Task 1

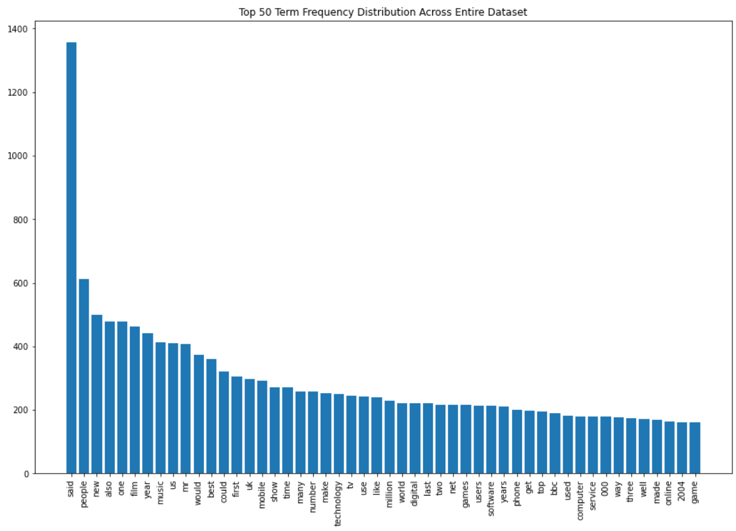
Part A

Feature vector for the first 5 articles:

	Features	Article 1	Article 2	Article 3	Article 4	Article 5	Total
0	000	0	0	2	0	0	2
1	05	0	0	0	0	3	3
2	06	0	0	0	0	3	3
3	10	0	1	0	0	0	1
4	10th	0	0	1	0	0	1
...	...	...	...	...	...	...	...
704	worldwide	0	1	0	0	0	1
705	would	1	1	0	1	0	3
706	ya	0	1	0	0	0	1
707	year	2	3	2	0	2	9
708	years	0	3	0	0	1	4

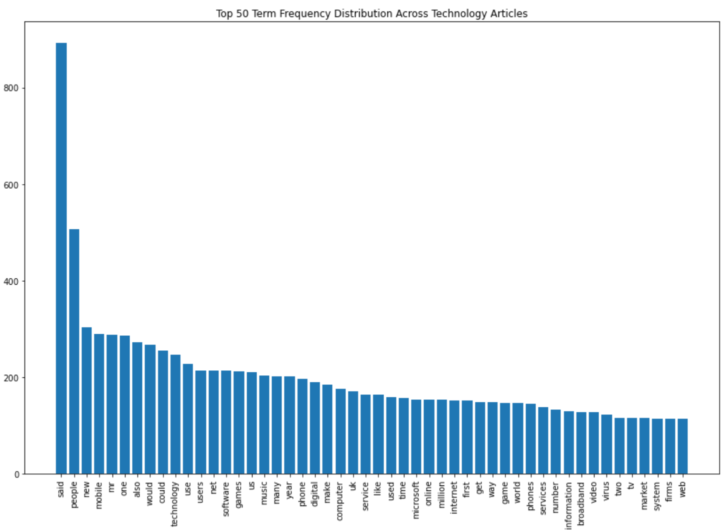
Part B

Top-50 term frequency distribution for entire dataset:

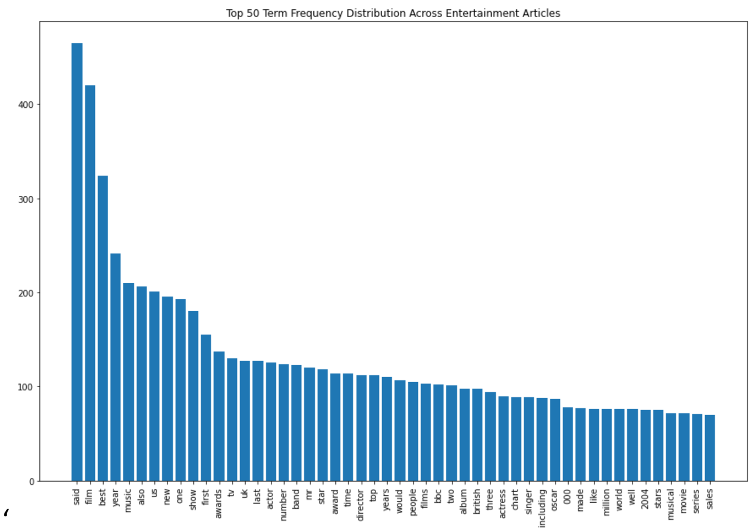


Split by article category

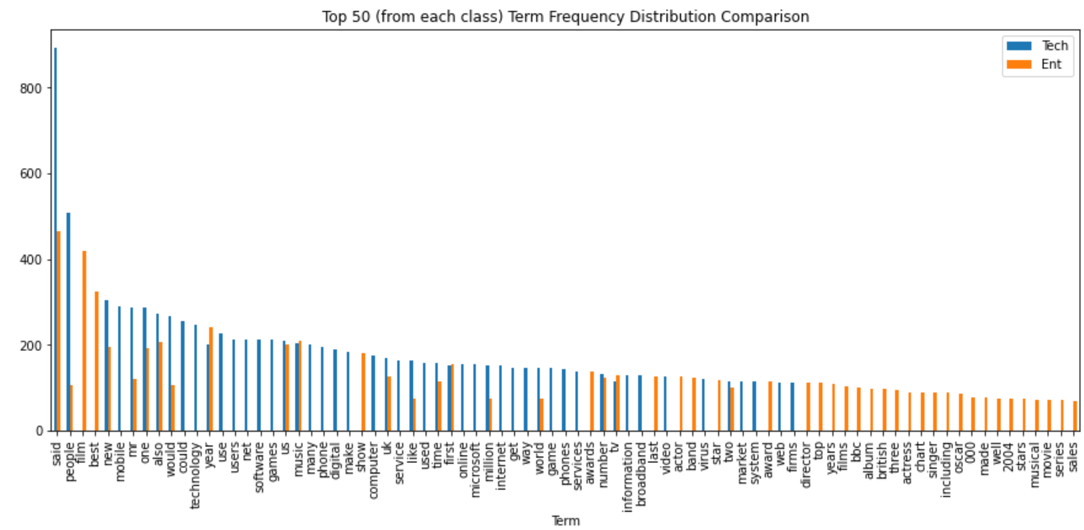
Technology:



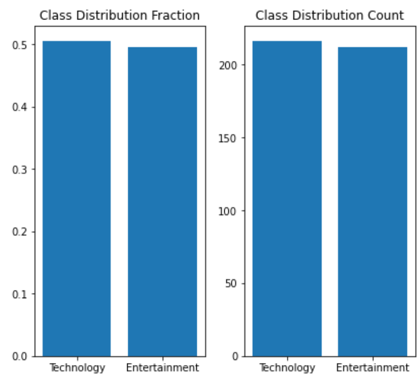
Entertainment:



Combined:



Class Distribution:



Task 2

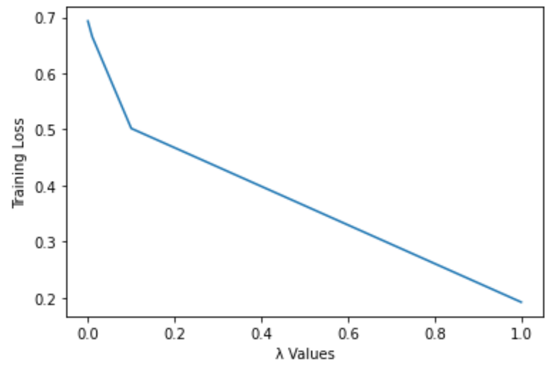
Logistic Regression

Regularization is the technique used to reduce error by fitting a function appropriately on the given training set and to avoid overfitting by controlling model complexity. L2-Regularization adds a regularization term to the loss function so it can prevent overfitting by penalizing larger parameters in favour of smaller parameters.

The effect of the regularization parameter λ on the outcome in terms of bias and variance is that as the lambda parameter increases, training error increases. Regularization forces weights towards 0 which causes the variance to decrease, but as we are allowing less flexibility, the model moves away from the true values, thus slightly increasing bias.

The plot shows the inverse of this as the C parameter in the LogisticRegression class is the inverse of the hyperparameter λ. Smaller values specify stronger regularization.

If λ is too high, the model becomes too simple and tends to underfit. If λ is too low, the effect of regularization becomes negligible, and the model is likely to overfit. If λ is 0, then regularization is completely removed and runs a high risk of overfitting.



Training loss (log loss) vs lambda values

Naïve Bayes

(i) Top 20 identifiable words, split by category:

Top 20 Tech:	
said	892
people	507
new	304
mobile	290
mr	288
one	286
also	273
would	267
could	255
technology	247
use	228
users	214
net	214
software	213
games	212
us	210
music	203
many	202
year	201
phone	196

Top 20 Entertainment:	
said	465
film	420
best	324
year	241
music	210
also	206
us	201
new	196
one	193
show	180
first	155
awards	137
tv	130
last	127
uk	127
actor	126
number	124
band	123
mr	120
star	118

(ii) Top 20 words, maximising $P(Xw=1|Y=y)/P(Xw=1|Y\neq y)$:

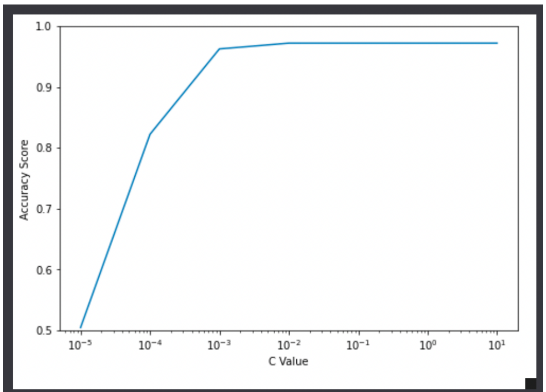
Top 20 Tech:	
users	107.500000
software	107.000000
mobile	97.000000
microsoft	77.500000
broadband	64.500000
virus	61.500000
firms	57.000000
pc	54.500000
net	53.750000
technology	49.600000
phones	48.333333
spam	42.500000
gadget	36.000000
games	35.500000
consumer	34.500000
mobiles	34.000000
gadgets	33.500000
windows	33.500000
machines	33.500000
phone	32.833333

Top 20 Entertainment:	
actress	45.500000
singer	45.000000
oscar	44.000000
band	41.333333
stars	38.000000
album	33.000000
aviator	31.500000
chart	30.000000
nominated	27.500000
rock	26.500000
festival	26.500000
actor	25.400000
nominations	24.000000
charles	23.500000
foxx	22.000000
comedy	21.666667
oscars	21.500000
starring	21.000000
singles	19.000000
musical	18.250000

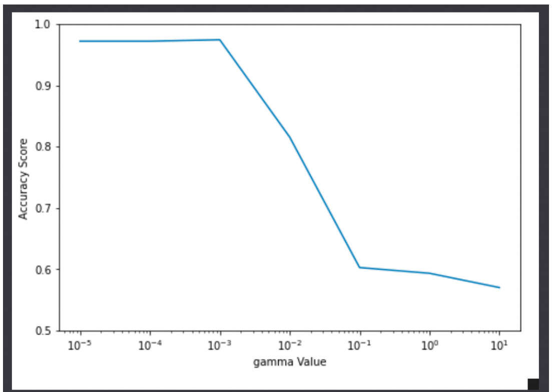
The second list of words describes the two classes better. The top 20 words for each class in (ii) look to be more relevant than the top 20 words for each class in (i).

SVM

For both soft-margin SVM and hard-margin RBF kernel, we tested to find the best C-value and gamma respectively:



The best C-Value for a soft-margin SVM is 0.01, with an accuracy of 0.9719822812846068.



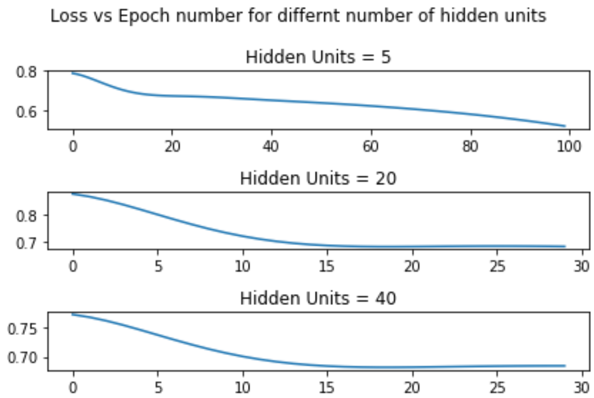
The best gamma-Value for a hard-margin SVM is 0.001, with an accuracy of 0.9743078626799557.

Soft-Margin Linear SVM: C = 10^-2 & accuracy = 0.9719

What the c-value does is tell the SVM how much slack we are allowing it when drawing its margins. The lower the value the more misclassifications we allow, in other words the more data points we allow to be on the wrong side of the margin. The higher the value the less slack we give the SVM to allow misclassifications.

Hard-Margin RBF Kernel: gamma = 10^-3 & accuracy = 0.9743
The gamma value can be thought of setting the 'spread' of the kernel, in other words deciding how much 'curve' to allow the decision boundaries. The lower the gamma value the decision boundaries will appear straighter. And with a high gamma value we are allowing the decision boundaries to curve around the data points more concisely.

Neural Networks



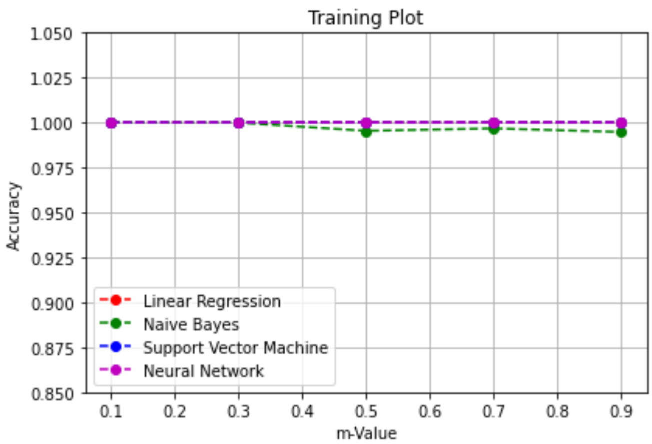
As we increased the number of hidden units, the minimum achieved loss increased. This is likely due to overfit as we have too many hidden units

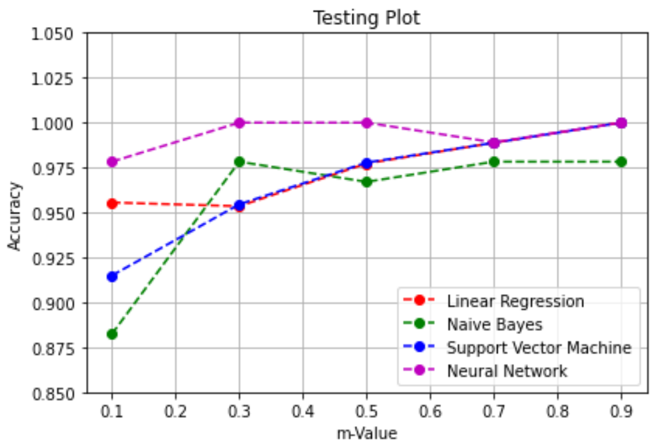
With 5 hidden units, we had a minimum loss of 0.5195
With 20 hidden units, we had a minimum loss of 0.6793
With 40 hidden units, we had a minimum loss of 0.6819

Task 3

Part A

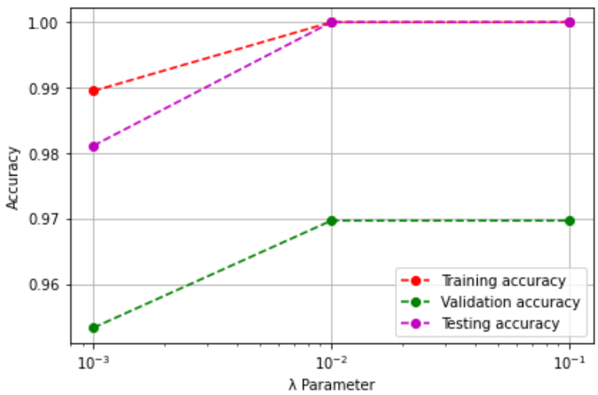
Comparing the training accuracy and testing accuracy for different values of m:



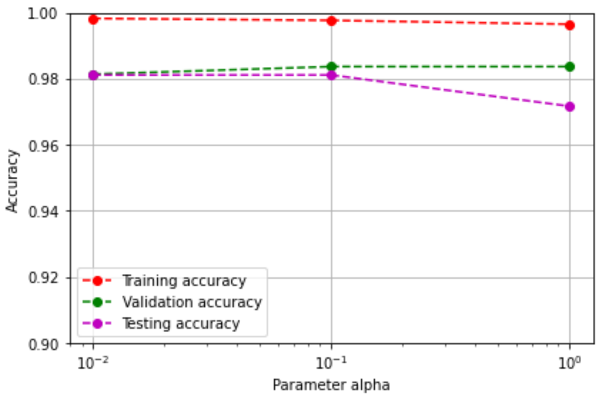


Part B

Logistic Regression accuracy vs C-value:



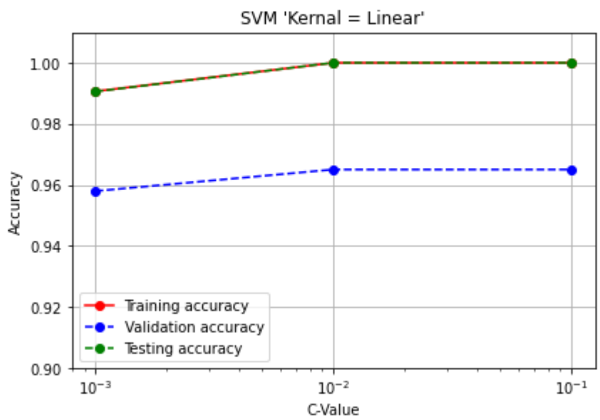
Naïve Bayes accuracy vs alpha parameter:



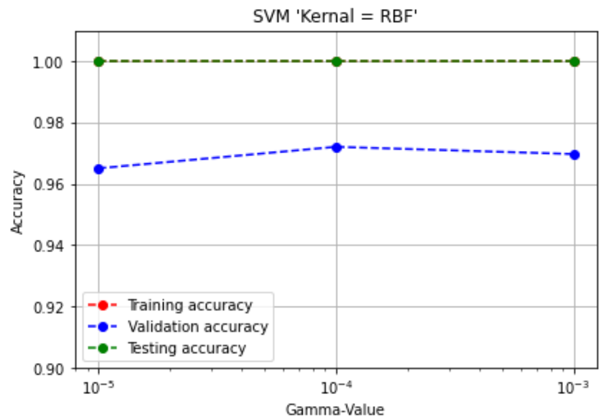
A C-value of 0.01 is ideal. Any lower and we see underfitting, any higher and it could lead to a lack of generalisation - although we don't see this in the given dataset.

0.1 is the ideal alpha, since at 1 we can see some signs of underfit (falling training accuracy) and at 0.01 we can see some signs of overfit (validation accuracy reducing)

SVM Linear accuracy vs C-value

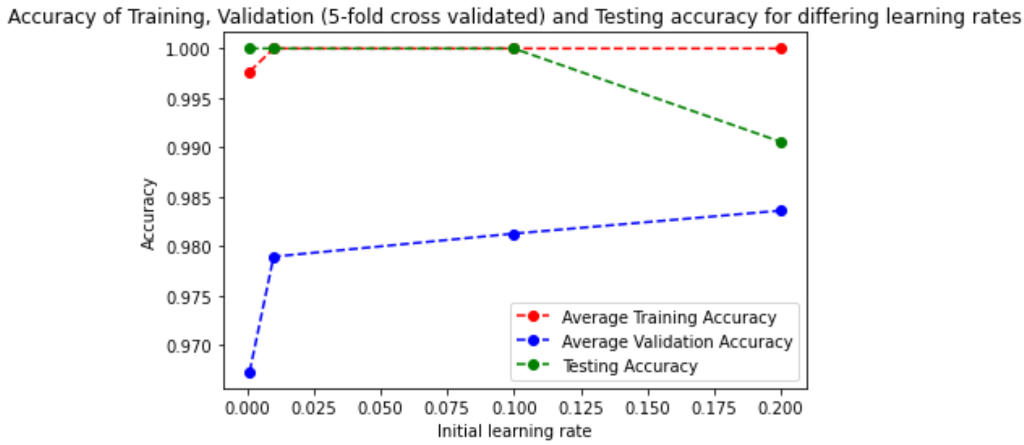


SVM RBF Kernel accuracy vs gamma-value:



For the linear SVM, a C-Value of 0.01 provides the strongest regulation without any underfit. For the RBF SVM, the only hyperparameter was the gamma value, since the C-value is set to 10^10, given that it is hard-margin. A gamma value of 10^-4 appears to provide the best balance of fit.

NN accuracy vs learning rate:



The learning rate of 0.1 appears to be the best in this scenario, as it provides both a high validation and testing accuracy.

Part C

With the chosen hyperparameters, the logistic regression, support vector machine and neural network all work perfectly on the test data. Naive Bayes is close behind with a near perfect score. Based on what we have, any of these models could be considered good enough for our purposes.

```
LR F1 score: 1.0
NB F1 score: 0.9833333333333333
SVM F1 score: 1.0
NN F1 score: 1.0
```

We would likely need more data for testing if we wanted to separate these models in terms of accuracy.