Project BlogPop: Predicting the Popularity of Blog Posts

CS-5644

Prateek Kumar Padhy
pprateekkumar@vt.edu

Aadyanth Khatri
aadyant@vt.edu

Introduction:

The primary objective of this project is to build a predictive model capable of estimating the number of comments a blog post will accumulate within the next 24 hours, specifically predicting the popularity of a blog post. Estimating the future popularity of a blog post can be crucial for content creators, bloggers, and website administrators for several reasons:

1. **Content Strategy**: Predictive models can help content creators plan their content strategy. If they know which types of posts are likely to generate more comments, they can focus their efforts on those topics.
2. **Engagement Measurement**: It can be used to measure the effectiveness of posts. If a post underperforms in terms of comments, it might signal that it didn't resonate with the audience.
3. **Resource Allocation**: It can guide resource allocation. If you know that a particular post is likely to become very popular, you might want to allocate more resources, such as marketing or moderation, to that post.

This estimation will be based on data available up to a specific basetime, 72 hours specifically, ensuring that the model can be used in practical situations where historical data is used to predict future outcomes.

Dataset Overview:

Our data has been collected from the UC Irvine's Machines Learning repository.

The dataset contains information on 52,397 individual blog posts. Each post is associated with 281 different features, one of which is the target variable representing the number of comments the post received within 24 hours after a specific "basetime." The features include data such as the number of comments before and after the basetime, blog post length, and binary indicators for the day of the week when the basetime occurred (Monday through Sunday). Additionally, there are 200 features representing the presence or absence of 200 frequent words in the blog post's text.

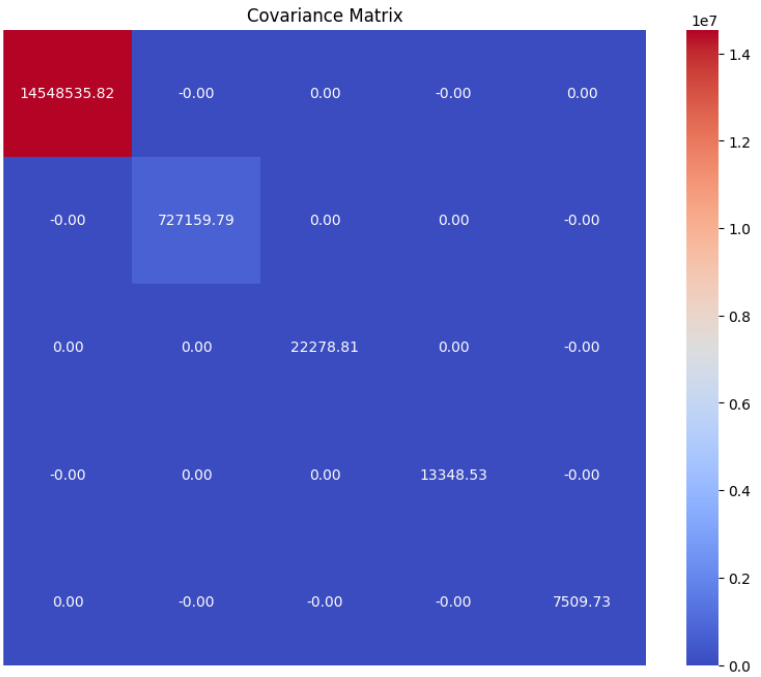
The training data comprises basetimes from the years 2010 and 2011, while the test data contains base times from February and March of 2012. This temporal separation between

training and test data ensures a realistic evaluation of the model's performance, reflecting a scenario where historical data is used to predict future events.

Exploratory Data Analysis (EDA):

1. After exploring the dataset, we realized that features 1 to 50 comprise average, standard deviation, min, max and median respectively, for the attributes 51 to 60. The various attributes and their descriptions are as follows:
    a. Attribute 51 indicates total comments before basetime,
    b. Attribute 52 indicates comments made in the last 24 hours,
    c. Attribute 53 indicates comments made within 24 to 48 hours,
    d. Attribute 54 indicates the comments which were uploaded in the first day,
    e. Attribute 55 is the difference between the features 52 and 53,
    f. Attributes 56 to 60 indicates the number of trackbacks,
    g. Attribute 61 has the length of time between the publication of the blog post and basetime,
    h. Attribute 62 is the length of the blog post,
    i. Attributes 63 to 262, comprises of the bag of words,
    j. Attributes 263 to 269 is onehot encoded for the day of the week (binary classification) of basetime,
    k. Attributes 270 to 276 (binary classification) for the day of publication,
    l. Attribute 277 has the number of parents the blog has or the number of links,
    m. Attributes 278 to 280 has the min, max and average number of comments the parents received,
    n. Attribute 281 is the target which indicates the number of comments in the next 24 hours.
2. We have standardized the data using Scikit-Learn's StandardScaler package. This will help in better predictions of the given data as it is a binary classification.
3. We used principal component analysis to find out the most important features which would make the model fit better for regression, though we realized that we need all the features to make a accurate prediction as seen in the values below
    a. PCA with n_components = 5:

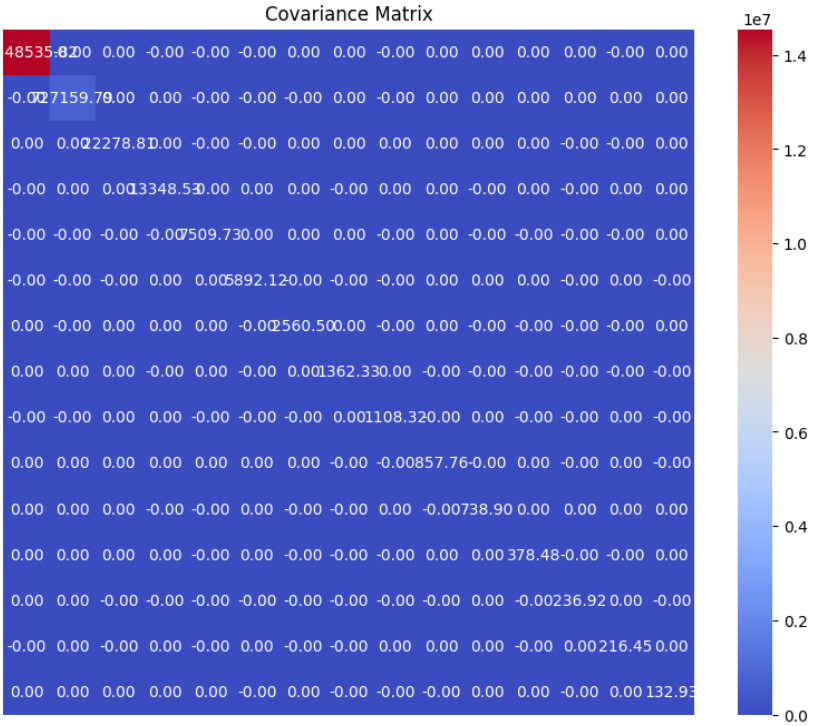
Component	Explained Variance
1	0.948860
2	0.047426
3	0.001453
4	0.000871
5	0.000490



b. PCA with n_components = 15:

Component	Explained Variance
1	0.948860
2	0.047426
3	0.001453
4	0.000871
5	0.000490
6	0.000384
7	0.000167
8	0.000089
9	0.000072
10	0.000056
11	0.000048
12	0.000025
13	0.000015

14	0.000014
15	0.000009



4. Data augmentation: Before dividing a given dataset into training and testing sets, data augmentations were performed on the data. This process applies random changes to each original data point using the augment_data function to create augmented samples. The apply_augmentation function specifies the particular augmentation, such as adding random noise. Subsequently, the train_test_split function from scikit-learn is used to divide the augmented dataset into training and testing sets. Initially, we had approximately 50,000 instances, but through data augmentation, our dataset became more robust, increasing to over 100,000 instances. By expanding the dataset, we improve model generalization by providing a wider variety of cases for model training.
5. We have split the data into 80-20 where 80% is used for training and 20% is for testing. The random state has been set to 42 in order to reproduce the results.

Evaluation Metrics:

We have used the following two metrics to evaluate the performances of our models

1. Mean Squared Error:
    a. It is a commonly used metric to measure the average squared difference between the actual (true) values and the predicted values by a model. It is particularly prevalent in evaluating the performance of regression models.
    b. The formula for Mean Squared Error is:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

c. MSE quantifies the average of the squared differences between predicted values and actual values. Lower MSE values indicate that the model's predictions are closer to the actual values, reflecting better model performance in terms of minimizing prediction errors.

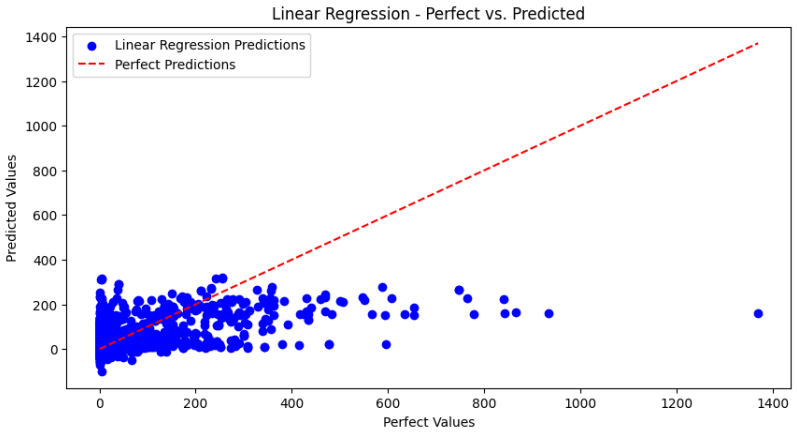
2. R^2 Error:
   a. R^2, also known as the coefficient of determination, measures how well the independent variables explain the variance in the dependent variable. It's a value between 0 and 1, where 0 means the model doesn't explain variance, and 1 means it perfectly explains it. It's calculated as:

$$R^2 = 1 - \frac{SSres}{SStot}$$

   b. Higher R-squared values indicate a better fit, showing that more variability in the dependent variable is explained by the model's independent variables.

Algorithms/ML methods:

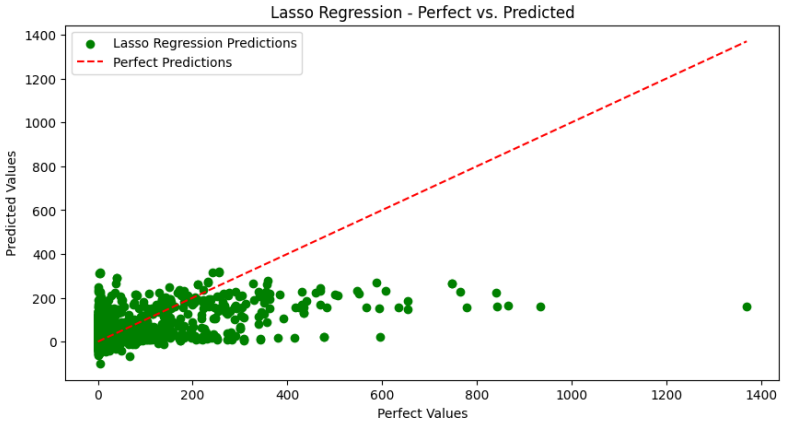
1. **Linear regression**
   a. On training a linear regression model on our data and testing it, we got the following results:
      i. MSE: 869.442
      ii. R-squared: 0.348



   b. We can see that the linear regression performs poorly on our data.

2. **Lasso Regression**

a. On training a lasso regressor on our data with alpha = 0.01, we got the following results:
   i. MSE = 868.643
   ii. R-squared: 0.348

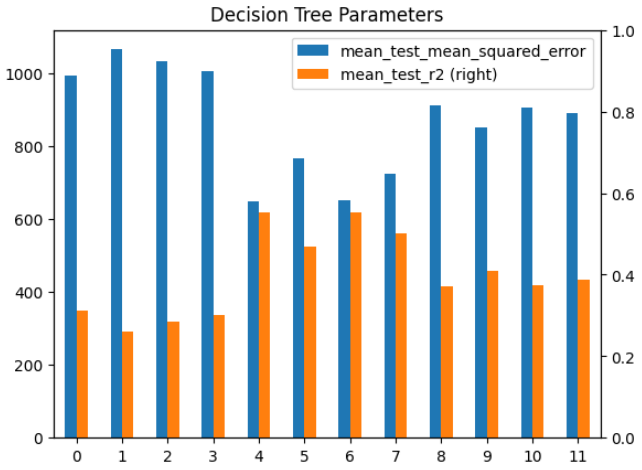


b. Though very slightly better than the linear regression model, the lasso regressor also performs poorly.

3. Decision Tree

a. We trained a decision tree and grid-searched the following parameter combinations:
   i. splitter: ['best', 'random'],
   ii. max_depth: [None, 5, 10],
   iii. max_features: [None, 'auto']

b. The performances of the various parameter combinations can be seen below:

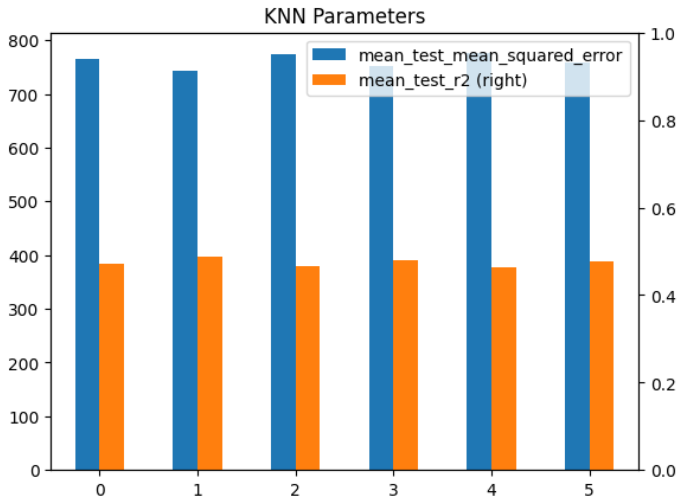


c. Here parameter set - 4 achieved the lowest mean squared error of 648.828 and an R-squared score of 0.553.

d. This parameter set had max_depth = 5, max_features = None and splitter =

4. K-Nearest Neighbors Regression

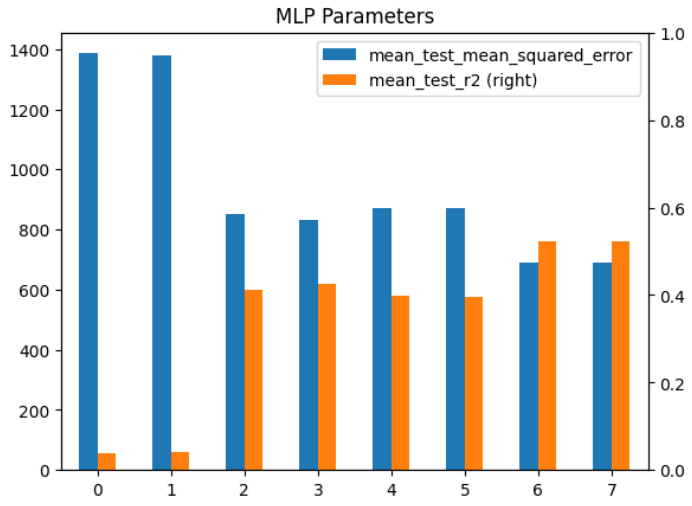
a. We trained a KNN regressor and grid-searched the following parameter combinations:
   i. n_neighbors: [3, 5, 8],
   ii. weights: ['uniform', 'distance']
b. The performances of the various parameter combinations can be seen below:



c. Here parameter set - 4 achieved the lowest mean squared error of 742.998 and an R-squared score of 0.488.
d. This parameter set had n_neighbors= 3, weights= distance

5. Multi-layer Perceptron Regressor

a. We trained an MLP Regressor and grid-searched the following parameter combinations:
   i. hidden_layer_sizes: [(100,), (100, 50,)],
   ii. activation: ['tanh', 'relu'],
   iii. learning_rate: ['constant', 'adaptive']
b. The performances of the various parameter combinations can be seen below:



c. Here parameter set - 4 achieved the lowest mean squared error of 689.289 and an R-squared score of 0.524.
d. This parameter set had activation = relu, hidden_layer_sizes = (100, 50) and learning_rate = adaptive

Conclusion:

From the various models we trained and tested, we can conclude that Decision Tree was the best performing one. This model serves as a valuable tool for gauging the potential popularity of the blog posts. By leveraging various features and employing robust machine learning techniques, this model offers insights into the expected engagement levels, enabling content creators and platform managers to anticipate and strategize effectively for maximizing the reach and impact of their blog posts. With further refinements and continuous evaluation, this predictive framework holds the promise of enhancing decision-making processes and fostering a more informed approach towards managing and promoting blog content to engage with the target audience.