Aleksandra Afonina: 15440117, Martin Ivanchev: 15425770

I. Introduction

The fast growth of online news platforms has created a very competitive space where thousands of articles fight for users' attention every day. The question of why some articles become popular while others do not is of key importance to journalism and marketing analytics. The aim of this project is to investigate which characteristics of news articles have the greatest impact on their popularity, measured by the number of times they have been shared.

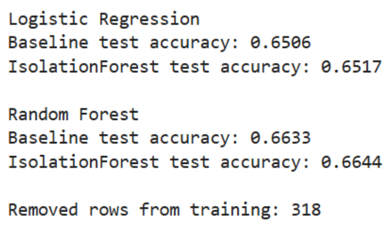
Research Question: "To what extent do unconventional indices (such as originality and effort) help predict online news popularity? If these scores provide limited predictive power, which alternative features are most influential, and which machine learning model performs best for predicting popularity?"

Since the aim of the study is to identify innovative topics, we wanted to understand how important effort and originality are for the popularity of online news. To answer this question, we use the Online News Popularity Dataset, which consists of 39,797 articles and includes 61 numerical and categorical features describing aspects such as article content, structure, keywords, sentiment, topics, and engagement metrics. The dataset also contains the target variable, *shares*, which we use to create a binary label showing whether an article is popular or not. We consider an article to be popular if it has more than 1,400 shares, which is the median of our data.

II. Methodology

Our first step is preprocessing the data. Since in the database information says that *url* and *timedelta* are not predictive and *shares* is the target variable, we need to remove them from the dataset before running our predictive models. For the last part of the preprocessing we decide to run an outlier detection.

We use Isolation Tree and it detects 318 outliers. To test how better we can predict popularity without the presence of outliers we run Logistic Regression and Random Forest on the dataset with and without outliers.



```
Logistic Regression
Baseline test accuracy: 0.6506
IsolationForest test accuracy: 0.6517

Random Forest
Baseline test accuracy: 0.6633
IsolationForest test accuracy: 0.6644

Removed rows from training: 318
```

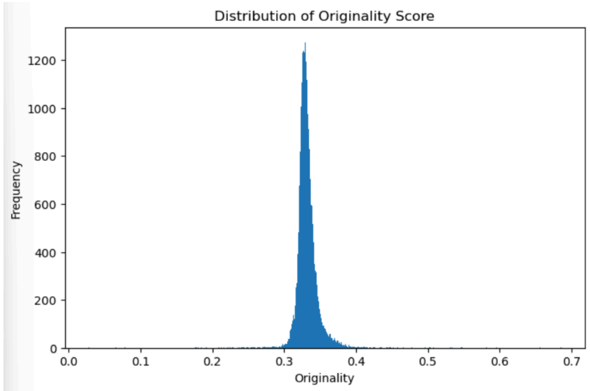
We observe that the boost in accuracy is minimal (around 0.001), and can be purely random, thus we decided to keep the outliers for extra information for our models and feature selection since outliers can carry important information and removing them does not necessarily improve our predictive power.

Originality and Effort Scores
We start by defining the originality index by three manually created features:

- *lexical_diversity (number of unique words / number of words),*
- *keyword_density (number of keywords / number of words),*
- *topic_niche=1-topic_popularity(average of kw_avg_avg and kw_avg_max*).

We do MinMax scaling of the three features, and then take the average of the three to compute our originality score.
This is our understanding of originality: text with high lexical diversity, a large number of keywords, and more niche topics. We compute our originality index to find out the average originality of all the

news present in our dataset. And we obtained a number close to 0.33. Then we look at the distribution of the data.



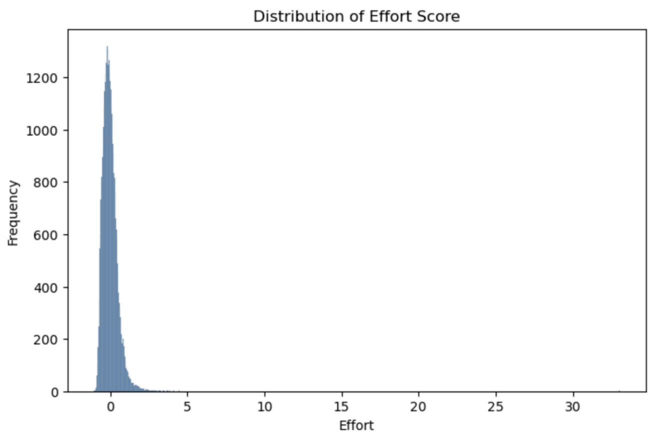
```
correlation:
                  originality3_new    shares
originality3_new          1.000000 -0.040749
shares                   -0.040749  1.000000
```

We see most of the articles have this originality score of 0.33, meaning that there is not a lot of variance captured. Without variance we lose predictive power, thus we expect to see an inaccurate model. We also obtain really low negative correlation with shares, thus it might be the case that lower originality is more popular.

Moreover, we created a composite effort score to capture how much work an article appears to require. The score combines five components:

- *writing effort: standardized average of text length and lexical richness (n_tokens_content, n_unique_tokens, n_non_stop_unique_tokens, n_tokens_title).*
- *research effort: number of hyperlinks (num_hrefs + num_self_hrefs).*
- *media effort: number of images and videos (num_imgs + num_videos).*
- *topical diversity: entropy of the LDA topic distribution (LDA_00–LDA_04).*
- *conceptual effort: number of extracted keywords (num_keywords).*

All components were standardized using StandardScaler and averaged into a single total_effort_score, providing a simple, interpretable measure of article effort used in later prediction models.



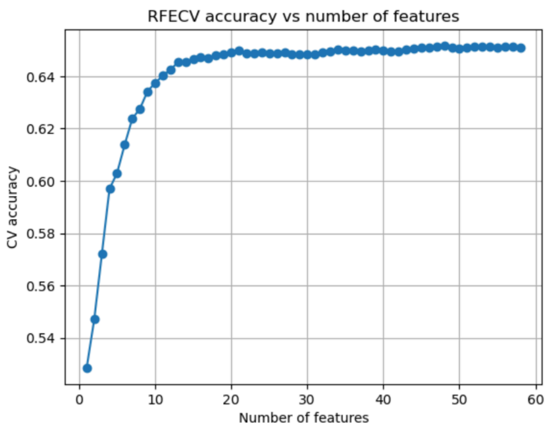
	total_effort_score	shares
total_effort_score	1.000000	0.044661
shares	0.044661	1.000000

However, the distribution of effort score also shows that there is not a lot of variance in the data, and as can be seen from the originality and effort scores, they are not really good predictors. Moreover, LogisticRegression with originality and effort has a test accuracy of 56% and 54%, respectively. Therefore, for further analysis, we need to identify other characteristics that more accurately predict popularity.

Feature Selection

To reduce dimensionality and improve model performance without losing important information, we applied several feature selection techniques.

First, we used RFECV (Recursive Feature Elimination with Cross-Validation) to iteratively remove weak predictors and identify the number of features that maximized validation accuracy. We visualized the RFECV accuracy curve to determine how model performance changed as features were removed.



RFECV identified 48 features as the optimal set based on cross-validated accuracy. However, 48 features is quite large and the RFECV curve shows that most of the performance gain happens much earlier. The biggest improvement occurs when increasing from 0 to about 10 features, followed by a smaller improvement from 10 to 20 features. After roughly 20 features, the accuracy does not improve much. This means that 20 feature sets might be enough to receive significant results, while also being more efficient.

Next, we applied LASSO regression, which performs embedded feature selection by shrinking irrelevant coefficients to zero. We also evaluated Random Forest feature importance to confirm which variables contributed most to predicting popularity.

After comparing model performance across these reduced feature sets, we found that the 20 features selected by LASSO provided the best feature set, which was therefore used for the final modeling experiments.

The features chosen by Lasso with C tuned to 0.002, in order to select exactly 20 features:

*['n_tokens_content','num_hrefs','num_imgs','num_keywords',
'data_channel_is_entertainment','data_channel_is_socmed',
'data_channel_is_tech','data_channel_is_world','kw_min_min',
'kw_avg_avg','self_reference_avg_sharess','weekday_is_friday',
'weekday_is_saturday', 'is_weekend', 'LDA_00', 'LDA_01', 'LDA_02',
'global_subjectivity', 'rate_negative_words', 'title_sentiment_polarity']*

Table I. Performance of models using different feature-selection methods and feature-set sizes

feature_set	***n_features***	***model***	***train_acc***	***test_acc***	***train_time_sec***
LASSO_10	*10*	*LinearSVC*	*0.637*	*0.6394*	*0.033*

MI_10	*10*	*LinearSVC*	*0.6135*	*0.6141*	*0.0692*
RF_10	*10*	*LinearSVC*	*0.618*	*0.6185*	*0.03*
LASSO_20	*20*	*LinearSVC*	*0.6414*	*0.6425*	*0.1035*
MI_20	*20*	*LinearSVC*	*0.6373*	*0.636*	*0.1535*
RF_20	*20*	*LinearSVC*	*0.6246*	*0.6261*	*0.4726*
LASSO_10	*10*	*LogReg*	*0.6374*	*0.638*	*0.019*
MI_10	*10*	*LogReg*	*0.616*	*0.6151*	*0.018*
RF_10	*10*	*LogReg*	*0.6196*	*0.6214*	*0.018*
LASSO_20	*20*	*LogReg*	*0.6442*	*0.6445*	*0.021*
MI_20	*20*	*LogReg*	*0.6382*	*0.6385*	*0.0275*
RF_20	*20*	*LogReg*	*0.6239*	*0.6234*	*0.0305*
LASSO_10	*10*	*RandomForest*	*0.9998*	*0.6123*	*1.2087*
MI_10	*10*	*RandomForest*	*0.9999*	*0.6277*	*2.4667*
RF_10	*10*	*RandomForest*	*0.9999*	*0.6293*	*2.6762*
LASSO_20	*20*	*RandomForest*	*1*	*0.6577*	*1.9249*
MI_20	*20*	*RandomForest*	*1*	*0.6508*	*2.4072*
RF_20	*20*	*RandomForest*	*1*	*0.6408*	*3.3921*

The column feature_set indicates the selection method (LASSO, Mutual Information, or Random Forest importance) and the number of selected features (10 or 20). train_acc and test_acc report the classification accuracy on the training and test sets, respectively, while train_time_sec shows the training time in seconds.

In our experiments we compared three feature-selection methods. We used LASSO, which fits a linear model with an L1 penalty and keeps only the most important features; Mutual Information, a filter method that ranks features by their individual relationship with the target; and Random Forest feature importance, which selects features that a Random Forest classifier finds most useful. For each method we created feature sets with 10 and 20 variables and trained three classifiers on them: LinearSVC, Logistic Regression, and Random Forest. We evaluated the models with training accuracy, test accuracy, and training time.

The results show that Random Forest as a classifier almost perfectly fits the training data (train accuracy is 1) but achieves only about 0.62-0.65 test accuracy, which indicates clear overfitting and comes with relatively long training times. The linear models (LinearSVC and Logistic Regression) have similar training and test accuracies and train much faster, suggesting better generalisation. Across all models, feature sets selected with LASSO, especially the 20-feature set, consistently achieve the highest test accuracy and when used with linear models, it was faster compared to the other sets of features. This is reasonable because LASSO is a linear method and selects features that work well for linear decision boundaries, which matches our main classifiers, while MI ignores feature interactions and RF-based selection is tailored to tree models. For these reasons, strong generalisation performance, stability and speed, we choose the LASSO_20 feature set for our final models.

Models Used

To evaluate the predictive power of our selected features, we used three supervised machine learning models: Logistic Regression, Random Forests, and a Neural Network (MLPClassifier). These models allow us to compare linear, tree-based, and nonlinear approaches to predicting article popularity. In addition to supervised learning, we applied K-means clustering as an unsupervised method to examine whether articles naturally group according to specific features. By comparing the resulting clusters with actual popularity rates, we assessed which determine natural separation in the data.

III. Results

For our experiment we utilised several ML algorithms with parameter tuning. Their results are presented in Table II.

Table II. Machine Learning Algorithms

Algorithms	***Number of Features***	***Best Parameters***	***Accuracy***	***Precision***	***Recall***	***F1-score***
Logistic Regression	*All*	-	*0.651*	*0.652*	*0.623*	*0.639*
Logistic Regression	*20 (Lasso)*	-	*0.645*	*0.643*	*0.628*	*0.635*
Logistic Regression	*Originality*	-	*0.550*	*0.545*	*0.524*	*0.534*
Logistic Regression	*Effort*	-	*0.542*	*0.548*	*0.411*	*0.469*
Random Forest	*All*	max_depth = None max_features = sqrt n_estimators = 400	*0.665*	*0.656*	*0.657*	*0.656*
Random Forest	*20 (Lasso)*	*max_depth = 15* *max_features = sqrt* *n_estimators = 400*	*0.661*	*0.658*	*0.662*	*0.660*
Random Forest	*Originality*	*max_depth = 5* *max_features = sqrt* *n_estimators = 150*	*0.555*	*0.579*	*0.357*	*0.441*
Random Forest	*Effort*	max_depth = 5 min_samples_leaf = 1 min_samples_split = 5 n_estimators = 150	*0.544*	*0.550*	*0.417*	*0.474*
Multilayer Perceptron	*All*	*alpha = 0.01* *hidden layers = 1* *hidden layer size = 32*	*0.649*	*0.651*	*0.622*	*0.636*
Multilayer Perceptron	*20 (Lasso)*	*alpha = 0.01* *hidden layers = 1* *hidden layer size = 32*	*0.652*	*0.650*	*0.640*	*0.645*
Multilayer Perceptron	*Originality*	*alpha = 0.01* *hidden layers = 1* *hidden layer size = 32*	*0.553*	*0.576*	*0.359*	*0.442*
Multilayer Perceptron	*Effort*	*alpha = 0.0001* *hidden layers = 2* *hidden layers size = (64, 32)*	*0.543*	*0.545*	*0.445*	*0.490*

As shown in Table II above, we compared Logistic Regression, Random Forest, and Multilayer Perceptron (MLP) models across four feature settings: all features, 20 Lasso-selected features, originality score only, and effort score only. All models were tuned using cross-validated grid search to ensure a valid comparison and avoid performance differences caused by non-optimal hyperparameters.

Across all models, effort and originality alone show limited predictive power. Logistic Regression using only originality achieves an accuracy of 0.550 (F1 = 0.534), while effort performs even worse (0.542, F1 = 0.469). Similarly, Random Forest and MLP models based solely on originality and effort achieve

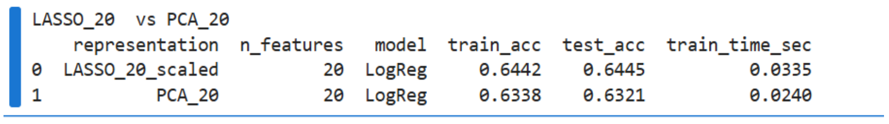
accuracy levels of 0.54-0.56, indicating that these unconventional metrics alone do not fully reflect the factors that determine popularity.
In contrast, models using structural and content-based features perform significantly better. Random Forest achieves the best overall performance, with an accuracy of 0.665 (F1 = 0.656) using all features, and almost identical results (0.661, F1 = 0.660) using only the 20 Lasso-selected features. This demonstrates that feature selection successfully reduces dimensionality without sacrificing performance.
Logistic Regression and MLP show similar trends. Restricting the tuned models to the 20 selected features yields performance comparable to using all features, indicating that the selected variables preserve most of the predictive information. In the case of MLP, performance even improves slightly with fewer features (accuracy with 20 features: 0.652, with all features: 0.649), suggesting that dimensionality reduction reduces noise and model confusion. Nevertheless, both models remain slightly weaker than Random Forest, likely due to Random Forest's ability to see nonlinear interactions.
Overall, the results suggest that tuned models relying on structural, topical, and contextual features outperform models based solely on effort or originality scores. Among the evaluated models, Random Forest combined with Lasso-selected features and tuned hyperparameters offers the strongest and most stable performance, making it the most suitable model for predicting online news popularity in this study.

Table III. Time and Accuracy Comparison

Model	Accuracy	Time
Logistic Regression All	0.6506	0.7287
Logistic Regression (Lasso 20)	0.6445	0.0426
Random Forest All	0.6660	5.4944
Random Forest (Lasso 20)	0.6626	2.8871

We compare the accuracy and speed of Logistic Regression and Random Forest ran with either all the features or only with the set from Lasso. Accuracy drops negligibly (around 0.006), but training becomes around 18 times faster, which is a significant difference. To sum up, logistic regression with the Lasso-selected features is the most convenient if you can afford to lose 0.02 accuracy but gain more than hundred times more efficiency compared to the most accurate model being Random Forest with all features(accuracy = 0.666).

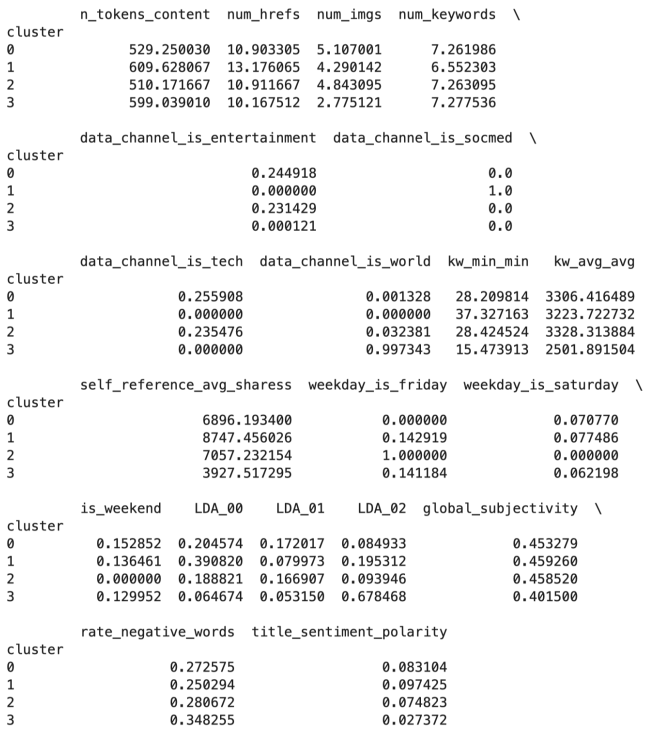
If we want to prioritize speed we need to reduce the dimensionality of our data. We continue our analysis by comparing two ways to reduce the data to 20 inputs: LASSO_20, which keeps 20 original features chosen to predict popularity, and PCA_20, which uses 20 principal components that mainly capture overall variance. Logistic Regression performed better with LASSO_20 (0.6445) than with PCA_20 (0.6321). PCA was only slightly faster, so the time savings were small compared to the accuracy drop. This shows that PCA can compress the data, but it does not necessarily keep the information that best separates popular from non-popular articles, so LASSO is the better choice here, although if time is a really important metric, someone can use principal components as features.



```
LASSO_20  vs PCA_20
    representation  n_features   model  train_acc  test_acc  train_time_sec
0  LASSO_20_scaled          20  LogReg     0.6442    0.6445          0.0335
1           PCA_20          20  LogReg     0.6338    0.6321          0.0240
```

IV. Discussion

To evaluate whether unconventional scores such as effort or originality meaningfully explain popularity, we complemented supervised models with K-means clustering (4 clusters).



```
         n_tokens_content  num_hrefs  num_imgs  num_keywords  \
cluster
0              529.250030  10.903305  5.107001      7.261986
1              609.628067  13.176065  4.290142      6.552303
2              510.171667  10.911667  4.843095      7.263095
3              599.039010  10.167512  2.775121      7.277536

         data_channel_is_entertainment  data_channel_is_socmed  \
cluster
0                             0.244918                     0.0
1                             0.000000                     1.0
2                             0.231429                     0.0
3                             0.000121                     0.0

         data_channel_is_tech  data_channel_is_world  kw_min_min   kw_avg_avg
cluster
0                    0.255908               0.001328   28.209814  3306.416489
1                    0.000000               0.000000   37.327163  3223.722732
2                    0.235476               0.032381   28.424524  3328.313884
3                    0.000000               0.997343   15.473913  2501.891504

         self_reference_avg_sharess  weekday_is_friday  weekday_is_saturday  \
cluster
0                       6896.193400           0.000000             0.070770
1                       8747.456026           0.142919             0.077486
2                       7057.232154           1.000000             0.000000
3                       3927.517295           0.141184             0.062198

         is_weekend    LDA_00    LDA_01    LDA_02  global_subjectivity  \
cluster
0          0.152852  0.204574  0.172017  0.084933             0.453279
1          0.136461  0.390820  0.079973  0.195312             0.459260
2          0.000000  0.188821  0.166907  0.093946             0.458520
3          0.129952  0.064674  0.053150  0.678468             0.401500

         rate_negative_words  title_sentiment_polarity
cluster
0                   0.272575                  0.083104
1                   0.250294                  0.097425
2                   0.280672                  0.074823
3                   0.348255                  0.027372
```

Table IV. Clustering

Clusters	Number of articles	% Popular articles
Cluster 0	24 841	51.7%
Cluster 1	2 323	71.4%
Cluster 2	4 200	53.1%
Cluster 3	8 280	34.3%

The largest cluster (Cluster 0, 24,841 articles) represents typical online articles with moderate length (529 tokens), average hyperlink usage (11), balanced topic distributions, and moderate keyword optimization (kw_avg_avg = 3306). Its popularity rate of 51.7% aligns closely with the dataset baseline, indicating that this cluster represents typical article performance.

Cluster 3 (8,280 articles) is the least successful group, with a popularity rate of 34.3%. This cluster consists almost entirely of world-news articles (data_channel_is_world = 0.997), shows the weakest keyword optimization (kw_avg_avg = 2502), uses fewer images, and has the highest rate of negative words. These characteristics are associated with consistently low levels of engagement.

Cluster 1 (2,323 articles) achieves the highest popularity (71.4%) and consists entirely of social media content (data_channel_is_socmed = 1.0). Articles in this cluster are longer (around 610 tokens), include more hyperlinks (13), and show strong keyword optimization and internal referencing. These characteristics suggest well-structured content that attracts higher engagement.

Cluster 2 (4,200 articles) performs slightly above average, with a popularity rate of 53.1%. It is characterized by a stronger presence in entertainment and tech channels, balanced keyword usage, and broader topic coverage, but lacks strong timing or optimization effects.

Overall, the clustering results align with the supervised models: unconventional effort and originality scores alone do not allow for a meaningful division between popular and unpopular articles, while concrete features such as content category, keyword optimization, topic composition, sentiment, and publication context naturally divide articles into groups with significantly different popularity outcomes.

V. Conclusion

We started this project with a simple question: do originality or effort put in an article help predict if it will be popular.
To test that idea, we built two indices: originality, which combined lexical variety, keyword density and the reverse of topic popularity, and also a feature measuring effort, which involves writing length/complexity, research links, media used and topic spread. This way we look at something closer to how a reader might judge the work behind an article. However, when we look at originality, we see that the scores are tightly clustered, and there is not a lot of variance. In addition, its relationship with shares was weak, so originality alone could not separate popular from non-popular articles. We obtained the same results when we looked at the effort index. This result pushed us to look at a broader solution: instead of choosing a single index, we asked which features actually help a model make the decision.

We then ran feature selection with different models: LASSO, Mutual Information and Random Forest feature importance, and finally RFECV to see how performance grows as we add information. The pattern showed us that accuracy improves fast in the first 10 features, then improves a bit more by 20, and then we do not get any significant improvement. We focused on 20 as a realistic compromise for the number of features since it achieves high accuracy, but also is much more efficient than going for all the features to obtain maximum precision. In the model comparisons, Random Forest often achieved the best test accuracy but also showed long training time, while Logistic Regression stayed stable and fast. They worked the most well with the features chosen from the Lasso Model. Finally, since we are trying to be both accurate and efficient, we also examined how far we can reduce the complexity of the features while keeping good performance. We applied PCA because it reduces the dimensions of the data to those that capture the most variance and therefore should allow the models to train faster. As expected, the model was faster, but still achieved lower accuracy relative to the features selected by LASSO.
Consequently, the results show that the popularity of an article cannot be explained solely by a higher level of effort or originality. Rather, popularity depends primarily on how the content is structured and presented, with characteristics related to discoverability, such as data channel indicators, keyword statistics, topic composition, the day of the week on which the article is posted and sentiment, playing a more important role.
In terms of method, our analysis shows that we get the best predictions with a smaller set of about 20 features. Adding more variables makes the model more complex and slower to run, but does not improve accuracy much. This result highlights how important it is to choose a limited set of useful features when building accurate popularity prediction models.

This study has several limitations. Firstly, the constructed effort and originality scores are inherently subjective, as they depend on the characteristics we have chosen, such as keyword density, lexical diversity, and topic entropy. Although these measures are interpretable, alternative definitions may lead to different conclusions. Future work could utilise advanced NLP methods to obtain more objective originality and effort scores.
Secondly, the popularity of an article was defined using a binary threshold based on the number of shares, which is arbitrary. Different threshold options may change the model's performance and interpretations. Future research could explore alternative popularity definitions to achieve higher accuracy.

References

[Fernandes et al., 2015] Fernandes, K., Vinagre, P., and Cortez, P. (2015). A proactive intelligent decision support system for predicting the popularity of online news. In *Portuguese Conference on Artificial Intelligence*, pages 535–546. Springer.