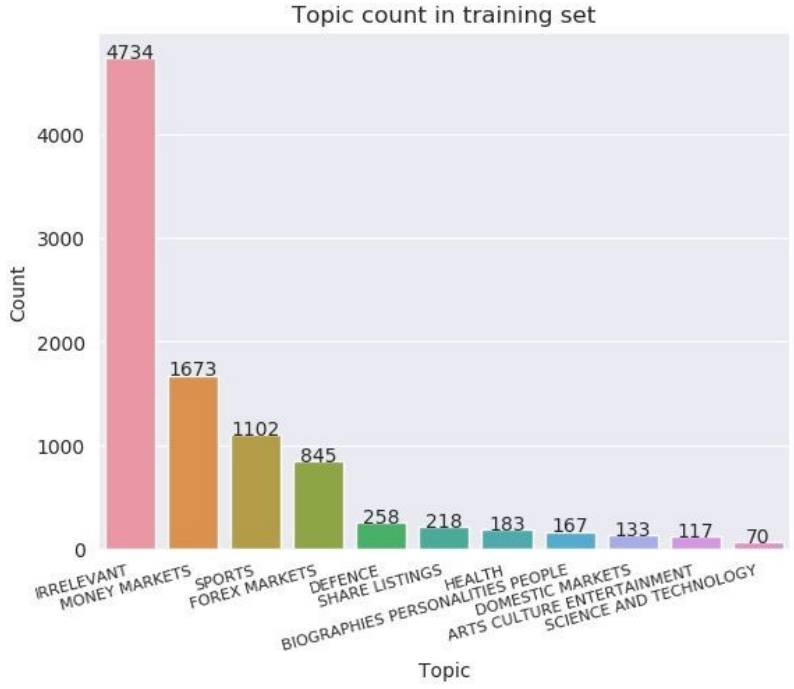
Introduction

A multi-class text classifier is required to analyse a large set of news articles and label them by topic; the classifier must then, if possible, recommend a maximum of 10 news articles per topic. The aim of this project is to record and explain the reasoning of the methodologies used to develop the proposed model. Support-vector Machines and Multinomial Naive Bayes implementations are used to create distinct models, and the best performing will be chosen as the final classifier. Issues faced in producing an optimal model include choosing optimal hyper-parameters, selecting the most relevant features, and selecting performance metrics that objectively rank model performance whilst considering the possibility of sample bias. Methods to resolve these issues involve using machine learning optimisation methods such as k-fold cross validation, comparing results of different features, and analysing criteria dependent on classifying power such as recall, precision, and f1-score.

Exploratory Data Analysis

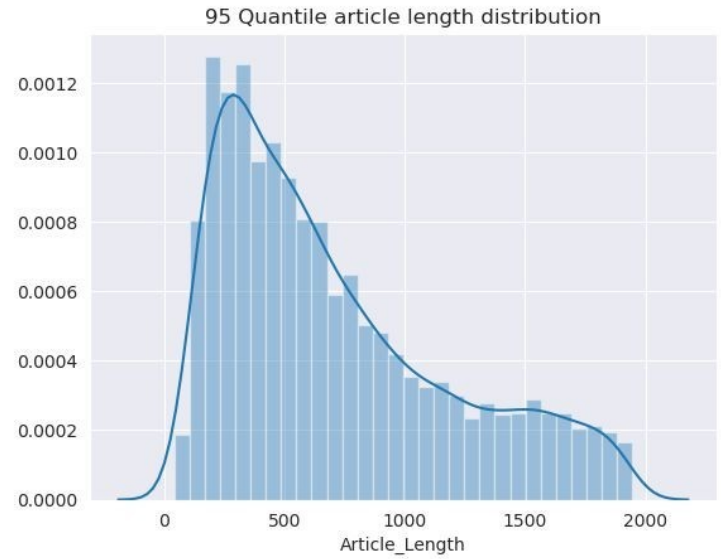
The number of articles per topic within the training set was counted to produce the bar graph below



.

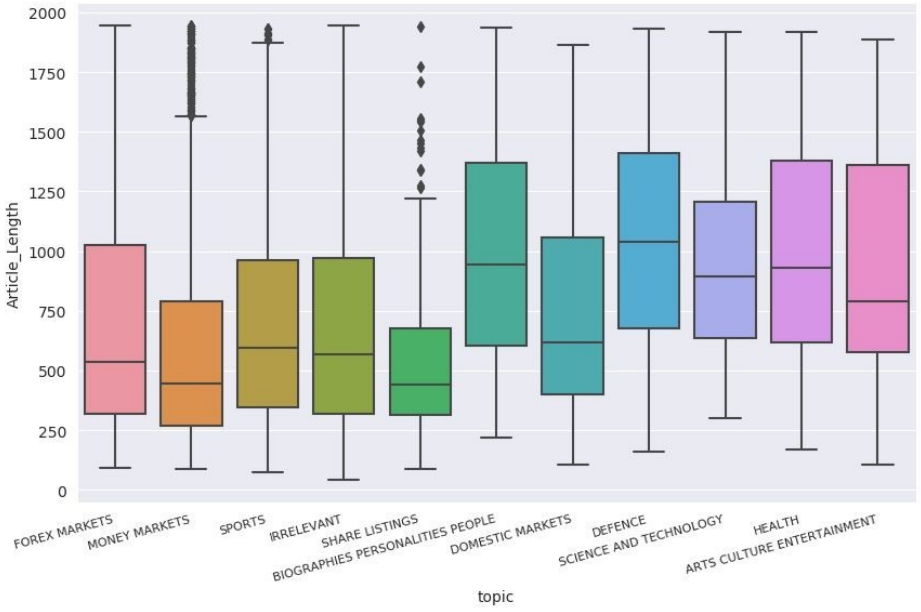
It can be seen that the vast majority of articles are “IRRELEVANT,” hence this topic will be used as the subject class for the Majority Topic classifier, which is the chosen method for the baseline model. There is a definitive imbalance within the dataset, where the majority of the topics given are far fewer proportionally to the 4 most numerous topics (IRRELEVANT, MONEY MARKETS, SPORTS, FOREX MARKETS). It is clear that the scoring metric chosen to evaluate model performances needs to deeply consider this aspect.

The probability distribution of article length was also plotted; only the results for the 95th quantile of the dataset is shown due to the outliers extending the graph more than necessary.



.

Further analysis of the article length distribution per topic has been conducted, culminating in the box plots below.



Share listing has the lowest mean, and the topics that are fewer in number tend to be longer. Perhaps this information could be used to improve classifying power for these less frequent topics.

Methodology

Majority Topic Classifier (Baseline Classifier)

The baseline classifier simply predicts all observations in the test set as the majority class observed in the training set irrespective of features; this type of classifier serves as the “baseline” to determine

whether or not our models are useful. The most numerous class observed during the exploratory data analysis is “IRRELEVANT.” Here the prediction vectors, for both the test and training set, is simply “IRRELEVANT” for all entries, with their respective sizes equal to the number of observations for their corresponding test/training sets.

There is no need for hyperparameter tuning or feature engineering in this method. The evaluation metrics used to evaluate the baseline’s performance are precision, recall, F1, and accuracy. These metrics are used in order to evaluate classifying power and compare with accuracy.

Multinomial Naive Bayes Implementation

Feature selection

TD-IDF method to all words with lemmatization. The TD-IDF value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general. And Lemmatisation (or lemmatization) in linguistics is the process of grouping together the inflected forms of a word so they can be analysed as a single item, identified by the word's lemma, or dictionary form.

Data processing

label coding: convert 11 text classifications into numbers from 0 to 10, with 0-9 being the 10 topics in alphabetical order and the number 10 to represent ‘IRRELEVANT.” The encoding table can be seen in appendix figure 1.1.

Method and reasons

All the words are counted as features using TD-IDF method with lemmatization, which are trained with the whole training set. with cross validation. TD-IDF can be implemented by TfidfVectorizer() of sklearn.feature_extraction.text, and lemmatization can be achieved by lemma() from pattern.en.Actually I tried with and without lemmatization and the results of the following parts are exactly the same. But the feature number reduces from 35822 to 35290.

To prevent overfitting, we should apply cross validation method. Multinomial Bayes function of sklearn has the parameter alpha as a smoothing hyperparameter, we should use cross validation to find optimal alpha. Here we use GridSearchCV() method of sklearn to efficiently find out the best alpha in range of 0.0-10.0 with a step of 0.01. We set the cross-validation method as 10-fold cross-validation and use the scoring method of f1-macro. The result of finding the best alpha for this training set is 0.01.

The specific code implementation uses the multinomialnb() in the sklearn package to fit the training set, and then forecasts the test set. The prediction accuracy after cross validation for the training set is 89.3% and for the test set is 76.6%.

SVM Implementation

Feature Selection

The Bag of words text representation was chosen for its simplicity. Both word count vectors and TF-IDF vectors were implemented due to different initial experimental observations with SVM methods. Article words were simply converted into the desired feature space using the TF-IDF and word count vectorizer methods when needed.

Hyper parameter tuning

SVM Models

Sci-kit's implementation of SVM, SVC(), allows us to choose between linear, polynomial, and rbf kernels. Each kernel gives rise to a different hyper parameter space to search in. Consequently, we have chosen to optimise the parameters, C - complexity (for all three), gamma (for polynomial and rbf), and degree (for polynomial); these parameters are imperative for determining whether or not the model over-fits or under-fits. Unfortunately, we had to limit the gamma and degree parameter values due to the extended run times resulting from those parameters having high values. The SVC() classifier was chosen with the class_weight parameter equal to "balanced." Allowing for this parameter minimises the effect of large data imbalance seen.

The LinearSVC method provided by sci-kit was also implemented. The main limitation of this method was that it can only use the linear kernel. This model was implemented because it utilised a one-vs-rest SVM approach to classification, which was different to the one-vs-one SVM approach implemented by the default SVC method. Like the SVC() method, the class_weight parameter was set to "balanced" to minimise the effects of data imbalance.

In addition, an optimal hyper parameter configuration will depend on the features chosen. Since two sets of features were considered, two separate sets of tuning methods, with identical processes, were implemented.

SVC() Implementation

Randomised Search - SVC()

Since the number of hyper parameter combinations are numerous, the first step in determining the optimal hyper parameters values is to limit the parameter space; a randomised search was used to do this. Randomised search is ideal for searching a large number of configurations, and utilising this search with detailed verbosity during run time allowed us to visually confirm which hyper parameters are contributing to high scores. The results from randomised search help us determine the grid search.

The scoring methods used (the criteria in which the randomised search would recommend us it's best model) was f1-macro. Precision allows us to make accurate recommendations however if there is a huge imbalance in data (like in both the sample training and test samples given), the model may miss predicting certain models completely, and so f1 was considered to give recall substantial weighing. The macro method of averaging was chosen due to the large imbalance in the data set. Micro would give us a false indication of useful classifying power of the model since the “IRRELEVANT” class would dominate the metrics.

The randomised method was implemented using a 3 fold fit cross validation for each randomly selected configuration; using a smaller number of folds decreased run time but compromised the reliability of the final model, but this is justified since we only needed to narrow in on the hyper parameter ranges. 50 random configurations were selected for the parameter space, totalling a number of 150 fits. In addition a specific random seed was given so that the models would sample from the data in the same order, minimising selection bias and giving more consistent results.

The randomised grid was defined as shown:
{‘C’: [0.0001, 0.001, 0.01, 0.1, 1], ‘Gamma’:[0.0001, 0.001, 0.01, 0.1, 1], ‘kernel’:[‘linear’, ‘rbf’, ‘poly’], ‘Degree’:[1,2,3]}

Using randomised search on the word vector feature space shows that a polynomial kernel model with parameters {‘gamma’:0.01, ‘degree’1, ‘C’,0.1} was best. Using randomised search on the TF-IDF feature space yields a linear kernel model with parameters {‘C’: 1}. Polynomial and Linear kernels were thus chosen to be explored using grid search.

Grid search - SVC()

It was seen through visual inspection of the score during run time, that the rbf kernel method generally provided extremely low scores, and so it was removed from this method of search. Much of the search settings for the grid search was similar to the random search (probability=false, 3-fold CV,

scoring methods) for the same reasons as above. It was decided to use a low number of k folds in this search method as well, as the grid search chosen had an extremely large parameter space; here we also define the maximum number of iterations to see whether generalisation can be improved, adding this search criterion lead to an immense number of model fits: 225. It is important to note that a max_iter value of -1 simply means iterate until the algorithm thinks it is good enough

The search grid was defined as shown:
C = [0.001, 0.01, 0.1, 1, 10], max_iter = [1000, 5000, -1], gamma = [0.1, 1], degree = [1, 2]
Grid Search = [{'kernel':['poly'], 'C':C, 'degree':degree, 'probability':probability, 'gamma':gamma, 'max_iter':max_iter},{'kernel':['linear'], 'C':C, 'probability':probability, 'max_iter':max_iter}]

The best parameters recommended for the word count vector and TF-IDF vector were:
{'C': 0.01, 'degree': 1, 'gamma': 1, 'kernel': 'poly', 'max_iter': 5000, 'probability': False},
{'C': 1, 'degree': 1, 'gamma': 1, 'kernel': 'poly', 'max_iter': 5000, 'probability': False} respectively.

A final grid search centered around the polynomial function was performed to search for a more optimal maximum iteration count, and increase the number of k folds. This increase in the number of folds for cross validation was also performed to compare the cross validation score with the next implementation of SVM.

The final grid search for the SVC method is shown below:
{‘C’:1, ‘degree’:1, ‘gamma’:1, ‘max_iter’ = [4000, 5000, 6000]} for TF-IDF and
{‘C’:0.01, ‘degree’:1, ‘gamma’:1, ‘max_iter’ = [4000, 5000, 6000]} for Word Vector and
Leading to the final hyper parameters:
{'C': 1, 'degree': 1, 'gamma': 1, 'max_iter': 4000} for TF-IDF and
{'C': 0.01, 'degree': 1, 'gamma': 1, 'max_iter': 4000} for Word Vector

LinearSVC() Implementation

There was no need to use an initial random search since the LinearSVC had only the parameters C-complexity, and maximum number of iterations to configure. In addition, the implementation was remarkably fast, allowing us to be generous with the range of values we want to explore. This allowed us to perform an exhaustive search on the possible hyper parameter values.

The hyperparameter search space was limited to.
{‘C’:[0.001, 0.01, 0.1, 1, 10], ‘‘max_iter’ = [1000, 5000, 10000]}
Resulting in the recommended parameters:
{'C': 0.1, 'max_iter': 1000} for TF-IDF and
{'C': 0.001, 'max_iter': 1000} for word vector

The model chosen for the final test: The LinearSVC() Implementation of SVM

In order to produce reliable recommendations, algorithm confidence was considered. The top 10 articles with the highest algorithmic probability for each topic were recommended. If there were less than 10 articles predicted for a particular topic, then all the predicted articles for that topic were recommended.

Results

Cross validation results on training set

Majority Topic Classifier (Baseline Model) Results

Method	Function Parameters	Feature Space	f1-macro on training
Majority topic	-	-	0.06

It is clear that this extremely simple model performed extremely poorly with respect to the scoring metric chosen.

Multinomial Naive Bayes Cross Validation Results

Method	Function Parameters	Feature Space	f1-macro score on 10 fold CV
MultinomialNB()	{'alpha': 0.01}	TF-IDF	0.588

For MNB model, there are three parameters. The parameter we need to tune is alpha, the other two parameters are fit_prior(Whether to learn class prior probabilities or not. If false, a uniform prior will be used) and class_prior(Prior probabilities of the classes. If specified the priors are not adjusted according to the data), which we leave them as default because it would certainly be better.

SVM Cross Validation Results

Method	Function Parameters	Feature Space	f1-macro score on 5 fold CV
SVC()	{kernel='poly', C=1, degree=1, gamma=1, max_iter=4000, class_weight='balanced'}	TF-IDF	0.665
SVC()	{kernel='poly', C=0.01, degree=1, gamma=1, max_iter=4000, class_weight='balanced'}	Word Count	0.647
LinearSVC()	{C=0.1, max_iter=1000,	TF-IDF	0.672

	class_weight='balanced'}		
LinearSVC()	{C=0.001, max_iter=1000, class_weight='balanced'}	Word Count	0.663

The score from a 5-fold cross validation and the results on the test set were analysed, showing that models utilising the TF-IDF feature space were superior to their word count vector counterparts for this method of scoring, highlighting the importance of proper feature space selection.

The optimised model using the LinearSVC() method performed marginally better than the optimised model using the regular SVC() method, as shown through comparing their respective cross validation scores and results on the test set.

Chosen Final Model Results

Final model information

Method function: SVM.LinearSVC(), from sklearn

Final model parameters: {C=0.1, max_iter=1000, class_weight='balanced'}

A 5 fold cross validation with f1-macro scoring produced a score of 0.672 on the training set and 0.64 on the test set.

Test set classification metrics

Classification Report:

	precision	recall	f1-score	support
ARTS CULTURE ENTERTAINMENT	0.29	0.67	0.40	3
BIOGRAPHIES PERSONALITIES PEOPLE	0.58	0.47	0.52	15
DEFENCE	0.71	0.92	0.80	13
DOMESTIC MARKETS	0.25	1.00	0.40	2
FOREX MARKETS	0.56	0.75	0.64	48
HEALTH	0.73	0.79	0.76	14
IRRELEVANT	0.95	0.85	0.90	266
MONEY MARKETS	0.63	0.61	0.62	69
SCIENCE AND TECHNOLOGY	0.33	0.33	0.33	3
SHARE LISTINGS	0.67	0.86	0.75	7
SPORTS	0.95	0.98	0.97	60
accuracy			0.81	500
macro avg	0.60	0.75	0.64	500
weighted avg	0.83	0.81	0.81	500

Test set results evaluation

The topics that had fewer test samples had poorer classification scores than the topics that had a greater number of test samples. Larger test sets for these few topics would be needed to gain a better understanding of the true classifying power of the model. This discrepancy is likely contributed by the data imbalance during training.

Final Model's Article Recommendations

Topic name	Suggested articles	Precision	Recall	F1
ARTS CULTURE ENTERTAINMENT	9952, 9703, 9789, 9830, 9933, 9526, 9604	0.29	0.67	0.40
BIOGRAPHIES,PERSONALITIES PEOPLE	9940, 9988, 9878, 9758, 9582, 9896, 9854, 9575, 9645, 9669	0.60	0.40	0.48
DEFENCE	9559, 9576, 9616, 9773, 9842, 9987, 9770, 9670, 9607, 9759	0.80	0.62	0.70
DOMESTIC MARKETS	9994, 9640, 9989, 9750, 9753, 9519, 9796, 9910	0.25	1.00	0.4
FOREX MARKETS	9551, 9986, 9875, 9588, 9682, 9977, 9530, 9772, 9693, 9625	0.50	0.10	0.17
HEALTH	9873, 9661, 9937, 9929, 9833, 9807, 9810, 9621, 9609, 9911	0.70	0.50	0.58
MONEY MARKETS	9998, 9755, 9618, 9871, 9835, 9516, 9765, 9769, 9602, 9761	0.80	0.12	0.20
SCIENCE AND TECHNOLOGY	9617, 9982, 9722	0.33	0.33	0.33
SHARE LISTINGS	9518, 9715, 9601, 9666, 9972, 9562, 9667, 9999, 9668	0.67	0.86	0.75
SPORTS	9857, 9574, 9569, 9760, 9568, 9922, 9848, 9596, 9886, 9787	1.00	0.17	0.29

Consistent with the trends seen with the test set evaluation, the larger test samples generally have higher precision. Analysis of recall and f1 are superfluous in this context because of the hard limitation of 10 article recommendations, treating all other correctly classified articles outside the 10 recommended as false negatives.

Discussion

Compare different methods, their features and their performance. State any general observations.

It is important to note that the baseline classifier performed poorly with the chosen metric (f1-macro), however it boasts a relatively high accuracy on the test set, as shown in appendices figure 1.2, highlighting that accuracy and weighted accuracy may be misleading metrics to assess model performance on. The results of all the new tested methods show a significant improvement in f1-macro and other metrics compared to the baseline, and thus we can conclude that our models are useful improvements.

It can be seen through comparing the CV score of all the models, that SVM models generally performed better than Multinomial Naive Bayes in terms of f1-macro scoring. Also, using TF-IDF vectors as the feature space, generally improved the model's performance with regards to f1-macro scoring.

Another insight is that the less complex models generally performed better than the more complex models. This is implied through the low complexity factors chosen (C and gamma for the SVM methods, alpha for MNB), and the more complex configurations ultimately rejected by the in depth grid search methods used. It can also be seen that the hyperparameter values were reliant on the feature space used. SVM models that utilised TF-IDF feature spaces required a higher value of complexity (C) than their Wordcount vector counterparts.

Interestingly enough, a grid search for the SVC method found that a polynomial kernel with degree 1 is superior to the linear kernel. Since a polynomial kernel of degree 1 would be thought of as a linear kernel, we could only speculate that the polynomial's usage of the gamma coefficient (which the linear kernel does not use) is what made a difference.

Method Implementation is also a significant factor in determining performance, where the LinearSVC method utilising a one-vs-rest style of multi-class classification showed superior results to the SVC method with a linear kernel.

Discuss the metrics in the above two tables. Which metric(s) is/are more appropriate and why?

It can be seen from analysing both tables that macro-precision would most likely be the best metric for determining a final model, closely followed by f1-macro scoring. A macro averaging would be the most important aspect to utilise since the topic datasets are incredibly unbalanced, and doing any other form of averaging would give too much weight to the more numerous topics; the more numerous topics are more accurately classified and so classification performance for all the proposed models within the context of the problem scope would be mis-represented. Precision is the most important metric, since it is important that the users are given accurate article recommendations.

F1-macro scoring on the test set was chosen in building the models because there was concern that only using macro precision scoring would sway the models into choosing 1 or 2 of the lower classes. Since over half the topics are very low in number, this would mean the classifier may make extremely few recommendations to optimise the scoring parameter; how many recommendations exactly, needs to be tested.

Accuracy and recall by themselves can be ignored. As observed in the exploratory data set, there is enormous data imbalance leaning towards “IRRELEVANT,” and only accurately classifying that topic is counter productive. High recall on the other hand is not useful without high precision, since we want to make accurate article recommendations.

If you continue this project, how would you improve it, e.p. Use of other methods and parameters that have a potential to be useful but not tried yet?

Improvements might be made by focusing on the precision and recall of the article recommendations. This would be difficult to implement since it would require coding functions specific to the project and the general search methods provided by scikit- learn may not be applicable.

A major improvement would be pipelining. There is data leakage in pre-fitting models before cross validation creating over optimistic cross validation scores (Mullet, Guido, 2017, pg 313). The grid search should be combined with a pipeline that fits the model to every new cross validation training fold before applying itself to that iteration’s test fold. This would be more costly in terms of computation, but would provide better assessment of hyperparameter combinations.

More powerful hardware would allow us to search a wider range of hyper parameter values. Some parameter combinations were unable to be analysed since computation time rose dramatically, like SVM polynomial kernel with a degree of 5. In addition, Randomized search was combined with Grid search in an attempt to narrow down the hyper parameter space. It is an option to search the entire hyper parameter space (or at least a wider range) with extremely powerful hardware.

It was notable how TDF-IF vectors were superior to Word Count vectors, so exploring the performance of other well-established feature space methods is essential in creating an optimal model. Creating custom feature spaces may also be worth considering. For example the mean length of “SHARE LISTINGS” was noticeably less than every other topic and the lower numbered topics were generally longer in length. Implementing article length with a bag of words model may improve classifying power for some topics. Additionally, other methods of machine learning, such as unsupervised or hybrid, should also be explored.

Conclusion

The aim of this project is to produce a model that could accurately classify articles and reliably recommend 10 articles per topic (if able). Classifying newspaper articles by topic is a Multi-class text classification problem. Multi-class text classification is a challenging task, and different supervised learning methods have many ways to tackle this problem. Through exploring different

implementations, feature sets, and hyper parameters, many different solutions arise and must be analysed before deciding on an optimal solution. Optimality criteria chosen for this task was the macro average of the f1-score since there were large data imbalances.

Two methods of supervised learning are explored in detail: Multinomial Naive Bayes and Support-vector Machines. The bag of word feature models, count vectors and TF-IDF vectors, were both implemented. Additionally, search methods with different implementations of these methodologies, such as Grid Search and Random search, were utilised to find optimal hyper parameters for each feature set representing this particular problem. The results showed that TF-IDF vectorisation was generally superior to Word count vectorisation, and that the SVM models out performed the MNB models noticeably. Both models were compared to a majority topic classifier baseline and outperformed it tremendously.

In evaluating the results, it was found that there may be improvements to article recommendations if macro precision was chosen, however this needs to be tested. Additionally, it was proven that data leakage occurred due to not combining search methods with pipelining; this would incur higher computational costs but add greater reliability to the final model chosen. Future improvements to this method of analysis involve using more powerful hardware to search a greater parameter space, trying different feature space representations, and testing different supervised learning methods.

Reference

Literature

Articles
https://towardsdatascience.com/random-search-vs-grid-search-for-hyperparameter-optimization-345e1422899d
https://towardsdatascience.com/text-classification-in-python-dd95d264c802
https://en.wikipedia.org/wiki/Tf%E2%80%93idf

Code Documentation
https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC
https://scikit-learn.org/stable/modules/svm.html
https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html#sklearn.svm.LinearSVC
https://scikit-learn.org/stable/modules/model_evaluation.html#scoring-parameter
https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

Text books
Intro to machine learning textbook, Andreas C.Mueller, Sarah guido - pg 313,

Source Code

https://github.com/miguelfzafra/Latest-News-Classifier/tree/master/

Appendices

1.1 - Topic Encoding

Category name	Category code
ARTS CULTURE ENTERTAINMENT	0
BIOGRAPHIES PERSONALITIES PEOPLE	1
DEFENCE	2
DOMESTIC MARKETS	3
FOREX MARKETS	4
HEALTH	5

MONEY MARKETS	6
SCIENCE AND TECHNOLOGY	7
SHARE LISTINGS	8
SPORTS	9
IRRELEVANT	10

1.2 - Majority class classifier results on test set

	precision	recall	f1-score	support
ARTS CULTURE ENTERTAINMENT	0.00	0.00	0.00	3
BIOGRAPHIES PERSONALITIES PEOPLE	0.00	0.00	0.00	15
DEFENCE	0.00	0.00	0.00	13
DOMESTIC MARKETS	0.00	0.00	0.00	2
FOREX MARKETS	0.00	0.00	0.00	48
HEALTH	0.00	0.00	0.00	14
IRRELEVANT	0.53	1.00	0.69	266
MONEY MARKETS	0.00	0.00	0.00	69
SCIENCE AND TECHNOLOGY	0.00	0.00	0.00	3
SHARE LISTINGS	0.00	0.00	0.00	7
SPORTS	0.00	0.00	0.00	60
accuracy			0.53	500
macro avg	0.05	0.09	0.06	500
weighted avg	0.28	0.53	0.37	500