Classification of Fake Restaurant Reviews from NYC Yelp Dataset

(Tashrif Chowdhury (tc2367), Terrell Nowlin (trn224), Akhilesh Chandrashekar (ab10138), Zhenghao Li(zl3954), Faiz Andrea Ganz (fag277))

ABSTRACT

Customer reviews have become an important factor to consider when making key decisions in a restaurant business. The reviews also give an idea of how good or bad a given restaurant is to other potential customers. This has led to a problem in review systems where fraudulent users generate fake reviews to alter the perception of a restaurant to influence readers' opinions. Our project is devoted to identifying fake reviews from the dataset by understanding the characteristics of users and the restaurant. We will explore the user-restaurant relations to find patterns in user review behavior to achieve the best fake review detection accuracy.

INTRODUCTION

Background

As text data is the prevalent type of data in our world, computer scientists, linguists, and mathematicians have extensively researched how to represent this type of data in mathematical form so that we can use the power of our technology to make strong inferences about it. A few years ago, this journey reached a turning point with the creation of BERT-like models, which achieved state-of-the-art performance on a variety of Natural Language Processing tasks. In order to understand the evolution, trajectory, and limitations of the field, and to possibly achieve new insight, we have decided to study and experiment with some of these techniques on the Yelp dataset.

Related Work

There are a few interesting approaches to detecting fraudulent reviews, and many focus on utilizing supervised learning, but in combination with other approaches as well.[1] A linguistic approach, analyzes the language of spam reviews vs. the language of genuine reviews to find certain language patterns in the reviews.

A behavioral approach analyzes the behavior of users to detect fake reviews. The models that are being used to solve such problems are typically supervised learning algorithms such as Support Vector Machine (SVM), logistic regression, and Neural Networks. Additional data mining

[1] Shebuti Rayana and Leman Akoglu. Collective opinion spam detection:bridging review networks and metadata. In ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 985–994, 2015.

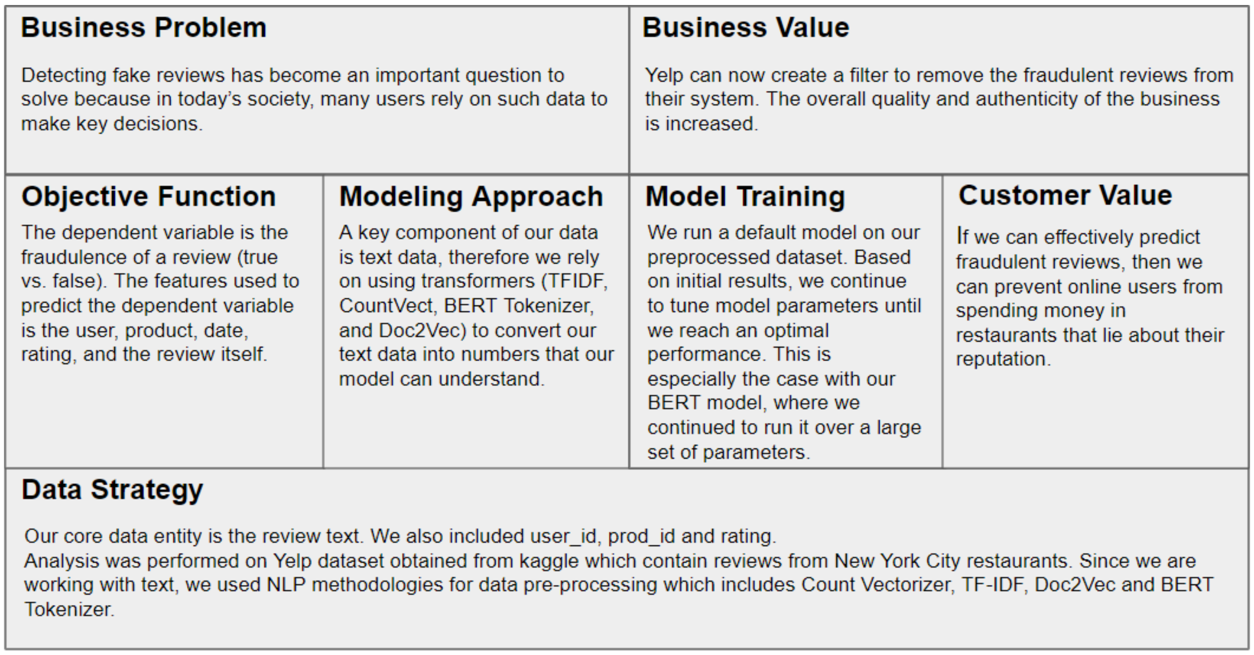
techniques such as k-fold cross validation have also been used to achieve accuracy scores of approximately 82%, an AUC score of 82.49%, and an F1 score of 81.42%.[2]

Based on our research into past projects, common strategies include utilizing word unigrams and bigrams, LIWC features and POS features.

Objectives
Our first objective was to understand how text data can be represented in mathematical form, what are the advantages and disadvantages of each representation, and what are their limitations. Similarly, our second and last objective was to see how different models performed on the best representation of the data and based on that, understand its pros and cons, and particularly see which are the cases that fail.

Methodology
An initial work of data preprocessing was done before proceeding with anything else. We then proceeded to find the best representation of the data by cross-checking the results obtained with that representation using a simple logistic regression model. Once we found the best representation we proceeded to study and implement different models and tested them on that representation or another representation if needed. To infer what were the limitations of each model we looked at where it failed.

DATA SCIENCE VALUE CANVAS



Business Problem

Detecting fake reviews has become an important question to solve because in today's society, many users rely on such data to make key decisions.

Business Value

Yelp can now create a filter to remove the fraudulent reviews from their system. The overall quality and authenticity of the business is increased.

Objective Function

The dependent variable is the fraudulence of a review (true vs. false). The features used to predict the dependent variable is the user, product, date, rating, and the review itself.

Modeling Approach

A key component of our data is text data, therefore we rely on using transformers (TFIDF, CountVect, BERT Tokenizer, and Doc2Vec) to convert our text data into numbers that our model can understand.

Model Training

We run a default model on our preprocessed dataset. Based on initial results, we continue to tune model parameters until we reach an optimal performance. This is especially the case with our BERT model, where we continued to run it over a large set of parameters.

Customer Value

If we can effectively predict fraudulent reviews, then we can prevent online users from spending money in restaurants that lie about their reputation.

Data Strategy

Our core data entity is the review text. We also included user_id, prod_id and rating.
Analysis was performed on Yelp dataset obtained from kaggle which contain reviews from New York City restaurants. Since we are working with text, we used NLP methodologies for data pre-processing which includes Count Vectorizer, TF-IDF, Doc2Vec and BERT Tokenizer.

[2] Wang, Z., Zhang, Y., & Qian, T. Fake Review Detection on Yelp. Stanford University. http://cs229.stanford.edu/proj2017/final-reports/5229663.pdf.

DATA EXPLORATION

Feature Name	Feature Description	Feature Information
1. user_id	Individual Yelp reviewer	Continuous whole numbers
2. prod_id	Individual Yelp restaurant	Continuous whole numbers
3. date	Date review was written	DateTime
4. review	Original yelp review	Text data
5. rating	Restaurant rating based on reviewer	Continuous float number
6. label	Review fraudulent or not	Binary (0, 1)

WORD CLOUDS

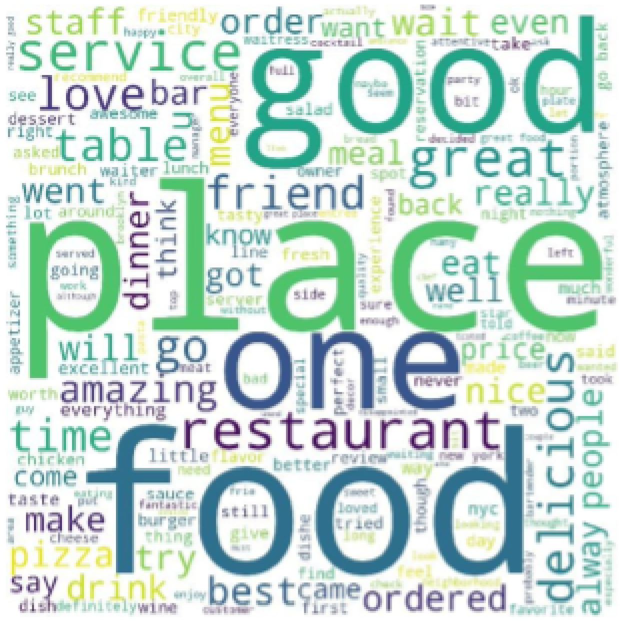


Fig 1. Word Cloud for Fraud Reviews

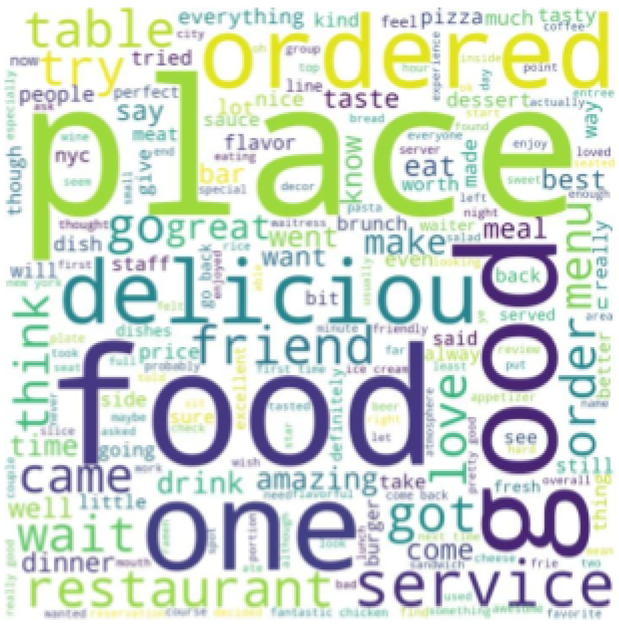


Fig 2. Word Cloud for Real Reviews

Above are the word clouds for fraud reviews (36,860 samples) and real reviews (322,097 samples) from our dataset. By reading these word clouds, we can have a more obvious view of the difference and similarities between fraud and real reviews. Both fraud reviews and real reviews share the same most frequent words: “place”, “food”, “good”, and “one”. However, the words “ordered” and “delicious” appear more frequently in real reviews than in fraud reviews.

MODEL SELECTION AND TESTING

Logistic Regression:

- **Model:**
    - Logistic Regression is a supervised machine learning algorithm that essentially uses a logistic function to model a binary dependent variable. Logistic regression is a simple and more efficient method for binary and linear classification problems. It is a classification model, which is very easy to realize and achieves very good performance with linearly separable classes.

- **Preprocessing:**
    - For logistic regression, we implemented multiple transformers such as TFIDF, CountVectorizer, and Doc2Vec, to encode our text data. In addition, we oversampled randomly to create a balanced training dataset to input into our model.
        - Size of Training Data: 431,610 (True/Fraudulent Reviews: 215,805 each)

- **Parameter Tuning: max_Iterations = 1000**
    - Logistic Regression: We only declare the max_iterations parameter to allow our model more time and data to converge. In addition, we also attempted to tune other hyper-parameters using GridSearchCV and RandomGridSearchCV, but the model did not perform as well using such parameter values.
    - TFIDF and CountVectorizer: Stop words set to english, we created unigrams and bigrams, and we set binary to True to model binary events rather than integer counts.
    - Doc2Vec: We set our number of epochs to be 30

Naïve Bayes:

- **Model:**
    - Naïve Bayes is a supervised machine learning algorithm that classifies the dataset using the assumption that the attributes in the dataset are independent of one another. It simplifies the Bayes classifier by narrowing down the specific probabilities for each of the X instances. Naïve Bayes is cheap and handles many attributes easily, and it is not sensitive to irrelevant features.

- **Preprocessing:**
    - For Naive Bayes, we tried multiple transformers such as TFIDF, CountVectorizer, and Doc2Vec, to encode our text data. In addition, we oversampled to create a balanced training dataset to input into our model.
        - Size of Training Data: 431,610 (True/Fraudulent Reviews: 215,805 each)

- **Parameter Tuning:**
    - Naive Bayes: We use the default parameters to make our predictions. We attempted to use GridSearchCV and RandomizedGridSearch in an attempt to tune hyper-parameters, however, the results were not as good compared to the default values.

    - TFIDF and CountVectorizer: Stop words set to english, we created unigrams and bigrams, and we set binary to True to model binary events rather than integer counts.
    - Doc2Vec: We set our number of epochs to be 30.

Random Forest:

- **Model:**
    - Random Forest is an ensemble classifier built using Decision Trees. It is built using the training data and outputting either the mode or the mean prediction. The mode of the classes is used for classification, while the mean prediction is used for regression.

- **Preprocessing:**
    - For Random Forest, we tried two different transformers, TFIDF and CountVectorizer, to encode our text data. In addition, instead of oversampling, we chose to undersample our training data, because the model would take too long to run on a larger dataset.
        - Size of Training Data: 49,392 (True/Fraudulent Reviews: 24,696)

- **Parameter Tuning:**
    - Random Forest: We also use default parameters to make our predictions. We also attempted to use GridSearchCV and RandomizedGridSearch CV in an attempt to tune our hyper-parameters, however, it again did not perform well. This could be due to having a selection of parameters that were not necessarily ideal for our dataset and is something we could look to adjust for future work.

Bidirectional Encoder Representations from Transformers (BERT):

- **Model:**
    - For our BERT model, we selected Hugging Face's BERT For Sequence Classification model. BERT is a transformer model and is the state-of-the-art model for Natural Language Processing tasks.

- **Preprocessing:**
    - The training and validation sets were sampled from a 'balanced portion of the dataset, while our three test sets were samples from another portion of the original dataset which was not balanced and had the original distribution: roughly 9 true reviews to 1 fraudulent.

    - We included the non-text features such as user_id, product_id, date, and rating as part of the reviews' text so that they would be fed into BERT, since it does not accept anything else.

    - To extract our encoded data, we used the endcode_plus method from Hugging's Face BERT Tokenizer to get our input_ids and attention_mask for training. For the encoding we used 512 as the maximum number of tokens. This is the maximum number of tokens the model allows to use. We chose to use the maximum because a good portion of the samples are even longer than 512 words. In the event that the review is longer than 512 words, the review is going to be truncated. If it is shorter, padding is going to be added.

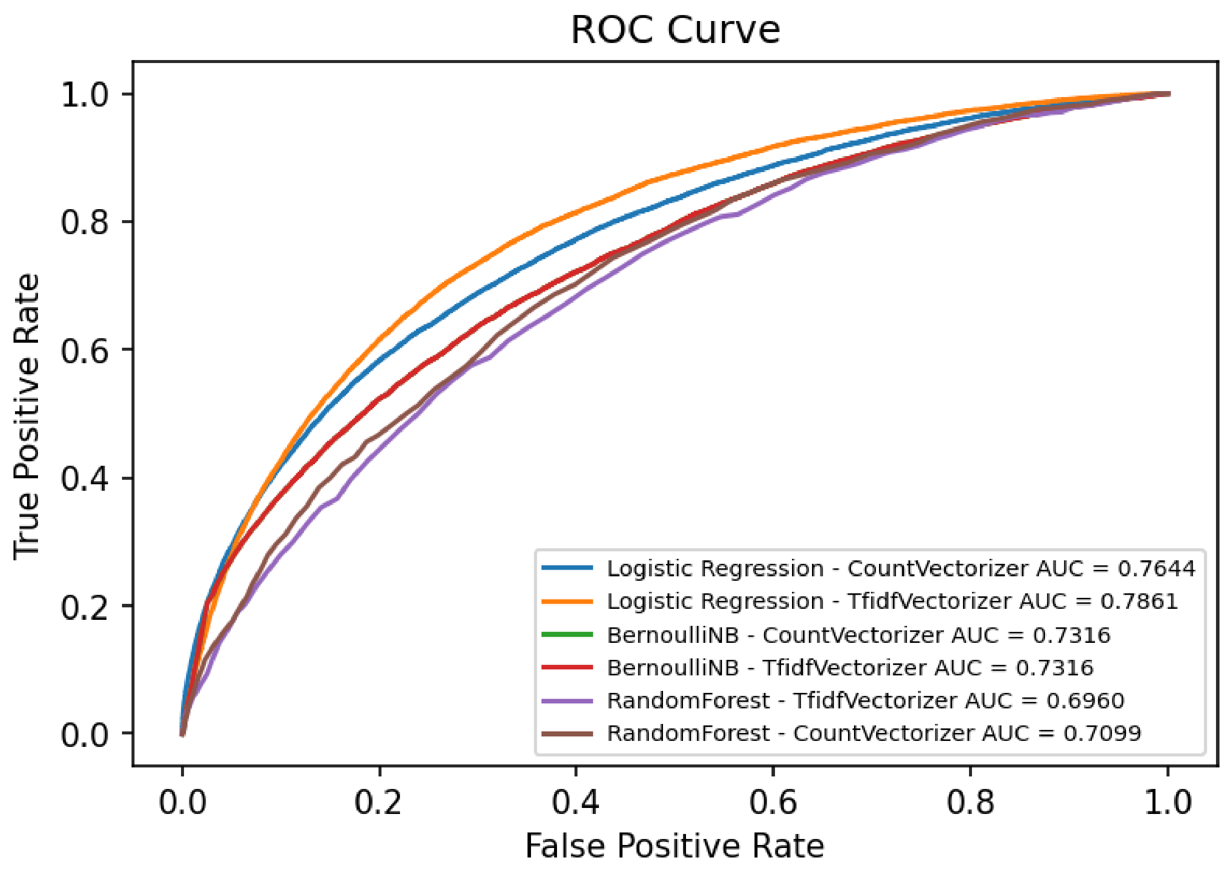
We reference the help of the following link for this part: https://medium.com/analytics-vidhya/a-gentle-introduction-to-implementing-bert-using-hugging-face-35eb480cff3

- **Parameter Tuning: Epochs, Optimizer, Scheduler, and Model Parameters**¶
    - After several testing observations, we chose a learning rate of 2e-5 and a number of warm-up steps proportional to the number of batches in our training dataset. The number of warm-up steps chooses how quickly the learning rate converges towards the specified value. Hence, the optimizer starts slowly and gains "confidence" over time. We have observed that for 10000 samples, 5000 warm-up steps are sufficient for preventing overfitting too soon and have extensively tested for different training set sizes. For our last run of the model we trained on 8 epochs, ~53k samples, with a warm-up step of 18000 and Adam is our optimizer of choice. The validation set is about 6000 points and for our three test sets we chose 5000 points.

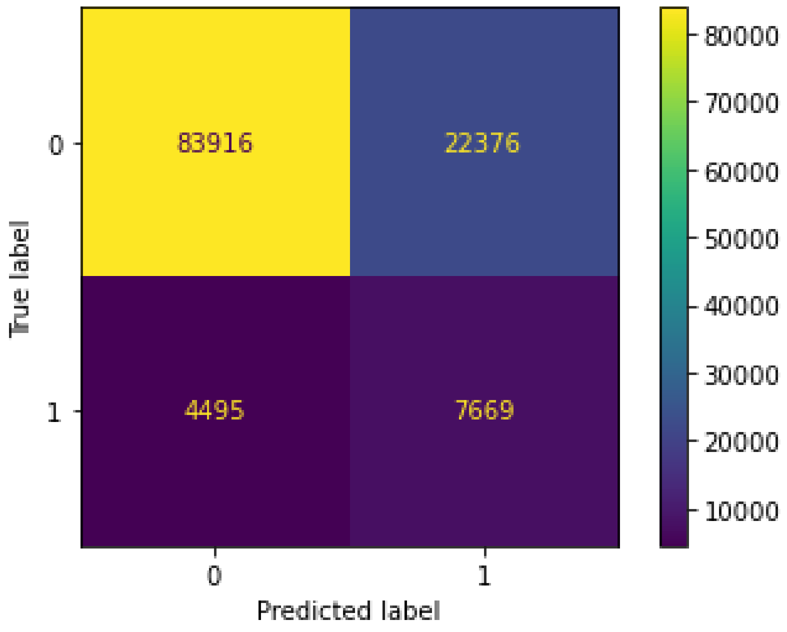
    - We have modified the model to have two-hundred neurons in the output layer and pass the output of each neuron through and hyperbolic tangent activation, and added on top of this a two-hundred dense layer with a dropout of 0.5, another 200 dense layer and a final output layer of two neurons corresponding to the two classes in our dataset. For this we took inspiration from https://www.kaggle.com/barelydedicated/yelp-review-predictions-using-huggingface-bert

RESULTS

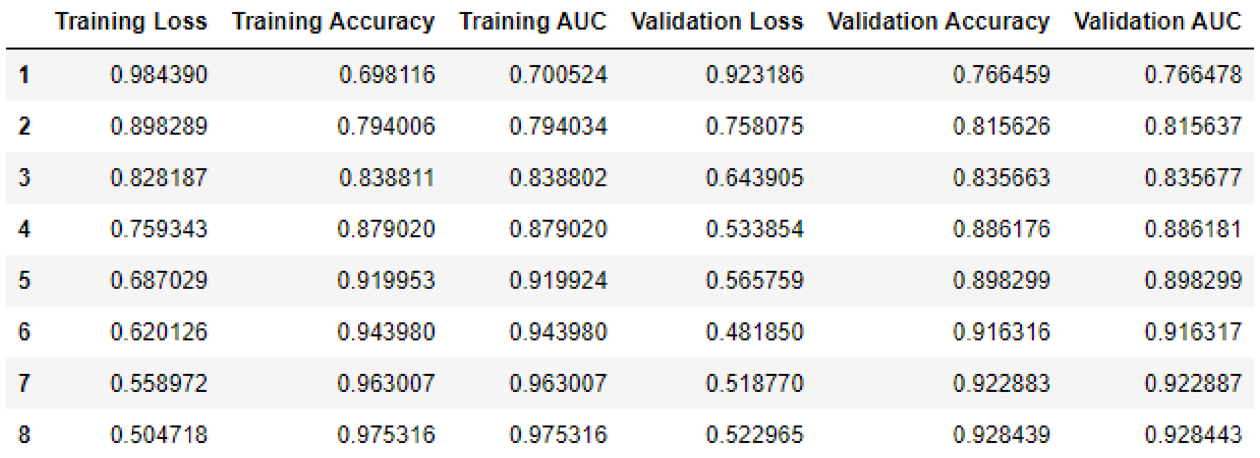
Models	Transformer	Accuracy	AUC	F1-Score
LogisticRegression	TFIDF	77.31%	78.61%	81.07%
	CountVectorizer	88.75%	76.43%	87.59%
	Doc2Vec	63.69%	58.93%	70.78%
Bernoulli Naive Bayes	TFIDF	61.08%	73.16%	68.69%
	CountVectorizer	61.08%	73.16%	68.69%
	Doc2Vec	72.46%	56.05%	76.92%
Random Forest	TFIDF	55.43%	69.60%	63.85%
	CountVectorizer	59.62%	70.99%	67.51%
BERT	BERT Tokenizer	86.1%	66%	86.67%



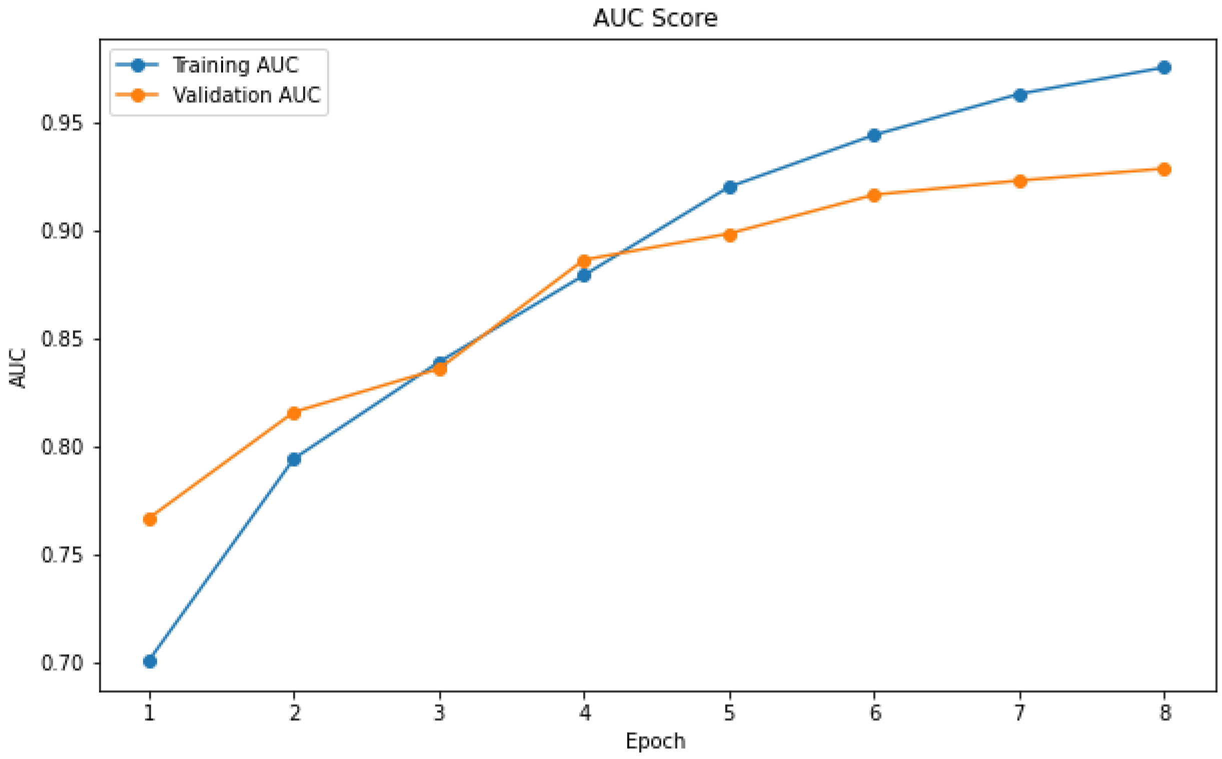
Logistic Regression + TFIDF Vectorizer Confusion Matrix

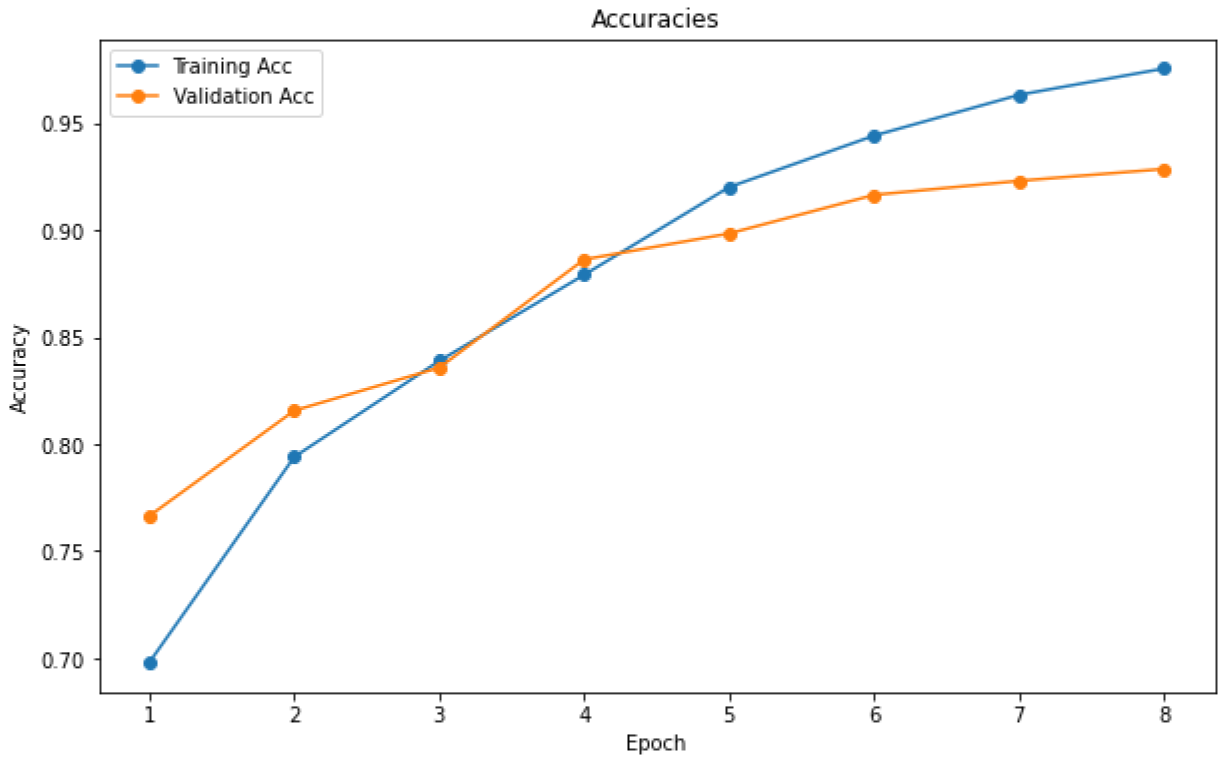


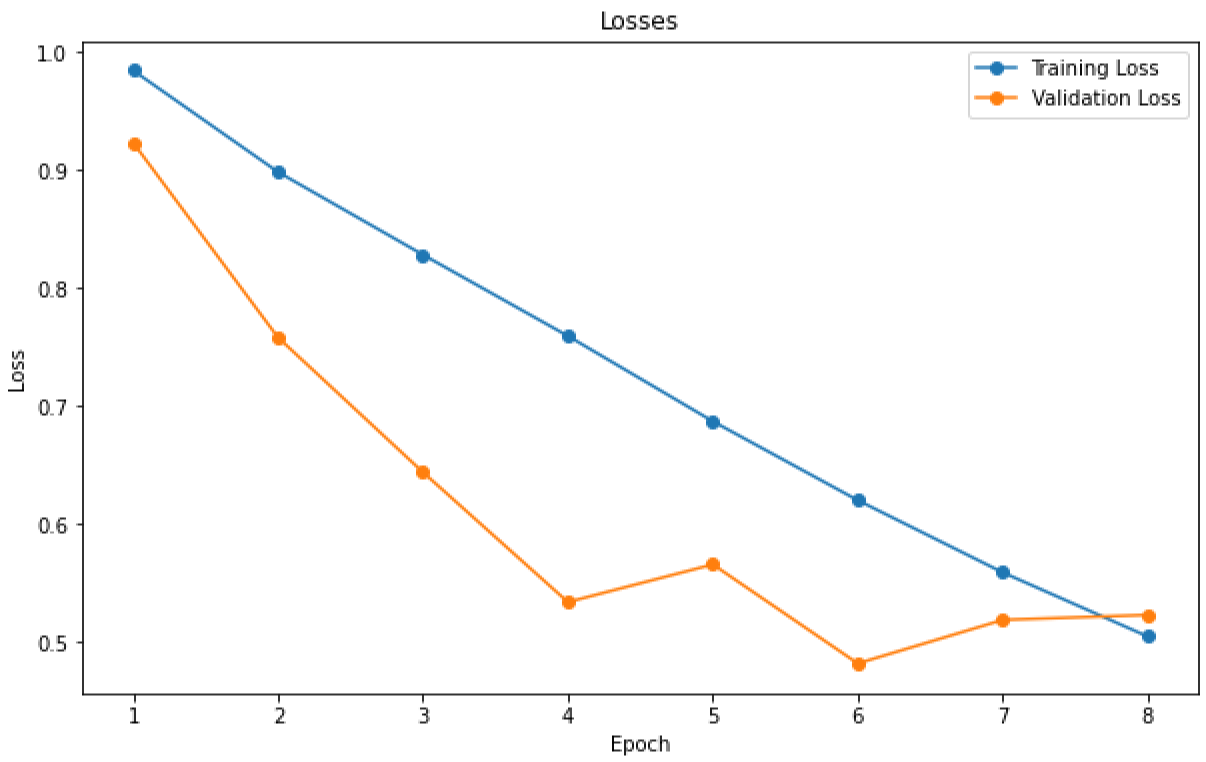
BERT Results on Training and Validation Dataset



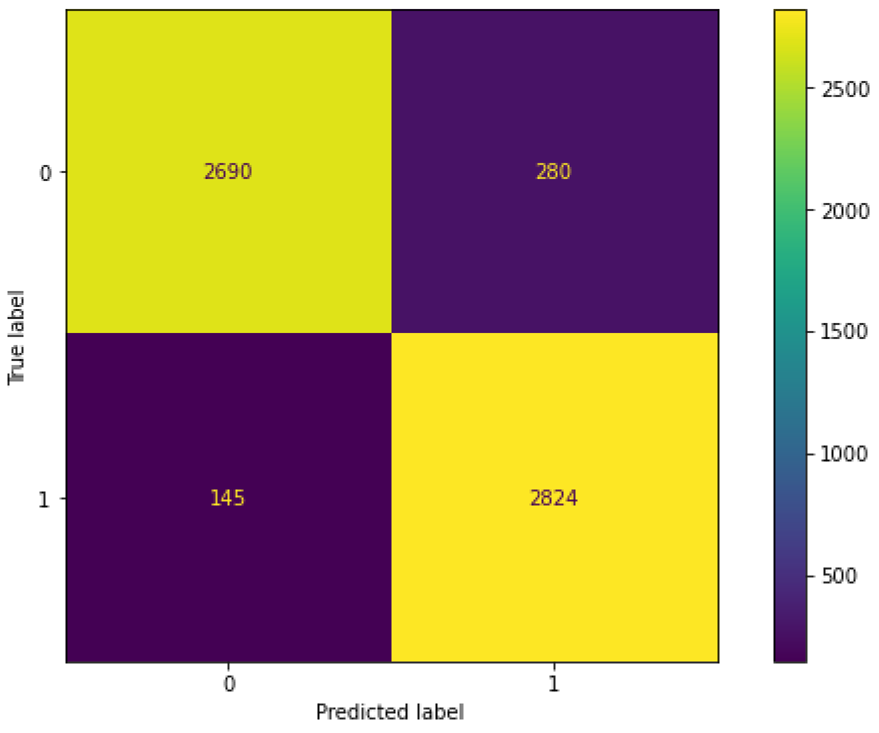
	Training Loss	Training Accuracy	Training AUC	Validation Loss	Validation Accuracy	Validation AUC
1	0.984390	0.698116	0.700524	0.923186	0.766459	0.766478
2	0.898289	0.794006	0.794034	0.758075	0.815626	0.815637
3	0.828187	0.838811	0.838802	0.643905	0.835663	0.835677
4	0.759343	0.879020	0.879020	0.533854	0.886176	0.886181
5	0.687029	0.919953	0.919924	0.565759	0.898299	0.898299
6	0.620126	0.943980	0.943980	0.481850	0.916316	0.916317
7	0.558972	0.963007	0.963007	0.518770	0.922883	0.922887
8	0.504718	0.975316	0.975316	0.522965	0.928439	0.928443


AUC Score
Training AUC
Validation AUC
AUC
0.70
0.75
0.80
0.85
0.90
0.95
1
2
3
4
5
6
7
8
Epoch

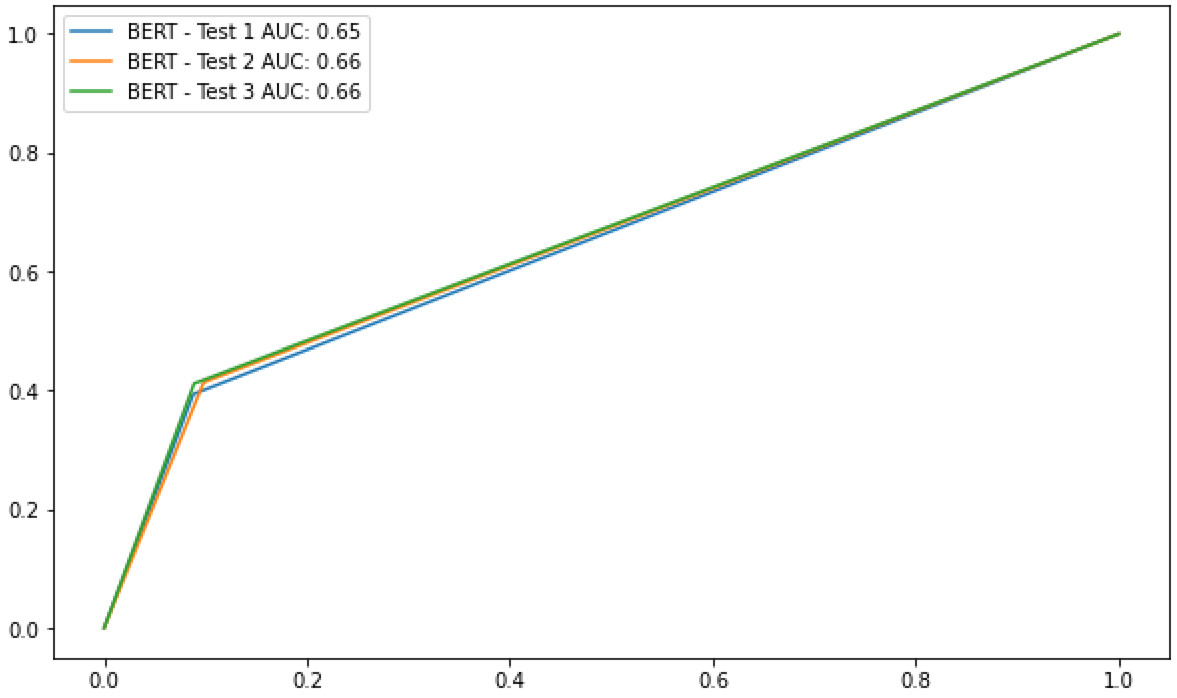

Accuracies
Training Acc
Validation Acc
Accuracy
0.70
0.75
0.80
0.85
0.90
0.95
1
2
3
4
5
6
7
8
Epoch

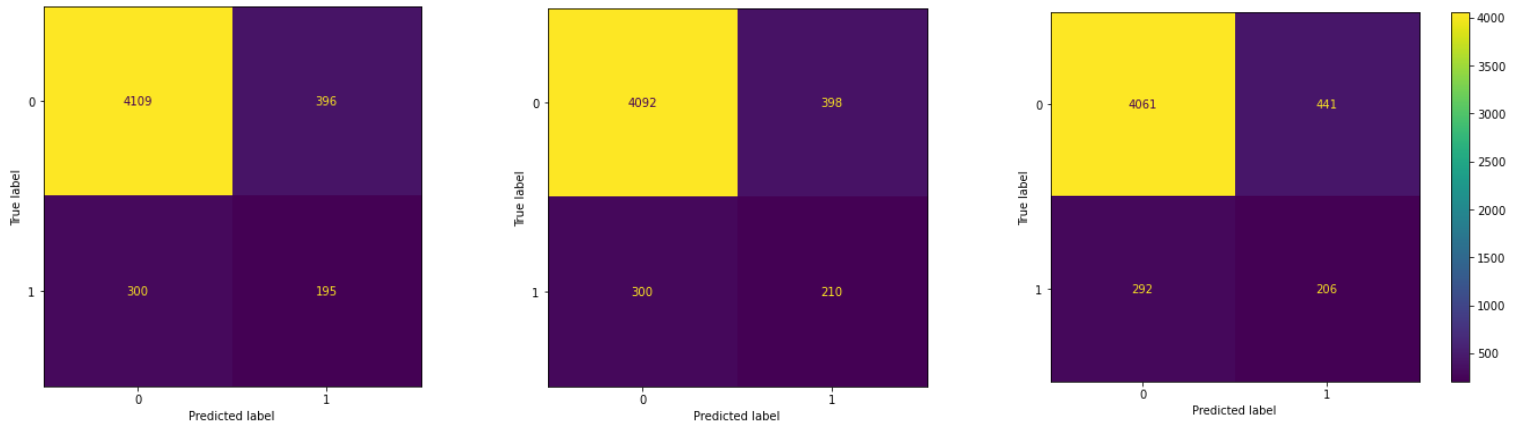


BERT Confusion Matrix on Validation Data



BERT Results on 3-Split Test Dataset





CONCLUSION

Logistic Regression, Naive Bayes, & Random Forest
When it comes to the supervised machine learning algorithms we implemented for our project, our results were not completely terrible. Logistic regression and Naive Bayes seemed to perform most consistently given the various transformations used. Random forest, however, did not perform up to our expectations. Additionally, it took a large amount of time to run even on an undersampled dataset. As opposed to Logistic regression and Naive Bayes which performed very efficiently on the oversampled dataset.

We can see that Logistic Regression, with a TFIDF transformer, works very well on our dataset. This is incredibly ironic because Logistic Regression and TFIDF are such commonly implemented methods in NLP, and we were not expecting them to perform this well when we started our project. One hypothesis as to why this combination performed so well is that we have more data points to train our model on. Our model might have performed so well because of the fact that we oversampled, which led to over 400,000 training points.

This leads us to wonder how well BERT could have performed if we had the means to run it on a training set of 400,000+ data points. This is just one example of how we can expand upon our project results and work to achieve better results in the future.

BERT (Bidirectional Encoder Representations from Transformers):
We can see that the model performs very well, displaying the power of transformer models in the field of Natural Language Processing, despite overfitting slightly by the end of the run-in epoch 7 and 8. The problem is really hard and the real-world data is very skewed. Not many reviews are fraudulent and when the model sees many samples of which only about 10 percent are of one class, it is gonna have a hard time predicting accurately for that specific class. This will obviously affect its AUC score. But when seeing a balanced dataset or a dataset of elements from only one label it is very confident in classifying them, easily reaching accuracies higher than 90 percent.

The problem is indeed challenging, even for us humans it is very hard to distinguish between false and true reviews. However, we believe that if we can train on a larger, "balanced", and more appropriately processed dataset the result would be excellent, even on imbalanced testing sets. This brings us to our next steps.

Because many samples have very high word count and BERT can only take a maximum of 512 word, we could use another BERT model to summarize the reviews that are larger than, let's say, 450 word, to allow also space for the other features to be added to the text as we did initially.

This would be a sort of dimensionality reduction on the data and would allow us to retain information from part of reviews which would be otherwise truncated. We were able to do this in another notebook, but unfortunately did not have the time to test it on the model. (explained further below)

Another thing that can be done is using a BERT model for Text Generation to both study how the model thinks of fraudulent review and generate more fraudulent reviews to balance our dataset, giving us more training samples and allowing us to train our model for even better performance. If these steps are done correctly, we believe the model could reach more than 90% accuracy even on a very imbalanced dataset and be an overall high-performing model. However, training the model with an oversampled dataset could mean training on over 700k parameters and that would require an extensive amount of time and high computational power to do.

This example shows the power of BERT and its derivatives, and why they have achieved many of the testing benchmarks set by the community in the field of Natural Language Processing, and how we can benefit from them in real-world problems such as the one we tackled.

Future Work: Text Summarization
We tried to use text summarization on the reviews that have more than 450 words to reduce the complexity of our model. We tried both extractive summarization and abstract summarization. Among LexRank, Latent Semantic Analysis, Luhn, T5 Tokenizer, Bert Summarizer, and GPT-2 Transformers we found that Bert Summarization has a good balance between sentence length and meaningfulness. However, eventually, we did not use summarized data to train our models since we did not tune the summarizer. Tuning the summarizer and using summarized data to train the model became one of our future words.

Future Work: Balancing Dataset with BERT for Text Generation
To further improve our models, we could try and implement a transformer for text generation and try to generate fraudulent reviews. This would aid in studying latent variables that are predictive of fraudulent reviews and to balance our dataset, thus giving us more training samples to train on.

Overall, our best performing models, Logistic Regression (w/ TF-IDF) and BERT, perform pretty well given the task at hand. When we performed our data exploration, and created our word clouds, we saw that our true reviews and fraudulent reviews shared similar words with a high level of frequency as well. From this, it is clear that our true reviews and fake reviews are generally similar which potentially makes it much more difficult for our models to truly distinguish between the two.