“La Sapienza” University of Rome
Faculty of information engineering, information technology and statistics
Department of informatics, automation and control engineering
"ANTONIO RUBERTI"
Degree program: Artificial Intelligence and Robotics

BIG DATA FOR OFFICIAL STATISTICS

OLUWATOYIN SANNI
Artificial Intelligence and Robotics
1871835

This report is accompanied with two major files:

1. **My code in Jupyter notebook.**
2. **My powerpoint presentation.**

1.0 DEFINITION

Project Overview

According to wikipedia, **Fake news** is false or misleading information presented as news.It often has the aim of damaging the reputation of a person or entity, or making money through advertising revenue. Fake news **can reduce the impact of real news** by competing with it; a Buzzfeed analysis found that the top fake news stories about the 2016 U.S. presidential election received more engagement on Facebook than top stories from major media outlets. It also has the potential to **undermine trust** in serious media coverage.

In this project, I have built a model to classify a news content as genuine or fake. Compared to the study in which **humans were able to detect fake news only 70 percent of the time**, the classifier developed was able to achieve a 95% accuracy on our test set.

Problem Statement

The dataset for this problem was obtained from MachineHack and Kaggle. The first dataset consists of the fake news, a corresponding tag, and the target class while the second consists of the fake news text, the author of the news and the target class. The columns not common to both datasets are tags and authors which were removed after combining the two datasets. They were removed in order to avoid our model from having to deal with too many rows of unknown values.

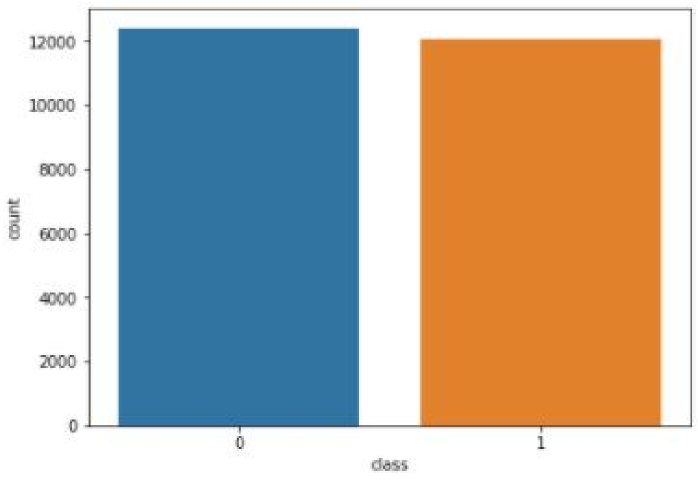


Figure 1

Figure 1 above shows a **normal distribution** of our binary class after a combination of our datasets in a shuffled way, this **balanced** dataset is perfect for any model classification.

Metrics

Metrics is one of the major way to benchmark a Machine learning model. Accuracy seems to be the most popular benchmark metric to validate machine learning models. However, I explored an additional metric which is the F1 score. This is because Classification accuracy alone is typically not enough information to evaluate the performance of our model. Hence, I have used both Accuracy and F1 measure because of this, while focusing more on F1. I have defined **two terms** below to elaborate on the F1 measure.

- **Precision**: It is the number of positive predictions divided by the total number of positive class values predicted. Precision can be thought of as a measure of a classifiers exactness. A low precision can also indicate a large number of False Positives.

- **Recall**: It is the number of positive predictions divided by the number of positive class values in the test data. It is also called Sensitivity or the True Positive Rate. Recall can be thought of as a measure of a classifiers completeness. A low recall indicates many False Negatives.

The **F1 Score** is the 2*((precision*recall)/(precision+recall)). Put another way, the **F1 score** conveys the balance between the precision and the recall.

2.0 DATA ANALYSIS

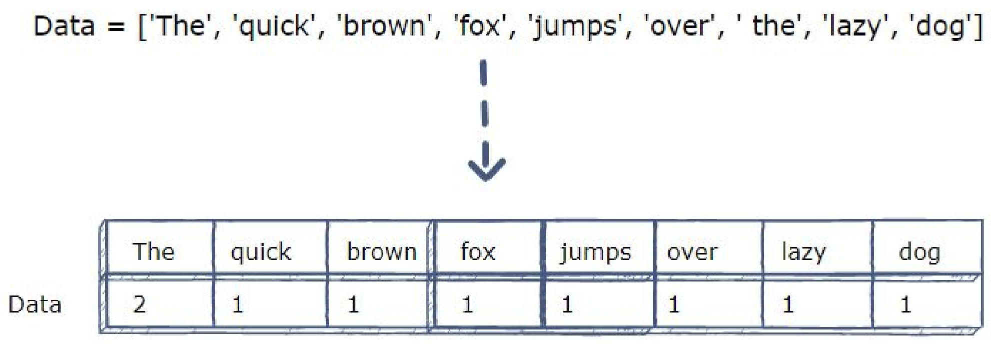
Data Cleaning

In order to make the data less noisy and to retain only important information that will be relevant to the model learning, I made use of the NLTK python package to do some data clean up. I removed punctuations, and stopwords as well.

Data Pre-processing and Extraction

Machine Learning models understand the language of "numbers". Since our data is in text. In order to use textual data for predictive modeling, the text must be parsed to remove certain words – this process is called **tokenization**. These words need to then be encoded as integers, or floating-point values, for use as inputs in machine learning algorithms. This process is called f**eature extraction (or vectorization)**. I explored two ways;

- **Count Vectorizer using Sklearn**: Scikit-learn's CountVectorizer is used to convert a collection of text documents to a vector of term/token counts. It also enables the pre-processing of text data prior to generating the vector representation. This functionality makes it a highly flexible feature representation module for text. A visual representation is shown below.

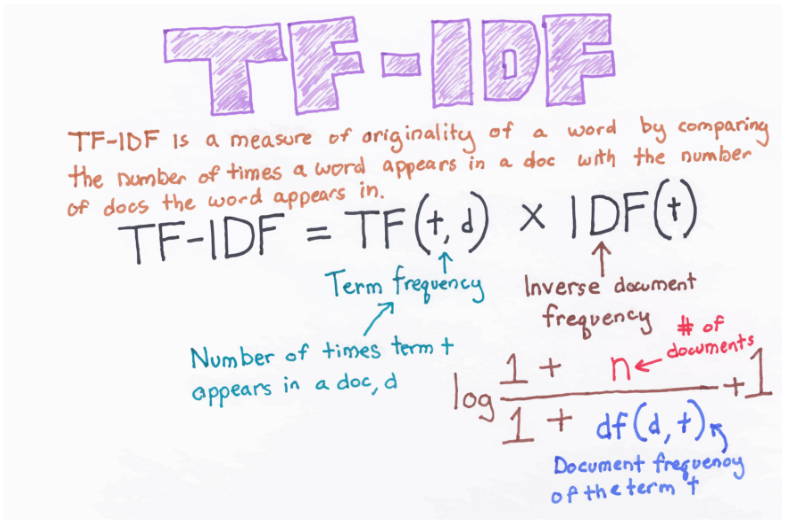


```
Data = ['The', 'quick', 'brown', 'fox', 'jumps', 'over', ' the', 'lazy', 'dog']
```

	The	quick	brown	fox	jumps	over	lazy	dog
Data	2	1	1	1	1	1	1	1

Figure 2

- **TF-IDF Vectorizer**: is an abbreviation for Term Frequency Inverse Document Frequency. This is very common algorithm to transform text into a meaningful representation of numbers which is used to fit machine algorithm for prediction. Unlike Count Vectorizer, TF-IDF considers the overall documents of weight of words. It weights the word counts by a measure of how often they appear in the documents.



TF-IDF is a measure of originality of a word by comparing the number of times a word appears in a doc with the number of docs the word appears in.

$$\text{TF-IDF} = \text{TF}(t, d) \times \text{IDF}(t)$$

$$\log \frac{1 + n}{1 + df(d, t)} + 1$$

TF-IDF
Term frequency
Number of times term t appears in a doc, d
Inverse document frequency
# of documents
Document frequency of the term t

Figure 3

Exploratory Visualization

1. Word Count of the most frequent Authors for Fake News:

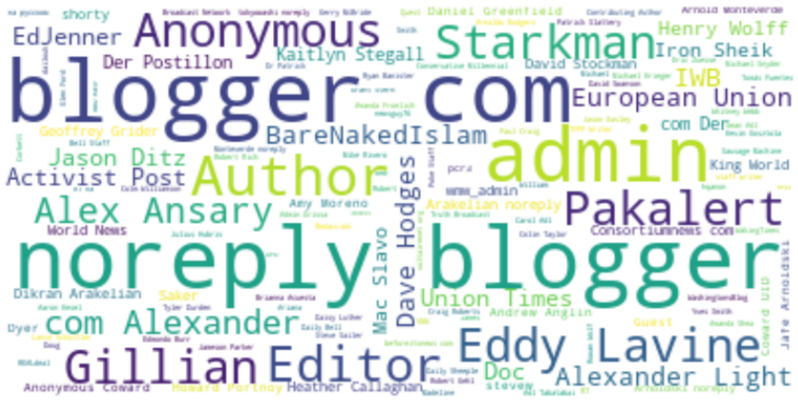


Figure 4

For the first dataset which has a column for the authors, Figure 4 shows a word count on the Authors that are mostly responsible for news content that are fake. As observed, **bloggers**, and key names like **noreply, admin, Anonymous, Author** are found to belong to this category. This is not so surprising as even spam emails authors have a resemblance with this occurrence.

2. Word Count of the most frequent Authors for Genuine News:

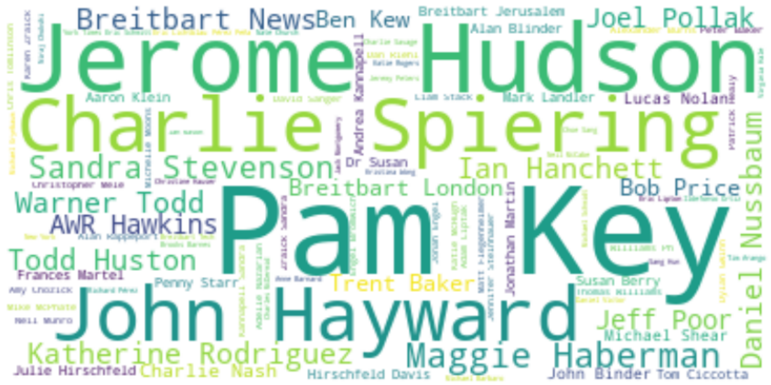


Figure 5

Figure 5 shows us real names for Genuine News as against the popular occurrence that we see in Figure 4. This is an important feature that our machine learning model can adopt in its learning process.

3. Pie Chart of Tags for Genuine News:

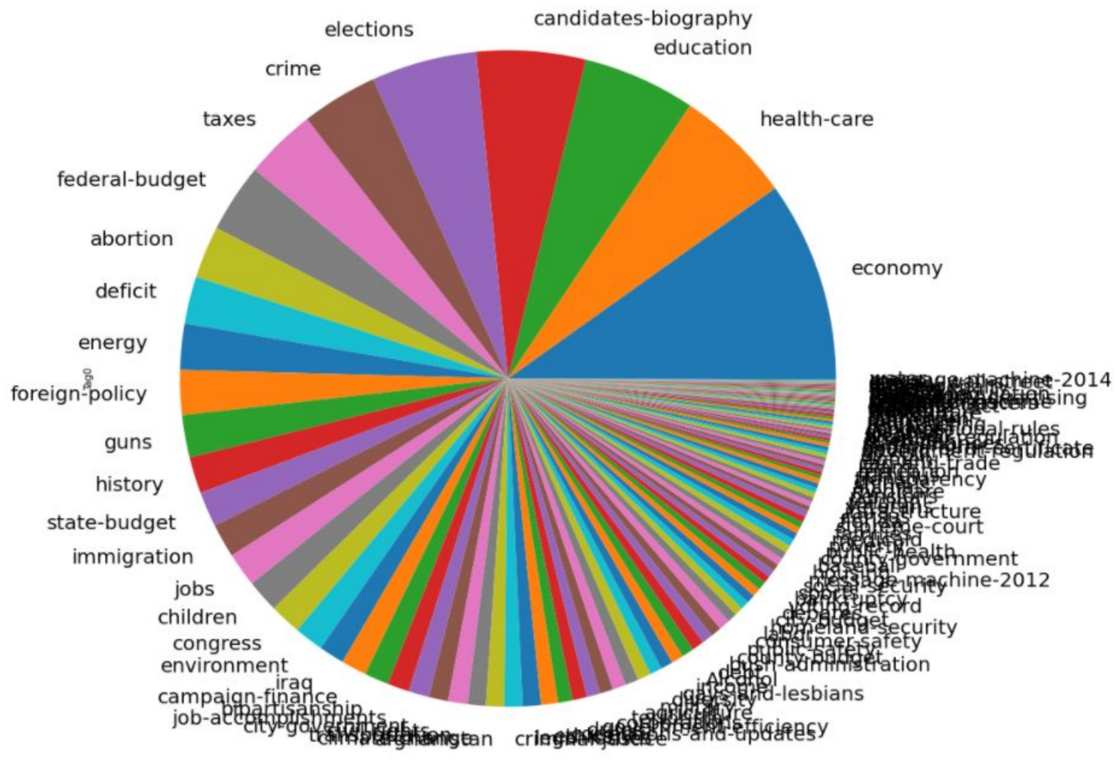


Figure 6

I also plotted a pie chart to display the tags that falls under Genuine News in their frequency order. As seen, tags like Economy, Health Care, Education are topping the list and this signifies that genuine news may belong most of the time belong to these categories.

4. **Pie Chart of Tags for Fake News:**

Figure 7 shows a similar pie chart, but for Fake News. Interestingly, Healthcare and Economy seems to belong to tags that belong to both Genuine and Fake news except that Health care tops the list for Fake news while Economy tops Genuine News(Figure 6). Hence, we can conclude there's no distinctive tag that does not belong to both genuine and fake news.

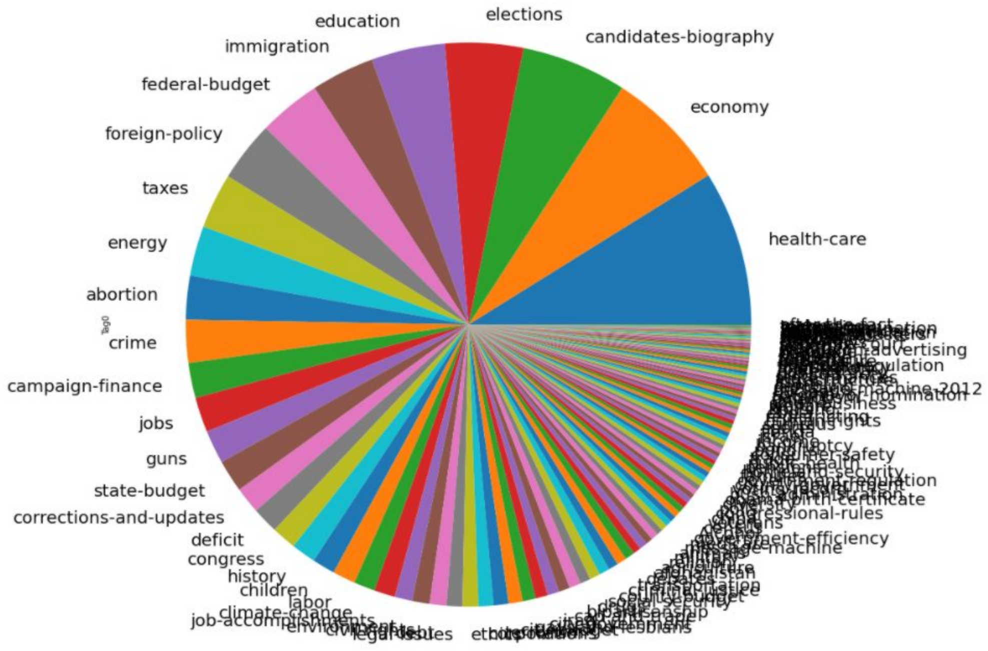


Figure 7

3.0 EXPERIMENTS AND EVALUATION

Since our combined dataset is balanced, it was straightforward to jump into building our model from our vectorization. The entire dataset was splitted into 70% for training and 30% for cross validation. The different methods explored and their results are further discussed.

- **Experiment One:** I employed the use of Statistical models using the python sklearn library. I used both Count Vectorizer and TF-IFD with **Random Forest Classifier, Support Vector Machine, Naive Bayes** and **Logistic Regression**. These models were trained on data that **has not been cleaned up** (i.e removal of punctuations, stop words, etc). They were all evaluated with the F1 measure. **Logistics Regression** with the **Count vectorizer** shows a better generalization of the data with an F1 score of **0.888** on the test set. Figure 8 shows these results in a horizontal bar chart.

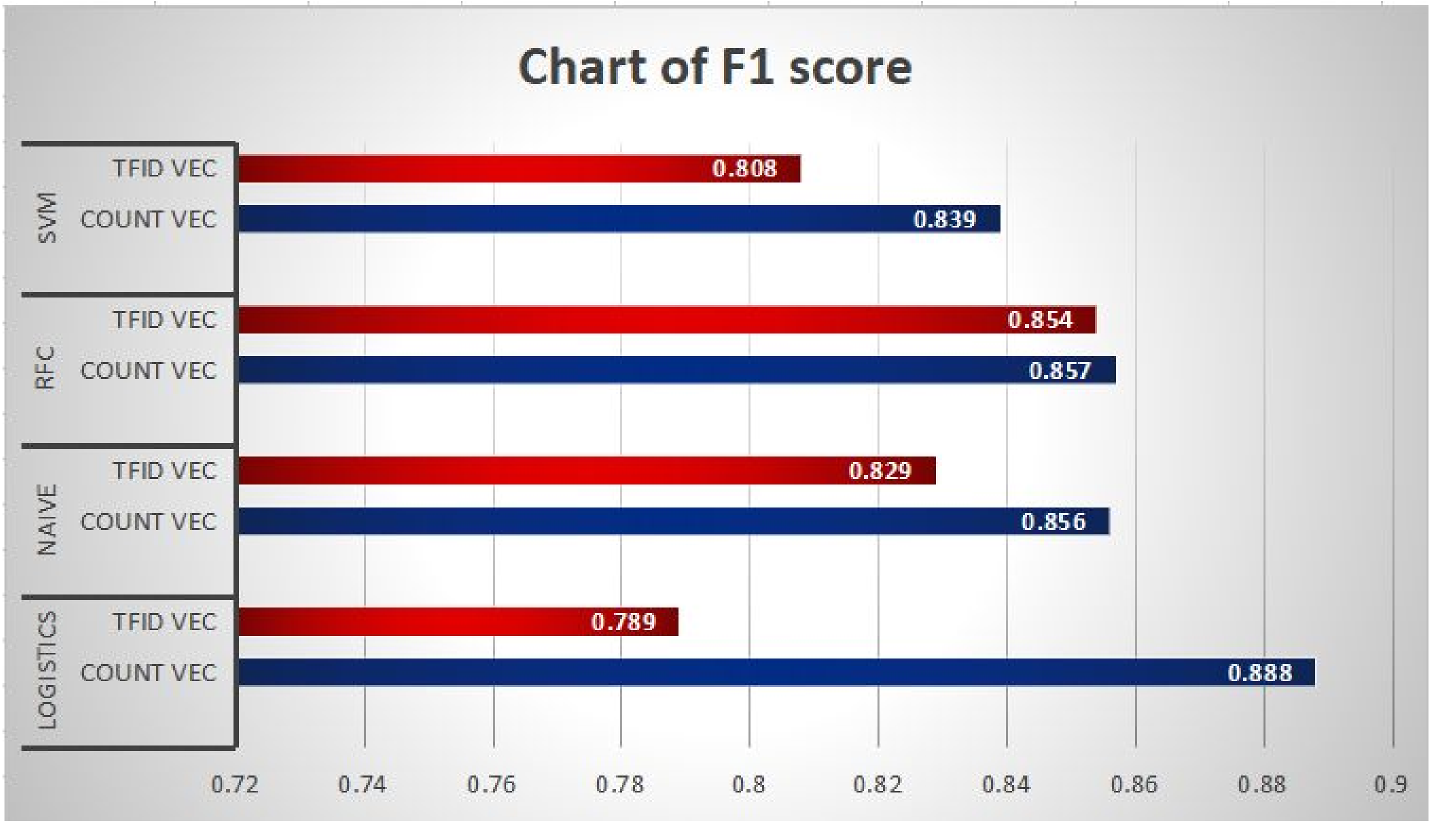


Figure 8

- **Experiment Two:** In this experiment, I did the same training except that I trained the models with **cleaned data**. This is to see

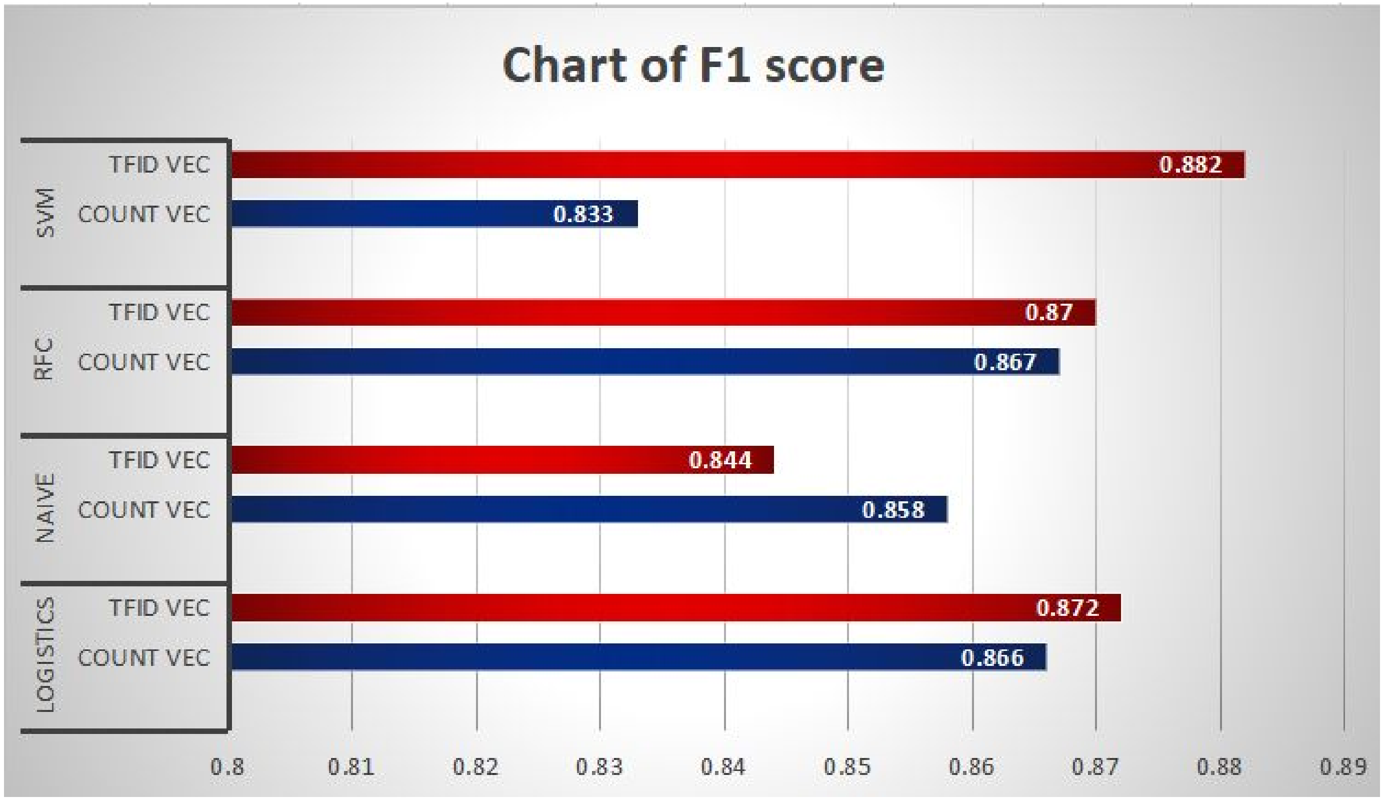


Figure 9

which model can perform best in the case where the data is not cleaned up. **SVM** with **TF-IFD vectorizer** outperformed the other models in generalizing over cleaned dataset. However, the score was 0.882 which is almost the same as the Logistic Regression score in Experiment one.

- **Experiment Three:** In order to select the best parameters from Logistic Regression and SVM which performs well in Experiement one and two, I performed a GridSearch, a process of performing hyper parameter tuning in order to determine the optimal values for a given model. In this case, SVM and Logistic Regression.

SVM : 0.904	LOGISTICS : 0.882

max_df = 0.5,
ngram_range =(1, 2))
C value = 1.
	max_df = 0.5,
	ngram_range =(1, 2))
	C value = 0.1

Table 1

Table 1 shows the parameters that GridSearch chooses as the best and also the F1 score of each classifier. As seen, SVM outperforms Logistic regression.

- **Experiment Four:** There are many approaches to our problem statement from statistical machine learning models (Logistic, Naive Bayes, SVM, etc.) to high-end deep learning models (CNN, RNN, Transformers, etc). Experiment four is a shift from Statistical models to exploring Deep learning models. The use of a Recurrent Neural Network architecture (precisely Bi-LSTM) was used in this experiment.

  These models take word embeddings as input. Word embeddings are basically a form of word representation that bridges the human understanding of language to that of a machine.

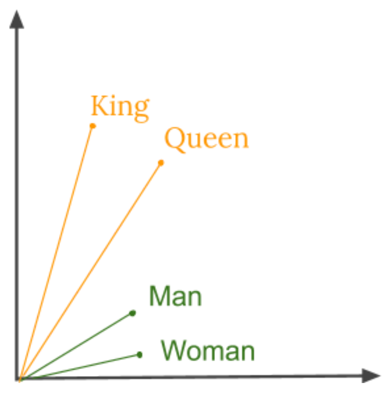


Figure 10

Embeddings have learned representations of text in an n-dimensional space where words that have the same meaning have a similar representation(Figure 10 shows this). I explored two experiments with the Embeddings representation by using our data to build a vocabulary for Embeddings and also using pre-trained GloVe embeddings to make the embedding dictionary.

- **Bi-LSTMS with Dataset vocabulary**: The Embeddings used in this experiment was obtained by building a vocabulary from all the words gotten from our dataset.
- **Bi-LSTMS with Glove Embeddings**: GloVe is an unsupervised learning algorithm for obtaining vector representations for words. This is achieved by mapping words into a meaningful space where the distance between words is related to semantic similarity.

The results of the two experiments using different Embeddings is shown in Table 2.

Bi-LSTMs with an embedding layer	Bi-LSTMs with GloVe vectors
0.936	**0.947**

Table 2

The incredible results from both approaches that are based on Deep learning shows how neural networks are powerful in text classification. And as we see, the better the representation of our Embeddings(like Glove), the more likely we have a better model performance.

Do note that all scores mentioned were obtained by the model performance on our test data which was not part of the dataset used during training.

The losses that were obtained during these two experiments were shown in the powerpoint presentation attached to this report.

4.0 CONCLUSION

It was an interesting challenge to perform different experiments with statistical and deep learning models and evaluate each of the classifiers.

- Statistical ML models perform well on sparse data representations while deep learning approaches on sparse data representations because they have no information of context semantics and sequence.
- RNNs perform greatly with sequential data especially when the data size is large. Hence, combining our dataset was a good call.
- Word embeddings are good for representing semantics and meaning, this is a fact why the deep learning models do well with this representation.

Overall, we can conclude that a deep learning approach is powerful for text classification.

5.0 REFERENCES

- **https://machinelearningmastery.com/classification-accuracy-is-not-enough-more-performance-measures-you-can-use/**
- **https://medium.com/@cmukesh8688/tf-idf-vectorizer-scikit-learn-dbc0244a911a**
- **https://en.wikipedia.org/wiki/GloVe_(machine_learning)**
- **https://www.mygreatlearning.com/blog/word-embedding/**
- **https://en.wikipedia.org/wiki/Fake_news**