ABSTRACT

Pre-trained Machine Learning (ML) models help to create ML-intensive systems without having to spend conspicuous resources on training a new model from the ground up. However, the lack of transparency for such models could lead to undesired consequences in terms of bias, fairness, trustworthiness of the underlying data, and, potentially even legal implications. Taking as a case study the transformer models hosted by Hugging Face, a popular hub for pre-trained ML models, this paper empirically investigates the transparency of pre-trained transformer models. We look at the extent to which model descriptions (i) specify the datasets being used for their pre-training, (ii) discuss their possible training bias, (iii) declare their license, and whether projects using such models take these licenses into account. Results indicate that pre-trained models still have a limited exposure of their training datasets, possible biases, and adopted licenses. Also, we found several cases of possible licensing violations by client projects. Our findings motivate further research to improve the transparency of ML models, which may result in the definition, generation, and adoption of Artificial Intelligence Bills of Materials.

CCS CONCEPTS

• Software and its engineering → Software libraries and repositories.

KEYWORDS

ML-Intensive Systems, Pre-trained models, Transparency, Bias, and Fairness, Deep Learning, Empirical Study

1 INTRODUCTION

Pre-trained Machine Learning (ML) models represent key reusable assets when developing ML-intensive software systems. A pre-trained model is a ML model that has been trained for one or more (relatively generic) tasks, or simply to “learn” a given language. Subsequently, developers/data scientists have the option to fine-tune these models, customizing them for specific tasks. Such a specialization requires fewer resources—in terms of training data, computational power, and time—than training a model from scratch. To make a metaphor, it would be like teaching math to a child who already speaks English rather than teaching it to a child who does not speak at all.

Inspired by the reuse of software libraries that happens through package managers (e.g., Maven Central, npm, or PyPi), or of containerization images, via specific forges like Docker Hub, Hugging Face (HF) [25] has created a hub for pre-trained models to democratize Artificial Intelligence (AI). HF provides (i) reusable pre-trained models for different purposes (e.g., natural language, image, or audio processing), (ii) datasets for model training, and (iii) a high-level API for conveniently using the models through back-ends such as Keras/Tensorflow or PyTorch. Forges similar to HF are Model Zoo [31], Tensorflow Hub [19], and PyTorch Hub [47].

Recently, researchers in software engineering have been examining the hurdles associated with employing HF models. Some research has studied in general how models are documented, and provided templates [35] and approaches/tools to guide models’ documentation [3, 12, 48]. Other work focuses on security concerns [26] or on the environmental impact of the models’ training [7]. That being said, there are other crucial aspects that a developer aiming to reuse a pre-trained ML model must take into account. These aspects include (i) the extent to which models document and provide access to the datasets used for training; (ii) the potential for biases in the models resulting from their pre-training process; and (iii) the presence of compatibility issues between the model’s license and software licenses.

This paper presents the findings of an empirical investigation that focuses on the analysis of pre-trained transformer models hosted on the HF Hub. The study aims to achieve the following goals: (i) determine the extent to which and how the models document
and provide access to their training data; (ii) assess the extent to which models are documented with potential biases they could suffer from; and (iii) identify licensing incompatibilities that may arise when GitHub projects reuse HF models. Overall, the study has been conducted by mining 159,132 models hosted on HF, and 44,823 open-source projects hosted on GitHub reusing such models.

Results of the study indicate that, so far, there is still limited transparency by the models in documenting the datasets on which they have been trained, and the biases to which they are subject. Concerning licenses, models tend to adopt permissive ones, and even ML-specific licenses, encompassing a “responsible” use of such models. Nevertheless, we found that software projects using models still create several potential cases of licensing incompatibilities.

This study has multiple implications. First of all, it provides empirical evidence about the extent to which training datasets, bias issues, and licensing are documented by pre-trained models. This raises the need for better documenting these models, as results indicate, very often, a scarce level of documentation. Second, the results indicate how the reuse of pre-trained models could possibly generate bias if no proper countermeasures are taken. Third, the analysis we performed on the type of documentation available with pre-trained models opens the road for better defining, and possibly automatically generating, bills of material for AI-based systems (ABOMs).

Overall, the paper’s contributions can be summarized as follows:

1. We analyze whether and how pre-trained transformer models document the datasets they have been trained on.
2. We analyze and discuss whether models document their bias and the type of bias they declare.
3. We provide information about the licensing of HF models, as well as potential licensing incompatibilities of projects using such models.

The paper is organized as follows. Section 2 describes the study design, including the data extraction and analysis process. Results are presented and discussed in Section 3, while their implications are discussed in Section 4, and their threats in Section 5. After Section 6 discusses related literature, Section 7 concludes the paper, and outlines directions for future work.

2. STUDY DESIGN

The goal of the study is to analyze pre-trained machine learning models hosted by HF, to understand the extent to which and how they document training datasets, potential biases derived from their training, and their licenses. The quality focus is the models’ transparency, the lack of which could cause unexpected/unfair behavior (e.g., due to bias), security issues, or legal problems. The perspective is of researchers, that would like to define approaches, including ABOM generators, aimed at enhancing the transparency of ML-based systems. The context consists of 159,132 pre-trained transformer models hosted on HF, and of 17,365 open-source projects hosted on GitHub and using (a subset of) such models.

The study aims to address the following research questions:

- **RQ1**: To what extent do models declare the datasets used for their training? We look at the pre-training transparency from its primary element, i.e., the dataset(s) used to train the models. The lack of dataset declaration, or, a vaguely-specified, inaccessible dataset reduces the transparency of a pre-trained ML model.
- **RQ2**: How do pre-trained models discuss fairness limitations? We conduct a qualitative assessment to determine the level of bias declaration in pre-trained models and, if present, the nature of the expected bias. This information would be valuable to model users as it enables them to safeguard their software against biases, e.g., through appropriate model fine-tuning or employing other approaches for bias prevention/mitigation.
- **RQ3**: To what extent do models declare their licenses, and to what extent do such licenses lead to potential incompatibilities among client projects? We analyze the models’ documentation from a legal perspective by looking at (i) the models’ declared licenses, and (ii) the relationship between the models’ licenses and the client projects’ ones, to identify possible incompatibilities.

### Table 1: Number of HF models downloaded by task.

<table>
<thead>
<tr>
<th>Tasks</th>
<th>#Models</th>
<th>Min</th>
<th>1Q</th>
<th>Median</th>
<th>Mean</th>
<th>3Q</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>NLP</td>
<td>62,416</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>4,292</td>
<td>11</td>
<td>47,032,389</td>
</tr>
<tr>
<td>RL</td>
<td>15,431</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>18</td>
<td>2</td>
<td>66,658</td>
</tr>
<tr>
<td>Audio</td>
<td>7,918</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>1,693</td>
<td>8</td>
<td>10,402,498</td>
</tr>
<tr>
<td>Multimodal</td>
<td>6,212</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>4,466</td>
<td>17</td>
<td>3,237,345</td>
</tr>
<tr>
<td>CV</td>
<td>3,867</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>4,728</td>
<td>53</td>
<td>10,487,900</td>
</tr>
<tr>
<td>Tabular</td>
<td>175</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2,571</td>
<td>1</td>
<td>205</td>
</tr>
<tr>
<td>Other</td>
<td>4</td>
<td>7</td>
<td>11</td>
<td>21</td>
<td>119</td>
<td>129</td>
<td>428</td>
</tr>
<tr>
<td>Not Available</td>
<td>36,109</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>562</td>
<td>0</td>
<td>7,707,765</td>
</tr>
<tr>
<td><strong>Summary</strong></td>
<td>159,132</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2,281</td>
<td>5</td>
<td>47,032,389</td>
</tr>
</tbody>
</table>

2.1 Context Selection and Data Extraction

Our study encompasses two main components: firstly, a dataset of pre-trained transformer models available on the HF hub, and secondly, a collection of open-source projects hosted on GitHub. The data for this study was collected in February 2023. We have chosen to study the transformer models because of their wide number of applications in various areas, including among others NLP, multimedia processing, and reinforcement learning.

To download the HF data, we leverage the HF Endpoints API [24]. Specifically, we perform a query to retrieve a JSON entry for all models hosted on HF. Each JSON entry contains information including the last modification date, model tags, numbers of downloads and likes, etc. In total, we collect data for 159,132 HF models.

Table 1 shows the distribution of downloaded HF models organized by task, and specifically Natural Language Processing (NLP), Reinforcement Learning (RL), Audio, Multimodal (e.g., text-to-image generation), Computer Vision (CV), Tabular (tabular classification and tabular regression), and Other (including time series forecasting). The table also reports descriptive statistics (minimum, 1st quartile, median, mean, 3rd quartile, and maximum) of the models’ downloads for the different task categories. As it can be noticed, the distribution is very skewed, as the majority of the models have a very small number of downloads.

To compute the distribution of Table 1, we start from the model modality, i.e., the value of pipeline_tag feature in the JSON file, and we assign the related task as shown above. For example, if the
Table 2: GitHub projects using the HF transformers library

<table>
<thead>
<tr>
<th>Task</th>
<th>#Dependent Prjs</th>
<th>#from_pretrained Usage</th>
<th>#Pjs re-using HF models</th>
</tr>
</thead>
<tbody>
<tr>
<td>NLP</td>
<td>2001</td>
<td>475</td>
<td>17,365</td>
</tr>
<tr>
<td>Not Available</td>
<td>475</td>
<td>286</td>
<td></td>
</tr>
<tr>
<td>Multimodal</td>
<td>171</td>
<td>155</td>
<td></td>
</tr>
<tr>
<td>CV</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Audio</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>RL</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Other</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Tabular</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

model under analysis has the pipeline_tag equal to text classification, then this model contributes to the count of models addressing the Natural Language Processing task. When a model does not specify the pipeline tag, we place it in a separate category called Not Available. As Table 1, this happens for ~ 23% of the models.

The second step of our data extraction aims at identifying GitHub projects1 gathered by following the process described above using pre-trained models from HF. To achieve this goal, we leverage the Dependencies feature available in GitHub, which, given a repository, lists all other repositories that depend on it. For Python projects, GitHub computes dependents by analyzing the projects’ package dependencies (e.g., requirements.txt). Specifically, we gather the list of dependents from the GitHub huggingface/transformers project. To this aim, we leverage Beautiful Soup and requests Python packages. More in detail, we use the former to extract the projects list from the HTML pages and the latter to perform GET requests to navigate the result pages. This is necessary as the GitHub API currently does not provide a query to collect dependents’ information. In total, we collect 44,823 projects depending on the HF transformers library. Starting from this list of projects, we clone each project, and then, we search, in all Python files, for all occurrences of the .from_pretrained function invocation, i.e., the function that imports a pre-trained model. Among 44,823 dependent projects, 33,249 use the .from_pretrained function at least once. Then, we filter such occurrences and collect only the projects for which the .from_pretrained refers to one of the HF models gathered in the previous step. At the end of this process, we collect 17,365 projects which re-use a pre-trained model belonging to the HF transformers library. As it will be discussed in Section 5, we are aware that this analysis may be imprecise and above all lead to false negatives, as (i) the string passed to the .from_pretrained function could be dynamically created; and (ii) in principle, yet less likely for this particular circumstance, the .from_pretrained could be assigned to a variable and passed to a function.

As reported in Table 2, we found 44,823 projects hosted on GitHub depending on the HF transformers library. Of these, we could match 17,365 onto pre-trained models hosted on the HF Hub. Those that were not matches could be (i) repositories having a dependency on transformers yet never using it; (ii) cases where the transformers library was used, e.g., to create a tool on top of HF, yet no model was explicitly imported; and (iii) more importantly, cases where the model name was passed as variable to .from_pretrained and, due to our simplified analysis (or just because the model name was never present in the source code), we could not match any model.

Several projects use more than one model. The number of reused models per project has a median equal to 1, with a first and third quartile equal to 1 and 2 respectively, and a large number of outliers (2449 out of 17,365 projects).

The maximum number of models used by a project is 463, and this happens for the aarnphm/transformer project, which is a fork of the transformers library used to contribute to it.

Furthermore, we found that Spearman’s rank correlation between models’ usage by GitHub projects (i.e., the number of GH projects reusing a specific model) and their number of downloads (i.e., the number of downloads per model on HF) is $\rho = 0.40$, indicating a moderate correlation. This means that, while the number of downloads is not a perfect indicator of project usage, it correlates enough with it to be used as a good indicator of models’ popularity in our analyses.

Figure 1 reports the distribution of model usages across different tasks. Consistently to what was mentioned before, the largest number of model usages is related to NLP, with 2,001 projects reusing such models. 475 projects use models with no specified task. Multimodal, CV, and Audio follow, with 286, 171, and 155 projects respectively. We found a few cases of usage for RL and other categories and no reuses for Tabular models. Note that 42 models are missing from this list as their metadata could not be retrieved.

Finally, for each project linked to the HF models, we use the Perceval [15] tool to retrieve its metadata from these repositories. Other than for capturing general information about the repositories, the metadata will be used to identify the projects’ declared license(s). To assess the reliability of the licensing in the metadata, we extracted a (randomly stratified over license types) statistically significant sample of projects, for a confidence level of 95% and a margin of error of ±5%. Given the sample size of 383 projects, we approximated by excess decimal numbers and analyzed all samples when the number of projects per license was 5 or less. In total, we obtained a sample of 406 projects. One author manually analyzed the LICENSE files and compared them with the licensing in the metadata. The task was fairly simple and not subjective, so to not require multiple annotators. Only in one case out of 406 (0.24%) we found an inconsistency. Therefore, we can assume that the licensing information in the metadata is reliable enough.

---

1 Although the correct GitHub term is "repository", in the following we will refer to them as "project".
2.2 Analysis Methodology

To address RQ1, we perform two types of analyses. The first one, performed on all 159,132 models, aims at reporting statistics about the number of models that include specific tags related to datasets (those starting with “dataset:”, which appear on the model page as a small cylinder). However, this only gives a partial picture of how models have been trained because a model’s owner could specify the datasets somewhere else in the model card description, without following a specific pattern. For this reason, we decided to answer this question by manually analyzing a statistically significant sample of the model cards.

Given the large number of collected model cards (159,132), we extracted a stratified sample, using the tasks in Table 1 as strata and ranking models in decreasing order of download. In other words, we do not perform a random sampling over the strata, but, rather, an “intensity” sampling by the number of downloads. We considered a usage.

At the same time, after a first partial scrutiny of HF model cards, we found that the types of bias are described at a finer-grained level than what foreseen in the paper by Mehrabi et al. [34], as it is pretty comprehensive in that regard. The proposed taxonomy foresees three high-level categories, namely: (i) Data to Algorithm (e.g., concerning how data is measured and characterized in terms of variables), (ii) Algorithm to User (related to the algorithmic outcome of a model when a user interacts with it), and (iii) User to Data (concerning model’s data being user-generated or user-related). Such three categories are then further detailed into sub-categories.

Table 3 reports, similarly to Table 1, descriptive statistics for the # of Downloads

<table>
<thead>
<tr>
<th>Tasks</th>
<th>#Models</th>
<th>Min</th>
<th>IQ</th>
<th>Median</th>
<th>Mean</th>
<th>SQR</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>NLP</td>
<td>249</td>
<td>66</td>
<td>743</td>
<td>111</td>
<td>393</td>
<td>199</td>
<td>974</td>
</tr>
<tr>
<td>RL</td>
<td>62</td>
<td>55</td>
<td>11</td>
<td>55</td>
<td>56</td>
<td>2</td>
<td>194</td>
</tr>
<tr>
<td>Audio</td>
<td>32</td>
<td>16</td>
<td>145</td>
<td>21</td>
<td>704</td>
<td>36</td>
<td>999</td>
</tr>
<tr>
<td>Multimodal</td>
<td>25</td>
<td>216</td>
<td>421</td>
<td>281</td>
<td>238</td>
<td>520</td>
<td>688</td>
</tr>
<tr>
<td>CV</td>
<td>16</td>
<td>87</td>
<td>907</td>
<td>115</td>
<td>650</td>
<td>171</td>
<td>889</td>
</tr>
<tr>
<td>Tabular</td>
<td>1</td>
<td>205</td>
<td>5</td>
<td>205</td>
<td>205</td>
<td>205</td>
<td>205</td>
</tr>
<tr>
<td>Other</td>
<td>4</td>
<td>7</td>
<td>11</td>
<td>21</td>
<td>119</td>
<td>128</td>
<td>428</td>
</tr>
<tr>
<td>Summary</td>
<td>389</td>
<td>7</td>
<td>68</td>
<td>397</td>
<td>148</td>
<td>329</td>
<td>766</td>
</tr>
</tbody>
</table>

indicating enough reliability in the classification. Then, the manual annotators jointly resolved the conflicting cases.

We report, for the analyzed sample, the number and percentage of models (overall and along the different categories) for which there is some dataset information, and, for the latter, the number and percentage of models that (a) use datasets hosted on HF, (b) provide a link to an external dataset, (c) provide generic information about the training dataset.

To address RQ2, first of all, we looked for pre-existing taxonomies of ML model biases. Based on the existing literature, we rely on a bias taxonomy from a systematic literature review on the topic conducted by Mehrabi et al. [34], as it is pretty comprehensive in that regard. The proposed taxonomy foresees three high-level categories, namely: (i) Data to Algorithm (e.g., concerning how data is measured and characterized in terms of variables), (ii) Algorithm to User (related to the algorithmic outcome of a model when a user interacts with it), and (iii) User to Data (concerning model’s data being user-generated or user-related). Such three categories are then further detailed into sub-categories.

However, the category Data to Algorithm does not apply, at least directly, to our scope (if anything, it could be used in a further study on HF datasets). As for the Algorithm to User category, while model cards discuss “intended uses” they do not discuss interaction bias. Instead, we included in our taxonomy Popularity Bias, which concerns (generic cases of) items present in the dataset with a high frequency.

At the same time, after a first partial scrutiny of HF model cards, we found that the types of bias are described at a finer-grained level than what foreseen in the paper by Mehrabi et al. for the User to Data category. Therefore, we decided to manually code the bias declared in the model cards, and then to map those levels of biases onto the 7 sub-categories of User to Data by Mehrabi et al.: Historical (related to peculiar data distribution in certain periods of time), Population (intrinsic characteristics of a sub-population), Self-selection (e.g., subjects participate in a study based on their interest), Social (related to social characteristics), Behavioral (different user behavior in different contexts), Temporal (differences in the population over periods of time), and Content Production (e.g., related to language differences).

Similarly to RQ1, to categorize models’ bias it is not possible to perform a reliable, automated classification of all model cards. Therefore we opted, again, for a manual analysis, considering the same sample of RQ1. For each model, each annotator, first of all, indicated on the online spreadsheet whether or not the model card was describing bias-related information. Then, each annotator indicated on the spreadsheet the type of bias described. Since we had no predefined categories for that, we adopted an open card sorting [45] procedure. Each annotator used a column of the online spreadsheet to select the bias category from a different sheet that was jointly populated by the three annotators (i.e., a new category was added when the available categories did not fit with what was reported in the model card). Multiple entries were added to the spreadsheet when a model card described multiple types of bias. To agree on the annotation criteria and create a first set of categories, the three annotators jointly annotated 20 model cards, and then continued independently.
After the annotation was completed, the annotators met to resolve the conflicting cases. Also, they jointly reviewed the list of categories and mapped them onto the categories of Mehrabi et al.

Also for RQ3, we assessed manual classifications in two ways, i.e., by computing (1) Cohen’s $k$ on the Boolean classification on the presence of bias, and (2) Krippendorff’s $\alpha$ [32] for the categories, because each annotator added multiple labels, and there could be cases where one annotator added a label and the other did not, resulting in a (categ, N/A) pair in the inter-rater computation table. In this circumstance, Krippendorff’s $\alpha$ is suitable as it handles incomplete ratings. We achieved a Cohen’s $k$ of 0.91, and a Krippendorff’s $\alpha$ of 0.93, both indicators of a very strong agreement.

For the overall sample and for each model task category, we report the number and type of models declaring some bias. Then, we describe and discuss the bias categories found during the analysis. In doing so, we provide examples of biases for the different categories, also explaining how the models detail such bias by providing input examples and expected outputs exhibiting the bias.

To address RQs, we first report the distribution of licenses among the HF models also highlighting how many models do not declare any license. To perform this analysis we check the license tag contained in the tag list of the JSON file of the model. We report results at two levels of abstraction, i.e., by first distinguishing between different categories of licenses in terms of permissiveness (the mapping between these levels and specific licenses is in our replication package), i.e., network restrictive (e.g., AGPL), restrictive (e.g., GPL), weakly restrictive (e.g., LGPL or MPL), and permissive (e.g., Apache, MIT, or BSD), and then in terms of specific licenses.

Then, still using the license categorization, we report the relationships between models’ licenses and client projects’ licenses, highlighting relations that would lead to incompatibilities. For those, we further detail and discuss the specific licenses used on both sides.

## 3 STUDY RESULTS

This section discusses results addressing the three research questions formulated in Section 2. Note: To access each model card mentioned as OWNER/REPO_NAME, one can use the following link https://huggingFace.co/OWNER/REPO_NAME (without OWNER if the latter is not specified). Similarly, GitHub projects (mentioned in RQs) can be accessed as https://github.com/OWNER/REPO_NAME.

### Table 4: Documented datasets from the models’ sample

<table>
<thead>
<tr>
<th>Task</th>
<th>Labeled</th>
<th>HF</th>
<th>External</th>
<th>No Link</th>
<th>Any</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>NLP</td>
<td>249</td>
<td>100</td>
<td>51</td>
<td>55</td>
<td>174</td>
<td>69.87</td>
</tr>
<tr>
<td>RL</td>
<td>62</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Audio</td>
<td>32</td>
<td>15</td>
<td>8</td>
<td>5</td>
<td>21</td>
<td>65.62</td>
</tr>
<tr>
<td>Multimodal</td>
<td>25</td>
<td>3</td>
<td>12</td>
<td>2</td>
<td>14</td>
<td>56</td>
</tr>
<tr>
<td>CV</td>
<td>16</td>
<td>4</td>
<td>4</td>
<td>7</td>
<td>12</td>
<td>75</td>
</tr>
<tr>
<td>Other</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>4</td>
<td>100</td>
</tr>
<tr>
<td>Tabular</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>389</td>
<td>125</td>
<td>75</td>
<td>72</td>
<td>226</td>
<td>58.09</td>
</tr>
</tbody>
</table>

### 3.1 RQ1: Training Datasets

Among the 159,132 analyzed models, only 22,572 (14.08%) specify the dataset(s) used during the training phase using the dataset tag. Of these, 15,437 only rely on datasets hosted on HF, 6,591 provide a link to external dataset(s), and 544 provide both types of datasets. The tag distribution is lowly-skewed, with first, second, and third quartile=1, and a maximum of 135 for sileod/deberta-v3-base-tasksource-nli.

As models could specify datasets somewhere else in the text of the model card, we perform a manual analysis on 389 model cards. Results are reported in Table 4. As it can be noticed, being these the top-most downloaded models, a fairly high number 226 (58.09%) of them provide some dataset documentation. Of these 226, only 143 (63%) belong to the 22,572 above, i.e., use the dataset tag to document the dataset.

125 models in the sample only leverage internally-hosted datasets (that are easier to retrieve and use by the users through the HF datasets library), while 75 provide external links, and 72 reference dataset names without providing links. Note that the sum of columns HF, External, and No Link exceeds the total # of models with declared datasets, as some models may declare multiple datasets in different ways.

While, for most task types, ~60% of the models declare datasets, this is never the case for RL models. The reason can be the learning strategy of such models: An agent interacts with the environment by taking actions, and in response, the environment provides a reward signal indicating how favorable the agent’s actions are. Such environment data seems not to be released. However, for 56 out of 62 RL models in our sample, there is a link to a GitHub repository (huggingface/ml-agents) containing environments on which RL models can be trained.

Some models declare several training datasets from different sources. For example, the sentence-transformers/all-mpnet-base-v2 model is a sentence transformer used to convert sentences and paragraphs. The training combines multiple datasets through concatenation, resulting in over 1 billion sentence pairs. In total, it incorporates 26 datasets from HF and 23 datasets from external sources for which a link is provided.

Other models mention datasets without sharing them. For example, the gpt2-medium model card mentions an external dataset named WebText, which comprises 40 GB of textual data, yet it has not been made publicly available.

Along a similar line, finetautomata/bertweet-base-sentiment-analysis is a BERT model fine-tuned on Tweets for sentiment analysis. As Tweets datasets cannot be shared any longer, the authors mention “Please be aware that models are trained with third-party datasets and are subject to their respective licenses.”

### Answer to RQ1

Only 14% of the analyzed models declare datasets as tags in the model cards. However, in a sample of 389 top-downloaded models, 61% declare datasets in some way.

### 3.2 RQ2: Bias Description

Table 5 illustrates the total number of models assessed for each task, along with the number of models with no detected bias and the number of models exhibiting at least one bias. In total, we manually inspected 389 models, of which only 72 (18%) were found to expose
bias, and 43 exhibited more than one type of bias. 55 of these models are related to NLP, 7 to Audio, 7 to Multimodal tasks, and 3 to CV.

Models belonging to Other, RL, and Tabular do not declare any type of bias. In the Other and Tabular categories, the number of models is still limited, which makes it challenging to speculate on their bias declaration. However, after a careful evaluation of 62 models in the RL category, no evidence of bias was found in any of them. The reasons behind that are related to how such models are trained, as also discussed in Section 3.1, yet, as existing literature discusses, the underlying training dataset may not be representative of the phenomenon captured by the RL model [44], or the algorithms’ approximation function introduces a bias [20].

The open card sorting on the 389 models of our sample resulted in a categorization depicted in Figure 2, and featuring three abstraction levels. The first level features five categories from the Mehrabi et al. taxonomy [34] and, specifically: (i) Popularity bias, a sub-category of their Algorithm to User category, and (ii) four sub-categories of their User to Data category. For the remaining sub-categories, we found no mapping for Self-Selection (as said, it mainly applies to survey studies), Behavioral (i.e., no user behavior captured by the model resulted in a documented bias), and Temporal (for the latter we found descriptions more appropriate to the Historical Bias sub-category).

Overall, the categorization features 17 leaf categories, including Popularity bias that does not have further subdivisions. Also, some categories (Content Production, Social, and Population) have an intermediate level of grouping.

In the following, we detail each level of categorization starting from the first level of Mehrabi et al. going through the intermediate levels down to the leaves.

**Population Bias** (35 models) is further specialized into two sub-categories: **Personal** and **Location**. **Personal** groups 3 types: Race, Gender, and Age biases. These refer to the prejudice or discrimination that occurs based on a person’s race, gender, or age. An interesting example of these biases is found in the model card of openai/whisper-medium (Audio) where it is stated that the model “... exhibit[s] ... higher word error rate across speakers of different genders, races, ages, or other demographic criteria.” This is because the model is trained “in a weakly supervised manner using large-scale noisy data.” We found 13 co-occurrences of gender and race bias, e.g., related to models predicting black women work as in distilbert-base-uncased.

The **Location** sub-category includes 2 types of biases: **Demographic Criteria** and Geographic. It refers to models that could
be skewed by the geographic distribution of the data used for training. Demographic Criteria bias affects models not representing the population diversity or could contain disproportions in demographic groups. Geographic bias emerges when some geographic areas are missing/limited in the training set. An example is the model distilbert-base-uncased-finetuned-sst-2-english. It states that for sentences like “This film was filmed in COUNTRY”, “...[the] binary classification model will give radically different probabilities for the positive label depending on the country (0.89 if the country is France, but 0.08 if the country is Afghanistan).”

Social Bias (27 models) is specialized into two intermediate levels: Society and Status. Society, in turns groups Culture and Religious biases. These biases can lead to discriminatory predictions stemming from the lack of sufficient representation of cultural or religious attributes. For example, the stabilityai/stable-diffusion-2 (Multimodal) model card mentions “While the capabilities of image generation models are impressive, they can also reinforce or exacerbate social biases.” Unsurprisingly, Culture bias occurs in 50% of the models exposing Demographic Criteria bias, due to the clear interconnection between the two elements.

The Status sub-category encompasses Occupational and Social bias, and Class bias. They occur when most of the existing occupations, social groups, or socioeconomic classes are not fully included in the training set or the latter contains harmful stereotypes. An example of Occupational and Social bias can be found in distilroberta-base model: “Predictions generated by the model may include disturbing and harmful stereotypes across protected classes; identity characteristics; and sensitive, social, and occupational groups.”, and they show a concrete example of such an unfair prediction. They ask the model to complete the phrase “The man/woman worked as a <mask>...”. and the model returns, among options, also some current prejudices, i.e., the man worked as mechanic/courier and the woman worked as nurse/maid.

Content Production Bias (25 models) is specialized into Speech and Inappropriate Content. Speech category includes 2 types of biases: Speech Disorders and Foreign Accent. These are related to the lack of specific speech characteristics, such as their accent, dialect, or speech disorders, thus models do not include information about accent variations, specific speech patterns, or any speech disorder. This category of bias mainly affects models related to the Audio task. For example, TalTechNLP/voxlingua107-epaca-tdnn faces speech bias. Its model card states that “[b]ased on subjective experiments, it doesn’t work well on speech with a foreign accent and, probably also, on persons with speech disorders.”

Inappropriate Content relates to Sexual Content and Offensive Content. For example, the google/flan-t5-xl model states that it “… can potentially be used for language generation in a harmful way” since it “…is fine-tuned on a large corpus of text data that was not filtered for explicit content or assessed for existing biases.”

The Historical Bias category (21 models) has two sub-categories: Time and Domain Dependent and Historical and Current Stereotypes. They refer to time-sensitive elements and historical patterns in the data used for training. An example of Time and Domain Dependent was found in in dslim/bert-base-NER, that is used for Named Entity Recognition (NER) “[the] model is limited by its training dataset of entity-annotated news articles from a specific span of time”.

### Table 6: The top 20 licenses used by HF models

<table>
<thead>
<tr>
<th>License</th>
<th>Permissiveness</th>
<th>#Models using it</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache-2.0</td>
<td>Permissive</td>
<td>24,123</td>
</tr>
<tr>
<td>MIT</td>
<td>Permissive</td>
<td>10,151</td>
</tr>
<tr>
<td>CreativeML-openRAIL-m</td>
<td>Permissive</td>
<td>3,805</td>
</tr>
<tr>
<td>OpenRAIL</td>
<td>Permissive</td>
<td>2,529</td>
</tr>
<tr>
<td>CC-BY-4.0</td>
<td>Permissive</td>
<td>1,776</td>
</tr>
<tr>
<td>AFL-3.0</td>
<td>Permissive</td>
<td>1,559</td>
</tr>
<tr>
<td>Other</td>
<td>-</td>
<td>1,340</td>
</tr>
<tr>
<td>None/Unknown</td>
<td>-</td>
<td>722</td>
</tr>
<tr>
<td>cc-by-nc-sa-4.0</td>
<td>Restrictive</td>
<td>556</td>
</tr>
<tr>
<td>cc-by-sa-4.0</td>
<td>Restrictive</td>
<td>556</td>
</tr>
<tr>
<td>cc-by-nc-4.0</td>
<td>Restrictive</td>
<td>490</td>
</tr>
<tr>
<td>cc0-1.0</td>
<td>Permissive</td>
<td>482</td>
</tr>
<tr>
<td>GPL-3.0</td>
<td>Restrictive</td>
<td>431</td>
</tr>
<tr>
<td>Artistic-2.0</td>
<td>Permissive</td>
<td>347</td>
</tr>
<tr>
<td>Bigscience-bloom-rail-1.0</td>
<td>Permissive</td>
<td>208</td>
</tr>
<tr>
<td>BSD-3-clause</td>
<td>Permissive</td>
<td>197</td>
</tr>
<tr>
<td>Bigscience-openrail-m</td>
<td>Permissive</td>
<td>163</td>
</tr>
<tr>
<td>wtfpl</td>
<td>Permissive</td>
<td>142</td>
</tr>
<tr>
<td>AGPL-3.0</td>
<td>Network Restrictive</td>
<td>104</td>
</tr>
<tr>
<td>Unlicense</td>
<td>Permissive</td>
<td>94</td>
</tr>
</tbody>
</table>

### Figure 3: The distribution of model licenses by tasks.

Finally, Popularity bias includes only four models that explicitly describe generic bias due to the prevalence of some instances in the dataset. In one model facebook/opt-125m, only such a bias is declared “Like other large language models for which the diversity (or lack thereof) of training data induces downstream impact on the quality of our model, OPT-175B has limitations in terms of bias and safety.”

### Answer to RQ2. Of 389 manually analyzed models, only 72 (18%) describe their possible bias, categorized along four sub-categories of User to Data from the Mehrabi et al. [34] taxonomy. These are in turn, specialized into 16 low-level categories.

### 3.3 RQ2: Licenses

Among 159,132 analyzed models, only 50,506 (32%) specify a license. In total, the models use 61 different licenses. Models are categorized in terms of permissiveness of their licenses as follows: 45,991 (91%) use a Permissive license, 72 (0.1%) a Weakly Restrictive, 2,276 (4.5%) a
Table 7: Models’ licenses (rows) vs. client projects’ (columns) licenses.

<table>
<thead>
<tr>
<th>Models</th>
<th>Network Restrictive</th>
<th>Not Available</th>
<th>Other</th>
<th>Permissive</th>
<th>Restrictive</th>
<th>Weak Restrictive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network Restrictive</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Not Available</td>
<td>42</td>
<td>16,576</td>
<td>615</td>
<td>23,700</td>
<td>257</td>
<td>6</td>
</tr>
<tr>
<td>Other</td>
<td>1</td>
<td>311</td>
<td>4</td>
<td>471</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>Permissive</td>
<td>164</td>
<td>33,818</td>
<td>1,153</td>
<td>40,288</td>
<td>732</td>
<td>13</td>
</tr>
<tr>
<td>Restrictive</td>
<td>0</td>
<td>570</td>
<td>18</td>
<td>707</td>
<td>15</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 8: Cases of licensing incompatibilities

<table>
<thead>
<tr>
<th>Models</th>
<th>Projects</th>
</tr>
</thead>
<tbody>
<tr>
<td>cc-by-sa-4.0</td>
<td>Apache-2.0</td>
</tr>
<tr>
<td>GPL-3.0</td>
<td>350</td>
</tr>
</tbody>
</table>

Answer to RQ3: 31% of the HF models declare a license, most of which (91%) a permissive one, and also employing ML-specific licenses, constraining a “responsible” usage of models. Over 4.7% of the models use a (network) restrictive license, and we found 707 GitHub projects released under a permissive license using such models.
4 IMPLICATIONS

The results of our study lead towards implications for Data Scientists, Software Developers, Software Engineering Researchers, and Educators.

Data Scientists, thanks also to initiatives like HF, are gaining awareness about model sharing and transparency of such models. However, there is still a long road towards having full transparency ensured by a large majority of models. To this extent, tools for automatically assessing the content and quality of model cards (similar to tools checking the quality of bug reports [9, 55]) would be desirable. Such tools may nicely complement tools such as DocML [3], helping developers in crafting new model cards, as well as tools such as AIMMX [48] that automatically extract metadata from ML models. Data Scientists should also carefully test the used models against bias, and, whenever possible, use techniques like the ones existing in literature to mitigate such bias. As it is further discussed in Section 6.3, this includes techniques based on re-ranking [37], rebalancing distorted predictions [40], random sampling [13], knowledge graph manipulation [33], or neural-network partitioning and pruning [5].

Developers need better support in selecting models based on their transparency (e.g., through the use of “transparency badges” on model hubs). Additionally, they should increase their awareness about the significance of transparency in models. Last, but not least, while the importance of licensing is gaining maturity for conventional software, it is still not the case for licensing, given the number of potential incompatibilities we found. This also triggers the need for licensing check tools accounting for ML models’ licenses.

Software Engineering Researchers can exploit transparency info for several purposes. (i) understanding how models are reused by projects and whether countermeasures are taken, (ii) leveraging the model card contents to test ML software against bias and to mitigate it, and (iii) enhancing the practice and support for ALBOM, e.g., by encompassing bias info, but, also, developing tools for automated ALBOM generation.

Educators should not only teach about ML bias (as it is already being done), but, also, challenges related to model’s transparency, explaining how to properly document newly created models. Also, with the growth of ML-intensive systems, it is becoming even more important to instruct future developers about legal implications arising when integrating existing, pre-trained ML models into their software systems.

5 THREATS TO VALIDITY

Threats to construct validity concern the relationship between theory and observation. This threat may be due to imprecision in our analyses. As already explained in Section 2.1, the identification of links between projects and models can suffer from imprecision due to the over-simplified analysis of the from pre-trained invocations. This threat does not affect RQ1 and RQ2, yet it may affect RQ3, where we identify licensing compatibility between projects and models. Truly, the performed analysis allowed us to gather a relatively large set of 17,365 projects, which could be sufficiently representative, yet we cannot exclude it can suffer from a bias.

As for RQ1 and RQ2, we performed a manual analysis of a sample of models. To ensure the reliability of our classifications and coding, we computed inter-rater reliability measures. For RQ1, the manual analysis of the sample is complemented by an exhaustive analysis on all 159,132, yet limited to datasets declared through tags.

Finally, for RQ3, besides the threat of the project-model tracing discussed above, we leverage the licenses declared in the model and GitHub projects’ metadata, and we cannot exclude that these are largely incomplete or even incorrect. As explained in Section 2.1, we mitigated this threat through the manual analysis on a sample of projects. As for the models, this is the way licenses should be declared when releasing them on HF [23]. As for the projects, the actual license is contained in the projects’ LICENSE* file, which might be wrongly classified by GitHub. Nevertheless, this file is usually created starting from the GitHub-provided templates, which mitigates the threat.

Threats to internal validity concern factors, internal to our study, that may influence our findings. Our study does not make causation claims, if not using downloads as a proxy of models’ usage when sampling to address RQ1 and RQ2. The sample for RQ1 and RQ2 is intentionally not random, as we wanted to focus on models largely downloaded, i.e., possibly used.

Threats to conclusion validity concern the relationship between observation and outcome. The analysis of RQ1 and RQ2 may not be exhaustive, as it is only based on a sample of the models.

Threats to external validity concern the generalizability of our findings. Although HF is a very popular hub for pre-trained transformer models, it only represents a partial view of the reality, and other hubs such as Model Zoo [31], Tensorflow Hub [19], or PyTorch Hub [47]. Moreover, we focus on transformers, but other studies may investigate other types of pre-trained models. Last, but not least, we look at the models’ usage by GitHub projects, which, again, only represent a partial view of the overall models’ usage.

6 RELATED WORK

Considering the scope of our work, we focus on three distinct aspects: (i) works mining data from HF; (ii) documentation of ML models, (iii) bias and fairness in software engineering, and (iv) licensing analysis in open-source.

6.1 Empirical Studies on Hugging Face

Jiang et al. [26] interviewed 12 experts from HF to gain insights into the current practices and challenges associated with reusing pre-trained models. They identified the key attributes of reused models, focusing on provenance, reproducibility, and portability. The study revealed that the primary challenges in this domain revolve around inconsistencies between actual and reported performances, as well as concerns regarding model risks. Our paper is complementary to Jiang et al. [26], first in terms of purpose (studying models’ transparency and level of documentation), and second in terms of methodology (mining study instead of interviews).

Castano et al. [7] conducted a study to analyze the carbon footprint of 1,417 models hosted on HF. They evaluated the carbon dioxide emission during the training phase and found correlations between emissions and factors such as model size, dataset size, and application domains. The results of their study serve to foster a “sustainable” ML-based development. Also in this case, our study tackles complementary challenges, focusing on models’ datasets,
was first advocated by Mitchell et al. They found that a comprehensive classification of software biases, moral norms, and values. In response, they propose a novel approach called "MoralDirection" to retrieve human-like biases regarding what is considered right and wrong when employing pre-trained language models for text generation.

Fairness emerges as a crucial concept in software engineering, aiming to achieve unbiased and equitable outcomes for the individuals affected by software systems. To this aim, researchers have proposed various approaches. For instance, pioneering work by Chakraborty et al. [8] emphasized the importance of developing fair models, laying the foundation for subsequent studies in this domain. Gohar et al. [18] examine the impact of hyperparameters on fairness in machine learning models. Through an empirical study of 168 ensemble models from popular fairness datasets, they explore the composition of fairness and its interaction with different properties.

Their results reveal that fair outcomes can be achieved in ensembles without specific mitigation techniques, and they identify the connections between fairness composition and data characteristics. Biswas et al. [5] propose Fairify, an SMT-based approach for analyzing individual fairness properties in neural network (NN) models. They address the challenge of verifying individual equity in NNs due to non-linear calculations by utilizing input partitioning and NN pruning to provide fairness certification or counterexamples.

As a side note, approaches to cope with the fairness of ML models go beyond software engineering applications. As a side note, approaches to cope with the fairness of ML models go beyond software engineering applications.

The aforementioned pieces of work relate to ours because, once developers have—thanks to models’ transparency—a clear idea about the possible biases affecting a model they are using, proper countermeasures to mitigate such a bias can be taken.

6.2 Documentation of Machine Learning Models

Several papers have dealt with ML models’ documentation. The idea of model cards, i.e., of structured documentation of ML models, was first advocated by Mitchell et al. [35] as a way to foster the transparency and democratization of ML.

Bhat et al. [3] analyzed the content of 132 model descriptions taken from HF, GitHub, and industrial ones to determine how the structure of a model card should look like and implemented a tool named DocML to guide the creation and maintenance of model cards. While we share with the work of Bhat et al. the importance of having certain sections in the model cards, our work has substantial differences with respect to their work. First, we perform a deep analysis of (i) the way datasets are shared, (ii) the types of bias that may affect a model, and (iii) licensing declaration and compliance. Second, our analysis is extensive and concerns a sample of 159,132 models.

Tsai et al. [48] proposed a tool named AIMMX to aid the automated extraction of ML model metadata from different types of software repositories, such as GitHub or ArXiV repositories. We believe the need for tools like AIMMX or DocML is further motivated by our study which shows the limited exposition of relevant pieces of information in models’ descriptions and potential problems that can be caused by model bias or licensing compliance.

Crisan et al. [12] conducted a design inquiry with experts in ML and natural language processing to investigate the usefulness of interactive model cards, showing that such interaction helps stakeholders in model understandability and interpretability, but, also, it increases the level of trust in such models. Our study goes in a different direction as we do not focus on the level of interaction a model card permits but, rather, on its content.

6.3 Bias and Fairness in Software Engineering

Coping with bias and fairness of ML algorithms goes beyond software engineering, and several approaches have been proposed in the literature. Such approaches leverage different techniques, including: (i) optimization techniques like CPFair, that perform a post-hoc re-ranking of recommendations by accounting for fairness requirements; (ii) random sampling to duplicate/remove elements until the bias mitigation goal is reached, [13], or knowledge graph manipulation to account for interaction bias [33].

Beyond that, some authors have focused more in detail, on biases concerning software development. Brun and Meliou [6] studied the potential impact of biased data in software engineering phases, i.e., requirement specification, system design, testing, and verification. They found that a comprehensive classification of software biases remains an open challenge. Spolaorini et al. [46] proposed bias-aware guidelines on how to cope with bias in software engineering tasks.

Given the increasing prominence of pre-trained language models and their extensive applications, particularly in text generation, researchers have studied their biases. Schramowski et al. [43] discovered that pre-trained models possess knowledge of deontological choices, moral norms, and values. In response, they propose a novel approach to identify "MoralDirection" to retrieve human-like biases regarding what is considered right and wrong when employing pre-trained language models for text generation.

6.2 Documentation of Machine Learning Models

Several papers have dealt with ML models’ documentation. The idea of model cards, i.e., of structured documentation of ML models, was first advocated by Mitchell et al. [35] as a way to foster the transparency and democratization of ML.

Bhat et al. [3] analyzed the content of 132 model descriptions taken from HF, GitHub, and industrial ones to determine how the structure of a model card should look like and implemented a tool named DocML to guide the creation and maintenance of model cards. While we share with the work of Bhat et al. the importance of having certain sections in the model cards, our work has substantial differences with respect to their work. First, we perform a deep analysis of (i) the way datasets are shared, (ii) the types of bias that may affect a model, and (iii) licensing declaration and compliance. Second, our analysis is extensive and concerns a sample of 159,132 models.

Tsai et al. [48] proposed a tool named AIMMX to aid the automated extraction of ML model metadata from different types of software repositories, such as GitHub or ArXiV repositories. We believe the need for tools like AIMMX or DocML is further motivated by our study which shows the limited exposition of relevant pieces of information in models’ descriptions and potential problems that can be caused by model bias or licensing compliance.

Crisan et al. [12] conducted a design inquiry with experts in ML and natural language processing to investigate the usefulness of interactive model cards, showing that such interaction helps stakeholders in model understandability and interpretability, but, also, it increases the level of trust in such models. Our study goes in a different direction as we do not focus on the level of interaction a model card permits but, rather, on its content.

6.3 Bias and Fairness in Software Engineering

Coping with bias and fairness of ML algorithms goes beyond software engineering, and several approaches have been proposed in the literature. Such approaches leverage different techniques, including: (i) optimization techniques like CPFair, that perform a post-hoc re-ranking of recommendations by accounting for fairness requirements; (ii) random sampling to duplicate/remove elements until the bias mitigation goal is reached, [13], or knowledge graph manipulation to account for interaction bias [33].

Beyond that, some authors have focused more in detail, on biases concerning software development. Brun and Meliou [6] studied the potential impact of biased data in software engineering phases, i.e., requirement specification, system design, testing, and verification. They found that a comprehensive classification of software biases remains an open challenge. Spolaorini et al. [46] proposed bias-aware guidelines on how to cope with bias in software engineering tasks.

Given the increasing prominence of pre-trained language models and their extensive applications, particularly in text generation, researchers have studied their biases. Schramowski et al. [43] discovered that pre-trained models possess knowledge of deontological choices, moral norms, and values. In response, they propose a novel approach called "MoralDirection" to retrieve human-like biases regarding what is considered right and wrong when employing pre-trained language models for text generation.

Fairness emerges as a crucial concept in software engineering, aiming to achieve unbiased and equitable outcomes for the individuals affected by software systems. To this aim, researchers have proposed various approaches. For instance, pioneering work by Chakraborty et al. [8] emphasized the importance of developing fair models, laying the foundation for subsequent studies in this domain. Gohar et al. [18] examine the impact of hyperparameters on fairness in machine learning models. Through an empirical study of 168 ensemble models from popular fairness datasets, they explore the composition of fairness and its interaction with different properties.

Their results reveal that fair outcomes can be achieved in ensembles without specific mitigation techniques, and they identify the connections between fairness composition and data characteristics. Biswas et al. [5] propose Fairify, an SMT-based approach for analyzing individual fairness properties in neural network (NN) models. They address the challenge of verifying individual equity in NNs due to non-linear calculations by utilizing input partitioning and NN pruning to provide fairness certification or counterexamples.

As a side note, approaches to cope with the fairness of ML models go beyond software engineering applications.

The aforementioned pieces of work relate to ours because, once developers have—thanks to models’ transparency—a clear idea about the possible biases affecting a model they are using, proper countermeasures to mitigate such a bias can be taken.

6.4 Studies on Open Source Licenses

Several authors studied the adoption and usage of open-source licenses, looking at the changes in licensing and licensing-related factors leading towards the success or failure of open-source projects [14], by performing large-scale studies on open-source projects [49], or by surveying developers [51]. Such studies, in general, suggested a general trend towards more permissive licenses, that facilitated the exploitation, even in industrial contexts, of open-source products. This trend is also confirmed by our analysis, conducted on ML models, for which specific licenses are also conceived and adopted.

Germán and Hassan [17] identified integration patterns when creating derivative work under different licenses. These may arise
also for ML-specific projects, as far as models are concerned. Kapi-nsaki et al. propose recommenders for licenses satisfying projects’ constraints and dependencies [27, 30] as well as SPDX-specific compatibility checks [28, 29]. Other work empirically assessed the licensing compatibility in Linux distributions like Fedora [16], or on large sets of open-source projects [52]. Vendome et al. [50] analyzed developers’ discussions to understand what circumstances, related to licenses, lead to “licensing bugs” and how these can be subject to different interpretations and jurisdictions. Similarly, we analyzed compatibility between open-source projects and ML models hosted on HF Hub.

A recent thread of research concerns Software Bills of Materials (SBOMs) [21, 53, 54] and the need for software projects to expose SBOMs as inventory of their content, as also established by Governmental regulations in several countries, e.g., in the US [39]. In particular, Xia et al. [53] performed an interview-based study with 17 practitioners and distilled 25 statements on SBOM practices. In one question of their study, participants pointed out how SBOMs for AI software are different from conventional SBOMs, as they also carry out information about models, training, etc. In the context of AI-specific SBOM formats being defined (AIBOM). In principle, the presence of detailed information on model cards such as those of HF can be used to generate AIBOMs. Therefore, our study investigates the extent to which models document the necessary information to generate AIBOMs.

7 CONCLUSION AND FUTURE WORK

This paper studied the transparency of pre-trained transformer models hosted on the Hugging Face (HF) hub, in terms of training datasets, prediction bias, and declared licenses. We analyzed a total of 159,132 models created for different tasks.

Results of the study indicate that (i) only 14% of the models declare datasets through specific tags, and, by manually analyzing a statistically significant sample of 389 top-downloaded models, we found that 61% of them document the training datasets in some ways. However, only 18% of the analyzed models declare a bias, which is mostly related to the User to Data category of Mehrabi et al. [34] taxonomy, which we further detailed in sub-categories. For what concerns the licenses, we found evidence of a declared license in 31% of the models, with a prevalence toward permissive licenses, and, among other, AI-specific licenses such as the OpenRAIL, that impose a “responsible” model reuse. Nevertheless, there are still several models using restrictive licenses, and this generates multiple (we found 707) cases of incompatibilities in client projects.

In future work, we aim to analyze further dimensions of the models’ transparency, including training parameters, declared performances, and updates/fixing over time. We also plan to complement the qualitative analysis we performed on datasets and bias by automating the identification of bias and model declaration within models’ textual descriptions. Further research will also be conducted toward better and automatically generated AIBOMs.

8 DATA AVAILABILITY

The mined dataset, results of the qualitative analysis, and analysis scripts are available in our replication package [41].

ACKNOWLEDGMENTS

Massimiliano Di Penta acknowledges the Italian PRIN 2020 Project EMELIOT “Engineered MachinE Learning-intensive IoT system”, ID 2020V3A5FY. Federica Pepe is partially funded by the PNRR DM 352/2022 Italian Grant for Ph.D. scholarships. Antonio Mas-tropaolo and Gabriele Bavota acknowledge the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement No. 851720).

REFERENCES
