Project 3: Restaurant Risk Forecaster

Jessica Hudiono

Project design

My goal was to create a tool to alert restaurants if they were at risk of failure within the next year. Experts identified common reasons for restaurant failure, such as bad customer experience, targeting the wrong demographics, or lack of visibility, which I then tried to extrapolate from numerical and categorical data from Yelp restaurant profiles snapshotted in 2016. Yelp data snapshotted one year later provided the outcome for each restaurant: failure or continued operation. Identifying the most influential features would help the at-risk restaurants prioritize areas to fix.

Tools

- Github: Version control.
- PostgreSQL: Initial data storage and exploration. The raw data was in CSV and JSON format and was too large for Pandas.
- Pandas: Feature engineering. Querying PostgreSQL for only the relevant entries resulted in a small enough dataset for Pandas to manipulate and perform more complex transformations.
- FuzzyWuzzy: Merging feature and target datasets. See data challenges.
- SMOTE: Interpolate new rows to balance dataset.
- Sklearn, Numpy: Feature engineering, training models.
- Pickle: Save trained models for later testing and experimentation.
- XGBoost: Additional model, similar to Gradient Boosting but with regularization and improved performance due to parallelization.
- Matplotlib: Generating visuals for analysis and presentation.

Data

I used the datasets from the Yelp Dataset Challenges Round 8 (July 2016) and 11 (December 2017). For each challenge Yelp made available a partial snapshot of their database, including business profiles, reviews, check-ins, tips, and user profiles. I filtered for businesses categorized as food vendors (Restaurants, Cafes, Bakeries). Then I took the ones that appeared in both datasets and assigned to the target value whether they closed or remained open in Round 11.

Data challenges:

- Some businesses had missing data (i.e. hours, price) that needed to be imputed.
- I had to derive my own key (business name, address) in order to perform joins between the two datasets since the business IDs were not consistent between datasets.

Furthermore, one dataset used the full address while the other used only the street address, so I needed a custom function to reasonably guess if the addresses matched. To do this I used the FuzzyWuzzy module to score the similarity between strings.

- The “snapshot” nature of the dataset made it tricky to take exact measures that depended on time. For example, I couldn’t track monthly rating, new reviews per month, price changes, or the exact date the restaurant opened or closed. For now I decided estimating was good enough but in future iterations I would want to investigate this issue more.
- Imbalanced dataset: 92% of the restaurants stayed open. I used SMOTE for interpolated resampling to get a balanced dataset for training.

Metrics

I wanted to focus on improving recall, or minimizing false negatives since incorrect predictions of success was very harmful to my goal. False positives would make users overly cautious and give them more incentive to improve. But an incorrect success prediction gives restaurant owners false confidence that they don’t need to improve their venue, which would be even more harmful than giving no information.

Algorithm(s)

Initially I tried Gradient Boosting, Random Forest, SVM, Naive Bayes, and Logistic Regression. I trained each algorithm on a variety of hyperparameters with GridSearchCV, then tested the optimized version on the holdout test data. Gradient Boosting emerged as the clear winner in all metrics except speed. From there I learned about the XGBoost variant of Gradient Boosting, which had similar, slightly better results and performed faster due to parallelization.

The recall score looked promising at first at over 85%. However, I realized that the original dataset marked the “stay open” outcome as positive (1) and “close” as negative (0). I swapped the target variable values, retrained the dataset, and retested on the unbalanced version of the data. Unfortunately the recall score dropped to 30%, meaning that the model is not capturing as many failed restaurants as it should. If I had more time I would try adjusting the threshold for positive classification.

Next time

Aside from not getting all the feature data that I wanted, one problem I had early on was losing track of what transformations I had applied to my data and what algorithm parameters I had tried, which led to a lot of repeated work. I would like to create a template for future projects based on my pipeline for this project. Each step should be self-contained and generalizable. Hopefully this will make future projects more organized and efficient.

Also, the Yelp data consisted of JSON objects, so a NoSQL database like MongoDB would have been a better fit than PostgreSQL.