Wine Rating Prediction

Ke Xu (kexu@), Xixi Wang(xixiwang@)

***Abstract*—In this project, we want to predict rating points of wines based on the historical reviews from experts. The wine data is scraped from WineEnthusiast[1] and we used the price, wine variety and several winery location related information as the training features, and output the predicted rating for a wine. Since the desired output was a real-number value of rating, We focused on exploring a variety of linear regression models and also explored one neural network model. We were able to get 5.711 Mean Square Error(MSE) on testing data.**

I. Introduction

The overall goal is to build models to predict the rating on a scale of 1-100 of a wine. The initial idea was to provide personalized recommendation of wines based on historical reviews from experts, which is similar to Winc[2], but we would like to empower users with the freedom of choosing recommendations instead of blindly trusting Winc to send users the choices they provide. However, personal review dataset is not available, only public ratings are accessible via websites like WineEnthusiast[1]. Therefore, we treated the group of experts of the website as one person and simplified the problem to predicting the rating from the experts.

The input to our models is {price, variety, {winery, country, region}} and label is rating points. We then use a variety of linear regression models and one type of neural network models to output a predicted rating on a scale of 1-100.

The rest of the paper is organized as follows: Section 2 describes the related work. Section 3 describes the dataset and the relevant features for prediction. Section 4 describes the models performed on the dataset. Section 5 discusses the results of using the models for prediction. Section 6 summarizes the insights gained from this project and the future work.

II. Related Work

We conducted research on the existing related work. From input data perspective, [4] and [5] use chemical attributes as features. [6], [7] and [8] use derived statistics from review text such as number of reviews, review time and number of adjectives. [6] also has some metadata of the data such as age of wine and variety. While those are definitely good input features to predict wine ratings, but the underlying assumption is that tasting experience dominates the wine rating. We would argue that other aspects, such as winery and price can change your expectation of the wine, thus giving impacts on the rating.

From model perspective, [4] and [5] treat this as a classification problem while [6], [7] and [8] use regression approach. Other than various versions of linear regressions and logistic regressions, other methods such as Support Vector Machine, Linear Discriminant Analysis [5] and Random Forest[6] are also used. We started with converting this problem into a classification problem by dividing the 1-100 scale points into 4 categories as shown in I. We tried logistic regression model to predict the rating category, but the model didn't perform well. The reason behind it was that there are many data points near the category boundaries so that it's easy to be off by 1 category. Therefore, we thought regression should be a better solution to tackle this problem. We went with the commonly used linear regression model and later tried neural network which is not explored in the existing work.

TABLE I
Classification Buckets

Original rating points	Point bucket
95 - 100	4
90 - 94	3
85 - 89	2
below 85	1

Compared with the results from the earlier effort with linear regression models, our models have better performance, showing that the features we used such as winery location and wine price are critical factors for determining the wine ratings.

Exploring with other models, such as Support Vector Machine and Linear Discriminant Analysis, will be our future plan.

TABLE II
SOURCE DATA EXAMPLE

country	province	region_1	winery	variety	price	points
US	California	Napa Valley	Heitz	Cabernet Sauvignon	235.0	96
US	California	Knights Valley	Macauley	Sauvignon Blanc	90.0	96
France	Burgundy	Chablis	Domaine Grard Duplessis	Chardonnay	45.0	91

III. Dataset and Features

A. Source Data

Our source data includes 150,000 wine review data points scraped from WineEnthusiast[1]. The dataset is available on Kaggle[3]. The original columns we used include:

- Price: the cost for a bottle of the wine.
- Variety: the type of grapes used to make the wine (ie Pinot Noir).
- Winery: the winery where the wine was produced.
- Country: the country that the wine is from.
- Province: the province or state that the wine is from.
- Region: the wine growing area in a province or state (ie Napa).
- Points: Rating points on 1-100 scale.

Table II shows examples of the original data.

Fig 1 shows the distribution of rating points. For the data set we use, points are always in the range of 80-100.

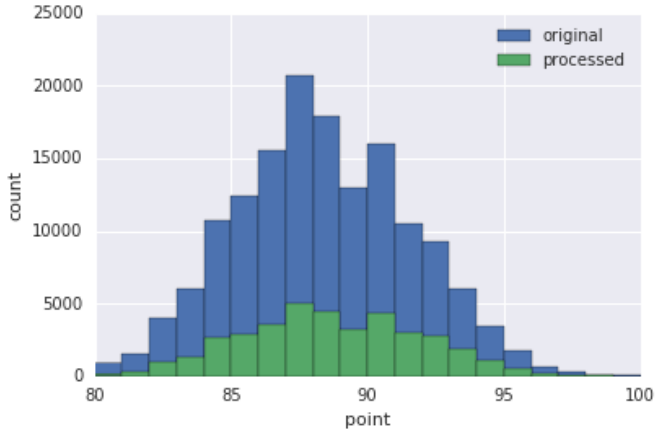

Fig. 1. Distribution of rating points

Table III shows statistics of points in processed, training and testing datasets. They have very similar stats.

B. Feature Engineering & Data Processing

There are a couple of location related columns in the original data. Treating them as separate input features and let regression model figuring out their relationship can be inefficient and unnecessary. Therefore, we combined country, province and region to produce a new signal: location. In order to train linear model, we preprocessed the input features by converted the string format features into categorical features. We use one-hot encoding[9] for those features. To improve data quality, we removed the duplicates data points and filtered out data points that have any empty features. In the end, to ensure that we have enough training data for a given value of a feature, we filtered out rarely seen values which are defined as values with less than 10 occurrence in the whole data set.

TABLE III
REVIEW POINT STATISTICS

Metric	Processed	Training	Testing
Max	100	100	100
Min	80	80	80
Mean	88.20	88.21	88.16
Median	88	88	88
Standard Deviation	3.29	3.29	3.29

After those steps, we got roughly 30,000 data points. As shown in Fig 1, the distribution of rating points are similar to the original data. Then we used 70% data as the training set and 30% as the testing set. The label used for training models is the original rating points from the experts with the range from 80 to 100.

IV. Methods

A. Linear regression

We began with basic linear regression approach introduced in class

$$\theta X = y$$

where X are the input features with intercept terms, θ are the weights associated to each feature, and y is the vector of the predicted ratings.

However, without any regularization, the model had a strong tendency to overfit. It had about 1k outliers, of which the predicted values were either extremely

large or extremely small. We took further exploration on the reason behind it, and observed that the coefficients were quite large. So we decided to add regularization into the model to improve it. We tried three different regularization techniques.

- Lasso (L1)

$$\hat{\theta} = \arg\min_{\theta}\{(y - \theta X)^2 + \lambda_1\|\theta\|_1\}$$

  We tried λ_1 with 0.1 / 1 / 10 / 100 and the best result comes with $\lambda_1 = 1$
- Ridge (L2)

$$\hat{\theta} = \arg\min_{\theta}\{(y - \theta X)^2 + \lambda_2\|\theta\|_2^2\}$$

  We used cholesky solver for Ridge, which obtains the closed-form solution. We tried λ_2 with 0.1 / 1 / 10 / 100 and the best result comes with $\lambda_2 = 1$
- Elastic Net (L1 and L2)

$$\hat{\theta} = \arg\min_{\theta}\{(y - \theta X)^2 + \lambda_1\|\theta\|_1 + \lambda_2\|\theta\|_2^2\}$$

  The best result comes with $\lambda 1 = 0.5$ and $\lambda_2 = 0.25$

Lasso regression model did not perform well for our datasets and so did Elastic Net regression model. The Ridge regression model provided the most reliable prediction while avoiding the issue of overfitting. Detailed evaluation results are shown in Table IV.

B. Neural Network

Even though Ridge regression model gave us the best result so far, one of our assumptions is that the correlation between our input features and the rating is often not linear. To explore other potential good-performance models, we tried with Multi-layer Perceptron, which is a class of feedforward artificial neural network.

Multi-layer Perceptron is sensitive to feature scaling, so we performed extra data processing by normalizing the price values into [0, 1] to be compatible with other categorical feature values.

We tuned the model with different parameter settings.

- Activation functions:
  - logistic, the logistic sigmoid function, returns $f(x) = 1/(1 + exp(-x))$
  - relu, the rectified linear unit function, returns $f(x) = max(0, x)$
- The solver for weight optimization:
  - L-BFGS, refers to Limited-memory Broyden-Fletcher-Goldfarb-Shanno, is an optimizer in the family of quasi-Newton methods. Like the original Broyden-Fletcher-Goldfarb-Shanno (BFGS), L-BFGS uses an estimation to the inverse Hessian matrix to steer its search through variable space, but where BFGS stores a dense nn approximation to the inverse Hessian (n being the number of variables in the problem), L-BFGS stores only a few vectors that represent the approximation implicitly.
  - SGD, refers to stochastic gradient descent, which performs a weight update for each training example x^i and label y^i.
  - Adam, short for Adaptive Moment Estimation, is an optimization algorithm that can used instead of the classical stochastic gradient descent procedure to update weights iterative based in training data. The classical stochastic gradient descent maintains a single learning rate for all weight updates and the learning rate does not change during training. What Adam differs from classical stochastic gradient descent is a learning rate is maintained for each weight and separately adapted as learning unfolds.
- Neurons:
  - 30 / 50 / 100 / 200 / 500
- Hidden Layers:
  - 1 / 2 / 3 / 5
- Max iterations:
  - 100 / 200 / 300 / 500

The network structure, which presents the best performance, consists of two fully connected hidden layers, and 100 neurons in each layer. ReLU was used as the activation function and optimization for the squared-loss used lbfgs with max 200 iterations in total.

V. Results & Discussion

A. Visualizing Labels vs Predictions

By visualizing Labels vs Predictions on the test data, we can get idea on whether there are outliers, whether we are underestimating / overestimating and how far away the predictions are from ground truth in general. In Fig 2, 3, 4 and 5, we plot the test dataset with x axis as label and y axis as prediction. The red line shows the where the perfect prediction lays.

As shown in Fig2, without regularization, we have outliers with huge predicted values. They are so large that the perfect line is almost like a flat line on the chart. That also explains why we have large errors in Table IV.

Fig 3 is the result for adding Ridge regularization and Fig 4 is the result with Lasso. They clears show that

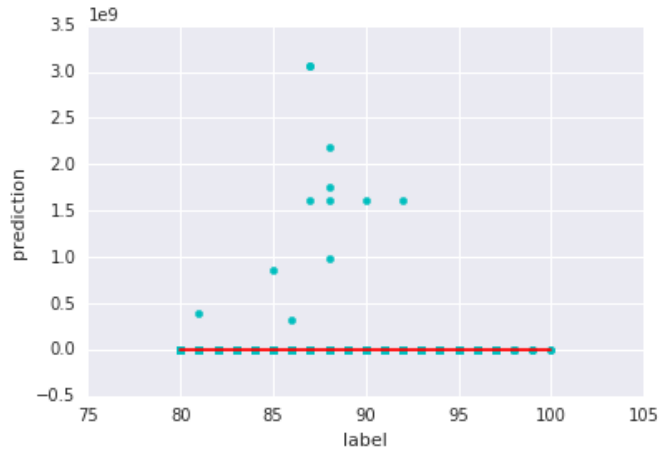

Fig. 2. Basic Linear Regression performance

regularization works much better. We no longer have huge outliers, although there are a few with prediction large than 100. By comparing the two, we can see that Ridge brings less and smaller outliers and predicts better in the lower range (label less than 85). Lasso always overestimate in the lower range and are more "loosely" gathered around higher range (larger than 95).

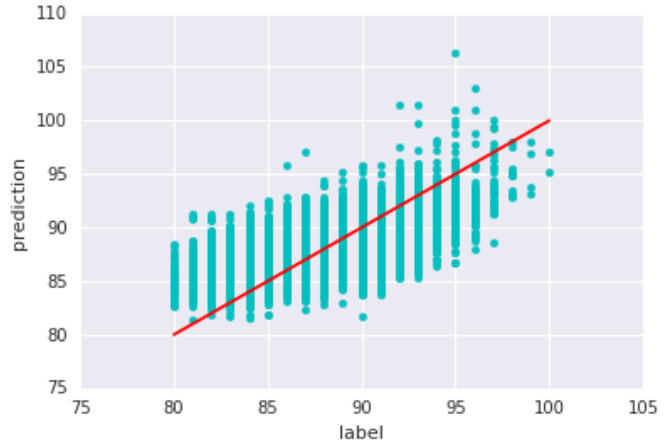

Fig. 3. Linear Regression w Ridge performance

Fig 5 shows the result of Neural Network. It has similar pattern as Ridge with a few outliers around 95.

B. Quantify Quality By Metrics

We defined three metrics to evaluate and compare the model performance, R^2 score, mean square error (MSE) and median absolute error (MAE).

- R^2, is a statistical measure of how close the data are to the fitted regression line. It is calculated by

$$R^2 = 1 - \frac{u}{v}$$

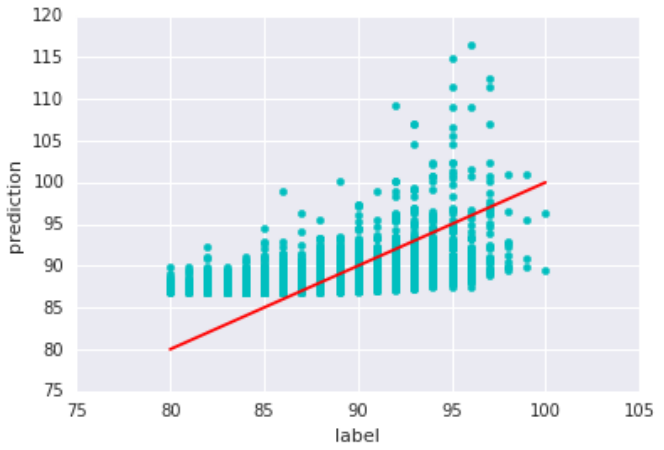

Fig. 4. Linear Regression w Lasso performance

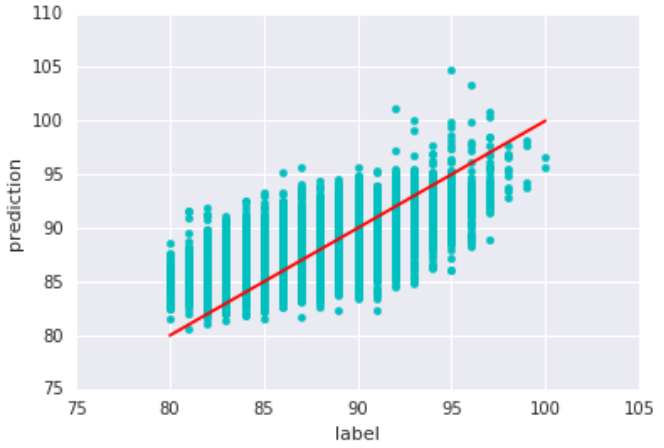

Fig. 5. Neural Network performance

u and v are defined as

$$u = \sum_{i=1}^{m} (y^{(i)} - \hat{y}^{(i)})^2 \tag{1}$$

$$v = \sum_{i=1}^{m} (y^{(i)} - \frac{1}{m} \sum_{j=1}^{m} y^{(j)})^2 \tag{2}$$

where $\hat{y}$ is the value predicted by the model.

- MSE, is the sum, over all the data points, of the square of the difference between the predicted and actual label value, divided by the number of data points.
- MAE, is the median of all of the absolute difference between the predicted and actual label value.

From these metrics as listed in TABLE IV, we can see the best performance model is the linear regression model using Ridge regularization followed by NN model with very close result. There are several observations and corresponding conclusions as listed below.

TABLE IV
EVALUATION RESULTS FOR ALL MODELS

	Training			Testing		
	R^2	MSE	MAE	R^2	MSE	MAE
Basic LinearRegression	0.537	5.035	1.468	-1.38E+17	1.48E+18	1.602
Linear Regression w Lasso	0.225	8.428	2.086	0.239	8.163	2.044
Linear Regression w Ridge	0.535	5.060	1.481	0.468	5.711	1.593
Linear Regression w Elastic Net	0.225	8.427	2.078	0.240	8.159	2.046
Neural Network	0.522	5.144	1.488	0.466	5.860	1.610

- Training error ratio is only slightly better than the testing error ratio for the two models, Ridge regression and neural network, with the best performance. It proves that our models does not overfit due to the selected regularization mechanism and feature engineering work.
- The error ratio on the training data shows that wine rating can't be perfectly predicted by the price, variety, winery and location. Therefore, combining what we have with data in related work such as chemical attributes, age of the wine, review statistics may give us better result.

VI. CONCLUSION & FUTURE WORK

We explored several different models and tuned with different parameters for each model, and had one common observation: for all the models, the performance for both training and testing data sets is not good as expected. This indicates that the review points cant be perfectly predicted by current features.

In the future, to improve quality of the results, we would like to

- add features such as acidity, alcohol by volume, the age of the wine, reviewers to the input set.
- explore with other models, such as Support Vector Machine and Random Forest.
- investigate more on tuning neural network parameters to have better results.

VII. CONTRIBUTIONS

We both worked on all parts of the project.

REFERENCES

[1] WineEnthusiast Ratings. http://www.winemag.com/?s=&drink_type=wine
[2] Winc. https://www.winc.com/member-benefits
[3] Kaggle wine review dataset. https://www.kaggle.com/zynicide/wine-reviews
[4] Amelia Lemionet, Yi Liu, Zhenxiang Zhou. Predicting quality of wine based on chemical attributes. *CS 229 project*, 2015. http://cs229.stanford.edu/proj2015/245_report.pdf
[5] Eric Sebastian Soto. Using Chemical Data to Predict Wine Ratings. *CS 229 project*, 2012. http://cs229.stanford.edu/proj2012/Soto-UsingChemicalDatatoPredictWineRatings.pdf
[6] Fan Chao, Pengbo Li, Renxiang Yan. Predicting Review Rating for Wine Recommendation. https://cseweb.ucsd.edu/~jmcauley/cse190/reports/fa15/020.pdf
[7] Dominic Rossi. Predicting wine ratings using linear models. https://cseweb.ucsd.edu/~jmcauley/cse255/reports/wi15/Dominic_Rossi.pdf
[8] Benjamin Braun, Robert Timpe. Text based rating predictions from beer and wine reviews. https://cseweb.ucsd.edu/~jmcauley/cse255/reports/wi15/Benjamin_Braun_Robert%20Timpe.pdf
[9] https://www.kaggle.com/dansbecker/using-categorical-data-with-one-hot-encoding