Machine Learning

Course Project Report (Classification)

(Draft - 02, Team No: 01)

Title of the project: Online News Popularity

Student 1 : Sarang Galada, sarang.g-25@scds.saiuniversity.edu.in
Student 2 : R Sai Pranav Aadhitya, saipranavaadhitya.r-25@scds.saiuniversity.edu.in

ML Category: Classification

---

1. Introduction

The Online News Popularity dataset contains a variety of features about news articles published by digital media platform Mashable over a period of 2 years. The goal here is to predict the popularity of the news article measured through the number of times the article has been shared on social media platforms. Framed as a classification problem, we implement various binary classification models to predict whether a news article is popular or not.

The significance of trying to model the relationship between the numerous features of an article and its popularity is that it helps authors understand what makes news articles popular and thereby equips them with the knowledge to improve their popularity.

- Problem Statement: (Binary Classification)

  Predicting whether news articles are popular or not based on the values of their various features.

2. Dataset and Features

- Details of the dataset:

This project makes use of the Online News Popularity dataset from University of California, Irvine's Machine Learning Repository, sourced from a Portugal based research team, K. Fernandes et al. The dataset consists of 39,644 rows and 61 columns. The rows represent instances of new articles, while the columns depict various features of the articles - such as number of words, videos, images, hyperlinks in the article, average word length, tags, to name a few. Of these 61 columns, 59 are the input features for our learning model, with the 61st column 'shares', signifying the number of times an article has been shared on social media (measure of popularity) being the target variable we wish to predict. The first column 'url' is simply to identify the url of each article instance in the dataset.

- <u>Exploratory Data Analysis</u>:

Initially, the dataset contains 39,644 rows and 61 columns.

	url	timedelta	n_tokens_title	n_tokens_content	n_unique_tokens	n_non_stop_words	n_non_stop_unique_tokens	num_hrefs
0	http://mashable.com/2013/01/07/amazon-instant-...	731.0	12.0	219.0	0.663594	1.0	0.815385	4.0
1	http://mashable.com/2013/01/07/ap-samsung-spon...	731.0	9.0	255.0	0.604743	1.0	0.791946	3.0
2	http://mashable.com/2013/01/07/apple-40-billio...	731.0	9.0	211.0	0.575130	1.0	0.663866	3.0
3	http://mashable.com/2013/01/07/astronaut-notre...	731.0	9.0	531.0	0.503788	1.0	0.665635	9.0
4	http://mashable.com/2013/01/07/att-u-verse-apps/	731.0	13.0	1072.0	0.415646	1.0	0.540890	19.0

5 rows × 61 columns

Apart from the 'url' column which is of type Object and not relevant to our analysis, and the target variable 'shares' which is of type Int64, the remaining 59 input attributes are of type Float64. The data neither contains any missing values nor any duplicate data instances.

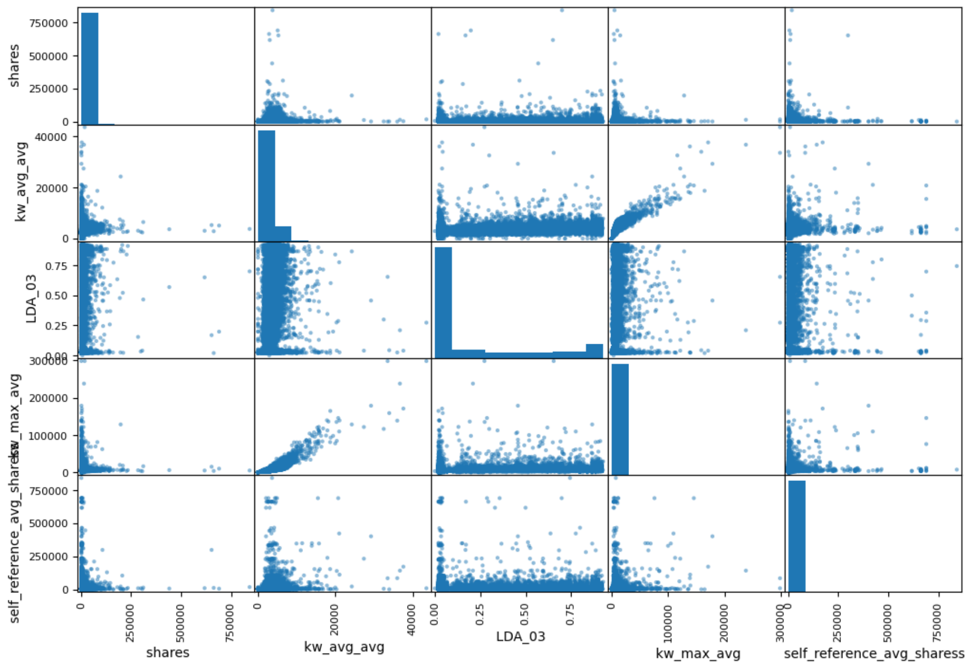
The target variable 'shares' has a right-skewed distribution, with a minimum value of 1, a median value of 1400, a mean value of 3395.38 and a maximum of 843300.

When inspecting the correlation values between the input features and the target variable, we see that apart from '*kw_avg_avg*', which has the highest correlation value of 0.110413, all other features have correlation values below 0.1. This implies that there is very low linear dependence between the attributes and the target variable.

```
# Observing the correlation values of features wrt target variable 'shares'
corr_values = data.corr(numeric_only=True)[' shares'].sort_values(ascending=False).drop(' shares')
corr_values
```

```
kw_avg_avg                    0.110413
LDA_03                        0.083771
kw_max_avg                    0.064306
self_reference_avg_sharess    0.057789
self_reference_min_shares     0.055958
self_reference_max_shares     0.047115
num_hrefs                     0.045404
kw_avg_max                    0.044686
kw_min_avg                    0.039551
num_imgs                      0.039388
global_subjectivity           0.031604
```

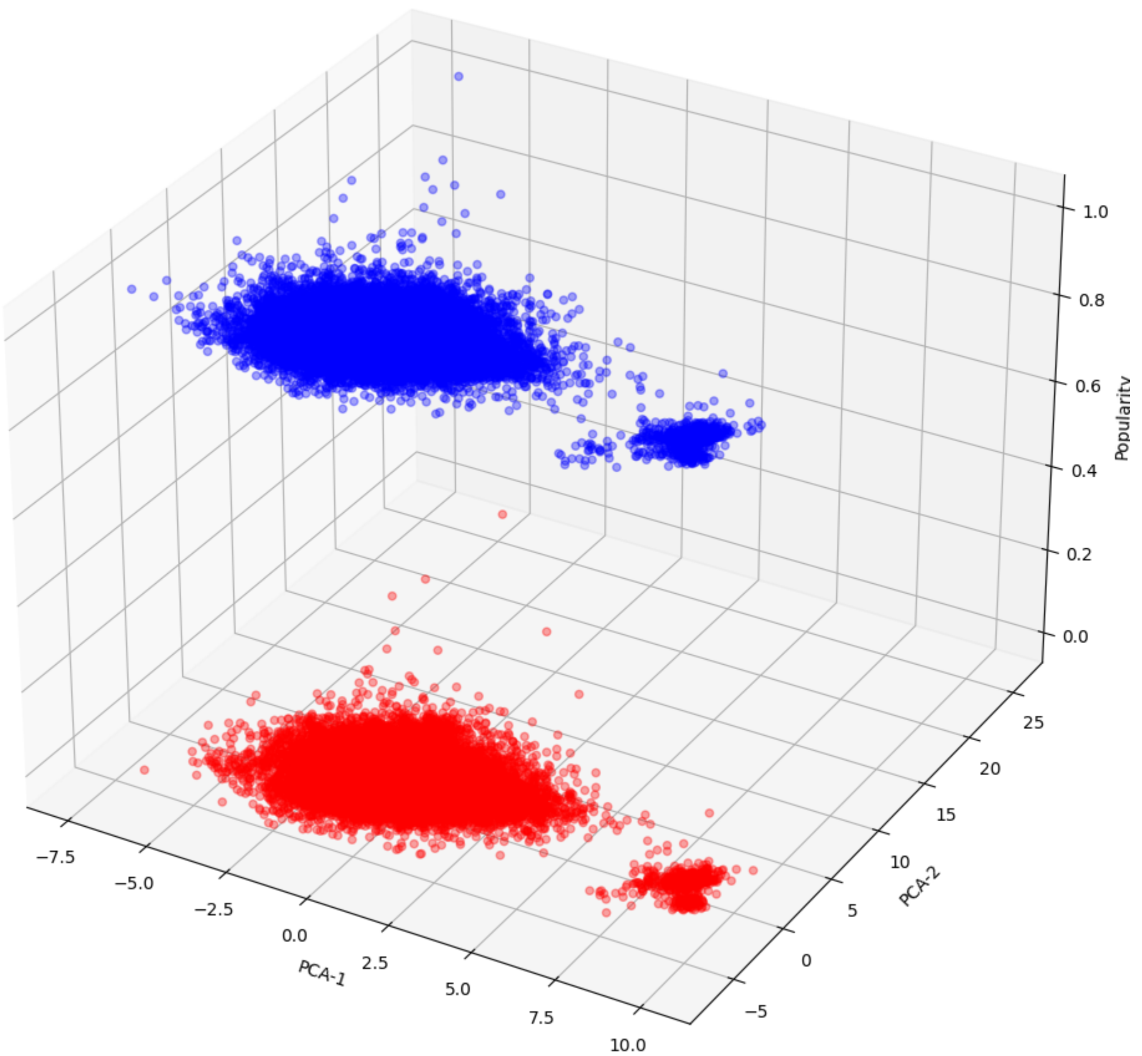
The first column of the following scatter matrix shows this lack of linear dependence.



- <u>Dimensionality Reduction & Data Visualisation</u>:

  We use dimensionality reduction techniques to reduce the number of features in the data from 59 to 2, so that the point cloud can be visualized in 3D when the features are plotted against the target variable 'shares'. In this project, we only implemented Principal Component Analysis (PCA), since t-SNE was very computationally intense even after using PCA to reduce the data first. The below plot shows the distribution of the data points with respect to the 2D PCA features, differentiated by colour based on their class - blue implies 'popular' and red, 'unpopular'.

Visualisation of PCA reduced data in 3D



3. Methods

Following are the various methods used in this project.

3.1 Baseline - Logistic Regression

The logistic regression model calculates the probability that a given input belongs to a particular class, and is the most popular baseline classification approach in machine learning. It is called logistic regression because it is based on the logistic function, a type of sigmoid function, which transforms values to between the range 0 and 1, to indicate the likelihood of a certain combination of features belonging to a particular class. It accomplishes this by mapping the input features to a real-valued output using a linear combination of the features. This linear combination is then transformed using the logistic function to obtain a probability value between 0 and 1.

We decided to make only two classes for our output data, since the accompanying research paper had performed their experiments with binary classification, and we wanted to maintain consistency with that. We performed logistic regression on the full training dataset, standardized, and the PCA reduced (20) dataset. Logistic regression on the regular data gave us an accuracy of approximately 0.59, standardized data gave us ~0.655, and the experiment with PCA reduced data gave 0.63728. The observation that logistic regression performs better on standardized and PCA data compared to the regular one becomes quite clear here with respect to our specific dataset.

3.2 Support Vector Machine

- <u>Linear SVM and Kernel SVM</u>

Linear SVC finds the optimal hyperplane by identifying the support vectors that lie closest to the decision boundary. It is different from the SVC class with linear kernel as it uses one-vs-rest classification, while the latter uses the one-vs-one approach. The algorithm aims to maximize the margin, i.e., the distance between the support vectors and the decision boundary, while minimizing classification errors. Although it has multiple hyperparameters, we used only the tolerance and the C parameters, which is the hardness/softness parameter. Finding very little correlation between the input data and the output, we decided to give some room for misclassification, so that a decent score may be achieved. We used LinearSVC on the regular data, standardized data, and the PCA reduced data of 20 features. On the regular data, we got an accuracy score of ~0.535, and obtained scores of ~0.654 and ~0.638 respectively. LinearSVC clearly performed better on standardized and PCA reduced data in our case. On linear kernel SVC, that is, the SVC class with `kernel='linear'`, we received an accuracy of approx. 0.625.

It is to be noted that we used a reduced PCA training set of 20 components to do our kernel SVM experiments, since the size of our data is quite big, and the running time for each SVM was never ending. We halved the size of our training data to accommodate for the computational requirements of running our experiments. LinearSVC class on the other hand was comparatively much faster and gave quick results with all the datasets.

- Polynomial and Radial Basis Function kernels

The Polynomial Kernel SVC works by mapping the original data into a higher-dimensional feature space using a polynomial function. This allows the SVC to find a linear decision boundary in a higher dimension that is linearly separable, and this boundary can be considered equivalent to a polynomial boundary in the original (lower) dimension. The degree of the model is same as the degree of the polynomial function used. Higher degrees can capture more intricate patterns in the data, while lower degrees may result in underfitting. Therefore, the choice of the polynomial degree should be carefully chosen, given the nature of our dataset. Since the number of features were quite high, we chose to go with a degree 3 so that no extreme fitting scenario is achieved, as well as a good enough model is mapped too. The accuracy obtained was approximately 0.648.

The Radial Basis Function is a kernel function that once again maps the data onto higher dimensions by selecting random data points and plotting gaussian distributions over them and comparing the similarity between distributions to classify. Once an appropriate division is discovered based on the given hyperparameters, it brings it back to its original dimensions, but this time with a clear division of classes known. We used `C=2` and `gamma='scale'` as the hyperparameters for training the model after tuning them. The accuracy obtained was ~0.645.

3.3 Decision Tree Classification

Decision trees are a non-parametric machine learning technique used for both classification and regression problems. (Non-parametric implies that they have a variable number of parameters). In both prediction performance as well as computation intensity, Decision Trees are considered superior to the previous techniques used in this project. They work by analysing the features of the data and develop a decision model in the form of a binary tree, using which it makes predictions. The algorithm used to train Decision Trees is the *CART Algorithm*.

The Classification And Regression Tree (CART) Algorithm is a greedy algorithm used to train Decision Trees. It works by recursively splitting the training data into two subsets with respect to a feature and a corresponding threshold, such that the *Gini impurity* of the split is minimised.

The *Gini* value measures the impurity within a node, ie. it captures the ratio of class instances among all training instances within a particular node. It is calculated using the formula:

$$G_i = 1 - \sum_{k=1}^{n} {p_{i,k}}^2$$

3.4 Random Forest Classification

Random Forest classification applies the Ensemble Learning approach of Bagging to a *forest* (ie. collection) of Decision Trees. In Random Forest classification, numerous Classification Trees are trained and the mode value of their predictions is selected, thereby improving prediction accuracy and curtailing overfitting. Random Forest models are considered among the best Machine Learning techniques, as they employ the power of Ensemble Learning on the high performance of Decision Trees.

4. Experiments & Results

4.1 Protocol

- Data Splitting:

  The data was split into 80% training and 20% testing, maintaining a random seed value of 42 throughout all experiments for reproducibility. To be able to meet the computational requirements of all the experiments, we halved the size of our Training data such that it now has 15,857 rows from the previous 31,715.

- Data Preprocessing.

  - Data cleaning:

    The dataset was searched for missing values and duplicates, which were dropped if found. In the case of this dataset, there were no missing values or duplicates..

  - Feature Scaling - Standardization:

    In addition to experimenting with the regular data, we were interested in reducing the error further by applying standard scaling. We hence scaled the values of X, and applied the fit functions on the scaled dataset. We observed that this indeed did affect the results in a large way - There was observed a consistent increase in the accuracy scores of the experiments run when compared (wherever comparable), with the training done on the regular data.

  - Dimensionality Reduction - Principal Component Analysis (PCA):

    Principal Component Analysis, or PCA, reduces the features of the data by finding their directions of maximum variance and projecting the data along those directions, thus preserving information of the original attributes and also reducing the size of the dataset for better computations and visualizations. In our case, we reduced our dataset from 59 features to 20 features, so that all experiments could handle the size computationally. Similar to the results obtained with standardization, wherever comparable, the accuracy scores showed a general increase in relation to the scores obtained when the training was done on the regular data.

4.2 Results

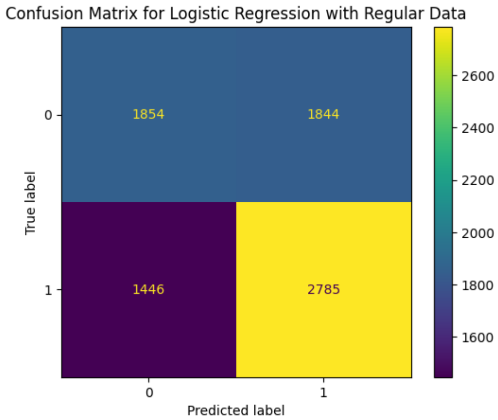
- Classification Accuracy

	Regular Data	Standardized Data	PCA Reduced Data
Logistic Regression	0.58507	0.65456	0.637410
LinearSVC	0.52882	0.65456	0.636400
Linear kernel SVC	-	-	0.624800
Polynomial kernel SVC	-	-	0.638290
RBF kernel SVC	-	-	0.644220
Decision Tree	0.64296	0.64296	0.611930
Random Forest	0.650271	0.65065	0.636524

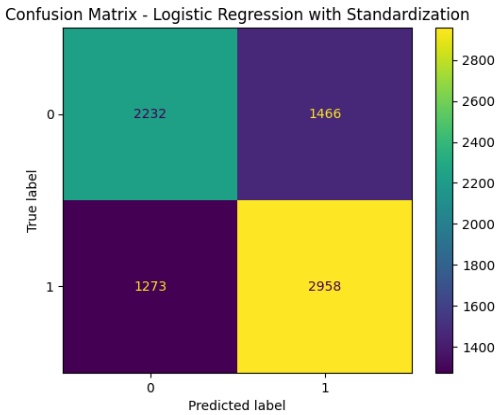
- Cross-validated F1-scores

	Regular Data	Standardized Data	PCA Reduced Data
Logistic Regression	0.59376 +/- 0.01425	0.65394 +/- 0.00577	0.63302 +/- 0.00167
LinearSVC	0.43682 +/- 0.04735	0.65403 +/- 0.00607	0.63468 +/- 0.00315
Linear kernel SVC	0	0	0.62253 +/- 0.00251
Polynomial kernel SVC	0	0	0.64035 +/- 0.00597
RBF kernel SVC	0	0	0.64891 +/- 0.0074
Decision Tree	0.64028 +/- 0.00744	0.64028 +/- 0.00744	0.62063 +/- 0.00349
Random Forest	0.65733 +/- 0.00784	0.65739 +/- 0.00765	0.63193 +/- 0.00523

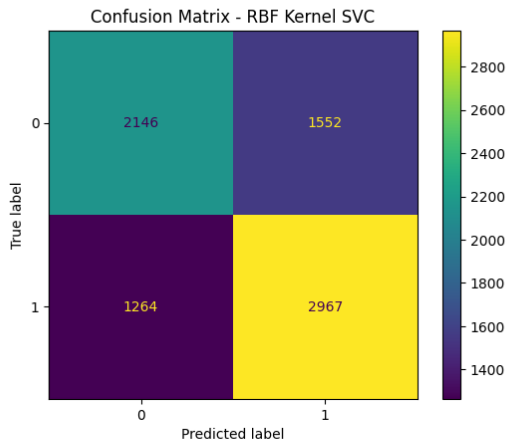
- Confusion Matrices & Classification Reports for select experiments



	precision	recall	f1-score	support
0	0.56	0.50	0.53	3698
1	0.60	0.66	0.63	4231
accuracy			0.59	7929
macro avg	0.58	0.58	0.58	7929
weighted avg	0.58	0.59	0.58	7929



	precision	recall	f1-score	support
0	0.64	0.60	0.62	3698
1	0.67	0.70	0.68	4231
accuracy			0.65	7929
macro avg	0.65	0.65	0.65	7929
weighted avg	0.65	0.65	0.65	7929



	precision	recall	f1-score	support
0	0.63	0.58	0.60	3698
1	0.66	0.70	0.68	4231
accuracy			0.64	7929
macro avg	0.64	0.64	0.64	7929
weighted avg	0.64	0.64	0.64	7929

5. Discussion

When judging based on Cross-validated F1-scores, Random Forest model gave the best score of 0.65739 on regular and standardised data, only marginally ahead of Linear Support Vector Classifier at 0.65403 and Logistic Regression at 0.65394. Decision Trees gave a CV score of 0.64028. Since Decision Trees are indifferent to standardisation, we didn't notice any improvement in the performance of Decision Tree experiments with standardised data over regular data, although Random Forest experiments showed a minor increment in scores for standardised data. In contrast, other methods performed quite average on the regular data.

The PCA data yielded scores slightly lower than that of the standardised data, probably due to the drastic reduction in dimensions of the dataset employed for it. When considering the PCA data alone, the RBF and Polynomial kernel Support Vector Classifiers gave the best scores of 0.64891 and 0.64035 respectively.

Extrapolating this, one could argue that the RBF and Polynomial kernel SVCs are the best performing Machine Learning models for the given dataset, among those used in this project. Nevertheless, based on the experiments we conducted, the difference in performance between the various models employed is marginal, especially on the standardised and PCA reduced data.

6. Conclusion

- Not required in first draft

7. References

[1] K. Fernandes, P. Vinagre and P. Cortez. A Proactive Intelligent Decision Support System for Predicting the Popularity of Online News. Proceedings of the 17th EPIA 2015 - Portuguese Conference on Artificial Intelligence, September, Coimbra, Portugal.