

Online News Popularity
Rohan Gupta

Introduction

For this assignment, we will explore the Online News Popularity dataset and the Online Shoppers Purchasing Intention dataset.

The first dataset contains over 30,000 news articles published by Mashable between 2013 and 2014. It provides several attributes describing the characteristics of online news articles. In total, there are 61 attributes: 58 predictive, 2 non-predictive, and one target class called *shares*, which represents the number of times a news article was shared. For this dataset, our machine learning models will aim to predict the popularity of a news article by calculating the number of shares using all or some of the attributes. Since the target class is a continuous value, we will treat this problem as a regression task. Metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE) will be used to evaluate model performance. Notable attributes in this dataset include the title length, content length, news type (e.g., Entertainment, World, Technology), and the number of negative and positive words in the content.

We will use the following machine learning algorithms for both datasets: Support Vector Machine (SVM), K-Nearest Neighbors (K-NN), and Decision Tree.

Objectives

This project aims to create machine learning models using Python to analyze real-world datasets. We will assess the accuracy and runtime of each model. The two datasets will be split into training and test subsets. Each model will be evaluated against the test set, and various performance metrics will be calculated for comparison. Based on these metrics, recommendations will be made on which model performed the best.

Dataset 1: Online News Popularity

Exploratory Data Analysis

We began by examining the distribution of our target feature, as shown in Figure 1. Immediately, we observed significant outliers in the dataset. Some news articles became extremely popular, while most remained within a similar range. We will address these outliers during the data-cleaning process.

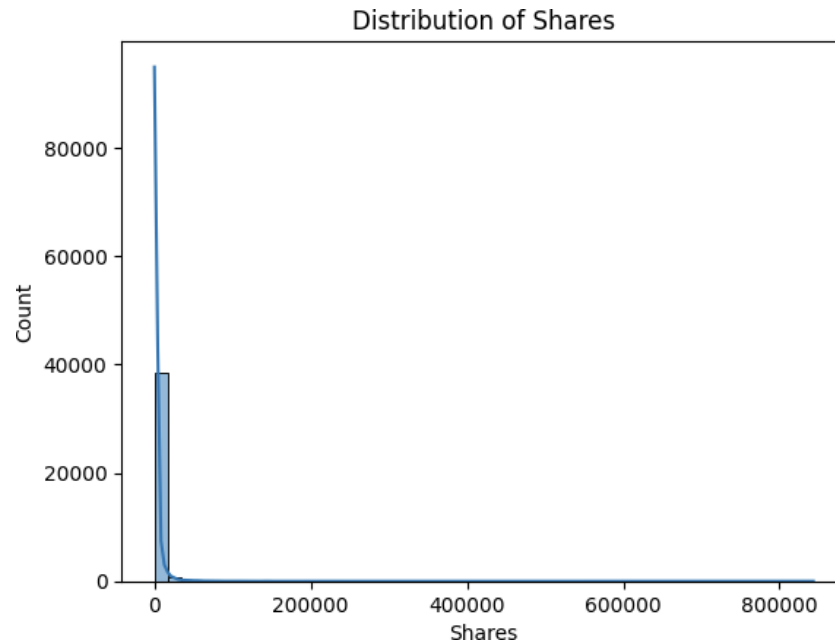


Figure 1. Distribution of Shares

Next, we explored the relationship between the popularity of news articles and the day of the week they were published. Unsurprisingly, news published on weekends tend to be the most popular. However, it is interesting to note that fewer articles are published on weekends. As a result, this attribute may prove to be highly influential in our model.

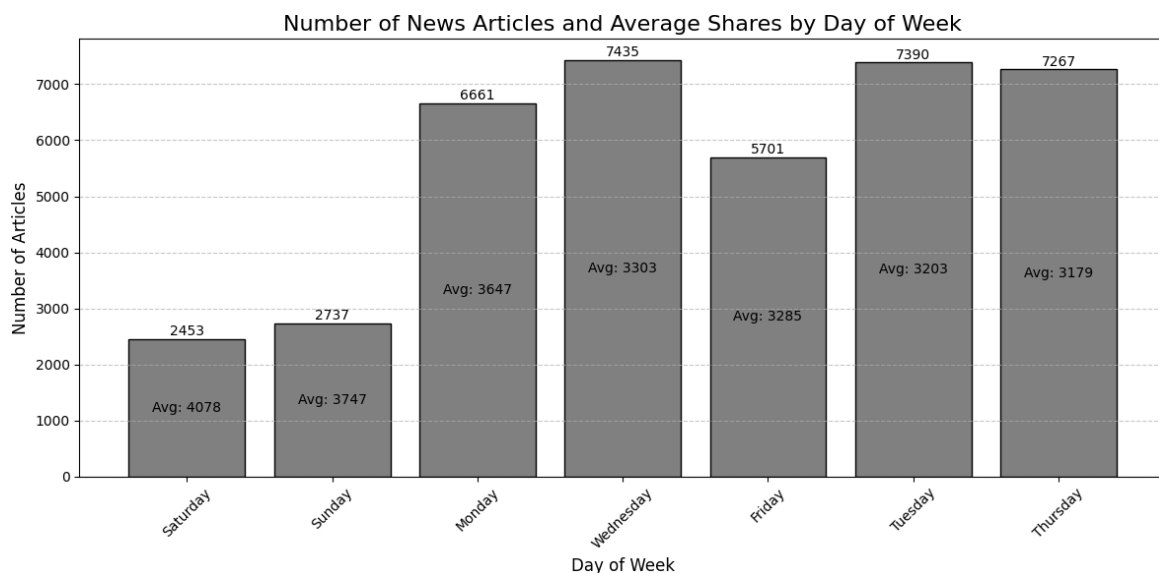


Figure 2. Average Popularity and Number of News by day of the week

We also examined the correlation between popularity and the data channel. We found that most channels (Tech, Entertainment, World, and Business) have similar numbers of shares. However, news related to Social Media and Lifestyle are significantly less popular.

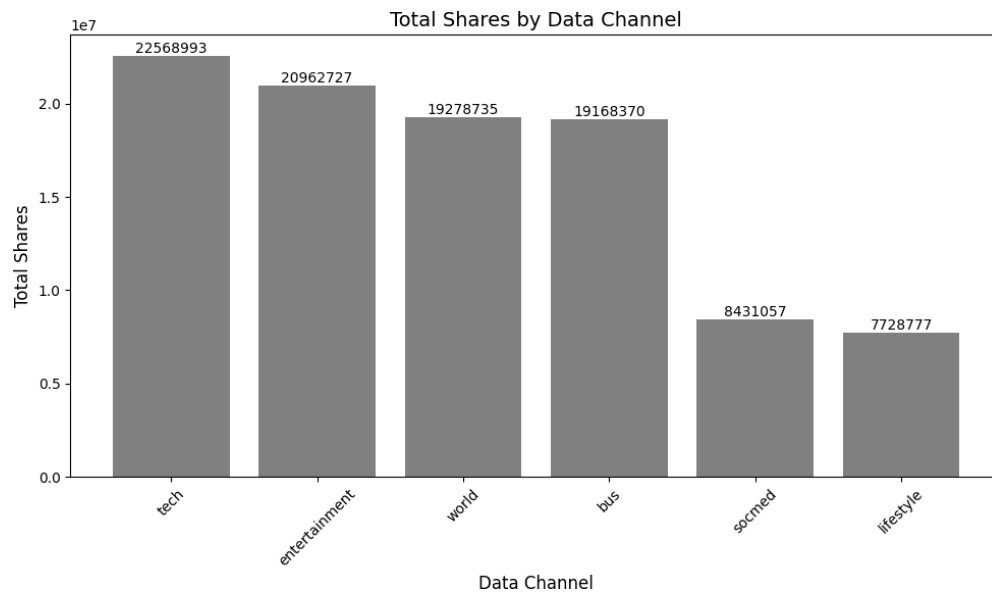


Figure 3. Number of shares by data channel

The scatter plot below illustrates the relationship between the number of words in an article and the number of shares. This graph helps us assess whether the length of a news article impacts its popularity. As seen in Figure 4, shorter articles tend to be more popular.

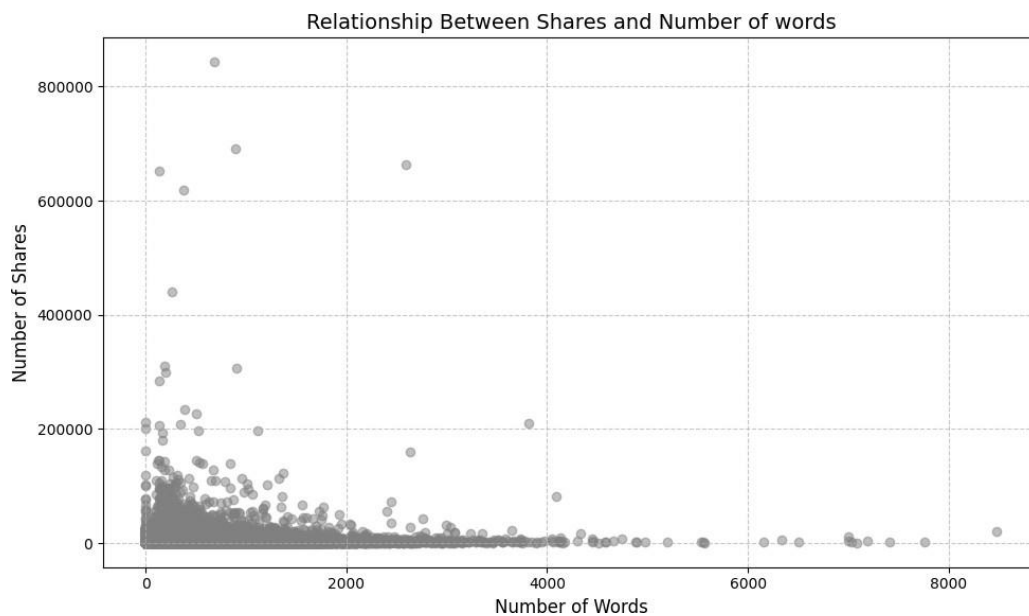


Figure 4. Number of shares versus number of words in an article

Lastly, we compared the positive and negative polarity scores side by side. We found that news articles with higher negative polarity tend to perform better than those with high positive polarity. This is understandable, as articles with more negative language may provoke a stronger reaction from users, leading to more shares, as shown in Figure 5.

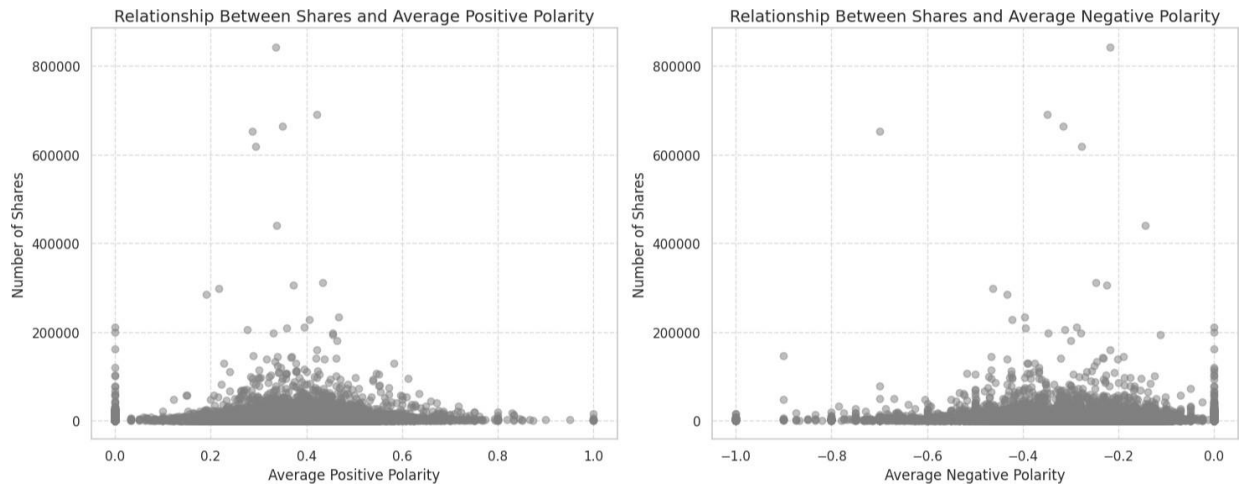


Figure 5. Number of Shares versus Average Positive and Negative Polarity

Data Cleaning Techniques

For our data cleaning process, we used a combination of built-in functions from the pandas package and statistical techniques to filter out outliers from our dataset.

First, we dropped records containing null values and duplicates using pandas. Next, for the following attributes, `n_tokens_title`, `n_tokens_content`, and `shares`, we calculated the quartiles (Q1 and Q3) of their distributions and then computed the Interquartile Range (IQR) by subtracting Q1 from Q3. The lower bound was defined as Q1 minus 1.5 times the IQR, and the upper bound was defined as Q3 plus 1.5 times the IQR. Any values falling outside these bounds were considered outliers and removed from the dataset. The updated distributions of the aforementioned columns are shown in Figure 6.

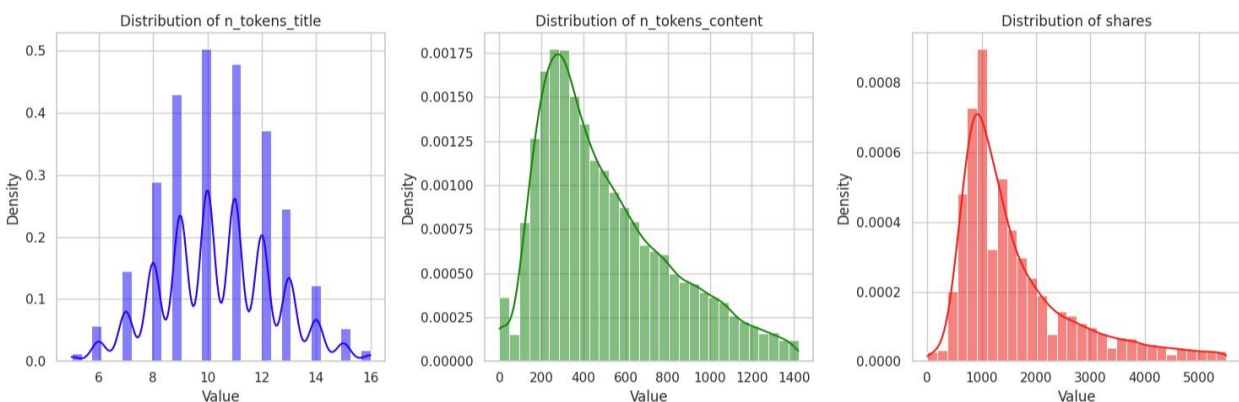


Figure 6. Updated distributions for `n_tokens_title`, `n_token_content`, `shares`

After data cleaning, the size of our dataset was reduced to 28.5k records. We then used a built-in method from the scikit-learn package to split the data into training and test subsets, allocating 70% to training and 30% to testing. This method enabled us to randomly split the cleaned dataset, ensuring a more reliable split.

Machine Learning Algorithms

Three algorithms were used: Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Decision Tree. All models used the same set of features to ensure a fair comparison.

Since this dataset contains over 60 features, we decided to run two models for each algorithm. The first model used all suitable features, except for the target feature, while the second model utilized a selected set of 19 key features identified during our exploratory data analysis.

The rationale behind these choices stems from two key considerations. First, selecting fewer features may help prevent overfitting, as using all attributes could lead to a complex model that fits the training data too closely. Second, a model with more features would result in longer training and prediction times. Therefore, a simpler model that uses fewer features and requires less time is preferable.

Since the target variable (shares) is numerical rather than categorical, we will treat this as a regression problem. Accordingly, we will evaluate the models using the following metrics: Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE).

Support Vector Machine

The Support Vector Machine (SVM) algorithm is well-suited for binary classification problems, particularly when the classes are linearly separable. Additionally, it performs effectively on datasets with a large number of attributes, making it an appropriate choice for this dataset.

During our training, SVM had the longest training time compared to other algorithms, taking 33.97 seconds when using all features and 15.54 seconds with a selected subset of features. Despite the extended training time, SVM delivered the most accurate results, achieving the lowest RMSE of 1,074.25. However, our exploratory data analysis indicated that the dataset might not be linearly separable. This raises the possibility that the algorithm's superior performance could be attributed to overfitting. To validate the model's robustness, it would be ideal to evaluate its performance on new, unseen data and assess how well it generalizes in real-world scenarios.

K-Nearest Neighbour

The K-Nearest Neighbors (KNN) algorithm performs exceptionally well on small to medium-sized datasets where the distance between data points can be meaningfully measured. This makes it a suitable choice for our model, as the dataset is relatively small, and most of the attributes allow for distance-based calculations.

Training the model was highly efficient, taking only 0.021 seconds when utilizing all features and just 0.008 seconds with the selected subset of features. Also, we configure the models to use 5 neighbors to make a prediction. However, KNN tends to struggle with high-dimensional data, which may explain why the RMSE remained consistent across both models: 1,111.46. Despite this limitation, KNN emerged as the second-best performing model, following closely behind SVM.

Decision Tree

The Decision Tree algorithm is well-suited for datasets with non-linear relationships and works particularly well with categorical data, though it can also handle numerical data effectively. A key observation with this algorithm is that trees with too many branches tend to overfit by capturing subtle nuances in the data, leading to narrow predictions and potentially incorrect results. This poses a challenge for our dataset, as even the models with selected features contain 19 attributes, which may impact the Decision Tree's performance.

In our case, the model using the selected features performed better, achieving an RMSE of 1,471.50 and requiring only 0.006 seconds to train. However, the Decision Tree algorithm delivered the poorest performance among all the models we evaluated.

Model Comparison

The table below presents the results of training all of our models. For each algorithm, we created two models: one using all the features available in the dataset and another using only 19 selected features, which were chosen based on our exploratory data analysis. In general, models utilizing all features took twice as long to train. However, since these models were trained on Google Colab, the runtime was also influenced by the availability of the service. Despite this, SVM delivered the best performance, followed by KNN, with the Decision Tree performing the worst. The table below shows the scores for MSE, RMSE, and MAE.

Model	Training Time	Prediction Time	MSE	RMSE	MAE
SVM (All features)	33.972	6.456	1,154,009.87	1,074.25	702.42
SVM (Selected features)	15.539	3.817	1,198,438.44	1,094.73	720.12
KNN (All features)	0.021	2.791	1,235,350.24	1,111.46	811.12
KNN (Selected features)	0.008	2.474	1,235,350.24	1,111.46	811.12
Decision Tree (All Features)	2.098	0.013	2,173,457.56	1,474.27	1,049.55
Decision Tree (Selected Features)	0.624	0.006	2,165,310.07	1,471.50	1,053.58

Dataset 1 Conclusion

While SVM performed the best on the training data, our exploratory data analysis revealed that the data is not linearly separable, raising doubts about the model's effectiveness. Without access to unseen data that was not used during training, it is impossible to confirm whether this model will perform similarly with real-world data. The model is likely overfitting to specific nuances of the dataset. Additionally, the training time for SVM was considerably longer than that of the other models.

Therefore, we recommend proceeding with the KNN model using only the selected features. This model offers several advantages, including a very short prediction time of 2.474 seconds. Furthermore, KNN produced results that were very similar to those of SVM, but with significantly less training time and without raising concerns about overfitting. The main drawback of KNN is that its performance may degrade with larger datasets. Thus, if retraining with a larger dataset becomes necessary in the future, our second recommendation would be to use SVM with selected features, as it produced nearly identical results to the full-featured SVM model but required half the time for both training and predictions.

