Predicting Business Closures and Star Ratings Using the Yelp Open Dataset

Kirill Sirik
Princeton University

December 16, 2025

Honor Code: I pledge my honor that I have not violated the Honor Code during this assignment. /s/ Kirill Sirik

Abstract

Yelp online reviews provide insightful signals about user satisfaction across various businesses, and the predictive power of these reviews in business success is often measured by the star ratings received by the business. This paper examines whether we can also forecast an additional, alternative business success metric, business closure, using information up to some defined cutoff date, T. The metrics used to approach this problem include review activity, ratings and dispersion, votes, tips, check-ins, hours, and feature categories. I will examine two questions in this paper: 1) Can we predict whether a particular business is closed after T?; and 2) Out of the business that remain open after T, can we accurately predict their average star ratings? Both classification and regression problems remained largely constrained in the researchers' approaches, but with recent advancement in Artificial Intelligence and Machine Learning methods, including the rapid growth Neural Nets, stakeholders now have access to more resources to help them approach this problem. In this paper, I will compare the performance between classical and modern ML techniques on both regression and classification tasks, and explore whether using Neural Networks leads to performance improvement.

Contents

1 Introduction

1.1 Overview

Problem to be solved.

There are many factors that go into business maintenance. Business owners always try to identify ways to ensure that their business operates successfully and is profitable. Hence, the problem of forecasting business closures carries great economic significance to both prospective and current business owners. Business closure is an accurate metric to assess business success since most businesses that close down are ultimately not successful. Furthermore, business closure directly reflects customer satisfaction and demand, since businesses without a continuous customer flow are not profitable, and are likely to close down at some point in the future. Accurately predicting these outcomes can help prospective business owners assess their risk and pursue the most optimal course of action.

Not all businesses remain closed, however. Many businesses find their customer base and continue operating successfully. In fact, of all the businesses present in my data set, nearly 80% were and remained open. For open businesses, customer satisfaction matters, since it directly influences whether that business remaining open. One potential way for business owners to assess future customer satisfaction is to predict review star ratings using data from the past.

In this paper, I will be tackling these two problems simultaneously: I will try to predict business closures and average review star ratings using common business attributes from a large Yelp Open data set.

Why is the problem important?

The problem of predicting business closures and future review star ratings is important as it directly assesses business success from two different angles: from an angle of businesses that worry about closing and businesses that remain open but would like to ensure high regard from potential customers. Having access to this information early on can help business owners implement initiatives to improve their businesses, or to make alternative arrangements about the location or type of their business, provided the data on past business successes and failures. Thanks to the recent advancements in Artificial Intelligence (AI) methods and a growing popularity of classical Machine Learning (ML) methods, stakeholders now have access to tools that can help them solve this problem with a variety of different approaches. Moreover, the presence of vast, publicly available, free-to-use datasets, like the Yelp Open Dataset, provide for an opportunity for business owners to leverage AI and ML tools to aid in the process of evaluating their business success.

Contributions.

My project is an addition to a growing literature that explores the Yelp Open dataset to forecast business survival and star ratings, but my approach differs from the pioneering work in several ways. Most former studies typically exclusively focus on either predicting business closures or predicting star ratings, solving either classification or regression problem at a time. Furthermore, many of the earlier studies frequently combine multiple yearly releases from Yelp or external Data sources to augment their datasets. My work uses a single Yelp Open Dataset, which contains data on 11 different metropolitan areas. I also define a cutoff date T, determined solely by the review volume, with every feature computed with information available at or before T. More on the calculation of T later. I deliberately restrict the scope of my paper to business attributes and simple review attributes, to explore the predictive power of rather simple and easy-to-obtain metrics. This design also allows me to explore feature selection to some limited capacity.

Techniques overview.

In this paper, I am using the Yelp Open Dataset to study two prediction tasks as they pertain to businesses: 1) A classification problem of predicting business closures; and 2) A regression problem of forecasting future

review star ratings for businesses that remain open. To avoid data leakage, I implement a time-cutoff design, where I compute all features using only information dated on or before a single calendar cutoff date T. I then define all regression labels using post-T reviews. I set T = 2020-12-31, which isolates a little over one year of most recent reviews, since the last review in the dataset was left on 2022-01-19. After fixing that date T, I construct *all* features using only pre-T data. This design introduces an artificial gap between features used to construct the models and features used to evaluate that model's performance. The classification task target is the variable `closed` $= 1 -$ `is_open`, constructed using the `is_open` boolean flag from the Yelp dataset. For the regression task, I select my target variable to be `avg_stars_postT`, which is defined only for businesses that remain open *and* have at least `3` post-T reviews. I impose an additional restriction on the minimum number of reviews to exclude restaurants with single or no reviews at all to avoid the issue of sparsity in my data. As is standard for model evaluation, I use an 80/20 train-test split at the `business_id` variable level.

I compare classical ML techniques, including logistic regression, calibrated Ridge regression, decision trees and random forests against two feedforward neural networks, which are multilayer perceptrons with 1 (MLP-1) and 2 (MLP-2) hidden layers respectively.

Features and pre-processing.

I parse `business.json`, `review.json`, `tip.json`, `checkin.json`, and `user.json` files provided inside the dataset to build three pre-T feature groups: `GEO` (e.g., `state`, `latitude`, `longitude`); `BUSINESS` (attributes from `business.json`, including `hours` and `categories`); and `REVIEW` (pre-T dynamics such as `n_reviews_total`, `n_reviews_recent`, `avg_review_stars`, review dispersion, and simple combinations of votes and tips). I also define a major feature group `ALL` $=$ `GEO` $\cup$ `BUSINESS` $\cup$ `REVIEW`, which is simply the union of all three disjoint feature sets. I do the pre-processing with the help of a `ColumnTransformer` by imputing numeric features with a median and standardizing with `StandardScaler`. I will elaborate on the process below. For categorical features, I use one-hot encoding with `OneHotEncoder(handle_unknown="ignore")` and replace any missing values with the table's mode.

1.2 Data Description and More on Pre-processing

A word on my dataset and how I obtained it.

I use the Yelp Open Dataset, which is an extensive, publicly available dataset provided by Yelp that spans 150,346 businesses, 11 metropolitan areas, and includes 200,100 pictures and 6,990,280 reviews. I obtained the dataset by downloading it from the following Yelp Open Dataset website. The provided dataset includes two large files: 1) a JSON file, which contains one PDF and five JSON files (which is the data set that I will be using in this paper); and 2) a TAR file that contains one JSON file, one text file, one PDF file, and a folder containing 200,000 photos. This dataset has been frequently explored in the relevant literature, both due to its vast coverage and its accessibility. Since the data set is provided at no cost by Yelp, many researchers and students use the dataset to test out different ML techniques.

In this study, I will be only using five `.json` files from the first archive provided by Yelp. I omit the prefix `yelp_academic_dataset_` from the filenames to avoid repetition. The used file names are: `review.json`, `tip.json`, `checkin.json`, `business.json`, and `user.json`.

Across these five JSON files, we have access to 38 features, all of which are listed in detail in Table 1, including the JSON file to which they belong.

How does the Yelp Dataset relate to the problem of business closures?

The Yelp Dataset relates to the problem of predicting business closure and average star ratings for open businesses quite well. Given its vast data inclusion, the Yelp Open dataset contains a binary variable that indicates whether a particular business is open (as mentioned above) or not. This provides for a great opportunity to simply define define a new binary variable for "Closed" businesses whose entry would

depend solely on the value of is_open for that business. Furthermore, the data set includes many useful review-derived features described in greater detail below. The inclusion of the many review features enables opportunities to explore the role of all of these features in predicting business star ratings.

Dataset size.

In total, I had access to 150,346 unique businesses and 6,990,280 reviews, which reflects the total data available on the Yelp Open Dataset website. To stay consistent with the time-aware design, I split the reviews such that 6,338,752 reviews are dated earlier than January 1st 2021, and 651,528 reviews are dated January 1st 2021 and above. Furthermore, for modeling purposes, I further restrict the number of businesses by excluding 2,311 businesses whose first review occurred after T. This design allows me to ensure that all feature construction is done with pre-T data only. After running this filtering step, all remaining businesses have at least one review dated prior to T. For regression, I altered the size of the dataset even further by requiring that all businesses that are open post-T have at least 3 or more reviews.

Main features and targets.

I defined two target variable levels for the two tasks studied. For the business closure classification task, I defined label closed, which is derived from Yelp's is_open indicator variable. In particular, closed = 1 - is_open, which means that closed = 1 when the restaurant is not open and 0 otherwise. For the regression task of predicting post-T average business star rating, I define my target variable to be avg_stars_postT, which is simply computed as the mean star rating over all reviews dated after T. This variable is only defined for businesses that remain open after T and have at least three post-T reviews to remove noise.

I organize the provided feature variables into three distinct groups: Geographic and spatial features (GEO), Business category features (BUSINESS), and customer activity features (REVIEW). GEO features are constructed from business.json, and include features like latitude, longitude, city, and state, which capture regional trends. I also use business.json features to define a categorical variable, primary_category, which takes the first listed business category to capture the type of business. Finally, I define the REVIEW category by aggregating customer activity up to T. Using review.json, I extracted total recent review volume `n_reviews_total`, `n_reviews_recent`, star ratings and relevant star dispersion `avg_review_stars`, `std_review_stars`, and metrics meant to track engagement `n_useful_total`, `n_funny_total`, `n_cool_total`, `n_useful_recent`, `n_funny_recent`, `n_cool_recent`. I also use the tip.json file to extract tip-based customer activity, including `tip_count_total`, `avg_tip_len_total`, `recent_tip_count`, `avg_tip_len_recent` and check-in activity from `checkin.json` (`n_checkins_prior`, `n_checkins_recent`).

Table 1: Yelp Open Dataset JSON files and available fields.

JSON file	**Available fields (columns)**
yelp_academic_dataset_checkin.json	business_id, date
yelp_academic_dataset_tip.json	business_id, compliment_count, date, text, user_id
yelp_academic_dataset_review.json	business_id, cool, date, funny, review_id, stars, text, useful, user_id
yelp_academic_dataset_business.json	address, attributes, business_id, categories, city, hours, is_open, latitude, longitude, name, postal_code, review_count, stars, state
yelp_academic_dataset_user.json	average_stars, compliment_cool, compliment_cute, compliment_funny, compliment_hot, compliment_list, compliment_more, compliment_note, compliment_photos, compliment_plain, compliment_profile, compliment_writer, cool, elite, fans, friends, funny, name, review_count, useful, user_id, yelping_since

More on pre-processing.

I have pre-processed the Yelp Dataset in a couple of ways. First, I had to convert review dates to a date-time format, remove any invalid dates (e.g dates that were outside of the defined/desired range), and keep only reviews dated at or before T to construct my predictor variables. Then, I grouped the reviews based on the businesss_id value to construct business-level features. I also removed all undefined values, which included filling values of undefined standard deviation to 0 or restricting my review standard deviation plots to only consider businesses with more than 2 reviews to remain consistent with the definition of a standard deviation. Furthermore, I removed businesses with 0 pre-T reviews before proceeding with model construction, since such businesses had little value in regards to their review-level features.

I also used one-hot encoding to turn categorical features like city, state, and primary_category using a OneHotEncoder function that turns categorical variables into binary (i.e. numerical) variables. I also impute the numeric feature columns with the median, which simply means that, any time a data entry is missing in a numerical column, it is replaced with that column's median value. For columns containing categorical variables, I follow the same process and impute them with the most common class found in that column. I then apply a StandardScaler function to all numeral features, which standardizes the data and scales the data by removing the mean across the data points and re-normalizing the variance to be 1. This step is among the most crucial of pre-processing steps that I do, since all models that include regularization, such as Ridge, Logistic, and MLP, can behave vastly different across features with features on different scales. For example, the models mentioned above would likely yield different predictions for the case when we standardize both latitude and review counts, which we should expect would have different distributions, and the case when we keep the scale of the two features as-is.

Finally, I ensure that all features used are constructed using only information available before the cutoff date, whereas all outcomes are strictly computed after the cutoff date. I also define my business feature group as to exclude review_count and stars from the group definition. I do so to avoid data leakage, since review_count and stars that are supplied by the Yelp data set contain information dated after the cutoff date. To stay consistent with my design, I manually recompute the review counts and stars that are uniquely based on pre-T data.

1.3 Literature Review

Most Relevant Prior Work on Predicting Business Closures and Average Star Reviews

Some of the most important work on my paper's goal is a Stanford CS229 paper on using machine learning to predict business success by Pan et al. [Pan et al., 2018] The paper frames the problem of predicting business success as a machine learning problem using real start-up business data with an imbalanced dataset, just like myself, since the majority of the businesses in their dataset were closed, opposite of my dataset. The paper includes many aspects that make it influential: 1) the authors compare a wide range of models, including logistic regression, random forests and KNN; 2) the paper explicitly suggests adding richer features, using word embeddings, neural networks, etc. The paper is influential in that it formalized the setup of the business success prediction problem as a supervised ML problem, demonstrated the benefits and limitations of simple ML models, and applied ML performance metrics to guide investor decision-making processes. However, the data set that the authors explore is both smaller and older than the Yelp Open Dataset. Pan used Crunchbase Data Export, which contains pre-December 2015 data on 60,000+ companies. The dataset that I will be using is more modern and extensive, and presents an opposite data imbalance to Pan's dataset.

Another important past study on predicting business reviews is Li's and Zhang's project using Yelp's Dataset Challenge, which contains data on 42,153 businesses and has a total of 1,125,458 reviews. Li and Zhang's explored classical ML tools like logistic regression, Naive Bayes, and support vector machines (SVM) to execute sentiment analysis of user reviews. [Li and Zhang, 2014]. This work is important

because it introduced the subject of review sentiment analysis to the ML research community. Some of the analysis done in my paper on REVIEW-derived features was heavily inspired by this work.

Most Relevant Prior Work using Yelp Open Dataset

The most relevant paper that used the Yelp Open Dataset to predict business closures is written by Lu et. al., and reads as follows: "Should I Invest it? Predicting Future Success of Yelp Restaurants" [Lu et al., 2018]. In the paper, the authors specifically target restaurant closure predictions in 2017 using the Yelp data from 2016. This work is relevant to one of the questions that my paper aims to address in the way that Lu et. al split the features into text and non-text features that they would later feed into logistic regression models.

Another relevant study that tries to predict restaurant survival using the Yelp Open Dataset specifically is the "Interpretable Business Survival Prediction" paper by [Vallapuram et al., 2021]. The authors of the study performed extensive feature engineering across four feature sets, including geography, user mobility, business attributes, and linguistic modeling to develop classifiers for business survival prediction. What is different about this study is that the authors augmented classifiers with an interpretability component. The authors' rationale for doing so was that "simply predicting the survival of a business is not adequate". While I concede the author's point that interpretability is important when working with classification models, I argue that we can interpret classification data without making the models more complex. Hence, although my paper will touch on similar topics to Vallapuram's paper, I do not go incorporate any additional architecture changes to the models that I will be working with.

Models Used.

My project does not use any special models or custom model architectures; all of the models that I used use standard model architectures as defined by the scikit-learn library. However, deploying alternative model architectures might be among promising future steps to supplement my work. The models that I will be using have been used extensively in prior research on business survival prediction (e.g., Lu et al., 2018; Vallapuram et al., 2021). My main contribution to the literature is the suggestion of feature design, cutoff-time setup, and comparative analysis of feature groups.

1.4 Exploratory Data Analysis

Data Set Visuals.

Figure 1 displays the imbalance between closed and open businesses in the Yelp Open dataset. As we can see from the bar chart, the ratio of open to closed businesses is approximately 3.9:1. This ratio indicates an imbalance within the data set, which should inform our analysis later on. Table 2 displays the total number of loaded reviews that I analyzed in my code, which is consistent with the number of reviews reported by Yelp. In other words, I accessed the entirety of the review data for the purposes of running my code, which I then split by the cutoff date, T. As shown in Table 2, I manually selected the cutoff date, T, to be December 31st, 2020. The ratio of reviews dated pre-T to reviews dated post-T is roughly 9.7:1.

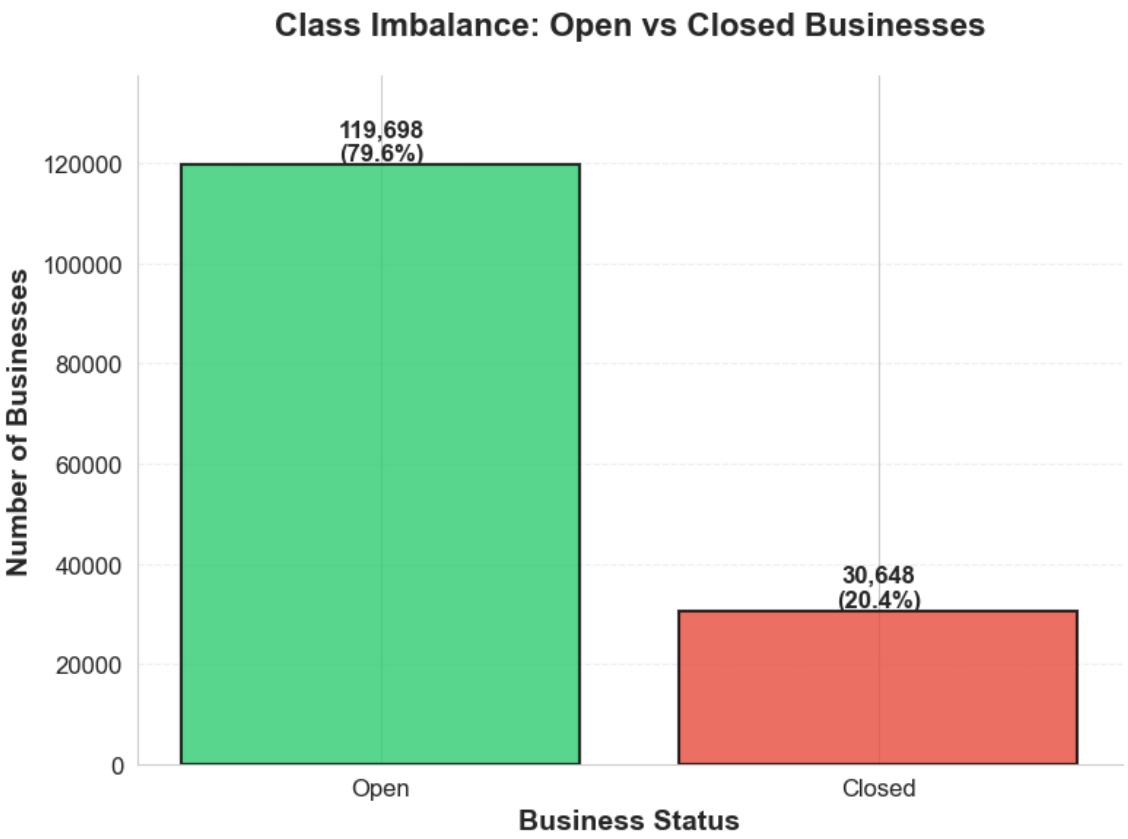

Figure 1: Class Imbalance between Open and Closed Businesses.

Table 2: Dataset size and cutoff used to construct pre-T features from `review.json`.

Quantity	Value
Total loaded review records	6,990,280
Latest review date in dataset	2022-01-19
End review year in dataset	2022
Chosen cutoff year (T's year)	2020
Cutoff date (T)	2020-12-31
# Reviews dated $\leq T$	6,338,752
# Reviews dated $> T$	651,528

Variable Distribution.

Figure 2 displays the distribution of the total as well as recent business reviews on the logarithmic scale. All reviews used to construct the graphs are strictly dated pre-T. The left plot displays the total amount of pre-T reviews, while the right plot shows the number of recent reviews across businesses, where recency is defined to be a 180-day period before T. In other words, the reviews that are labeled "recent" were left within 180 days before T. As is clearly evident from Figure 2, the data set is inherently skewed to the right, with zero recent reviews dominating the 180-day window. A potential confounding variable that could have impacted our data set is the COVID-19 pandemic, when many more businesses closed down than we would expect in a world scenario with no global pandemic.

Table 3 summarizes the main metrics relevant to Figure 2, offering additional insights into the review activity distribution in the Yelp Dataset. According to Table 3, a median restaurant in the Yelp Dataset had 14 reviews, while the range between restaurants with the lowest and highest 25% of reviews was 28

reviews. In other words, businesses that were among the 25% of those with highest reviews had 28 or more reviews than did businesses in the bottom 25% category. Top 10% of businesses had 93 or more reviews, and the top 1% of businesses had 455 or more reviews. This distribution confirms the right-skewed pattern that we observed in Figure 2, with the top percent of businesses having a disproportionately higher review count than the majority of businesses in the Yelp Dataset. Table 3 also shines light on the distribution of recent reviews. As a reminder, recent reviews are defined in the same way as they are in Figure 2, in that they are dated within 180 days at or before T. Based on Table 3, the median number of recent reviews was 0, which is consistent with the Figure 2, where the leftmost bar is highest, with about 80,000 businesses having 0 recent reviews. Businesses with the top 25% of recent reviews had 2 or more recent reviews than did businesses in the bottom 25%. The top 1% of businesses, once again, dominated the other businesses in that they had a disproportionally higher number of reviews.

Based on figure 4, there was a clear sign of high correlation between review recency and business closures. In other words, businesses with the last review dated more than 365 days from T were much more likely to be closed than businesses with the last review dated between 0-30 days before T. Higher values of days since last review were correlated with a much higher count of closed business and hence a higher closure rate. For instance, whereas the 0-30 category days since last pre-T review had only 2.94% closed businesses, when the last review was dated earlier than a year before T, the closure rate was much higher, 43.96%.

What stands out in particular is the distribution of open and closed businesses. As we can see, open businesses have a slightly higher average star rating than closed businesses, although not by a significant amount. What is interesting is that closed businesses actually have a higher spread of businesses with medium-to-high reviews as opposed to open businesses, which spike at high reviews above 4.5 but also have fatter tails for reviews lower than 2 stars. The standard deviations for open and closed businesses are practically the same, which suggests little difference in rating dispersion between the two classes. It is important to note that, in Figure 3, I only consider the standard deviations of business with 2 or more reviews because standard deviation is not a meaningful metric for analyzing businesses with a single or no reviews at all. However, even though there were noticeable differences in the distribution of open and closed businesses, there was also significant overlap between the distribution of the two classes.

Noisiness in the Dataset.

The Yelp Dataset indeed had some inherent noise to it. To be more precise, the noise mentioned above stems from the fact that Yelp permitted businesses with 0 reviews, businesses with reviews dated after my selected cutoff T, and businesses with fewer than 2 reviews to be listed in the data set. This leads to measurement noise for my review-derived predictions avg_review_stars (which is undefined for businesses with 0 reviews) and std_review_stars (which is undefined for businesses with 0 or 1 reviews). For short time frames, "recent review volume" is mainly zero for many businesses. Furthermore, there are a few heavy-tailed businesses with extremely large review counts that have a disproportionally higher review volume than their peer businesses.

To combat the inherent noisiness in the Yelp dataset, I took three actions. Firstly, I restricted the businesses to at have at least one pre-T review for all features that were derived from reviews to at least some capacity. This design allowed me to avoid sparsity in the business data. Secondly, I stabilized the regression task label by imposing a restriction on businesses to have enough post-T reviews. That is, I required that all businesses have at least 3 or more reviews dated after T, since the mean that I had computed using a restriction of only a single review yielded a noisy estimate. Finally, I indirectly reduced the noisiness even further by selecting models that are able to tolerate skewed data, like Random Forests and regularized linear models.

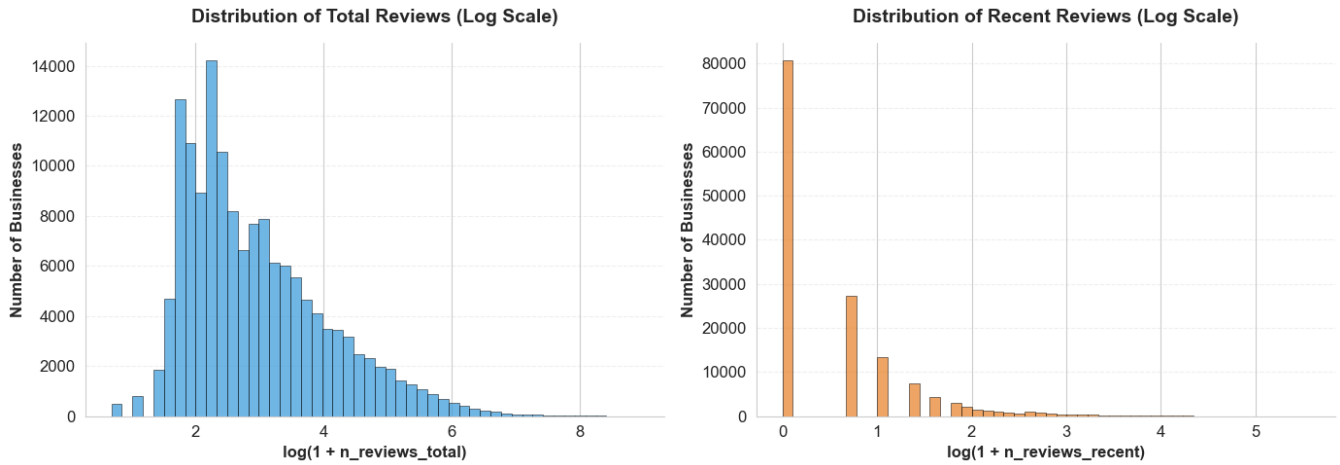

Figure 2: Distributions of review activity signals computed using reviews dated $\leq T$. Left: $\log(1 +$ `n_reviews_total`$)$. Right: $\log(1 +$ `n_reviews_recent`$)$ for a 180-day window ending at T.

Discussion (Figure 2). Figure 2 shows that review-based features follow an extremely right-skewed distribution, which is why I used the log transformation to display the data distribution. Total pre-T review volume has a very long right tail, since only a handful of businesses receive such comparatively high number of reviews. Recent reviews are mainly concentrated around 0, suggesting that the majority of businesses had 0 reviews dated within 180 days before T. That might be the case because many businesses may have obtained their reviews in the historic past, but with little to no activity in the present, which is generally indicative of a business becoming more prone to closing.

Table 3: Summary statistics for review activity features (corresponding to Figure 2).

	`n_reviews_total`	`n_reviews_recent`
Median	14.0	0.0
IQR	28.0	2.0
90th percentile	93.0	5.0
99th percentile	455.0	23.0

Discussion (Table 3). Table 3 Table 3 quantifies the distribution trends that we observe in Figure 2. The median business in our dataset has only 14 reviews in total and 0 reviews within 180 days before T. 99th percentile businesses, however, have both disproportionately high total reviews and high recent reviews. These numerical findings reiterate the heavy-tailed distribution that we can observe in Figure 2.

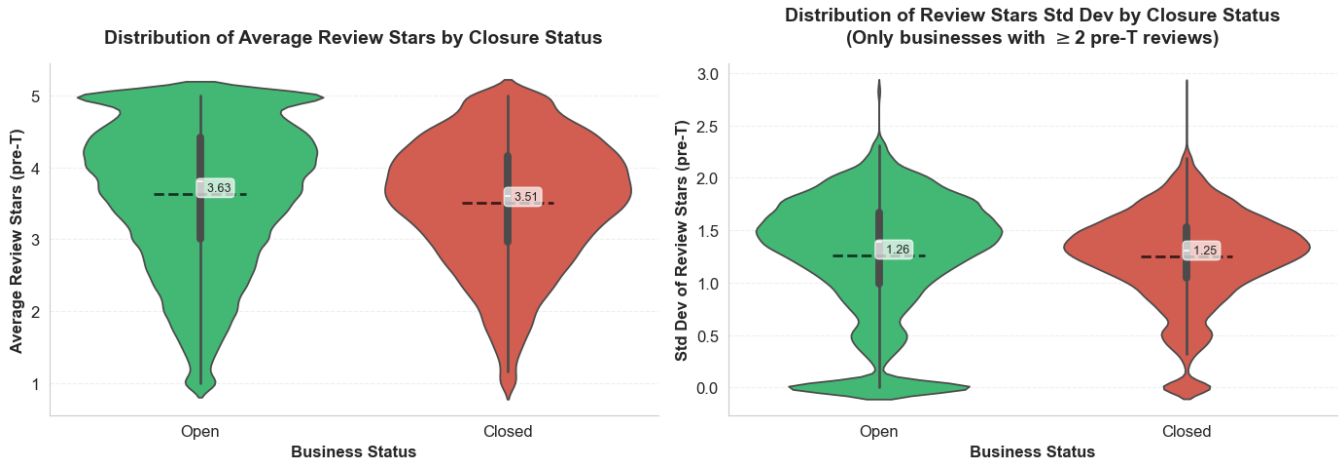

Figure 3: Distributions of pre-T rating signals by closure status. Left: pre-T average review stars (businesses with ≥ 1 pre-T review). Right: pre-T star-rating dispersion (standard deviation) shown only for businesses with ≥ 2 pre-T reviews.

Discussion (Figure 3). Figure 3 compares signals derived from user reviews among closed and open businesses using pre-T user review data. Although open businesses have on average a higher mean star rating, closed businesses have a rather similar mean that is a fraction of a standard deviation away. Moreover, the visual plot suggests that the distributions between open and closed business ratings are more similar than different. Moreover, the difference between the standard deviation distribution between the two groups is even more subtle, with standard deviations having almost identical magnitude.

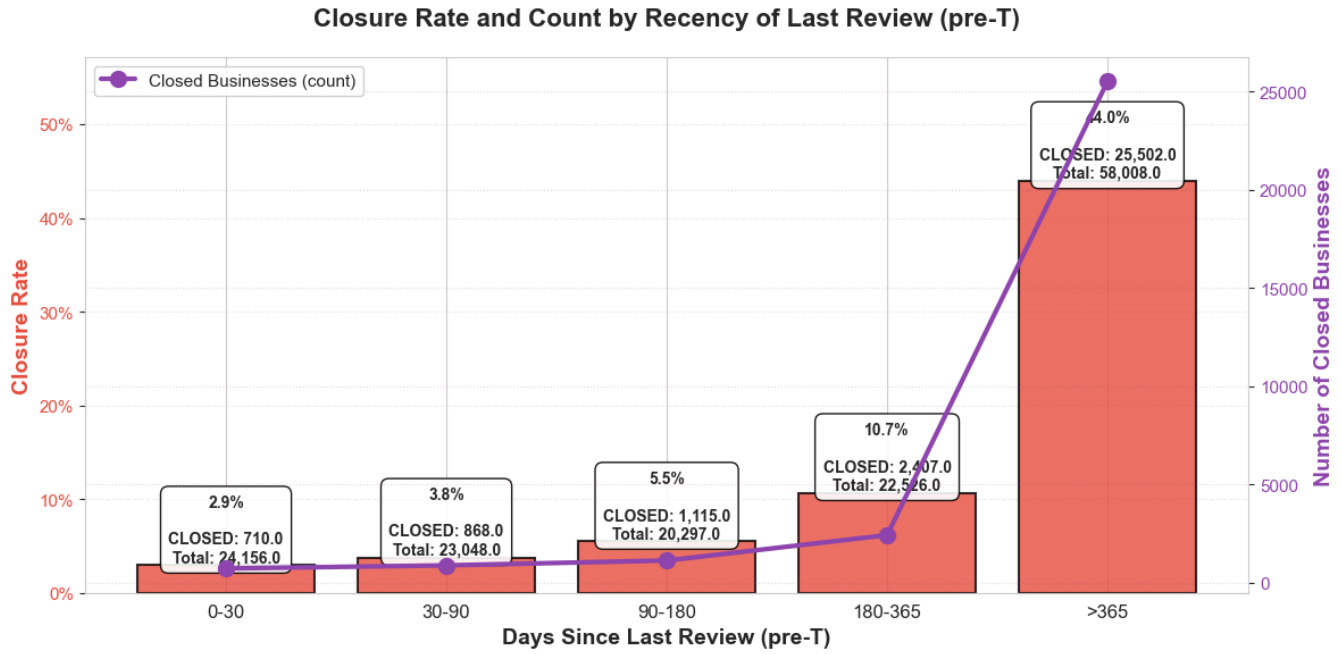

Figure 4: Closure rate by recency of the last pre-T review. Businesses are binned by days since last review prior to T (computed on the pre-T timeline, $N = 148{,}035$).

Discussion (Figure 4). Figure 4 displays a strong relationship between review inactivity and subsequent business closure: Business closures rose sharply as the last reviews became less recent. Businesses with the last review dated more than a year prior to T have had dramatically higher closure rates than any of the other categories. The sharpest jump between two consecutive categories was between the businesses

whose reviews were posted within 180-365 days before T and those whose most recent review was dated more than a year prior to T.

Table 4: Closure rate by recency of the last pre-T review (corresponding to Figure 4).

Days since last pre-T review	Total businesses	Closed count	Closure rate
0–30	24,156	710	0.0294
30–90	23,048	868	0.0377
90–180	20,297	1,115	0.0549
180–365	22,526	2,407	0.1069
>365	58,008	25,502	0.4396

Discussion (Table 4). Table 4 provides numerical data to supplement the findings of Figure 4. Business closure rates rose sharply from 2.94% in the '0-30' day category to 43.96% in the >365 category, which is almost a 15-fold increase. Exploratory Data Analysis findings suggest that this feature is highly predictive, so incorporating this feature further in future studies might be a feasible and interesting design extension.

2 Methods

2.1 Problem Setup: Classification and Regression

For each business, I constructed a binary label `close`, which indicated whether the business was closed after a fixed calendar cutoff date T. I derived this feature by mapping the provided "is_Open" Boolean feature to `closed` = 1. I observed a class imbalance between "closed" and "open" businesses, since around 20.4% of the data belonged to the former class, while the rest belonged to the latter.

2.2 Metrics

For the classification problem, I computed six metrics: Accuracy, Precision, Recall, F1, ROC-AUC, and PR-AUC, where the latter is the most important metric due to the imbalance present in the data set. Accuracy measures how well the model predicts true values out of the entire dataset. This metric might be misleading on our imbalanced dataset because a model that predicts majority "Open" label for businesses might have a high accuracy score simply because there are substantially more open businesses in our dataset. Hence, although I compute accuracy, I do not treat it as the main metric that reflects the model's success. Instead, I emphasize precision more. Precision, in our case, measures the true number of closed businesses of all businesses that are predicted to be closed. In our data set, it is important to keep the status of false positive predictions under control, especially if the predictions based on my models are to be used to make important decisions about any particular businesses. Having high precision implies that my model does not predict too many false positives, which is an ideal goal.

Recall tells us how many truly closed businesses were predicted by our model, out of all closed businesses in our dataset. This metric matters in our data set because, since the businesses are dominated by the "Open" label, it might be easy for our model to miss business closures while maintaining high Accuracy. F1 score sits in-between Precision and Recall, and is meant to provide a trade-off between the two metrics. Calculating this metric is useful for the Yelp data because it is hard to achieve high Recall and Precision simultaneously, since raising precisions tends to reduce recall, and vice versa. Since F1 score represents the harmonic mean of Precision and Recall, it will be high only when the two quantities are high, in which case we would have both few overseen closures and few missed closures.

Unlike the previous four metrics, both ROC-AUC and PR-AUC calculate areas under a curve (which is the reason why they have the AUC suffix). ROC-AUC measures our model's ranking quality across all possible thresholds by plotting true positive rate against the false positive rate. I used ROC-AUC for

hyperparameter selection, since this metric is threshold-independent, and is generally anchored across CV folds. PR-AUC, on the other hand, conveys the classifier's performance by plotting Precision against Recall. In the case of the Yelp Dataset, PR-AUC is especially valuable due to the relative rarity of "Closed" businesses. A potential issue in similar instances of imbalanced datasets is that our model might look promising judging on the four broad metrics above while still unsuccessfully identifying the few closed businesses that are present in our dataset without accidentally misclassifying open businesses as closed. Hence, a high PR-AUC value in the context of this study would imply that, when the model predicts a business closure, it is generally correct in that prediction, without suffering from the problem of having too many false positives.

For the regression problem, I compute Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R^2 metrics.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

As we can see from the construction, MSE places a large penalty on errors with high magnitude due to squaring, which we might find useful if we would like to avoid making large (star rating) errors as much as possible. Moreover, MSE is so common in being defined as the default loss function that most models are tuned to work well with MSE. MSE's prominence makes this metric the best choice for hyperparameter selection. Unlike MSE, RMSE represents the error using the same units as those of our target variable.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} = \sqrt{\text{MSE}}$$

In our case, because we are targeting the average business star ratings, RMSE can be interpreted directly as the error in our prediction of the said star rating.

Mean Absolute Error is generally more robust to outliers than MSE/RMSE because it does not inflate the magnitude of extreme data points by squaring them. Instead, MAE is linear, which is useful for our data set which contains select businesses with the extreme value for the total number of star ratings.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

MAE is typically comparable in performance to the more frequently used MSE and RMSE while not letting potential outliers affect our model too much. Finally, the most intuitive metric among all regression performance metrics is

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

This metric is familiar to most from high school statistics, and is simply defined as a quantity that measures how much of the data variance our model explains compared to the baseline model that only predicts the mean. However, $\text{R}\hat{2}$ may not be as good of a predictor for models that either underfit the data or models that are tasked with predicting noisy datasets.

2.3 Custom Path: Cutoff-Time Design and Feature Engineering

I organized the features into three distinct groups: GEO, BUSINESS, and REVIEW. GEO features contained geographic information such as state and coordinates (longitude and latitude). BUSINESS contained static business attributes, like hours, and other primary_category. REVIEW features captured pre-T dynamics in that business' reviews, including review volume, mean star rating, star rating variance, vote

counts, tips, check-ins, and others. I also defined ALL to be a union of GEO, BUSINESS, and REVIEW. The detailed brakedown of all features that went into the construction of GEO, BUSINESS, and REVIEW ablations are summarized in Table 5.

Table 5: Feature sets grouped by data source. Review-derived features are computed using information dated $\leq T$ (with a separate recent window where indicated).

Group	Subtype	Features
GEO	Numeric	`latitude`, `longitude`
	Categorical	`city`, `state`
BUSINESS	Numeric	none; leakage-prone business metrics excluded
	Categorical	`primary_category`
REVIEW	Reviews (history + recent)	`n_reviews_total`, `avg_review_stars`, `std_review_stars`, `avg_review_length`, `n_reviews_recent`, `avg_stars_recent`
	Votes (history + recent)	`n_useful_total`, `n_funny_total`, `n_cool_total`, `n_useful_recent`, `n_funny_recent`, `n_cool_recent`
	Tips (history + recent)	`tip_count_total`, `avg_tip_len_total`, `recent_tip_count`, `avg_tip_len_recent`
	Check-ins (history + recent)	`n_checkins_prior`, `n_checkins_recent`

2.3.1 JSON Parser Function

As part of the assignment, I was asked to implement a JSON loader function. I will describe the logic of the function here, but the function itself is available alongside my code. My function, parse_yelp_json_lines(path, n_rows = None, verbose = True) loads JSON files from the Yelp Open Dataset. The inputs to my function are the file location for the Yelp JSON file, the optional [int] cap on how many valid JSON objects the function will load, and a flag that is meant to print progress updates and final summary of the loader. My function's output is a pandas.DataFrame object where each row corresponds to a single parsed JSON record, and column names are derived from JSON file keys.

2.3.2 Brief Note on Custom Path Tasks

Regarding the custom path tasks for the Yelp Dataset, I described the regression and classification problem in detail (including my implementation) for both classical ML methods as well as for neural networks. I also included a description for my JSON parser function above. It is available through my code submission on Canvas.

2.4 Classical Machine Learning Models

2.4.1 Classification Models

For the classification class, I employed three baseline models: Logistic Regression with an L2 penalty, a decision tree, and a random forest (RF). By its construction, logistic regression serves as an interpretable model with log-odds coefficients. In particular, logistic regression models the probability of business closure using a sigmoid function, where each coefficient can be interpreted as a feature shifting the log odds of business closure, holding other predictors fixed or constant. This model is especially useful when dealing with the Yelp Open data set due to the fact that many of the predictors in the data set essentially act as signals of activity associated with a particular business. For instance, higher recent review volume indicates active customer engagement and business visibility, whereas declining activity might signal imminent profitability losses and potential closure. Coefficients fitted using logistic regression are highly interpretable about

clusre risk, and can be used as a strong reference point for exploring how flexible models like decision trees and random forests improve logistic regression findings through the use of non-linearities and interactions.

Logistic regression helps us evaluate whether these signals are additive in their log-odds. For instance, a negative coefficient associated with avg_stars_recent would imply that a lower value of recent reviews are associated with higher business closure probability. Furthermore, regularizing logistic regression with an L2 norm was an integral choice due to the presence of correlated and highly variable features like n_reviews_total, n_useful_total, and others. The Ridge penalty shrunk estimated coefficients towards 0, stabilized estimates despite multicollinearity, and prevented overfitting to noisy training data.

The decision tree was used to supplement logistic models by allowing non-linearities and capturing any interactions, which are plausible in the Yelp data set. For instance, business closure risk might increase significantly should the recent review category change, as we saw in the Data Description section. Yet another example may be when low star ratings coincide or interact with low review volume. One limitation of using decision trees is the fact that they might highly overfit the data, which introduces additional considerations to me when selecting appropriate hyperparameters. I also used Random Forests, which are simply an aggregate of multiple decision trees. RF is generally a stronger model than single decision trees because Random Forests and other ensemble methods de-correlate the trees that they later aggregate, thus reducing variance.

2.4.2 Regression Models

For businesses that remained open after T and had sufficient activity, defined as having more than 3 reviews, I predicted the post-T average star rating using the features constructed with pre-T data. The regression problem target is the average post-T rating for businesses with more than 3 reviews.

2.5 Neural Network Models

2.5.1 MLP-1: One-Hidden-Layer Network

As a first neural network model for both classification and regression, I use a feedforward multilayer perceptron with a single hidden later (MLP-1) that take the same features calculated before the cutoff, T, that were used for classical ML techniques. MLP-1 consists of an input layer, and a single hidden layer with a non-linear activation, and a single output layers. Unlike logistic regression, where features may be combined in the log-odds, MLP-1 learns non-linearities within the table and maps feature interactions as a subsequence of this learning. The flexibility of MLP-1 is useful for the Yelp Open Dataset because many of the predictor variables in the data set are unlikely to have purely additive effects. In addition to these benefits, MLP-1 is not as computationally demanding as deeper networks, making it an appealing NN choice. MLP-1 architecture in this case represents a one-hidden-layer neural network with the hidden layer width chosen from 64, 128, 256 units, using dropout and l2 regulatization. I explain the hyperparameter configuration later. I used the MSE loss and linear output for the regression problem; and I used a sigmoid function, ROC-AUC, and BCE loss for the classification problem.

2.5.2 MLP-2: Two-Hidden-Layer Network

MLP-2 shares the exact same benefits as MLP-1, in that the model is non-linear, and is not nearly as computationally expensive as some of the deeper NN models. However, unlike the model with just a single hidden neuron layer, MLP-2 has twi hidden layers, and can hence theoretically capture more non-linearities between the data instances. MLP-2 represents a two-hidden-layer neural network with the first hidden layer width selected from 128, 192, 256 units and the second hidden layer width selected from 32, 64, 96, 128 units using BatchNormalization. The hyperparameter selection will be discussed in greater detail below.

2.6 Training, Hyperparameters, and Model Selection

This section describes how I selected: 1) model hyperparameters, including tree depth, regularization strength, hidden-layer sizesand others; 2) optimizer andtraining hyperparameters like the learning rate and epochs for neural networks; and 3) how I selected final models. Hyperparameter feature spaces as well as cross-validation (CV) design are summarized in Tables 6 (classification) and 7 (regression), for convenience.

Classical ML hyperparameter selection.

For predicting business *closure (classification)*, I trained three models: `LogisticRegression` (L2), DecisionTreeClassifier, and `RandomForestClassifier`. All three models were trained on each feature group separately and on the feature set union, (`ALL`). I tuned model hyperparameters with `GridSearchCV` using *stratified 5-fold CV* to preserve the observed class imbalance in each fold (Table 6), since only a fifth of businesses in the dataset were closed. The hyperparameter search space was selected as to strike a balance between computational complexity while permitting the exploration of various models. The explored hyperparameters included: logistic regularization strengths $C \in \{0.1, 10\}$; tree complexity via `max_depth` $\in \{3, 5, 10, \texttt{None}\}$ and `min_samples_leaf` $\in \{1, 5, 10\}$; and RF capacity via `n_estimators` $\in \{100, 300\}$, `max_depth` $\in \{\texttt{None}, 10\}$, `min_samples_leaf` $\in \{1, 5\}$, and `max_features` $\in \{\texttt{sqrt}, \texttt{log2}\}$. For each hyperparameter configuration, I evaluated model performance using `accuracy`, `precision`, `recall`, `F1`, `ROC-AUC`, and `PR-AUC` metrics. These metrics permitted me to evaluate the model on an imbalanced dataset. The purpose and definition of these metrics was described above. The selected hyperparameters were the ones maximizing mean CV performance (Table 6).

For post-T average star-rating (regression) prediction, I use two models: Ridge regression and Random Forest Regressor. Ridge Regression is effective for working with high-dimensional, potentially correlated features, since its ℓ_2-norm penalty term stabilizes estimates and reduces variance across overlapping predictors, which in our case corresponds to interactions among reviews. I alter the regularization strength over the range $\alpha \in \{0.01, 0.1, 1, 10\}$, sampling low to high penalty terms.

I also use, Random Forest Regressor, which captures nonlinearities and interactions between predictors that a linear model is not capable of capturing. These may include review volume thresholds, nonlinearities associated with averaging star ratings, and any interactions between the business's category and its underlying review activity. I alternate a few RF hyper parameters: (`n_estimators` $\in \{200, 400\}$), maximum depth (`max_depth` $\in \{5, 8, 12\}$), minimum leaf size (`min_samples_leaf` $\in \{1, 5\}$), and feature sub-sampling (`max_features` $\in \{\texttt{sqrt}, \texttt{log2}\}$). (Table 7).

Both NN models are trained using the *same pre-processing process*, which includes identical data cleaning, encoding, scaling, and feature construction, which is done to ensure that data handling does not influence final results. All NN hyperparameters are selected with `GridSearchCV` using 3-fold CV (Table 7). In this problem, I use 3 folds because neural-network hyperparameter tuning is a more computationally-heavy process, making 3-fold CV a reasonable choice for balancing model stability and maintaining reasonable runtime. Final model performance is reported using `MSE`, `RMSE`, `MAE`, and R^2 metrics..

Neural Network hyperparameter selection.

To compare classical models to simple neural nets, I implement two multilayer perceptrons in Keras: `MLP-1` (using one hidden layer) and `MLP-2` (using two hidden layers). I intentionally restrict the NN architecture to MLPs and vary model *capacity* through the number of units within each hidden layer (and, for `MLP-2`, the pair of layer widths respectively). This design helps me explore whether additional non-linear model capacity improves performance compared to classical ML techniques. In my MLP architectures, hidden layers use the `ReLU` activation function, which is a fairly common choice. I also use `Dropout` and L2 weight decay as regularizers. I describe why the weight decay and dropout are relevant later. I also enable `BatchNormalization` for classification to remain consistent with the search spaces defined in Tables 6 and 7.

For the classification task, I encode the output layer into a `sigmoid` that computes `BinaryCrossentropy`, while for the regression problem, I define the output layer to be linear with the `MSE` loss.

Optimizer and training hyperparameters.

All neural networks are trained with the `Adam` optimizer, and I treat the learning rate as the primary optimizer hyperparameter. Because NN tuning involves continuous ranges (learning rate, dropout rate, L2 penalty), I use `RandomizedSearchCV` with *3-fold CV* instead of performing exhaustive grid search over discrete configurations. This design provides broad coverage of plausible model configurations while maintaining reasonable runtime (Tables 6 and 7). The effective number of epochs is determined via validation-based early stopping. In particular, I use `EarlyStopping(patience=4, restore_best_weights=True)` to prevent overfitting and `ReduceLROnPlateau(factor=0.5, patience=2, min_lr=1e-5)` to make additional training progress at smaller values of the learning rates if and when validation performance plateaus. This approach makes the training process more robust to validation set noise and reduces model sensitivity to the initial learning rate value.

Model selection and reporting.

For every model and task, I select hyper parameters by via the mean CV performance (Tables 6–7). I then refit the selected configuration on the full training split and evaluate the model *once* on the held-out test data set. For the classification task, I report a model-comparison table on `ALL` features and a feature-group ablation table, which notes the best model per feature group, along with a confusion matrix for the best classifier, which happens to be MLP-2. For the regression task, I report a consolidated comparison table with details on `Ridge`, `RandomForestRegressor`, `MLP-1`, and `MLP-2` methods using `MSE`/`RMSE`/`MAE` and R^2 metrics.

2.6.1 Classical Models

For all classical models, I tuned hyperparameters using GridSearchCV with a 5-fold stratified cross-validation for classification problem and a 5-fold CV for regression problem. The idea behind GridSearchCV is to estimate how well the model will generalize without using our test data set. Using GridSearchCV provided me with performance measures of all hyperparameters that I was considering. I selected the final hyperparameters based on the best average CV score. For classification, it was also important to use a stratified CV because of the class imbalance between "closed" and "open" businesses. To avoid getting folds with either too many or too little closed businesses, which would yield noisy accuracy measures, I stratified the folds so that the class proportions in each fold would be comparable to my original dataset. Using 5 folds is a popular choice within the field, since it corresponds to a more stable process than CV = 3 and much less computationally expensive than 10 folds. Due to the computational complexity of using 5-fold CV on my classification problem, I have decided to reduce the number of folds to 3 for the classification problem with Neural Nets as well as the regression problem with both classical ML methods as well as Neural Nets.

In the classification task for neural networks, I have tuned a combination of continuous and discrete hyperparameters. Continuous hyperparameters include the learning rate, the L2 penalty, and dropout regularization proportion. Discrete hyperparameters include units, the number of layers in hidden layer 1 (H1), the number of layers in hidden layer 2 (H2), and the batch_size. Since the hyperparameters space is substantially larger than in the classical ML scenario, I used RandomizedSearchCV to sample 8 random hyperparameter combinations, and then, using 3-fold CV, identify the best configuration.

I use the L2 regularization to mitigate the bias-variance tradeoff, where smaller regularization permits the model to be more flexible, whereas larger regularization imposes higher models on the model that helps control for and thus decrease model variance. Controlling for the maximum depth of decision trees allowed me to balance trees that were too complex (and thus lagged in interpretability) and trees that were too

Table 6: Hyperparameter search and cross-validation design for **classification** models.

Task	Model	Search type	CV scheme	Tuned hyperparameters
Classification	Logistic regression (L2)	`GridSearchCV`	Stratified 5-fold	Regularization strength $C \in \{0.1, 1, 10\}$.
Classification	Decision tree classifier	`GridSearchCV`	Stratified 5-fold	Tree depth and leaf size: max_depth $\in \{3, 5, 10, \text{None}\}$; min_samples_leaf $\in \{1, 5, 10\}$.
Classification	Random forest classifier	`GridSearchCV`	Stratified 5-fold	Ensemble size and tree complexity: $n_{\text{estimators}} \in \{100, 300\}$; max_depth $\in \{\text{None}, 10\}$; min_samples_leaf $\in \{1, 5\}$; max_features $\in \{\text{``sqrt''}, \text{``log2''}\}$.
Classification	MLP-1 (NN)	`RandomizedSearchCV`	3-fold	Network width and regularization: units $\in \{64, 128, 256\}$; dropout $\sim \text{Unif}(0, 0.3)$; $\lambda_2 \sim \text{LogUnif}(10^{-5}, 10^{-2})$; lr $\sim \text{LogUnif}(10^{-4}, 3 \times 10^{-3})$; batch_size $\in \{256, 512, 1024\}$; BatchNorm $\in \{\text{on}, \text{off}\}$.
Classification	MLP-2 (NN)	`RandomizedSearchCV`	3-fold	Two hidden layers: $h_1 \in \{128, 192, 256\}$, $h_2 \in \{32, 64, 96, 128\}$; dropout $\sim \text{Unif}(0.05, 0.2)$; $\lambda_2 \sim \text{LogUnif}(10^{-5}, 10^{-3})$; lr $\sim \text{LogUnif}(5 \times 10^{-4}, 2 \times 10^{-3})$; batch_size $\in \{512, 1024\}$; BatchNorm = on.

Table 7: Hyperparameter search and cross-validation design for **regression** models.

Task	Model	Search type	CV scheme	Tuned hyperparameters
Regression	Ridge regression	`GridSearchCV`	5-fold	Regularization strength $\alpha \in \{0.01, 0.1, 1, 10\}$.
Regression	Decision tree regressor	`GridSearchCV`	5-fold	Tree depth and leaf size: max_depth $\in \{8, 12, None\}$; min_samples_leaf $\in \{1, 5, 10\}$.
Regression	Random forest regressor	`GridSearchCV`	5-fold	Ensemble size and tree complexity: $n_{\text{estimators}} \in \{200, 400\}$; max_depth $\in \{8, 12, None\}$; min_samples_leaf $\in \{1, 5\}$; .
Regression	MLP-1 (NN, regression)	`RandomizedSearchCV`	3-fold	Same search space as MLP-1 classification.
Regression	MLP-2 (NN, regression)	`RandomizedSearchCV`	3-fold	Same search space as MLP-2 classification.

simple and at the cost of higher interpretability, perhaps did not capture the underlying trends among the data points nearly as well. Setting the minimum number of sample values allowed per leaf/node allowed me to avoid high-variance leafs with a very limited number of data points. Neural Networks are well-known for their capacity to find patterns in complex data, but when the datasets that are fed into NNs are sparse and high-dimensional, these powerful models tend to overfit the training data, which negatively affects future model generalizability, thus leading to significantly worse performance on the test data sets. There are a few regularization techniques that I employed in regards to Neural Networks that helped me avoid the issue of overfitting: Dropout, L2 weight decay, Early Stopping, and Learning Rate Reduction. Dropout regularization randomly selects a subset of activation functions that are masked during training, which discourages co-adaptation, where multiple hidden layer units capture similar patterns that are unique to the training data (e.g. noise). This technique was especially relevant and necessary to be applied to the Yelp Dataset because, in the data set, many of the predictor variables are binary indicator variables, which leads to inherently sparse data. Without implementing dropout, MLP would very likely overfit the training data by memorizing niche category-specific trends that would otherwise be absent from test data. Using dropout helped the model generalize better due to more robust learning process, which help de-correlate redundant inputs and to prevent the model from over-fixating on noise. L2 regularization adds a penalty that is proportional to the square of the weight, which heavily penalizes large model coefficients. Using this technique on the Yelp data was of good use because there were several correlated variables as intended, like n_reviews_total and n_reviews_recent. Without the L2 penalty, the network could assign large weights to correlated features, which would overfit the training data by placing greater emphasis on pure noise. Imposing an L2 penalty helped me distribute the coefficients more evenly across correlated variables. Furthermore, L2 regularization made the model more robust to outliers by shrinking all weights toward zero, which prevents any outlier from significantly influencing the final model output. This bonus result of Ridge regularization is especially useful for the Yelp dataset, which contains features with heavy tails. As we may recall from Figure 2, there was a small fraction of businesses with disproportionally high reviews, which makes them outliers. Ridge penalty helped the model avoid placing significant emphasis on any one of these outlying businesses, thus reducing overfitting.

Finally, I implemented an early stopping criterion and a learning rate reduction policy. Rather than setting a default number of epochs, I set a maximum possible epoch count and instead impose a validation-based with the goal of avoiding overfitting. In particular, Early Stopping continuously monitors performance on a validation set, and once progress slows down significantly, the criterion stops the training altogether. However, since the last epoch on which the model was trained may not necessary contain the best weights, I also restore the best weights over all epochs, which ensures that the final weights correspond to the ones with the best CV validation set performance. Restoring the best weights is an important design choice for the Yelp Dataset, since otherwise, MLP might further reduce training loss by fitting noise from sparse categorical data, even if no further improvements are made in the validation set. An additional criterion, ReduceLROnPlateau lowers our learning rate when training progress plateaus (hence its name). Using this function is helpful to avoid setting learning rates to extremely small values at the start of the training, which generally allows for faster initial learning. Moreover, reducing the learning rate in mid- and late training stages allows us to converge to better local minima. The two criteria work in tandem to prevent long model training and reduce model sensitivity to "Adam" settings.

I selected L2 regularization strength out of the 0.1, 1, 10 values, which correspond to strong, moderate, and weak regularization values, respectively (since I am assumign an inverse penalty). Stronger regularization lambda = 0.1 reduces variance, while weaker regularization lambda = 10 increases model flexibility, which is the classic bias-variance tradeoff. For decision trees, the hyperparameters used included max_depth and min_samples_leaf, which correspond to the maximum depth of the tree and the minimum number of values per node, respectively. Max_depth 3,5,10,None and min_samples_leaf 1,5,10. These hyperparameter values controlled the expressivity of the tree, and capping the tree depth from 3 to 10 allowed for interpretability. Finally, setting the minimum number of values per leaf helped avoid having very small, unstable leaves. Trees with depth of 3 and 5 are very shallow and highly interpretable. Trees with a depth of 10 on the

other hand are moderately deep, capable of modeling more nuanced business attributes. The trees with no maximum depth would represent the maximally flexible tree, since in this case the tree would grow until all leaf nodes are pure and cannot be split further. Cross validation helped me confirm whether this flexibility carried predictive use, or simply lead to overfitting. I also set the minimum samples per leaf to deal with the high amounts of categorical variables, since it is easy to end up with leaves that contain a handful of business in this case, which would lead to unstable predictions. Using 1 minimum sample per leaf allows for very small leaves, and is included as a high variance, default case. I also try setting the minimum at 5 and 10 trees, which prevents the model from relying on small sub-samples of our data.

For random forests, I computed four hyperparameters: `n_estimators` $\in \{100, 300\}$, `max_depth` $\in \{$`None`$, 10\}$, `min_samples_leaf` $\in \{1, 5\}$, and `max_features` $\in \{$`sqrt`$,$ `log2`$\}$. Having 100 and 300 trees would let me balance bias, and avoid high computational cost. Max depth regulated the growth of individual trees. I used 100 trees as a default number with an expected stable performance, and 300 trees to see whether adding additional trees results in a significant performance improvement, provided the dataset's complexity. As in the case of single decision trees, varying maximum forest length helps us control complexity. when no limit is imposed, the trees may grow deep, and our random forest would need to implement bagging for regularization. With the maximum depth of 10, each tree is individually constrained, which can reduce overfitting and improve interpretability of feature importances. Using minimum samples per leaf in random forests carries the same purpose as in individual decision trees. Finally, setting the maximum number of features considered at each split helps control for the within-tree randomness. Using a square root of (# features) encourages within-tree diversity, and is used as a default setting. Taking the base-2 log of (# features) leads to a greater reduction to the tree-to-tree correlation, increasing the tree diversity from the "sqrt" case.

2.6.2 Neural Networks

For MLP-1 and MLP-2 classification and regression neural networks, I fixed the optimizer to Adam, which is a default and extremely popular optimizer known for its efficiency for prediction problems with high-dimensional features. I treated the base learning rate as the main optimizer hyperparameter: For both classification and regression, I sampled lr log-uniformly from $[10^{-4}, 3 \times 10^{-3}]$ for MLP-1, and similarly for MLP-2, I sampled lr from $[5 \times 10^{-4}, 2 \times 10^{-3}]$. The lower values in this range result in more stable but slower training times, while the higher values generally result in faster convergence at the expense of increased risk of unstable convergence.

To select good combinations of learning rate and model hyperparameters described above, I use RandomizedSearchCV with 3-fold cross-validation. I found that RandomSearchCV is preferable because: 1) the hyperparameter space includes continuous variables; and 2) I use only 8 configuration training per model due to computational constraints. I further optimize the process by imposing EarlyStopping and ReduceLROnPlateau criteria during model training. Early stopping with a patience set to 4 allows me to monitor overfitting without stopping too early in the process upon encountering noise in the data. Setting patience to this value lets me reduce overfitting while maintaining reasonable training time. I also restore the best weights since the last epoch is not guaranteed to have produced the best weights for final model configuration.

I also use ReduceLROnPlateau to monitor the case in which the validation metric stops improving, which suggests that the current learning rate is too large to make progress. If that happens, I halve the learning rate, which is a common adjustment that will help the optimizer identify the optimal minimum. I intentionally impose a shorter patience of 2 than for EarlyStopping, to provide the optimizer with an opportunity to try a lower learning rate value once or twice before stopping the training. Finally, I impose a lower bound of 1e-5 to guarantee that the learning does not shrink too much, in which case learning would stagnate while still consuming epochs. All other optimizer parameters were kept at their default values, since these parameters have been tuned to solve a wide range of problems. Given that I directly altered

the learning rate and controlled the learning schedule, I did not see much practical benefit in altering more default parameters.

For classical ML models that I tested, spanning logistic regression, Ridge, decision trees, and random forests, I mainly relied on scikit-learn's built-in optimizers.

3 Results

3.1 Classification Results

Table 8 reports test set performance for all classical ML classification models using the "ALL" features. RF achieves the strongest performance across all classical models tested, with the ROC-AUC score of 0.877 and PR-AUC of 0.705, which suggests that RF offered the best precision-recall tradeoff for predicting "Closed" businesses. What is worth noting is that its recall is the lowest among all other models. This suggests that, at the default model threshold, RF has higher precision and lower sensitivity measures. Logistic Regression and Decision Tree classifiers achieve the opposite results, with much higher recall but lower precision than RF, which means that these two models label more businesses in the data set to be "Closed". The calibrated Ridge classifier sits somewhere in between, with moderate precision of 0.664, which is better than the logistic and decision tree models but not quite as good as RF; and a recall of 0.271, which is the lowest recall across all models surveyed, and by a substantial margin. However, this result is expected from a model that rarely predicts business closures at default threshold, since Ridge is not the best model at predicting minority class instances. As is evident, neural nets perform quite similarly to RF, with MLP-1 having slighly lower performance and MLP-2 sitting just above RF in terms of ROC-AUC, PR-AUC, F1, and Recall. Across all models tested, a two-layer Neural Network with 256 neurons in the hidden layer, a drop-out rate of 0.2345, and the l2-penalty of 0.0006, MLP-2, achieved the best performance, ROC-AUC-wise. However, the results for the best configuration of MLP-2 suggest that there is a trade-off between the model having higher recall, which allows it to capture more closed businesses, high ROC,PR-AUC values, but having a lower prevision and lower overall accuracy due to more frequent "Closed" predictions. MLP-2 best configuration achieved a CV AUC of 0.879 (not reported, in the code).

Table 10 lists the confusion matrix achieved for the best MLP-2 configuration. Out of 6,120 closed businesses from the dataset, the model correctly predicted 4,762 of them, which roughly corresponds to a 77.8% recall. The model misclassified the remaining 1,358 businesses as false negatives, which aligns with the high model recall presented in Table 8. Out of all open businesses, the model correctly predicted 19,395 as true negatives and misclassiffied the remaining 4,092 as false positives, which justifies why MLP-2's precision was substantially worse than that achieved by Random Forest.

Table 9 compares classical ML method performance by considering one feature group at the time. As was emphasized multiple times throughout the paper, review-derived features drove business closure predictions for the large part. Using REVIEW features alone resulted in a ROC-AUC of 0.862, which was slightly below the highest ROC-AUC value, achieved for ALL features by a RF. On the other hand, geographic (GEO) features carried the least predictive signal, with the lowest ROC-AUC of 0.607 and PR-AUC of 0.301., with BUSINESS attributes doing slighly better and achieving ROC-AUC of 0.704 and PR-AUC of 0.352. These results are consistent with our logical expectations that behavioral predictor variables that reflect customer engagement, like all features derived from user reviews, carry the greatest predictive power in business closure identification.

Table 8: Classification performance on the held-out test set using **ALL** features.

Model	Accuracy	Precision	Recall	F1	ROC-AUC	PR-AUC
Logistic (L2)	0.755	0.446	0.764	0.563	0.843	0.629
RidgeClassifier (cal.)	0.821	0.664	0.271	0.385	0.777	0.520
Decision Tree	0.756	0.451	0.831	0.585	0.853	0.613
Random Forest	0.862	0.764	0.478	0.588	0.877	0.705
MLP-1 (1 hidden layer)	0.803	0.516	0.786	0.623	0.875	0.700
MLP-2 (2 hidden layers)	0.816	0.538	0.778	0.636	**0.879**	**0.716**

Table 9: Ablation over feature groups on the test set using the **best classifier per group** (model shown).

Feature group	Best model	ROC-AUC	PR-AUC
GEO	Random Forest	0.607	0.301
BUSINESS	Logistic (L2)	0.704	0.352
REVIEW	Random Forest	0.862	0.651
ALL	Random Forest	**0.877**	**0.705**

Note: Best models were determined by highest ROC-AUC values, and the ties were broken by PR-AUC.

Table 10: Confusion matrix for the best-performing classification model (MLP-2) on the test set.

	Predicted	
True	Open	Closed
Open	19,395	4,092
Closed	1,358	4,762

3.2 Regression Results

Table 11 compares all tested regression models on the task of predicting post-T average star ratings for businesses with sufficient (3 or greater) reviews. Across two classical ML and NN models tested, all models receive comparable, average performance based on the four performance metrics as described in the Metrics section. The best model, RF (with the best parameters), achieves an R^2 of just 0.5293, which means that the model barely explains 53% of the variance present in the dataset. MLP-1 comes in close second with an R^2 of 52.6%. Ridge Regression and MLP-2 trail behind. MLP-1 achieves the lowest MAE error of 0.6205. RF achieves the lowest MSE and RMSE errors. Since RF dominates the other three models across three of the four metrics studied, it is reasonable to conclude that RF is a better model for conducting the regression task.

Table 11: Regression performance on the held-out test set (ALL features).

Model	MSE	RMSE	MAE	R^2
Ridge Regression (best α)	0.6647	0.8153	0.6275	0.5200
Random Forest (best params)	**0.6518**	**0.8074**	**0.6210**	**0.5293**
MLP-1	0.6565	0.8103	0.6205	0.5259
MLP-2	0.6988	0.8359	0.6528	0.4954

Notes: Target = post-T average star rating for businesses with ≥ 3 post-T reviews. Preprocessing is identical across all models. No explicit feature selection is used for regression.

3.3 Feature Importance and Interpretation

To add an interpretability component to the classification problem of business closure, I also present Table 12. Table 12 reports the coefficients with the largest magnitude from sparse L1 logistic regression with standardardized features. The obtained coefficients signal that two types of categories in particular strongly inform business closure risk: customer engagement and the restaurant category. The most relevant coefficient from the table is the one for the number of recent reviews, which has the highest negative magnitude. This numerical results supports my argument that customer engagement is an integral and accurate indicator of the businesses remaining afloat. Many of the factors that increase restaurant closure turn out to be cuisine denominations, like Asian, Middle Eastern, Korean, and others. Prior to claiming that these findings link to a causal relationship between the restaurant of a particular type being more likely to close, one first needs to take into account the differences in market structure across businesses that belong to these categories as well as the customer engagement unique to each business.

Table 12: Top positive and negative coefficients from the L1-logistic regression (standardized). Positive values increase the log-odds of closure; negative values decrease it.

Top increases (positive β)

Feature	Coefficient
cat_Vegetarian	2.554
cat_Asian Fusion	2.391
cat_Vietnamese	2.346
cat_Middle Eastern	2.332
cat_Korean	2.311
cat_Barbeque	2.296
cat_Mediterranean	2.204
cat_Cuban	2.201
cat_Greek	2.189
cat_Indian	2.174

Top decreases (negative β)

Feature	Coefficient
num_n_reviews_recent	-2.645
cat_Religious Organizations	-2.190
cat_state_FL	-1.954

4 Conclusion

4.1 Summary of Findings

This study explored two prediction tasks using a spatial time design: 1) a binary classification problem of evaluating business closures after T using only information available before T; and 2) predicting `post_T` average star ratings for business that remained open and had at least 3 post-T reviews. Upon evaluating classical ML methods and Neural Networks, using four features groups, {`GEO`, `BUSINESS`, `REVIEW`, `ALL`}, three results stand out.

Review-derived features performed extraordinarily well in predicting business closures.

`REVIEW` features alone yielded ROC-AUC = 0.862 and PR-AUC = 0.651, respectively, substantially outperforming `GEO`- and `BUSINESS`-derived feature groups, which yielded (ROC-AUC, PR-AUC) values of (0.607, 0.301) and (0.704, 0.352), respectively. As expected, the group combining all three feature groups, `ALL`, performed the best, reaching ROC-AUC = 0.877 and PR-AUC = 0.705, respectively. Given that the business closure rate was roughly 21% in the dataset, a PR-AUC of 0.705 reflects a substantial improvement over baseline results, indicating that `REVIEW`-derived features were most informative in predicting business closures, while geographic and business attributes added useful, albeit less informative, signals.

Best classifiers depended on the metric used.

Among the reported classical models evaluated on `ALL`, Random Forest offered the strongest overall accuracy, achieving Accuracy = 0.862, ROC-AUC = 0.877, PR-AUC = 0.705, F1 = 0.588, Precision = 0.764, and Recall = 0.478. However, with change in objective comes the change to the choice of models. If we wanted to prioritize recall to capture as many closures as possible, Logistic (l2) and Decision tree models would have been our top choice, (Logistic: Recall = 0.764, F1 = 0.563; Decision Tree: Recall = 0.831, F1 = 0.585), at the expense of precision. If instead we wanted to prioritize precision, Random Forest and Calibrated Ridge Classifier (Precision = 0.664, Recall = 0.271, ROC–AUC = 0.777, PR–AUC = 0.520, accuracy = 0.821) would have been our model selections, at this time at the expense of lower recall and ROC-AUC, PR-AUC values, with a rather strong accuracy.

A quick note on Calibrated Ridge Classifier.

Using `CalibratedClassifierCV` with `RidgeClassifier` was an intentional choice done to obtain corrected probabilities. The model's lower AUC values compared to those obtained by RF show expected discrimination gaps between linear and nonlinear models. Furthermore, the model's low recall when using the default 0.5 cutoff reflects the class imbalance in the Yelp data rather than a modeling error. One potential solution to mitigate this in future studies is selecting a better cutoff using a validation set or implementing cost-sensitive criteria. Even with these adjustment set in place, RF remains the strongest model on the classification task. The classification results indicate that, when using `ALL` features and standard pre-processing, the business closure problem is highly predictable, with random forests achieving superior performance among all tested models.

Average Star Rating Prediction achieved moderate performance, with RF and simple MLP architectures yielding similar results.

For businesses with more than 3 post-T reviews, Random Forest provided the best generalization across all models tested, with RMSE = 0.8074, MAE = 0.6210, and $R^2 = 0.5293$. A 1-hidden layer MLP (MLP-1) came in close second, slightly lacking in R^2 performance, achieving $R^2 = 0.5259$, MSE = 0.6565, and RMSE = 0.8103. Ridge regression was trailing behind with an $R^2 = 0.52$, with a deeper, 2 hidden-layer MLP (MLP-2) having the worst performance, with an $R^2 = 0.4594$. Overall, these results indicate meaningful but imperfect explanatory power of pre-T signals alone. In future studies, we might try to further improve the regression model's explanatory power by adding rich text features derived from review contents using NLP methods.

4.2 Future Directions

This study introduces a time-spatial design for predicting post-T business closures and average star ratings for those businesses that remain open. My results demonstrate that features derived from customers' reviews carry the greatest predictive signal for business closure, and that tree ensembles and simple NN multi-layer perceptron architectures achieve comparable performance for predicting star reviews. There exist multiple extensions to this project that researchers might want to explore in future studies. One such possible change is adding XGBoost to improve Random Forest performance for structured data. Moreover, future researchers might want to delve deeper into feature engineering using the same cutoff strategy to the one that I had developed in this paper. In particular, future studies might want to use NLP processing to conduct a thorough analysis of review data by aggregating pre-T n-grams, overall lexicon sentiment, and embeddings for each business. Future studies might also want to correct the selection criterion imposed in my regression study. That is, to further reduce bias in rating predictions, future studies might instead want to explore inverse-probability weighting rather than a strict cutoff threshold. The researchers might further want to introduce categorical embeddings to my MLP architectures, which are generally used to solve the problem of representing high-dimensional, sparse data.

4.3 AI Use Statement

I used the help of ChatGPT Thinking Models 5, 5.1, and 5.2 to come up with and debug the code for this assignment, as it was permitted by course instructors. I used the following prompt structure to compile the code: "I am working with the Yelp Open Dataset for a class project. For that project, I need you to help me write a function that would [provide descriptive statistics and figures about the dataset, parse five .JSON files inside my directory, preprocess data, complete my classification task, complete my regression task]. The restrictions are as follows: [Pasted the entire project description] and [Pasted the write-up pieces that were relevant for that particular task, like my regression design for the regression code prompt]." I prompted the chatbot step-by-step, for different pieces of the assignment. Upon getting the code, I debugged any errors and added all assignment pieces together into a single notebook, which I then ran.

Furthermore, as per instructor's instructions, I used AI (ChatGPT thinking model 5) to conduct literature search to identify papers related to my research topic.

References

[Li and Zhang, 2014] Li, C. and Zhang, J. (2014). Prediction of yelp review star rating using sentiment analysis. Cs229 project final report (autumn 2014), Stanford University.

[Lu et al., 2018] Lu, X., Qu, J., Jiang, Y., and Zhao, Y. (2018). Should i invest it? predicting future success of yelp restaurants. In *Proceedings of the Practice and Experience in Advanced Research Computing (PEARC '18)*, New York, NY, USA. Association for Computing Machinery.

[Pan et al., 2018] Pan, C., Gao, Y., and Luo, Y. (2018). Machine learning prediction of companies' business success. Cs229 course project report, Stanford University.

[Vallapuram et al., 2021] Vallapuram, A. K., Nanda, N., Kwon, Y. D., and Hui, P. (2021). Interpretable business survival prediction.