Using Textual Features to Predict Online Article Popularity

Presented by Claudia Adam,
Anastasiia Poverenova,
Vincent So

Agenda

→ Introduction & Dataset

→ Models: Method and Performance

→ Comparing Results

→ Conclusion

→ Future Direction

Introduction

Dataset

- Data from Mashable, an online media website
- Article metadata and natural language processing features
- 60 features (58 predictive, 2 non-predictive)
- Target variable – number of shares
- 39,000+ articles
- Connected social media data analysis with predictive modeling

Our Project

- Can we predict article popularity before they're published?
- Build and compare three predictive models: Logistic Regression, KNN, Random Forest
- Convert shares into a categorical variable (popular vs. unpopular)
- Utilized different methods to split the data, evaluate the model, and adjust parameters to customize models
- Compared our findings to the research paper when applicable

Predictive Models →

Logistic Regression

Logistic Regression:

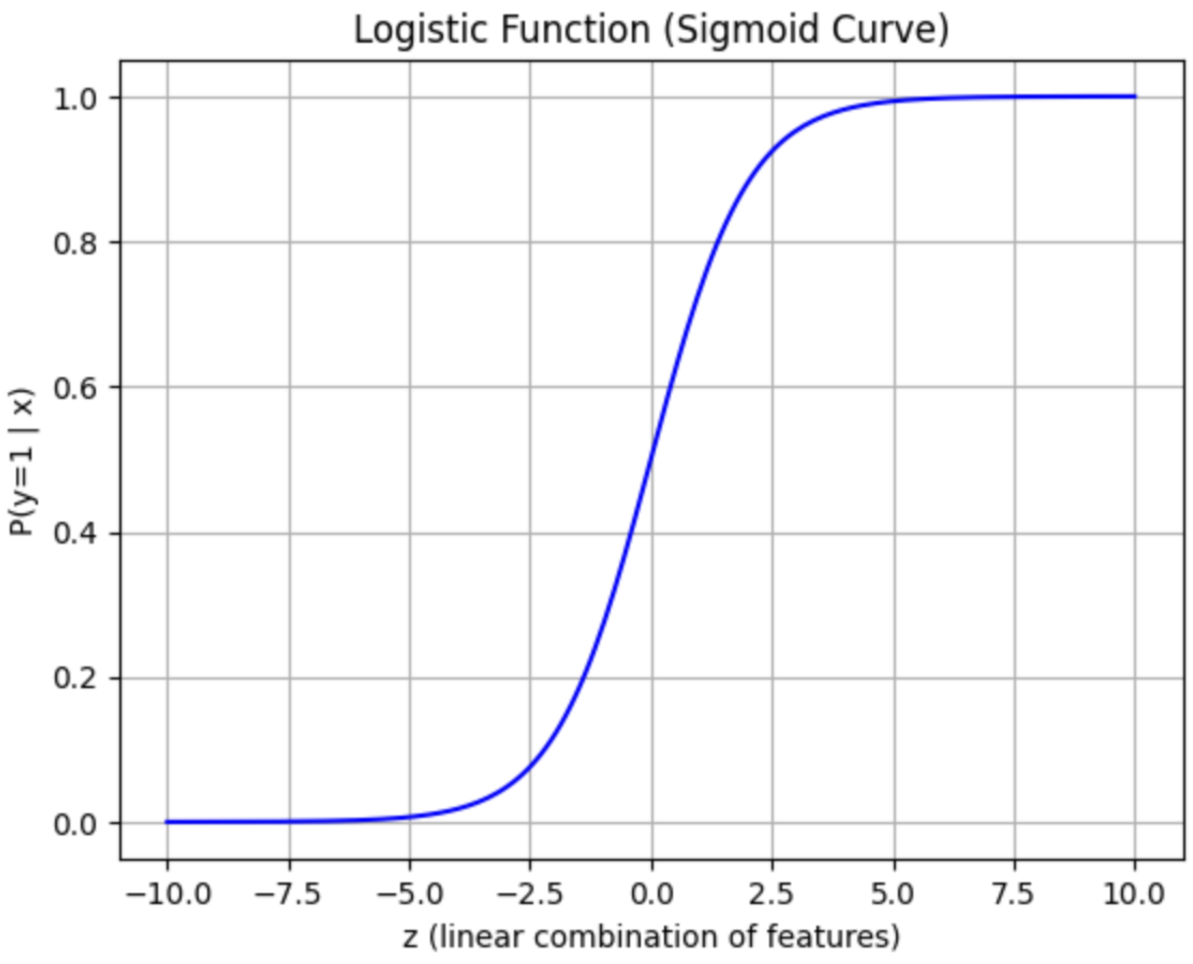
- a linear classifier that predicts probability of popularity using a weighted sum of features passed through a sigmoid

We tuned hyperparameters with GridSearchCV:

- Best penalty = L2, C = 100.0
- Solver = liblinear

Data pre-processing:

- Features scaled with StandardScaler
- Train/test split = 80/20, stratified



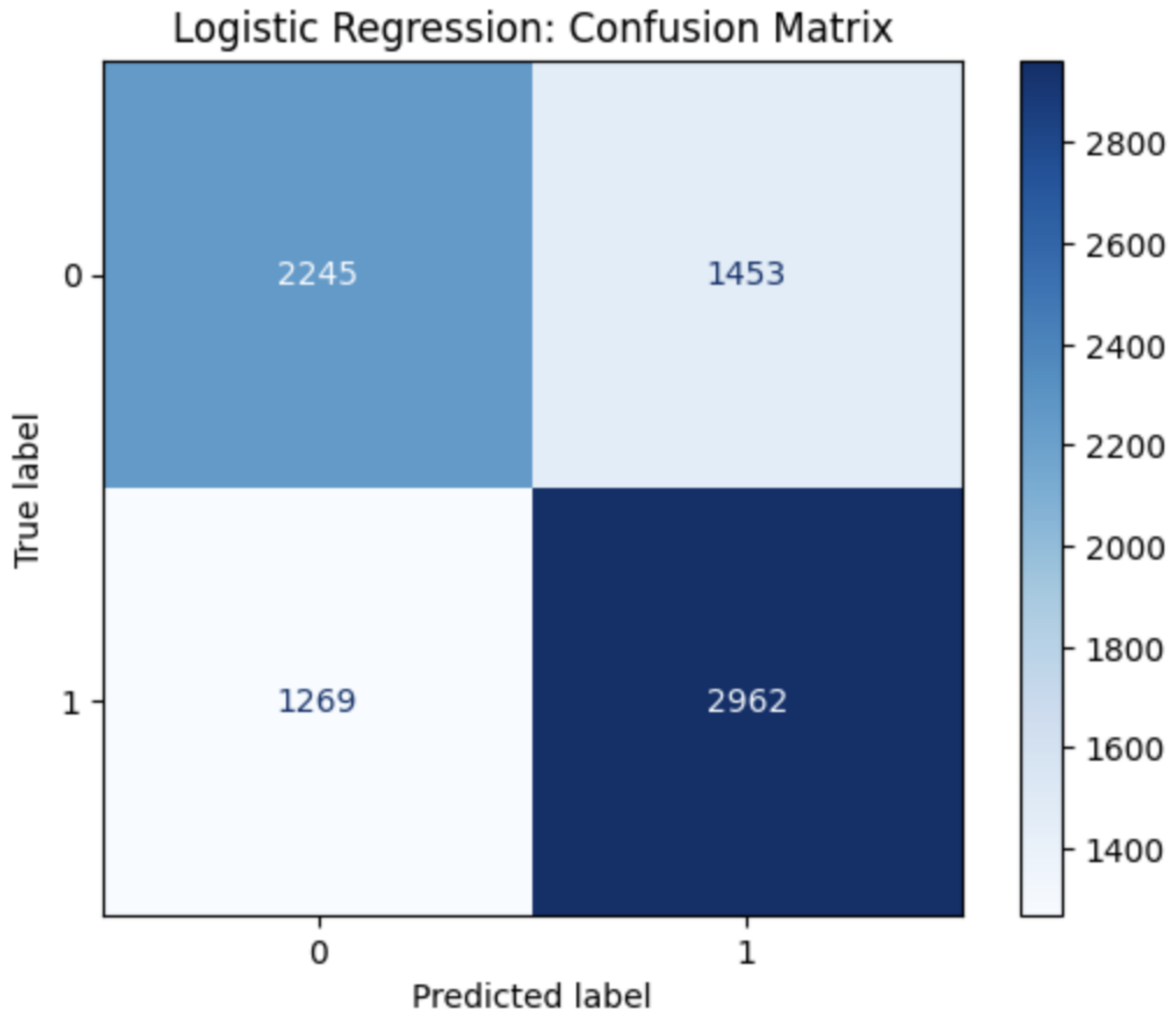
Logistic Regression

Key Results:

- Accuracy: 0.657
- Precision: 0.671
- Recall: 0.700
- F1-score: 0.685
- AUC: 0.708

Key insights:

- Balanced performance, with stronger recall → good at catching popular articles.
- Top positive drivers: more non-stop words, higher avg keyword weight, weekend publishing.
- Top negative drivers: high ratio of unique tokens, extreme keyword max values.



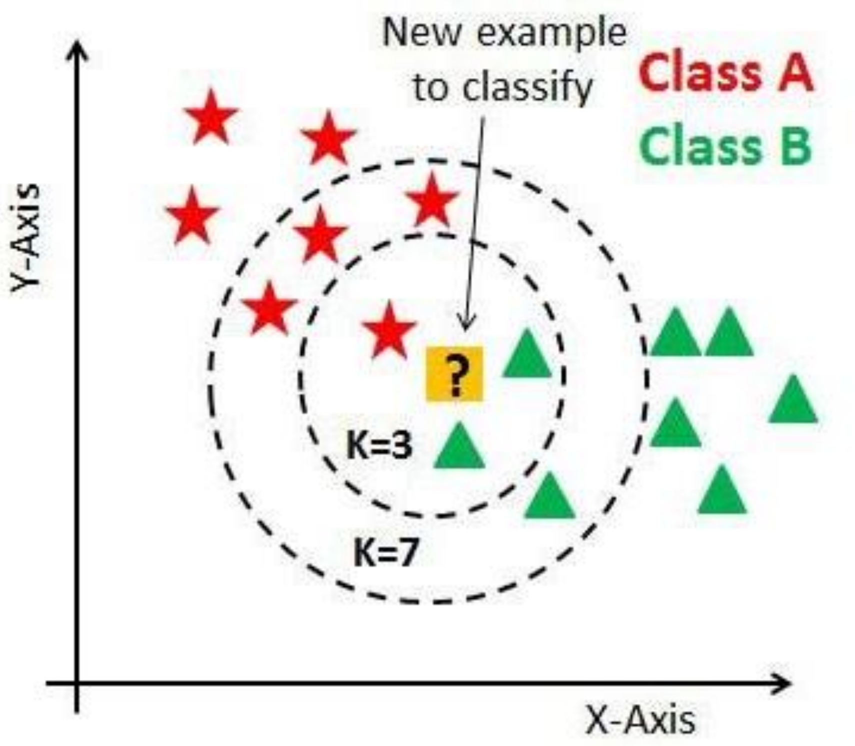
Advantages

- **Simple and efficient baseline** model for binary classification
- **Interpretable coefficients** → shows how each feature increases/decreases popularity
- **Good balance between precision and recall**
- **Fast to train and deploy**, even on large datasets
- **Calibrated probability outputs** allow flexible decision thresholds

Deployment

- Useful for **news organizations and editors** needing transparent predictions
- Can guide **headline/keyword optimization** and publishing timing
- Should monitor **precision/recall trade-offs** depending on business goals
- Easy to deploy via a **Pipeline in scikit-learn**, wrapped in an API or CMS plugin
  - Immediate feedback on predicted probability of popularity
- Collect additional signals (author history, early engagement, social trends) to improve accuracy
- Update regularly to **adapt to changing reader interests**

KNN Classification



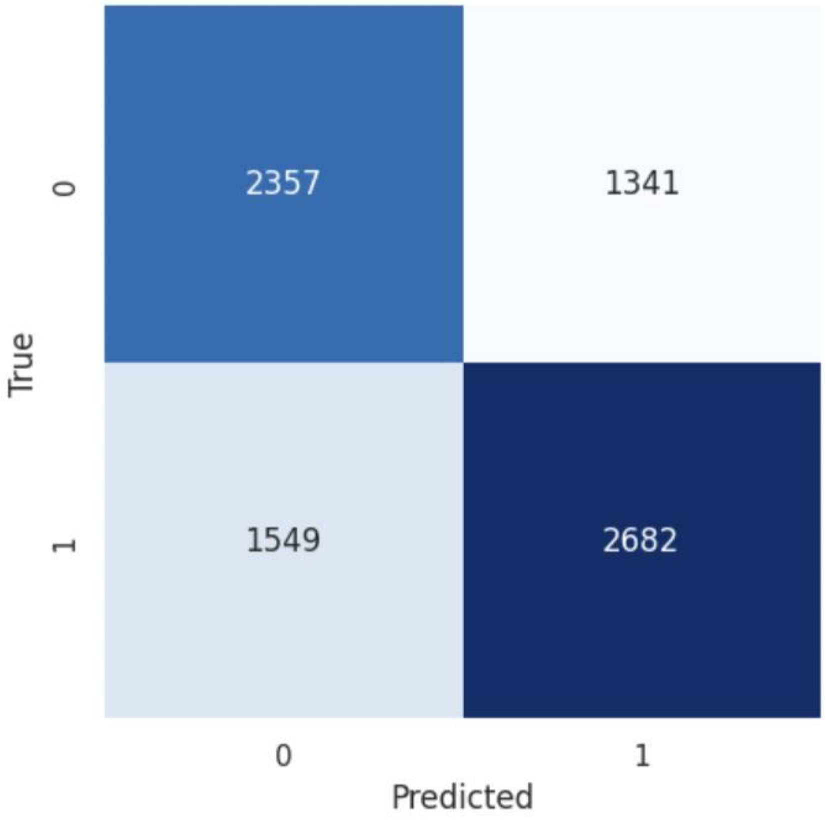
Data pre-processing:

- Features scaled with StandardScaler to ensure fair distance calculations
- Train/test split = 80/20
- **Hyperparameters Tuning (using Pipeline + GridSearchCV):**
- Tested different values of k, weighting schemes, and distance metrics
- Best config: k = 15 neighbors, uniform weights, Euclidean distance

KNN Model Mechanism:

- For each test article, it finds the k most similar training articles
- It assigns the article to the majority class among those neighbors

KNN Classification



Key Results:

- Accuracy: 0.636
- Precision: 0.667
- Recall: 0.634
- F1-score: 0.650
- AUC: 0.681

- The model performs moderately well – better than random guessing
- Good balance between precision (correctly flagging popular articles) and recall (capturing most popular articles)
- AUC shows the model has some ability to separate popular from non-popular articles across thresholds

Advantages

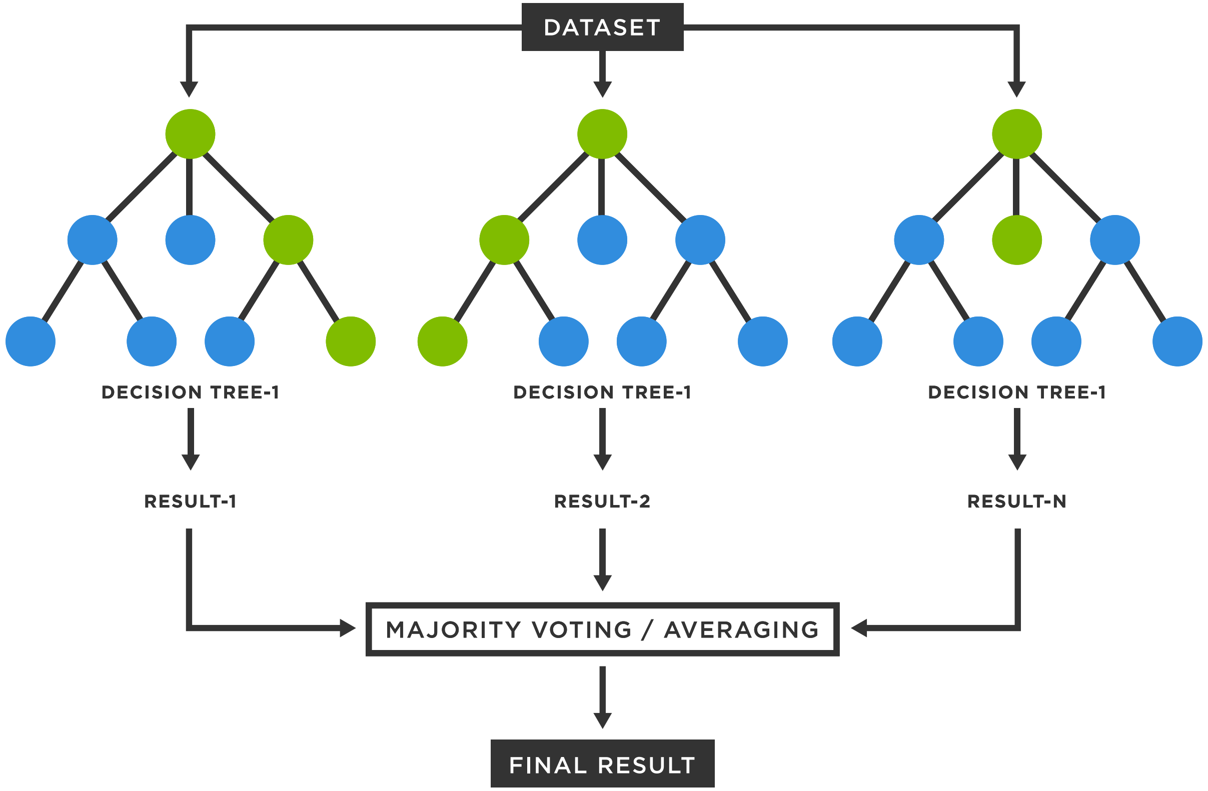
- **Predicts popualrity based on similarity to past articles**
- **Non-parametric: no assumptions about feature distributions**
- **Flexible: adapts to local data structures as neighbors drives the prediction**
- **Can show which neighbors influenced the classification**
- **Competive performance: balanced precision (0.667) and recall (0.634)**

Deployment

- **Useful for news organizations and digital publishers to flag likely-popular articles before release**
- **Can guide content strategy and supports audience targeting**
- **Deploy via a Pipeline in sci-kit learn, ensuring scaling and prediction consistency in production**
    - Immediate feedback loop: probability output can be shown to editors as likelihood of popularity
- **Update reguarly regularly with new articles to keep neighborhood comparisons current**

Random Forest

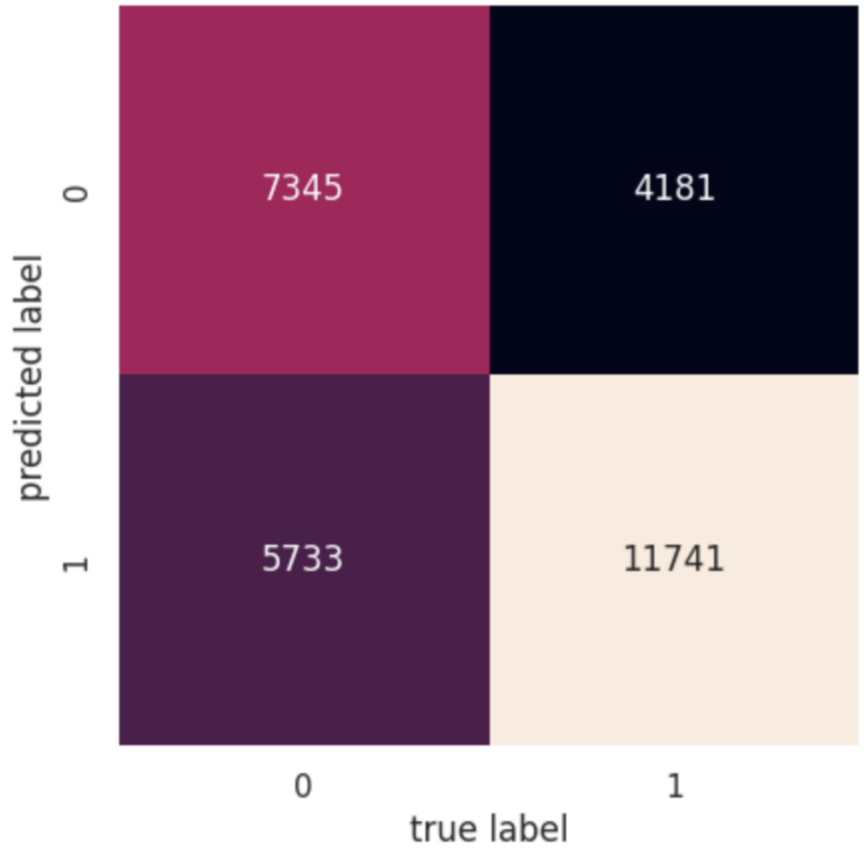
- **Authors best performing model**
- **Preprocessing**
    - Is_popular variable
    - Sorting data by time
    - Log transforming unbounded features
- **Rolling windows evaluations**
    - Training sets of 10,000 articles
    - Obtained the fitted model and best estimator from a grid search
    - Saved metrics for each window



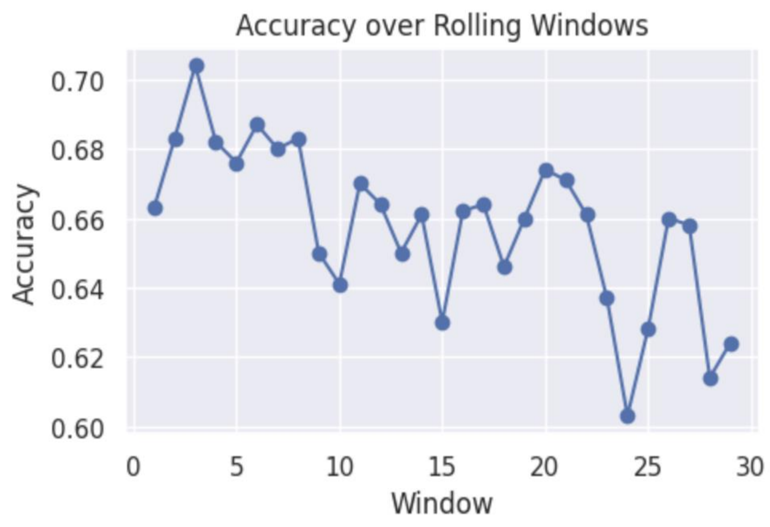
Random Forest: Results

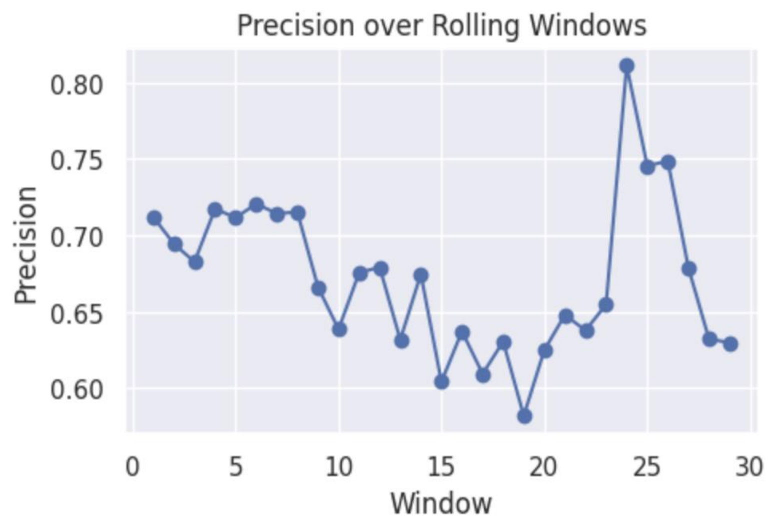
- Closely matched author's metrics
- ROC AUC = 0.71
- Model was better at predicting class 1 (popular articles)

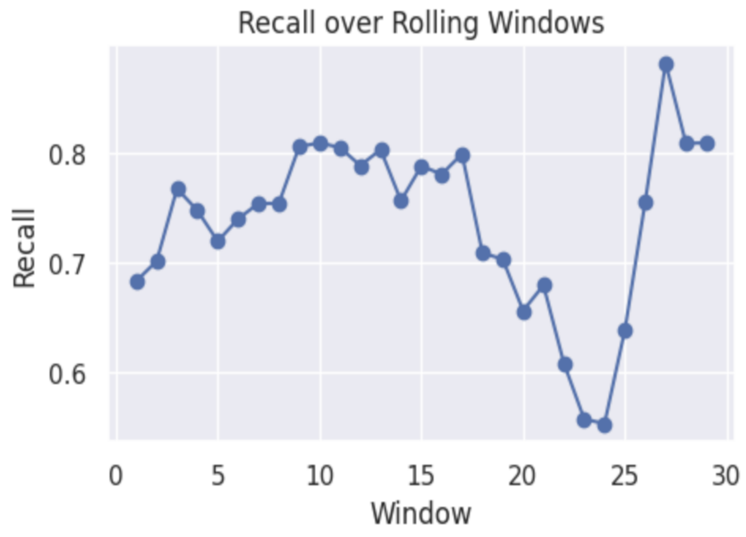
```
AUC: 0.7122
Accuracy: 0.6581
Precision: 0.6719
Recall: 0.7374
F1: 0.7031

Classification report:
              precision    recall  f1-score   support

 not popular       0.64      0.56      0.60     13078
     popular       0.67      0.74      0.70     15922

    accuracy                           0.66     29000
   macro avg       0.65      0.65      0.65     29000
weighted avg       0.66      0.66      0.66     29000
```

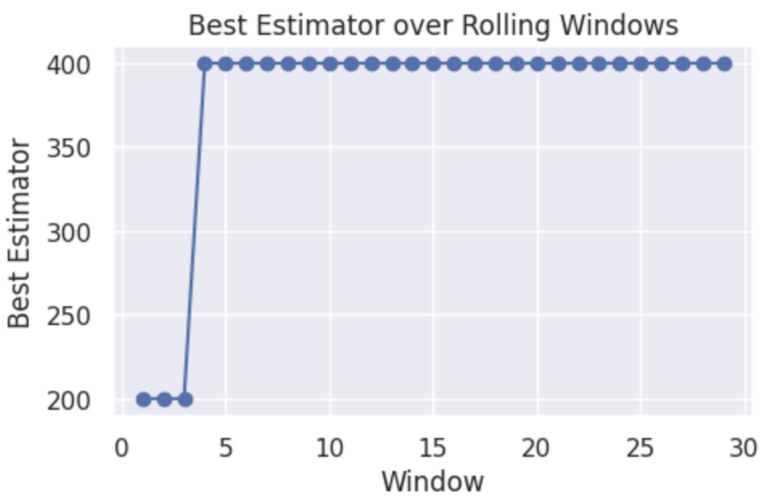


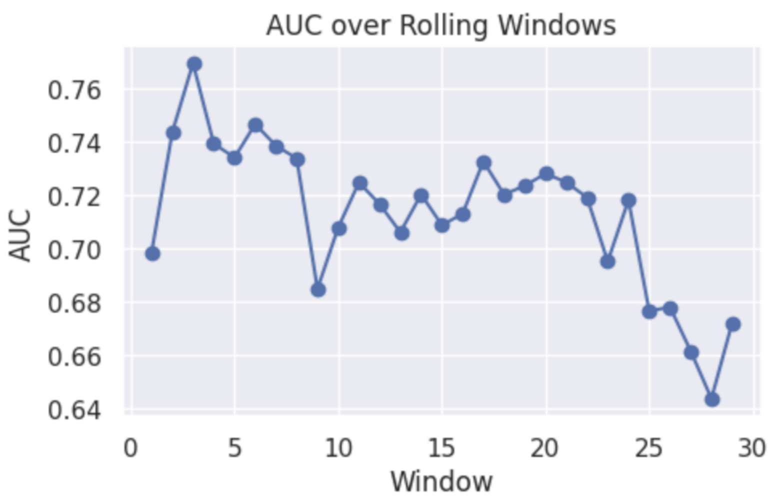
Window Metrics

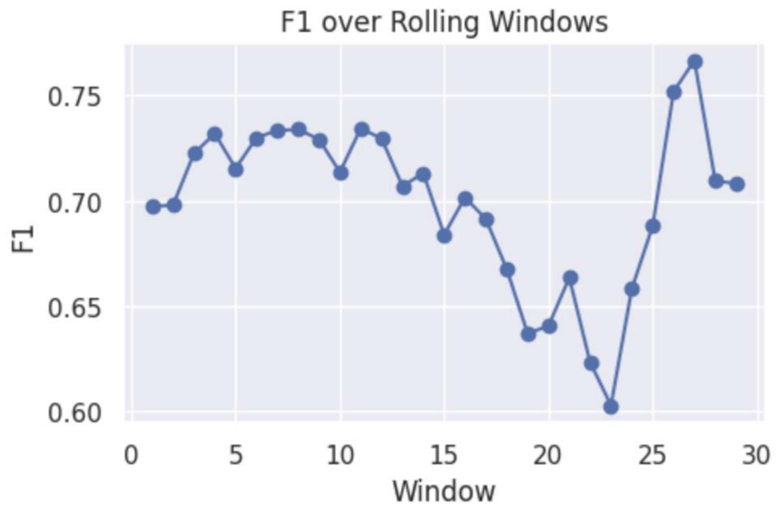












Advantages

- **Random Forest handles the mix of numerical and categorical variables well**
- **Robust to outliers and skewed distributions**
- **Captures non-linear feature combinations**
  - Very applicable to article popularity
- **Reduces variance and makes more consistent predictions over time**

Deployment

- **Useful for news organizations, advertisers, and editors**
- **Guide resource allocations, targeted advertising, and editorial decisions**
  - Increase engagement and revenue
- **Should track engagement metrics and compare to baseline with no prediction**
- **Deploy through content management system or via an API**
  - Immediate feedback on popularity probability and feature importance
- **Collecting additional data to increase model accuracy**
- **Update on a frequent basis using rolling windows approach**

Comparing Our Models

Metric	Logistic Regression*	KNN	Random Forest
ROC AUC	0.7381	0.6814	0.7122
Accuracy	0.6749	0.6355	0.6581
Precision	0.6781	0.6667	0.6719
Recall	0.7438	0.6339	0.7374
F1	0.7094	0.6499	0.7031

***logistic regression model was redone with an 80/20 split**

```
LR vs RF (McNemar):
McNemar p-value (LR vs RF): 0.015661  -> significant @95%? True
LR vs RF (Bootstrap ΔF1 = F1(LR) - F1(RF)):
ΔF1 95% CI [-0.0376, -0.0104] (mean -0.0240)  -> SIGNIFICANT
```

Conclusions

Random Forest was the best performer on the 80/20 split (F1 = 0.71, AUC = 0.74)

Logistic Regression is still valuable as an interpretable, fast baseline with decent recall (~0.70)

Statistical tests confirmed RF significantly outperforms Logistic Regression and KNN at 95% confidence

KNN underperformed relative to both LR and RF, with weaker recall and F1

Moving forward

Richer data sources

We could include social media engagement, user demographics, and temporal trends to improve predictive power.

Explore advanced models

XGBoost or Neural Networks for potentially higher accuracy and better handling of certain interactions.

Enable adaptive learning

Use rolling windows and other learning approaches so models stay current with changing interests and trends.

Business & Industry Impact

News organizations and advertisers can leverage predictions to increase engagement.


dente, era nell'ordine (...)
mento non sono il gesto di Bara-
Ma ci sarà quello che trascinerà
bile è l'amido, ossia il pane, la
pasta, la pizza, precisamente i
zione di coraggio e civiltà
dell'Occidente.
Raffa e Fontana
alle pagine 7 e 8

Thank you