**Data set:**

Data set includes recent online review data for locations across the US. The file **reviews.csv** includes

reviews left for these locations across a handful of review sites. The file **test\_reviews.csv**

contains the same columns, but the ratings are not provided.

The fields in **reviews.csv** (and **test\_reviews.csv** ) are:

● location\_id: an id identifying which location this review is about.

● review\_id: an id unique to each review

● source: the source of the review

● date: the date the review was left

● rating: the rating of the review between 1 and 5 where 5 is the best possible rating.

**A/B Testing:**

**Compare the two locations ( 4962\_201 and 4962\_380 ).**

We compare the mean ratings of these locations

**Mean of location 4962\_201 = 4.75**

**Mean of location 4962\_380 = 4.669**

We perform t test to compare the mean ratings because means don’t take into account the distribution and variance of the two samples.

First, we perform F test to compare the variance of the two samples

F test :

Null hypothesis: Variance location 4962\_201 = 4962\_380

Alternate hypothesis: Variances not equal

F test p value = 0.087

At 95% confidence level, p value > 0.05. Thus, we don’t reject the null hypothesis and conclude that the variances are equal.

**T test:**

We suspect the mean(rating )of location 4962\_201 to be greater than 4962\_380.

Null hypothesis: Mean location 4962\_201 <= 4962\_380

Alternate hypothesis: Mean location 4962\_201 > 4962\_380

At 95% confidence level, p value < 0.05. Thus, we reject the null hypothesis and conclude that **Mean location 4962\_201 > 4962\_380**.

**Compare the two locations ( 4962\_381 and 4962\_915 ).**

We compare the mean ratings of these locations

**Mean of location 4962\_381 = 4.55**

**Mean of location 4962\_915 = 4.62**

We perform t test to compare the mean ratings because means don’t take into account the distribution and variance of the two samples.

First, we perform F test to compare the variance of the two samples

F test :

Null hypothesis: Variance location 4962\_381 = 4962\_915

Alternate hypothesis: Variances not equal

F test p value = 0.254

At 95% confidence level, p value > 0.05. Thus, we don’t reject the null hypothesis and conclude that the variances are equal.

**T test:**

We suspect the mean(rating )of location 4962\_381 to be similar to 4962\_915.

Null hypothesis: Mean location 4962\_381 = 4962\_915

Alternate hypothesis: Means not equal

T test p value = 0.254

At 95% confidence level, p value > 0.05. Thus, we don’t reject the null hypothesis and conclude that **Mean location 4962\_381 = 4962\_915**.

**The t test suggests that the means are equal, but based on just mean statistic location 4962\_915 performs slightly better.**

**Machine Learning Model:**

Build a model to predict the rating a reviewer will give a location (the rating field from the

reviews file) given the data from all other columns. The goal of the model is not just to

predict the most likely rating of any review. We want a model that can accurately predict

the average rating across a set of reviews (so please do not just develop a model that

predicts 5 for every review just because that is by far the most common rating.)

I have used Gradient boosting along with SMOTE sapling.

The reason I chose this model is to handle imbalanced data where the rating classes are not represented equally.

Rating counts:

5.0 139838

1.0 18192

4.0 13636

3.0 4171

2.0 3439

We have greater no. of class 5 and lesser no. of other classes.

SMOTE:

SMOTE is an oversampling method that uses nearest neighbor of observations to create synthetic data.

Algorithm:

1. Finding the k-nearest-neighbors for minority class observations (finding similar observations)
2. Randomly choosing one of the k-nearest-neighbors and using it to create a similar, but randomly tweaked, new observation.

This sampling will oversample the minority classes and tries to equalize the ratio between minority and majority classes. This is a well-known technique to improve models on imbalanced data.

The create way to sample is to break the entire data into train and test sets. Then, sample the training data so that the results are generalized.

Gradient Boosting:

Gradient boosting is an ensemble method which iteratively learn weak classifiers and add them to form a strong classifier. The examples that are misclassified gain weight and those that are classified correctly lose weight. Weighing is important so that learner does not lose track of the examples that it has already mastered.

The target outcomes for each case are set based on the gradient of the error with respect to the prediction. Each new model takes a step in the direction that minimizes prediction error, in the space of possible predictions for each training case. The result of these learners is combined using weighted vote.

Gradient boosting deals with class imbalance by creating successive training sets based on incorrectly classified examples.

Performance Metrics:

* Accuracy is ratio of correctly predicted instances over all instances used

Accuracy = TP+ TN/P + N

Accuracy can be misleading, it is possible to get a decent model accuracy while having incorrect predictions for the minority classes.

* ROC curve also over estimates the performance of classifier on imbalanced data.
* We can use precision, recall, F1 score metrics for model evaluation as it is a measure of classifier exactness.

Precision is defined as the number of true positives divided by the number of true positives plus the number of false positives.

Precision = TP/ TP + FP

Recall is defined as the number of true positives divided by the number of true positives plus the number of false negatives.

Recall= TP/ TP + FN

F1 Score is harmonic mean of precision and recall

F1 score = 2\* (Precision \* Recall/Precision + Recall)

Model Evaluation:

Iter Train Loss Remaining Time

1 776923.2620 4.74m

2 767461.6235 4.60m

3 759031.3518 4.74m

4 752873.2085 4.61m

5 747251.8348 4.47m

6 741685.2245 4.43m

7 736513.1152 4.42m

8 731845.4850 4.46m

9 728927.1701 4.44m

10 725538.0921 4.38m

20 703212.3558 3.95m

30 690902.5484 3.44m

40 683610.6981 2.91m

50 677072.4343 2.45m

60 670576.3774 1.99m

70 666202.3610 1.49m

80 661863.2124 59.63s

90 658204.6877 29.99s

100 654603.5666 0.00s

Accuracy of train set: 0.406

Accuracy of test set: 0.371

Confusion Matrix:

[[ 2656 50 1265 597 926]

[ 406 13 323 161 190]

[ 341 17 470 255 205]

[ 987 34 1144 763 1096]

[10671 347 8812 5988 16066]]

Classification Report

precision recall f1-score support

1.0 0.18 0.48 0.26 5494

2.0 0.03 0.01 0.02 1093

3.0 0.04 0.36 0.07 1288

4.0 0.10 0.19 0.13 4024

5.0 0.87 0.38 0.53 41884

avg / total 0.70 0.37 0.45 53783

We try to get a higher F1 score, that is done by not having extreme precision or recall. Our model performs well as get moderate precision and recall, not extremes.

Also, we get similar accuracy for training and test data.

**Other ways to improve the model:**

1. Collect more datapoints to get a balanced dataset.
2. Try to capture more features to reduce bias.
3. Using different resampling techniques.
4. Down sampling the majority class
5. Up sampling the minority class
6. Using bagging such as random Forest to balance the data. They work by bootstrapping the data when creating the individual estimators, then aggregate (vote in case of classification) the predictions across individual estimators.
7. Use class weight balance in models such as Random Forest. We want to give a higher weight to the minority class and a lower weight to the majority class.
8. Try penalized models impose additional cost on the model for wrongly classifying the minority classes. These penalties can bias the model to pay more attention to minority classes. For eg. Penalized (ridge or lasso) logistic regression or SVM.
9. Use hyper parameter tuning to get the best prediction results. (Code mentioned in the file)
10. Try to further improve the F1 score , i.e. balance of precision and recall.