Optimizing Attribute Selection for 3D Model Popularity Prediction

Kamyar Fatemifar, Suchismitaa Chakraverty, Joseph Hardin, Abhishek Misra, Max Pan

Optimizing Attribute Selection for 3D Model Popularity Prediction

Project Report

Introduction/Background and Literature Review

With the expansion of 3D digital content creation landscape, many established model repositories and markets are evolving, so the need for data-driven approaches to optimize decision making has become increasingly critical. The choice of proper tags like product name, product price, etc to maximize the model popularity therefore becomes a crucial need.

Studies have shown the effectiveness of content-based methods in organizing and retrieving 3D models, particularly when leveraging metadata and semantic features to enhance classification [1], [2]. In this project, we will utilize popular ML models and techniques to automate the process of decision making on attributes associations. By analyzing metadata (the existing tags associated with products), the proposed system will be able to predict the optimal attributes for new models. This in turn, will provide product publishers insights into the factors driving model popularity, facilitating better decision-making and enhancing content marketing strategies [3], [4].

Dataset Description

The output dataset provides a well-organized structure capturing essential information about 3D models, such as their names, URLs, pricing, and user engagement metrics like comments and likes. This structured format allows for efficient retrieval and analysis of model recommendations, tailored for further use in ml model training workflows. This data was ethically obtained in compliance with 3dsky’s robots.txt.

Link to Dataset (Tags)
Link to Dataset (Images)
Link to Data Source

Problem Definition and Motivation

With the increasing implementations of 3D model repositories and virtual asset marketing, there is a growing need for efficient market prediction methods. There are many businesses forming around virtual product marketing such as 3D models. Most of these businesses rely on creating a platform for connecting model publishers and customers. In this scenario, publishers and platform owners value higher sales and one of the most important factors affecting sales is product attributes/ tags which help customers identify the best products based on their needs. For every product published, publishers need to determine an ideal set of tags such that their products' popularity get maximized, which consequently increases the sales of their products and overall market performance.

In this project, we are trying to determine the ideal set of tags/attributes that helps maximize the popularity of a product, a publisher is about to publish in a particular sub-category under a category. Random forest has been used to determine the weights of the factors determining a product's popularity. Using unsupervised learning methods like K-means clustering and DBScan, we are determining the ideal set of tags that contribute to higher popularity of a product in a specific sub-category.

For many publishing organizations, the servers are loaded with lots of different product images which need to be classified automatically in sub-categories. Otherwise, the manual task of doing it can be quite tedious. This is another aspect of the problem that our ML model CNN aims to solve by automating the classification task of those images in accurate sub-categories quickly and efficiently.

Methods

Data Preprocessing Methods

In this project, we have implemented 3 models namely K-means and DBScan(unsupervised learning methods) and CNN (supervised learning method). Because of the huge size of the dataset and data containing lot of unwanted values (not suitable for model training), a variety of techniques have been implemented to clean the dataset. The pre-processing methods used are described as follows:

Data Cleaning

While working on the datasets, we found various models with empty tags, models without categories and models with missing data. As a result, we employed this technique to remove those models to make our dataset cleaner and more effective for training purposes. String slicing has been used to extract values required for the later implementation of popularity score generation, random forest and K-means.

Standardization

Standardization: One of our features for the K-means model is popularity score, which we standardize such that the value of the popularity score lies between 0 to 10. To accomplish this, we developed a Python script which gets the maximum score and minimum score in every sub-category and calculates the new weighted popularity score using the following equation:

$$x' = \frac{(x - \min)}{(\max - \min)} \times 10$$

In this equation, x is the original popularity score based on the popularity score function and x’ is the new weighted score.

Feature Engineering

Popularity score is a new feature that has been developed to feed into the K-means model such that the ideal set of tags that would maximize the popularity score of a product can be identified before the product’s publication. This score was developed by integrating multiple user engagement metrics such as likes, comments, sentiment analysis, and tags to reflect overall user interest and engagement with the 3D models. To identify the weights for tags in the popularity score calculation, random forest has been employed. In this way, the popularity score provides us with a new feature for each subcategory.

Random Forest (preprocessing for k-Means)

A random forest regressor was utilized to assess importance of different metrics in determining the popularity of 3D models based on various engagement metrics. A custom computed popularity score was created based on metadata from the 3D models. To see the impact per feature, a parameter grid was used to generate different combinations of weights for each metric. A decay factor was implemented to account for time since the model’s publication. The popularity score along with the corresponding weight and decay factor were used as inputs features and target values for the random forest model.

Machine Learning Models

To address the 2 aspects of the problem definition i.e. identifying the optimal set of tags that increases a product's popularity and automating the classification of thousands of products into accurate sub-categories, three models have been implemented respectively.

Unsupervised learning methods, namely K-means and DBScan have been employed to address the first part of the problem whereas supervised learning method CNN has been employed to address the second part of the problem. These methods are described in detail below:

K-Means Approach with Popularity Score calculation

In this approach, we tried to determine the ideal set of tags/attributes that helps maximize the popularity score and recommendation score of a product, a publisher is about to publish in a particular sub-category under a category. Popularity score is a feature that has been developed based on likes count, recommendation count (provided by users for each product), comments count and the sentiment of each comment (positive, negative or doubtful), the number of tags used for a product and the date of publication of that product. Random forest has been used to determine the weights of the factors determining the popularity score. After generating a popularity score, a recommendation score is calculated based on how many times a specific product URL has been recommended by other models on the website (this recommendation score is different from the above recommendation count mentioned). Using these 2 scores and K-means clustering, we determined the ideal set of tags that contribute to higher recommendation and popularity score for each product in a specific sub-category.

K-Means Approach with Word2vec Encodings

In this attempt, items that belonged to the category "Furniture" and the subcategory "chairs" were studied.

Word2vec is an encoding method that converts words to vectors using neural networks. The model combines skip-gram and continuous bag-of-words to predict a word based on its neighbors. The benefit of this method is that similar words will have similar vector representations. Products were inputted into the word2vec algorithm as "sentences" in the following way: [category, subcategory, tag 1, tag 2, ..., product name]. The resulting output vector size was set to 100 and the window size for prediction was set to 10. Afterwards, to reduce the dimensionality and therefore noise, PCA was used such that 95% of the variance was maintained.

For the algorithm, likes, recommendations, sentiment score, and the average of the item's vectorized tags were used as features. The values were normalized such that all quantities had a mean of 0 and a standard deviation of 1. The vectorized tags were normalized such that the total standard deviation from the mean vector was 1. This placed an equal importance on all the features.

Each product’s sentiment and how questioning it was given a score. For the positive sentiment value, it was determined by counting the number of “positive” comments. If there were no detected “positive” words, textblob’s sentiment polarity score was used. For the negative sentiment value, it was determined by counting the number of “negative” comments. If there were no detected “negative” words, textblob’s sentiment polarity score was used. For the questioning value, it was determined by counting the number of “questioning” comments. If there were no detected “questioning” words, textblob’s sentiment subjectivity score was used. The score was determined as:

$$\frac{2P + 2N - 0.5Q}{total\ \#\ of\ comments}$$

where P is the number of positive comments, N is the number of negative comments, Q is the number of questioning comments.

DBScan Attempt with Word2Vec Encodings

For DBScan, items within the category “Furniture” and category “chair” were studied. The same approach was used as the K-Means to vectorize the tags and normalize the features.

The minimum distance was selected to reduce the number of points that were considered as noise yet increase the silhouette score. The number of clusters was selected in the same way. At the end, 14% of the items were considered noise.

CNN Classification

A convolutional neural network (CNN) was used such that an input image was given and a subcategory classification was the output. Much of the code was based on CS 6476 Assignment 4. Modifications were used to accommodate a pretrained ResNet34 model instead of a ResNet18 model and the architecture was changed to specify the training set to validation set ratio as well as accommodate color images instead of black-and-white images.

The images for training and validation were sampled from a collection of ~130,000 images. Each subcategory had at most 500 images. If the subcategory did not have that many, all images from that subcategory was used. The training and validation set was split roughly into 70% and 30%.

We used a pretrained resnet34 model, where we froze all the layers except for last fully connected layer. The last fully connected layer was modified such that the output dimensions was the number of subcategories. For the training, various transforms such as random horizontal flip, gaussian blur, random rotation, and random scaling were used for data augmentation.

Results

The results of this study focused on evaluating the performance of a machine learning model designed to predict the popularity of 3D models based on various product attributes. Two primary machine learning approaches, Random Forest and K-means clustering, were employed to assess attribute significance and optimal tag selection.

Popularity Score and Recommendation Score Calculation

Popularity Score: This score was developed by integrating multiple user engagement metrics such as likes, comments, sentiment analysis, and tags to reflect overall user interest and engagement with the 3D models.

Website-Based Recommendation Score: Separate from the popularity score, a recommendation score was calculated based on how frequently a model's URL was recommended alongside other models on the platform. This score captures the model's interconnectedness within the recommendation system of the website, providing insights into broader visibility and potential reach.

Random Forest Regression

The Random Forest model was trained on a dataset that included multiple popularity-driving features, such as likes, recommendations, comments, sentiment analysis, and product tags. At this stage the goal was to determine the proper parameter weight settings for the popularity score function based on correlation between different parameters and the output.

Model performance was evaluated based on error metrics, including:

Mean Absolute Error (MAE): 2.09e-4
Mean Squared Error (MSE): 1.26e-7
Root Mean Squared Error (RMSE): 3.56e-4
R-Squared (R^2): 0.015

Although the R^2 value indicates a lower explained variance, the model achieved a low MAE and RMSE, reflecting accuracy in absolute error terms. This suggests that while the model captures the general trend of popularity scores, there may be room for improvement in variance prediction.

Feature Importance and Sensitivity Analysis

Using feature importance analysis, we found that:

Recommendations (0.647) and Tags (0.277) were the most influential features in predicting popularity scores.
Sentiment (0.060) and Comments (0.008) had lower impact.

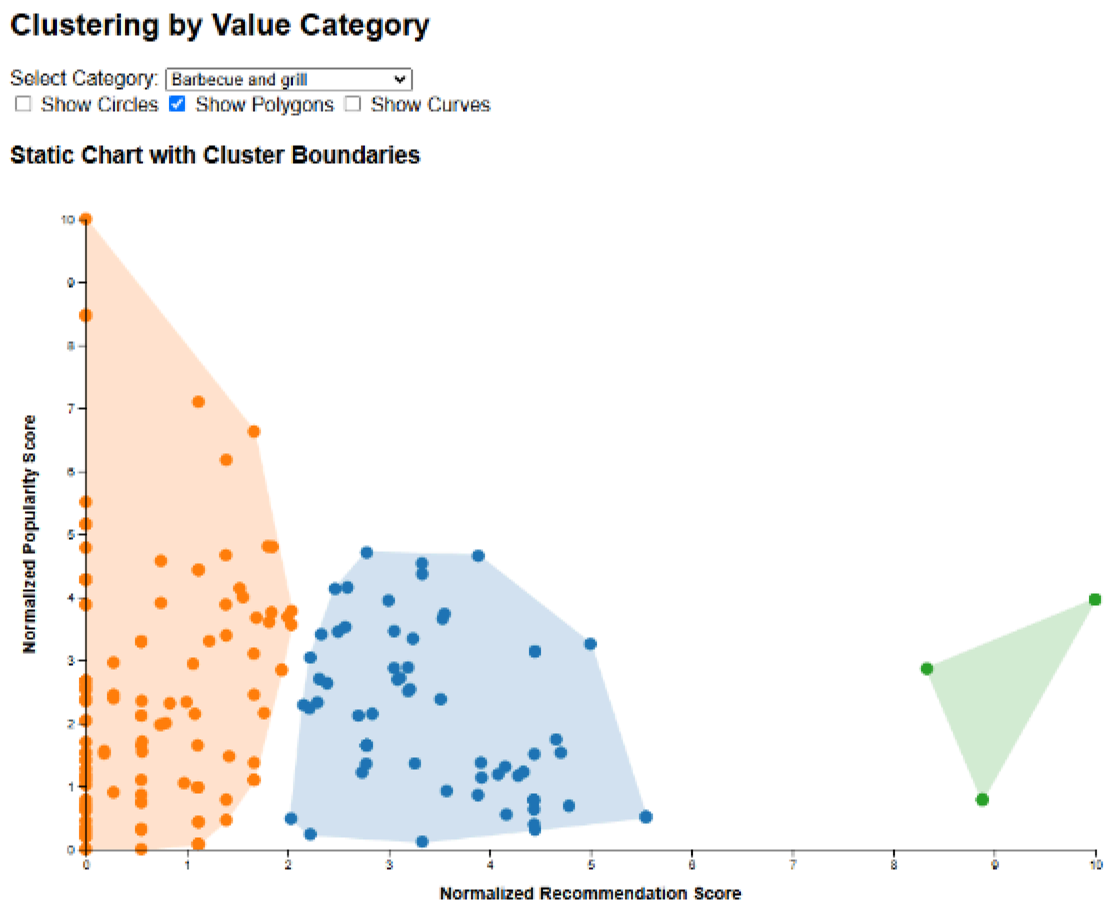
These findings indicate that user recommendations and the presence of relevant tags substantially affect popularity, highlighting the importance of these features in 3D model marketing.

K-Means Clustering and Tag Optimization

To identify ideal tag configurations, K-means clustering was applied using both the popularity and recommendation scores. By combining these two scores, we aimed to pinpoint tag sets that not only boost the popularity score (user engagement) but also align with website-driven recommendations (visibility and reach).

Clustering Methodology: The dataset was divided into clusters based on popularity and recommendation scores, grouping products into categories where specific tag combinations led to higher combined scores. This clustering enabled us to identify patterns in tag usage across different popularity and recommendation levels, uncovering the tags most frequently associated with successful models.

Tag Optimization Results: The clustering revealed that models with tags targeting both user preferences (captured by popularity score) and system-driven visibility (captured by recommendation score) were more likely to succeed. This insight can guide publishers in selecting tags that optimize both engagement and visibility, enhancing the model’s marketability.

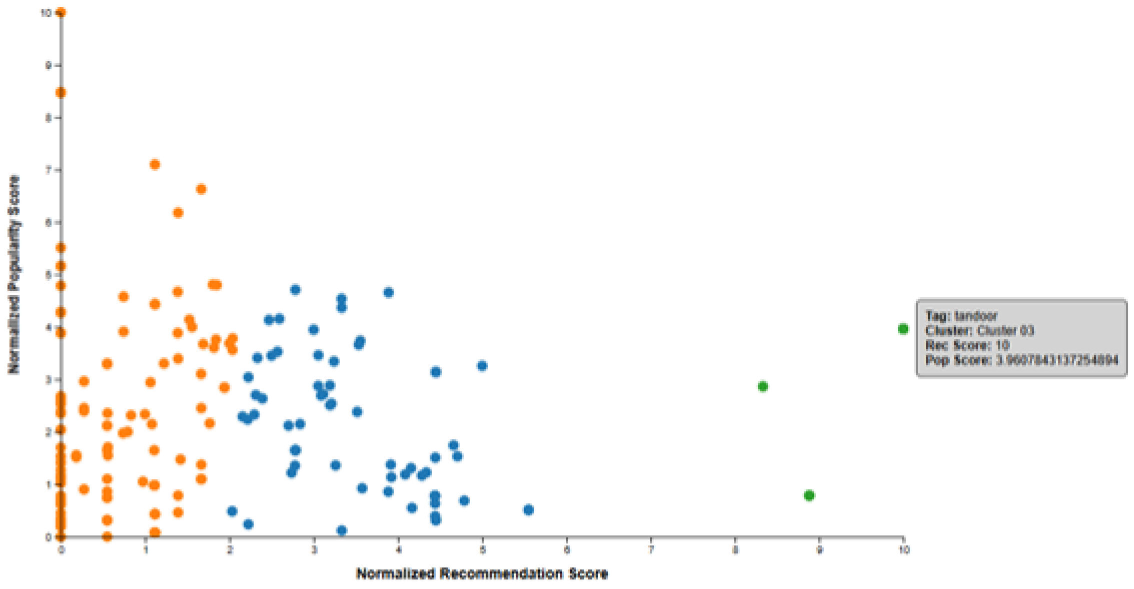

Clustering by Value Category

Select Category: Barbecue and grill

Show Circles [✓ Show Polygons] Show Curves

Static Chart with Cluster Boundaries

Visualizing K-means clustering with D3



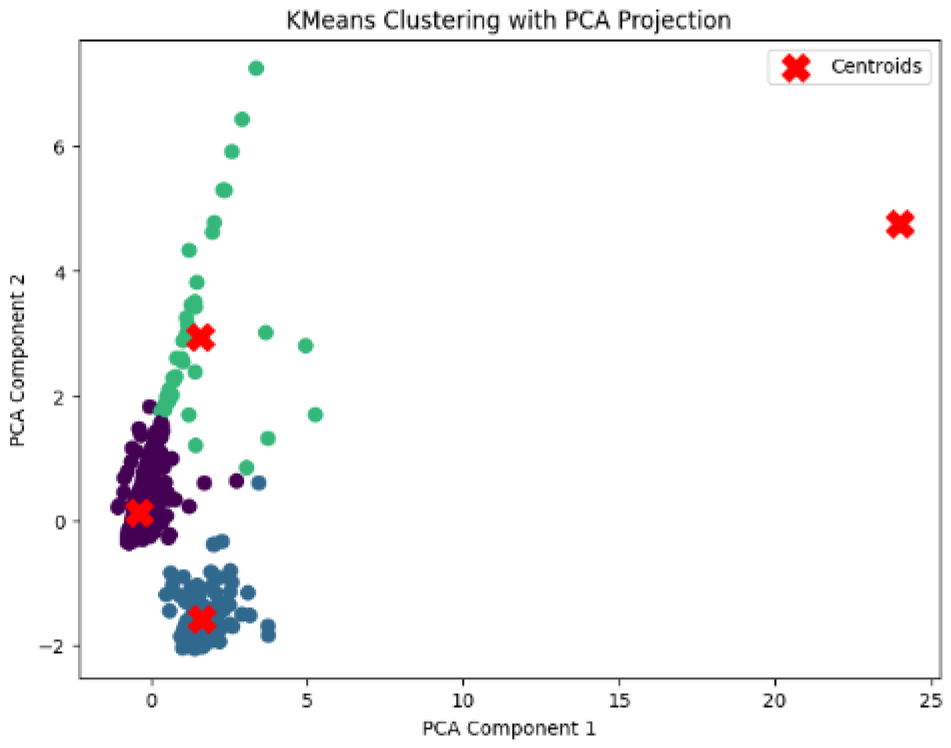
Interactive tag review chart

Validation and Consistency Analysis

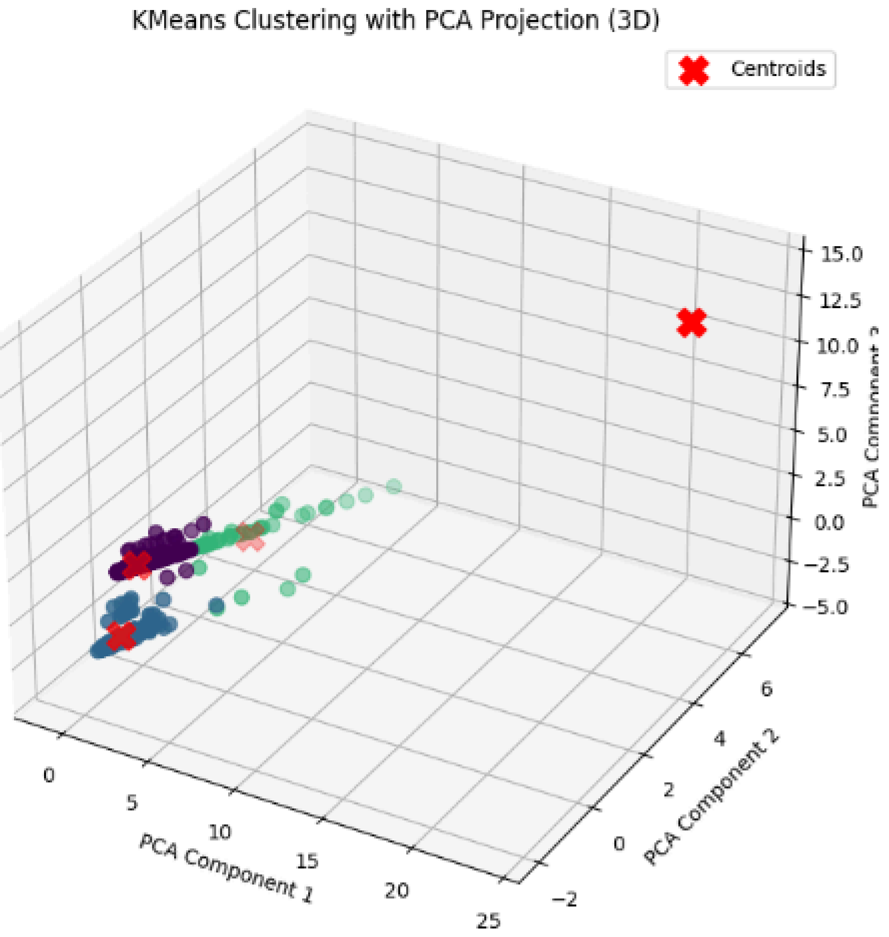
To validate the effectiveness of the popularity scores generated, we compared the Weighted Popularity Score with the Original Popularity Score using MAE and MSE. High correlation values between likes and both weighted and original scores confirmed that the computed scores aligned closely with user interest, thereby reinforcing the model’s predictive relevance.

K-Means with Word2Vec Results

For the K-Means, there were four clusters, resulting in a silhouette score of 0.507. The first center had ~11 likes, ~ 2 recommendations, sentiment score of 0, and the most similar tags were “stool”, “bar”, “mebelform”, “horeca”, and “chairs”. The second center had ~10 likes, ~10 recommendations, sentiment score of 1.9, and the most similar tags were “pink”, “minimal”, “white”, “yellow”, “homary”, and “red”. The third center had ~92 likes, ~ 6 recommendations, sentiment score of 0.2, and the most similar tags were “loftdesigne”, “restoraica”, “loft”, “furnitureform”, and “bellotti”. The fourth center had ~181 likes, ~353 recommendations, sentiment score of 2, and the most similar tags were “street”, “patio”, “square”, “outdoor”, “hamilton”.

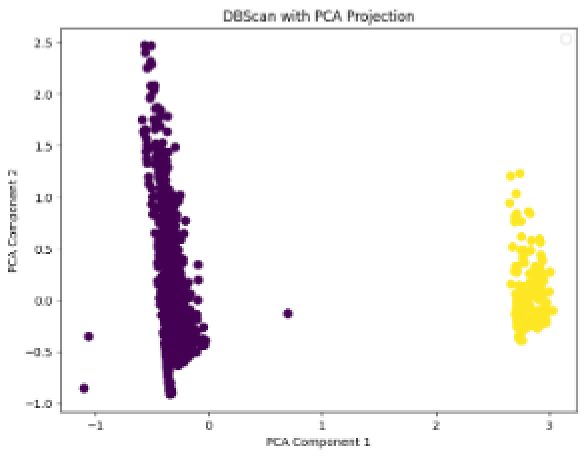


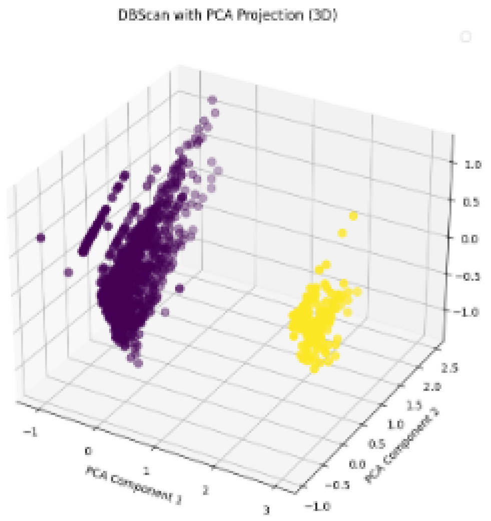
\



DBScan with Word2Vec Results

When plotting the non-noise points, the results were two clusters with 14% of the points considered noise. The resulting silhouette score was 0.583. This clustering seemed to be slightly improved from K-Means algorithm.



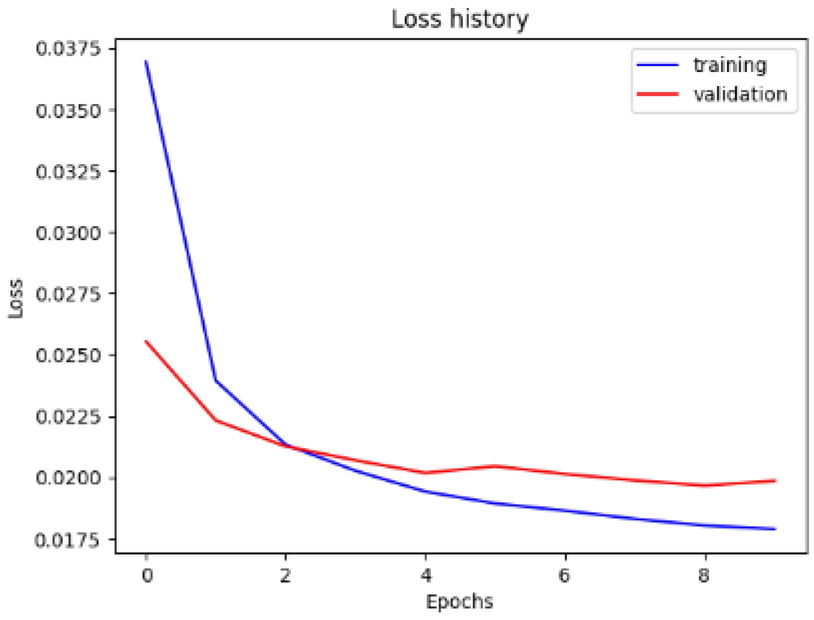


CNN Classification Results

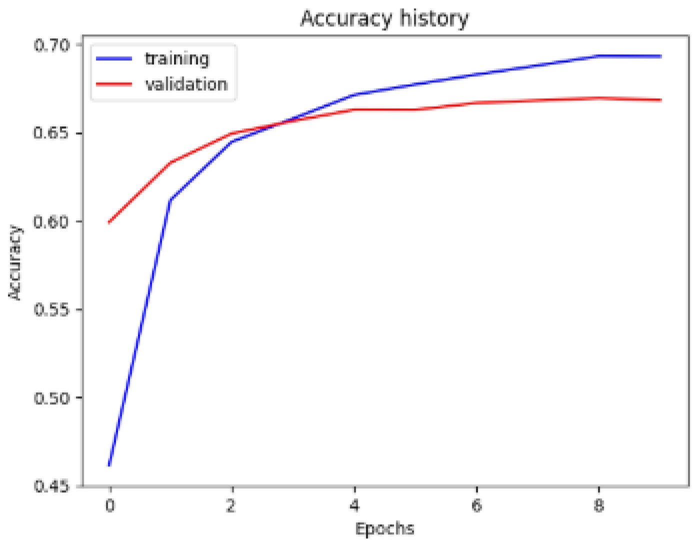
To tackle this issue, we trained a CNN using a dataset of approximately 130,000 images, which were already categorized into main categories and subcategories. The data was divided into training and testing sets. The CNN model achieved the following results:

- 66.8% accuracy in predicting the subcategory of a given image output \
- 80.0% accuracy in predicting the main categories of products

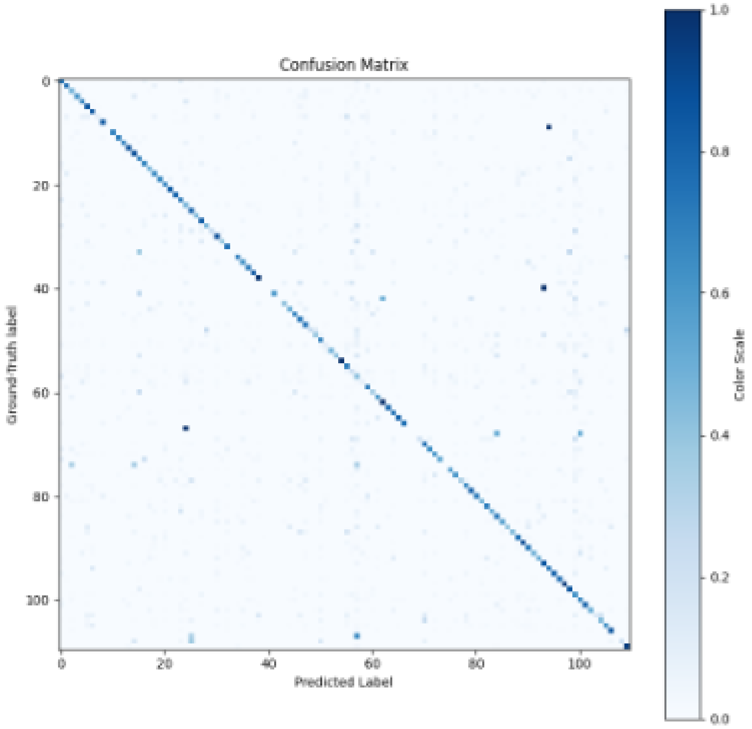
Training and Validation Loss Over Time:



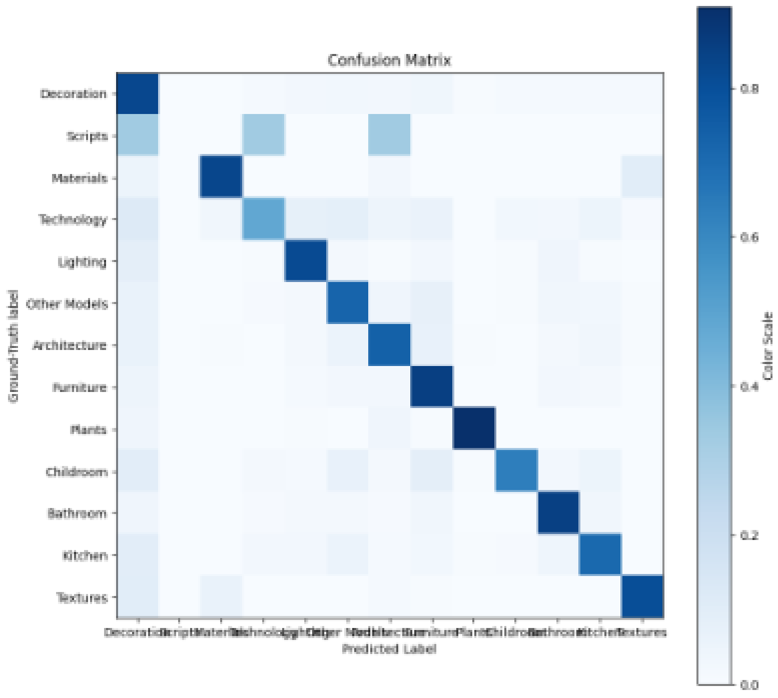
Training and Validation Accuracy Over Time:



Confusion Matrix of Subcategory:

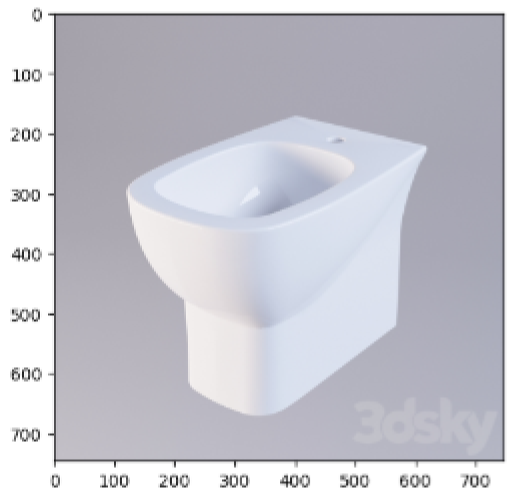


Confusion Matrix of Category:

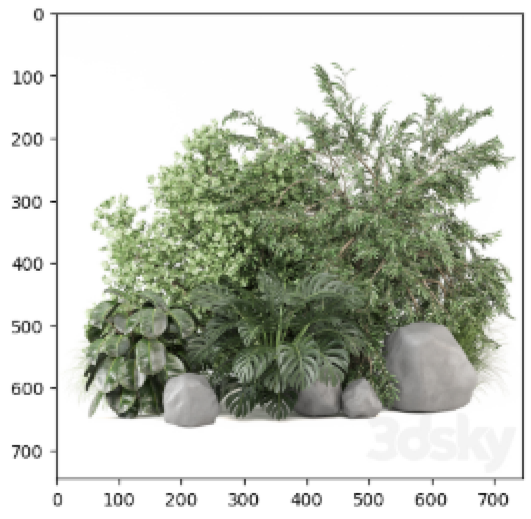


In terms of relative frequency, the highest misclassification occurs when the target subcategory is “bidet” and the predicted subcategory is “toilet & bidet”. In terms of the number of items, the highest misclassification occurs when the target subcategory is “outdoor” and the predicted subcategory is “bush”. For the two cases above, the target and predicted subcategory belong to the same category. In terms of relative frequency, the highest misclassification, where the category is also misclassified, occurs when the target subcategory is “glass” and the predicted subcategory is “tile”.

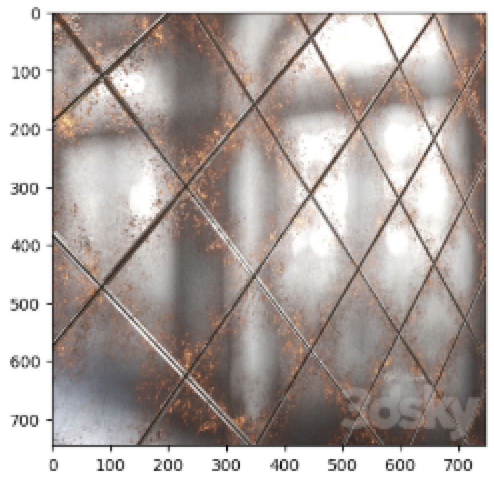
Item belonging to “bidet” misclassified as “toilet and bidet”:



Item belonging to “outdoor” misclassified as “bush”:



Item belonging to “glass” misclassified as “tile”:



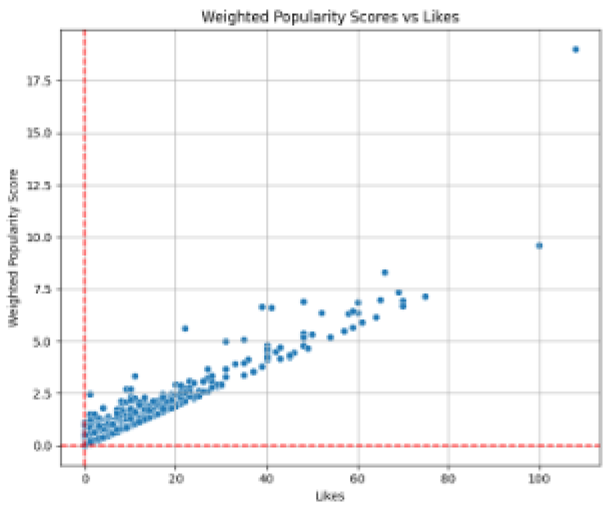
Discussion

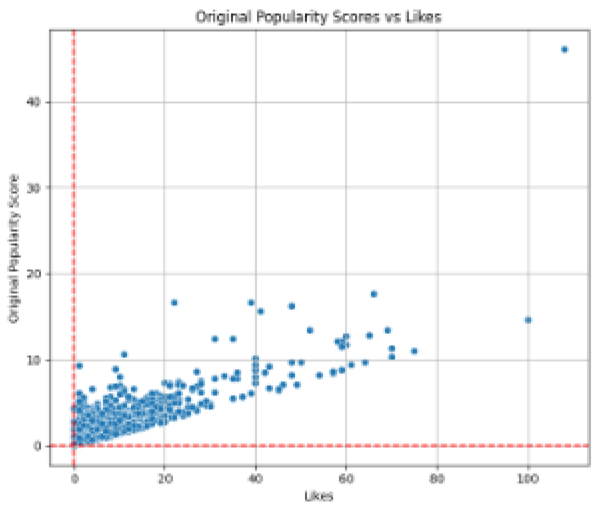
For the random forest model, the dataset was divided into a training and testing dataset in a ratio of approximately 80%/20%. The metrics for determining the performance of the random forest model included mean absolute error (MAE), mean squared error (MSE), root mean squared error (RMSE), and R squared values with the following values: 2.09e-4, 1.26e-7, 3.56e-4, 0.015. For the sensitivity analysis of the random forest model, the features like, recommendations, tags, sentiment, and comments had the following importances: 0.647, 0.277, 0.060, 0.008, 0.008. Although the R squared value indicates need for improvement in variance explanation, a low error rate indicates model accuracy.

In the model evaluation, feature importance analysis was combined with error and correlation metrics to evaluate the popularity scoring model. A Random Forest Regressor was utilized to analyze the importance of different features in predicting popularity scores for 3D models. By training the dataset on computed popularity scores (based on various parameters such as likes, recommendations, etc.) the random forest regressor can determine the sensitivity of each parameter. Feature importance is determined based on how much each attribute affects the prediction accuracy, providing insights into the relative impact of each factor. This sensitivity analysis allows us to optimize the popularity model by focusing on the most influential features, enhancing the model's interpretability and predictive power.

In addition to feature importance analysis, we evaluate our popularity scoring model by comparing the Weighted Popularity Score with an Original Popularity Score based on baseline weights. To quantify the difference between these scoring methods, we use Mean Absolute Error (MAE) and Mean Squared Error (MSE). The MAE provides a straightforward measure of the average absolute difference between the weighted and original scores, highlighting general alignment. The MSE, on the other hand, emphasizes larger deviations, offering insights into any significant discrepancies between the models. To further validate the model, we assess the correlation between each popularity score and the number of likes. The correlation values between likes and both the weighted and original scores reveal the strength and direction of association between user engagement and the computed scores, offering a measure of consistency and predictive relevance. Higher correlation values indicate that the model's popularity scores align closely with user interest, as indicated by likes. Together, these evaluation metrics ensure that our weighting approach produces a reliable and interpretable popularity score that can guide optimization efforts effectively.

Additionally, our new weighted popularity score and original popularity score were then measured against likes. They were measured against likes because likes represent a good approximate "ground truth" to popularity vs our calculated popularity score. This analysis showed that there was a much stronger correlation between the new weighted popularity score that had used the feature importance values got from the random forest regressor.





Correlation between Weighted Popularity Score and Likes: 0.9555894095939579

Correlation between Original Popularity Score and Likes: 0.8223807009734683

The silhouette score values given by the K-Means and DBScan results indicates that reasonable clustering is occurring. Since DBScan yields a slightly higher silhouette score, this indicates that non-linear regression methods might be more appropriate than linear regression. The vectorization does show some validity as tags such as “white”, “pink”, “yellow”, and “red” are associated with the second K-Means cluster. However, it seems that the encoding methods would have to be further tested or modified to see trends regarding tags. Due to the success of the clustering, it confirms that it is valid to use non-numerical features such as tags, categories, subcategories, style, and material to confirm the number of likes and recommendations.

For the evaluation of CNN, specifically ResNet34 included multiple metric-based analysis to evaluate the performance. Training and validation accuracy were tracked over multiple epochs to ensure the model’s learning and generalization. ResNet34 was able to achieve the highest validation accuracy with a score of 66.8% for the subcategory over 10 epochs. We have tested several implementations of the CNN: with black and white images and ResNet 18 (~62% validation accuracy), with black and white images and ResNet34 (~64%), with color images and ResNet34 (66.8% validation accuracy), and with color images, ResNet34, & a dropout layer (~64%). The decrease in performance from the addition of the dropout layer is due to accidentally including the dropout layer during the validation phase. Cross entropy loss was the criterion that was used. Since training and validation loss values consistently decreased across the epochs, it was shown that neither overfitting nor underfitting occurred.

Comparison of our various approaches

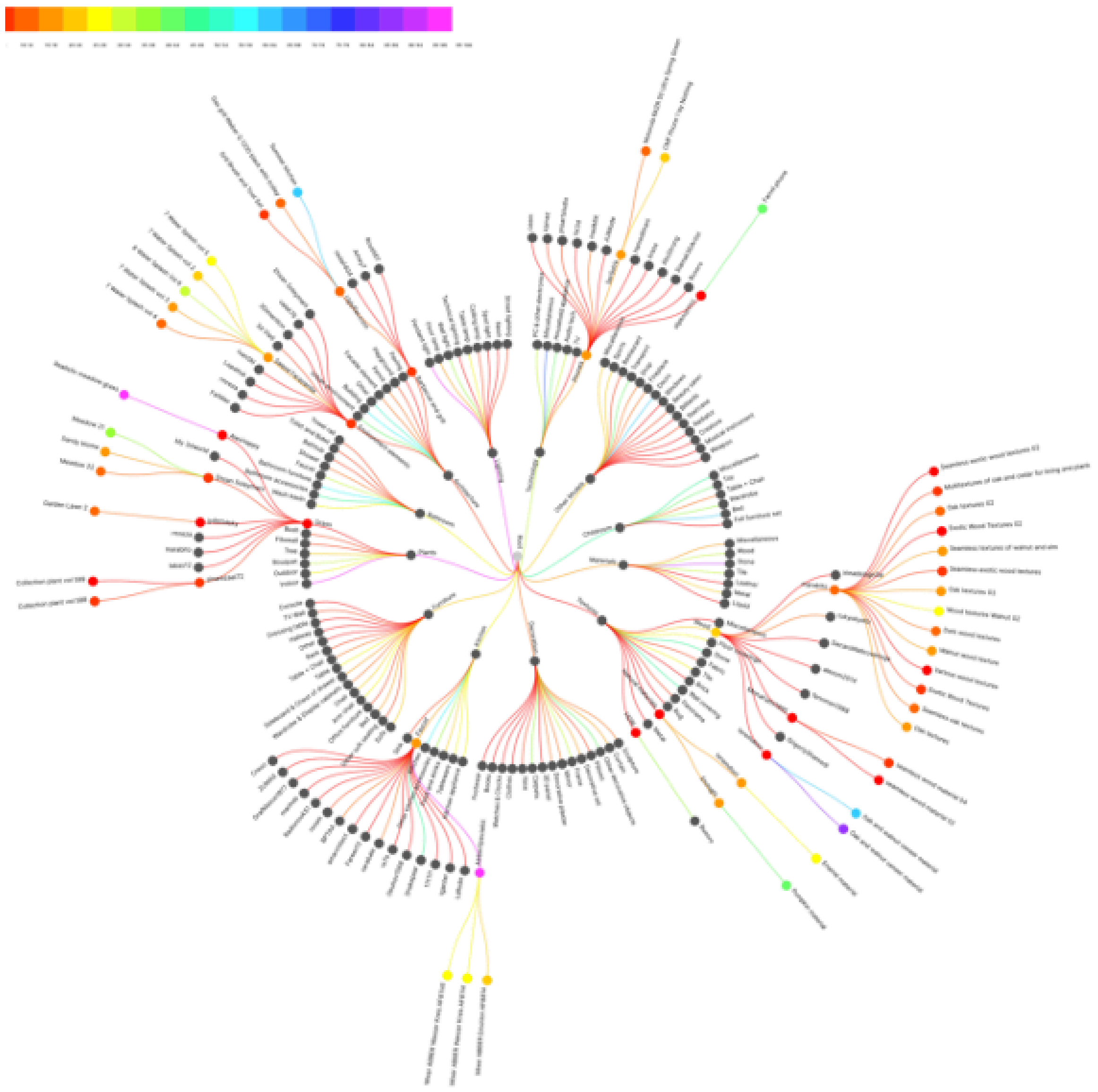
In the first approach, using K-Means, we focused on clustering items belonging to specific product tags by leveraging two critical metrics: the calculated popularity score and the recommendation frequency provided by the website. These metrics were used to determine an optimal set of tags and attributes that publishers could use to increase the visibility and popularity of their products within specific subcategories. The calculated popularity score incorporated user engagement metrics such as likes, recommendations, sentiment, and comments, while the recommendation frequency highlighted the interconnectedness of a product with others on the platform. Both methods successfully identified tag combinations that enhance a product’s potential for engagement and market reach.

In the second approach, using K-Means and DBSCAN with the vectorized tags, items within a subcategory were clustered based on likes, recommendations, vectorized tags, and sentiment score. Based on the clustering score, moderate amount of clustering did occur. From this, especially using the centers of K-Means, we can identify the tags typically associated with clusters that have a high number of likes or recommendations.

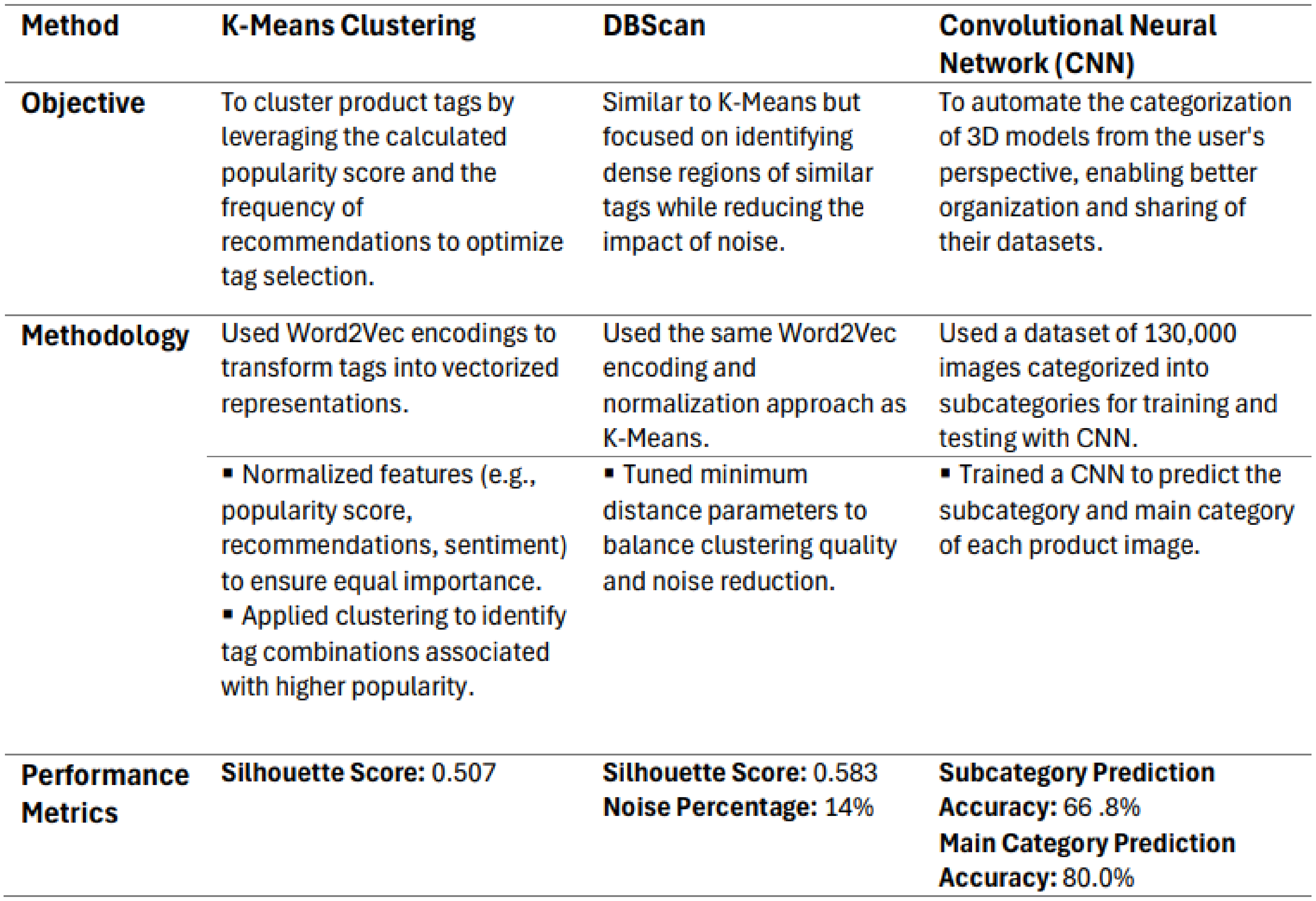
For our third approach, we adjusted the problem to adopt a user-centric perspective to leverage the capabilities of Convolutional Neural Networks (CNNs). While the first two methods aimed to assist publishers, the CNN model was designed to address challenges users face in managing their 3D models. Typically, when users initially interact with organized repositories, they benefit from the website's structured setup (see figure below). However, once they download these models, they often become scattered, disorganized, and difficult to share or reuse. This lack of universal organization results in thousands of models being stored without any standardized categorization, making their management inefficient.

The CNN approach demonstrated significant potential for improving data organization. By automating the categorization process, users can efficiently organize and share their 3D models in a standardized structure, regardless of the original website's organization system. This universal framework can greatly enhance the usability and accessibility of 3D models across various platforms.

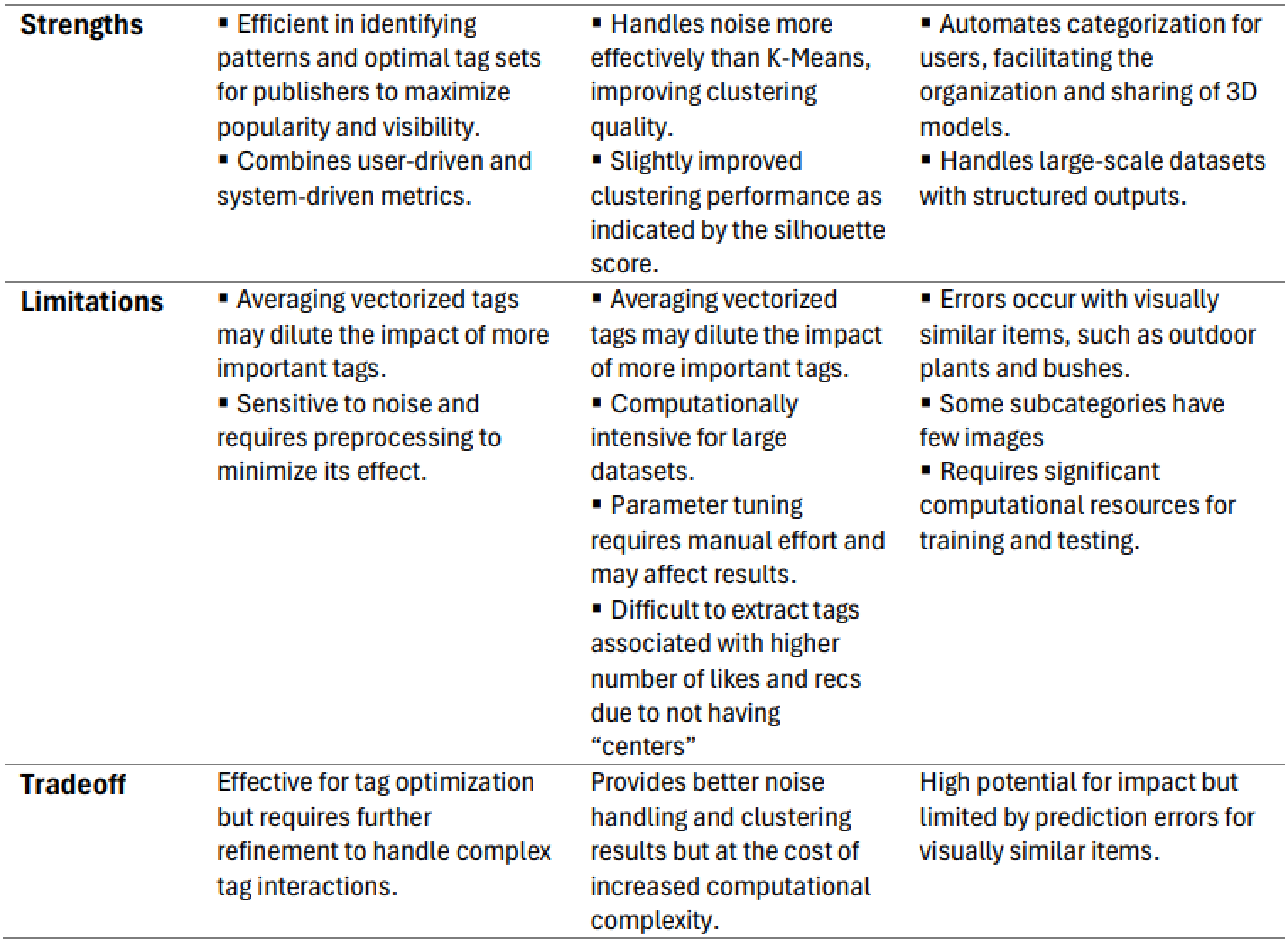
The following image is a visualization of categories and subcategories of 3d products offered on the reference website, with nested publisher ID's and top 50 product names. Edges are color coded based on the number of products in each branch and products are sorted and color coded based on popularity scores in the final layer using radial tree visualization script from D3.js.



Please refer to the following table for a summarized report of our methods and evaluations.



Method	**K-Means Clustering**	**DBScan**	**Convolutional Neural Network (CNN)**
Objective	To cluster product tags by leveraging the calculated popularity score and the frequency of recommendations to optimize tag selection.	Similar to K-Means but focused on identifying dense regions of similar tags while reducing the impact of noise.	To automate the categorization of 3D models from the user's perspective, enabling better organization and sharing of their datasets.
Methodology	Used Word2Vec encodings to transform tags into vectorized representations.	Used the same Word2Vec encoding and normalization approach as K-Means.	Used a dataset of 130,000 images categorized into subcategories for training and testing with CNN.
	▪ Normalized features (e.g., popularity score, recommendations, sentiment) to ensure equal importance. ▪ Applied clustering to identify tag combinations associated with higher popularity.	▪ Tuned minimum distance parameters to balance clustering quality and noise reduction.	▪ Trained a CNN to predict the subcategory and main category of each product image.
Performance Metrics	**Silhouette Score:** 0.507	**Silhouette Score:** 0.583 **Noise Percentage:** 14%	**Subcategory Prediction Accuracy:** 66 .8% **Main Category Prediction Accuracy:** 80.0%



Strengths	▪ Efficient in identifying patterns and optimal tag sets for publishers to maximize popularity and visibility. ▪ Combines user-driven and system-driven metrics.	▪ Handles noise more effectively than K-Means, improving clustering quality. ▪ Slightly improved clustering performance as indicated by the silhouette score.	▪ Automates categorization for users, facilitating the organization and sharing of 3D models. ▪ Handles large-scale datasets with structured outputs.
Limitations	▪ Averaging vectorized tags may dilute the impact of more important tags. ▪ Sensitive to noise and requires preprocessing to minimize its effect.	▪ Averaging vectorized tags may dilute the impact of more important tags. ▪ Computationally intensive for large datasets. ▪ Parameter tuning requires manual effort and may affect results. ▪ Difficult to extract tags associated with higher number of likes and recs due to not having “centers”	▪ Errors occur with visually similar items, such as outdoor plants and bushes. ▪ Some subcategories have few images ▪ Requires significant computational resources for training and testing.
Tradeoff	Effective for tag optimization but requires further refinement to handle complex tag interactions.	Provides better noise handling and clustering results but at the cost of increased computational complexity.	High potential for impact but limited by prediction errors for visually similar items.

Possible Improvements and Suggestions for Future Directions

For the K-Means with the word2vec encodings, it is noted that representing the product as the average of its tags may not be accurate since some tags may be more important than others. It is noted that other word encoding methods should be tested such as GloVe, BERT, and term frequency – inverse document frequency methods. Furthermore, features such as style and material should also be encoded.

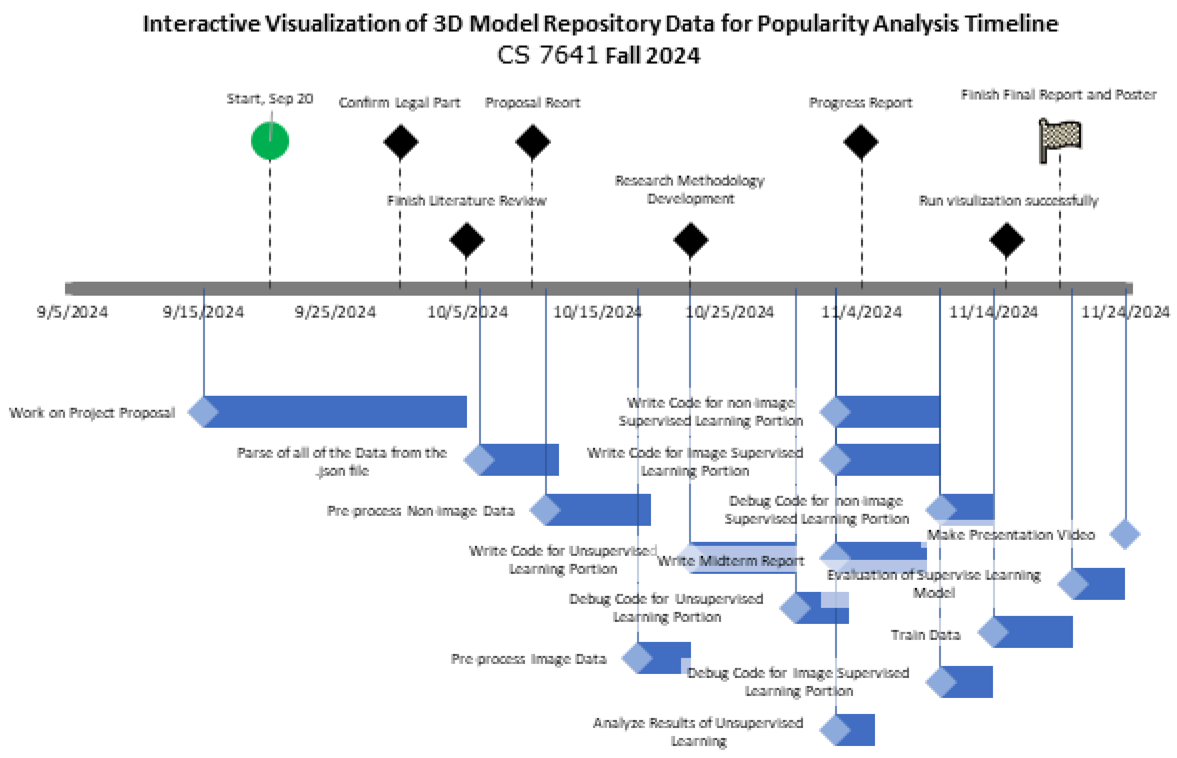
There is a possible modification to make to the project to use transfer learning. We could get a pretrained convolutional neural network such as ResNet 18 or ResNet 34. and fine tune it on classification of category, subcategory, material, style, and tags. Afterwards we can modify the model for regression and determine the number of likes and recommendations. This is a plausible idea since, based on the K-Means and DBScan results, it is indicated that there exists some relationship between the number of likes, recommendations, tags, and sentiment score.

For CNN, implementing Transfer Learning could improve accuracy. To address the accuracy limitations in subcategory predictions, we recommend adopting a transfer learning approach. First, a pre-trained model can be fine-tuned to predict main categories, leveraging the broader and more easily distinguishable patterns. Subsequently, the model can be further trained to predict subcategories within a single main category. This hierarchical approach is expected to improve the model's accuracy by focusing on nuanced distinctions within subcategories after establishing a robust foundation at the category level. In addition, to prevent risk of overfitting, dropout layers could be added throughout the neural network. Furthermore, since some subcategories had very few images, generative adversarial networks (GANs) could be used for data augmentation of those subcategories.

To improve model generalization and reduce prediction errors, we suggest using a larger and more diverse dataset for training and testing. Additionally, testing the models with images sourced from other 3D model repositories will help evaluate the robustness of our approach across varied platforms and organizational structures. This expansion would also provide insights into the adaptability of the models to different tagging systems and data formats.

While our CNN approach focused primarily on image-based predictions of categories and subcategories, integrating additional features such as tags, attributes, and metadata from the dataset could significantly enhance the model's predictive capabilities. Tags and other textual attributes often provide semantic context that complements visual data, helping the model make more informed and accurate predictions. Using a multimodal approach that combines image and metadata inputs could result in a more comprehensive categorization framework.

Gantt Chart



Link to Gantt Chart

Contribution Table

Name	Proposal Contributions
Kamyar Fatemifar	Problem Definition, Dataset Description, Methods (Implemented ML Models), GitHub Code Upload
Suchismitaa Chakraverty	Introduction/Background, Problem Definition, Methods (Preprocessing, Implemented ML Models), References, Final Report Edits
Joseph Hardin	Methods (Implemented ML Models), Contribution Table, Report Website, Final Report Edits, GitHub Code Upload
Abhishek Misra	Methods (Implemented ML Models), Results/Evaluation Metrics, Discussion
Max Pan	Methods (Implemented ML Models), Results/Evaluation Metrics, Gantt Chart, GitHub Code Upload

References

[1] Y. Yang, H. Lin., and Y. Zhang, “Content-Based 3D Model Retrieval: A Survey,” *IEEE Trans. Syst., Man, Cybern. C,* vol. 37, no. 6, pp. 1081-1098, Nov. 2007, doi: 10.1109/TSMCC.2007.905756

[2] J. Flotynski and K. Walczak, “Customization of 3D content with semantic meta-scenes,” *Graphical Models*, vol. 88, pp. 23-39, Nov. 2016, doi: 10.1016/j.gmod.2016.07.001

[3] C.E. Catalano, M. Mortara, M. Spagnuolo, and B. Falcidieno, “Semantics and 3D media: Current issues and perspectives,” *Computers & Graphics*, vol. 35, no. 4, pp. 867-877, Aug. 2011, doi: 10.1016/j.cag.2011.03.040

[4] S. Biasotti, A. Cerri, A. Bronstein, and M. Bronstein, “Recent Trends, Applications, and Perspectives in 3D Shape Similarity Assessment,” *Computer Graphics Forum*, vol. 35, no. 6, pp. 87-119, Sep. 2016, doi: 10.1111/cgf.12734

[5] H. Laga, M. Mortara, and M. Spagnuolo, “Geometry and context for semantic correspondences and functionality recognition in man-made 3D shapes,” ACM Trans. Graph., vol. 32, no. 5, pp. 1–16, Sep. 2013, doi: 10.1145/2516971.2516975.

[6] S. Hijazi, R. Kumar, C. Rowen, “Using Convolutional Neural Networks for Image Recognition”, Computer Science, 2015.

[7] A. R. Mesa, J. Y. Chiang, “Multi-input Deep Learning Model with RGB and Hyperspectral Imaging for Banana Grading”, Agriculture, July. 2021.

[8] L. Linhui, J. Weinpeng, W. Huihui, “Extracting the Forest Type from Remote Sensing Images by Random Forest”, IEEE, Dec. 2020, 10.1109/JSEN.2020.3045501. \

GitHub Repository

Link to GitHub Repository