Detection Of Fake News Using Machine Learning

Hamza Luqman
Muhammad Daud Sheikh
Okeoghenemarho Obuareghe

Summary of Contributions

Data Collection:

All the data was collected by all the members. The labelling of the data was also done individually by each member on the 1000 additional data.

Coding:

The sections in our notebook that each person worked on is listed below

Hamza Luqman: Importing Dataset, Pre-processing, Original Data ML (all Unigram with TF, all Bigram with TF), Extended Data ML (all Unigram with TF-IDF, all Bigram with TF-IDF)

Muhammad Daud Sheikh: Pre-processing, Original Data ML (all Unigram with TF-IDF, all Bigram with TF-IDF), Extended Data ML (all Unigram with TF, all Bigram with TF)

Okeoghenemarho Obuareghe: Selecting the Best model and writing code to generate the misclassified datapoints, and all code to investigate the reasons for the misclassification.

Report:

Hamza Luqman: Abstract, Introduction (a, b, c.i, c.ii, c.iii, c.iv), Results (Section 3.c and 3.d)

Muhammad Daud Sheikh: Results (Section 3.c, 3.e, and 3.f), Discussion, Conclusion

Okeoghenemarho Obuareghe : Results (Section 3.a, 3.b), Discussion

Link to Databricks Notebook: https://databricks-prod-cloudfront.cloud.databricks.com/public/4027ec902e239c93eaaa8714f173bcfc/2185213181139538/433056252362217/772439939466424/latest.html

Contents

Abstract

Context

Access to technology and social media has revolutionized the generation and distribution of information. However, this increase in the flow of information does not necessarily translate to increased level of truthful public knowledge. Fake News has been gaining a lot of traction in recent years. Spreading fake news can lead to making uninformed decisions on political issues, health care and even create social conflict.

Objective

In this project, we focus on the hyper parameter tuning on the top three best performing models from the research paper Detection of Online Fake News Using N-Gram Analysis and Machine Learning Techniques [1] and determining the performance of best model along with its hyper parameters on classifying news source as either Fake or Real. We want to minimize the misclassification of an actual “Fake” news as “Real” which Is why recall was considered as one of the more important model evaluation metrics.

Method

The original data was collected from Kaggle and Reuters. To extend the dataset, web scrapping was carried out to look for News related to 2016 US elections. Two main sources for this were Politifact & Reuters. Once the original and the extended dataset was loaded into Databricks, the data was pre-processed and split into training and test set to carry out our classification analysis using varying hyper parameters for Logistic Regression, LSVM and Decision Trees.

Results

It was found that model with the best accuracy was logistic regression with a score of 70% with 10,000 features, regularization parameter of 0.1, and the features being bigram. We found that when it comes to recall, Decision Trees performed the best out of the three models. The performance of the models was better with Bi-grams compared to Unigrams.

Conclusion

Logistic regression with many bigram features results in the best accuracy score for fake news detection after pre-processing and noise removal is performed (Accuracy score of 70%). Recall is chosen to be the most important metric when evaluating the which machine algorithm to choose for the use case of identifying as many fake news articles as possible.

Introduction

Access to technology and social media has revolutionized the generation and distribution of information. Social media platforms provide easy access for users to share information throughout the globe. However, this increase in the flow of information does not necessarily translate to increased level of truthful public knowledge.

Fake News has been gaining a lot of traction in recent years. NewsGuard reported that In the US, 8% of engagements with 100 top performing news sources were unreliable in 2019. This number jumped to 17% in 2020. Spreading fake news can lead to people making uninformed decisions on political issues, health care and even create social conflict. Social media platforms such as Facebook and Twitter can be used to spread misinformation to sway election results and create confusion about the basic facts of current affairs and events.

Research in the field of fake news detections is currently an emerging field and there isn't an abundance of work done in this area. With further work in developing more accurate fake news detection models, Machine Learning can be used to help reduce the spread of misinformation online and help people in making more informed and fact-based decisions.

In this project we will focus on using an n-gram feature-based text analysis to detect fake news. The top three best performing classification models from the research paper were selected, namely Logistic Regression, LSVM and Decision Tree. The experimental procedure involved creating a few pipelines for various feature combinations. The combinations are as follows:

- Unigram with TF
- Unigram with TF-IDF
- Bi gram with TF
- Bigram with TF-IDF

Each of these combinations were applied to the original dataset as well as the extended dataset. For each of the feature combinations, grid search was conducted to find the best hyper parameters for each of the ML models mentioned earlier. The hyper parameters tested for each model are:

- Logistic Regression
    - Vocabulary size, [1000, 5000, 10000]
    - Regularization parameter, [0.1, 0.01]
- Linear Support Vector Machine
    - Vocabulary size, [1000,5000, 10000]
    - Regularization parameter, [0.1, 0.01]
- Decision Tree
    - Vocabulary size, [1000, 5000, 10000]
    - Max depth, [2, 5]

The original data consists of 1000 “Real” and 1000 “Fake” news articles. The new data was collected using web scrapping for news related to 2016 US elections. Two main sources were Reuters and Politifact. The Title, Text, and Date information for each of the news article was extracted and the Subject labelled as news. Once the additional data was collected, it was combined with the original dataset, for a total count of 3000 datapoints. This newly added data was then labelled by each group member by finding the news online and verifying if the source is legitimate or not and whether the text contains profanity that is not typical in articles from legitimate news sources.

To carry out the pre-processing of the original and the original + extended dataset, the following steps were taken:

- Lower casing
- Regex filtering
- Tokenization
- Stop word removal
- Noise removal
- Stemming

In the original dataset, the best performing model in terms of recall was the Decision Tree. The recall score was 0.93 for bigrams with TF as well as Bigrams with TF-IDF. For the original + extended dataset the best performing model in terms of recall was again the Decision Tree. The recall score was 0.97 for bigrams with TF as well as bigrams with TF-IDF. When it comes to the recall score, Logistic Regression performed the worst, followed by LSVM and finally Decision Tree being the top performer. This was observed for both, the original and the original plus extended dataset.

The performance of the models’ changes based on the type and value of hyper-parameter chosen. The three model that were chosen were logistic regression, LSVM, and Decision Tree. Logistic regression and LSVM had regularization parameter as the hyper-parameter, and it were varied between the values of 0.01 and 0.1. It was found that the more general the model (higher regularization) the better than performance of the model in terms of accuracy. Decision Trees maximum tree depth was chosen as the hyper-parameter, and it was varied between the two values of 2 and 5. It was found that 5 resulted in better results which suggests improving recall, increased model complexity is desired as increasing the tree depth increased model complexity.

Results

Data Collection & Labeling

The original data consists of 1000 “Real” and 1000 “Fake” news articles. New data was collected using a web-scrapping script for 2016 U.S. Presidential Election. The main sources were obtained from two websites Reuters.com and Politifact.com. Information on the Title, Text, and Date for each of the news article was extracted.

The newly added data was then labelled “Real” or “Fake” by each member of the team by searching the news online and verifying if the source is legitimate or not. If a consensus could not be reached, the text of the article is searched for vulgarity, profanity, or bias that is not typical of articles from credible news sources.

Data labelling was done individually by each member of the team, and Cohen’s Kappa metric used to calculate the level of agreement between labelers.

Cohen’s Kappa is a statistical metric used to measure inter-rater and intra-rater reliability for qualitative (categorical) items, thus measuring the level of agreement between two Labelers or judges. It is generally thought to be a more robust measure than simple percent agreement calculation, as it considers the possibility of the agreement occurring by chance.[3]

Cohen’s Kappa ck, is calculated as:

$ck = (p_o - p_e) / (1 - p_e)$

Where:

p_o: Relative observed agreement among Labelers

p_o = (Both said Fake + Both said Real) / (Total Labels)

p_e: Hypothetical probability of chance agreement

p_e = Probability of Chance of Real (p_{real}) + Probability of Chance of Fake (p_{fake})

P_{real} = (Total times Labeler 1 said “Real”/ Total Labels) x (Total times Labeler 2 said “Real”/ Total Labels)

p_{fake} = (Total times Labeler 1 said “Fake”/ Total Labels) x (Total times Labeler 2 said “Fake”/ Total Labels)

Rather than just calculating the percentage of items that the Labelers agree on, Cohen’s Kappa considers the fact that the Labelers may happen to agree on some items purely by chance.[4] Cohen’s Kappa ranges between 0 and 1, with 0 indicating no agreement between the two Labelers and 1 indicating perfect agreement between the two Labelers.

Table 1 – Cohen Kappa & Percent Agreement Metrics

		Daud	
		Real	Fake
Hamza	Real	421	3
	Fake	0	576

p_o	0.997
p_{real}	0.179
p_{fake}	0.334
p_e	0.512
ck	0.994
% Agreement	0.997

		Oke	
		Real	Fake
Hamza	Real	416	4
	Fake	7	573

p_o	0.989
p_{real}	0.178
p_{fake}	0.335
p_e	0.512
ck	0.977
% Agreement	0.989

		Daud	
		Real	Fake
Oke	Real	420	7
	Fake	4	569

p_o	0.989
p_{real}	0.181
p_{fake}	0.33
p_e	0.511
ck	0.978
% Agreement	0.989

The Cohen's Kappa values show near perfect agreement between the labels of each team member with Daud and Hamza disagreeing on 0.003% of the labels, Oke and Hamza 0.011%, and Daud and Oke 0.011%. However, there were few points where the labeller's disagreed on some labels. For Instance, a news article was labeled as real by a labeler due to the headlines "**Trump Dominates South Carolina, Dooming The Republican Party"** and mention of South Carolina in the body of the article.

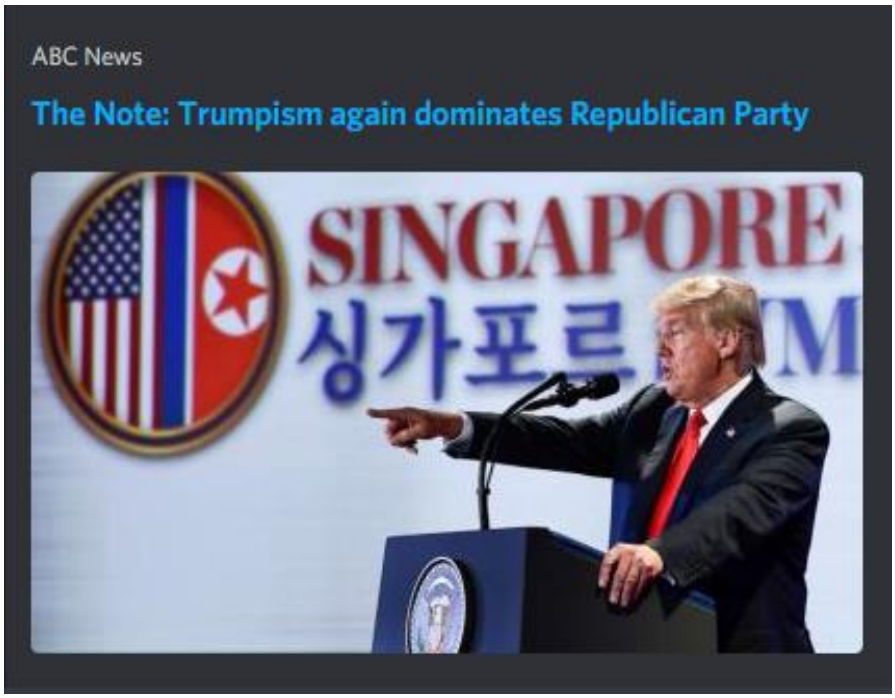

Figure 1 - Disagreed Upon Article

The president sought to make his dominance known in South Carolina by attacking Rep. Mark Sanford from aboard Air Force One even as he headed home from Singapore. Sanford went down to defeat Tuesday – a Freedom Caucus member, but insufficiently Trump-ish, it would seem.

Rick Klein and MaryAlice Parks (2018). The Note: Trumpism again dominates Republican Party. Retrieved June, 13, 2018, from: https://abcnews.go.com/Politics/note-trumpism-dominates-republican-party/story?id=55849587.

However, on careful examination of the news content, it was discovered that the article contained racial overtones, clearly depicting the article as fake. An excerpt of the article is shown below:

"No amount of Bible-thumping, gun-toting or terrorist fear-mongering could draw the crowds away. They wanted one thing and one thing only: Someone to tell them that not only was their hate of Those People permissible, it was proper and necessary to make America great again."

It was then decided by the team that though the headline seemed to depict the news as "Real" the article contained unfactual racial slurs, and so the team decided to label it "Fake" news.

Original Vs. Extended Dataset Comparison

Approach

In Machine Learning, data satisfying Normal Distribution is beneficial for model building. It makes math easier. Several models like are explicitly calculated from the assumption that the distribution is a bivariate or multivariate normal. Therefore, it was important to make sure that we have as much evenly distributed class of data in our datasets as possible.

Results

The Real and Fake News in the original data was sourced from Kaggle.com, having an equal distribution of 1000 “Real” and 1000 “Fake” news articles. Our web-scrapping script retrieved several news articles, and we randomly selected 1000 news.

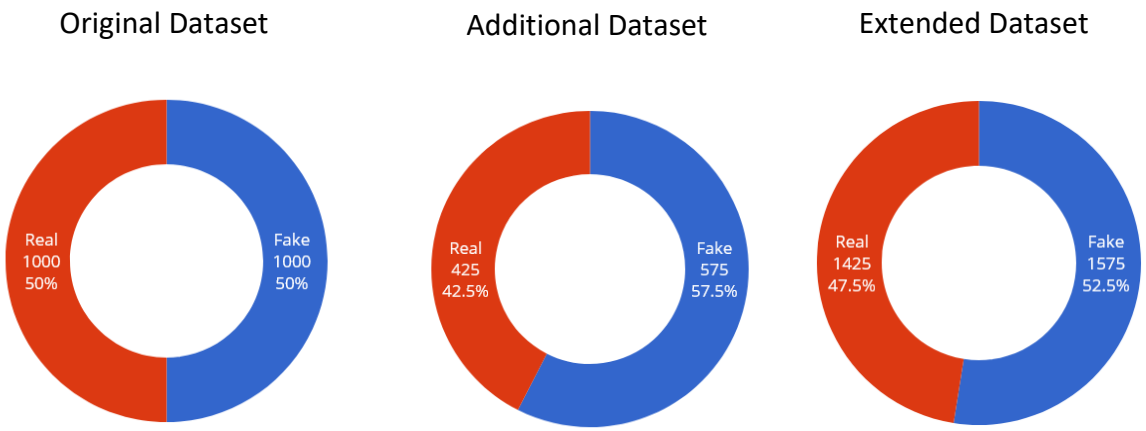

Figure 2 - Dataset Labeling Distribution

After manual labelling was done, the 1000 additional data consisted of 425 “Real” and 575 “Fake” news, and was combined with the original dataset, for a total count of 3000 datapoints.

The Data distribution chart in *Figure 2* shows the Extended dataset consists of a total 1425 Real News, 42.5% more than the real news original dataset, and 1575 Fake News, 57.5% more than the number of fake news in the original dataset.

In Total our Extended dataset showed an almost evenly split distribution between Real and Fake news. This is a suitable distribution of data for our Machine learning models to work with.

Data Pre-Processing

Approach

To carry out the pre-processing of the original and the original + extended dataset, the following steps were taken:

- Lower casing
- Regex filtering
- Tokenization
- Stop word removal
- Noise removal
- Stemming

The initial step was to lower case the “Text” column and only carry forward the “Text” and “Category” column forward in our analysis. This was followed by using regular expressions to filter the “Text” column for alphanumeric words and filter out any hyphens. Once this was done, we utilized nltk libraries sentence and word tokenizer to carry out sentence segmentation and then word segmentation of the “Text” column. Since stop words don’t really add any meaningful distinguishing between different document, these were removed by creatin a list of stopwords found in nltk.corpus stopwords and then filtering out our “Text” column data to remove any instance of those words from our dataframe. Once stopwords are removed, we further pre-process the data by defining what we consider to be noise and filter out “Text” column to discard words whose match is found in our list of noise words. The last step of our pre-processing is to carry out stemming, which will convert all the words to its base root form. This helps us in reducing the textual dimensionality. Word stemming is carried out using the PorterStemmer.

Results

Table 2 – Before Pre-Processing – Original Dataset

Word	Count
the	52101
to	30887
of	25039
a	23313
and	22444

Table 3 **– Before Pre-Processing – Extended Dataset**

Word	Count
the	84502
to	46433
of	37697
a	36082
and	34549

Table 4 – After Pre-Processing – Original Dataset

Word	Count
the	57338
to	31280
of	25257
a	24474
and	23406

Table 5 – After Pre-Processing – Extended Dataset

Word	Count
Trump	17591
Said	10770
Clinton	7441
election	5680
would	5392

Model Performance Comparison

Approach

For our classification we have two possible values, “Fake” or “Real”, where “Fake” is true positive and “Real” is true negative. The confusion matrix layout is shown below where we have:

- TN: True Negative
- TP: True Positive
- FN: False Negative
- FP: False Positive

***Table 6* – Confusion Matrix Example**

Actual Values	Negative	TN	FP
	Positive	FN	TP
		Neagtive	Positive
		Predicted Values	

The three-evaluation metrics reported in our results along with the confusion matrix are:

Precision:

Of all the samples classified as positive, what fraction was actually positive in the dataset. In our case this tells us the ratio of how many samples predicted as “Fake” were also labelled as “Fake” in the dataset.

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

Recall:

Of all the samples that were positive in the dataset, what fraction were misclassified as negative. In our case, this tells the ratio of how many "Fake" samples were misclassified as "Real". This metric has been given the most importance by our group. We wanted to minimize the misclassification of "Fake" news as "Real" because we believe this to have the potential to cause serious harm on human health. For instance, if a certain "Fake" news articles that talks about vaccines states that covid vaccines are not real or that they reduce a person's lifespan and our model labels this as "Real", the people that will believe this to be true will make decisions regarding their health that will not be in their best interest.

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

F1-score:

$$f1 = 2 * \frac{precision * recall}{precision + recall}$$

Results

The top 10,000 features were selected, with the best model hyper-parameters for each of the model. More on hyper-parameters in the next subsection. The entire data was split into an 80/20 train test split for the evaluation on the original and the original + extended dataset. The sample split count was as follows:

***Table 7* – Train-Test split of original and extended dataset**

	Training	Test
Original	1600	400
Original + Extended	2394	606

In the original dataset, the best performing model in terms of recall was the Decision Tree. The recall score was 0.93 for Bi-grams with TF as well as Bigrams with TF-IDF.

For the original + extended dataset the best performing model in terms of recall was again the Decision Tree. The recall score was 0.97 for Bi-grams with TF as well as Bi-grams with TF-IDF.

When it comes to the recall score, Logistic Regression performed the worst, followed by LSVM and finally Decision Tree being the top performer. All the results are shown below.

Original Dataset

UNIGRAM & TF

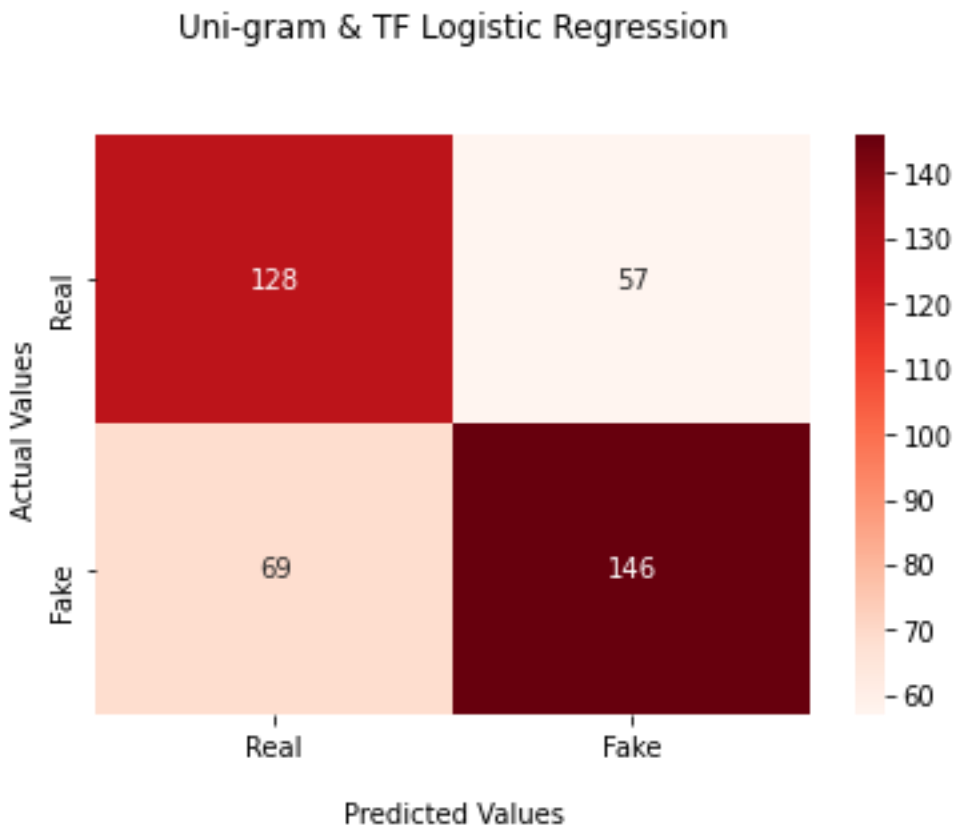

Figure 3 - Unigram and TF Logistic Regression confusion matrix

Table 8 – Unigram and TF Logistic Regression evaluation metrics

Precision	0.72
Recall	0.68
F1 Score	0.70

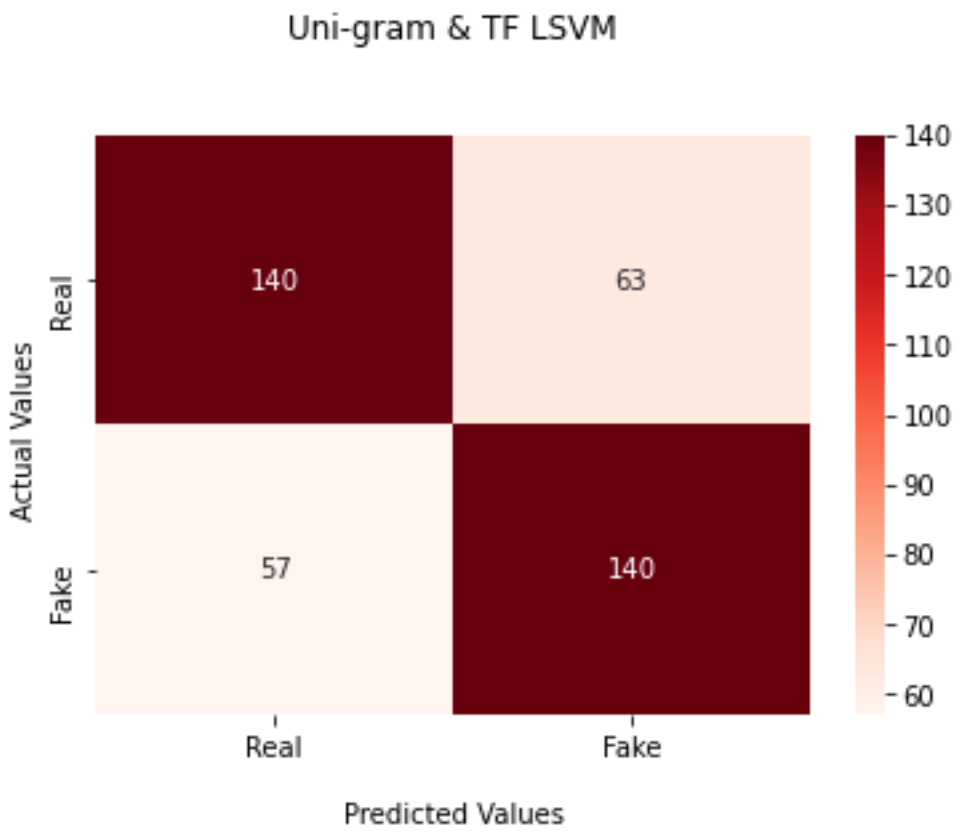

Figure 4 - Unigram and TF LSVM confusion matrix

Table 9 – Unigram and TF LSVM evaluation metrics

Precision	0.69
Recall	0.71
F1 Score	0.70

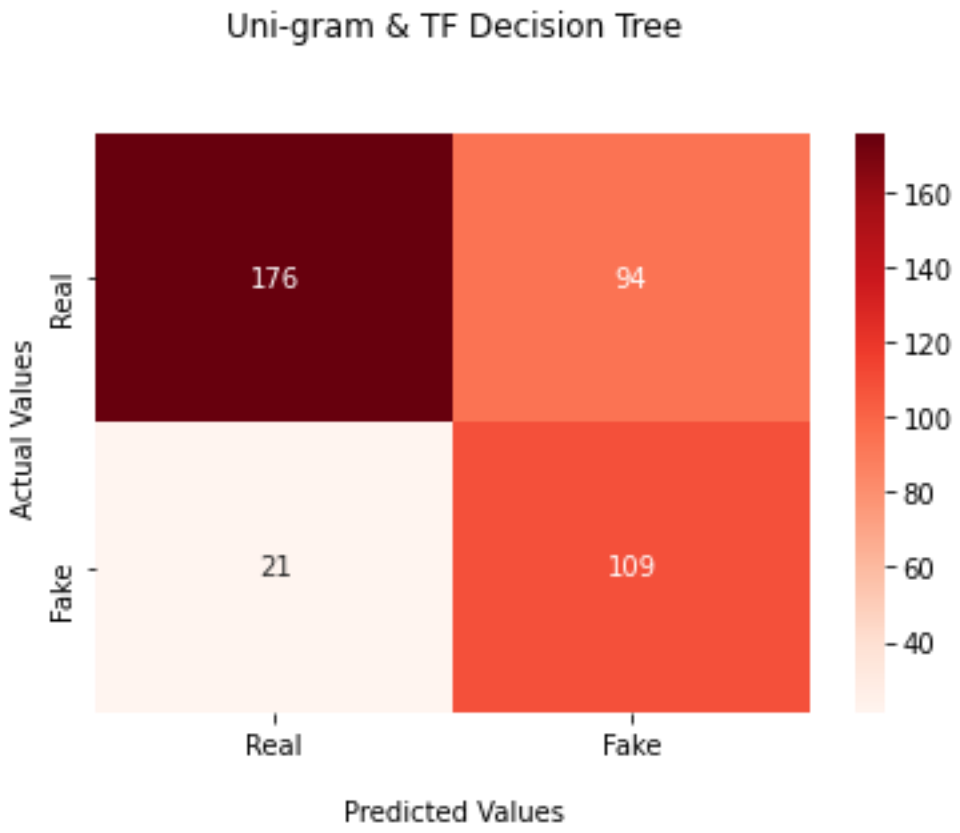

Figure 5 - Unigram and TF Decision Tree confusion matrix

Table 10 – Unigram and TF Decision Tree evaluation metrics

Precision	0.54
Recall	0.84
F1 Score	0.65

UNIGRAM & TF-IDF

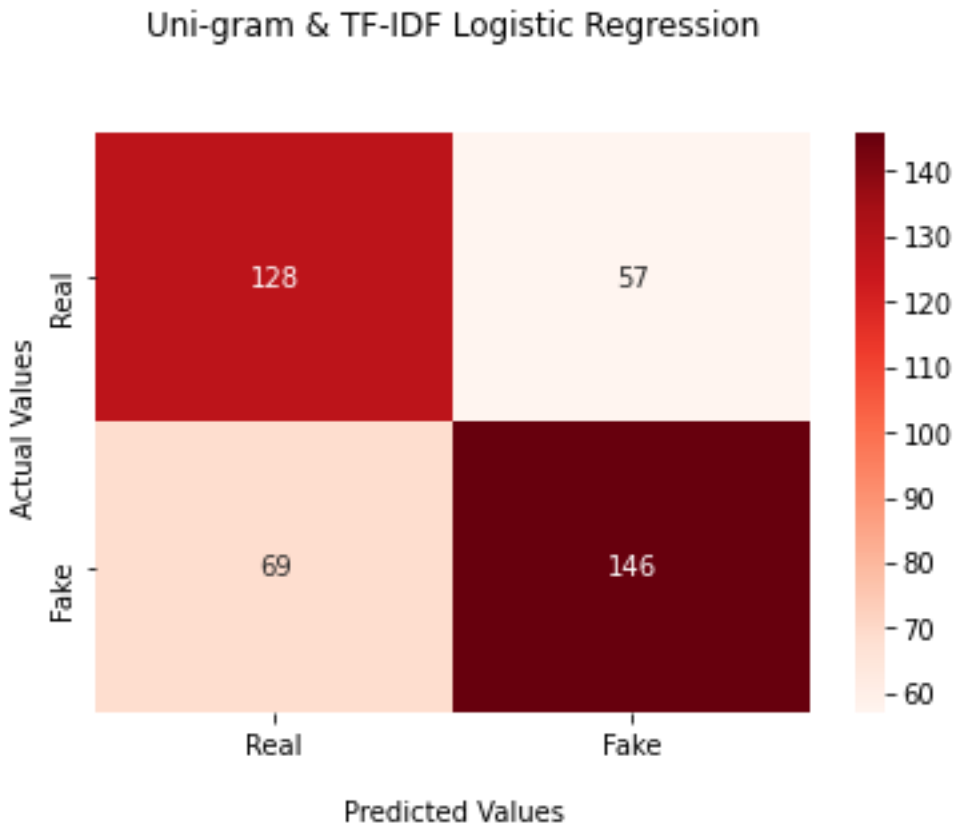

Figure 6 - Unigram and TF-IDF Logistic Regression confusion matrix

Table 11 – Unigram and TF-IDF Logistic Regression evaluation metrics

Precision	0.72
Recall	0.68
F1 Score	0.70

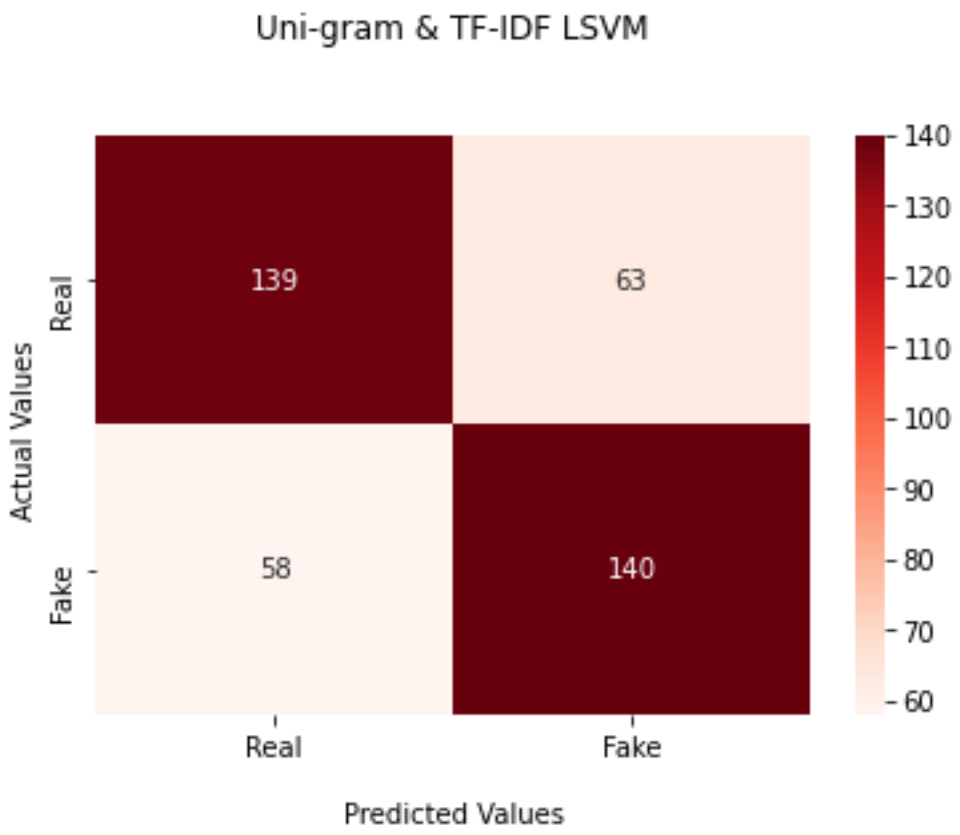

Figure 7 **- Unigram and TF-IDF LSVM confusion matrix**

Table 12 **– Unigram and TF-IDF LSVM evaluation metrics**

Precision	0.69
Recall	0.71
F1 Score	0.70

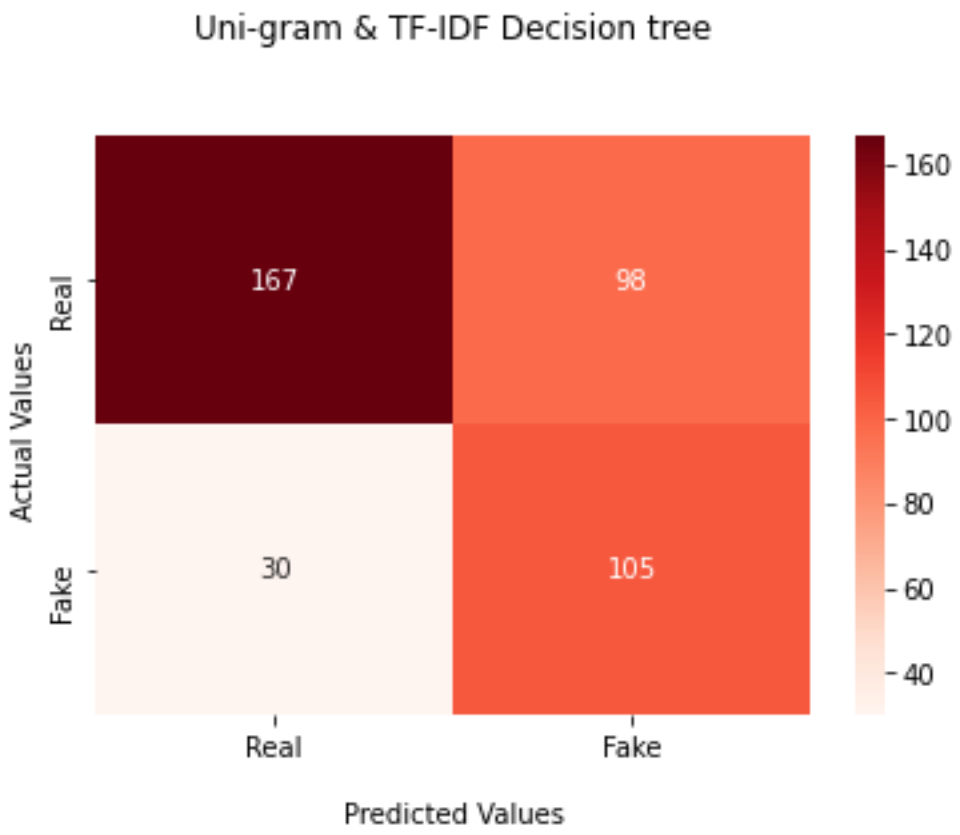

Figure 8 - Unigram and TF-IDF Decision Tree confusion matrix

Table 13 –Unigram and TF-IDF Decision Tree evaluation metrics

Precision	0.52
Recall	0.78
F1 Score	0.62

BIGRAM & TF

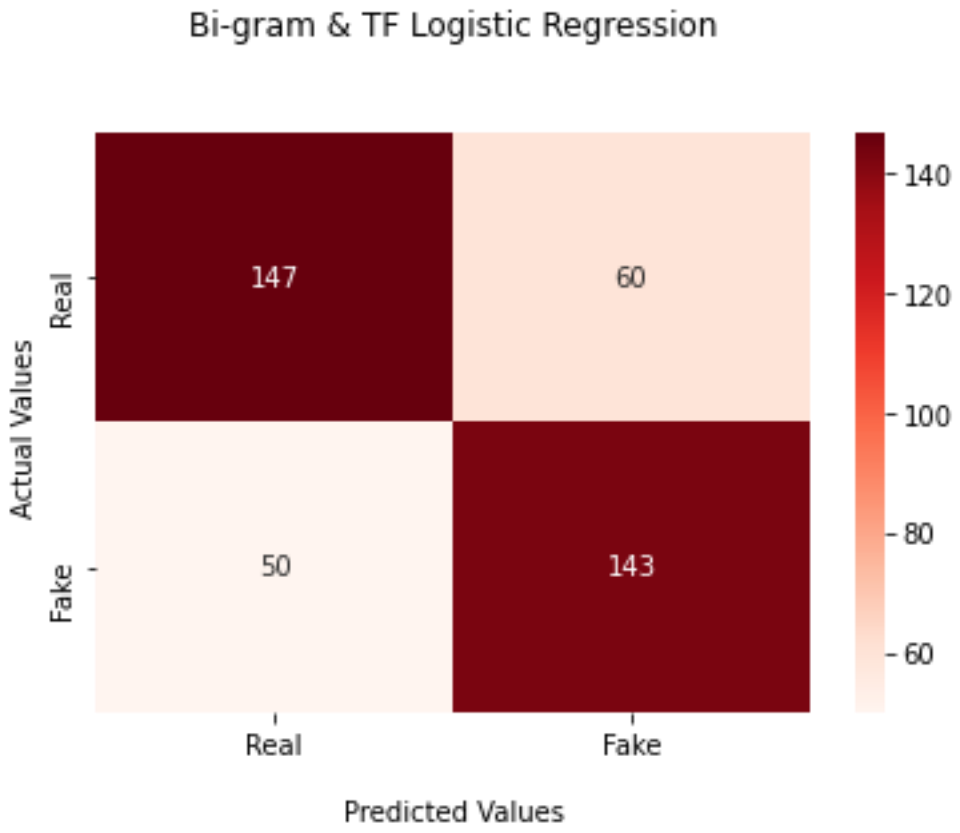

Figure 9 - Bigram and TF Logistic Regression confusion matrix

Table 14 – Bigram and TF Logistic Regression evaluation metrics

Precision	0.70
Recall	0.74
F1 Score	0.72

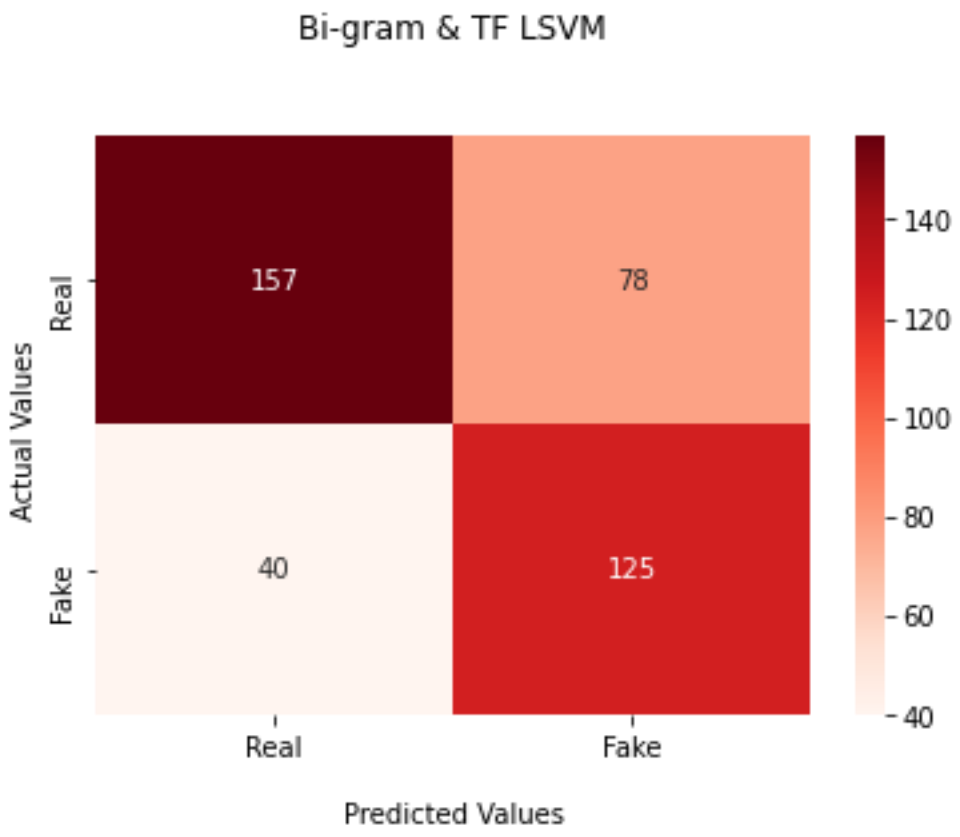

Figure 10 - Bigram and TF LSVM confusion matrix

Table 15 – Bigram and TF LSVM evaluation metrics

Precision	0.62
Recall	0.76
F1 Score	0.68

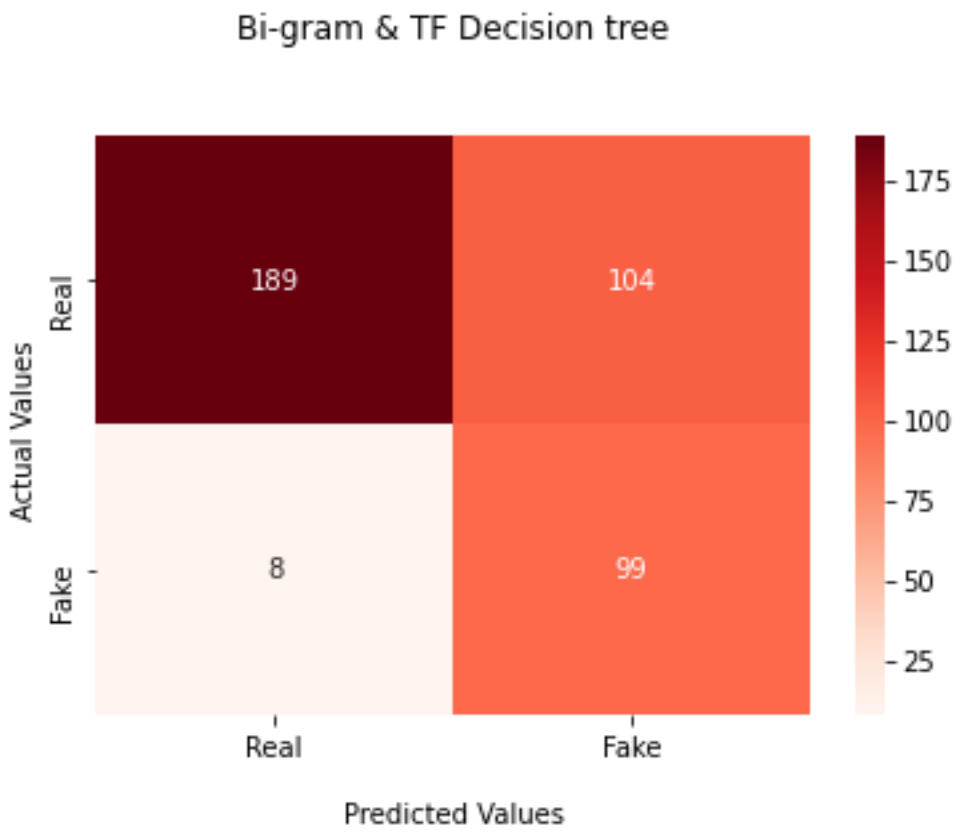

Figure 11 - Bigram and TF Decision Tree confusion matrix

Table 16 – Bigram and TF Decision Tree evaluation metrics

Precision	0.49
Recall	0.93
F1 Score	0.64

BIGRAM & TF-IDF

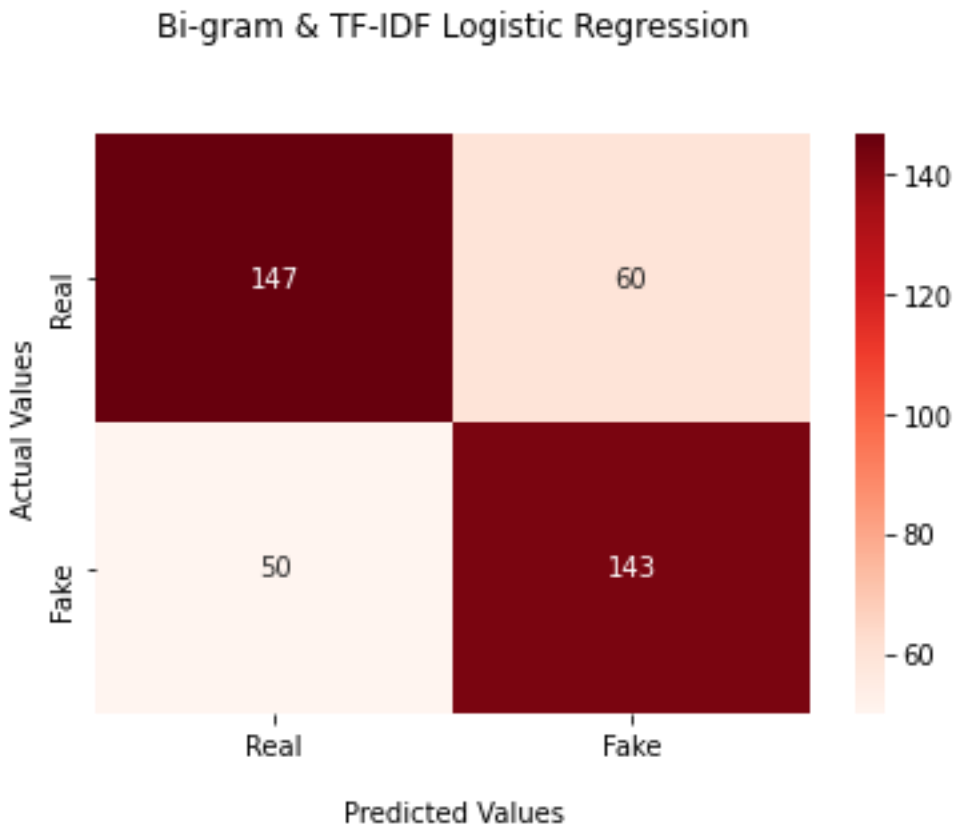

Figure 12 - Bigram and TF-IDF Logistic Regression confusion matrix

***Table 17* – Bigram and TF-IDF Logistic Regression evaluation metrics**

Precision	0.70
Recall	0.74
F1 Score	0.72

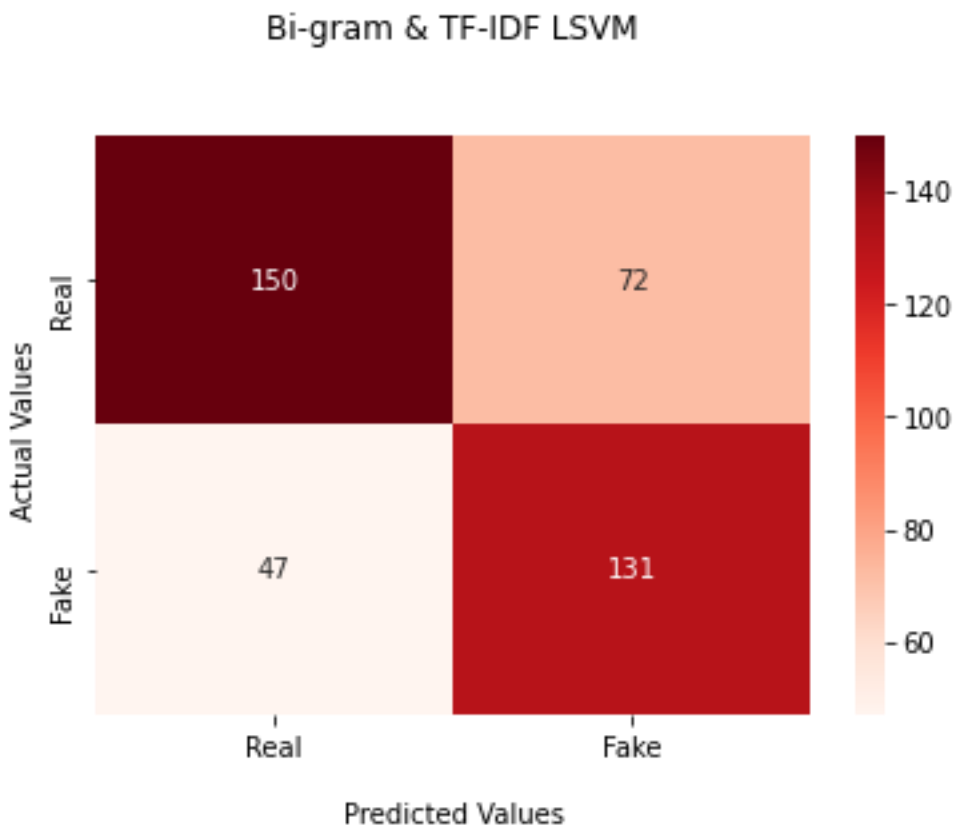

Figure 13 - Bigram and TF-IDF LSVM confusion matrix

Table 18 – Bigram and TF-IDF LSVM evaluation metrics

Precision	0.65
Recall	0.74
F1 Score	0.69

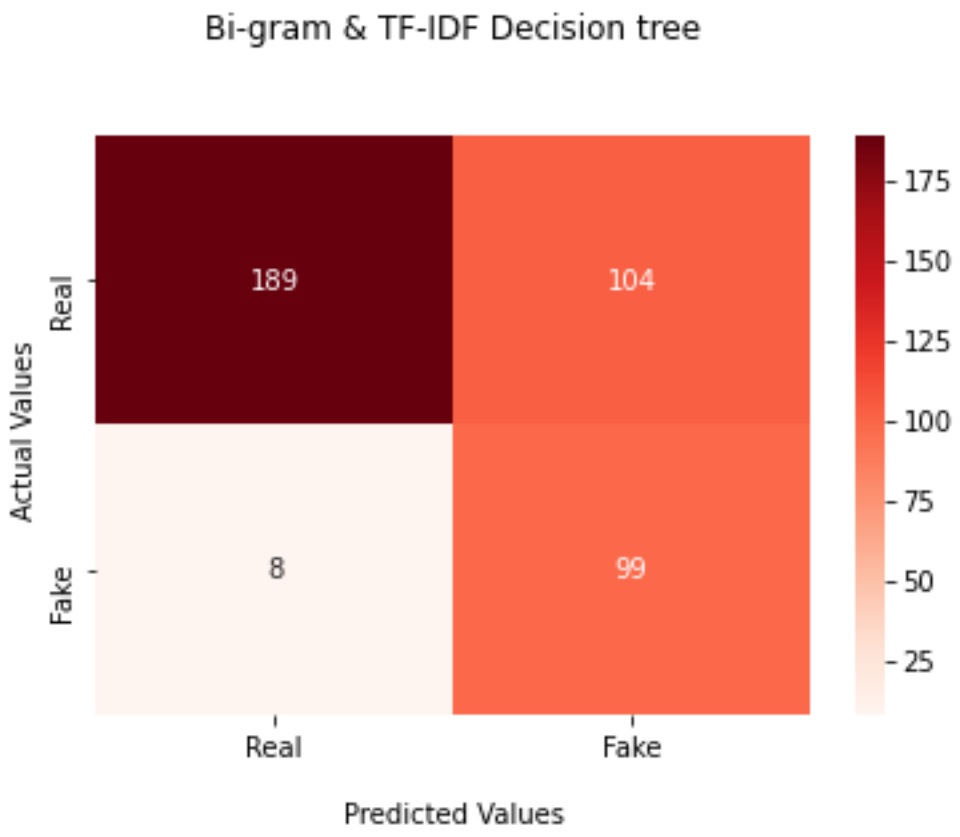

Figure 14 - Bigram and TF-IDF Decision Tree confusion matrix

Table 19 – Bigram and TF-IDF Decision Tree evaluation metrics

Precision	0.49
Recall	0.93
F1 Score	0.64

Original & Extended Dataset

UNIGRAM & TF

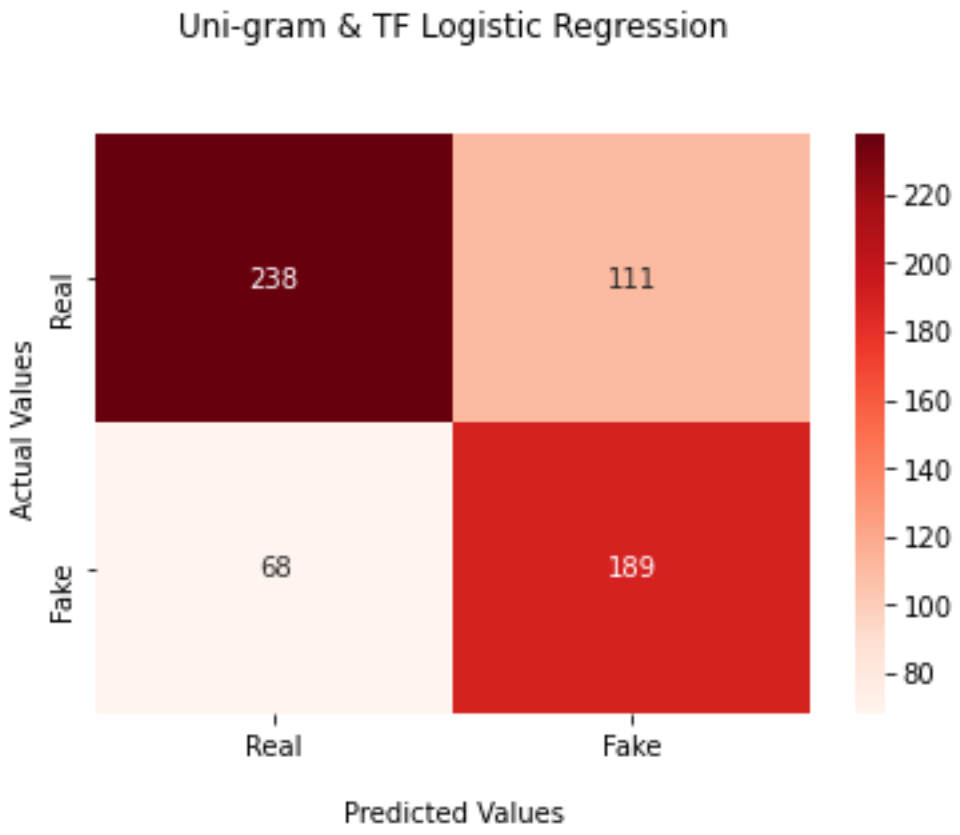

Figure 15 - Unigram and TF Logistic Regression confusion matrix

Table 20 –Unigram and TF Logistic Regression evaluation metrics

Precision	0.63
Recall	0.74
F1 Score	0.68

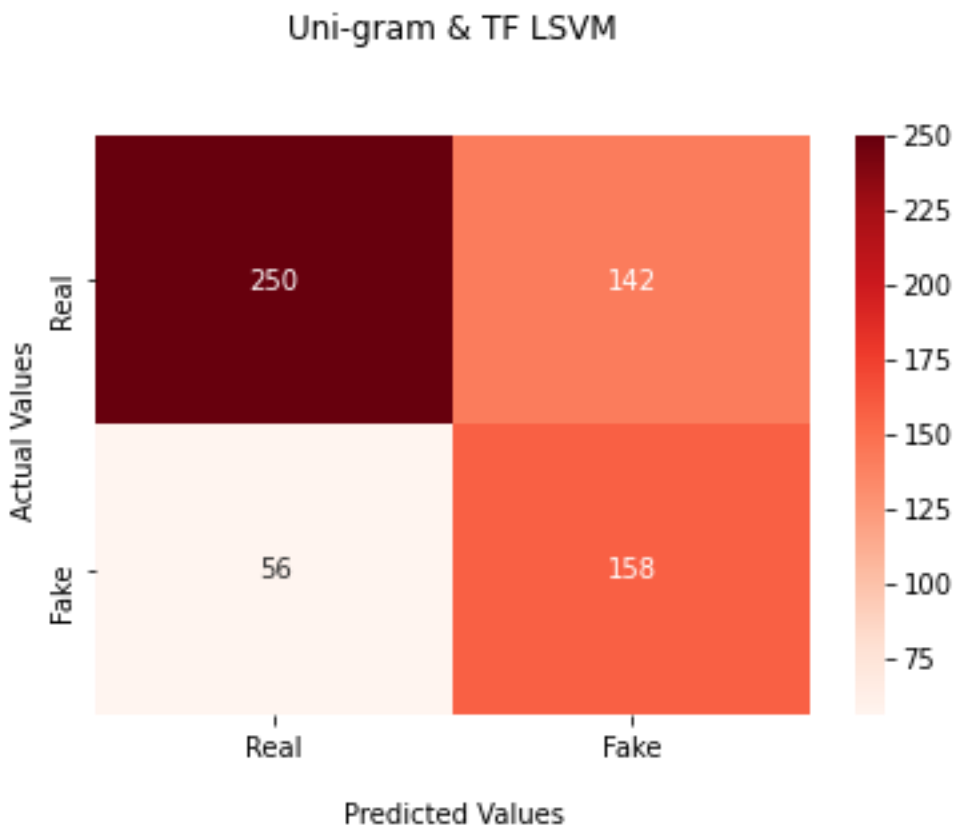

Figure 16 - Unigram and TF LSVM confusion matrix

Table 21 – Unigram and TF LSVM evaluation metrics

Precision	0.53
Recall	0.74
F1 Score	0.61

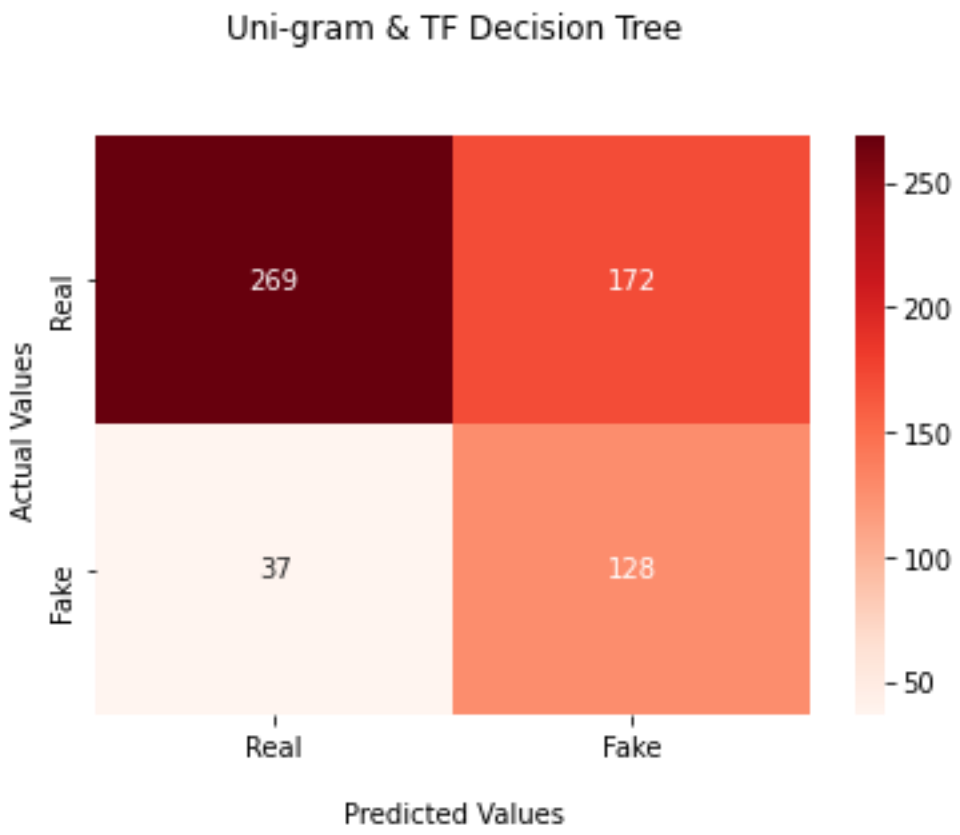

Figure 17 - Unigram and TF Decision Tree confusion matrix

Table 22 – Unigram and TF Decision Tree evaluation metrics

Precision	0.43
Recall	0.78
F1 Score	0.55

UNIGRAM & TF-IDF

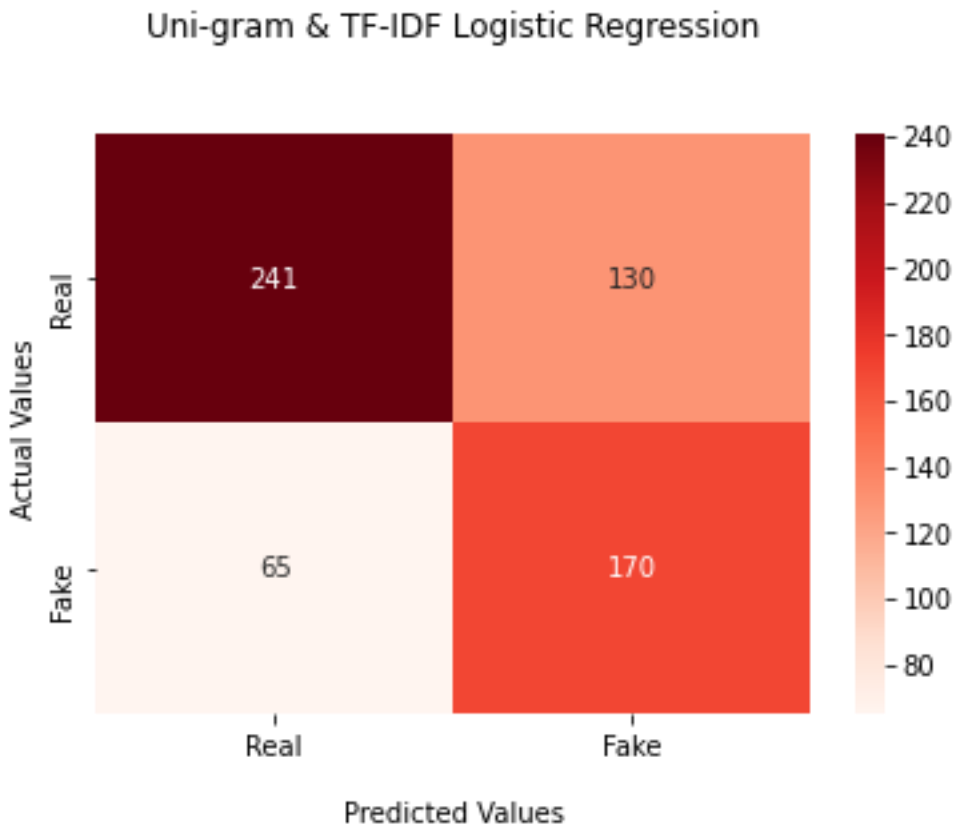

Figure 18 - **Unigram and TF-IDF Logistic Regression confusion matrix**

Table 23 – Unigram and TF-IDF Logistic Regression evaluation metrics

Precision	0.57
Recall	0.72
F1 Score	0.64

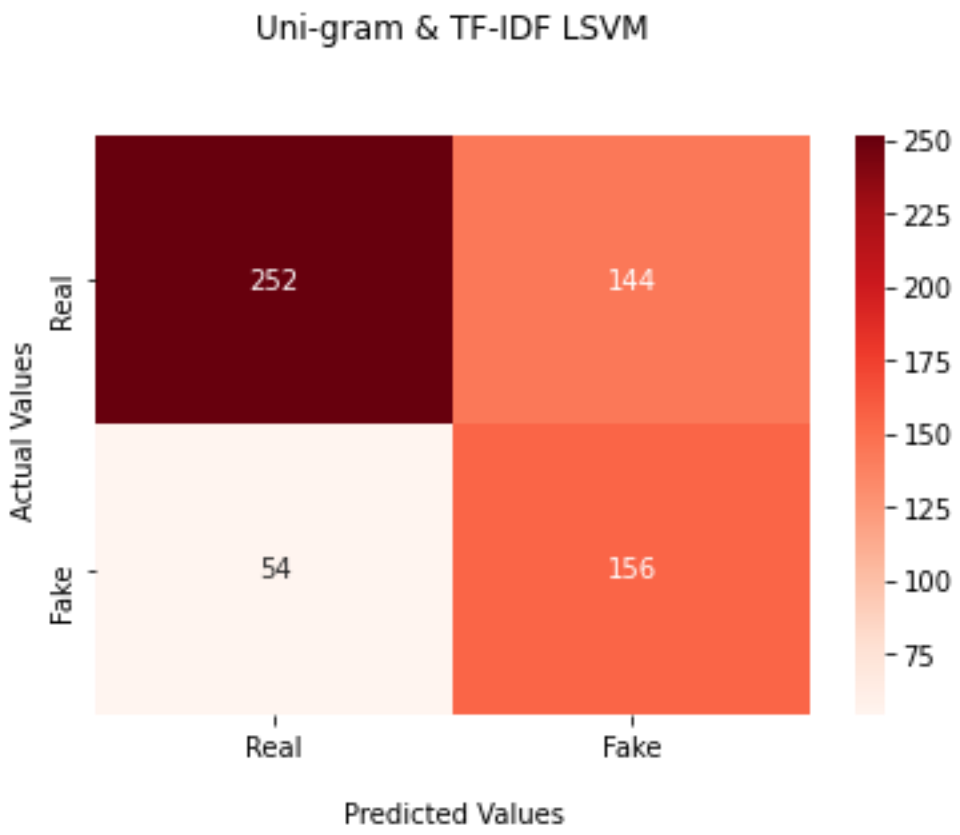

Figure 19 - Unigram and TF-IDF LSVM confusion matrix

Table 24 – Unigram and TF-IDF LSVM evaluation metrics

Precision	0.52
Recall	0.74
F1 Score	0.61

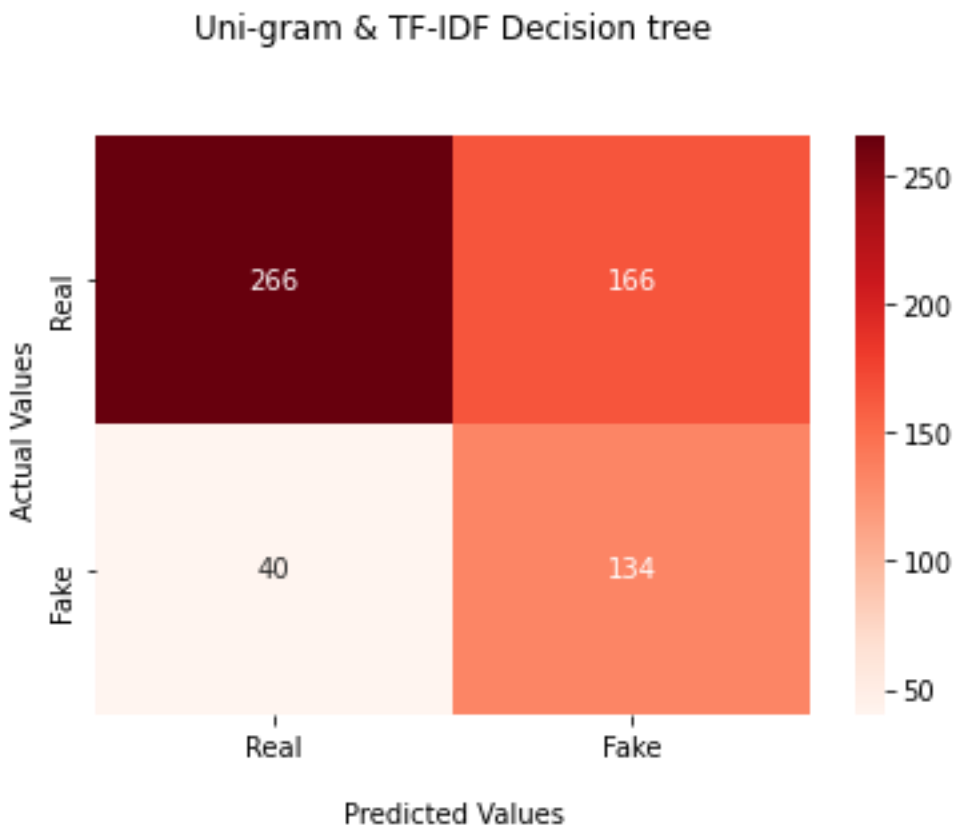

Figure 20 - Unigram and TF-IDF Decision Tree confusion matrix

Table 25 – Unigram and TF-IDF Decision Tree evaluation metrics

Precision	0.45
Recall	0.77
F1 Score	0.57

BIGRAM & TF

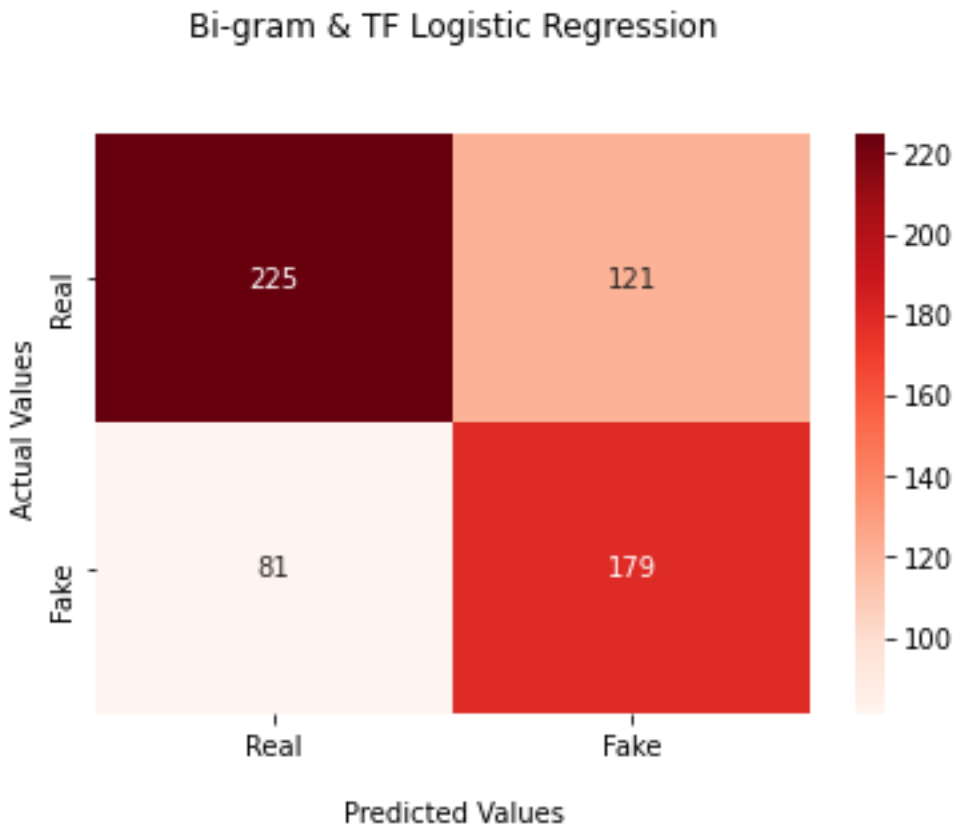

Figure 21 - Bigram and TF Logistic Regression confusion matrix

Table 26 – Bigram and TF Logistic Regression evaluation metrics

Precision	0.60
Recall	0.69
F1 Score	0.64

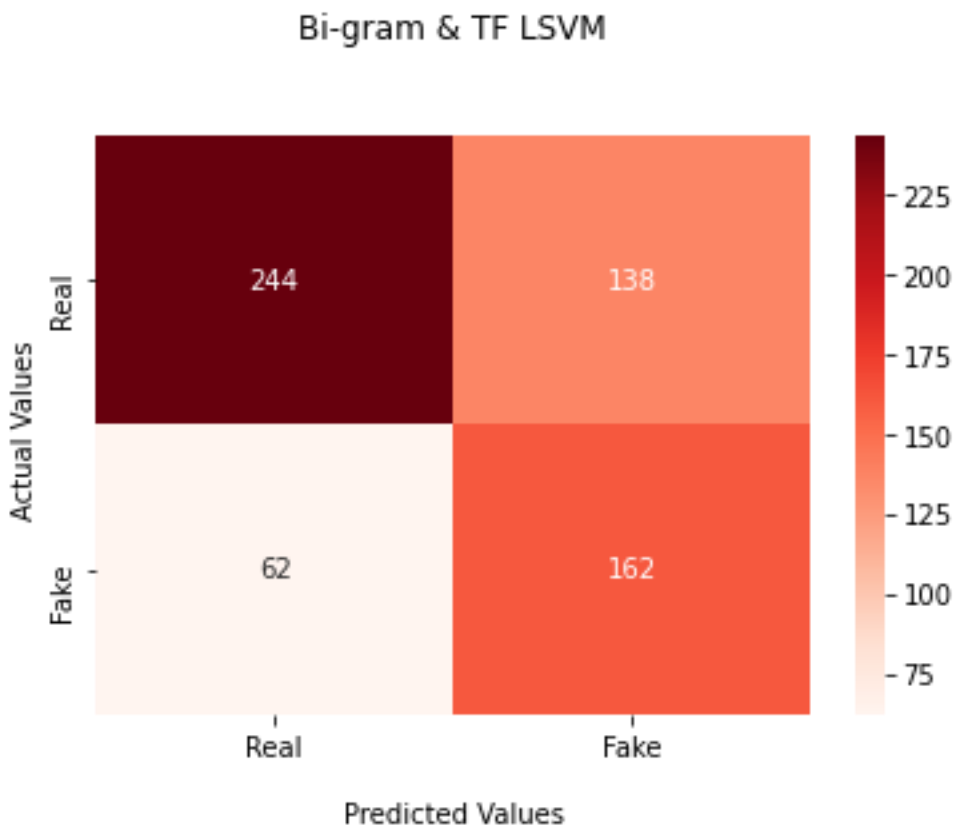

Figure 22 **- Bigram and TF LSVM confusion matrix**

Table 27 – Bigram and TF LSVM evaluation metrics

Precision	0.54
Recall	0.72
F1 Score	0.62

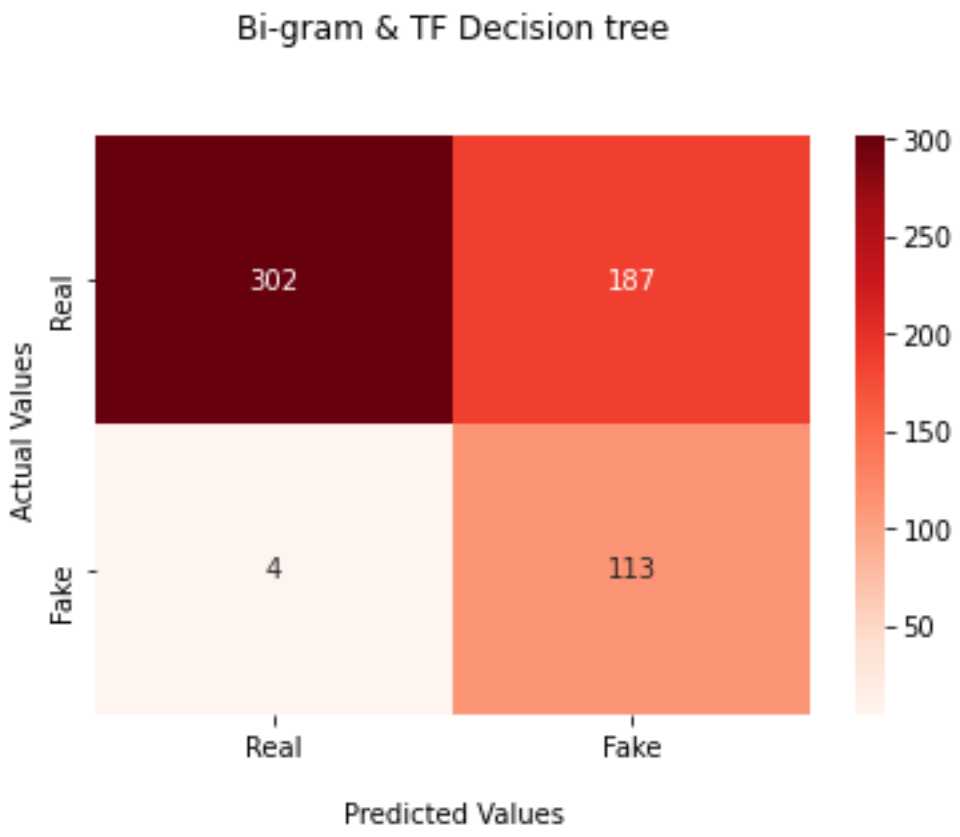

Figure 23 - Bigram and TF Decision Tree confusion matrix

Table 28 – Bigram and TF Decision Tree evaluation metrics

Precision	0.38
Recall	0.97
F1 Score	0.54

BIGRAM & TF-IDF

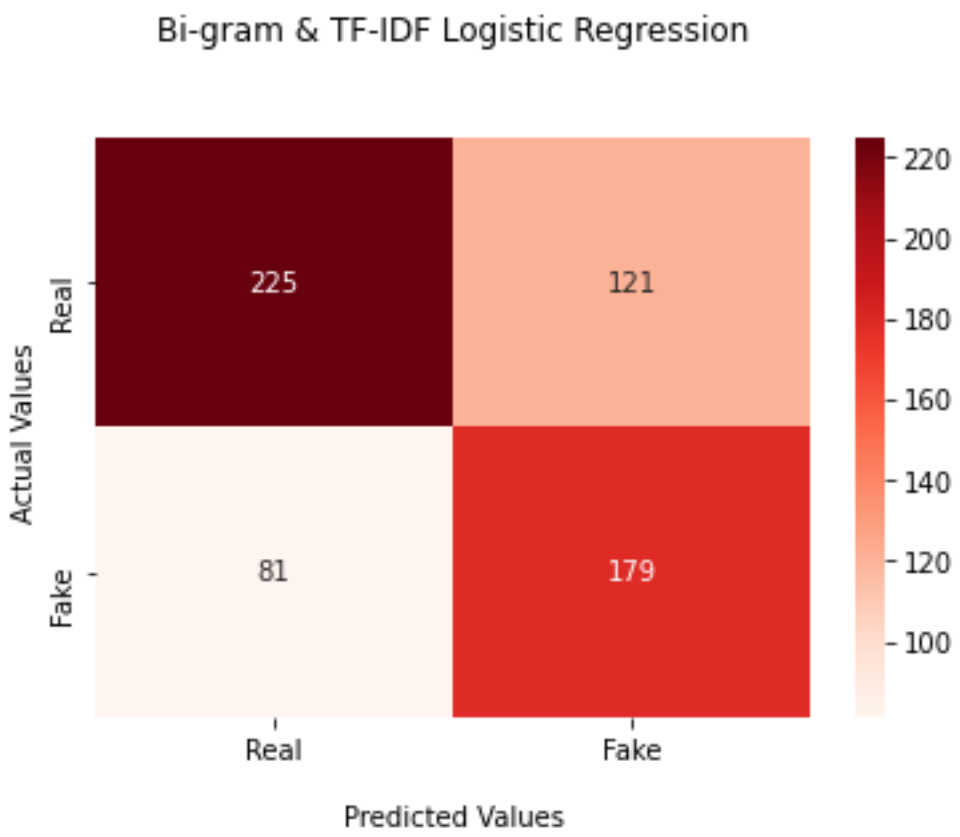

Figure 24 - **Bigram and TF-IDF Logistic Regression confusion matrix**

Table 29 – Bigram and TF-IDF Logistic Regression evaluation metrics

Precision	0.60
Recall	0.69
F1 Score	0.64

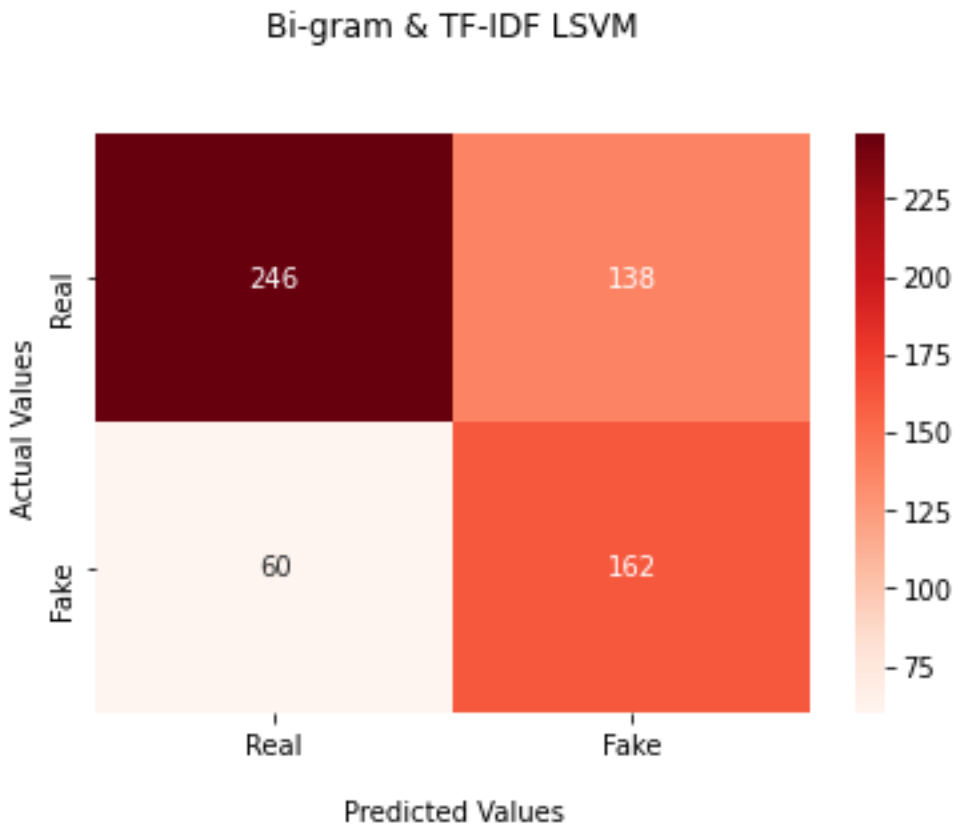

Figure 25 - Bigram and TF-IDF LSVM confusion matrix

Table 30 – Bigram and TF-IDF LSVM evaluation metrics

Precision	0.54
Recall	0.73
F1 Score	0.62

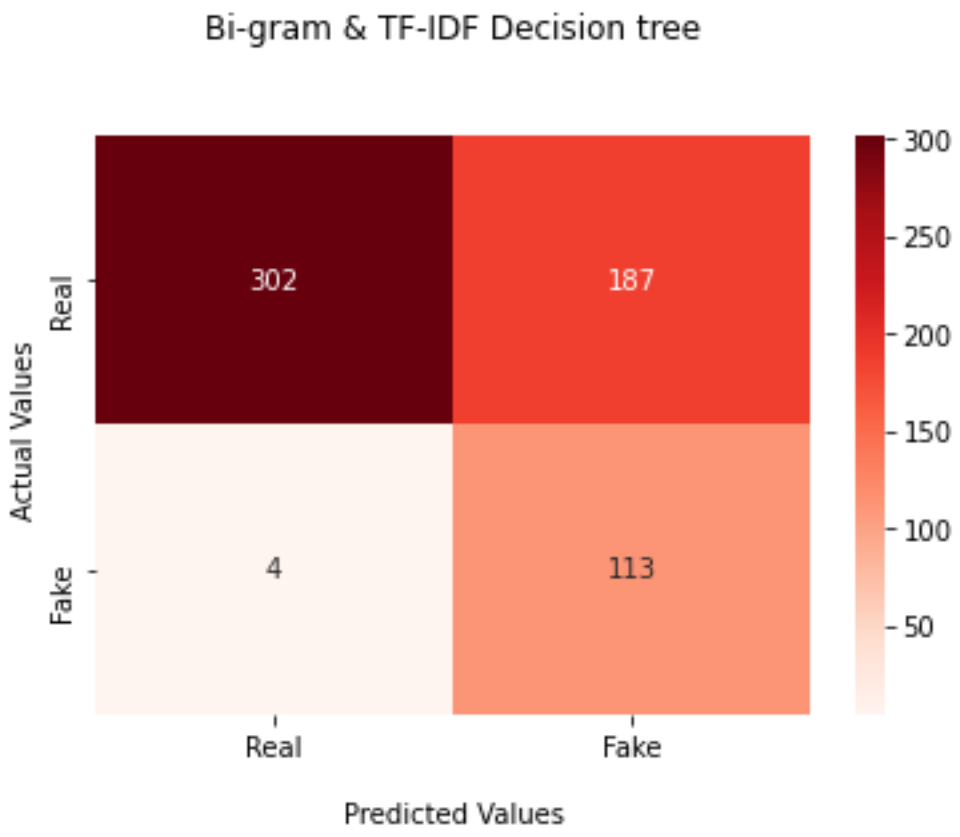

Figure 26 - **Bigram and TF-IDF Decision Tree confusion matrix**

Table 31 – Bigram and TF-IDF Decision Tree evaluation metrics

Precision	0.38
Recall	0.97
F1 Score	0.54

Hyper-parameter Performance Effect

Approach

Tuning the hyperparameters of our models either causes the models to be more or less accurate. This is the main purpose behind performing a grid search. We feed the grid search lists of hyperparameters that are to be tested for each model after it is trained on a training set. The grid search performs cross validation and compares the score of each hyper parameter combination and selects the best one to be used for the final model

The hyper-parameter used for the logistic regression and LSVM was regularization parameter. This parameter imposes a penalty for on the optimization function to prevent overfitting. Therefore, this parameter was a hyper-parameter of interest as the effect of generalization was being test using the cross validation. It was found that the more general model resulted in better predictions.

The following hyper-parameters were explicitly defined for each model during the grid search and cross-validation:

Logistic Regression:

- *Regularization Parameter:* [0.01, 0.1]
- *Count Vectorizer Vocabulary Size:* [1000, 5000, 10000]

LSVM:

- *Regularization Parameter:* [0.01, 0.1]
- *Count Vectorizer Vocabulary Size:* [1000, 5000, 10000]

Decision Tree:

- *Maximum Depth:* [2, 5]
- *Count Vectorizer Vocabulary Size:* [1000, 5000, 10000]

It was found that the best accuracy score was obtained with using logistic regression as the model with 10,000 bigram features and regularization parameter of 0.1. This resulted in the best overall accuracy score of 70%.

Results

The following tables show the results of the hyper-parameter tuning for the Logistic Regression model:

Table 32 –Accuracy Results Hyper-Parameter Tuning - Logistic Regression (Original Dataset)

Logistic Regression - Original Dateset							
N-gram Size	**Regularizaton Parameter**	**TF-IDF**			**TF**		
		1000	5000	10,000	1000	5000	10000
Uni-gram	0.1	64.4	66.5	67.9	64.4	66.5	67.9
	0.01	63.1	67	67.6	63.1	66.9	67.6
Bi-gram	0.1	66.1	65.8	65.9	69.4	69.1	70.6
	0.01	66.5	66.1	66.2	68.1	69.5	69.9

Table 33 –Accuracy Results Hyper-Parameter Tuning - Logistic Regression (Extended Dataset)

Logistic Regression - Extended Dateset							
N-gram Size	**Regularizaton Parameter**	**TF-IDF**			**TF**		
		1000	5000	10,000	1000	5000	10000
Uni-gram	0.1	66.5	67.7	67.1	65.8	67.4	66.7
	0.01	66	67.5	66.7	64.3	66.7	67.1
Bi-gram	0.1	68.9	70.0	69.5	69.0	70	69.5
	0.01	67.9	69.0	69.2	67.9	69	69.2

The following tables show the results of the hyper-parameter tuning for the LSVM model:

Table 34 –Accuracy Results Hyper-Parameter Tuning - LSVM (Original Dataset)

LSVM - Original Dateset							
N-gram Size	**Regularizaton Parameter**	**TF-IDF**			**TF**		
		1000	5000	10,000	1000	5000	10000
Uni-gram	0.1	64.5	65	66.7	64.6	64.9	66.5
	0.01	64.4	65	67	64.5	65.2	67
Bi-gram	0.1	63.6	64.5	65.3	68.0	70.0	70.0
	0.01	63.5	64.7	64.9	68.2	70.0	70.0

Table 35 –Accuracy Results Hyper-Parameter Tuning - LSVM (Extended Dataset)

LSVM - Extended Dateset							
N-gram Size	**Regularizaton Parameter**	**TF-IDF**			**TF**		
		1000	5000	10,000	1000	5000	10000
Uni-gram	0.1	67	66.7	67.3	66.5	66.3	67.4
	0.01	66.7	67.1	67.3	65.8	66.8	67.1
Bi-gram	0.1	69.5	69.6	70.1	69.4	69.8	70.2
	0.01	69	69.8	70.3	69.2	69.4	70.2

The following tables show the results of the hyper-parameter tuning for the Decision Tree model:

Table 36 –Accuracy Results Hyper-Parameter Tuning - Decision Trees (Original Dataset)

Decision Tree - Original Dateset							
N-gram Size	**Max Depth**	**TF-IDF**			**TF**		
		1000	5000	10,000	1000	5000	10000
Uni-gram	2	67.1	67.1	67	65.3	65.3	65.3
	5	67.4	68.3	68.4	65.9	65.9	66.0
Bi-gram	2	65	64.9	64.9	64.8	64.7	64.8
	5	67.1	66.4	66.9	66.1	65.8	66.6

Table 37 –Accuracy Results Hyper-Parameter Tuning – Decision Trees (Extended Dataset)

Decision Tree - Extended Dateset							
N-gram Size	**Max Depth**	**TF-IDF**			**TF**		
		1000	5000	10,000	1000	5000	10000
Uni-gram	2	69.1	69.5	69.8	67.6	67.9	67.9
	5	68.9	70.1	70.1	67.75	68.5	68.5
Bi-gram	2	67.3	67.1	67.2	67.4	67.2	67.2
	5	68.3	68.6	69.1	68.3	68.6	69.1

Misclassification Handling

The best model in terms of accuracy was found to be logistic regression, using 10,000 features, bigrams with a regularization parameter of 0.1. The model did not perform well when bigrams appeared in a category (e.g., fake) with low TF-IDF for that specific bigram but which had a high TF-IDF in the other (e.g., real).

The way to determine which bigram occurred in which misclassified record would be to generate a figure with the importance matrix for the features (bigrams) in each category of label and look for the opposing label high importance feature. Unfortunately feature importance cannot be accessed for logistic regression and so now a complex user defined function would have to be made that obtains the coefficients for each bigram, which is the beyond the scope of this course.

An example, however, can be given to further illustrate the point. If an article is classified as fake but it was real, it would often contain bigrams such as “doom extinction” however, this is taken out of context and is not representative of the articles message. To avoid this from happening, higher number of features should be included along with more datapoints so that a richer vocabulary can be used as features in the model.

Discussion

Hamza Luqman:
Industrial sectors benefit most from machine learning. The advantages of implementing AI through machine learning algorithms and Classification techniques can be implemented in speech recognition, handwriting recognition, biometric identification, document classification.

Muhammad Daud Sheikh:
The rise of false news resulting in large fluctuations in the stock market is a well-known phenomenon. Many predatory news sources frequently target investor which are easily influenced by fake news to buy and sell stocks. This practice is known as “pump and dump” schemes and are common for penny stocks. Using the machine learning model that is described in this report, some of these false news stories can be caught early and probed to determine if they are true or false. This will result in investors making informed decision and a general reduction of crime involving financial gain due to misinformation.

Okeoghenemarho Obuareghe:
The common job of classification algorithms is to recognize objects and be able to separate them into categories. It helps us segregate large quantities of data into discrete values distinctively i.e. like 0/1, True/False, or a pre-defined output label class like in our case Real/Fake. Our classification model can find applications in industry in Email Spam detection and filtering, where a model can help organize incoming email and detect computer viruses and spam, so appropriate action can be taken against them before causing harm. Our model can also find usefulness in Text Analysis and Text Predictions programs utilizing efficient NLTK techniques to process Language.

Conclusion

Fake news detection is becoming increasingly important in a highly digitised society. Misinformation is now affecting important political decisions. To combat this phenomenon, fake news detection algorithms can be used to identify possible fake news.

This report details an approach taken by Hadeer [1] involving using various statistical models with various combinations of features and hyper-parameters to determine the best model which can be used for identifying fake news in terms of accuracy. Out team has given more importance to recall since the we want to identify as many fake news articles as possible (fake is defined as the positive case).

The machine learning efforts involved pre-processing the data by lower casing, regex filtering, tokenization, stop word removal, noise removal, and stemming. After this was performed, The data was then split into a training dataset and test dataset. A grid search was run various hyper-parameters for three models. These models included logistic regression, LSVM, and decision trees as well as various combination of unigram/bigram and number of features used in the feature matrix.

The best model to optimize for accuracy in detecting fake news uses Logistic Regression. The optimal parameters for this model were found to be 10,000 bigram features and a regularisation parameter of 0.1. Accuracy: 70% was found to be highest amongst both the extended and original dataset.

The model that resulted in the best recall was the decision tree with maximum depth of 5 and the features used were 10,000 bigrams.

References

[1] Hadeer Ahmed, Issa Traore, and Sherif Saad. 2017. Detection of Online Fake News Using N-Gram Analysis and Machine Learning Techniques. *Lecture Notes in Computer Science Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments* (2017), 127–138. DOI:http://dx.doi.org/10.1007/978-3-319-69155-8_9

[2] Anon. Extracting, transforming, and selecting features. Retrieved December 16, 2021 from https://spark.apache.org/docs/latest/ml-features

[3] wikipedia.org. 2021. Cohen's kappa - Wikipedia. [online] Available at: https://en.wikipedia.org/wiki/Cohen%27s_kappa [Accessed 15 December 2021].

[4] Zach, V., 2021. Cohen's Kappa Statistic: Definition & Example. [online] Statology. Available at: https://www.statology.org/cohens-kappa-statistic/ [Accessed 15 December 2021].

[5] Emily Stweart, America's growing fake news problem, in one chart,Dec 22, 2020. [Online]. Available: https://www.vox.com/policy-and-politics/2020/12/22/22195488/fake-news-social-media-2020. [Accessed October 10, 2021].

[6] Clément Bisaillon. 2020. Fake and real news dataset. (March 2020). Retrieved December 16, 2021 from https://www.kaggle.com/clmentbisaillon/fake-and-real-news-dataset