Online News Popularity Project Report

Group 6

IE7300 Statistical Learning for Engineers SEC 03 Spring 2023

Professor: Ramin Mohammadi

Group Members
Phanindra Raja Varma Gadiraju
Venkata Sasank Jonnalagadda
Sai Sruthi Kalidindi
Henna Shah

Abstract:

Nowadays, many news channels utilize machine learning models to predict whether a news article will be popular or not. These models use various techniques such as linear regression to forecast the number of shares an article might receive. To classify an article as popular or not, models like logistic regression, support vector machines and neural networks are used. We also applied Linear Regression for continuous target variable

The aim of this study is to determine the best machine learning model for predicting the number of reshares an article will receive and whether it will be popular or not. The study utilizes different data mining techniques and feature engineering to extract relevant features from the dataset obtained from the UCI repository.

To evaluate the performance of the regression model, root mean square error (RMSE) and mean absolute error (MAE) are used, while accuracy is used to evaluate the classification models for predicting popularity. Neural Networks outperformed all other classification models with an accuracy rate of 98%.

This study's results can be applied to social media platforms as a predictive tool before publishing an article to determine its potential popularity.

Introduction:

In today's modern world, when the internet has mostly replaced conventional newspapers and periodicals, it is critical for news outlets to forecast the popularity of their pieces on the internet. Because marketing consumes a big percentage of a company's cash, it is critical to have an effective method of gauging an article's popularity. This can be accomplished by developing a model capable of identifying the characteristics that influence the output and predicting the score of a specific article or blog.

The goal of this project is to examine data using data mining techniques and develop various machine learning models to predict the popularity of a news story. The program will find the most essential elements that contribute to an article's popularity and forecast the score based on these qualities after analyzing the data. This will enable news outlets to make more educated marketing decisions and improve the effectiveness of their content.

Data Source:

The data was taken from the UCI Machine learning repository.
https://archive.ics.uci.edu/ml/datasets/online+news+popularity
The data consists of statistics of articles that were published by Mashable.

Problem Definition:

There are several ways to estimate the popularity of an online news article, such as the number of views, clicks, likes, dislikes, and re-shares. For this project, we will be focusing on the number of shares to determine an article's popularity.

As part of this project, we will be making two assumptions. First, we will assume that there is no direct relationship between an article's popularity and the article itself or the current events happening in the world. Second, we will ignore the quality of the article and the reader's affinity towards it.

The project will aim to identify the contributing factors and attributes that make an article popular and predict the factors that influence the number of shares an article receives. The study will also focus on identifying the best machine learning models to predict and classify an article as popular based on the extracted features.

By analyzing the data, we will be able to identify the most important predictors that affect the shares of an article and provide insights on the factors that make an article popular. The project's results can be applied to various social media platforms to predict an article's potential popularity and help news channels optimize their marketing strategies.

Data Description & Cleaning:

The dataset utilized for this study came from the UCI Machine Learning Repository and comprises information about Mashable articles. It has 39797 records and 61 attributes, including 58 predictive factors and two non-predictive variables (URL and time), as well as a target variable, the number of shares.

The predictive factors include both numerical and categorical fields, such as the length of the article, the length of the title, the count of unique words, stop words, the average token length, the type of channel in which the item was published, and the day of the week in which the article was published. Furthermore, sentiment analysis was done on the textual data, yielding sentiment polarity scores for the articles and titles, as well as positive and negative word count data, all of which are included as predictive factors.

The dataset provides a thorough breakdown of the factors that influence an article's popularity, and sentiment analysis adds an extra layer of insight into how the language used in the post impacts its popularity.

Sample records from the dataset will be investigated as part of the study to understand the distribution and structure of the data and how it can be utilized to develop machine learning models to predict and classify article popularity.

The dataset used for this project was relatively clean, with a low number of missing or null values. This means that the data is relatively complete and can be used for analysis and modeling without significant data cleaning or imputation.

Having a clean dataset is essential for accurate analysis and modeling, as missing or null values can introduce bias and reduce the effectiveness of machine learning models. Therefore, the relatively clean nature of the dataset is a positive attribute that will allow for more accurate and reliable predictions of article popularity.

Model Description:

Several machine learning models will be trained on a training dataset as part of this project and will be used to estimate the number of shares for each news story and classify articles as popular or not. These models will comprise regression models, which predict numerical values, as well as classification models, which predict categorical values.

Once these models have been trained on the training dataset, they will be evaluated on a separate validation dataset to determine their accuracy and effectiveness. The validation dataset is a separate subset of the overall dataset that has been reserved specifically for testing the performance of the models.

By testing the models on a separate validation dataset, it is possible to determine how well they can generalize to new, unseen data. This is an important step in the model development process, as it ensures that the models are not simply memorizing the training data but are instead learning to identify patterns and relationships that can be applied to new data.

Linear Regression:

Linear regression is a statistical technique used to model the relationship between a dependent variable and one or more independent variables. It assumes a linear relationship between the independent variables and the dependent variable and uses this relationship to make predictions about the dependent variable.

In some cases, the independent variables may be highly correlated with one another, which can cause issues with the accuracy of the linear regression model. In these cases, Ridge regression can be used as a regularization technique. Ridge regression adds a penalty term to the regression equation that limits the size of the regression coefficients and prevents overfitting. This helps to reduce the impact of the correlated independent variables on the model's predictions.

Lasso regression is another regularization technique that can be used for linear regression. Like Ridge regression, it adds a penalty term to the regression equation to limit the size of the regression coefficients. However, Lasso regression uses an L1 penalty term, which has the additional benefit of performing feature selection by setting some of the coefficients to zero. This means that Lasso regression can be used to identify the most important independent variables for predicting the dependent variable.

Logistic Regression:

Logistic Regression is a classification method that predicts the likelihood of a binary result (whether an event will occur or not). It is assumed that the independent variables have a linear connection with the logarithm of the binary outcome.

When we wish to forecast the likelihood of a new observation belonging to a given class based on its characteristics or predictors, we can use logistic regression. For example, based on demographic information and purchase history, we can apply logistic regression to forecast whether a consumer will make a purchase or not.

Unlike linear regression, which assumes a continuous dependent variable, logistic regression assumes a binary dependent variable. It also presupposes that the independent variables are not significantly correlated with each other (i.e., there is no multicollinearity) and that the independent variables have a linear connection with the logarithm of the probability of the binary result.

Because of its simplicity, interpretability, and efficacy in predicting binary outcomes, logistic regression is a popular categorization method. It is frequently utilized in many industries, including healthcare, finance, and marketing.

Hard Margin SVM:

Hard margin SVM (Support Vector Machine) is a type of supervised machine learning algorithm used for classification tasks. The goal of SVM is to find the best hyperplane that separates the data into two or more classes.

Hard margin SVM operates under the assumption that the data is linearly separable, i.e., that there is a hyperplane that can completely divide the data points of various classes. After that, the SVM algorithm seeks out the hyperplane that maximizes the separation (margin) between the hyperplane and the nearest data points for each class. Support vectors are used to describe these nearest places.

The weights and biases that form the hyperplane must be set so that the margin is maximized, and every data point is accurately classified according to its class labels. This is the SVM algorithm's optimization challenge. Convex quadratic programming techniques can be used to effectively tackle this problem, which can be expressed as such.

When the data can be linearly separated and there is no noise or outliers, hard margin SVM performs well. Soft margin SVM is a more commonly used method because it is uncommon in real-world circumstances to have completely separable data.

Soft Margin SVM:

Soft margin SVM allows some misclassifications by introducing a slack variable for each data point. The slack variable measures the degree to which a data point violates the margin constraint. The optimization objective of the soft margin SVM is to find the hyperplane that maximizes the margin, while minimizing the sum of the slack variables.

The soft margin SVM algorithm also includes a hyperparameter, known as the regularization parameter or C, which controls the tradeoff between maximizing the margin and minimizing the slack variables. A larger value of C means the model will try to minimize the slack variables more, potentially overfitting the data, while a smaller value of C means the model will tolerate more errors in classification, potentially underfitting the data.

Neural Networks:

Neural networks are a type of machine learning system that mimics the structure and function of the human brain. The model is made up of layers of interconnected nodes or neurons, with each neuron receiving input from other neurons and producing output that is passed on to neurons in the next layer. Neural network models with any number of layers and nodes can model intricate interactions between predictors and target variables.

One of neural networks' primary advantages is their capacity to learn and generalize from data. Unlike standard statistical models, which require prior assumptions about the underlying data distribution, neural networks may learn complicated patterns in data and make predictions on fresh, previously unknown data.

Neural networks are especially well-suited for modeling highly nonlinear datasets with complicated interactions between predictors and target variables. They have been used successfully in a variety of applications, such as image and audio recognition, natural language processing, and predictive modeling in finance and healthcare.

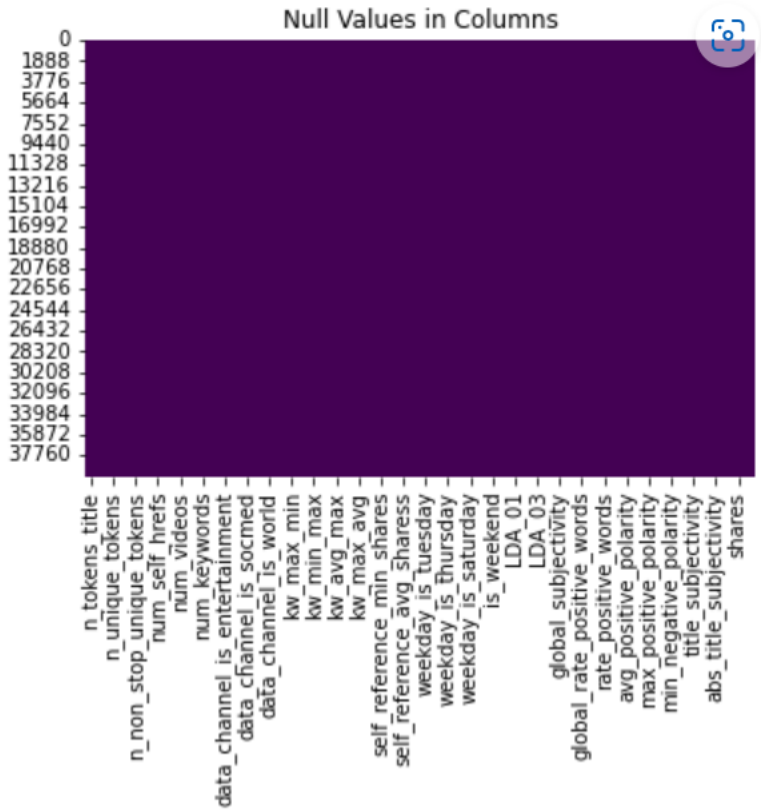
Exploratory Data Analysis (EDA) :

Exploratory Data Analysis (EDA) is the process of analyzing and displaying data to discover patterns and relationships. It aids in the detection of discrepancies or anomalies in the data and provides insights into the data.

Checking the dataset for missing or null values is one of the initial processes in EDA. Missing values can impair the accuracy of the analysis and predictions; thus they must be handled carefully. The dataset in this project contained no missing or null values, which is a good sign because it saves time and effort in data cleansing.

After checking for null values, other exploratory analyses like data distribution, correlation between variables, and outliers were performed to gain a deeper understanding of the data. This helps in selecting the appropriate models and features for the prediction task.

Firstly, we checked for the Null values in all the columns.



From the above visualization, we can clearly see that there are no null values in any of the columns. Hence, we are good to go ahead with the next steps.

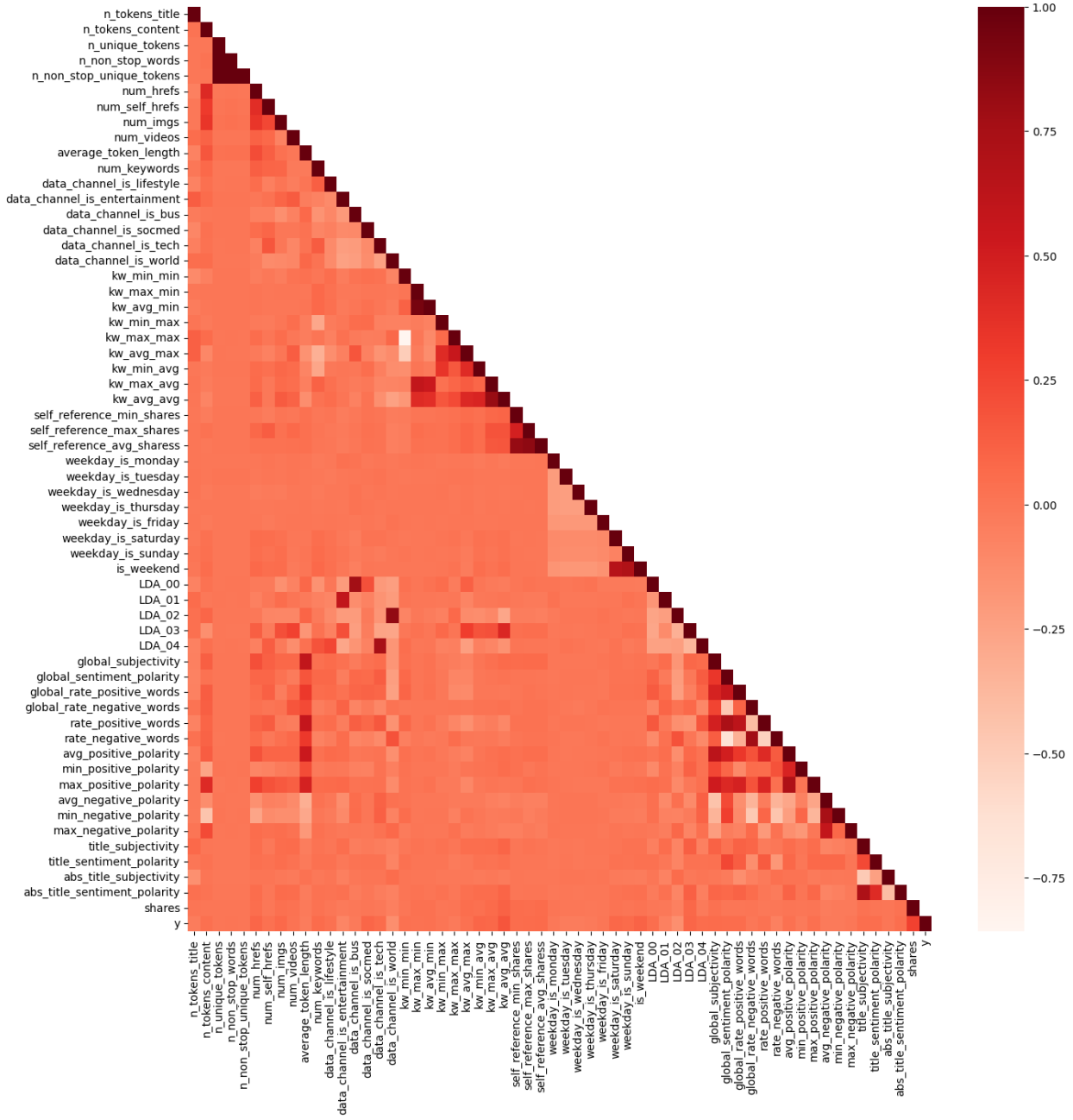


fig:1

A correlation map was generated as shown in Fig.1 to determine the strength of the relationship between the different features in the dataset. In order to identify highly correlated features, we set a threshold of correlation greater than or equal to 0.7. Whenever a pair of features had a correlation coefficient equal to or above 0.7, we dropped one of the variables amongst the highly correlated pairs to avoid multicollinearity issues, which can adversely affect the performance of the machine learning models.

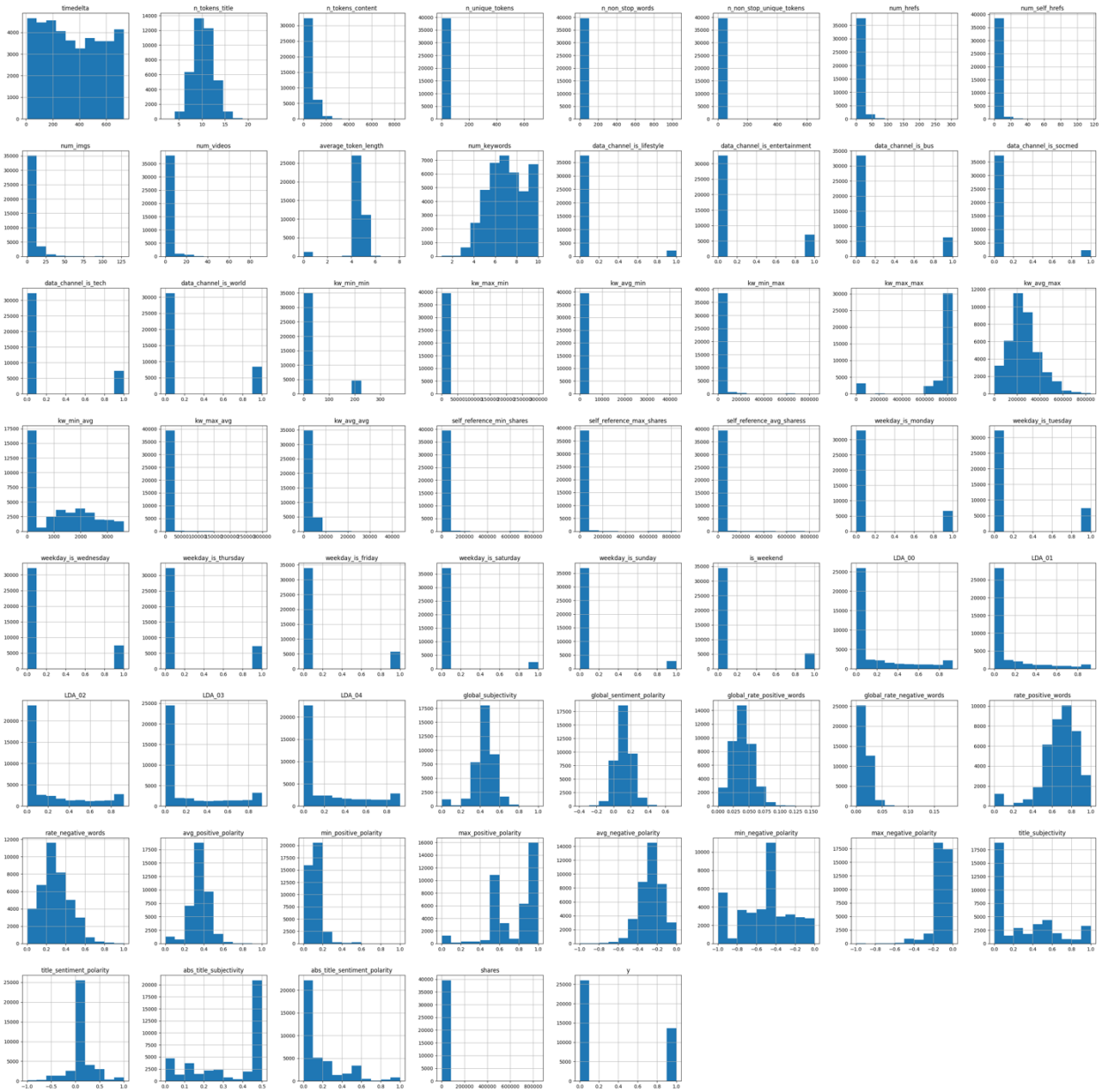


Fig:2

Figure 2 displays histograms of all the features in the dataset. Histograms are used to visualize the distribution of the data. From these histograms, we can observe that most of the features in the dataset are not normally distributed and do not have a single mode. Additionally, some features are binary, which means they only have two discrete values - 0 and 1. Understanding the distribution of the data is important in selecting appropriate machine learning algorithms and preprocessing techniques.

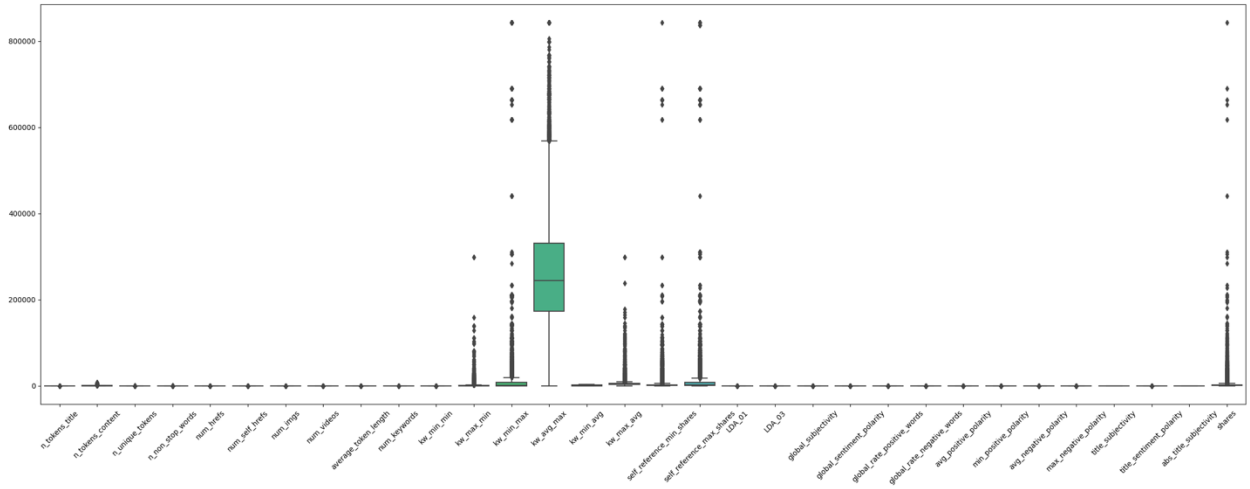


Fig:3

Figure 3 displays box plots of the continuous data columns after dropping highly correlated columns. We can observer that there are many outliers and the data is not normally distributed.

So, to overcome this problem and have a normally distributed data we decided to log transform the data. Log transformation is a typical technique for reducing the skewness of a distribution's distribution. When data is excessively skewed, a log transformation can assist make it more symmetric and closer to a normal distribution. This can aid in the improvement of the performance of some machine learning algorithms that are sensitive to data distribution.

Furthermore, log transformation can aid in decreasing the impact of outliers on data. The impact of huge values is lessened, and the dispersion of the data is compressed, by calculating the logarithm of the values.

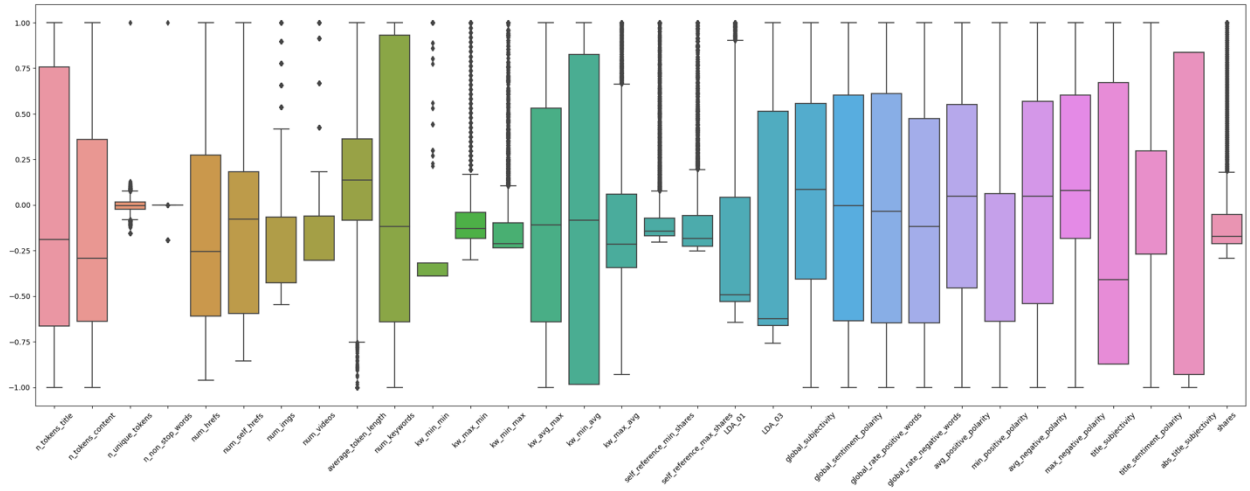


Fig:4

Here in the Figure 4 we plotted the box plot for log transformed data. We can clearly see the reduction of outliers and normally distributed data.

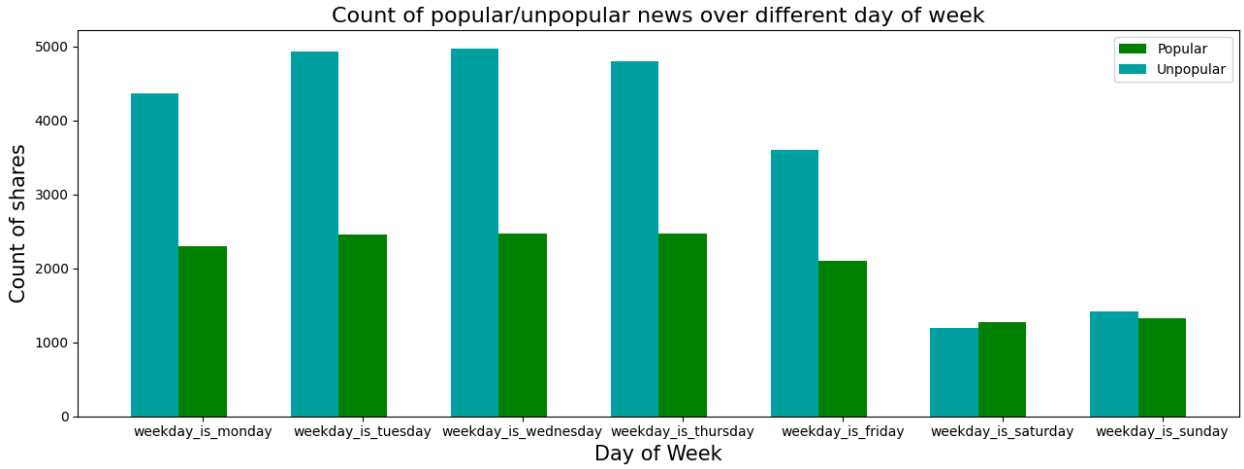


Fig:5

Fig.5 displays a bar chart that shows the count of articles published on each day of the week, as well as the number of popular articles released on each day. From the chart, it is evident that there are more articles released during the weekdays (Monday to Friday) as compared to weekends (Saturday and Sunday). However, during the weekends, there are relatively fewer articles released, but among those released, a higher proportion of them seem to be popular. This information could be useful in deciding the best days to release articles for maximum engagement and popularity.

Fig:6

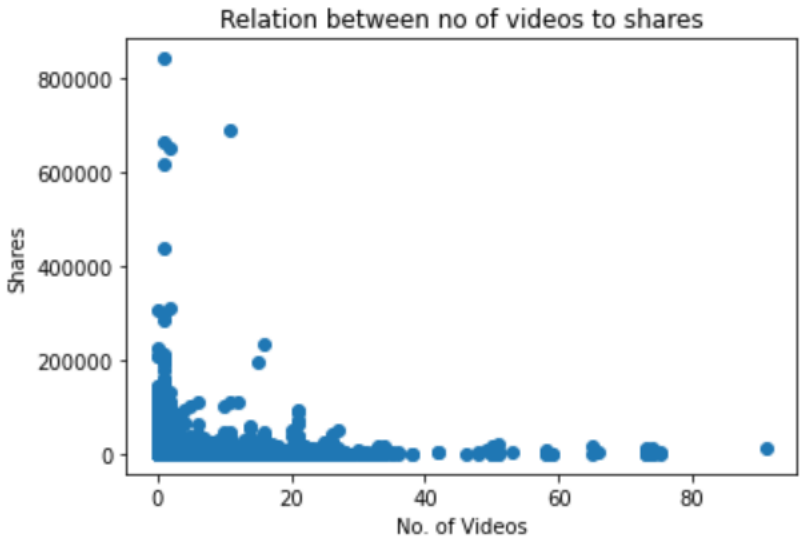


Fig:7

In Figures 6 & 7 we looked at the relationship between the quantity of videos and photographs in a news story and the number of shares it received in Fig.5. We discovered that as the quantity of images and videos in an article increased, so did the number of shares. To go deeper into this connection, we classified the amount of images and videos as either above or below the median value and compared the number of shares for popular and unpopular articles in each category. This analysis would help us identify whether the amount of images and videos has an effect on the number of shares based on the popularity of the article.

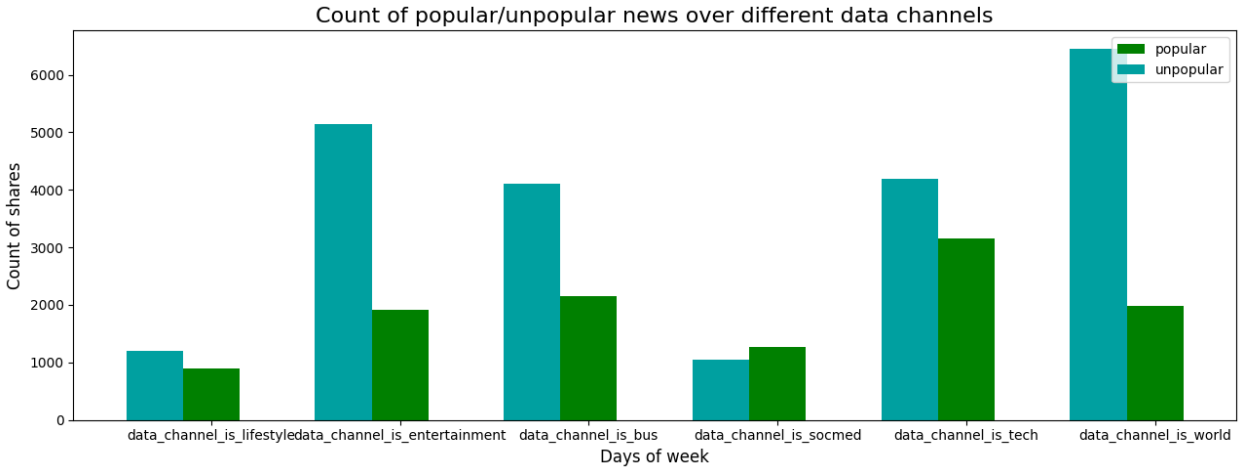


Fig:8

To further analyze the dataset, we examined the relationship between the type of channel an article belongs to and its popularity. We found that articles categorized under the "Technology" channel had the highest popularity, while those under the "Entertainment" and "World" channels had the lowest popularity. This could

indicate that readers are more interested in articles related to technology as compared to entertainment or world news.

Dimension Reduction:

One of the dimension reduction is PCA.
A statistical method called PCA (Principal Component Analysis) is used to shrink the size of huge datasets. It functions by locating the most significant recurring patterns and trends in the data and representing them as a more condensed collection of variables known as principal components. These primary components, which can be utilized for data exploration, data compression, and visualization, represent the most significant causes of variance in the original data. In data science, machine learning, and other domains where huge datasets need to be processed and visualized, PCA is frequently employed.
A statistical method known as PCA (Principal Component Analysis) is used to reduce the dimensionality of highly complex data while maintaining its variability.
The steps in PCA are as follows:
1, Standardize the data.
2, Computed the covariance matrix.
3, Calculate the eigenvectors and eigenvalues.
4, Pick the principal components.
5, Interpreting the outcomes.
These are the steps we followed while performing PCA these are the variances captured by various principal components.

```
Variance captured by 0 Principal Components is 0.0%
Variance captured by 1 Principal Components is 0.14010061163168722%
Variance captured by 2 Principal Components is 0.2101286817113992%
Variance captured by 3 Principal Components is 0.26417059998328923%
Variance captured by 4 Principal Components is 0.309603427020482%
Variance captured by 5 Principal Components is 0.34858270298886656%
Variance captured by 6 Principal Components is 0.3852440572229469%
Variance captured by 7 Principal Components is 0.42168899184253333%
Variance captured by 8 Principal Components is 0.4560799124629713 6%
Variance captured by 9 Principal Components is 0.48546141116494 13%
Variance captured by 10 Principal Components is 0.5133446042046215%
Variance captured by 11 Principal Components is 0.5406130223950381%
Variance captured by 12 Principal Components is 0.5666860898436262%
Variance captured by 13 Principal Components is 0.5923581332574309%
Variance captured by 14 Principal Components is 0.6173918807769195%
Variance captured by 15 Principal Components is 0.6420677835888364%
Variance captured by 16 Principal Components is 0.6661503863510808%
Variance captured by 17 Principal Components is 0.6899840365636607%
Variance captured by 18 Principal Components is 0.7133035894173374%
Variance captured by 19 Principal Components is 0.7359696230961116%
Variance captured by 20 Principal Components is 0.7579348943383946%
Variance captured by 21 Principal Components is 0.7787781658833021%
Variance captured by 22 Principal Components is 0.7992253983231069%
Variance captured by 23 Principal Components is 0.8189486846131778%
Variance captured by 24 Principal Components is 0.837701464864365%
Variance captured by 25 Principal Components is 0.8550729044060554%
Variance captured by 26 Principal Components is 0.8702816799656447%
Variance captured by 27 Principal Components is 0.8843649877149932%
Variance captured by 28 Principal Components is 0.8975927529296146%
Variance captured by 29 Principal Components is 0.9098530997437045%
Variance captured by 30 Principal Components is 0.9212597773691855%
Variance captured by 31 Principal Components is 0.9323901039061238%
Variance captured by 32 Principal Components is 0.9420895127347304%
Variance captured by 33 Principal Components is 0.9510953532479606%
Variance captured by 34 Principal Components is 0.9596607065330609%
Variance captured by 35 Principal Components is 0.9668969444970621%
Variance captured by 36 Principal Components is 0.9736645591014003%
Variance captured by 37 Principal Components is 0.9794712739289022%
Variance captured by 38 Principal Components is 0.9844277980425865%
Variance captured by 39 Principal Components is 0.9884750900181042%
Variance captured by 40 Principal Components is 0.9924623295947627%
Variance captured by 41 Principal Components is 0.9958608806794292%
Variance captured by 42 Principal Components is 0.998449568016 43%
Variance captured by 43 Principal Components is 0.9991538880305069%
Variance captured by 44 Principal Components is 0.9997789133418149%
Variance captured by 45 Principal Components is 1.0%
```

Post calculating the variances, we wanted to observe if there is any cost efficiency in terms of computation. Below is the table, which represents the CPU Percent used and Memory percent used when the number of components changes

No. of components	CPU percent used	Memory percent used
26	10.1	40.7
27	9.6	40.1
28	8.9	35.8
29	11.5	36.3
30	11.5	38
31	16.1	39.8
32	8.6	40.5
33	11.6	41.5
34	34.5	44.8
35	17.3	45.1
36	13.1	43.6
37	12.8	43.7
38	20	45.5
39	12.6	44.3
40	12	44.2
41	14.8	44.5
42	12.3	44.3
43	9.9	44.6
44	11.9	44.1

From the above table, we can conclude that there is no much cost saving in terms of computation. Hence we have decided to go ahead without Performing PCA.

Below are the two types of datasets which we used in our project

- Original data, after removing the highly correlated values
- Log transformed data

Balance in Train-Test Split:

For Classification

To evaluate the performance of a model, it is common in machine learning to divide a dataset into training and testing subsets. However, the dataset may not have a balanced distribution of the target variable in some circumstances. This indicates that the quantity of samples in each class of the target variable may be significantly varied, affecting the model's accuracy.

Stratified sampling can be utilized in such instances to establish a balanced split of the dataset. Stratified sampling ensures that in both the training and testing subsets, the proportion of samples in each class is the same. This is significant since it ensures that the model is not biased toward forecasting the majority class.

To create stratified sampling, we use the stratify argument in Python's Scikit-learn library's train_test_split method. The stratify option takes as input the target variable and ensures that the same proportion of samples from each class is included in both the training and testing groups.

The y variable in the preceding code sample represents the goal variable, which is the number of shares. We ensure that the training and testing subsets have a balanced distribution of shares by using stratify=y as a parameter in the train_test_split function.

For Regression

Since the above mentioned stratify mechanism doesn’t work for the target variable which is of type continuous, we have decided to split the data into percentiles and then take equal portions of data from each percentile range, so that we will have the train and test data from all the buckets, thereby providing the uniform data split for both train and test.

Implementation of models with evaluation metrics:

In this section, we implemented all the models using Python code, leveraging the NumPy and Pandas libraries without relying on the scikit-learn library. We evaluated the performance of these models using various metrics such as accuracy for classification models, and RMSE and MAE for regression models. We split the datasets into a 70:30 train-test ratio, and further split the train data into train and

validation sets in the same 70:30 ratio. This allowed us to train our models on the training data, tune the hyperparameters on the validation data, and finally evaluate the performance on the test data.

Linear Regression:

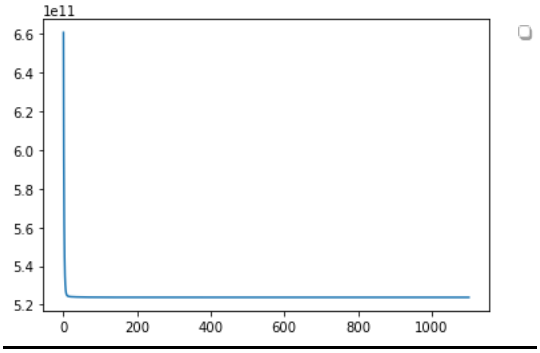
Linear regression is a model used to predict a continuous target variable based on one or more independent variables. In this implementation, gradient descent is used to optimize the model parameters. Gradient descent is an iterative algorithm that minimizes the loss function by updating the model parameters in the direction of steepest descent. The loss function used here is the sum of squared errors.
To start with, the model randomly initializes the weights and updates them using gradient descent until convergence. The final weights obtained are then used to predict the output for the testing data. The performance of the model is evaluated using two metrics, RMSE and MAE, for the testing data.
To avoid overfitting, ridge regularization is implemented by adding an L2 penalty term to the loss function. This helps in reducing the variance of the model and leads to better generalization.

The parameters used in this implementation are as follows:

Learning Rate	Lambda
0.01	0.01
0.01	0.001
0.001	0.01
0.001	0.001
0.0001	0.01
0.0001	0.001
0.00001	0.01
0.00001	0.001

Here are the results for Linear Regression

Dataset	Regularization Method	RMSE
Original data after removing highly correlated columns	Lasso	8191.888858
Original data after removing highly correlated columns	Ridge	8191.891272
Log Transformed	Lasso	4285.918132
Log Transformed	Ridge	4285.91934
Log Transformed With data Partition	Lasso	8883.256896
Log Transformed With data Partition	Ridge	8883.255539



The figure give information about learning rate vs No of iterations

Logistic Regression:

Logistic regression with gradient descent is a popular algorithm used for binary classification problems. The goal of logistic regression is to learn a decision boundary that separates the two classes, 0 and 1, in the dataset. In logistic regression, we use the sigmoid function to predict the probability of a sample belonging to a particular class. The sigmoid function is used because it maps any input value to a value between 0 and 1.

Gradient descent is a variant of gradient descent that updates the model weights based on a single training example at a time. This is in contrast to batch gradient descent, which updates the weights based on the average gradient of the loss function over the entire training set. By using stochastic gradient descent, we can

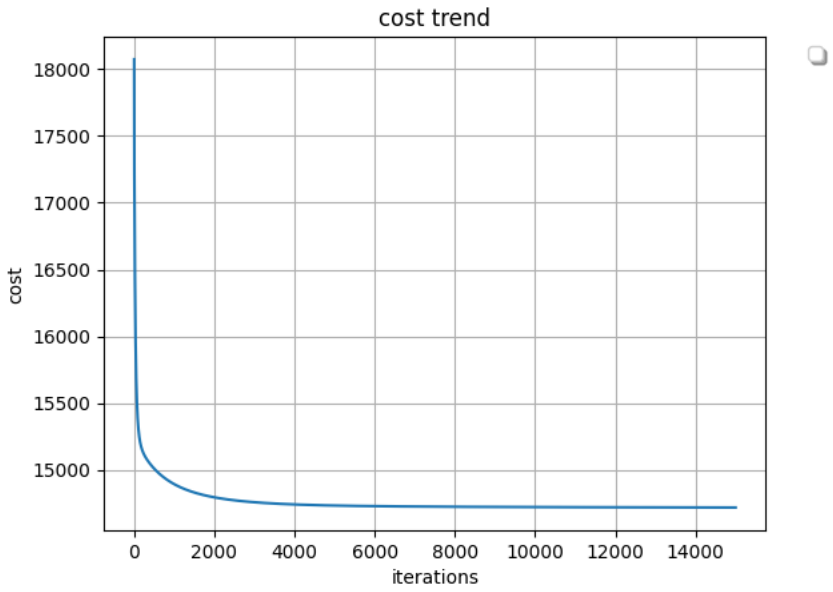
update the model weights more frequently, which can lead to faster convergence and better results.

In our implementation of logistic regression with gradient descent, we first split the dataset into training and testing sets. We then randomly initialized the model weights and used gradient descent to optimize the model weights based on the training set. We evaluated the model's performance using the accuracy metric on the validation set and made changes to the hyperparameters, such as the learning rate and regularization parameter, to improve the performance.

Once we found the best set of hyperparameters, we used the optimized model to make predictions on the testing set and evaluated its performance using the accuracy metric. The goal of logistic regression is to maximize the likelihood of the training data, so a higher accuracy on the testing set indicates a better-performing model.

Here are the results for Logistic Regression:

Metrics	Correlated - Stratify-Yes	Correlated - Stratify-No	Log transformed - Stratify - Yes	Log transformed - Stratify - Yes
Accuracy	76.5%	85.2%	94.2%	94.21%
Recall	0.416	0.743	0.897	0.895
Precision	0.808	0.812	0.932	0.933
F1-Score	0.549	0.776	0.914	0.9143



The figure show the relation between cost and number of iterations

Support Vector Machine (SVM):

Support Vector Machine (SVM) is a popular machine learning algorithm used for both classification and regression tasks. In classification tasks, SVM finds a decision boundary (hyperplane) that maximizes the margin (distance) between the two classes. Soft-margin SVM allows some misclassifications in order to accommodate data that cannot be separated linearly.

In order to use SVM, the target variable must be transformed to have values of either 1 or -1. This is because the objective function of SVM is formulated to maximize the margin between the classes and is based on the dot product between the feature vector and the weight vector. The optimization problem involves finding the alphas that minimize the objective function subject to the constraint that the dot product of the weight vector and the feature vector for each data point must be equal to the target variable multiplied by the alpha. This problem can be solved using Quadratic Programming, which can be computationally expensive for large datasets.

To reduce the computational cost, we can sample the data and consider only a subset of the data points while running the SVM model. This can significantly reduce the time required to train the model, but may also reduce the accuracy of the model due to the reduced sample size.

Without Kernel

Metrics	Correlated-Stratify-N	Correlated-Stratify-Y	Log-Transformed Stratify-N	Log-Transformed Stratify-Y
Accuracy	65.6%	64%	76.4%	80.4%
F-1	0.666	0.64	0.629	0.068
Precision	0.659	0.65	0.678	0.801
Recall	0.673	0.63	0.586	0.573

With Gaussian Radial Basis Function Kernel

Metrics	Correlated-Stratify-N	Correlated-Stratify-Y	Log-Transformed Stratify-N	Log-Transformed Stratify-Y
Accuracy	46.8%	49.2%	66%	64.7%
F-1	0.334	0.316	0.068	0.074
Precision	0.461	0.495	0.510	0.382
Recall	0.262	0.232	0.036	0.041

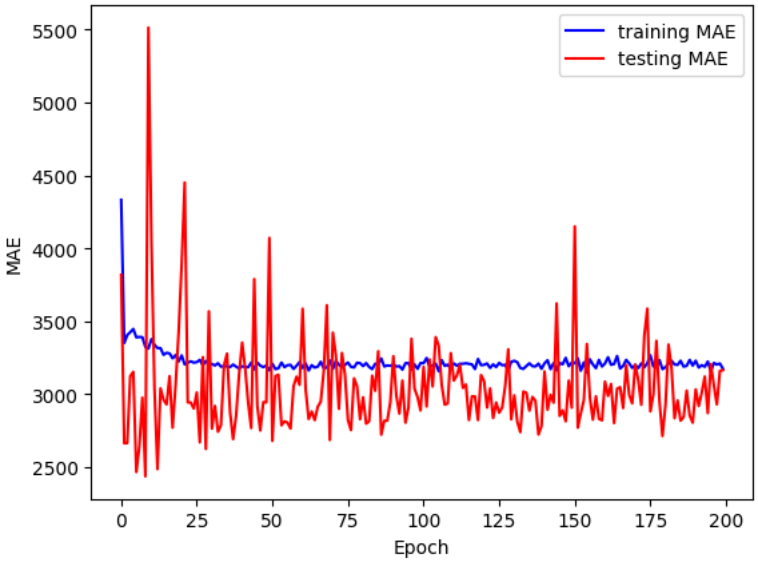
Neural Networks:

Neural networks are a type of machine learning algorithm that can learn complex relationships between input features and output targets. In this project, two neural network models were implemented using the Keras library.
Model 1 was a simple neural network with 2 dense layers. The first layer had 45 nodes with a Rectified Linear Unit (ReLU) activation function, and the second layer had 1 node with a sigmoid activation function. The ReLU activation function is commonly used in neural networks as it allows for faster training and can help prevent the vanishing gradient problem. The sigmoid activation function is often used in the output layer of binary classification problems as it outputs values between 0 and 1, which can be interpreted as the probability of belonging to a certain class.
Model 2 was a more complex neural network with 43 dense layers. The first 40 layers had 45, 30, and 20 nodes, respectively, with a ReLU activation function. The last 2 layers had 1 node each, with a sigmoid activation function. Additionally, there were 2 dropout layers with a rate of 0.5 each, which randomly dropped out 50% of the input values during training. Dropout layers are commonly used in neural networks to prevent overfitting, which is when the model performs well on the training data but poorly on new, unseen data.
The performance of both models was evaluated using accuracy as the performance metric. Accuracy measures the percentage of correct predictions made by the model on the test data.
Here are the results:

For Regression Task:

Dataset	Test MAE

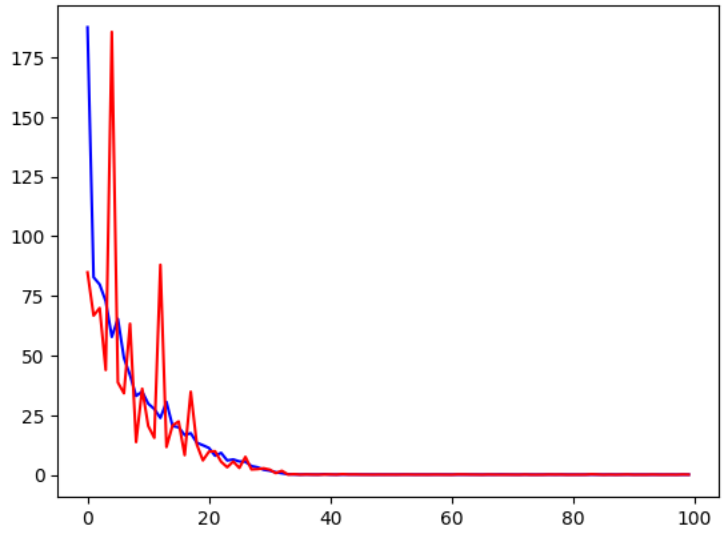
Correlated data with one hidden layer	3419.322
Correlated data with Three hidden layers	3041.29
Log Transformed data with one hidden layer	2573.885
Log Transformed data with Three hidden layer	2178.575



The figure shows the Train and Test MAE for each Epoch

For Classification Task

Dataset	Accuracy
Correlated data with one hidden layer	97.8%
Correlated data with Three hidden layers	98.8%
Log Transformed data with one hidden layer	96.8%
Log Transformed data with Three hidden layer	97.1%



The figure shows the Train and Test loss for each Epoch

Discussion and Conclusion:

Based on the criteria for model selection, the optimal models would be

For Regression Task:

Model	Dataset	Regularization	RMSE
Linear Regression	Original data after removing highly correlated columns	Lasso	8191.88
Linear Regression	Log Transformed	Ridge	4285.91

For Classification Task

Model	Dataset	Startify	Accuracy
Logistic Regression	Log Transformed	Yes	94.2%
SVM	Log Transformed (without Kernal)	Yes	80.4%
Neural Networks	Correlated data with Three hidden layers	-	98.8%

Profiling:

We also wanted to observe the attributes which have influence on the target variables. Based on the weights obtained from the linear regression model, below are the list of top 10 attributes, which have high influence on the target variable

	Columns	Weights
4	num_hrefs	273.052091
20	global_subjectivity	248.733328
15	kw_max_avg	231.100738
14	kw_min_avg	181.338577
6	num_imgs	176.206068
13	kw_avg_max	164.976411
16	self_reference_min_shares	155.783920
9	num_keywords	146.043880
43	weekday_is_sunday	123.961811
19	LDA_03	123.458706

Here is the brief description about the top 10 columns, which have high influence on the target variable 'Shares'

Number_hrefs: This column likely contains the number of hyperlinks It could represent the count of external links or internal links within the article.

global_subjectivity: This column likely represents a measure of the subjectivity of the online news article

kw_max_avg: This column likely contains the average maximum popularity of the keywords used in the online news article.

kw_min_avg: This column likely contains the average minimum popularity of the keywords used in the online news article.

num_imgs: This column likely represents the number of images included in the online news article.

kw_avg_max: This column likely contains the average maximum popularity of the keywords used in the online news article.

self_reference_min_shares: This column likely represents the minimum number of shares or engagement received by the self-referential links or mentions in the online news article.

num_keywords: This column likely contains the count of keywords used in the online news article.

weekday_is_sunday: This column likely represents if the publication day of the online news article falls on a Sunday.

LDA_03: This column likely represents the probability or relevance of the article to the third identified topic in the dataset, with higher values indicating a higher association with that topic.

github link: https://github.com/jvsasank/Online-News-Popularity