DS-GA 03 Fake Review Detection Final Paper

Amelia Chu[1] Dee Hahm Evaristus Ezekwem Gabriella Hurtado

New York University

Email:{ac4119, ece278, drh382, gh1408}@nyu.edu

1 Introduction

In the past decade, consumer product reviews have become increasing available to the general public. These reviews impact consumer decisions and, in turn, influences business revenue [Doward, 2012]. With their livelihoods at risk, some businesses attempt to influence these reviews in increasingly unethical ways from offering comped meals to paying crowdworkers for fake reviews [Filloon, 2019],[Segal, 2011]. This 'deceptive opinion spam' further complicates the path to make an informed consumer decision; the sheer volume of consumer product reviews make it impossible to manually curate. In an attempt to learn about deceptive opinion spam and capture this phenomenon, this project examines a subset of Yelp reviews for New York City restaurants and categorizes them as real or fake.

2 Related Works

As the Yelp Review dataset is accessible via Kaggle and was previously used for the Yelp Dataset Challenge [Yelp Inc., 2020], there are many other projects which incorporated this dataset and analyzed it in different ways. There were projects which examined review sentiment (e.g., [Jones et al., 2018]), those that examined behavioral features (e.g., [Ghosh and Zargar, 2015]), and projects that incorporate a combination of feature types and models (e.g., [Gupta et al., 2019]). Similar to Gupta et al. (2019), our project will also incorporate a combination of feature types and models. However, Gupta et al. (2019) used oversampling to address the problem of class imbalance. In our project, we will try undersampling in addition to oversampling methods to reduce overfitting the minority class (see Section 4.1).

[1]Member responsible for uploading submissions

3 Problem Definition and Algorithm

3.1 Task

Classifying reviews as fake or real is a supervised learning problem, specifically, a binary classification problem. In this project, we will incorporate a combination of text and behavior features (See Section 4.1) as input and output a classification of 'real' (label = 0) or 'fake' (label =1).

3.2 Algorithm

For this project, we chose logistic regression as our baseline, and then examined four different algorithms (Support Vector Machine, Random Forest, K-Nearest Neighbors, and Naive Bayes).

3.2.1 Logistic Regression

Logistic regression is a statistical model used to predict the probability of an input belonging to either of a set of binary classes. Logistic regression is similar to linear regression in that its input values are linearly combined, but is distinguished by its use of a logistic function to transform predictions into binary classifications. Logistic regression computes the probability of an input X belonging to the default class, Y=1, conditioned on X ' s features, $P(X) = P(Y = 1|X)$. This probability is then transformed with the logistic function $f(\eta) = \frac{1}{1+e^{-\eta}}$, which maps the probability to the range $(0, 1)$. For binary classification, $y \geq 0.5$ indicates a classification of 1 (in our case, that a review is fake), and $y < 0.5$ indicates a classification of 0 (a review is real). Logistic regression assumes a Bernoulli response distribution, independent observations, and negligible multicollinearity between feature variables. For parameters θ, logistic function $f(\eta)$, and a training point (x, y), logistic regression assumes $P(Y = 1|X = x) = f(\theta^T x)$ and $P(Y = 0|X = x) = 1 - f(\theta^T x)$. As such, the proba-

bility of a single data point can be expressed

$$P(Y = y|X = x) = f(\theta^T x)^y \big(1 - f(\theta^T x)\big)^{(1-y)}$$

To determine values for the parameters θ, maximum likelihood estimation (MLE) is used. The goal is to find the values of θ that maximize the log-likelihood function. Given that the likelihood of independent variables is expressed $L(\theta) = \prod_{i=1}^{n} P(Y = y^{(i)}|X = x^{(i)})$, we can substitute the likelihood of our Bernoulli response variable and take the log of this function to achieve

$$LL(\theta) = \sum_{i=1}^{n} y^{(i)} log f(\theta^T x^{(i)}) + (1-y^{(i)}) log\big(1 - f(\theta^T x^{(i)})\big)$$

We can use gradient ascent to maximize the log likelihood and thus find the optimal values of θ.

3.2.2 Naive Bayes

The NaiveBayes is a probabilistic algorithm mostly used in classification problems. The algorithm is based on the Bayes' Theorem. The key idea of NaiveBayes is to compute the posterior probabilities of all labels based on the prior probabilities observed in the data.

According to the Bayes Theorem:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)} \rightarrow Posterior = \frac{likelihood \cdot prior}{evidence}$$

To make classifications, we need to use X to predict Y. In other words, given a data point $X = (x_1, x_2, ..., x_k)$, what the odd of Y being y. This can be rewritten as the following equation:

$$P\big(Y = y|X = x_1, x_2, ..., x_k)\big)$$

Find the most likely $y \rightarrow prediction =_y P\big(Y = y|X = x_1, x_2, ..., x_k)\big)$ The Naive Bayes algorithm assumes that all features are independent of each other.

3.2.3 K-Nearest Neighbors

The K-Nearest Neighbors (KNN) algorithm maps data and groups them together based on their similarity. This is done by calculating distances among data. In this project, all reviews were tokenized. The k is a parameter that determines the number of the nearest neighbors. This parameter is also used to control overfitting. In order to prevent overfitting, the low boundary of k value was set equal to the log of the total data points, which is 5. Then k values of 10, 15, and 20 were tested. Additional k values of 11, 13, and 14 were tested. Overall, most k values led to approximately 53% ROC-AUC score, however, k value of 10 has the highest average precision rate.

3.2.4 Random Forest Classifier

The random forest classifier is an ensemble method that uses the outputs of several decision trees to predict a classification. It makes predictions on test cases by choosing the class which received the most amount of votes from the decision trees. The ideology behind the random forest classifier is that by training decision trees on bootstrapped samples of training data and aggregating their results, a low-bias, high-variance classifier may be achieved.

To avoid overfitting, some of the parameters that may be adjusted are maximum depth and minimum samples needed to split. The maximum depth parameter controls how large the decision trees are permitted to grow, and the minimum samples to split parameter prevents nodes with less than the minimum value from being split any further.

3.2.5 Support Vector Machine

For support vector machine, the LinearSVM and SGDClassifer implementations were chosen as opposed to SVC because of how large the tokenized dataset is. LinearSVM scales the intercept differently ($\frac{1}{2}\|wb\|^2 + c\sum x_i^i$). LinearSVM is a convex optimization problem whereas SGDClassifier uses stochastic gradient descent, so it does not always converge to the same solution.

```
# LinearSVM
init parameters
iter min(examples, max_iter):
    get prediction
    if pred != label:
 update params
end when hit min(examples, max_iter)


# SGDClassifer
init parameters
iter examples:

 select random example
    get prediction
    get loss
    get gradients
    update parameters
end when hit max iter
```

Table 1: Pseudocode for Linear SVM and SGD Classifier

In both LinearSVM and SGDClassifier, there was the ability to set the loss function and regularization (C and alpha, respectively) which helped control overfitting.

Through experimentation, SGDClassifier appeared to perform better than Linear SVM. The best parameters for SGDClassifier were $\alpha = 0.1$, modified huber loss, and l2 as penalty function.

4 Experimental Evaluation

4.1 Data

For this project, we were provided 250,874 training examples and 35,918 validation examples. The label classes were '1' for fake review and '0' for real review and were imbalanced. To address the class imbalance, both oversampling and undersampling techniques were attempted.

	Train	Dev
Number of Positive Examples	25, 819 (10.3 %)	3,648 (10.1%)
Number of Negative Examples	225,055 (89.7%)	32,270 (89.8%)
Total Number of Examples	250, 874	35,918

Table 2: Fake review dataset summary

4.1.1 Oversampling & SMOTe

For oversampling, the imbalanced-learn library, Synthetic Minority Over-sampling Technique (SMOTe), was used to resample data. [Chakrabarty, 2019]. Since there are more negative examples, the positive examples were scaled multiple times in order to compensate for imbalance data.

4.1.2 Undersampling

For undersampling, the training set was first separated by label. The negative examples were divided into nine parts so that each part had roughly the same number of examples as the total number of positive examples. To reduce the chance of overfitting, a random sample of 80% of the positive examples were then appended to each for the parts containing negative examples (See Appendix B).

4.1.3 Text Features

To extract meaningful text features, the spaCy, string, and pyspellchecker libraries were used to tokenize the review text data. First, any spelling mistakes were corrected by pyspellchecker. Then, stop words and punctuation, as defined by spaCy and string, were removed. After that, the top words that occurred most frequently between both real and fake reviews were removed. These top overlapping words were generated based on the overlap between the top 100 recurring words in fake and real reviews. Finally, the remaining words were stemmed to reduce the number of duplicative features.

4.1.4 Behavioural Features

Since businesses with fewer reviews are more incentivized to post fake reviews [Luca and Zervas, 2013], we added a feature num_prod_reviews which is the total number of reviews a product page has received. Likewise, users who have less activity are most likely to be categorized as 'fake' by Yelp [Yelp Inc., 2013], therefore we added a feature num_user_reviews of the total number of reviews the user has given. From our exploratory analysis, we found that fake reviews were twice as likely to have a rating of '1' than real reviews (Figure X), so the rating column was also considered as a feature.

4.2 Methodology

For this project, we will be using the Average Precision Score and Area Under the Receiver Operating Characteristic Curve (ROC AUC) Score from scitkit-learn to evaluate our models. Because our dataset is heavily imbalanced, we will prioritize Average Precision when choosing our final model. For each individual model, Different parameter values were attempted to find the best model parameter combination (See Appendix A). The model with the best metrics overall will be selected as the final model.

4.3 Results

Overall, the SGDClassifier model with $\alpha = 0.1$, modified huber loss, and l2 as penalty function, performed the best.

Model (parameters)	AUC	Average Precision
SVM (loss = 'modified_huber', $\alpha = 0.1$)	0.77	0.24
RFC (N trees=100, max depth=20, min. sample split=0.05)	0.72	0.21
Naive Bayes (α=3.0)	0.71	0.21
KNN (NN=13)	0.53	0.11

Table 3: Summarized model results

5 Discussion & Conclusion

However, for the fake reviews we expected ratings of 1 and 5 to have the highest frequency but it appears that while 5 had the highest frequency there was a huge margin between 4 and 5 which was unlike what we saw in the real-reviews also as expected reviews with a 1 rating had a higher frequency than those withe 2 3 ratings.

One of the major challenges was a larger overlap of words in both 'fake' and 'real' reviews. In this project,

fake and real reviews were tokenized to run various algorithms. However, since there were a large group of frequent words that appeared in both fake and real reviews, this impacted performance across all models. This also explains why the KNN algorithm had the worst accuracy rates. Since KNN predicts data based on approximations with other data, having the same tokenized words appearing often in both fake and real reviews, made KNN not an ideal model for this project.

It is also worth noting that fake and real reviews are labeled by Yelp. In this project, it was uncertain whether the goal was to identify genuine fake and real reviews, or try to replicate Yelp's fake review detecting algorithm. The labels are not ground truth. Therefore, it is possible that reviews labeled as fake could be genuine review and vice versa.

As a next step, using more than one model to detect fake reviews should be considered. All models tested in this project have their own strength and weakness. Some may be suited to detect certain kinds of fake reviews but do poorly on detecting other kinds. There numerous possible motives for someone to post fake reviews, such as paid promotion, competitor, and freebies, etc. Therefore, even though SVM outperformed all other models it is possible that other models can be more efficient to detect certain genuine fake reviews.

References

[Chakrabarty, 2019] Chakrabarty, N. (2019). Application of synthetic minority over-sampling technique (smote) for imbalanced datasets. *Towards AI*.

[Doward, 2012] Doward, J. (2012). Here's proof that yelp rating can make or break a retailer. *Business Insider*.

[Filloon, 2019] Filloon, W. (2019). Yelp elite are becoming obsolete. *Eater*.

[Ghosh and Zargar, 2015] Ghosh, S. K. and Zargar, S. (2015). Detection of Review Spam in Online Review Websites. Technical report, Stony Brook University, Department of Computer Science.

[Gupta et al., 2019] Gupta, A., Gawad, A., Hule, A., Shanbhag, G., Kadam, V., and Gupta, P. (2019). Yelp Fake Review Detection. Technical report, Dublin City University, School of Computing.

[Jones et al., 2018] Jones, C., Mellachervu, N., Sinha, N., Young, J. W., and Zhen, W. (2018). Yelp Review Analysis in Spark. Technical report.

[Luca and Zervas, 2013] Luca, M. and Zervas, G. (2013). Fake It Till You Make It: Reputation, Competition, and Yelp Review Fraud. *SSRN Electronic Journal*.

[Segal, 2011] Segal, D. (2011). Here's proof that yelp rating can make or break a retailer. *The New York Times*.

[Yelp Inc., 2013] Yelp Inc. (2013). Why does yelp recommend reviews?

[Yelp Inc., 2020] Yelp Inc. (2020). Yelp dataset.

DS-GA 03 Fake Review Detection Final Paper

Amelia Chu[1] Dee Hahm Evaristus Ezekwem Gabriella Hurtado

New York University

Email:{ac4119, ece278, drh382, gh1408}@nyu.edu

1 Introduction

In the past decade, consumer product reviews have become increasing available to the general public. These reviews impact consumer decisions and, in turn, influences business revenue [Doward, 2012]. With their livelihoods at risk, some businesses attempt to influence these reviews in increasingly unethical ways from offering comped meals to paying crowdworkers for fake reviews [Filloon, 2019],[Segal, 2011]. This 'deceptive opinion spam' further complicates the path to make an informed consumer decision; the sheer volume of consumer product reviews make it impossible to manually curate. In an attempt to learn about deceptive opinion spam and capture this phenomenon, this project examines a subset of Yelp reviews for New York City restaurants and categorizes them as real or fake.

2 Related Works

As the Yelp Review dataset is accessible via Kaggle and was previously used for the Yelp Dataset Challenge [Yelp Inc., 2020], there are many other projects which incorporated this dataset and analyzed it in different ways. There were projects which examined review sentiment (e.g., [Jones et al., 2018]), those that examined behavioral features (e.g., [Ghosh and Zargar, 2015]), and projects that incorporate a combination of feature types and models (e.g., [Gupta et al., 2019]). Similar to Gupta et al. (2019), our project will also incorporate a combination of feature types and models. However, Gupta et al. (2019) used oversampling to address the problem of class imbalance. In our project, we will try undersampling in addition to oversampling methods to reduce overfitting the minority class (see Section 4.1).

[1]Member responsible for uploading submissions

3 Problem Definition and Algorithm

3.1 Task

Classifying reviews as fake or real is a supervised learning problem, specifically, a binary classification problem. In this project, we will incorporate a combination of text and behavior features (See Section 4.1) as input and output a classification of 'real' (label = 0) or 'fake' (label =1).

3.2 Algorithm

For this project, we chose logistic regression as our baseline, and then examined four different algorithms (Support Vector Machine, Random Forest, K-Nearest Neighbors, and Naive Bayes).

3.2.1 Logistic Regression

Logistic regression is a statistical model used to predict the probability of an input belonging to either of a set of binary classes. Logistic regression is similar to linear regression in that its input values are linearly combined, but is distinguished by its use of a logistic function to transform predictions into binary classifications. Logistic regression computes the probability of an input X belonging to the default class, Y=1, conditioned on X ' s features, $P(X) = P(Y = 1|X)$. This probability is then transformed with the logistic function $f(\eta) = \frac{1}{1+e^{-\eta}}$, which maps the probability to the range $(0, 1)$. For binary classification, $y \geq 0.5$ indicates a classification of 1 (in our case, that a review is fake), and $y < 0.5$ indicates a classification of 0 (a review is real). Logistic regression assumes a Bernoulli response distribution, independent observations, and negligible multicollinearity between feature variables. For parameters θ, logistic function $f(\eta)$, and a training point (x, y), logistic regression assumes $P(Y = 1|X = x) = f(\theta^T x)$ and $P(Y = 0|X = x) = 1 - f(\theta^T x)$. As such, the proba-

bility of a single data point can be expressed

$$P(Y = y|X = x) = f(\theta^T x)^y \big(1 - f(\theta^T x)\big)^{(1-y)}$$

To determine values for the parameters θ, maximum likelihood estimation (MLE) is used. The goal is to find the values of θ that maximize the log-likelihood function. Given that the likelihood of independent variables is expressed $L(\theta) = \prod_{i=1}^{n} P(Y = y^{(i)}|X = x^{(i)})$, we can substitute the likelihood of our Bernoulli response variable and take the log of this function to achieve

$$LL(\theta) = \sum_{i=1}^{n} y^{(i)} log f(\theta^T x^{(i)}) + (1-y^{(i)}) log\big(1 - f(\theta^T x^{(i)})\big)$$

We can use gradient ascent to maximize the log likelihood and thus find the optimal values of θ.

3.2.2 Naive Bayes

The NaiveBayes is a probabilistic algorithm mostly used in classification problems. The algorithm is based on the Bayes' Theorem. The key idea of NaiveBayes is to compute the posterior probabilities of all labels based on the prior probabilities observed in the data.

According to the Bayes Theorem:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)} \rightarrow Posterior = \frac{likelihood \cdot prior}{evidence}$$

To make classifications, we need to use X to predict Y. In other words, given a data point $X = (x_1, x_2, ..., x_k)$, what the odd of Y being y. This can be rewritten as the following equation:

$$P\big(Y = y|X = x_1, x_2, ..., x_k)\big)$$

Find the most likely $y \rightarrow prediction =_y P\big(Y = y|X = x_1, x_2, ..., x_k)\big)$ The Naive Bayes algorithm assumes that all features are independent of each other.

3.2.3 K-Nearest Neighbors

The K-Nearest Neighbors (KNN) algorithm maps data and groups them together based on their similarity. This is done by calculating distances among data. In this project, all reviews were tokenized. The k is a parameter that determines the number of the nearest neighbors. This parameter is also used to control overfitting. In order to prevent overfitting, the low boundary of k value was set equal to the log of the total data points, which is 5. Then k values of 10, 15, and 20 were tested. Additional k values of 11, 13, and 14 were tested. Overall, most k values led to approximately 53% ROC-AUC score, however, k value of 10 has the highest average precision rate.

3.2.4 Random Forest Classifier

The random forest classifier is an ensemble method that uses the outputs of several decision trees to predict a classification. It makes predictions on test cases by choosing the class which received the most amount of votes from the decision trees. The ideology behind the random forest classifier is that by training decision trees on bootstrapped samples of training data and aggregating their results, a low-bias, high-variance classifier may be achieved.

To avoid overfitting, some of the parameters that may be adjusted are maximum depth and minimum samples needed to split. The maximum depth parameter controls how large the decision trees are permitted to grow, and the minimum samples to split parameter prevents nodes with less than the minimum value from being split any further.

3.2.5 Support Vector Machine

For support vector machine, the LinearSVM and SGDClassifer implementations were chosen as opposed to SVC because of how large the tokenized dataset is. LinearSVM scales the intercept differently ($\frac{1}{2}\|wb\|^2 + c\sum x_i^i$). LinearSVM is a convex optimization problem whereas SGDClassifier uses stochastic gradient descent, so it does not always converge to the same solution.

```
# LinearSVM
init parameters
iter min(examples, max_iter):
    get prediction
    if pred != label:
 update params
end when hit min(examples, max_iter)


# SGDClassifer
init parameters
iter examples:

 select random example
    get prediction
    get loss
    get gradients
    update parameters
end when hit max iter
```

Table 1: Pseudocode for Linear SVM and SGD Classifier

In both LinearSVM and SGDClassifier, there was the ability to set the loss function and regularization (C and alpha, respectively) which helped control overfitting.

Through experimentation, SGDClassifier appeared to perform better than Linear SVM. The best parameters for SGDClassifier were $\alpha = 0.1$, modified huber loss, and l2 as penalty function.

4 Experimental Evaluation

4.1 Data

For this project, we were provided 250,874 training examples and 35,918 validation examples. The label classes were '1' for fake review and '0' for real review and were imbalanced. To address the class imbalance, both oversampling and undersampling techniques were attempted.

	Train	Dev
Number of Positive Examples	25, 819 (10.3 %)	3,648 (10.1%)
Number of Negative Examples	225,055 (89.7%)	32,270 (89.8%)
Total Number of Examples	250, 874	35,918

Table 2: Fake review dataset summary

4.1.1 Oversampling & SMOTe

For oversampling, the imbalanced-learn library, Synthetic Minority Over-sampling Technique (SMOTe), was used to resample data. [Chakrabarty, 2019]. Since there are more negative examples, the positive examples were scaled multiple times in order to compensate for imbalance data.

4.1.2 Undersampling

For undersampling, the training set was first separated by label. The negative examples were divided into nine parts so that each part had roughly the same number of examples as the total number of positive examples. To reduce the chance of overfitting, a random sample of 80% of the positive examples were then appended to each for the parts containing negative examples (See Appendix B).

4.1.3 Text Features

To extract meaningful text features, the spaCy, string, and pyspellchecker libraries were used to tokenize the review text data. First, any spelling mistakes were corrected by pyspellchecker. Then, stop words and punctuation, as defined by spaCy and string, were removed. After that, the top words that occurred most frequently between both real and fake reviews were removed. These top overlapping words were generated based on the overlap between the top 100 recurring words in fake and real reviews. Finally, the remaining words were stemmed to reduce the number of duplicative features.

4.1.4 Behavioural Features

Since businesses with fewer reviews are more incentivized to post fake reviews [Luca and Zervas, 2013], we added a feature num_prod_reviews which is the total number of reviews a product page has received. Likewise, users who have less activity are most likely to be categorized as 'fake' by Yelp [Yelp Inc., 2013], therefore we added a feature num_user_reviews of the total number of reviews the user has given. From our exploratory analysis, we found that fake reviews were twice as likely to have a rating of '1' than real reviews (Figure X), so the rating column was also considered as a feature.

4.2 Methodology

For this project, we will be using the Average Precision Score and Area Under the Receiver Operating Characteristic Curve (ROC AUC) Score from scitkit-learn to evaluate our models. Because our dataset is heavily imbalanced, we will prioritize Average Precision when choosing our final model. For each individual model, Different parameter values were attempted to find the best model parameter combination (See Appendix A). The model with the best metrics overall will be selected as the final model.

4.3 Results

Overall, the SGDClassifier model with $\alpha = 0.1$, modified huber loss, and l2 as penalty function, performed the best.

Model (parameters)	AUC	Average Precision
SVM (loss = 'modified_huber', $\alpha = 0.1$)	0.77	0.24
RFC (N trees=100, max depth=20, min. sample split=0.05)	0.72	0.21
Naive Bayes (α=3.0)	0.71	0.21
KNN (NN=13)	0.53	0.11

Table 3: Summarized model results

5 Discussion & Conclusion

However, for the fake reviews we expected ratings of 1 and 5 to have the highest frequency but it appears that while 5 had the highest frequency there was a huge margin between 4 and 5 which was unlike what we saw in the real-reviews also as expected reviews with a 1 rating had a higher frequency than those withe 2 3 ratings.

One of the major challenges was a larger overlap of words in both 'fake' and 'real' reviews. In this project,

fake and real reviews were tokenized to run various algorithms. However, since there were a large group of frequent words that appeared in both fake and real reviews, this impacted performance across all models. This also explains why the KNN algorithm had the worst accuracy rates. Since KNN predicts data based on approximations with other data, having the same tokenized words appearing often in both fake and real reviews, made KNN not an ideal model for this project.

It is also worth noting that fake and real reviews are labeled by Yelp. In this project, it was uncertain whether the goal was to identify genuine fake and real reviews, or try to replicate Yelp's fake review detecting algorithm. The labels are not ground truth. Therefore, it is possible that reviews labeled as fake could be genuine review and vice versa.

As a next step, using more than one model to detect fake reviews should be considered. All models tested in this project have their own strength and weakness. Some may be suited to detect certain kinds of fake reviews but do poorly on detecting other kinds. There numerous possible motives for someone to post fake reviews, such as paid promotion, competitor, and freebies, etc. Therefore, even though SVM outperformed all other models it is possible that other models can be more efficient to detect certain genuine fake reviews.

References

[Chakrabarty, 2019] Chakrabarty, N. (2019). Application of synthetic minority over-sampling technique (smote) for imbalanced datasets. *Towards AI*.

[Doward, 2012] Doward, J. (2012). Here's proof that yelp rating can make or break a retailer. *Business Insider*.

[Filloon, 2019] Filloon, W. (2019). Yelp elite are becoming obsolete. *Eater*.

[Ghosh and Zargar, 2015] Ghosh, S. K. and Zargar, S. (2015). Detection of Review Spam in Online Review Websites. Technical report, Stony Brook University, Department of Computer Science.

[Gupta et al., 2019] Gupta, A., Gawad, A., Hule, A., Shanbhag, G., Kadam, V., and Gupta, P. (2019). Yelp Fake Review Detection. Technical report, Dublin City University, School of Computing.

[Jones et al., 2018] Jones, C., Mellachervu, N., Sinha, N., Young, J. W., and Zhen, W. (2018). Yelp Review Analysis in Spark. Technical report.

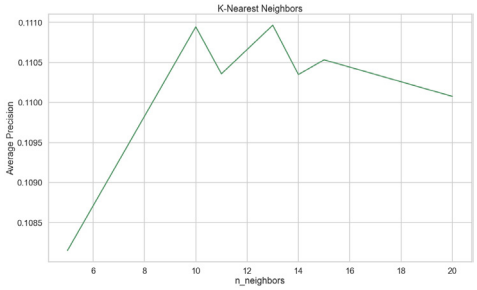
[Luca and Zervas, 2013] Luca, M. and Zervas, G. (2013). Fake It Till You Make It: Reputation, Competition, and Yelp Review Fraud. *SSRN Electronic Journal*.

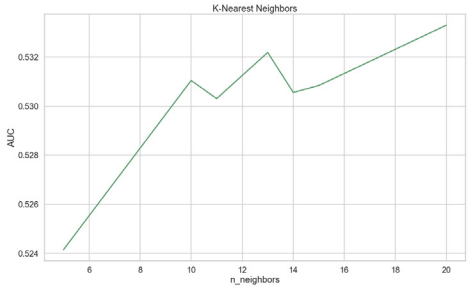
[Segal, 2011] Segal, D. (2011). Here's proof that yelp rating can make or break a retailer. *The New York Times*.

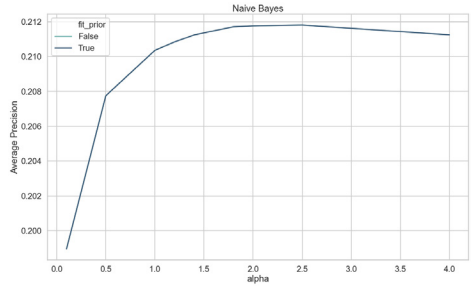
[Yelp Inc., 2013] Yelp Inc. (2013). Why does yelp recommend reviews?

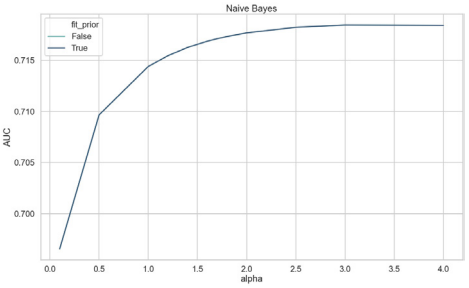
[Yelp Inc., 2020] Yelp Inc. (2020). Yelp dataset.

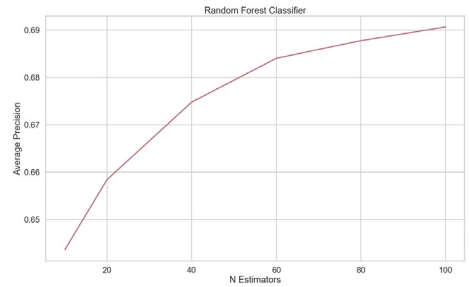
Appendix A. Average Precision and AUCs
for K-Nearest Neighbors, Random Forest Classfier, and SGDClassifier
by Different Parameters

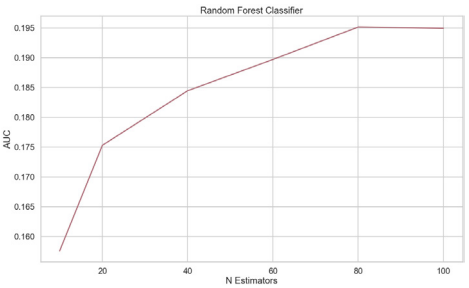


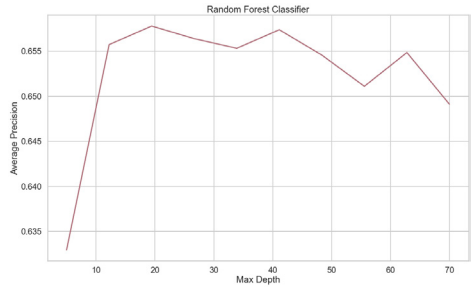


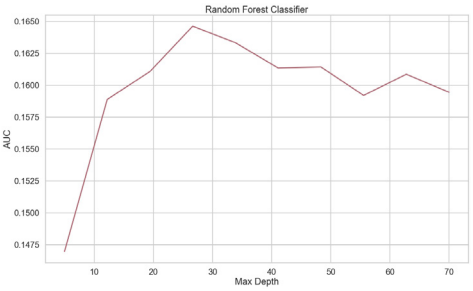


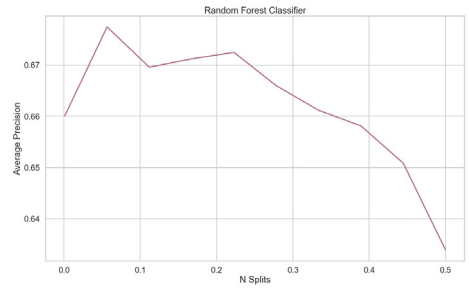


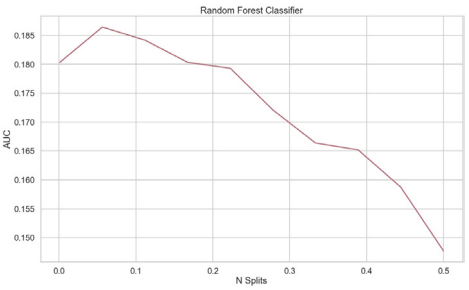


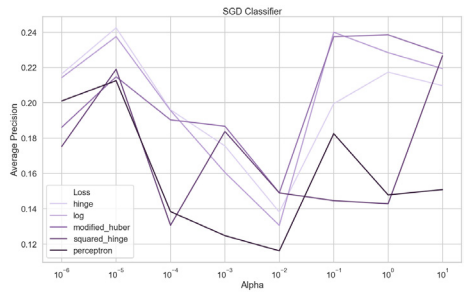


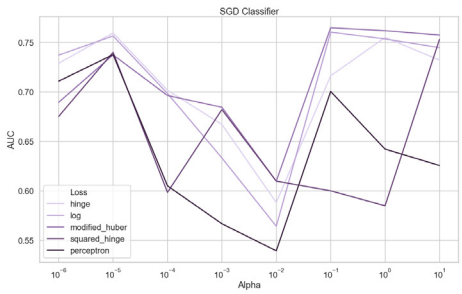












Appendix B. Undersampling Methodology