Virality Predictor

For this problem three different approaches are tested with different models and different data preprocessing methods.

Only aticles with eventType == CONTENT SHARED are used.

Virality is calculated after replacing eventType categories with the event weights as given in the project definition. First the virality values are replaced with integer weights from the Virality formula. Later, they are grouped by per person and per article and summed. Finally, they are grouped by per article and summed.

As this method gives virality values in range [1, 7907], some models are also tested with normalized virality values as well as the raw values

In the rest of the document the below questions will be answered for each model.

a. What features did you consider? - Answer is below
b. What model did you use and why? - Answer is below
c. What was your evaluation metric for this? - Answer is below

d. What features would you like to add to the model in the future if you had more time?

   I would investigate timestamp and try to find a correlation between virality and the day an article is shared (weekend/weekday). Based on my inspection on this, I know that users are more actively interacting with the articles on weekends, while new articles are shared on mostly weekdays.

   It would also be interesting to set a number of days as a limit while calculating the virality. For example, an article's virality value could be V in X many days. If we do not set a limit, V can grow indefinitely in time.

   In this work I only focused on the language that article is written. If I had more time I would like to experiment on the user region and author country.

e. What other things would you want to try before deploying this model in production?

   I think creating the term-document matrix using BM25 scoring would be worth investigating.

   More detailed feature analysis would be helpful to find really representative terms for term-document matrices.

Discarding adverbs and adjectives during preprocessing of bag-of-words could be meaningful.

I would like to try more models, and do a more broad hyperparameter search.

Approaches

1. **Collaborative Filtering Recommender System**
For this approach Surprise, a Python scikit for recommender systems is used. For each problem a benchmark is completed to determine the optimal algorithm. Later for each algorithm a hyperparameter seach is done using Grid Search.

Problems:
1. Articles in English.
2. Articles in Portuguese
3. Articles in both languages

Utility matrix : Users & Articles matrix. Ratings as virality values

The models are selected based on the top 3 algorithms from the benchmark results., excluding the Normal Predictor.

	EN	PT	EN and PT
Features	Utility matrix	Utility matrix	Utility matrix
Model	CoClustering	KNNBasic (Item Based)	SlopeOne
Evaluation Metric	Precision@k, Recall@k k=[5,10]	Precision@k, Recall@k k=[5,10]	Precision@k, Recall@k k=[5,10]

Results:

With raw virality values all three models did significantly worse compared to the models trained on normalized virality values.

	EN		PT		EN & PT	
Raw virality	P@5	R@5	P@5	R@5	P@5	R@5
	0.018	**0.017**	**0.003**	**0.004**	**0.016**	**0.013**
	P@10	R@10	P@10	R@10	P@10	R@10
	0.009	**0.009**	**0.001**	**0.001**	**0.008**	**0.006**
Normalized virality	P@5	R@5	P@5	R@5	P@5	R@5
	0.24	**0.22**	**0.36**	**0.48**	**0.32**	**0.37**
	P@10	R@10	P@10	R@10	P@10	R@10
	0.11	**0.11**	**0.18**	**0.26**	**0.16**	**0.21**

2. **Classification using TFIDF**

   For this approach TFIDF matrix is constructed using shared articles in both languages, separately. First, article domain name, title and text are combined into bag of words. Later, bag of words is cleaned (lemmatized, no stopwords, no punctuation or digits) using nltk library. Finally TFIDF matrix is constructed using sklearn TFIDFVectorizer.

   Virality values are normalized and categorized. As the dataset is highly imbalanced, minority classes are resampled to create balance. Finally virality class labels are predicted. For categorization, empirically 15 bins created, which categorized the normalized values into 4 classes.

   A hyperparameter seach is done using Grid Search before training and testing the models.

   KNN Classifier is preferred because a similarity pattern between the articles is desired.

   <u>Problems:</u>
   1. Articles in English
   2. Articles in Portuguese

	EN	PT
Features	200 words from vectorizer	200 words from vectorizer
Model	KNNClassifier	KNNClassifier
Evaluation Metric	Precision, Recall, F1-Score	Precision Recall F1-Score

   <u>Results:</u>

   For English articles F1-Score: 0.54
   The model struggled to learn class 9, which corresponds to highest ever virality values. Upsampling this class didn't also help.

```
Features: ['ad', 'afraid', 'ambiguity', 'analytics', 'angular'
, 'assert', 'assertion', 'bdd', 'bitcoin', 'blockchain', 'b
ock', 'callback', 'changelog', 'compete', 'couchdb', 'deepeq
ual', 'destination', 'drupal', 'electron', 'elixir', 'enforc
e', 'fixture', 'frontend', 'growth', 'handler', 'humility',
 'icon', 'independent', 'jquery', 'kurzweil', 'le', 'liquiba
se', 'logical', 'martin', 'meeting', 'mysql', 'npm', 'overr
ide', 'parent', 'procedure', 'progressive', 'qunit', 'respon
sible', 'ruby', 'science', 'superior', 'synchronous', 'types
cript', 'vr', 'xml']
```

Classes	Precision	Recall	F1-Score
1	1.00	0.63	0.78
2	0.99	0.70	0.82
3	0.38	1.00	0.55
9	0.00	0.00	0.00

For Portuguese articles F1-Score: 0.82
As a smaller but more balanced subset, it is easier to generalize.
Upsampling under-represented classes helped the model to learn better.

```
Features: ['afirma', 'api', 'apis', 'aprendizado', 'arduino',
 'arquitetura', 'arquivo', 'arquivos', 'automatizado', 'capyb
ara', 'competências', 'crase', 'cucumber', 'definido', 'elem
ento', 'encontro', 'erros', 'ex', 'expressão', 'geração', '
humanos', 'id', 'liderança', 'líder', 'líderes', 'medida',
'microserviços', 'mulheres', 'mãe', 'pagamento', 'palavra',
'passo', 'path', 'perceber', 'português', 'preposição', 'pri
ncípio', 'produtividade', 'robô', 'robôs', 'slack', 'sprint'
, 'substantivo', 'swagger', 'ti', 'trello', 'type', 'ui',
'ux', 'verbo']
```

Classes	Precision	Recall	F1-Score
1	1.00	0.60	0.75
2	0.90	0.94	0.91
3	0.92	0.77	0.84
4	0.62	1.00	0.77

3. **Regression using TFIDF**

For this approach TFIDF matrix is constructed using shared articles in both languages, separately. First, article domain name, title and text are combined into bag of words. Later, bag of words is cleaned (lemmatized, no stopwords, no punctuation or digits) using nltk library. Finally TFIDF matrix is constructed using sklearn TFIDFVectorizer.

Virality values are normalized. Prediction is done using normalized virality as labels.

A hyperparameter seach is done using Grid Search before training and testing the models.

SGDRegressor is used as a simple linear model.

Problems:
1. Articles in English
2. Articles in Portuguese

	EN	PT
Features	200 words from vectorizer	200 words from vectorizer
Model	RandomForestRegressor	RandomForestRegressor
Evaluation Metric	R2	R2

Results:

Models struggle to learn both datasets, articles in English and Portuguese. It might be helpful to add more features to the input data besides the term frequencies. Weekday/weekend, timestamp, content type, author region could be one of those additional features.

For English articles R2 score 0.37

```
Features: ['ad', 'ai', 'algorithm', 'analytics', 'architecture
', 'artificial', 'aws', 'bank', 'bitcoin', 'blockchain', 'b
ot', 'brand', 'browser', 'button', 'car', 'card', 'class',
 'command', 'compute', 'container', 'database', 'docker', 'd
rupal', 'element', 'enterprise', 'error', 'financial', 'grow
th', 'host', 'input', 'java', 'javascript', 'layer', 'leade
r', 'load', 'map', 'marketing', 'module', 'neural', 'node',
  'pattern', 'percent', 'query', 'rule', 'sale', 'science',
 'storage', 'stream', 'study', 'window']
```

For Portuguese articles R2 score 0.05

```
Features: ['agora', 'alguns', 'ante', 'aqui', 'caso', 'client
e', 'clientes', 'coisas', 'criar', 'dado', 'dentro', 'desen
volvimento', 'deve', 'digital', 'diz', 'então', 'gestão', '
google', 'grande', 'hoje', 'informações', 'internet', 'menos
', 'milhões', 'negócio', 'negócios', 'novo', 'onde', 'outra
s', 'parte', 'paulo', 'plataforma', 'possível', 'primeiro',
 'processo', 'produtos', 'projeto', 'projetos', 'qualquer',
'rede', 'segundo', 'seguro', 'sempre', 'serviços', 'sistema'
, 'trabalho', 'tudo', 'usuário', 'valor', 'vida']
```