Learning From the Best: What Makes Popular Hugging Face Models? A Registered Report

Yinan Wu
Department of Computer Science
North Carolina State University
Raleigh, USA
ywu92@ncsu.edu

Zhou Yang
Department of Computing Science
University of Alberta
Edmonton, Canada
zy25@ualberta.ca

Bowen Xu
Department of Computer Science
North Carolina State University
Raleigh, USA
bxu22@ncsu.edu

Bo Wang
School of Computer Science and Technology
Beijing Jiaotong University
Beijing, China
wangbo_cs@bjtu.edu.cn

Bach Le
School of Computing and Information Systems
The University of Melbourne
Melbourne, Australia
bach.le@unimelb.edu.au

David Lo
School of Computing and Information Systems
Singapore Management University
Singapore
davidlo@smu.edu.sg

***Abstract*—Background: Hugging Face has become a central open-source platform for sharing and reusing pre-trained models across the world. Models with higher popularity gain greater visibility, leading to increased adoption and broader impact in both applications and research domains. However, as the number of available models continues to grow, it remains unclear what makes a model stand out in this ecosystem. Practitioners lack concrete guidelines for improving the visibility of their models.**

Aim: Our study seeks to identify which models' features are associated with model popularity on Hugging Face, thereby presenting practical guidelines for practitioners to follow based on our identified features.

Method: We plan to extract features from models' metadata across four complementary dimensions: *Documentation*, *Models*, *Platforms*, and *Techniques*. Based on these features, we propose to build prediction models to identify the most important features and assess their effectiveness in popularity prediction.

***Index Terms*—Hugging Face, Pre-trained Models, Software Popularity, Mining Software Repositories**

I. Introduction

Artificial Intelligence (AI) models, especially pre-trained models (PTMs), have achieved phenomenal success in many domains, including natural language processing (NLP) [1], [2], computer vision (CV) [3], speech recognition [4], and code generation [5]. The advances and development of these models have been greatly facilitated by the open-source community. Practitioners share their models on platforms like Hugging Face, allowing more users to further reuse and expand on their work, e.g., fine-tuning models for new tasks. New models are continuously being shared to the platform, leading to a growing and diverse ecosystem of AI models [6]. This sharing-extension-sharing cycle has led to a practical requirement of pre-trained model practitioners:

How to make their models more popular so that they are visible to more users and can have a greater impact?

High popularity is important as it often reflects perceived usefulness, quality, and community trust [7]–[9]. In open-source software (OSS) platforms like GitHub, popularity is typically measured using indicators such as number of stars and forks [7]–[10]. Most existing studies on popularity on GitHub have analyzed how project features (e.g. repository's age, number of contributors) affect popularity indicators [7], [9], [11]. These studies extract certain features within projects and aim to predict which projects are likely to gain attention based on selected features. Therefore, they can provide valuable insights for developers to boost their projects' popularity. In comparison, the features that drive the popularity of pre-trained models remain unexplored. Prior research on Hugging Face models has concentrated on model reuse, usability, and documentation quality [12]–[14]. However, these studies have not discovered what makes a model more popular in this ecosystem.

Models with high popularity tend to appear frequently in search results and platform interfaces, making them more visible to individual users and organizations. This increased visibility can lead to more likes and downloads, greater community engagement (e.g. more discussions and pull requests), and ultimately, a broader impact across application and research domains. Taking Hugging Face as an example, it provides the "rank by number of likes (descendingly)" and "rank by number of downloads (descendingly)" options for searching for models. This means that models with more likes or downloads are more visible to users and thus have a higher chance of being reused and extended [6], [12], [15]. Besides, models with high popularity are considered important in research. Many empirical studies in machine learning and software engineering domains select highly popular models as representative or benchmark subjects for experimentation [16].

Despite the practical and academic value of model popularity, little is known about what drives popularity in the

Hugging Face ecosystem. There is still a lack of empirical evidence and actionable guidance on how features influence their popularity. Therefore, it motivates us to explore a key problem: *what makes popular models on Hugging Face?* To answer this question, our study will analyze the differences between popular and unpopular models to identify the key features that drive a model's popularity. These key features lead to actionable insights for developers and researchers to enhance the visibility and impact of their models, facilitating the development of AI models.

To measure the popularity of models on Hugging Face, we consider two indicators provided by the platform: the number of likes and the number of downloads. The number of likes is similar to GitHub stars, which reflects community endorsement from registered users [9]. The number of downloads indicates recent usage and community engagement. Rather than relying on a single indicator in the previous study [7], [8], we analyze both indicators independently. This allows us to capture complementary aspects of popularity and generate broader empirical insights to inform practitioners on how to make their models popular.

We will collect a fresh snapshot of model metadata and repository content from Hugging Face. We will explore and extract features from each model repository, categorized into four dimensions: *Documentation*, *Models*, *Platforms*, and *Techniques*. To analyze which features are associated with popularity indicators, we will first apply non-parametric statistical tests, including the Mann-Whitney U test and chi-square test, to identify significant differences between popular and unpopular models. We will then build classifiers to assess the predictive power of each feature and rank their importance using permutation-based methods. Finally, we will evaluate whether our proposed features would generalize across different model domains (e.g., NLP, CV) and contributor affiliations (e.g., community, companies) by performing cross-domains and cross-affiliations predictions. Primary expected contributions from this proposed study are as follows:

- Empirical Insights: We will analyze how specific model features relate to popularity (downloads and likes), offering a deeper understanding of what drives model success on Hugging Face, and identifying the most influential features that contribute to model appeal and visibility.
- Practical Recommendations: Based on our analysis, we will propose actionable recommendations for developers and researchers aiming to increase their models' visibility and impact. These guidelines will be tailored to the unique context of the Hugging Face platform, providing clear directions for improving model popularity.
- Research Foundation: This work will establish a framework for future studies on AI model development and adoption practices.

II. Background and Related Work

A. Popularity in Open-Source Community

Understanding what makes OSS projects popular has been a long-standing focus in software engineering research. Most prior studies have centered on GitHub, where popularity is typically measured using indicators such as stars and forks [9]–[11]. Researchers have examined how various project-level features, including documentation quality, repository age, contributor count, and commit frequency, are related to popularity indicators. For example, Zhu et al. [10] found that the presence of standard folders such as docs, tests, and examples was associated with a higher number of forks, indicating higher popularity correlated with greater potential for reuse. Borges et al. [9] then performed a large-scale analysis of GitHub repositories and showed that documentation quality and maintenance activity significantly influence star growth.

These studies demonstrate that repository metadata and structural features are important in determining popularity within OSS ecosystems. However, they are primarily focused on project-based platforms and have not been extended to model-sharing platforms such as Hugging Face. Our work fills this gap by studying models' popularity at Hugging Face.

B. Model Sharing and Analysis on Hugging Face

Hugging Face has emerged as a leading platform for sharing pre-trained models in NLP, CV and other fields [1], [2], [5], [17]. Recent research has begun to investigate how models on this platform are used, adapted, and documented. Jiang et al. [12], [18] conducted empirical studies from model naming practices to model reuse, versioning, and adaptation across tasks. Pepe et al. and Liang et al. [13], [14], [19] then focused on documentation practices and the transparency of model cards. Similarly, Jones et al. [15] define an integer scale from 0 to 5 as a proxy of a model's documentation quality, with higher scores meaning better documentation. They find the top 1000 most popular models exhibited higher scores than the bottom 1000 models.

While these studies provide valuable insights into how models are maintained and consumed, they focus largely on downstream aspects such as usability and documentation quality. They do not address what drives a model's popularity, or adoption within the Hugging Face ecosystem. Our study aims to close this gap by examining which model-based features are associated with higher popularity, leading to broader impact.

C. Feature Engineering and Popularity Modeling

Feature-based prediction approaches have been widely used to analyze OSS outcomes. For example, Bao et al. [20] applied feature engineering techniques to developer activity and working hour data in order to predict employee attrition, using classifiers such as naive Bayes, SVM, decision trees, kNN, and random forests. Wang et al. [7] explored how various aspects of README files affect the popularity of GitHub repositories. The most relevant work to ours is by Fan et al. [8], who extracted 21 features from GitHub repositories of academic AI projects, categorized into three dimensions: *documentation*, *code*, and *reproducibility*. They applied random forest classifiers to identify which metadata fields had the greatest predictive power. They find features like the number of links to other GitHub repositories and the number of images in

the README file, significantly influence the number of stars an academic AI repository receives.

Building on this line of work, our study shifts the focus from software repositories to pre-trained models hosted on Hugging Face. We extract 32 features across four dimensions, including *Documentation*, *Models*, *Platforms*, and *Techniques*. Our goal is to investigate which of these features are most correlated with model popularity, as measured by likes and downloads on Hugging Face. To the best of our knowledge, this is the first exploratory study that aims to systematically analyze how model features influence visibility and engagement.

III. Data Collection Plan

Hugging Face provides the official API endpoints [21] to retrieve hosted models' information. We will call the API to collect a timely and comprehensive snapshot of all public model repositories available to ensure data freshness. As the recently added models may not have enough time to accumulate downloads and likes, we excluded the models added within 180 days before our data collection date. Such a strategy of excluding newly added data entries is widely adopted in previous studies [8]. Each Hugging Face model is hosted as a Git repository. We notice that some repositories may not contain model-related files (e.g., files with '`.bin` or '`.safetensors` extensions') and thus fall out of our study scope. We observe that repositories without model files are often small in size, thus we will filter out those repositories that are smaller than 300MB. We would use this filtered dataset to conduct subsequent analysis.

We will then employ another API [22] to extract the detail information of each model, including the number of downloads and likes, whether the model is using Hugging Face Spaces, etc. We will retrieve each model's README file by accessing the URL https://huggingface.co/{model-ID} /blob/main/README.md, where the model-ID is the name of the model, e.g., `microsoft/codebert-base`. We will use the extracted metadata and README files to distinguish features, as explained in Section IV.

To address potential bias introduced by highly reused base models (e.g., BERT or CodeBERT), which may have artificially inflated download counts, we adopt both automated and manual strategies to identify such models. First, we use the Hugging Face model tree metadata [23] to identify models that are referenced as base models by others. Second, we heuristically include models whose names contain the suffix "`-base`" or whose README files explicitly describe them as base models. All identified base models will then be manually reviewed for confirmation. To assess their influence, we will conduct a robustness check by repeating our research questions analyses with and without these base models.

IV. Features Extraction Plan

Existing studies have uncovered some features that may influence the popularity of software projects and AI models [8], [15], [18]. However, they fall short of providing a comprehensive feature set that are designed for Hugging Face models. Although Hugging Face models are hosted as Git repositories, we propose that they would have unique features that are different from general software repositories.

To better understand model popularity and publishing practices on Hugging Face, we initially propose a set of 32 features as shown in Table I, informed by prior work and our own observations. These features span four dimensions: *Documentation*, *Models*, *Platforms*, and *Techniques*. While this set forms the foundation of our analysis, we plan to explore additional features in future iterations of the study.

1) Documentation Dimension: Previous studies have shown that maintaining high-quality documentation is crucial for the success and popularity of open-source projects [7], [8], [24], [25]. Intuitively, effective documentation can help interested users understand the repository's purpose, features, and usage, which is essential for attracting users to download and like the model. Hugging Face hosts models as Git repositories and automatically generates a documentation page (also known as *model card*) if a README.md file is present in the repository. The README file consists of two parts: (1) descriptions written in Markdown format and (2) metadata written in YAML format. We will focus on analyzing the first part, which is just like the README file in a general software repository.Following Fan et al. [8], we adopt the Mistune library to parse Markdown files and extract documentation features. We reuse and extend their proposed features, obtaining a total of 14 features for the documentation dimension.

Intuitively, a longer README file indicates more detailed information about the model. Thus, we will use `length-yaml` and `length-doc` to represent the length of the YAML and Markdown files, respectively. Using lists and tables to present information is a sign of clear writing [26], so we will measure `num-list` and `num-table` (the number of lists and tables in the README file).

Code snippets are also important in documentation. Yang et al. [27] find that unclear code instructions to run the model is one of the main reasons for users to raise issues in GitHub AI repositories. We will measure the number of code blocks (`num-code-blk`) and the number of inline code (`num-inline-code`). Other formats of visualization help users understand the model include images and videos. Following Fan et al. [8], we would take `num-static-img` (e.g., files ending with '.jpg', or '.png'), `num-animated-img` and `has-video` into consideration.

Fan et al. [8] also count the number of GitHub links (`num-gh-link`) in the README file, we therefore will consider the number of Hugging Face links (`num-hf-link`). As many models are based on academic papers, we will count the number of arXiv links (`num-arxiv-link`). Additionally, we will extract two binary features: `has-license` and `has-bibtex`, if the metadata contains license information and a BibTeX entry.

2) Model Dimension: This study specifically analyzes the popularity of Hugging Face models; thus we need to extract and analyze the features that are unique to models. How these model-related features affect popularity is mainly ignored in

TABLE I
THE FEATURES OF HUGGING FACE MODELS ANALYZED IN OUR STUDY AND THEIR CORRESPONDING DESCRIPTIONS.

Dimensions	Feature Name	Description
Documentation	length-yaml	The total number of characters in the YAML part.
	length-doc	The total number of characters in the Markdown part.
	has-video	Whether the documentation includes URLs to videos.
	num-inline-code	The number of inline code snippets in the documentation.
	num-code-block	The number of code blocks in the documentation.
	num-static-img	The number of static images (e.g., .jpg, .png).
	num-animation	The number of animated images (e.g., .gif).
	num-lists	The number of lists used in the documentation.
	num-tables	The number of tables included in the documentation.
	num-gh-links	The number of GitHub links in the documentation.
	num-hf-links	The number of Hugging Face links in the documentation.
	num-arxiv-links	The number of arXiv links in the documentation.
	has-bibtex	Whether a BibTeX entry is provided in the metadata.
	has-license	Whether license information is provided in the metadata.
Models	has-config	Whether the model repository contains a configuration file.
	has-results	Whether the model repository contains results demonstrating the model's performance.
	has-dataset	Whether the metadata specifies any dataset information.
	num-dataset	The number of datasets linked to the model.
	match-hf-dataset	Whether the linked datasets are hosted on Hugging Face.
	model-size	The total size of the model files in the repository.
	num-root-files	The number of files in the root directory.
	num-modules	The number of modules (subdirectories) in the repository.
	num-model-files	The number of model files (e.g., files with .bin or .safetensors extensions) in the repository.
	has-quantized	Whether the model is quantized.
Platforms	has-space	Whether the model is featured in Hugging Face Spaces.
	has-safetensors	Whether the model uses SafeTensors to store model weights.
	has-widgets	Whether the model's inference API contains widget data.
Techniques	has-pipeline	Whether the model's metadata specifies a named pipeline.
	has-impl-lib	Whether the model's metadata specifies a primary implementation library.
	has-supported-lib	Whether the model uses a library officially supported by Hugging Face.
	has-additional-fw	Whether the model supports additional frameworks beyond the primary one specified.

previous studies [8], [15]. We will analyze the model cards (mainly the metadata part) of the models, as well as the structure and sizes of the repositories, to extract features in the model dimension.

Researchers find that the structure of a repository can influence the popularity of a software project [8]. For example, the number of modules (i.e., subdirectories) in a repository usually indicates how the project is organized and maintained. Thus, we will consider num-modules) in a Hugging Face model repository. Additionally, we will count the number of files in the root directory (num-root-files). Some files are relevant to the models, e.g., those with '.bin' or '.safetensors' extensions. We will count the number of these files in each model repository (num-model-files). We wiil check whether the model repository contains a configuration file (has-config), which is essential for loading the model. Model size can impact performance, deployment feasibility, and user accessibility. Accordingly, we define the model-size feature to capture this aspect.

Some model developers offer the information about their models' training datasets, which can improve transparency and enhance credibility. Therefore, we consider whether the model specifies any dataset metadata (has-dataset), the number of linked datasets ((num-dataset), and whether those datasets are hosted on Hugging Face (match-hf-dataset). In addition, models that report evaluation results can help users better understand their capabilities and limitations. We include has-results feature to capture the presence of such reported results in the metadata. We also consider has-quantized to indicate whether the model has been quantized, as quantization often reflects real-world efficiency or consumption concerns [28].

We are interested in expanding the *Model* dimensional features by considering features that would reflect real-world usage or deployment cases, such as whether it is directly loaded by users, or indirectly used through integration in popular tools or libraries (e.g., BertScore or RAG libraries). These features may offer further insights into practical adoption guidelines beyond what is currently captured in metadata.

3) Platform Dimension: Hugging Face platform provides a set of advanced functions to help users better understand and use the models. We analyze the usage of these functions, which are categorized into the platform dimension.

We define the has-safetensor feature to indicate whether the model uses SafeTensor to store the model weights. Loading models from certain file formats (e.g., .pt or .bin files) relies on pickle (the default Python serialization library), which poses the risk of malicious code injection and execution. Hugging Face introduces SafeTensors for storing and loading model weights securely. Users may consider models using SafeTensors to be more secure and trustworthy, which can enhance the model's popularity.

Hugging Face provides a function called *Spaces*, which allows users to host, share, and run models on the Hugging Face platform without the need for configuring servers or environments. The space is then accessible to the public, and users can interact with the model through the Hugging Face website, which can enhance the model's visibility and accessibility. Another function allowing users to easily interact with the model is the *Widget*. This function provides a widget

on the right side of the model's webpage, and users can run the model with default examples of text or data. We define the `has-space` and `has-widget` features to indicate whether the model is featured in Hugging Face Spaces and whether the model's inference API contains widget data, respectively.

4) Techniques Dimension: We define several features in the techniques dimension to capture the technical aspects of the models, focusing on their implementation details and supported frameworks.

We introduce the `has-pipeline` feature to check whether the model's metadata specifies a named pipeline, indicated by the presence of the `pipeline_tag` field. In Hugging Face, a pipeline is a set of APIs that simplify the model inference process, allowing users to perform tasks like text classification or named entity recognition by providing inputs without writing additional code for tokenization or decoding. Models that specify a pipeline can be more easily discovered by users searching for solutions to specific tasks, potentially increasing their visibility and popularity.

Then, we define the `has-implementation-lib` feature to indicate whether the model's metadata specifies a primary implementation library through the `library_name`. This field reveals the main library used to implement the model (e.g., `transformers`), helping users understand how to load and utilize the model effectively. We will use the `has-supported-lib` feature to determine if the model uses an officially supported library by Hugging Face. Hugging Face officially supports libraries such as `transformers`, and `diffusers` [29]. Models associated with these libraries can be loaded and used with minimal code, enhancing their accessibility and potentially their popularity.

Finally, we define the `has-additional-framework` feature to indicate whether the model supports additional frameworks (e.g., `pytorch`, `tensorflow`, `jax`) beyond the primary one specified. Models supporting multiple frameworks offer greater flexibility to users, which may increase their attractiveness and popularity. To validate feature extraction, we will conduct a manual sanity check on a statistically representative subset of our collected models, including but not limited to verifying extracted values against the actual metadata and README content.

V. Research Questions and Analysis Plan

We use two indicators of popularity on Hugging Face: the number of likes and the number of downloads. Likes may reflect user appreciation, while downloads indicate recent usage in the past 30 days. Since these two metrics may capture different aspects of popularity, we compute **Spearman's rank correlation coefficient (ρ)** to assess their relationship. Following prior work [7], we interpret $|\rho| < 0.3$ as weak, $0.3 \leq |\rho| < 0.7$ as moderate, and $|\rho| \geq 0.7$ as strong correlation. A weak or moderate correlation would justify analyzing them as distinct outcomes in our subsequent analyses.

RQ1: Are there significant differences in features between popular and unpopular models?

We begin our study by identifying the characteristics that distinguish highly popular models from less popular ones. Understanding these differences can reveal practices or attributes associated with greater visibility and adoption. We expect popularity to follow a long-tail distribution [8], meaning a comparison between the extremes is necessary to highlight key differentiators. We will perform this analysis separately for both likes and downloads.

Our execution plan involves several steps:

1) Grouping: We will rank models based on likes and downloads separately. Inspired by Fan et al [8], we will define *popular* models as those in the top 10% and *unpopular* models as those in the bottom 80%. We exclude the middle 10% as a 'buffer zone' to establish a clearer contrast between the groups, enhancing our ability to detect significant differences. We will also explore alternative splits (e.g., top and bottom 10–20% with an explicit middle group) by the actual distribution.
2) Feature Comparison: We will compare these groups across our set of predefined model features (e.g., model size, documentation, metadata, framework).
3) Statistical Tests: For continuous/numeric features, after checking for normality (e.g., using Shapiro-Wilk [30]) and anticipating non-normal data, we will apply the non-parametric Mann-Whitney U test [31]. For binary/categorical features, we will use the Chi-square test of independence. We will use a significance threshold of $p < 0.05$, and will explore the use of Holm-Bonferroni [32] or False Discovery Rate (FDR) [33] to correct comparisons.
4) Effect Size Measurement: To gauge practical significance, we will calculate effect sizes. For the Mann-Whitney U test, we will use Cliff's Delta (δ), interpreting its absolute magnitude using established thresholds (negligible: 0-0.147, small: 0.147-0.33, medium: 0.33-0.474, large: $>$ 0.474). For the Chi-square test, we will compute Phi (ϕ) or Cramer's V.

RQ2: What are the most critical features that influence a model's popularity?

It is crucial to understand which features are the most influential in driving a model's popularity. Some features may be more effective in distinguishing a model's appeal, enabling practitioners and developers to adopt practices optimizing these key features. Our goal is to build predictive models and analyze them to identify which features hold the most predictive power for model popularity.

Our execution plan involves three main stages:

First, we will refine our feature set to enhance model performance and interpretability. This involves addressing multicollinearity by calculating Spearman's rank correlation coefficient (ρ) for all feature pairs. We plan to identify

pairs with a coefficient above a predefined threshold (e.g., $|\rho| > 0.7$) and remove one feature from each such pair to reduce redundancy [8], [20], [34]. We will also employ mutual information to assess the relationship between each remaining feature. Features will be ranked by MI score, and we will select the top-k features, where k will be tuned based on cross-validation performance [35], [36]. These steps aim to ensure that our models focus on features with stronger predictive relationships, thereby reducing noise and potential overfitting.

Second, using the refined feature set, we will train and evaluate predictive models. We will use the same 'popular' (top 10%) and 'unpopular' (bottom 80%) labels defined in RQ1. We plan to evaluate Decision Trees, Random Forests, Support Vector Machines (SVMs), Naive Bayes and K-Nearest (KNN) classifiers using a grid search for hyperparameter tuning, following practices from prior work [20]. To assess model performance, we will use either 10-fold cross validation [20], [34] or bootstrap resampling with 1,000 iterations [8]. We will report the mean Area Under the Curve (AUC) as our primary evaluation metric to select the best-performing model(s) for each popularity indicator. Accuracies, including precision, recall and F1-score will be reported as secondary metrics for completeness. A classifier's performance will be considered acceptable if it significantly exceeds a random baseline or the mean AUC is larger than 0.7 [8], [20], [34].

Third, to identify the most influential features, we will analyze the best-performing classifier(s). Our primary planned method is **Permutation Importance**. This technique involves systematically shuffling the values of each feature one at a time within a test set and measuring the decrease in the model's predictive performance (e.g., AUC). Features that cause a larger drop in performance when shuffled are considered more important. We will perform this analysis across multiple iterations (e.g., using bootstrap samples [8] or in the context of deep learning [37]) to obtain stable rankings. This analysis will highlight which features are most critical for predicting popularity, providing actionable guidance for practitioners seeking to enhance their models' reach and impact.

> **RQ3:** How effective are our proposed features in predicting popularity across different model domains and affiliations?

We will reuse the best-performing classifier identified in RQ2 to explore whether our proposed features generalize across different Hugging Face subgroups. Prior work has shown that feature-based models trained in one subgroup may not transfer well to others, such as across programming languages or project sizes [34]. Inspired by this, we will test whether a model trained on one domain or affiliation group performs well when applied to another. We also apply either 10-fold validation or bootstrap sampling to assess whether the features are consistent or context-dependent.

For domain labels, we will extract the `pipeline_tag` (e.g., "text-classification", "image-to-text") in each model's metadata. We will then map the tag into one of the higher-level domain categories: *Audio*, *Computer Vision*, *Multimodal*, *NLP*, *Tabular*, and *Reinforcement Learning*, based on our preliminary analysis. For affiliation labels, we will extract information from the model owner's Hugging Face profile page. Each model has an associated owner, and owners can choose an affiliation type that is publicly displayed on their Hugging Face page. We will collect this data by parsing the HTML source of the owner's page and extracting the affiliation tag. Based on our preliminary analysis, we have identified the following affiliation categories: *company*, *university*, *non-profit*, *organization or individual*, *community*, *unknown*, *classroom*, and *government*. This cross-group analysis allows us to explore whether popularity-related features are consistent across different sub-communities, or whether different groups require different optimization strategies.

VI. Threats to Validity

Threats to Internal Validity: Our results may be affected by biases introduced during feature extraction and data filtering. Although we carefully designed and categorized the 32 features to represent various aspects of model structure, documentation, and platform engagement, it is possible that certain important features are not captured. Additionally, our choice of classifiers and statistical tests (e.g., Mann-Whitney U) may introduce assumptions that influence outcomes, despite being widely used in similar prior work [8], [20].

Threats to External Validity: Our study focuses exclusively on Hugging Face models. Although this platform is one of the most popular hubs for pre-trained models, our findings may not generalize to other ecosystems such as GitHub, or domain-specific model registries (e.g., Pytorch Hub) [38]. Furthermore, popularity dynamics may evolve over time as community practices and interface algorithms change. We may require a study to assess the long-term stability of observed patterns.

Threats to Construct Validity: While counts of downloads and likes metrics are widely interpretable, they may not fully reflect long-term impact or actual usefulness. Likes can be driven by interface exposure or social dynamics, and automated pipelines may inflate download counts. However, using both indicators allows us to triangulate insights and reduce bias and reliance on any single indicator. In addition, we do not consider task-specific and model external factors. Instead, we focus on observable and actionable features at the metadata and repository levels. We believe this perspective provides complementary value. We focus on identifying influential features for prediction based on filtering strategies (e.g., correlation and mutual information) to reduce redundancy and noise. We acknowledge that this may overlook certain individual effects, which can be explored in future work. We also use permutation importance to assess feature relevance. While this does not indicate directionality, it offers a model-agnostic view of predictive contribution. Future work can incorporate more casual interpretable methods. Finally, although we filter out small or newly added models, some low-quality or inactive repositories may remain. We acknowledge this as a limitation and an area for refinement.

REFERENCES

[1] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[2] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[4] R. Ardila, M. Branson, K. Davis, M. Henretty, M. Kohler, J. Meyer, R. Morais, L. Saunders, F. M. Tyers, and G. Weber, "Common voice: A massively-multilingual speech corpus," *arXiv preprint arXiv:1912.06670*, 2019.

[5] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. D. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman *et al.*, "Evaluating large language models trained on code," *arXiv preprint arXiv:2107.03374*, 2021.

[6] Z. Yang, J. Shi, P. Devanbu, and D. Lo, "Ecosystem of large language models for code," *ACM Transactions on Software Engineering and Methodology*, 2024.

[7] T. Wang, S. Wang, and T.-H. P. Chen, "Study the correlation between the readme file of github projects and their popularity," *Journal of Systems and Software*, vol. 205, p. 111806, 2023.

[8] Y. Fan, X. Xia, D. Lo, A. E. Hassan, and S. Li, "What makes a popular academic ai repository?" *Empirical Software Engineering*, vol. 26, no. 1, Jan. 2021. [Online]. Available: http://dx.doi.org/10.1007/s10664-020-09916-6

[9] H. Borges, A. Hora, and M. T. Valente, "Understanding the factors that impact the popularity of github repositories," in *2016 IEEE international conference on software maintenance and evolution (ICSME)*. IEEE, 2016, pp. 334–344.

[10] J. Zhu, M. Zhou, and A. Mockus, "Patterns of folder use and project popularity: A case study of github repositories," in *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, 2014, pp. 1–4.

[11] H. Borges, A. Hora, and M. T. Valente, "Predicting the popularity of github repositories," in *Proceedings of the The 12th international conference on predictive models and data analytics in software engineering*, 2016, pp. 1–10.

[12] W. Jiang, N. Synovic, M. Hyatt, T. R. Schorlemmer, R. Sethi, Y.-H. Lu, G. K. Thiruvathukal, and J. C. Davis, "An empirical study of pre-trained model reuse in the hugging face deep learning model registry," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023, pp. 2463–2475.

[13] F. Pepe, V. Nardone, A. Mastropaolo, G. Bavota, G. Canfora, and M. Di Penta, "How do hugging face models document datasets, bias, and licenses? an empirical study," in *Proceedings of the 32nd IEEE/ACM International Conference on Program Comprehension*, 2024, pp. 370–381.

[14] W. Liang, N. Rajani, X. Yang, E. Ozoani, E. Wu, Y. Chen, D. S. Smith, and J. Zou, "What's documented in ai? systematic analysis of 32k ai model cards," *arXiv preprint arXiv:2402.05160*, 2024.

[15] J. Jones, W. Jiang, N. Synovic, G. Thiruvathukal, and J. Davis, "What do we know about hugging face? a systematic literature review and quantitative validation of qualitative claims," in *Proceedings of the 18th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, 2024, pp. 13–24.

[16] Y. Shen, K. Song, X. Tan, D. Li, W. Lu, and Y. Zhuang, "Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. 2023," *arXiv preprint arXiv:2303.17580*, 2023.

[17] M. Taraghi, G. Dorcelus, A. Foundjem, F. Tambon, and F. Khomh, "Deep learning model reuse in the huggingface community: Challenges, benefit and trends," in *2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2024, pp. 512–523.

[18] W. Jiang, C. Cheung, M. Kim, H. Kim, G. K. Thiruvathukal, and J. C. Davis, "Naming practices of pre-trained models in hugging face," *arXiv preprint arXiv:2310.01642*, 2023.

[19] W. Liang, N. Rajani, X. Yang, E. Ozoani, E. Wu, Y. Chen, D. S. Smith, and J. Zou, "Systematic analysis of 32,111 ai model cards characterizes documentation practice in ai," *Nature Machine Intelligence*, vol. 6, no. 7, pp. 744–753, 2024.

[20] L. Bao, Z. Xing, X. Xia, D. Lo, and S. Li, "Who will leave the company?: a large-scale industry study of developer turnover by mining monthly work report," in *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*. IEEE, 2017, pp. 170–181.

[21] Hugging Face, "Hub API Documentation," 2024, accessed July 2025. [Online]. Available: https://huggingface.co/docs/hub/en/api

[22] ——, "Model Repository API," 2024, accessed July 2025. [Online]. Available: https://huggingface.co/api/models

[23] ——, "Model Cards: Specifying a Base Model," 2024, accessed July 2025. [Online]. Available: https://huggingface.co/docs/hub/model-cards#specifying-a-base-model

[24] G. A. A. Prana, C. Treude, F. Thung, T. Atapattu, and D. Lo, "Categorizing the content of github readme files," *Empirical Software Engineering*, vol. 24, pp. 1296–1327, 2019.

[25] K. Aggarwal, A. Hindle, and E. Stroulia, "Co-evolution of project documentation and popularity within github," in *Proceedings of the 11th working conference on mining software repositories*, 2014, pp. 360–363.

[26] J. Kimble, "Plain english: A charter for clear writing," *TM Cooley L. Rev.*, vol. 9, p. 1, 1992.

[27] Z. Yang, C. Wang, J. Shi, T. Hoang, P. Kochhar, Q. Lu, Z. Xing, and D. Lo, "What do users ask in open-source ai repositories? an empirical study of github issues," in *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*. IEEE, 2023, pp. 79–91.

[28] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[29] Hugging Face, "List of model libraries in huggingface.js," https://github.com/huggingface/huggingface.js/blob/main/packages/tasks/src/model-libraries.ts, 2024, accessed July 2025.

[30] S. S. Shapiro and M. B. Wilk, "An analysis of variance test for normality (complete samples)," *Biometrika*, vol. 52, no. 3-4, pp. 591–611, 1965.

[31] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The annals of mathematical statistics*, pp. 50–60, 1947.

[32] S. Holm, "A simple sequentially rejective multiple test procedure," *Scandinavian journal of statistics*, pp. 65–70, 1979.

[33] Y. Benjamini and Y. Hochberg, "Controlling the false discovery rate: a practical and powerful approach to multiple testing," *Journal of the Royal statistical society: series B (Methodological)*, vol. 57, no. 1, pp. 289–300, 1995.

[34] L. Bao, X. Xia, D. Lo, and G. C. Murphy, "A large scale study of long-time contributor prediction for github projects," *IEEE Transactions on Software Engineering*, vol. 47, no. 6, pp. 1277–1298, 2021.

[35] J. R. Vergara and P. A. Estévez, "A review of feature selection methods based on mutual information," *Neural computing and applications*, vol. 24, pp. 175–186, 2014.

[36] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of machine learning research*, vol. 3, no. Mar, pp. 1157–1182, 2003.

[37] M. Wojtas and K. Chen, "Feature importance ranking for deep learning," *Advances in neural information processing systems*, vol. 33, pp. 5105–5114, 2020.

[38] W. Jiang, J. Yasmin, J. Jones, N. Synovic, J. Kuo, N. Bielanski, Y. Tian, G. K. Thiruvathukal, and J. C. Davis, "Peatmoss: A dataset and initial analysis of pre-trained models in open-source software," in *Proceedings of the 21st International Conference on Mining Software Repositories*, 2024, pp. 431–443.