“Exploring the effect of supervised learning algorithms for article classification on BBC news datasets”

Task 1: Exploratory Data Analytics

(a) Introduction to the Data

Within this dataset, there is a total of **428 BBC news articles** that are classified into two categories: **technology** and **entertainment**.

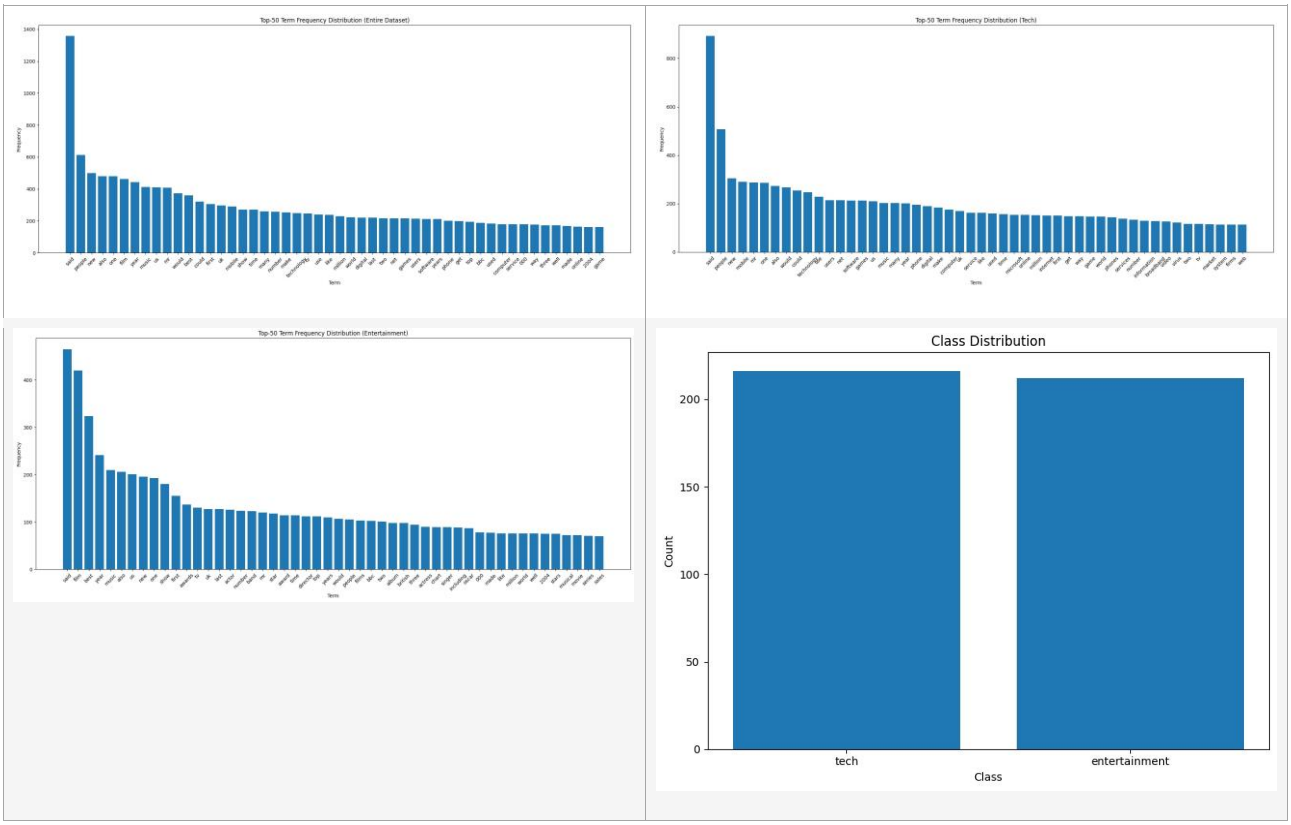
After analysis of each article, by means of vectorisation, the total number of **Extracted Features is: 13,518.**

In this report, we explore and compare the performance of four different models: Naïve Bayes (NB), k-Nearest Neighbours (kNN), Support Vector Machines (SVM), and Neural Networks (NN) - on our dataset.

(b) Frequency Analysis

We implemented frequency analysis to analyse the frequency of words within all articles in the dataset. We created 4 histograms to demonstrate the frequencies for the following:

1. top-50 term frequency distribution across the entire dataset
2. top-50 term frequency distribution across articles categorised as ‘Tech’
3. top-50 term frequency distribution across articles categorised as ‘Entertainment’
4. frequency distribution showing the number of articles for each category



Task 2: Exploratory Data Analytics

(a) Naive Bayes

After applying our Naive Bayes classifier we could identify the top-20 words that are most likely to occur in the articles over the two classes: **Tech** and **Entertainment**

Tech Top 20 = ['said' 'people' 'also' 'new' 'one' 'would' 'could' 'year' 'use' 'many' 'technology' 'make' 'way' 'mr' 'us' 'users' 'time' 'like' 'get' 'used']	**Entertainment Top 20** = ['said' 'year' 'also' 'one' 'film' 'us' 'new' 'first' 'last' 'best' 'show' 'two' 'time' 'star' 'tv' 'years' 'including' 'music' 'uk' 'would']

It was then in our interest to identify the top-20 words that have a high probability of appearing in Tech but not in Entertainment and Vice-Versa (**maximising the conditional probability**). Below are our results with the calculated probability next to each word, ordered in ascending fashion from the highest unique probability.

Tech Top 20 = ['users': **157.34**, 'software': **156.61**, 'microsoft': **113.43**, 'mobile': **106.47**, 'broadband': **94.40**, virus': **90.01**, 'firms': **83.42**, 'pc': **79.77**, 'spam': **62.20**, 'phones': **53.06**, 'gadget': **52.69**, 'net': **52.45**, 'consumer': **50.49**, 'mobiles': **49.76**, 'gadgets': **49.03**, 'machines': **49.03**, 'windows': **49.03**, 'technologies': **47.57**, 'systems': **46.84**, 'device': **46.10**]	Entertainment Top 20 = ['actress': **124.35**, 'singer': **122.98**, 'oscar': **120.25**, 'stars': **103.85**, 'aviator': **86.09**, 'band': **84.72**, 'nominated': **75.16**, 'festival': **72.42**, 'rock': **72.42**, 'album': **67.64**, 'nominations': 65.59, 'charles': **64.22**, 'chart': **61.49**, 'foxx': **60.12**, 'oscars': **58.76**, 'starring': **57.39**, 'singles': **51.93**, 'jamie': **47.83**, 'swank': **45.09**, 'comedy': **44.41**]

Summary:

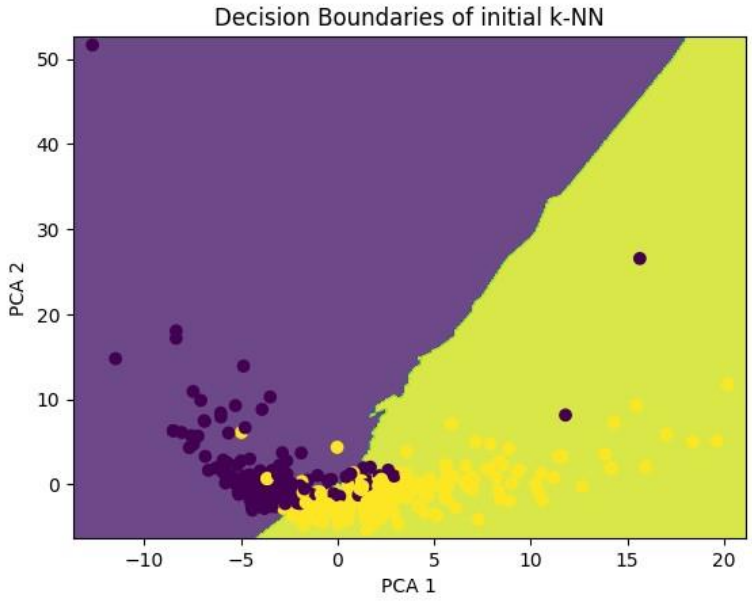
It appears that maximising the ratio of conditional probability gives us "better" words that reflect each class. The words that are given above show words that are very specific to either entertainment or tech and thus give a better reflection of what is likely to be a tech article vs what is likely to be an entertainment article.
This makes intuitive sense. The NB classifier Top 20 list gives us the top 20 words bsaed on the conditional probabilities of a word appearing in a class independent of it appearing in the other class. As such, words such as "said" and "people" crop up in both lists as they are common words and likely to appear in both tech and entertainment.
However, by maximising the conditional probability ratio as above, we get the top 20 words that are most likely to come up in 'tech' and least likely to come up in 'entertainment', thus giving us tokens that are more specific to the category

(b) kNN

Decision Boundaries of kNN

- For our initial kNN model we will estimate that a good hyper-parameter for k, is the general rule of thumb of using the square root of n where n is the number of instances of the training set.
- For the initial distance metric hyper-parameter, we will use the euclidean distance metric as it is one of most common calculations of distance by calculating the straight line distance between two points in the euclidian space.

Plot of kNN with hyper-parameters. *(See below)*



Impact of k and distance metric on decision boundary¶

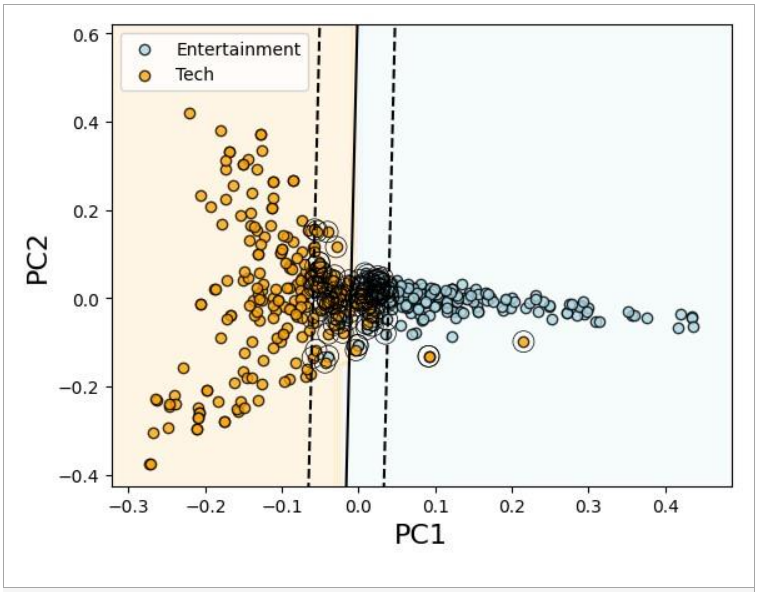
- The hyper-parameter k determines the number of neighbouring data points considered when making predictions. Smaller k means that only the closest neighbour is considered. If k is too small, it can be sensitive to noise points, which can lead to jagged decision boundaries.
- A larger k considers more neighbours, leading to more flexibility and a possibly more smoother decision boundary. However, if k is too large, it makes it more computationally expensive and more likely to include instances of different classes on the wrong side of the decision boundary.
- The distance metric hyper-parameter is the type of technique used to compute distance between instances. These differences between distance metrics for what is used for distance can affect the decision boundary.
- For example, Euclidian distance calculates straight-line distances between points, and decision boundary tends to be smooth, circular or spherical.
- Manhattan distance calculates distance by summing absolute differences between the points and so the decision boundary tends to be orthogonal to the co-ordinate axes creating a blocklike pattern.

The decision to use PCA (Principal Component Analysis) for kNN

- We used PCA as the reduction technique to reduce the training features to two features. This is because it is quite difficult to manually select two specific features from the set of words that would reflect the impact of the hyper-parameters value, as there are many words features to choose from.
- By reducing the training features to two principle components, we know that it will identify and select the two principle components that capture the maximum variance in the data, so would be effective in showing the impact of the hyper-parameters values.

(c) Support Vector Machines (SVM)

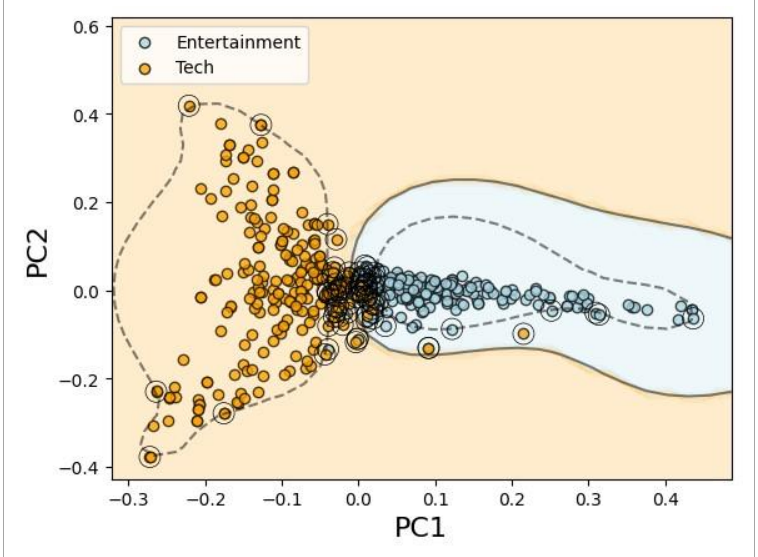
Soft-margin linear SVM plot *(see below)*



Choice of misclassification penalty **(C) = 10** in this SVM

A penalty of C = 10 was selected because the data points were very clustered. In this case, a higher value of C is optimal because it would result in a smallermargin hyperplane. If a low value of C, such as C = 1, was selected, the hyperplanes would be almost as large as the graph, resulting in a non-optimal training soft margin SVM for predictions

Hard-margin RBF kernel SVM plot *(see below)*



Choice of kernel width (σ) = Using scikit-learn's implementation of RBF SVM, we set Gamma = 'scale'. It is calculated as 1 / (n_features * X.var()), where X is the input data and n_features.

Selecting an optimal value for the gamma hyperparameter is crucial in an SVM with an RBF kernel, as it can significantly change the decision boundaries and hyperplanes. Using an optimal value for

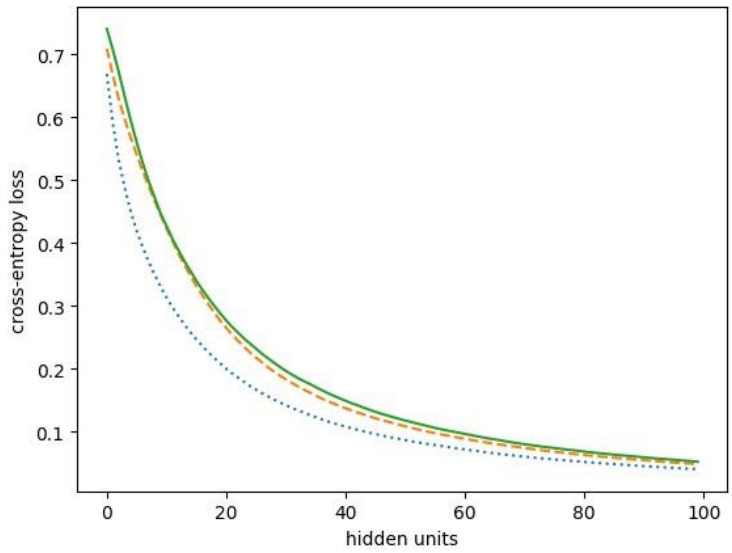
gamma is important to achieve good decision boundaries.

The decision to use PCA (Principal Component Analysis for SVM)

- In both models we use PCA dimensional reduction to plot in 2D space.
- We use dimensional reduction because our SVM data is highly dimensional.
- PCA is a popular choice for dimensionality reduction with soft-margin SVMs because it makes the assumption that the data is linearly separable, which is also an assumption made by linear SVMs.

(d) Neural Networks (NN)

The Average Training Cross-entropy Loss vs the number of hidden units plot *(see below)*



The presence of different hidden units has distinct effects on a model. When the number of hidden units is reduced, both the training error and generalisation error tend to be high due to under-fitting and the high statistical bias. Having a larger number of hidden units may lead to a lower training error; however, it can result in a higher generalisation error due to overfitting.

Task 3: Classification Quality Evaluation

(a) Testing Accuracy Over Varying Sizes of the Data

Naive Bayes

Train - model	Test - model
We see that with the training data, we only need around 30% of the training data size to get a perfect F-score. The F-score stats at a minimum when we only use 10% of the training data set. The f-score then quickly increases to approximately 1 when we increase the sample size to 30% and remains around 1 as the sample size grows.	Our Testing Dataset starts at a minimum Fscore as shown on the graph when we only use 10% of our training data. When using 30% of our training data, our testing F1-score increases to 0.975, which is slightly less than the testing F1-score for the training data with the same proportion. After 30%, our training F1 and our testing F1 essentially converge around the F1=1 mark.

kNN

Train - model:	Test - model:

For the training data, initially F1-score increases as the proportion of data used for training also increases. It reaches its peak F1Score when using around 30% of the data for training. After this there is a steep decline in the F1-score as a higher proportion of data is used for training. This continues up until when we use around 50% of the training data, after which F1-score will start to increase again, but still not as high as when we used 30% of the training data.	For the testing data, the F1-score also initially increases as the proportion of data used for training increases. However, this increase continues up until using 50% of the data where it reaches a higher peak F1-score than the training data. Similarly to the training data, after this peak, there is a steep decline in the F1-score as more data is used for training. This continues up until 70% of the data is used for training, after which F1-score will stay the same.

SVM - Hard-margin RBF

Train - model:	Test - model:
Its common that when the training set is small that the classifier will easily fit the training set which is why we are getting 100% accuracy at the training set of 0.1. When training set size increases the training set becomes harder to fit which slightly decreases the training accuracy. This explains Training set size of 0.1 is 100% Training set size of 0.3 is 98% Training set size of 0.5 – 0.9 is 96%	When the training set is small the data easily fits the training data which causes overfitting. So when we run our SVM on unseen data like our testing set we get very bad F1 performance. However an increase in the training set size can help the SVM to capture generalized patterns this reduces our overfitting problem giving us better F1 performance this explains Training set size of 0.1 is 90% Training set size of 0.3 – 0.9 is 98% We can see with our training set size of 0.3 we can get optimal results

NN - Neural Networks

Train - model:	Test - model:
For the training model, the Fl-score starts very high and stays at the same F1-score until around 50% where the F1-score has a slight drop off. The drop-off is minor though and the F1-score will finally slightly increase when the whole training data set is used.	For the testing model, the F1-score starts off at a very low score but increases rapidly with the training dataset's size. The rapid increase in the f1-score will stop at 30% and will decrease gradually until 50% where the F1-score doesn't change. The f1-score of the testing model was always be lower then the training model.

(b) 5-fold Cross-validation Analysis

Classifier	**Hyper-parameter**	**how these hyperparameters impact on the testing accuracy.**
NB	Laplace Smoothing Paramater a.	When looking at 5 fold cross validation, we found that a = 0 had the worst accuracy (0.49). This intuitively makes sense as it suggests that some words have 0 probability of occurring when they actually might not appear specifically in the sample. The optimal accuracy we found was when a = 1 (default value) with accuracy of 0.957. We tested a = 2, 3, 4, 5 and found that the accuracy decreased as we increased a.
kNN	k, distance metric	Each distance metric tested had a difference performance on the testing accuracy, where some distance metrics performed poor on certain k, but very well for others. In particular the cosine distance metric had one of the worse accuracy for very small k, but performed the best if the k chosen was not too small or not too large. For each of the distance metrics, having too small of a k and also too high of k both decreased the testing accuracy.

SVM - Hardmargin RBF	C penalty, Gamma	We can see when the C penalty is low or the gamma values are low that our 5 fold cross validation accuracy is at 57-60% accuracy However when our of our C value or high gamma values we receive 5 fold cross validation accuracy is at 92-98% accuracy We see that on our testing data that our hyperparameters do better when our C and gamma values are high
NN	learning_rate_init	It impacts the testing accuracy because when trying to find the loss function, the learning rate determines the step size at each repetition.

(c) Final Model for Each Classifier

Classifier	Best Hyper-parameter	F1 Score - Based on the testing Dataset
NB	Alpha (Smoothing Parameter) = 1	1.0000
kNN	K=7, distance_metric = cosine	0.841121495327103
SVM - Hardmargin RBF	C: 10.0, gamma: 100.0 C: 100.0, gamma: 10.0	0.981132075471698
NN	learning_rate_init = 0.01	0.989010989010989

To summarise, all the classifiers show a high level of classification accuracy on the testing dataset for BBC News Articles with the exception of kNN being a little bit lower than the rest. The Naive Bayes classifier was the best performer achieving the highest F1 score with perfect accuracy at **100%**. It was followed closely by the Neural Network classifier with a score of **98.9%** and the SVM classifier with a score of **98.11%**, while the kNN classifiers performed lower with a score of **84.11%**