A Machine Learning based Yelp Recommendation System

Pranav Bhardwaj
Department of Statistics
Stanford University
pranavb@stanford.edu

Nicolas Bievre
Department of Statistics
Stanford University
nbievre@stanford.edu

Frederik Johan Mellbye
Institute for Computational and Mathematical Engineering
Stanford University
frederme@stanford.edu

***Abstract*—The Yelp Data Challenge has produced hundreds of academic papers using modern techniques to extract insights from user, business, and review data. A variety of binary classification models are used to predict if a user will rate a business highly, with the downstream goal of generating business recommendations for users. Linear models, tree based models, and neural networks were used, with an XGBoost classifier achieving the best performance and generalizing well to future data.**

***Index Terms*—Machine Learning, Deep Learning, Boosted Trees, Recommendation System**

Code available: github.com/fredmell/CS229Project

I. Introduction

Yelp is one of the most widely used restaurant and merchant information platforms in the United States. Information on businesses is crowd-sourced from users in the form of reviews. In order to gain insights on this wealth of data, Yelp released a data set with the 13^{th} round of its data challenge. The Yelp data set consists of over 6 million reviews of nearly 200,000 businesses from 1.7 million users. This includes business information, user information, and reviews containing a star rating.

The recommendation of a business to a user can be framed as a binary classification problem. By aggregating the ratings from 1 to 4 stars as negative labels and isolating the ratings of 5 stars as positive labels, a near class balance of 44% positive labels and 56% negative labels is observed, as seen in Table I. Correctly predicting positive labels then corresponds to successful recommendations of businesses to users.

TABLE I
Proportion of each star rating given

Rating (Stars)	Proportion
1	15%
2	8%
3	11%
4	22%
5	44%

The input to the learning algorithms are features corresponding to a given user and a given business. A variety of different binary classification algorithms, including logistic regression, gaussian discriminant analysis, decision trees, random forests, boosted trees, and neural networks, are used to output the predicted probability of a 5 star rating between the given user and business pair.

II. Related Work

Recommendation Systems (RSs) are tools and techniques providing suggestions for items to be of use to a user. Their emergence is largely due to a rather simple observation; users rely on suggestions from other people when making decisions. An example of this is recommendation letters in recruitment processes. With the growth of websites with large amounts of options, a need for user-specific filtering has also emerged. The sheer amount of data makes it increasingly difficult to arrive at ideal choices [6].

Two main paradigms have emerged. Content-based RSs, which attempts to recommend items similar to those a user has chosen previously and Collaborative RSs, which makes recommendations based on what similar users have chosen [5]. Machine learning algorithms naturally lend themselves to these types of problems with their ability to leverage large amounts of data to model the vast network of users and options.

III. Data

A. Training Split

A temporal train-validation-test split is made roughly corresponding to 80-10-10 proportions. This temporal split allows for "implementation" of the recommendation system, and an estimate on how the system's performance generalizes to the future Yelp community. Information on the data split is presented in Table II.

B. Data Processing

Business categories, such as *Restaurants* and *Nightlife*, with more than 100000 observations are one-hot encoded as predictors. Business attributes with a clear binary relationships, such as providing *WiFi*, are also one-hot encoded. Business attributes with an ordinal relationship, such as *NoiseLevel* with categories quiet, average, loud, and very loud, are encoded as continuous with unit distance. Missing information in the business data was imputed with the mode of the predictor.

TABLE II
TRAIN-VALIDATION-TEST SPLIT

	Start Date	End Date	Reviews	Users	Businesses	Five Star Reviews
Train	10/19/2004	1/1/2018	4,903,362	1,259,558	125,159	43.1%
Validation	1/1/2018	6/7/2018	527,276	262,764	75,605	51.5%
Test	6/7/2018	11/14/2018	527,276	274,722	75,254	51.8%

A small amount of user features were readily available, such as *Avg. Stars* (average rating given by the user) and compliments. A user's number of friends in the Yelp system was extracted from the data and used as a predictor.

Input information to learning algorithms, x, is a concatenation of the user features, x_u, and the business features x_b. The label y is positive (1) if the user u left the business b as 5 star review, and negative (0) otherwise. A single example i is then:

$$\begin{bmatrix} x_u^{(i)} \\ x_b^{(i)} \end{bmatrix} = x^{(i)} \in \mathcal{R}^d, y^{(i)} \in \{0, 1\} \tag{1}$$

Where $d = 166$ is the number of predictors.

IV. METHODS

A variety of machine learning algorithms are used to perform binary classification and subsequently generate recommendations. In the following description of learning algorithms used let n be the number of training examples.

A. Linear Models

1) Logistic Regression: In logistic regression, the predicted probability of a 5 star rating is of the form:

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}} \tag{2}$$

Where an affine transformation, $\theta^T x$, of the input user and business features x is mapped onto the probability space by the sigmoid function. To fit the parameters θ of the model, log-likelihood $\ell(\theta)$ is maximized.

$$\ell(\theta) = \sum_{i=1}^{n} y^{(i)} \log h(x^{(i)}) + (1 - y^{(i)}) \log(1 - h(x^{(i)}) \tag{3}$$

B. Generative Learning Algorithms

1) Gaussian Discriminant Analysis: In Gaussian Discriminant Analysis (GDA), it is assumed that $p(x \mid y)$ follows a multivariate normal distribution. The model is of the form:

$$y \sim \text{Bernoulli}(\phi) \tag{4}$$

$$x \mid y = 0 \sim \mathcal{N}(\mu_0, \Sigma) \tag{5}$$

$$x \mid y = 1 \sim \mathcal{N}(\mu_1, \Sigma) \tag{6}$$

Here the parameters are ϕ, the probability of a generated data point having a positive label, and μ_0, μ_1, and Σ which define the gaussians generating negative and positive examples, respectively. These parameters are fit to the model by maximizing the log-likelihood of the data given by:

$$\ell(\phi, \mu_0, \mu_1, \Sigma) = \log \prod_{i=1}^{n} p(x^{(i)} \mid y^{(i)}; \mu_0, \mu_1, \Sigma) p(y^{(i)}; \phi) \tag{7}$$

By maximizing these parameters with a shared covariance matrix Σ, a linear decision boundary, at which $p(y = 1 \mid x) = 0.5$, is created.

C. Tree-Based Models

1) Decision Trees: In Decision Trees, the input space $\mathcal{X}$ is repeatedly split into two child regions by thresholding a single feature. Given a parent region R_p, and two children region R_1 and R_2 resulting from a given split on a single feature, the decrease in loss is given by:

$$L(R_p) - \frac{|R_1| L(R_1) + |R_2| L(R_2)}{|R_1| + |R_2|} \tag{8}$$

Partitions are made greedily, i.e., the feature maximizing this decrease in loss is selected by the algorithm at each step. There are many viable loss functions to use. The work presented uses cross-entropy loss, which is of the form:

$$L_{cross}(R) = -\sum_c \hat{p}_c \log_2 \hat{p}_c \tag{9}$$

Where $\hat{p} \log_2 \hat{p} = 0$ if $\hat{p} = 0$.

2) Random Forests: Random Forest Classifiers are constructed by bagging Decision Tree Classifiers which are trained using a random subset of features. The algorithm for constructing a random forest classifier is described in Algorithm 1 [3].

Algorithm 1 Random Forest Classifier Construction

1: **for** $b = 1, \ldots, B$ **do**
2: Draw a bootstrap sample Z_b^* of size n from the training data
3: Select a random subset S_b of size m of the d features
4: Train a Decision Tree Classifier T_b using variables S_b and training data Z_b^*
5: **end for**

To predict on a new example x', the majority vote from the Decision Tree Classifiers T_b is returned.

3) AdaBoost: AdaBoost classifiers sequentially apply base classifiers to modified versions of the data. The initial version of the data has uniform weighting. In each successive iteration m, observation weights are modified to place more weight on examples misclassified by the previous classifier $G_{m-1}(x)$. This process is described in Algorithm 2 [3]. In the work

presented, Decision Tree Classifiers are used as the base classifier.

Algorithm 2 AdaBoost Classifier

1: Initialize the observation weights $w_i = \frac{1}{n}$ for $i = 1, 2, \ldots n$
2: **for** $m = 1, \ldots, N$ **do**
3: Fit a classifier $G_m(x)$ to the training data using weights w_i
4: Compute

$$\text{err}_m = \frac{\sum_{i=1}^{n} w_i I(y^{(i)} \neq G_m(x^{(i)}))}{\sum_{i=1}^{n} w_i}$$

5: Compute $\alpha_m = \log(\frac{1-\text{err}_m}{\text{err}_m})$
6: Set $w_i = w_i \cdot \exp\left(\alpha_m \cdot I(y^{(i)} \neq G_m(x^{(i)})\right)$ for $i = 1, 2, \ldots, n$
7: **end for**
8: Output $G(x) = \text{sign}[\sum_{m=1}^{M} \alpha_m G_m(x)]$

4) XGBoost: XGBoost is a scalable end-to-end tree boosting system which has achieved state-of-the-art results on a variety of machine learning tasks [1]. Let $\hat{y}_i^{(t)}$ be the prediction of the i-th training example at the t-th iteration. Instead of adding base estimator f_t to minimize the standard objective

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) \tag{10}$$

Where l is a differentiable convex loss function and Ω penalizes the complexity of the model, a second-order approximation is minimized

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t) \tag{11}$$

where g_i and h_i are first and second order gradient statistics on the loss function.

$$g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}) \tag{12}$$

$$h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}) \tag{13}$$

D. Multi-layer Perceptron

Multi-layer Perceptrons are fully connected neural networks characterized by a set of hidden layers, activation functions, and an optimizer used during back propagation. In this work, a network architecture of 1 hidden layer of 10 units with ReLU activation is used. The Adam Optimizer [4] update of a parameter X for loss function L and learning rate α is defined as below.

$$m \leftarrow \beta_1 * m + (1 - \beta_1) * \nabla_X L \tag{14}$$

$$v \leftarrow \beta_2 * v + (1 - \beta_2) * (\nabla_X L)^2 \tag{15}$$

$$X \leftarrow X - \alpha \frac{1}{\sqrt{v} + \epsilon} * m \tag{16}$$

Recommended parameter values $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$ were used.

E. Summary

Logistic regression models with L_1 and L_2 regularization were trained with unit regularization strength. Decision trees were tuned for maximum depth via grid search with the criterion of validation set accuracy. Random forests and AdaBoost were tuned for maximum depth and number of estimators in the same fashion. XGBoost was trained with a maximum depth of 4, adding base estimators until training error did not decrease in consecutive iterations. A final model was selected based on validation set performance. This model was then evaluated on the test set and used to generate recommendations.

V. Results

A. Metrics

The following metrics are used to assess model performance

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{Total}} \tag{17}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{18}$$

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \tag{19}$$

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{20}$$

B. Model Selection

Based on the validation performance results summarized in Table III, XGBoost was selected as the model with most predictive power. Logistic regression with L_1 regularization as well as tree-based models performed well and generalized to future data well. These models effectively use subsets of the predictors, meaning most likely many predictors are not adding useful signal.

Accuracy was the primary metric used for model selection. In the recommendation system setting, false positives correspond to recommendations that the user did not like, and are therefore particularly undesirable. For this reason, precision is also an important metric for this application.

C. Test Performance

Test performance metrics are reported in Table IV, Table V and Figure 1. The XGBoost classifier generalizes well to future data, having better accuracy, recall, precision, and F1 on the test set when compared to the validation set.

TABLE IV
Test set performance metrics

	Accuracy	Recall	Precision	F1
XGBoost	76.65	74.33	79.33	76.75

TABLE III
MODEL PERFORMANCE METRICS

	L_2-Logistic	L_1-Logistic	GDA	Decision Tree	Random Forest	AdaBoost	XGBoost	MLP
Train Accuracy	63.73	74.15	74.07	75.37	71.68	75.33	**75.41**	74.74
Validation Accuracy	60.53	74.87	74.69	75.97	71.56	76.13	**76.15**	75.53
Validation Recall	37.25	**74.26**	73.70	73.63	58.85	73.46	73.42	71.03
Validation Precision	72.74	76.27	76.31	78.37	**80.66**	78.73	78.79	79.26
Validation F1	49.27	75.25	74.98	75.93	68.05	76.00	**76.01**	74.92

TABLE V
CONFUSION MATRIX

		Predicted	
		0	1
True	0	200968	52932
	1	70178	203198

D. Business Insights

Business recommendations can be generated for a particular user by predicting probabilities for all nearby businesses and returning the highest values. This way, recommendations are created based off the models most confident predictions.

1) User Integration: It is important for recommendation systems to not only be able to model the preferences of existing users, but also successfully integrate new users into the community with good recommendations. There are 133537 users who made their first review during the time frame of the test set. The XGBoost classifier was able to provide even more accurate predictions for these new users, as summarized in Table-VI

TABLE VI
NEW USERS TEST SET PERFORMANCE METRICS

	Accuracy	Recall	Precision	F1
XGBoost	80.93	81.14	83.64	82.38

2) Fresh Content: Another important quality of good recommendation systems is their ability to recommend new, fresh content. There are 1025 businesses that are first reviewed during the time frame of the test set. The XGBoost classifier is also able to generalize well to this fresh content, as summarized in Table-VII.

TABLE VII
NEW BUSINESSES TEST SET PERFORMANCE METRICS

	Accuracy	Recall	Precision	F1
XGBoost	77.99	82.22	81.28	81.75

VI. CONCLUSION

In this work, binary classification models were created with the end goal of generating business recommendations for Yelp users. Linear models, tree based models, and neural networks were trained. The final model selected for evaluation was an XGBoost classifier, which generalized well to future data and was able to successfully integrate new users and businesses.

To improve model performance, more of the information contained in the available Yelp data can be leveraged. This includes user tips, review texts, photos and their captions. Additionally, the community structure present in Yelp can be modeled with graph based models [2] and collaborative filtering techniques. Users can be grouped by behavior (e.g. hard to please) and modeled separately to create more accurate predictions and section the market for more actionable insights. The model output suggests that more rigorous feature engineering could improve accuracy and simultaneously reduce computational cost. Recommendations can be made to businesses on improvements they can make to better engage with the Yelp community.

CONTRIBUTIONS

- Pranav Bhardwaj: Logistic regression, tree-based models, testing
- Nicolas Bievre: Data processing, neural networks
- Frederik Johan Mellbye: Data processing, generative learning algorithms

ACKNOWLEDGMENT

Thank you to the CS229 teaching staff for all the support with this project.

REFERENCES

[1] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. *CoRR*, abs/1603.02754, 2016.

[2] William L. Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *CoRR*, abs/1709.05584, 2017.

[3] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference and prediction*. Springer, 2 edition, 2009.

[4] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[5] Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends, Oct 2010.

[6] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook, Oct 2010.

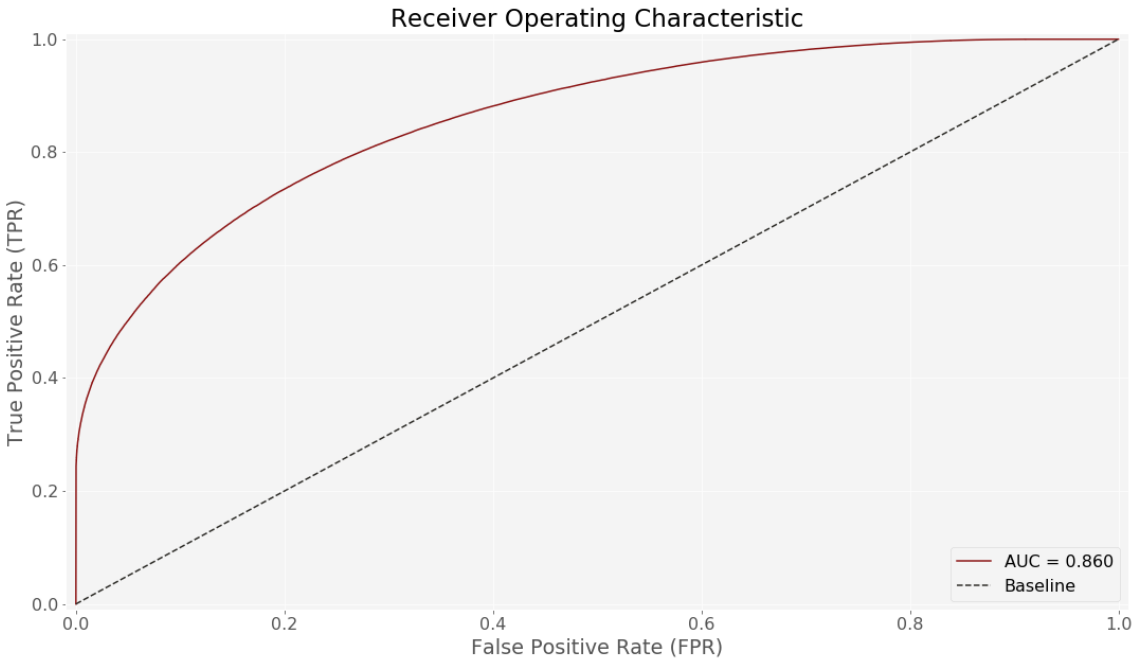

Fig. 1. Test set ROC curve for the XGBoost classifier