a different machine at a different time. To ensure this, your scripts should (at least) follow the following guidelines:

- Import all packages that you use **at the very top** of the file.
- At the top of your script, right after the imports, set all relevant seeds to **'2024'** to seed random number generators (rerunning the script will then give you the same results in random operations). Some machine learning packages manage their random number generator not managed by Python. If you use such packages, set the seed in the same manner.
- Do NOT change the file names of the training and test data sets. Your script should `read` the files (and write submissions) from/to **its own directory**. (i.e., `pd.read_csv('customers.csv')`, rather than `pd.read_csv('C:/Users/name/my_files/more_directories/I_renamed_the_customer_file.csv)'
- Do NOT modify the content of the data files provided. All data preparation should happen WITHIN the provided script.
  You may want to save intermediate results that took a long time to generate (data, models, etc.) to disk and reread them. That is fine for prototyping but not for the final script you submit.

The following last point will not be handled as strictly, but you should nevertheless adhere to it:

- Your submitted script should be a (reasonably) minimal implementation to generate your model. We don't expect you to spend any time optimizing this, but please use good judgment to avoid unnecessary computation in evaluation.
  Example 1: *You performed a hyper-parameter search that took three days to run to find your perfect model. Your submitted script should only train your final model using the (hard-coded) final hyperparameters you found. Don't include the search in your file. In such a case, add a short comment about how you arrived at the hyperparameters (or comment out the code for the search)*
  Example 2: *You trained 20 models and decided on your favorite one to create a submission at the end. Your submitted script should only trigger training of your favorite model, not all 20. (Delete or comment out the code for the other models in your submission.)*
- Although good solutions should be possible in <<10 min runtime on modest hardware (e.g., 5-year-old laptops), some groups might have models that take longer to train. If your script takes a long time to run, please include a comment at the top containing the approximate runtime and your computer's information. (e.g., `#1.5 hours on a dual-core laptop with 4GB RAM`).

If your submitted solution is not reproducible, we reserve the right to disqualify your team from the Analytics Cup.

Frequently Asked Questions

Additional External Data

You might want to consider including external data that is not part of the data set but might be valuable for predicting. This is generally allowed, but you must ensure your submission stays reproducible. Thus, load the data directly via the URL in your script or include it with the *dput* command (R) or pickle command (Python). Do not store any additional data states on your disk.

Languages other than Python

Some students have asked whether they may use languages other than Python (such as R) for the Analytics Cup. This is permitted in general, but we cannot provide you with any support, and you must adhere to the same standards of reproducibility found above.
If you want to use R, you must submit a single, self-contained R script.
Please contact us beforehand if you want to use any language other than R or Python.

Jupyter and Rmd-Notebooks

Some students prefer writing code in Rmd or Jupyter notebooks rather than flat .py scripts.
Submissions uploaded as notebooks are generally accepted but must adhere to the same Reproducibility requirements outlined above. Especially the cells in your notebook must run **in order**, **from top to bottom**. They should not contain additional exploratory analysis steps, particularly those
that take long to run. For submissions uploaded as .ipynb files, you **must** strip all cell output from the file before submitting.

Cloud Services, Google Colab, etc.

The challenge is designed to run fairly comfortably on modest hardware (e.g., 5 year old laptops with ~8GB RAM and dual-core processors). If you want to use cloud platforms to build your models, you may do so, but you must submit self-contained scripts that can run on our local machine. (Instead, you may **not** submit a link to a repository, etc.)