ML to Identify Fake Calgary Hotel Reviews

Group 20 Members:

Gregory Slowski, Drew Burritt, Oluwajolasun Jaiyesimi

Course:

ENSF 612 – Fall 2021 - Dr. Gias Uddin

Submission Date:

December 15, 2021

Contributions	Data Collection	Coding	Writeup
Greg	333 collected 1000 labelled 23 misclassified labelled	Feature Generation, Random Forest Model, Gradient Boosting Model	1, 2.3.2, 2.3.3, 2.4.3, 2.4.4, 2.6, 3.2, 4, 5, References, Appendix B, C
Drew	333 collected 1000 labelled 23 misclassified labelled	Preprocessing, Logistic Regression Model, SVC model	Abstract, 2.3.1, 2.4.1, 2.4.2, 3.1, References Appendix A
Oluwajolasun	333 collected, 1000 labelled, 23 misclassified labelled	Tuning parameters for best model,	2.1, 2.2, 2.5, 2.6.1, References

Databricks Notebook:
https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/3496431073893844/1401843874240607/4037204114810262/latest.html
ipynb file also submitted to D2L.

Submission for ENSF 612 Final Project

ML to Identify Fake Calgary Hotel Reviews

Gregory Slowski

University of Calgary, gregory.slowski@ucalgary.ca

Drew Burritt

University of Calgary, drew.burritt@ucalgary.ca

Oluwajolasun Jaiyesimi

University of Calgary, oluwajolasun.jaiyesimi@ucalgary.ca

Abstract: With the rise of online sales of products and services has come online reviews associated with businesses or products. Among these reviews fake or deceptive reviews have become quite prevalent. These fake reviews can either be of positive or negative in nature, from the reviewed company trying to boost their ratings, or from competitors trying to hurt competition. Potential customers usually look at these reviews before purchasing. With the rise of fake reviews it can be difficult for people to pickout truthful reviews to make an informed purchase. This paper attempts to develop a machine learning model that can distinguish between fake and truthful reviews using features generated from the review text. The strategy used in this paper was to start with a standard dataset consisting of known fake and real hotel reviews, then add 1000 additional reviews pulled from hotel websites. These added reviews were manually labelled as either fake or real. Features were extracted from these reviews using term frequency-Inverse document frequency (TF-IDF) as well as a few others such as review length, word length and sentiment score. For classification these reviews were trained and tested on three supervised machine learning models using the extracted features. The results of these models showed that logistic regression, linear support vector classifier and random forest had 82.6%, 81.7% and 68.4% accuracy respectively on a combined test set containing a subset of both reviews from the original dataset as well as the additional reviews. The best model, logistic regression, was tuned further to obtain a 85.7% accuracy on the test set.

1 Introduction

In a world where more and more consumers are making decisions on their purchases through the use of online tools that include the review sites, consumer sites themselves, blogs, and social media, the pursuit of getting the best value out of purchases is ever evolving and growing. One of the primary tools for making these choices is using consumer reviews. Although typically helpful, consumer reviews can also be plagued with fake reviews used to either bolster the overall impression of a product or service, or negatively to hurt the reputation of a competitor's product or service. The purpose of this study is to create machine learning models to assist in identifying fake, deceptive, or dishonest hotel reviews, such that consumers can trust the reviews they are using in selecting accommodations.

The models tested in this study leverage a dataset of 1600 reviews for Chicago hotels [1], as well as 1000 reviews for Calgary hotels. The Chicago dataset is an existing dataset which was also used in the similar study *Data Analytics for the Identification of Fake Reviews Using Supervised Learning* [2]. The Calgary dataset was collected from online review sites such as TripAdvisor or Google Reviews, then manually labelled as either fake or real by the three group members for this study. All 2600 reviews are preprocessed to split the data into training and testing sets of approximately 80% and 20%, respectively. Further preprocessing generates and scales features from each review such as TF-IDF features, length of reviews, average word length, sentiment scoring, and amount of punctuation in the reviews. The study uses these features to train three classification models in Logistic Regression, Linear Support Vector Classifier, and Random Forest Classification, results of which are in section 2.4. The best scoring preliminary model is explored and refined further through hyperparameter tuning in section 2.5. All misclassifications from the best scoring, hyperparameter tuned model are then explored and categorized in section 2.6.

2 Results

2.1 Data Labeling and Collection

With the original data consisting of 1600 Chicago hotel labelled reviews, based on the scope of the project we then extended it by an additional 1000 Calgary hotel reviews. These reviews were selected based on the same criteria as the research paper, manually going through hotel booking websites review sections such as booking.com, Expedia.com, Hotels.com, TripAdvisor.com etc. The major difference with our data labeling and the research paper's labeling was that they used Amazon Mechanical Turk, which is a crowdsourcing marketplace that allows people and organizations to outsource their procedures and jobs to a dispersed workforce that can complete these operations remotely [3], which they used to generate 400 deceptive positive reviews and 400 deceptive negative reviews. For this project, each group member manually collected approximately 330 reviews from hotel booking websites as mentioned above, and also each member labelled the new total 1000 reviews manually.

Since we were a group of three data labelers, in order to agree on the overall deceptiveness or truthfulness of a review (i.e. fake or real respectively), we implemented a voting system were if at least two labelers label a review as fake and the other labeler decides its real, majority carries the vote and the overall deceptiveness or truthfulness will be labelled as fake, and vice versa as shown below.

		Drew	Greg	JJ		OVERALL
Hotel	Review	real_fake	real_fake	polarity	real_fake	real_fake
The Elan	This hotel is amazing. Ar	FAKE	FAKE	positive	FAKE	fake
The Elan	Good-sized, well-furnish	real	FAKE	positive	real	real
The Elan	Clean, new, and well equ	FAKE	FAKE	positive	FAKE	fake
The Elan	Great condo style hotel a	FAKE	real	positive	FAKE	fake
The Elan	This a a great hotel for b	real	real	positive	real	real

Table 2.1.1 Section in labeling where truthfulness and deceptiveness were not all agreed

Agreement Distribution		
3 Votes Fake:	85	8.50%
3 Votes Real:	670	67.00%
2 Votes Fake:	99	9.90%
2 Votes Real:	146	14.60%

Table 2.1.2 Overall agreement distribution

This allowed us to determine our target vector that would be used for our machine learning preparation. We then combined our new dataset with the original papers dataset in our databricks python notebook for preprocessing and our selected models learning this data.

2.2 Data Comparison

As mentioned earlier the original paper dataset consists of 1600 Chicago hotel reviews of which there was a proper balance of truthful and deceptive reviews of 800 each as seen in the table below.

Original Data Distribution		
Real Positive	400	25%
Real Negative	400	25%
FakePositive	400	25%
Real Negative	400	25%

Table 2.2.1 Original paper data distribution

While our dataset consists of 1000 Calgary hotel reviews of which there is an imbalance between truthful and deceptive reviews.

New Data Distribution		
Real Positive	446	44.60%
Real Negative	368	36.80%
Fake Positive	121	12.10%
Real Negative	65	6.50%

Table 2.2.2 New data distribution

The combination of the total dataset led to an imbalance in overall truthfulness and deceptiveness of the review.

Combined Data Distribution		
Real Positive	846	32.54%
Real Negative	768	29.54%
FakePositive	521	20.04%
Real Negative	465	17.88%

Table 2.2.3 Combined data distribution

This overall imbalance occurred due to the original paper using Amazon Mechanical Turk to source deceptive review samples, and our new data manually classifying whether a review was truthful or deceptive.

2.3 Data Preprocessing

2.3.1 Text Processing

Our data was copied directly from the websites, the first step of preprocessing was removing all formatting such as new lines, as well as any non-standard characters like emojis, this was done in Excel. All further preprocessing of the review text was done using NLTK libraries [4]. This included tokenizing words, removing stopwords, removing noise and stemming. Word tokenization was done using NLTKs word_tokenize function. For removing stopwords, NLTK's standard english stopword list was used. Noise was also removed, noise was defined as punctuation, non-alphanumeric words, and any words that were less than 3 characters. Stemming was done using NLTK's Snowball stemmer for english. Stemming was done to reduce the vocabulary size before using TF-IDF.

Details of the functions used for preprocessing can be found in Appendix A

2.3.2 Feature Creation

TF-IDF utilized hashing TF to produce anywhere between 10 000 and 50 000 features, based on the vocabulary size of the corpus. TF-IDF then performed inverse document frequency (IDF) on those features using a minimum document frequency of 2, to eliminate any significant outlier words for better generalization and cut down on the vocabulary size for faster processing times.

To accommodate using TF-IDF to produce a large feature matrix, we also generated 8 additional features which included the sentiment score, polarity, length of each review, the average word length in each review, the ratio of capitalized characters, the ratio of words greater than 5 characters in length, the ratio of non-stop words, and the amount of punctuation in relation to the total number of words for each review.

Sentiment score was calculated using the NLTK library's Sentiment Intensity Analyzer on each full review before any tokenization. The value used was the compound score from the analyzer which uses the normalized combination of positive, negative, and neutral sentiment scores.

Polarity of either positive or negative was initially determined by taking the review score associated with each review from their respective websites, then later confirmed during the data labelling process.

Calculation details of the remaining six features can be found in Appendix B.

2.3.3 Machine Learning Preparation

In preparation to train models on the data available, we decided to train on as much data as possible, while also still maintaining the integrity of independent test sets without any data leakage. As shown in figure 2.3.1 this meant doing an 80 / 20 split on both the collected and original datasets, such that we could test any trained model on either only collected reviews, only original reviews, or the combination of both collected and original reviews. The result of these splits allowed us to train our models on approximately 2080 reviews, with a combined test set of approximately 520 reviews. As the datasets are first partitioned, then the splits occur after the partitions are completed, the exact number of reviews in each split varies slightly in the actual model.

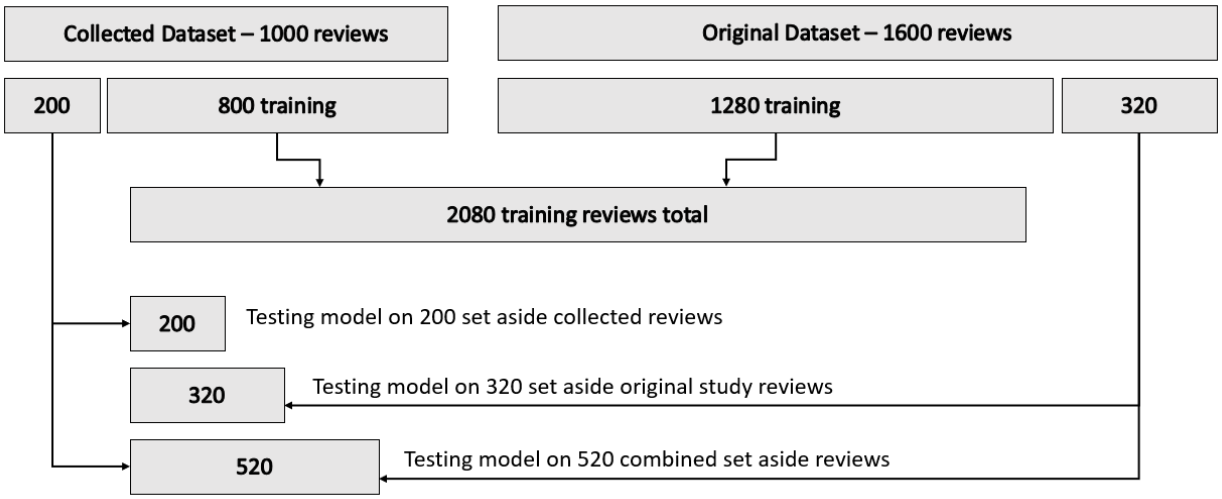

Fig. 2.3.1 *splitting of data into test and training sets*

After the training reviews were subjected to the text processing and feature generation noted above, each review was passed through a preprocessing pipeline which included TF-IDF feature creation from the tokenized text, scaling of the 8 added features using a standard scaler, and string indexing for the polarity feature and the real/fake target vector.

2.4 Model Performance for Original and New Datasets

Three different types of models were trained and tested on the datasets. Comparison of the different models was done using F1-score. For each model accuracy, precision and recall were also calculated for each of the test sets, new data, old data and combined data. The calculations for each are as follows. Results for F1-score, precision and recall are the weighted averages of both classes.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 - score = \frac{2 * precision * recall}{precision + recall}$$

2.4.1 Logistic Regression

Logistic Regression(LR) from PySpark's ML library [5] was one of the models used. Training was done using grid search with 5 fold cross validation to obtain the best hyperparameters. Cross validation was done using a parameter grid with the following parameters:

HashingTF - Number of features = 10000, 20000, 50000

LR - Regularization = 0.1, 0.3, 0.5

Evaluation for each set of parameters was done with the default metric of F1-score. The F1-scores for all parameter combinations are shown below in table 2.4.1.1.

Table 2.4.1.1 F1-scores for grid search cross validation of Linear Support Vector Classifier

:

	f1	numFeatures	regParam
0	0.791431	10000	0.1
1	0.785407	10000	0.3
2	0.780055	10000	0.5
3	0.789159	20000	0.1
4	0.783702	20000	0.3
5	0.780651	20000	0.5
6	0.800504	50000	0.1
7	0.789846	50000	0.3
8	0.782900	50000	0.5

From this grid search it was determined that the best parameters were 50000 and 0.1 for number of features and regularization respectively, resulting in an F1-score of 0.801. PySpark's Multiclass Classification Evaluator was used to obtain further metrics on the model performance. Accuracy, F1-score, precision and recall are shown below for the model with the best parameters in figure 2.4.1.1. In the results, new data is the data from Calgary hotels that we collected, old data is the original dataset of Chicago hotels.

```
new data accuracy:  0.7766990291262136
new data f1:  0.7498168897754367
new data precision:  0.7313915857605179
new data recall:  0.7766990291262136
old data:  0.8563049853372434
old data f1:  0.8562234001424587
old data precision:  0.8579280116193675
old data recall:  0.8563049853372434
combined data:  0.8263254113345521
combined data f1:  0.8236641191470369
combined data precision:  0.8253535150332125
combined data recall:  0.8263254113345522
```

Figure 2.4.1.1 Accuracy, F1-score, precision, and recall for Logistic Regression model on the new, original, and combined dataset

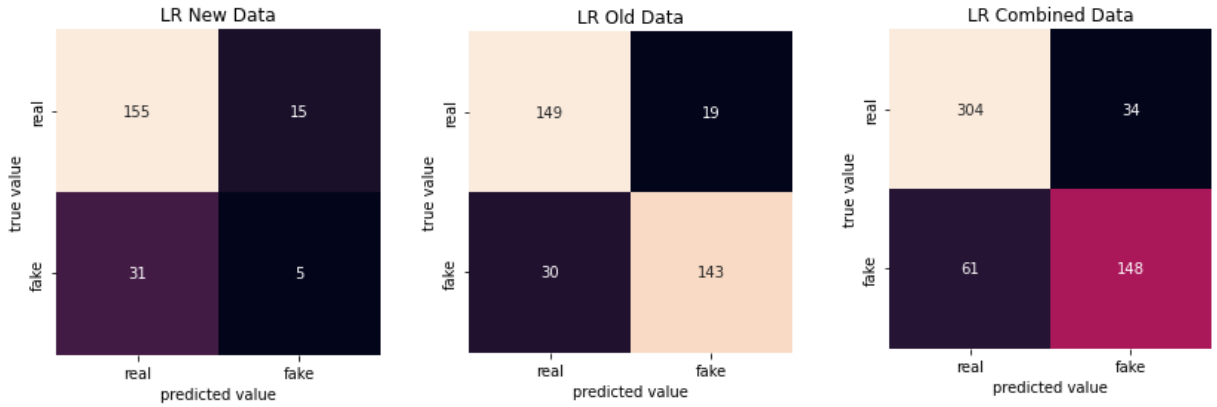

Figure 2.4.1.2 Confusion matrices for Logistic Regression model on the new, original, and combined dataset showing true negatives (TN), false positives (FP), false negatives (FN), and true positives (TP)

As shown by these scores as well as the confusion matrices in figure 2.4.1.2, Logistic Regression performs quite well on the data with minimal tuning, having an accuracy of 82.6% on the combined test set. Precision and recall are very similar showing that this model is neither overfitting or underfitting. As shown in the confusion matrices, the model is more likely to wrongly predict a review as real over fake. The model does perform better on the original data, likely due to the original data being more balanced, as well as having completely certain labels whereas ours were labelled manually.

2.4.2 Linear Support Vector Classification (Linear SVC)

Linear Support Vector Classification (linear SVC) was the second model tested.Training was done using grid search with 5 fold cross validation to obtain the best hyperparameters. Cross validation was done using a parameter grid with the following parameters:

HashingTF - Number of features = 10000, 20000, 50000

SVC - Regularization = 0.001, 0.01, 0.01, 1.0

Evaluation for each set of parameters was done with the default metric of F1-score. The F1-scores for all parameter combinations are shown below in table 2.4.2.1

Table 2.4.2.1 F1-scores for grid search cross validation of Linear Support Vector Classifier

	f1	numFeatures	regParam
0	0.764708	10000	0.001
1	0.771835	10000	0.010
2	0.771835	10000	0.010
3	0.776334	10000	1.000
4	0.759178	20000	0.001
5	0.765598	20000	0.010
6	0.765598	20000	0.010
7	0.769944	20000	1.000
8	0.764153	50000	0.001
9	0.766673	50000	0.010
10	0.767398	50000	0.010
11	0.771298	50000	1.000

From this grid search it was determined that the best parameters were 10000 and 1 for number of features and regularization respectively, resulting in an F1-Score of 0.771 PySpark's Multiclass Classification Evaluator was used to obtain further metrics on the model performance. Accuracy, F1-score, precision and recall are shown below for the SVC model with the best parameters in figure 2.4.2.1.

```
new data accuracy:  0.8058252427184466
new data f1:  0.7443490647374142
new data precision:  0.7099514563106796
new data recall:  0.8058252427184466
old data:  0.8240469208211144
old data f1:  0.8237803252466009
old data precision:  0.827162198268406
old data recall:  0.8240469208211143
combined data:  0.8171846435100548
combined data f1:  0.8117053014057447
combined data precision:  0.8192416083452032
combined data recall:  0.8171846435100548
```

Figure 2.4.2.1 Accuracy, F1-score, precision, and recall for Linear SVC model on the new, original, and combined dataset

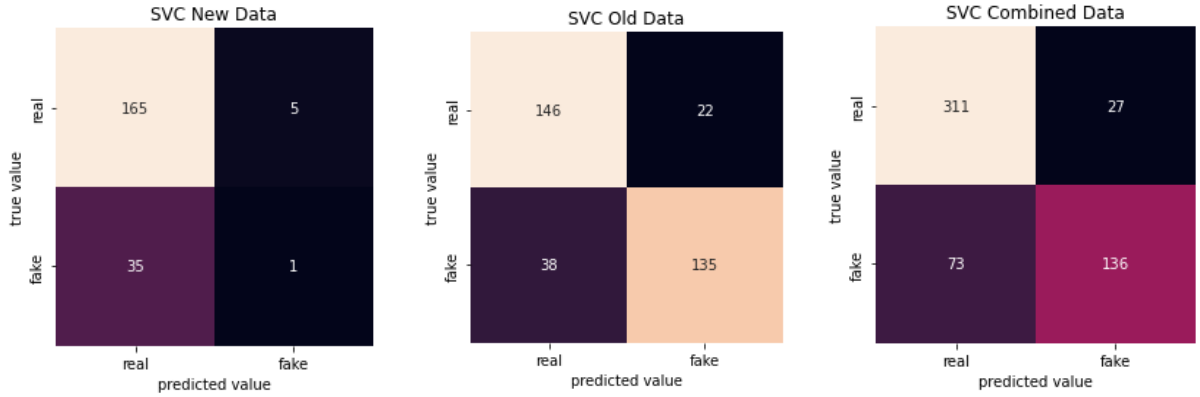

Figure 2.4.2.2 Confusion matrices for Linear SVC model on the new, original, and combined dataset showing true negatives (TN), false positives (FP), false negatives (FN), and true positives (TP)

As shown by these scores and the confusion matrices in figure 2.4.2.2, Linear SVC performed well on the new data and the old data, having an accuracy of 81.7% on the combined set, only slightly worse than logistic regression that had 82.6% accuracy. Similar to logistic regression the SVC had more false negatives than false positives, likely for the same reasons discussed above in section 2.4.1.

2.4.3 Random Forest Classifier

The last model trained was using RandomForestClassifier (RFC) from PySpark ML classification library. After the aforementioned preprocessing of data, the random forest model was generated by using a grid search with, 5 fold cross validation using the following hyperparameters:

HashingTF - Number of features = 10000, 20000, 50000
Max tree depth = 3, 5, 9
Number of trees = 10, 20, 50

The scoring used in the grid search was using F1-score to determine the best set of parameters to use on the test data. The resulting F1-scores for each set of parameters can be seen in table 2.4.3.1.

	f1	numFeatures	maxDepth	numTrees
0	0.480488	10000	3	10
1	0.479452	10000	3	20
2	0.476418	10000	3	50
3	0.507955	10000	5	10
4	0.491458	10000	5	20
5	0.494541	10000	5	50
6	0.584245	10000	9	10
7	0.572209	10000	9	20
8	0.568334	10000	9	50
9	0.487361	20000	3	10
10	0.478572	20000	3	20
11	0.476418	20000	3	50
12	0.515538	20000	5	10
13	0.487017	20000	5	20

	f1	numFeatures	maxDepth	numTrees
14	0.480769	20000	5	50
15	0.577798	20000	9	10
16	0.554077	20000	9	20
17	0.535860	20000	9	50
18	0.483851	50000	3	10
19	0.477560	50000	3	20
20	0.476418	50000	3	50
21	0.499991	50000	5	10
22	0.481943	50000	5	20
23	0.477506	50000	5	50
24	0.533764	50000	9	10
25	0.509176	50000	9	20
26	0.504979	50000	9	50

Table 2.4.3.1 F1-scores for grid search cross validation of Random Forest Classification

The best cross-validation model resulted in an F1-score of 0.584 for 10000 TF features, a max tree depth of 9 with 10 trees. Using this best model we transformed the original data test set, the new data test set, and the combined data test set to produce a prediction vector for each set. Using each of these prediction vectors accuracy, F1-score, precision, and recall scores were generated as shown in figure 2.4.3.1. For an extension of these scores a heatmap for each resulting data set was generated showing true positives (TP), false negatives (FN), false positives (FP), and true negatives (TN) as seen in figure 2.4.3.2.

```
new data accuracy:  0.8252427184466019
new data f1:  0.7462301177442676
new data precision:  0.6810255443491375
new data recall:  0.8252427184466019
old data:  0.5982404692082112
old data f1:  0.545471240426477
old data precision:  0.6957179982357009
old data recall:  0.5982404692082111
combined data:  0.6837294332723949
combined data f1:  0.6200275836187676
combined data precision:  0.7307651422165033
combined data recall:  0.6837294332723949
```

Figure 2.4.3.1 accuracy, F1-score, precision, and recall for Random Forest Classification model on the new, original, and combined dataset

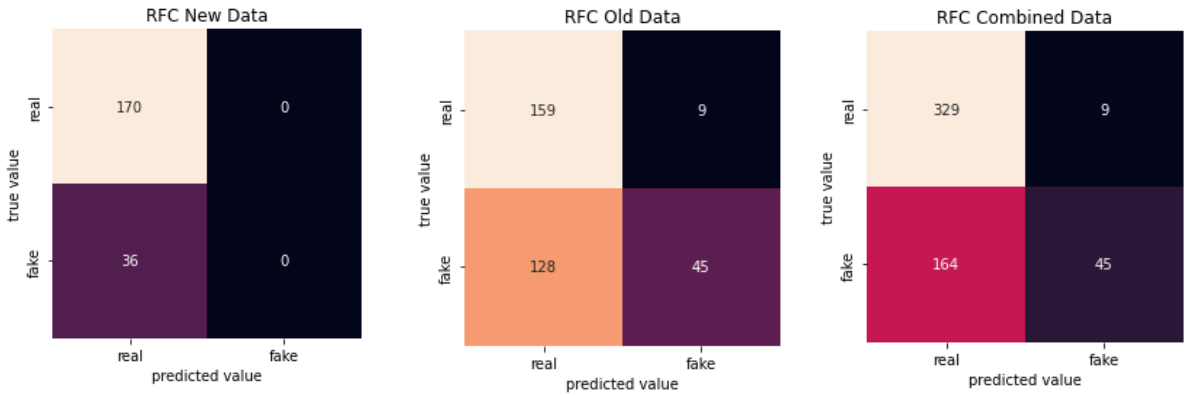

Figure 2.4.3.2 confusion matrices for Random Forest Classification model on the new, original, and combined dataset showing true negatives (TN), false positives (FP), false negatives (FN), and true positives (TP)

Random forest performed quite poorly as indicated by F1-scores and accuracy scores of < 60% on the original data test sets, while only performing a bit better on the new data test set.The confusion matrix for the new data test set in figure 2.4.3.2 makes it apparent that the only reason it is performing better on the new dataset is that it is simply just predicting every single review is real, hence the accuracy score lines up with the datasets being imbalanced towards real reviews. The poor scoring of random forest can likely be accounted for by the fact that the data can only be split as many times as there is depth of trees, so on a sparse, many featured matrix one would have to sacrifice a lot of performance to get sufficiently deep trees to label the datasets well. For this reason our group chose not to continue with further study of tree based models including both random forest and gradient boosting trees.

2.4.4 Gradient Boosting and Naive Bayes Classifiers

Initially, gradient boosting models were considered to model classification of the reviews. After attempting some preliminary parameter tuning on a gradient boosting model it was quickly determined that the amount of time required to train the model, combined with the fact that random forest classification performed so poorly and the initial gradient boosting models were doing just as poorly, it was simply not worth pursuing further as tree based models were not performing well with the sparse data.

A Naive Bayes model was initially considered for study, but the current version of PySpark (2.4.0) the group was working with did not support a Naive Bayes gaussian model for continuous data. With other models already performing well, the group decided to not continue with Naive Bayes modeling.

2.5 Model Performance Based on Hyperparameter Tuning

After Logistic Regression(LR) as mentioned above gave us an accuracy of 82.6%, we decided to add more values for the initial parameters and another parameter, standardization, to see if we could get better performance from the initial result. Further training was done using grid search with 5 fold cross validation, adding more values to the initial parameters for the number of features such as 5000 and 30000, adding the minimum regularization parameter value of 0.0 and a new parameter standardization with the following values:

HashingTF - Number of features = 5000,10000,20000,30000,50000
LR - Regularization = 0.0, 0.1, 0.3, 0.5
LR - Standardization = True, False

numFeatures = Param(parent='undefined', name='numFeatures', doc='Number of features. Should be greater than 0.')
regParam = Param(parent='undefined', name='regParam', doc='regularization parameter (>= 0).')
standardization = Param(parent='undefined', name='standardization', doc='whether to standardize the training features before fitting the model.')

The evaluation for each set of parameters was done with the default metric of F1-score. The F1-scores for all parameter combinations are shown below in table 2.5.1

	f1	numFeatures	regParam	standardization
0	0.755115	5000	0.0	True
1	0.755115	5000	0.0	False
2	0.797848	5000	0.1	True
3	0.812191	5000	0.1	False
4	0.791091	5000	0.3	True
5	0.802436	5000	0.3	False
6	0.784227	5000	0.5	True
7	0.773065	5000	0.5	False
8	0.757075	10000	0.0	True
9	0.757075	10000	0.0	False
10	0.799572	10000	0.1	True
11	0.816591	10000	0.1	False
12	0.796400	10000	0.3	True
13	0.806925	10000	0.3	False
14	0.789723	10000	0.5	True
15	0.781105	10000	0.5	False
16	0.754398	20000	0.0	True
17	0.754398	20000	0.0	False
18	0.796579	20000	0.1	True
19	0.815359	20000	0.1	False
20	0.793592	20000	0.3	True
21	0.805224	20000	0.3	False
22	0.780627	20000	0.5	True
23	0.778583	20000	0.5	False
24	0.746646	30000	0.0	True
25	0.746646	30000	0.0	False
26	0.796748	30000	0.1	True
27	0.812454	30000	0.1	False
28	0.795744	30000	0.3	True
29	0.803892	30000	0.3	False
30	0.786312	30000	0.5	True
31	0.783057	30000	0.5	False
32	0.749340	50000	0.0	True
33	0.749340	50000	0.0	False
34	0.791006	50000	0.1	True
35	0.811776	50000	0.1	False
36	0.793646	50000	0.3	True
37	0.805118	50000	0.3	False
38	0.786196	50000	0.5	True
39	0.777699	50000	0.5	False

Table 2.5.1 F1-scores for grid search 5 fold cross validation of Logistic Regression with an extra parameter

From table 2.5.1 grid search, it was determined that the best parameters were, hashingTF number of features of 10000, regularization parameter of 0.1 and standardization parameter of False. Accuracy, F1-score, precision and recall are shown below for the model with the overall best parameters in figure 2.5.1 In the results, new data is the data from Calgary hotels that we collected, old data is the original dataset of Chicago hotels, and the combination of both.

```
new data accuracy:  0.8212290502793296
new data f1:  0.7839164951484859
new data precision:  0.7657135301312353
new data recall:  0.8212290502793297
old data:  0.8774834437086093
old data f1:  0.8776758957056398
old data precision:  0.8801960613454165
old data recall:  0.8774834437086093
combined data:  0.8565488565488566
combined data f1:  0.8540718908990863
combined data precision:  0.8583665250331917
combined data recall:  0.8565488565488566
```

Figure 2.5.1 accuracy, F1-score, precision, and recall for Best Logistic Regression Classification model on the new, original, and combined dataset

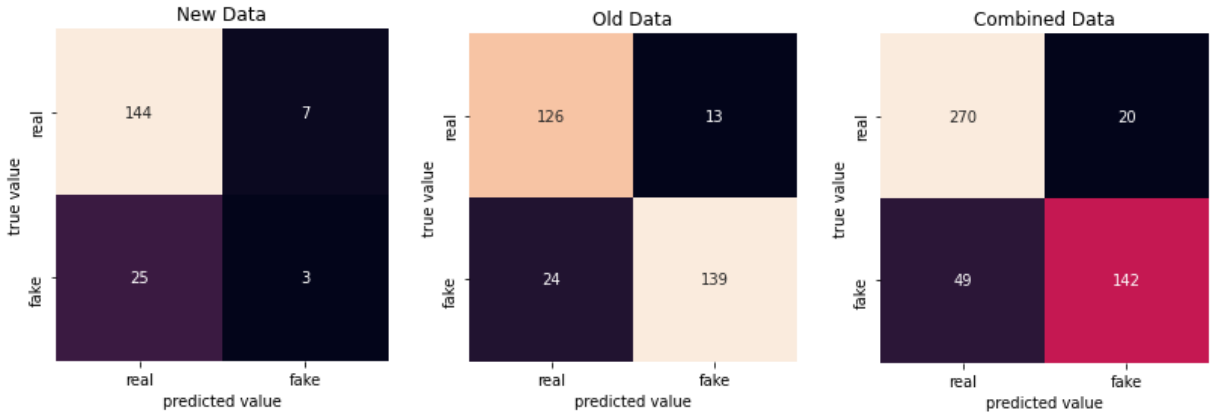

Figure 2.5.2 confusion matrices for Tuned Logistic Regression model on the new, original, and combined dataset showing true negatives (TN), false positives (FP), false negatives (FN), and true positives (TP)

As shown by the scores in figure 2.5.1, as well as the confusion matrices in figure 2.5.2, Logistic Regression performs quite well with a very slight increase in accuracy of 82.1% on the new test data set, 87.7% on the old test set and 85.7% on the combined test set, generally the model performs slightly better with parameter tuning, and we see the precision and recall score are very close, similar to how close they were pre-parameter tuning showing that this model is neither over fitting or underfitting. The confusion matrix for the combined data had 49 false negatives (FN) and 20 false positives (FP), which sums up to 69, the total number of reviews that were mislabelled which will be elaborate on in section 2.6 below.

Machine learning model used	Original paper result	Proposed work result
Logistic Regression (LR)	-	85.7%
Linear Support Vector Classification (Linear SVC)	93%	81.7%
Random Forest Classifier (RFC)	95%	68.4%

Table 2.5.2 Comparison of original paper results versus our proposed work results

Our models did not perform as well compared to the original papers model's results due to the fact there was an imbalance classifying the labels of our new data and combining it with the original while the original paper had a perfectly balanced distribution of truthful and deceptive reviews as mentioned in section 2.1. As mentioned in section 2.2 and 2.4.1 the labels in the original dataset were 100% certain as they generated fake reviews for the purpose of creating the dataset, whereas ours were manually labelled therefore not completely certain, introducing some error.

2.6 Misclassifications of Labels

In total of the 481 combined test reviews, only 69 were mislabelled. Each group member went through 23 reviews to identify reasons why the classifications were mislabelled. In this process 10 different categories, as shown in table 2.6.1, were created. The categories were broken down into 6 types for false positives (real predicted as fake) and 4 types for false negatives (fake predicted as real). Some of the reviews were given multiple reasons for misclassification, hence why a total of 105 reasons were produced.

Category Type	Categories	Count	Percentage
False Positive	Repeated words	7	7%
False Positive	Generic words	7	7%
False Positive	Misspelled words	4	4%
False Positive	Describe all amenities	5	5%
False Positive	Excessive punctuation	7	7%
False Positive	Excessive capitalization	1	1%
False Negative	Diverse words	16	15%
False Negative	Unique words	27	26%
False Negative	Proper grammar	12	11%
False Negative	First person voice	19	18%
		105	100%

Table 2.6.1 potential causes for misclassification using the finalized logistic regression model

Table 2.6.1 shows that for false negatives, it was common that having many unique words tricked the models into thinking the review was real, but overall also having diverse words, proper grammar, and first person voice also commonly contributed to these misclassifications. On the other hand false positives were much less common, where the model was thinking these reviews were fake. Not any of these categories stood out but things like using generic or repeated vocabularies, and using too much punctuation or capitalization contributed to the misclassifications. Examples of each category can be found in Appendix C, while the full table can be found in the attached document "Misclassifications With Index.xlsx".

2.6.1 Tag Cloud For keywords

A tag cloud is a graphic representation of text data that is commonly used to display keyword information on websites or to view free-form text. Tags are generally single words, with the font size or color indicating the relevance of each tag [6]. For this project, tag clouds were considered to graphically identify the repeated, common words, generic words as seen in our misclassification of labels section in the combined dataset.

Figure 2.6.1.1 Tag clouds for real reviews of combined dataset

Figure 2.6.1.2 Tag clouds for fake reviews of combined dataset

3 Discussions

3.1 Applying Model to Other Types of Reviews (Drew)

The original paper focused on hotel reviews and we extended it with different hotels. Although hotels are used, the models could be easily adapted for other types of reviews. Such as online sales sites like Amazon or Ebay where reviews are left on products. The model could also be used for general google reviews like non-hotel businesses. As discussed in 3.2 monitoring reviews can be an important part of review hosting sites. Especially on product sites like Amazon where fake reviews are very common. Accurately removing fake reviews gives customers more accurate representation of the product.

3.2 Review Site Quality Control Application (Greg)

In practice this modeling could be used by review sites to try to keep all of the reviews on their site as honest reviews, so consumers can know to trust reviews on that site. If we control the model to get high recall, where we would identify as many potential fake reviews as possible. In doing this you could have a customer service team reviewing potential fake reviews and deleting them as necessary. From a business standpoint if only half the reviews are passed to a manual review process, the cost of labor in reviewing these reviews should also be halved.

3.3 Real Life Application (JJ)

In real life scenarios, this model could be used as a means to filter any sort of application that receives input from an end-user, such as this projects Hotel reviews, Movie reviews, Online products reviews, online service reviews, etc. applying this model on such applications would most likely enable end-users who actually go through reviews before making a purchase or reservation of goods and services to make a solid decision on whatever choice they want to make, provided all or majority of the reviews are truthful reviews regardless of the polarity (i.e Negative or Positive)

4 Conclusions

The purpose of this study was to create a machine learning model that could adequately predict whether or not reviews for hotels on the internet were real or fake. Using a complete dataset of 2600 total reviews leveraging 1600 existing Chicago hotel reviews, and 1000 collected and labelled Calgary hotel reviews, Logistic Regression, Linear SVC, and Random Forest classification models were trained on the preprocessed data. The preprocessing included splitting the datasets into training and testing sections, completing text processing on the reviews, and generating features using TF-IDF and simple calculated summaries of each review like length and average word length, with scaling.

Using a grid search with cross validation method on each preliminary model, it was found that logistic regression was able to achieve an accuracy score on the combined test set of 82.6% with 50000 TF-hashing features and a regularization parameter of 0.1 Linear SVC was able to achieve an accuracy score of 81.7% on the combined test set using 10000 TF-hashing features and regularization parameter of 1. The Random Forest Classifier achieved an accuracy of 68.4% on the combined test set using 10000 TF-hashing features, a max tree depth of 9 with 10 total trees.

Due to its strong initial performance, Logistic Regression was chosen to continue with further hyper-parameter tuning where an accuracy of 85.7% was attained on the combined test set using 10000 TF-hashing features, a regularization parameter of 0.1, and setting standardization to false within the model.

Using this model 69 reviews of the 481 total reviews in the combined test set were mislabelled by the model. Each mislabelled review was categorized into 10 different categories as to why, the most common of which was because of false negative reviews (fake review predicted as real) having many unique and distinct words.

As the final model is quite accurate at an 85.7% correct identification of real or fake reviews, this model can be deployed into applications such as applying this model to hotel reviews or other consumer reviews, applying the model to a selection for review process in a quality control environment, or for a consumer to test various product reviews on the likelihood of each review being real or not.

5 Citing Related Work

The purpose of this paper was to take an existing paper with an associated dataset, do a similar analysis process, and expand on it some as seen fit. The paper used was *Data Analytics for the Identification of Fake Reviews Using Supervised Learning* [2] where they explored similar analysis with the 1600 original Chicago reviews dataset [1]. This paper underwent a similar machine learning modeling approach but used naive Bayes, SVM, adaptive boosting, and random forest.

6 Acknowledgements

Dr. Gias Uddin, for allowing the opportunity to conduct this study for ENSF 612 at the University of Calgary.

References

[1] **M. Ott, C. Cardie and J. T. Hancock, *Negative Deceptive Opinion Spam* v1.4 in Proc. of the 49th Annual Meeting of the Association for Computational Linguistics, Portland, Oregon, pp. 309–319, 2011. Accessed on: Oct 10, 2021. [Online]. Available: https://myleott.com/op-spam**

[2] **Alsubari, S. N., Deshmukh, S. N., Alqarni, A. A., Alsharif, N., H., T. et al. Data Analytics for the Identification of Fake Reviews Using Supervised Learning. *CMC-Computers, Materials & Continua, 70(2)*, 3189–3204, September 27, 2021. Accessed on: Oct 10, 2021. [Online]. Available: https://www.techscience.com/cmc/v70n2/44647**

[3] **Amazon, "Amazon Mechanical Turk," Amazon Mechanical Turk, 2005. Accessed on: Oct 10, 2021. [Online]. Available: https://www.mturk.com/.**

[4] **S. Bird, E. Loper, E. Klein. et al. (2021) Natural Language Toolkit (NLTK) (Version 3.6.5) [Python Library] https://www.nltk.org/index.html**

[5] **PySpark (2018) PySpark (Version 2.4.0) [API Framework] https://spark.apache.org/docs/2.4.0/api/python/index.html**

[6] **Techopedia, "Tag Cloud," *Techopedia*, 11-May-2015 [Online]. Accessed on: Dec, 14 2021 Available: https://www.techopedia.com/definition/5200/tag-cloud.**

Appendices

A Text Preprocessing

Using the following review we will demonstrate the preprocessing of review text

Review
Our 'getaway' hotel in our own city. Since its opening, my wife and I have stayed at Le Germain whenever we want to escape without escaping the city. The decor, amenities, location to amazing dining, and the hotel staff themselves will always bring us back!

A.1 Word Tokenization

Pyspark function:

```
#Drew
#Create UDF for tokenizing words
def tokenize1(text):
    words = nltk.word_tokenize(text)
    return words
tokenize_word = udf(lambda x: tokenize1(x)  , ArrayType(StringType()))
```

Tokenized output:

tokenized_words
["Our", "'getaway", "", "hotel", "in", "our", "own", "city", ".", "Since", "its", "opening", ",", "my", "wife", "and", "I", "have", "stayed", "at", "Le", "Germain", "whenever", "we", "want", "to", "escape", "without", "escaping", "the", "city", ".", "The", "decor", ",", "amenities", ",", "location", "to", "amazing", "dining", ",", "and", "the", "hotel", "staff", "themselves", "will", "always", "bring", "us", "back", "!"]

A.2 Removing Stopwords

Pyspark function:

```
#Drew
#Create UDF for removing stopwords
from nltk.corpus import stopwords
nltk.download('stopwords')
nltk.download('punkt')
stop_en = stopwords.words('english')
def remove_stopwords1(word_list):
    filtered_words = [word for word in word_list if word not in stop_en]
    return filtered_words
remove_stopwords = udf(lambda x: remove_stopwords1(x) , ArrayType(StringType()))
```

Output:

no_stopwords

▸ ["Our", "'getaway", """, "hotel", "city", ".", "Since", "opening", ",", "wife", "I", "stayed", "Le", "Germain", "whenever", "want", "escape", "without", "escaping", "city", ".", "The", "decor", ",", "amenities", ",", "location", "amazing", "dining", ",", "hotel", "staff", "always", "bring", "us", "back", "!"]

A.3 Removing Noise

Pyspark function:

```
#Drew
#Create UDF for removing noise
def remove_noise1(word_list):
    filtered_words = [word for word in word_list if word.isalnum() and len(word)>2]
    return filtered_words
remove_noise = udf(lambda x: remove_noise1(x) , ArrayType(StringType()))
```

Output:

no_noise

▸ ["Our", "hotel", "city", "Since", "opening", "wife", "stayed", "Germain", "whenever", "want", "escape", "without", "escaping", "city", "The", "decor", "amenities", "location", "amazing", "dining", "hotel", "staff", "always", "bring", "back"]

A.4 Stemming

Pyspark function:

```
#Drew
#Create UDF for stemming words
from nltk.stem import SnowballStemmer
def stem1(word_list):
    snowball = SnowballStemmer(language='english')
    stemmed_words = [snowball.stem(word) for word in word_list]
    return stemmed_words
stem = udf(lambda x: stem1(x) , ArrayType(StringType()))
```

Output:

stemmed

▸ ["our", "hotel", "citi", "sinc", "open", "wife", "stay", "germain", "whenev", "want", "escap", "without", "escap", "citi", "the", "decor", "amen", "locat", "amaz", "dine", "hotel", "staff", "alway", "bring", "back"]

B Generated Features Calculation

Using the following review we will demonstrate how these additional features were calculated.

Review
Our 'getaway' hotel in our own city. Since its opening, my wife and I have stayed at Le Germain whenever we want to escape without escaping the city. The decor, amenities, location to amazing dining, and the hotel staff themselves will always bring us back!

The review after tokenization:

tokenized_words
["Our", "'getaway", "", "hotel", "in", "our", "own", "city", ".", "Since", "its", "opening", ",", "my", "wife", "and", "I", "have", "stayed", "at", "Le", "Germain", "whenever", "we", "want", "to", "escape", "without", "escaping", "the", "city", ".", "The", "decor", ",", "amenities", ",", "location", "to", "amazing", "dining", ",", "and", "the", "hotel", "staff", "themselves", "will", "always", "bring", "us", "back", "!"]

B.1 Length of Review

Pyspark function:

```
#greg
#create udf for adding length of reviews by word count for all alphanumeric "words"

def get_length1(word_list):
  alphanum_words = [word for word in word_list if word.isalnum()]
  N = len(alphanum_words)
  return N

get_length = udf(lambda x: get_length1(x), IntegerType())
```

Sample Calculation:

alphanum_words = ["Our", "'getaway", "hotel", "in", "our", "own", "city", "Since", "its", "opening", "my", "wife", "and", "I", "have", "stayed", "at", "Le", "Germain", "whenever", "we", "want", "to", "escape", "without", "escaping", "the", "city", "The", "decor", "amenities", "location", "to", "amazing", "dining", "and", "the", "hotel", "staff", "themselves", "will", "always", "bring", "us", "back"]
N = len(alphanum_words)
N = 44

B.2 Average Word Length

Pyspark function:

```
#greg
#create udf for adding average word length by review

def get_average_word_length1(word_list):
  alphanum_word_lengths = [len(word) for word in word_list if word.isalnum()]
  avg_word_len = sum(alphanum_word_lengths)/len(alphanum_word_lengths)
  return avg_word_len

get_average_word_length = udf(lambda x: get_average_word_length1(x), FloatType())
```

Sample Calculation:

alphanum_word_lengths = [3, 7, 5, 2, 3, 3, 4, 5, 3, 7, 2, 4, 3, 1, 4, 6, 2, 2, 7, 8, 2, 4, 2, 6, 7, 8, 3, 4, 3, 5, 9, 8, 2, 7, 6, 3, 3, 5, 5, 10, 4, 6, 5, 2, 4]
Avg_word_len = sum(alphanum_word_lengths) / N = 197 / 44
Avg_word_len = 4.477

B.3 Ratio of Capitalized Characters to Total Characters (n), non-whitespace

Pyspark function:

```
#greg
#create udf for ratio of capitalized characters to total characters (n), with no white spaces

def capital_ratio1(word_list):
  s = "".join(word_list)
  n = len(s)
  cap_n = len([char for char in s if char.isupper()])
  ratio_cap_to_total = cap_n / n
  return ratio_cap_to_total

capital_ratio = udf(lambda x: capital_ratio1(x), FloatType())
```

Sample Calculation:

s = “Ourgetaway’hotelinourowncity.Sinceitsopening,mywifeandIhavestayedatLeGermainwheneverwewanttoescapewithoutescapingthecity.Thedecor,amenities,locationtoamazingdining,andthehotelstaffthemselveswillalwaysbringusback!”

n = len(s) = 213
cap_n = len([“O”, “S”, “I”, “L”, “G”, “T”]) = 6
ratio_cap_to_total = 6 / 213
ratio_cap_to_total = 0.0282

B.4 Ratio of long words (>5 letters) to total words (N)

Pyspark function:

```
#greg
#create udf for ratio of long words (>5 letters) to total words (N)

def long_word_ratio1(word_list):
  alphanum_words = [word for word in word_list if word.isalnum()]
  N = len(alphanum_words)
  long_words = [word for word in alphanum_words if len(word) > 5]
  long_words_count = len(long_words)
  ratio_long_to_total = long_words_count / N
  return ratio_long_to_total

long_word_ratio = udf(lambda x: long_word_ratio1(x), FloatType())
```

Sample Calculation:

alphanum_words = ["Our", "'getaway", "hotel", "in", "our", "own", "city", "Since", "its", "opening", "my", "wife", "and", "I", "have", "stayed", "at", "Le", "Germain", "whenever", "we", "want", "to", "escape", "without", "escaping", "the", "city", "The", "decor", "amenities", "location", "to", "amazing", "dining", "and", "the", "hotel", "staff", "themselves", "will", "always", "bring", "us", "back"]

N = len(alphanum_words) = 44

long_words = ['opening', 'stayed', 'Germain', 'whenever', 'escape', 'without', 'escaping', 'amenities', 'location', 'amazing', 'dining', 'themselves', 'always']

long_words_count = len(long_words) = 13

ratio_long_to_total = long_words_count / N = 13 / 44

ratio_long_to_total = 0.295

B.5 Ratio of words after stop word removal to total words (N)

Pyspark function:

```
#greg
#create udf for ratio of words after stop word removal vs total words

stop_en = stopwords.words('english')

def filtered_word_ratio1(word_list):
  alphanum_words = [word for word in word_list if word.isalnum()]
  N = len(alphanum_words)
  non_filtered_words = [word for word in alphanum_words if word not in stop_en]
  num_non_filtered_words = len(non_filtered_words)
  ratio_non_filtered_to_total = num_non_filtered_words / N
  return ratio_non_filtered_to_total

filtered_word_ratio = udf(lambda x: filtered_word_ratio1(x), FloatType())
```

Sample Calculation:

alphanum_words = ["Our", "'getaway", "hotel", "in", "our", "own", "city", "Since", "its", "opening", "my", "wife", "and", "I", "have", "stayed", "at", "Le", "Germain", "whenever", "we", "want", "to", "escape", "without", "escaping", "the", "city", "The", "decor", "amenities", "location", "to", "amazing", "dining", "and", "the", "hotel", "staff", "themselves", "will", "always", "bring", "us", "back"]

N = len(alphanum_words) = 44

non_filtered_words = ['Our', 'hotel', 'city', 'Since', 'opening', 'wife', 'I', 'stayed', 'Le', 'Germain', 'whenever', 'want', 'escape', 'without', 'escaping', 'city', 'The', 'decor', 'amenities', 'location', 'amazing', 'dining', 'hotel', 'staff', 'always', 'bring', 'us', 'back']

num_non_filtered_words = len(non_filtered_words) = 28

ratio_non_filtered_to_total = num_non_filtered_words / N = 28 / 44

ratio_non_filtered_to_total = 0.636

B.6 Ratio of punctuation to total words (N)

Pyspark function:

```
#greg
#create udf for ratio of punctuation "words" vs total alphanumeric words

def punctuation_word_ratio1(word_list):
  alphanum_words = [word for word in word_list if word.isalnum()]
  N = len(alphanum_words)
  punctuation_words = [word for word in word_list if not(word.isalnum())]
  num_punctuation_words = len(punctuation_words)
  ratio_punctuation_to_total = num_punctuation_words / N
  return ratio_punctuation_to_total

punctuation_word_ratio = udf(lambda x: punctuation_word_ratio1(x), FloatType())
```

Sample Calculation:

alphanum_words = ["Our", "'getaway", "hotel", "in", "our", "own", "city", "Since", "its", "opening", "my", "wife", "and", "I", "have", "stayed", "at", "Le", "Germain", "whenever", "we", "want", "to", "escape", "without", "escaping", "the", "city", "The", "decor", "amenities", "location", "to", "amazing", "dining", "and", "the", "hotel", "staff", "themselves", "will", "always", "bring", "us", "back"]

N = len(alphanum_words) = 44

punctuation_words = ["'getaway", "'", '.', ',', '.', ',', ',', ',', '!']

num_punctuation_words = len(punctuation_words) = 9

ratio_punctuation_to_total = num_punctuation_words / N = 9 / 44

ratio_punctuation_to_total = 0.205

C Examples of Misclassifications

C.1 Repeated Words

Real review:

“How about this had reservation confirmation in hand and still had to argue with the registration desk about price on reservation. She tried to charge me price on bill they slide under your door. Sorry not on my watch! Then about this room you gave me I understand this place is old but for the price you charge how about some indoor plumbing! You take a shower and the tub fills with water will not drain this is absolutely redickulas for this being a Hilton I think you should step back and take a look at how this place is operating. Registration: Fail Parking price: Fail Plumbing: Fail Thank you Hilton for for my super 8 experience in Chicago.”

This review notably repeated the words reservation, registration, plumbing, fail, Hilton, etc. potentially leading to the model perceiving the review as fake.

C.2 Generic Words

Real review:

“I had a very nice stay at the Wyndham Garden Calgary Airport Hotel. No stress. All services were done efficiently without sacrificing customer care. All the staff were friendly and welcoming. I’ve found the perfect place to stay whenever I have to go to Calgary on business.”

This review did not really use any unique words outside of the hotel names themselves. For that reason the model may have interpreted it as fake.

C.3 Misspelled Words

Real review:

“The location is amazing, right across from Nordstrom's on Michigan Avenue. The hotel is beautiful and the rooms are clean. But... We are Intercontinental Ambassadors (the rewards club) and to get to the Ambassador suite level, we have to ride a private "express" elevator. The elevator took three minutes to get to the thirtieth floor. When we checked in, the person behind the counter was rude and acted as if we were lucky to get checked-in. Also, there was a language barrier becuase she was Russian. The pool and spa were both beautiful, but we had to pay an extra fee because of "upkeep". If you want a nice hotel all the way around...go to the Omni.”

This review incorrectly spelled “because”, as this is a common word that should be easy to spell there is a chance other reviews with this same incorrect spelling in the training set were fake, leading to the model identifying this one as fake.

C.4 Describe all amenities

Real review:

“This is best hotel bargin in Chicago if you are not overly unhappy with a small room. Great hotel just redone by Kimpton. This is a great hotel chain that really provides a the sense of escape and fun to all of their hotels. This is a location just off of LaSalle and next to the Cadillac theater. It has a 4 star resturant which we enjoyed in a relaxed atmosphere an amazing Italian meal. The wine list is one of the best in a city of great restaurants. The Kimptons have a social hour 5 to 6 with wine at all of their hotels. You sit in a designer, kind of London circa 1968 couch with a fire place nearby and have that lost art form of relaxation with you fellow guests. They are super animal friendly and provide complete services to your dog if you wish to bring it. The exercise room is great and there is a spa here as well. There are two other Kimptons here but for the money the Allegro is the best value and location. Always consider Kimptons as they never disappoint where ever we have used them.”

This review reads like an advertisement, which at least while labelling the new dataset our group figured often these reviews seemed fake. The review reads like an advertisement because it describes all amenities. For this reason the model may have interpreted this review as fake.

C.5 Excessive Punctuation

Real review:

“While I had stayed in the Hotel Monaco before, this was my husband's first trip to Chicago and I insisted we stay here as I loved it the first time. Well the second stay was even better -- we had one of the corner suites on the river -- the service was excellent and staff was very friendly and helpful. The warm cookies in the afternoon is a nice touch after a day of sightseeing. Unfortunately we missed the wine hour. One thing that realy impressed me is that b/c of flight delays we got to Chicago a day late -- Hotel Monaco manager waived the cost of the first night on our room even though it was a guaranteed reservation.”

This review has a significant amount of punctuation, more than most normal reviews would ever have in relation to the amount of words in the review. The model may have interpreted this as fake because of the excess of punctuation.

C.6 Excessive Capitalization

Real review:

“Overall I had a VERY Bad experience when hosting a meeting for company, but when my parents came in from out of town and wanted to stay at the Swissotel, I thought we should give them another try. Turns out my first impression was correct, this is the LEAST friendly hotel I've ever been in! Even the concierge was rude and disagreeable. I could go on and on, but instead I'll just say GO SOMEWHERE ELSE!”

Any review with a lot of caps lock often will read as fake, especially during the labelling process. The model may have seen this one as fake for that reason.

C.7 Diverse Words

Fake review:

“The Ambassador East Hotel is a terrible place to stay. All of the rooms are old and dirty as though they haven't been cleaned or remodeled for years. The hotel staff is not friendly or accomodating. Do not waste your money in this crappy old hotel full of rude employees.”

This review may not have a ton of unique words, but all of the words are fairly diverse rather than repeating the same key words over and over again. For that reason the model may have seen this as a real review.

C.8 Unique Words

Fake review:

“This hotel was incredible! Haven't had a better stay in a hotel in my life. I will definitely come here again next time I am in Chicago! The rooms were so elegant, with their modern color schemes. The staff members were friendly and willing to lend a hand. The beds were surprisingly comfortable, with interesting covers. The TVs were HD, which was a step up from most hotels i have stayed at before. There was a cool martini bar that I spent an hour at one night, which was enjoyable, especially because of the live jazz band”

This review has a few unique words that stand out like incredible, elegant, modern, martini, and jazz. As these are quite unique the model may have seen this review as real, therefore mislabelling it.

C.9 Proper Grammar

Fake review:

“My wife and I were guests at the Hotel Allego in Chicago for a long weekend getaway (September 2 - 5, 2011). We had checked out the hotel's website long before booking and were quite excited about our trip. Once we arrived, the hotel staff informed us that the room we requested was unavailable and brought us to a different room that smelled quite moldy. When I asked to be moved to another room (my wife has asthma) we were politely told that we were unable to be switched at that time. The room was not particularly well-cleaned: we found previous guests hair in the bathroom sink and on the floor, and a film of dust across the television set. We are not the type to complain, so we went out to explore the city and assumed that would be the worst of it. At nighttime when we tried to sleep, we discovered the bed sagged terribly in the middle and the walls were so thin, every nearby conversation could be heard perfectly. We were tossing and turning half the night. In the morning we hoped a quick workout might help energize us for the day, but we discovered that the wellness and fitness center described on the website was really not at all what we anticipated: a few treadmills and a dilapidated weight rack. The concierge was almost never around in the lobby (her sign always said "Be right back!") and when we did ask her questions, she was actually somewhat cold and rude towards us. The second night was without incident, luckily, except for the terrible bed, but the third night we were woken up three times by hotel staff trying to enter the room to clean despite our "Do Not Disturb" sign. I would not recommend this hotel.”

Throughout this review the sentences are well structured with a proper amount of punctuation and capitalization. For that reason the model may have assumed it was a real review.

C.10 First Person Voice

Fake review:

“My stay at the Sofitel was great. I loved the location it was in. From the moment I arrived the experince was great, the attendants were helpful and the front desk was too. I thought the room was elegant and very clean as well as comfortable. The room service was excllent and they were very friendly. Checkout was a breeze and I couldnt ask for anything more in a hotel stay.”

This review used a lot of first person, which often gives the impression that a review is real. The model may have also seen this similarly and determined this review was real.feat