DSApps 2023 @ TAU: Final Project Snack Classification

Snacks!

Yoni Slutzky Itay Ofer

2023-08-23

Welcome

Welcome to the modelling chapter of our final project. In this chapter, we will go into detail and describe our work towards crafting the snack classification models. For the actual code, we refer the reader to the attached .Rmd files .
Our method of modelling included three main parts, with each part including several steps. We will describe for each part the work done, and to which of the models it applies.
Our work consists of 4 total models:
- The first one, model 0, was used as a benchmark model based only on the tabular data and not used for prediction.
- The remaining models, models $1, 2, 3$, were based on model 0 and extended using the image data as-well. These models were the ones used for prediction.

Part 1 - feature engineering

Part 1 is the primary part of our work, and is also the one who took the longest. For this part, we used 60% of the training data. This part was split into 5 feature-engineering steps which focused on the tabular data, and a single feature-engineering step which focused on the image data. We will begin by detailing the steps relevant for the tabular data.

The first step was to unify the tabular data which addresses the nutrient information with the tabular data which addresses the actual snacks. The rational for adding these features was some intuition we gained from the EDA chapter of this project, where we saw that different categories differ by their nutrient data.
We performed this by first combining the food_nutrients table with the nutrient table. This created a table which included for each snack, all of its nutrients' data. Next, we combined the created table with the original food_train table such that now all snacks had columns for each of their nutrients.
By doing this step, we added the relevant nutrients as numerical features, where a nutrient that didn't appear at all wasn't included, and a snack with a missing nutrient was imputed with zeros.
Following this, we created a function that chooses the top k appearing nutrients, in the sense of average frequency (similar to the EDA chapter). All models took the $k = 12$ top appearing nutrients, as we didn't record significant changes when increasing this hyperparameter any further.

The second step was to extract features from the household_serving_fulltext variable. This variable includes the serving size for household measurement, and we believed it carried a signal for the identity of the snack's category.
We performed this by first cleaning some of the text. Specifically, we removed punctuation and generic serving sizes (which do not carry any significant signal). Then, we filtered words which have higher frequencies and created binary features for their appearance in this variable.
In models 0 and 1, the minimal frequency was 50, and included more than 90% of the appearances. In

models 2 and 3, the minimal frequency was 15, and included more than 95% of the appearances.

The third step was to extract features from the brand variable. This variable included some signal towards the category for a significant amount of snacks. However, the top appearing words were actually not indicative. Thus, we chose some common-sense words and created binary features for their appearance in this variable. This step was performed identically for all of the models.

The fourth step was to extract features from the description variable. This variable includes a great deal of signal towards the category for nearly all of the snacks. We performed this by first cleaning some of the text. Specifically, we removed punctuation and frequent words that do not indicate on the category. Then, we filtered words which have higher frequencies and created binary features for their appearance in this variable. In models 0 and 1, the minimal frequency was 50, and included more than 70% of the appearances. In models 2 and 3, the minimal frequency was also 50, but we decided to also include some common-sense words which didn't passed the threshold.

The fifth step was to extract features from the ingredients variable. This variable also includes a great deal of signal towards the category for nearly all of the snacks. We performed this by first cleaning some of the text. Specifically, we removed punctuation and standardized the words to be as neat as possible. Then, we filtered words which have higher frequencies and created binary features for their appearance in this variable.
In models 0 and 1, the minimal frequency was 50, and included more than 85% of the appearances. In models 2 and 3, we performed extra standardization and lowered the minimal frequency to 30. This resulted in inclusion of more than 90% of the appearances.

We will now describe the feature-engineering of the image data. This part was performed on Google Collab. We decided to fine-tune an existing model to our data. We used MobileNetV2, which is a relatively lightweight NN model featured on Keras Apps. In this step, we ignored the data partition we made earlier (for simplicity) and decided to withstand the occurring data-leakage.
We split the training data into 90% training and 10% validation. In training, we only augmented the data by a horizontal flip (for computation reasons). The only further augmentation made was the application of the preprocess function that is built-in with the MobileNetV2 model.
We chose to freeze the 130 first layers of the model, and to tune the latter ones. The performance on the validation set was almost 70% accuracy. We decided to not re-train on all of the training data to avoid overfitting (since we already had the data-leakage in place).
We proceeded to predict on all of the data (training and test). Model 1 took the top 1 prediction. Models 2 and 3 took the actual sigmoid output for each of the categories, as-well as the ranks of the categories' sigmoid output. The latter features were included to give the GBT we trained later better flexibility.
We want to point out the long training times required for fine-tuning. Even when using the GPU option from Google Collab, training times were in the hours and required us to be flexible and fit our requirements from the model.
The Google Collab notebook, as-well as the actual finetuned Keras model are added to the final submitted work.

Part 2 - model finetuning

The model family of choice was gradient boosted trees. This was a natural selection as our engineered data was tabular and had many binary and categorical features - the perfect setting for decision trees. We used 30% of the training data to fine-tune the model, using 5-fold cross-validation.
In our tuning process we concluded that the best hyperparameters to be used are $lr = 0.01$ and $sample_size = 0.75$. Due to long training times, we tuned the amount of trees across $\{25, 75, 100\}$, with 100 being the clear favorite.
This fine-tuning process was actually performed only in model 0, where subsequent models used the best hyperparameters. Model 1 used 100 trees, model 2 used 300 trees, and model 3 used 1000 trees. An important note is that the training times were also in the hours, again requiring us to be efficient and flexible.

Part 3 - model validation

To test our models, we used 10% of the training data. The models tested were trained on all training data besides the validation set. The first metric we examined was validation accuracy. Model 0 (without any image data) reached $\sim$ 91% validation accuracy. Model 1 reached 94.7% validation accuracy, model 2 reached 95.1% validation accuracy and model 3 reached 96% validation accuracy.
In order to be realistic, we also crafted a benchmark score which took the proportion between the validation accuracies of our model and of the benchmark model presented in class, multiplied by 0.6 which was our conservative estimation of the test accuracy on the benchmark model. This is due to a mention in class that test accuracy on the benchmark model was more than 60%.

Summary and final thoughts

Overall, our models reached excellent results (at-least in our opinion). We tried to make the best of all of the data at hand. We would like to point out a few insights we came across during our work, which may be the causes for our results not being even better.
The first important and main issue with the data is that a small subset of the training observations are classified with a misleading category. The classification of the data for some snacks is very misleading and not on par with the rest of the classified category (for instance, plain chocolate being tagged as candy, nut bar being tagged as chocolate, etc.). We believe this issue is typical when treating human generated data as in our case, and we are happy we can say we some-what overcame it. Another small issue is that the image data can also be misleading for some images, as it includes a non-negligible amount of non-relevant images. However, they still provide some signal which helped get better results.

In our work, we did our best to utilize all of the data sources, and to apply all relevant tools we were taught in the course. If given time, we believe this work can be improved even further. One example of future work is the application of LLMs to the textual variables (description, ingredients etc.).