Datamufe - Project Report
The Right Price

Kenil Tanna kyt237 | Thibault Fevry tjf324 | Victor de Murat vdm245

"The creamy one" @ *Le 23 Clauzel*
Predicted 14.3€, True 15€

"Scallop carpacio" @ *Le train bleu*
Predicted 28.7€, True 35€

Contents

1 Business Understanding

Restaurants are hard: 60% of restaurants fail within their first year, 80% within five years. These failure rates are much higher than most industries. Restaurants owners have to find the perfect mix of location, pricing, food quality, advertising, suppliers and timing to survive. Often, that ends up being a daunting task for people that might have little business training.
With that in mind, we set to tackle the pricing issue. More precisely, we wanted to predict the price of specific items, given their content and restaurant-level data (location, kind of restaurants, etc.). For this, we used data from lafourchette.com (the fork, now acquired by TripAdvisor), which features restaurants that generally have existed for more than one year (thus being comparatively successful). We trained a model to be able to predict menu item prices from this information, which can then be used to price new restaurants items.

For new restaurants owners, such a solution will be useful to assess competition prices and to define a price range for each of their menu items. Crazy as it may seem, how much margin you are making on beer versus beef bourguignon can be a matter of life or death for French "bistros" in the 8^{th} borough of Paris. For websites like lafourchette, there are plenty of advantages to such a solution:

- Capturing new restaurants earlier in the funnel by offering this pricing feature
- Upselling this solution to existing customers that want to assess their competitiveness
- Helping the restaurant industry thrive (and particularly those on Lafourchette), which helps Lafourchette's business

Therefore, this represents a win-win opportunity for both new restaurants and lafourchette's existing customers.

So why did we use Lafourchette instead of other providers such as Yelp or Opentable? Lafourchette has a unique data edge in Paris, which makes our pricing solution much more accurate:

- Lafourchette features more than 5000 restaurants in Paris, which translates to more than 50% of the market[1]
- Lafourchette has menu accuracy and completeness is unequaled

With that in mind, the first step was to be able to recover this data, which we did by scraping.

2 Data Understanding

2.1 Data collection & Extracted features

To predict the price, we had to scrape data about the restaurant, their menu items, tags, etc. As stated earlier, Lafourchette's data quality made it an ideal choice.

In scraping the data, we first had to write a scraper that recovered every restaurant on the website and then scraped each of them one by one. The site being dynamic (it used AJAX and javascript to load progressively), simple HTTP requests were not good enough and we had to simulate the behavior of a browser to be able to retrieve the data. For this process, we used the *selenium*[2] library and to parse the HTML tree we used *beautifulsoup*[3].

We were able to retrieve all the restaurant pages by putting an empty restaurant query and browsing through each of the 200+ pages to recover 25 restaurants per page, which led to ≈ 6550 restaurants. Then, each restaurant page has general information about the restaurant (tags, estimated price of a one-person meal, tags, location, etc) as well as information on each menu item (description and price). We wrote a parser class that processes each restaurant page, retrieving and caching all the menu items. Because for some restaurants the data is incomplete or they have special offers (i.e buffets instead of individual item prices), we ended up with ≈ 4600 parsed restaurants.

We then export this to a csv file where each row is a menu item, ending up with $\approx 80\,000$ rows (so nearly 20 items per restaurant).

For each restaurant/item, we extracted

- Price of the menu item (our target)
- Description of the menu item
- Location of the restaurant
- Tags of the restaurant
- Average estimated restaurant price

Of all these features, the only controversial choice is to include average estimated restaurant price as a feature. Indeed, this might be seen as a weak form of data leakage as the total restaurant price is roughly a function of the items. However, our goal is for a restaurant that knows its general price range to be able to price its individual items and assess the competition at the same price range. Therefore, the restaurant would have access to this information in the pricing process, so that it is not a issue to use it as a feature.

2.2 Selection bias

Although Lafourchette.com covers a wide range of restaurants, there is still some bias to our selection process. Particularly, some restaurants which are more old-school and refuse discriminative pricing and restaurants which are always full anyways (e.g the emblematic *Le Bouillon Chartier* or the hip restaurants *Big Mama*) tend not to be on the platform.

3 Data Preparation and Analysis

3.1 Target variable & Feature engineering

The target variable is the menu item price. As such, it is a numeric, continuous and positive variable.
An instance of data in our baseline problem is composed of menu item, location, tags, restaurantAveragePrice, target=price. Here is a summary of the main transformations that we performed:

- Deleting lines with negative prices (special offers) and lines with prices over 200 (outliers, comma misplacing and items not very relevant for analysis, like "champagne").
- Transforming the "location" feature (neighborhood name) to zip codes only, attributing 0 in the ≈ 200 instances were zip code was missing
- Handling restaurants' tags: we collected the 25 most common tags and create a binary feature for each of them (one-hot encoding), plus a discrete numeric feature for the number of tags for the restaurants.
- Handling menu items: as it is the most important step, we tried several methods, including 1-hot representation with TF-IDF for the X most common words and training continuous word embeddings with a deep learning model (GloVe[4], a variant of Word2Vec) and getting the menu item embedding by taking the mean of its words' embeddings. The later replicates the famous Document2Vec[5]. We started with X = 250.

3.2 Main descriptive statistics

First, the distribution of the menu items' prices (for prices inferior to 100) is presented in Figure 1. As expected, it is highly concentrated around 10 and has an exponential shape. One can notice a high number of items with prices inferior to 10: this is because we kept the drinks/sides.

The "location" feature expected importance can also be viewed in Figure 1. Disparities exist between neighborhoods, and are interestingly not negligible for some of them. Particularly, the 8^{th} borough (75008), which hosts most investment banks, consulting firms and law firms is more expensive than its

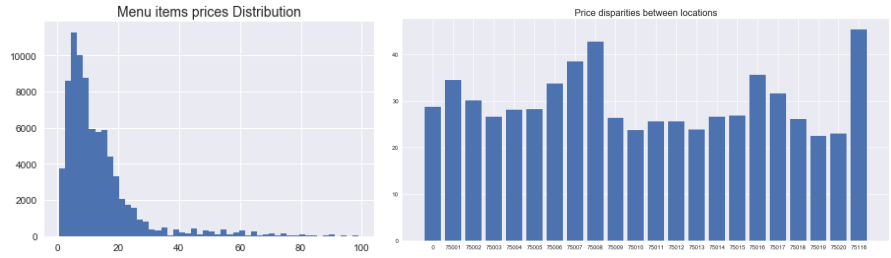


Figure 1: Visualization - Distribution of the prices of the menu items (left). Disparities between neighborhoods in terms of prices (right).

counterparts.
Other interesting statistics that are important for evaluation:

- 72% of the menu items have less than 7 words, so taking the mean of the embeddings of the most common ones should be a "not too bad" representation
- Standard deviation of the prices inside a restaurant : 11.3€. Given we use the mean absolute error, a good model would have an error less than that. Note however that because restaurants have a different number of items this is not the error a baseline model (that predicts the mean for every restaurant) would get. Also, given we train on different restaurants than test (with a custom cross-validation function), this is something we do not have access to.

4 Modeling & Evaluation

4.1 Choices of algorithms

The task at hand is regression. As a metric, we chose the sum of the absolute distances (mean average error, or l_1-distance, or Manhattan distance). The issue with the very common least squares is that it would have overweighted even more expensive restaurants (because square errors would be huge) when those clearly do not represent the majority of Lafourchette's core clients. Using cross-validation enables to have robust estimates, where we report both the mean score as well as the standard deviation over a 5-fold cross-validation.

We described quickly the feature engineering process, but it is worth going a bit more into details for the GloVe method. It is an unsupervised method relying on the distributional hypothesis, which consists in saying that similar words appear in similar context. This hypothesis motivated either distributed or distributional word representations, and Glove achieves to make both worlds meet. Indeed, the loss discriminates both on the words/contexts similarities

(cosine similarity) and on the number of co-ocurrences word/context.

For TF-IDF we tried with dimension size of 250 and 2500. For Glove embeddings, we first experimented with a vocabulary size of 250, without French stop words and with an embedding size of 20. We also tried changing the vocabulary size to both 2500 and 5000, the embedding dimension to 50 and using French stop words. The reasoning behind using vocabulary sizes and embedding dimensions that are much lower than in the GloVe paper is that our domain is very limited and we have limited data (food menu items) whereas the GloVe paper spans the whole English language and uses huge corpora.

Given our task needed to use a wide range of features in a non-linear fashion, we relied mostly on tree-based methods. Linear regression performance was very lackluster, even after several transformations. Nearest neighbor was effective for simple items (such as "water", "chocolate mousse") but made big mistakes on more complex ones (such as "Gambas linguine with garlic and olive oil").

- **Random Forests:** Random forests are an ensemble learning method which operates by building a multitude of decision trees. The training algorithm applies the bagging algorithm with decision trees as weak learners. More particularly, random forests use a modified tree learning algorithm that at each candidate split in the learning process, choose a random subset of features. We use the sklearn implementation of Random Forests. For our purpose we try different number of trees, we try the number of estimators as 10, 20, 35, 50, 100, 150, 400, as well as different options for max_depth.

- **Gradient Boosting Regressor:** Gradient boosting algorithm is an ensemble model of weak learners such as decision trees. The algorithm is comprised of three parts: a loss function to minimize, a weak learner to make predictions and an additive model to add weak learners to minimize the loss. Here the weak learners used are decision trees. In the additive model, trees are added one at a time and the existing trees are not changed. A gradient descent method is used to minimize the loss while selecting which tree to add. Here again, we use the sklearn version of Gradient Boosting Regressor. For our purpose we try different number of trees in the training process, we try the number of estimators as 10, 20, 35, 50, 100, 150, 400.

- **XGBoost:** XGBoost stands for extreme gradient boosting, which provides gradient boosting, stochastic gradient boosting, and also regularized gradient boosting. It is focused on computational speed and model performance, often outperforming sklearn's gradient boosting regressor in practice.

- **Ensemble:** As a final model, we tried an ensemble model which is an average of the predictions returned by the models above. The idea behind it is that each model will make different errors and thus averaging will tend

to average out some of the errors (as long as they are not perfectly correlated). There are more complex ensembling schemes (such as weighted averages, model stacking, model blending, etc.) that would likely have yielded even more improvements on accuracy but as our focus was not a kaggle-like quest for accuracy but rather practicality, we chose not to delve further into those.

To test each model, we wrote our cross validation function, that does a random split among restaurants (rather than just items) to ensure there is no data leakage. We then used 5-fold cross-validation to estimate the error and standard deviation of our models. For all tested algorithms, standard deviation over scores is quite low so that we are quite confident in our results.

For our baseline model, we wanted to see what a model that did not use the text itself would be able to achieve. Therefore, we trained a random forest model on restaurant data and number of words in the item description. This very crude baseline enables us to highlight the improvements that word level features bring as well as why this is useful to restaurants.

4.2 Results

As a reminder, the mean is the average of the 5 mean average errors of price predictions, RF stands for Random Forest and GB for Gradient Boosted Trees (XGBoost implementation). Word features used refer to how item description was encoded. GloVe was run with French stop words removed, an embedding dimension of 50 and a vocabulary size of 5000.

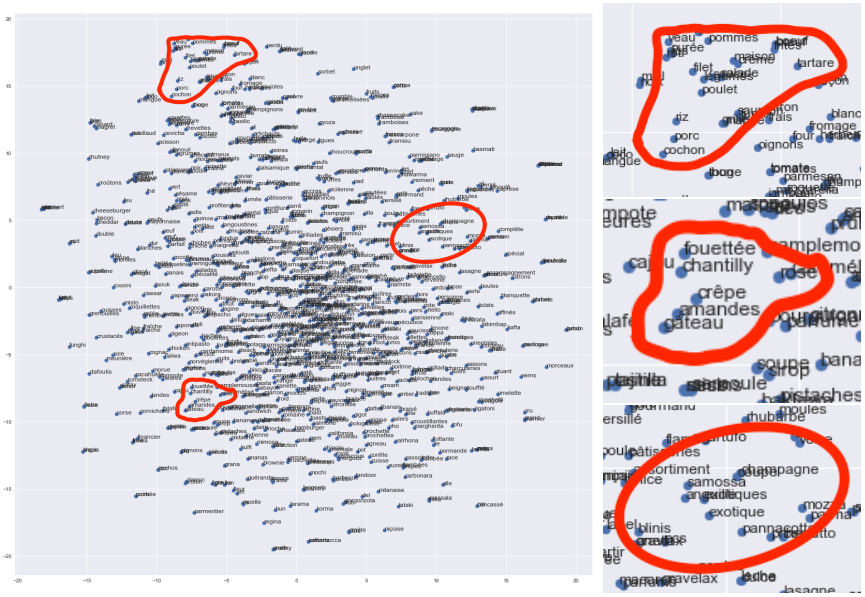
Model with hyperparameters	**Word features used**	**Mean**	**Std**
Baseline	Only text length	7.65	n.a
Non forest based*	Various	5+	n.a
RF[100 estimators]	TFIDF of dim 2500	**3.53**	0.14
GB[100 estimators]	TFIDF of dim 2500	3.62	0.09
RF[400 estimators]	GloVe with dim=50 vocab=5000	**3.57**	0.12
GB[400 estimators]	GloVe with dim=50 vocab=5000	3.64	0.08
Ensemble	GloVe with dim=50 vocab=5000	**3.5**	0.11

* Non forest based methods include linear regression and nearest neighbors.

We can see from our model that forest-based methods seem to outperform their counterparts but perform similarly, even with a lot of parameter tuning. Both GloVe and TFIDF improve our model considerably and perform quite similarly, with the ensemble model outperforming all others.

4.3 Embedding visualization

The visualization of 1000 words out of the 5000 words using the glove embeddings and the French stop words for is shown in Figure 2. The first and second



(a) Word embeddings represented in 2D

(b) Zooms

Figure 2: 2D-Representation of the first 1000 words. In (a) is the full representation, whereas in (b) zooms on interesting regions are presented.

ones present situations where embeddings are coherent : meats are close with each other (beef, pork, pig, chicken, veal), same for desserts (cake, crepe, Chantilly (cream), caramel, almond). The last one shows the major limit of our embeddings: isolated words (one-word description) cannot be learned correctly samoussa, champagne, pannacotta.
A possible improvement due to this insight would be to use embeddings when there are many words and TFIDF when they are few. However, this requires to have both at inference time, which is not very practical.

4.4 Model's ability to solve the problem

Our model seems to deliver good performance overall. An error of 3.5€ on average is low when we consider that the standard deviation of prices inside a restaurant is more than 11€. However, the model should be A/B tested properly to ensure that its performance is good enough for customers in practice. Our feeling is that such a model would not be that useful for pricing drinks but would be quite useful for plates at restaurants in the 30-70€ range.

5 Deployment

5.1 Concrete deployment in LaFourchette's system

Although ensemble methods are the best-performing model, they are impractical as their performance increase are likely not worth the added complexity in implementation, the computational cost and the loss of interpretability. In fact, we explained that this model should likely be deployed by Lafourchette as an additional feature on their website.
Given our embeddings are quite lightweight (<5Mb) and random forest performance, this would be the solution we would suggest to Lafourchette.
A wise approach would start by A-B testing the model to ensure that the customer group which has the tool shows better metrics in terms of net customer score and revenue and profit. This will enable Lafourchette to gather data on how effective this tool proves to be and check whether a SaaS solution or a free offering makes more sense.

5.2 Risks associated

Ethically, this solution should not cause any problems as it is not based on personal/confidential data, as long as it is deployed in the company providing the restaurant's data (LaFourchette for instance).
Otherwise, the main risks of our solution are pricing errors, with the fact that small errors on cheap items can cost much more than substantial errors in highly priced items to a restaurant cost. However, restaurant's owners common sense should mitigate most of those are emphasis should be put that this is just advice, rather than a guaranteed best price. We would still recommend that Lafourchette follow-up with customers to ensure that there is no such problem with the solution.

5.3 Code

All of our code is made available open-source in our Github.

6 References

1. Number of restaurants in Paris
2. Selenium
3. BeautifulSoup
4. Pennington et al., "GloVe: Global Vectors for Word Representation", 2014
5. Le, Q, Mikolov, T, "Distributed Representations of Sentences and Documents", 2014

Appendix - Contributions

Further to defining the project all together, we separated the work between all of us in the following way:

- Thibault: Notebooks to scrape the data (with Kenil), first clean-up of the data, set up of the machine learning framework (metric choice, cross-validation)
- Kenil: Scraping the data (with Thibault), model selection and hyper-parameter tuning, embedding visualization
- Victor: descriptive statistics, GloVe implementation, first tests with GloVe and with TF-IDF representation