# Business Failure Prediction on Yelp Dataset

DS-GA 1003 Machine Learning Final Project Report

Spring, 2017

I-Wu Lu, Ying-An Lai, Chun-Yi Yang

# Contents

# 1 Problem

Owning a business, shopkeepers always have concerns about whether their business can thrive in the future. It will be valuable to know the risk of closure in advance. At the same time, as the growing of online evaluation platforms such as Yelp, people now have new ways to share experiences. We can collect useful information from these crowd-sourced resources as an evaluation of a business as well. While there are various reasons that could lead to store shutters, we will approach this task using business attributes, user information and reviews on Yelp.

To enable practical usage, we consider the real-world scenario where 8 out of 10 new businesses fell after the first 18 months. From this aspect, it is priceless if we can identify these two survivals beforehand. Therefore, we build a business failure model to predict if a business will survive another half-year, defined as **test interval**, based on data accumulated in the first running year, defined as **observation interval**. This is a classification problem with two classes.

# 2 Dataset

We use data from "The Yelp Dataset Challenge 2017", which is an open data competition held by Yelp. The dataset contains 140k business and 4.1 millions reviews across 11 cities in four countries around the world. The total data size is around 4.8 GB. The original dataset has three major json files of our interest: business, reviews and users, which are mapped together using encrypted keys. Business data including information such as `latitude`, `longitude`, `attributes`, `category`, `hours` and the most important labeled outcome `is_open`. Review includes `text`, `date`, and `stars`. User information such as `review_count`, `yelping_since`, and `num_compliment_received` are also included.

Yelp Dataset is informative and representative for research yet some concerns need to scrutinized. For example, the exact time when our target variable `is_open` is set is not included in dataset, which means careful definition is needed to certain that we will not add extra noise to our data. More details will be addressed later in Section 3.

In addition, as mentioned in Section 1, we only extract data in the observation interval to train our models. To make sure we do not "cheat" by peeking future data, we have to extract data in interval before further manipulation. Any data that contains future information, such as useful and funny attributes of reviews marked by later visiting customers should be eliminated.

# 3 Data Engineering

## 3.1 Target Variable

In original data, each business has a flag **`is_open`** set to zero if it is permanently closed. However, it only reflects the status at the moment, not exactly the time we are interested in. Thus, we defined our own target variable (TV) a binary value. TV = True indicates whether a firm will shutdown within the test interval, which is 6 months after our 1-year observation interval; otherwise TV = False.

For each business, we mark the date of the first review as its start time `start_t`, and the date of the last review as its end time `end_t`. These two timestamps will

help us determine the status of each business. We count the observation period from `start_time`, and the value of TV is determined at

$$TV\_time = \texttt{start\_time} + observation\_interval + test\_interval.$$

Table 1 shows how we get the value of TV:

When end_t $> TV\_time$, shown in Figure 1-(a)(b). Users leave their reviews on this business after $TV\_time$, we can infer the business is still open. Therefore, we set TV = False in these cases. On the other hand, when end_t $< TV\_time$, we can set TV = True if **is_open** = 0, because it closed down during test interval, as shown in 1-(c). However, there is a case as shown in Figure 1-(d), in which we can not use (**is_open** = 1 and end_t $< TV\_time$). Since we have no information on whether it can survive after test interval in this case, we simply omit data in the case.

Figure 1: Timeline of different conditions

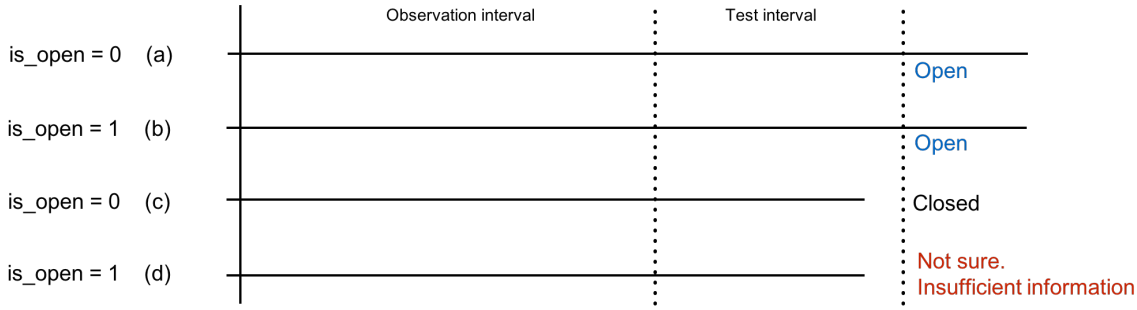



Table 1: Target variable under different conditions

| | is_open = 1 | is_open = 0 |
|---|---|---|
| end_t $> TV\_time$ | False | False |
| end_t $< TV\_time$ | Not applicable | True |

## 3.2 Data Preprocessing

The dataset needs some modification before fed into learning models. The procedures are illustrated in Figure 2. The detailed methods of each step are listed as follows.

1. We first deal with missing values. Some of them can be easily filled using other fields; such as missing value in `city` can be inferred from `longitude` and `latitude`. However, others such as `categories` and `attributes` don't

Figure 2: Data preprocessing steps

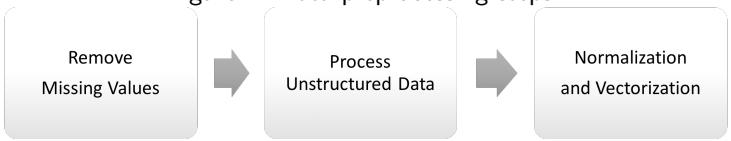


Table 2: Features applied to our models

| index | Feature | Description |
|---|---|---|
| 0-27 | state | state where the business locates |
| 28 | name_size | the length of business name |
| 29 | stars | average stars of reviews to this business |
| 30-205 | attributes | business attributes |
| 206-406 | category | business category |
| 407-575 | hours | business opening hours |
| 576 | elite_cnt | number of elite visits |
| 577 | popularity | number of reviews received in unit time |
| 578 | star_trend | the general movement of review stars given over time |
| 579 | local_pref_tot | |
| 580 | local_pref_cat | |
| 581 | competitor_tot | |
| 582 | competitor_cat | |
| 583 | competitor_ratio | |
| 584 | pos_score | average positive sentiment score of reviews |
| 585 | neg_score | average negative sentiment score of reviews |
| 586-685 | top_adj_occ | occurrence frequency of 100 most frequent adjectives in reviews |
| 686-785 | top_noun_occ | occurrence frequency of 100 most frequent nouns in reviews |

have the luck. For categorical data, we add an extra feature as a mask and set its value to 1 if missing occurs. For numerical features, we do not only add a mask but also fill it with the arithmetic mean of that features.

2. Next, we process the unstructured data in our datasets. We use **NLTK**, a python natural language Toolkit to process text in reviews. First, we clean up the text and tokenized it into words. Then we part-of-speech tag each word and use a subset of the keywords (labeled adjectives and nouns) that appear most frequently among all reviews to generate feature vectors given a single review. We also do sentiment analysis for each review, more detailed method in sentiment analysis will be further discussed in Section 4.

3. In the end, we adapt normalization for numerical attributes and use one-of-N encoding method for categorical attributes.

# 4 Features

Table 2 is the complete list of features and their brief description. Our features include static features and dynamic features. Static features, such as `state` and `name_size`, are determined once the business started, while dynamic features, such as `popularity` and `stars`, are generated using the average among our 1-year observation interval.

## 4.1 Text-based Features

A group of features, indexed from # to # in Table 2, are extracted from review contents. In order to transform the review content from plain text to index score,

we utilize a python library **TextBlob** to generate sentiment scores.

Sentiment scores are ranging from 0 to 1, indicate how positive or negative the reviews of the business are. We average all positive sentiment scores as `pos_score` and negative sentiment scores as `neg_score` so that they don't cancel out. We also compute the subjective/objective score of reviews but the feature is too sparse and sometimes make noise to the model.

We go through the instances labeled "True" (failed) in the training set and find 100 most common nouns and adjectives as key words. For each instance, we calculate the occurrence frequency of each word in the reviews of each business and generate its word count `top_adj_occ` and `top_N_occ` which are both 100-dimensional vectors.

### 4.2 Features of Temporal Information

Temporal information reflects the progress of a business as time passes. For example, `star_trend` is applied to capture the information from users' perspective. If we consider the stars of reviews with timestamps as a time series, what we are curious about is its trend. We ran linear regression on the time series and solve the best fit of formulation

$$stars = at + b \tag{1}$$

where stars and t are the rating and its corresponding timestamp perspectively. The slope of the line ($a$ in Equation 1) is the trend of stars.

### 4.3 Features of Spatial Information

Besides temporal information, we collect useful information from geometrical data as well. For example, we define the difference between stars of reviews given by the same user left on a business within 1 mile as `local_pref_tot`. It shows users' preference over these stores. If there is no user who reviews both stores, the value will be 0 as no preference. `local_pref_cat` is another feature with similar concept yet count only if these stores share one or more categories.

`competitor_tot` is the count of stores within 1 mile, while `competitor_cat` is the count of stores in same category within 1 mile. And `competitor_ratio` is just the ratio of these two variables.

## 5 Evaluation Metrics

A key observation of our dataset is that output classes are imbalanced: the survived businesses greatly outnumber the shutdown ones, where the ratio of true false instances is around around $2\%$ $(2297/115350)$. Therefore, we need to find evaluation metrics different from accuracy to avoid our model being dominated by majorities.

We choose Cohen's Kappa coefficient [2], a statistic that measures inter-rater agreement for categorical items. It is more robust than simple percent agreement calculation. The following is a short example showing how it works.

Consider the confusion matrix in Table 3. The accuracy of this prediction is given by

$$p_a = \frac{a+d}{a+b+c+d}$$

Table 3: Example confusion matrix

| | Predict True | Predict False |
|---|---|---|
| Real True | a | b |
| Real False | c | d |

To exclude the part of random agreement, $p_e$ is introduced as

$$\begin{aligned} p_e &= Pr\{\text{both of pred. and real say true}\} + Pr\{\text{both of pred. and real say false}\} \\ &= \frac{a+c}{a+b+c+d} \cdot \frac{a+b}{a+b+c+d} + \frac{b+d}{a+b+c+d} \cdot \frac{c+d}{a+b+c+d} \end{aligned}$$

Finally, we get Kappa coefficient by

$$\kappa = \frac{p_a - p_e}{1 - p_e}$$

Kappa coefficient reaches its theoretical maximum value of 1 only when both the prediction and the real results codes are the same, that is, when corresponding row and column sums are identical ($b = c = 0$). If there is a random prediction, assume that

$$\frac{c}{a} = \frac{d}{b} = r \tag{2}$$

Then,

$$p_e = \frac{(a+b) \cdot a(r+1)}{(a+b)^2(r+1)^2} + \frac{(a+b)r \cdot b(r+1)}{(a+b)^2(r+1)^2} = \frac{a+br}{(a+b)(r+1)} = p_a \tag{3}$$

Therefore, $\kappa = 0$ when the prediction is random. To sum up, a good prediction should have $\kappa > 0$ (at least better than random) and as close to $1$ as possible.

# 6 Modeling

## 6.1 Baseline

Since we are dealing with classification problem, we select the class that has the most observations (in our case is 0 which means store is going to survive) and use that class as the result for all predictions. From the evaluation method mentioned in Section 5, this implies $b = d = 0$ in Table 3,

$$p_a = p_e = \frac{a}{a+c} \Rightarrow \kappa = 0 \tag{4}$$

It is no surprise to get this result because we have chosen a metric that is sensitive to skewed datasets. A valuable model should be able to exceed this score.

## 6.2 Models

We begin our modeling with various machine learning algorithms, including:

- Logistic Regression
- Linear SVM

- Random Forest
- AdaBoost

Among all models, Random Forest has the best performance in Kappa metrics; therefore, we base our model on Random Forest and work to improve its performance with Principle Component Analysis and parameter tuning.

### 6.3 Principle Component Analysis

We then added Principle Component Analysis (PCA) in our pipeline before Random Forest for dimensionality reduction. PCA performs a linear transformation on our 786-dimensional features vector into a new coordinate system with projections of original observations. We treated the number of PCA components as an hyper-parameter, and the results are shown below:

Table 4: PCA + Random Forest

| **Number of Components** | **8** | **16** | **32** | **64** | **128** | **256** | **w/o PCA** |
|---|---|---|---|---|---|---|---|
| **Kappa** | 0.112 | 0.146 | **0.158** | 0.141 | 0.100 | 0.060 | 0.156 |

### 6.4 Parameter Tuning

We performed an exhaustive grid search through hyper-parameter space and evaluated the the model performance using 5-fold cross-validation. For Random Forests, search through the parameter in the following range:

```
"Random_Forest": { "n_estimators": [50,100,200,300,500], \
                   "max_depth": [None, 3], \
                   "max_features": ["log2", "sqrt"], \
                   "min_samples_split": [10*i for i in range(7,15)], \
                   "bootstrap": [True], \
                   "criterion": ["gini", "entropy"], \
                   "class_weight": ["balanced", None] \
                   }
```

We got the best result of Random Forest with the following parameter settings:

```
RandomForestClassifier(bootstrap=True, class_weight='balanced',
            criterion='entropy', max_depth=None, max_features='sqrt',
            max_leaf_nodes=None, min_impurity_split=1e-07,
            min_samples_leaf=1, min_samples_split=100,
            min_weight_fraction_leaf=0.0, n_estimators=500, n_jobs=-1,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

## 7 Results

### 7.1 Model Comparison

Table 5 demonstrates the best performance of each model. Random Forest gives the best result of 0.156 by Kappa coefficient and the AUC is 0.564, while Logistic

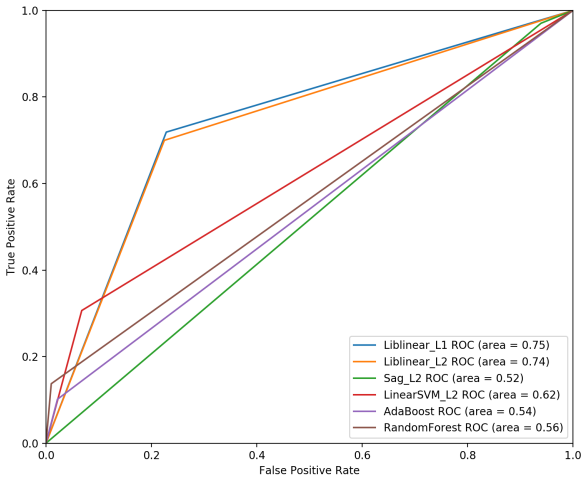
Table 5: Model Comparison

| | TN | TP | FN | FP | Precision | Recall | AUC | Kappa |
|---|---|---|---|---|---|---|---|---|
| **LR** | 17644 | 340 | 133 | 5218 | 0.061 | 0.719 | 0.745 | 0.078 |
| **SVM** | 21719 | 111 | 362 | 1143 | 0.085 | 0.101 | 0.592 | 0.102 |
| **RF** | 22633 | 65 | 408 | 229 | 0.221 | 0.137 | 0.564 | 0.156 |
| **AdaBoost** | 22365 | 497 | 48 | 425 | 0.088 | 0.101 | 0.540 | 0.074 |

Regression has the best AUC. The reason of this inconsistence is that Kappa score emphasizes agreement between prediction and the true value. This phenomenon can be further illustrated by ROC curves.

ROC curves of implemented models are shown in Figure 3. As we choose Kappa coefficient as our performance metric, our best models tend to prefer lower false positive, which make our curves more leaning to lower left corner.

Figure 3: ROC curves




## 7.2 Feature Importance

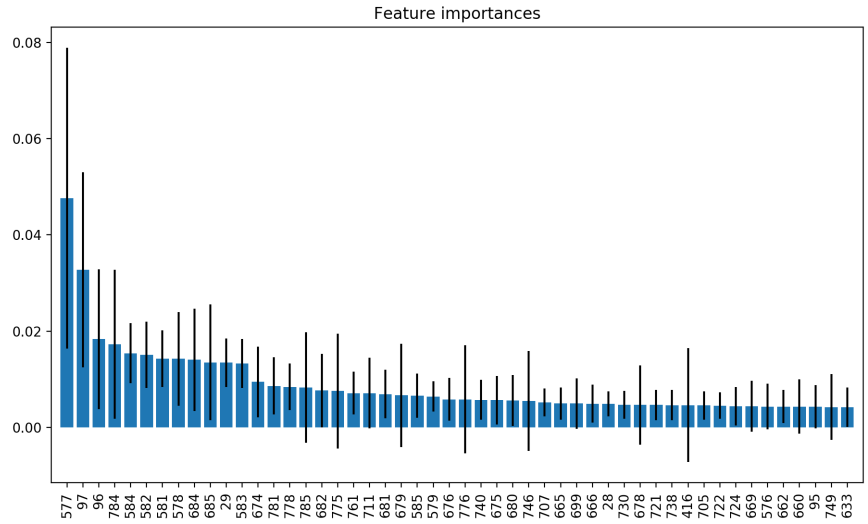
Figure 4 shows the top 50 important features from the Random Forest model. Table 6 lists the top 15 features and their importance. We observe that the list contains 7 features engineered from the sentiment analysis, 3 features from spatial information and 1 feature from temporal information. Only 4 of them are primitive feature in Yelp datasets.

# 8 Discussion

## 8.1 Challenges

- Imbalanced Classes in Dataset

Figure 4: Top 50 important features




As mentioned in Section 5, we leverage Kappa evaluation metrics to prevent the negative effects of imbalance on classification.

- Efficiency of Data Processing

  Some of our features are compute-intensive due to the complexity of spacial and temporal factors. Calculating the distance between two business using longitude and latitude is one of this example. Given this property and the large size of dataset, we use Spark in Python to speed up feature generation.

## 8.2 Future Work

One direction we want to take is to generalize our evaluation using the concept of survival analysis. In survival analysis, we replace current binary target variable with a random variable and try to fit this random variable model. In addition, we can improve our model by taking geographical information into account.

# 9 Conclusion

Utilizing crowd-sourced data for prediction is a common application of machine learning in recent decades. Collecting data from Yelp users, our model can come up with promising results (Kappa coefficient $= 0.156$) in business failure classification problems. In addition, PCA can further improve our performance by projecting our features into lower dimensions.

Although our model and evaluation is built using Yelp datasets, our model is not limited to restaurants. There are various crowd-sourced datasets which can be future research directions. Online hospitality service provider Airbnb, for example, can adapt our model to determine whether a host will last for long time. Choosing

Table 6: Top 15 features

| Rank | Feature Index | Feature Name | Importance |
|---|---|---|---|
| 1 | 577 | popularity | 0.048 |
| 2 | 97 | Attr(dessert) | 0.033 |
| 3 | 96 | Attr(lot) | 0.018 |
| 4 | 784 | NOUN(place) | 0.017 |
| 5 | 584 | pos_score | 0.015 |
| 6 | 582 | competitor_cat | 0.015 |
| 7 | 581 | competitor_tot | 0.014 |
| 8 | 578 | star_trend | 0.014 |
| 9 | 684 | ADJ(great) | 0.014 |
| 10 | 685 | ADJ(good) | 0.013 |
| 11 | 29 | Stars | 0.013 |
| 12 | 583 | competitor_ratio | 0.013 |
| 13 | 674 | ADJ(new) | 0.009 |
| 14 | 781 | NOUN(service) | 0.009 |
| 15 | 778 | NOUN(time) | 0.008 |

the best model as well as the most important features can bring potential profits to both hosts and the service provider. In this way, our model can make biggest impact.

# References


[1] Five Reasons 8 Out Of 10 Businesses Fail Forbes, Sep 2013

[2] J. Cohen. *A coefficient of agreement for nominal scales* Educational and Psychological Measurement, 20, 37-46, 1960.