Potential and Problems in Detecting Privacy Risks in Source Code Using Large Language Models

Ambarish Gurjar
Department of Intelligent Systems Engineering
Indiana University, Bloomington, USA
agurjar@iu.edu

Xinyao Ma
Department of Informatics
Indiana University, Bloomington, USA
maxiny@iu.edu

Anesu Chaora
Department of Informatics
Indiana University, Bloomington, USA
achaora@iu.edu

Vinayak Kumar
Department of Computer Science
Indiana University, Bloomington, USA
kvinayak@iu.edu

Abhijeet Muralidharan
Department of Computer Science
Indiana University, Bloomington, USA
abhmura@iu.edu

L. Jean Camp
Department of Informatics
Indiana University, Bloomington, USA
ljcamp@indiana.edu

***Abstract*—Developers regularly rely on third-party code obtained from collaborative software platforms and code repositories. Despite the benefits, this introduces potential security and privacy risks. The Software Bill of Materials requires a manifest of all code components; however, understanding the possible risks from the inclusion of the components is a deeper challenge. Both privacy and security risks may be contextual, depending on operational environments. In order to evaluate the potential for large language models to identify potential risk, we focus on privacy risks as these are contextual and nuanced. We focus on risks from the inclusion of data types that have been adjudicated as personally identifiable information (PII). We explore the efficacy of Large Language Models (LLMs) in meeting this challenge by testing their performance in the classification of code snippets as processing sensitive data types, or not. We report on the feasibility of using a range of zero-shot and few-shot approaches to automate the detection of source code that includes processing of sensitive data. For this purpose, we developed a labeled corpus of code snippets from GitHub. We report on the accuracy of fine-tuned LLMs (GraphCodeBERT, LongCoder, Mistral, LLama2, and CodeLLama) and commercial prompt-based LLMs (OpenAI's ChatGPT and Google's Gemini). We conclude that language models have the potential to identify privacy risks; however, such models must be trained to meet specific regulatory requirements. Moreover, we find that commercial LLMs may need further development before they are suitable for general use in identifying privacy risks in code.**

***Index Terms*—secure by design, data minimization, privacy, risk management, AI, SBOM, compliance, code quality, LLM**

I. Introduction

Developers frequently rely on code reuse, leveraging collaborative code management systems such as GitHub, SourceForge, or Stack Overflow. However, the advantages of code reuse which add speed and ease to the task of coding also create the disadvantage of incurred security and privacy risks [1]. The inclusion of code that references sensitive data can expose organizations to compliance risk or legal liability. This underscores the pressing need to ensure that organizations have a clear grasp of what data they're compiling, not only for purposes of respecting user's rights but also for avoiding operational risks.

The definition of what constitutes privacy risk varies depending on the context, making it challenging for developers to reach consensus on which data compilations and uses to consider privacy sensitive. Moreover, different jurisdictions and cultures may have distinct guidelines for quantifying the sensitivity of information. Due to the contested and complex nature of privacy, here we focus on potentially sensitive data types from explicit legal categorizations and enumerations. The legal definition of sensitivity of data types also varies across contexts of use; for example, healthcare provision and healthcare research have different standards. Consequently, organizations face difficulties in safeguarding against compliance risks. Even large organizations with extensive compliance processes, including Google, Amazon, H&M, British Airways, and Marriott, have - in the past few years - been fined substantial amounts in excess of 20 million pounds for violating the General Data Privacy Regulation [2] . Identifying indicators of sensitive data, such as variable names or the code in which a variable is embedded, could enable developers to avoid incidental inclusion of code with potential privacy and compliance risks. It could also inform operator compliance decisions when code goes between jurisdictions (e.g., US/EU or across state borders), or across business functions (e.g., patient care to public health dataset). The potential for one-shot and few-shot learning offers the possibility that organizations can train for their own unique combination of code and context.

The overall goal of this work is to explore the possibility of automating identification of sensitive data in code. Traditional methods, like string matching or regular expressions search, have been routinely employed to identify the presence of named variables in code [3]. This study aims to document the efficacy of more sophisticated approaches in capturing potentially sensitive data types. The insights provided by the comparisons of the accuracy of the models and failure modes can inform software development choices and risk management.

Through our study, we seek to answer the following research questions:

RQ1: Is it possible to identify potential privacy-related data risks, arising from third-party code, by applying a Natural Language Processing (NLP) based approach to search for for indicators of sensitive data in code?

RQ2: How well do different artificial language models compare in terms of their ability to identify variables that correspond with the handling of sensitive data?

RQ3: Are there identifiable patterns, or types of failures characteristic of different models?

II. Related Work and Motivation

In this section, we begin with a discussion of how we define sensitive data types. Subsection II-A describes the basis for the labelling of different data types as sensitive. The identification and labelling of data as sensitive is grounded in GDPR and CCPA. Our goal is to both provide a motivate for our selections of data types and to describe our method in enough detail, to enable the reproduction of this research with any enumerated list of sensitive data types.

A. Defining Sensitive Data

In this Subsection II-A, we define what constitutes sensitive data in this experiment. We begin with an overview of two privacy regulations that have well-defined categories for personally identifiable information, i.e., data that is considered personal or protected. Given the variation in definitions across different jurisdictions, we refer to these data types as *sensitive* or *privacy sensitive*.

The General Data Protection Regulation (GDPR) of the European Union and the California Consumer Privacy Act (CCPA) of the State of California are used to define what data are sensitive in our analysis. These two privacy regulations have significant requirements constraining the compilation and processing of Personal Identifiable Information (PII). These regulations are significant factor in software production as California produces roughly 5% of the total global economy and the EU accounting for between 15% and 20%. [4]. Both emphasize foundational principles such as data minimization, which calls for the least amount of data collection necessary, and purpose limitation, to ensure data is used solely for the reasons it was collected (GDPR Article 5 [2]; CCPA Section 1798.100(d)(1) [5]). More importantly, for our purposes, both also clearly enumerate what data is considered sensitive. This method can be reproduced by smaller jurisdictions or industry segments, as long as there are clearly enumerated data categories. Concurrently, the Software Bill of Materials (SBOM) initiative has emerged as a complementary effort aimed at transparency in the software supply chain. As SBOM matures, it has expanded beyond licensing and vulnerability tracking to considerations of development practices. compliance. and privacy [6].

The GDPR was grounded in the 1995 Data Protection Directive, which was an early attempt to identify privacy-sensitive data types and data processing. The GDPR was adopted in 2016 to define individiual data rights. It was intended to provide a unified, consistent, comprehensive framework to safeguard the rights and freedoms of individuals within the European Union. The GDPR defines the responsibilities of organizations compiling, processing, storing, and managing personal data. CCPA and GDPR compliance require that organizations confirm that personal data processed through code dependencies aligns with the regulations and the organizations' stated privacy policies. As a result, organizations are obligated to determine the potential impact of code dependencies on user data privacy and to institute measures that protect personal data throughout software development and supply chain processes. Recognizing that a particular data type covered by the GDPR or CCPA is in a library, utility, or other dependency remains an open challenge.

The CCPA defined statutory principles of data minimization and purpose limitation (Section 1798.100(d)(1) [5]). As with GDPR, it identifies a range of data types. The statute endows consumers with multiple rights, including the right to be informed, to access, and to delete their personal information. In terms of code analysis, aligning with the CCPA requires organizations to evaluate data compilation and processing in code, including dependencies, where the code is used by Californians. The Act has an expansive definition of sensitive data types and defines consumer rights over those types.

There are additional regulations based on specific contexts, notably for health, biometrics, education, employment, children‘s data, and regulations in other legal contexts [7], [8]. This work contributes to understanding the potential, and the risks, of using few-shot and zero-shot techniques described in this work to identify sensitive data types in code. Few-shot training in the context of this paper refers to training samples on a very small dataset. Zero-shot learning refers to prompting LLMs like ChatGPT to leverage their pre-existing knowledge without providing them with any labeled training samples.

III. Methodology

Our research methodology was structured into five stages. First, we conducted a literature survey, as summarized in related work (Section II). We then compiled a data corpus using the GitHub search API for code. We validated all samples through human review of both the code and its implicit or documented use context. With the data validated, we then proceeded to the third stage of fine-tuning large language models. Initially, our evaluation process focused on CodeBERT and GraphCodeBERT [9], [10]. However, the results were not impressive. With the open-sourcing of LLaMA2 by Facebook and the release of Mistral 7B we were able to expand our work [11]–[13]. Our final results include evaluations of GraphCodeBERT, LongCoder, Mistral, LLaMA2, and CodeLLaMA. LongCoder performed particularly well, validating its potential across diverse coding tasks. Results from salesforces T5,T5+ and CodeBERT have been reported [9], [14], [15].

Due to their robust performance and widespread adoption, we included ChatGPT and Gemini Advanced as the fourth stage of our method. This enabled us to evaluate their advanced zero-shot capabilities and benchmark their effectiveness in our specific use case. Lastly in the final stage, we further analyze the responses and the data that we compiled during these stages to answer the research questions that we outline in the Introduction/

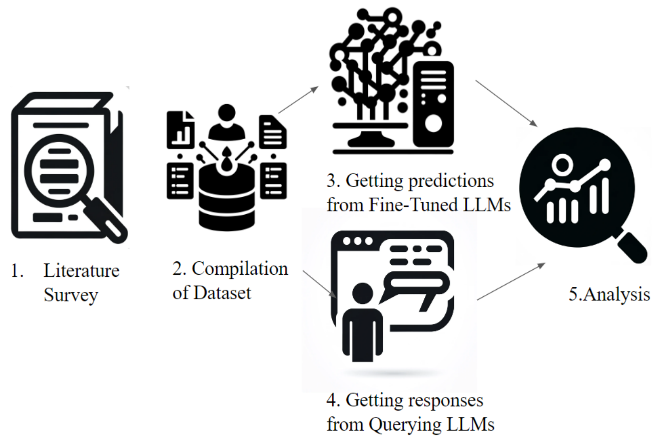
1. Literature Survey
2. Compilation of Dataset
3. Getting predictions from Fine-Tuned LLMs
4. Getting responses from Querying LLMs
5.Analysis

Fig. 1. Overview of methodology

A. Data Corpus

To create a corpus of code containing sensitive information, we began by selecting a subset of categories that are identified in both the California Consumer Protection Act (CCPA) and the European Union's General Data Protection Regulation (GDPR) as sensitive. The ten data information categories we chose were as follows: These were 1. Unique Device ID, 2. Account/Individual Identifier, 3. Demographics, 4. Internet traffic, 5. Commercial/Financial information, 6. Biometrics, 7. Multimedia data, 8. Employment information, 9. Educational information, and 10. Location information.

We leveraged common expressions (e.g., DoB) and code functionality (e.g., payment) to search for code candidates in repositories, using GitHub's code search API. We list the search terms in the appendix. We selected code from commercial public repositories and other popular open-source code. While no small sample can be representative of every production environment, our goal is to explore the potential of zero-shot and few-shot learning for different contexts so we sought widely used code.

To create a control group, we included an equal number of including a large number of code snippets that were not considered sensitive. These code snippets were selected from popular code, and include diverse functionality. The inclusion of non-sensitive code snippets enabled us to assess the specificity and accuracy of our models in distinguishing between sensitive and non-sensitive code.

To ensure the contextual sensitivity of the identified variables these were manually reviewed. The research team (including a professor, a post-doctoral scholar, three interdisciplinary doctoral candidates, one computer science graduate researcher, and one undergraduate computer science researcher). Each individually reviewed the code snippets and determined whether the identified variables were indeed sensitive within the given code contexts. We began with only those samples where the Fliess Kappa score of agreement amongst researchers was equal to 1. Any differences in opinion were identified and discussed in real-time meetings until there was consensus. This collaborative knowledge construction method [16], [17] was a verification stage to add a layer of accuracy to the dataset. Further validation serendipitously occurred when

Category	Repos	Lines	Variable	Code Snippets
Unique Device ID	6	958	27	25
Individual Identifier	9	1643	29	28
Demographics	6	179	20	21
Internet Traffic	7	2549	46	57
Financial Information	8	1068	44	50
Biometrics	5	490	32	38
Multimedia Data	4	1388	20	31
Employment	7	943	29	31
Location	8	1472	50	58
Education	2	102	7	6

TABLE I
SUMMARY OF SEARCHING CODE CORPUS

For the scope of this study, we collected Python code samples to create our dataset as it is widely used and easy to understand for most of the human participants. We searched for identifiable variables in the selected categories, based on CCPA, GDPR, Privado rules, and previous literature. GitHub, as the largest open-source code-sharing platform, offers an extensive repository of publicly accessible code contributed by a diverse community of developers. Its vast collection of code makes it an ideal resource for acquiring a large-scale public dataset suitable for research and analysis in various domains. As we focused on code with potential privacy implications, we searched all Python GitHub repositories for a small set of data privacy-related keywords and data types. Searching for these keywords resulted in a corpus of 53 repositories, 30 keywords, and 86 variables with their related variables and code expressions in total, see Table I for details. We used different keywords for each category domain to search the repositories on the GitHub platform. We also extract the nearby variables and code segments that have the value transmission with the picked variable. The final dataset consisted of 57 code fragments containing sensitive data and 57 fragments without, each roughly 100 lines. Preprocessing was applied to the collected code fragments, involving the removal of comments and non-essential characters.

B. Fine-Tuning Large Language Models for Identification of Privacy risks

Fine-tuned models have exhibited a strong proficiency in few-shot learning, where they can quickly generalize to new tasks by learning from a limited number of examples like our dataset. This contrasts with zero-shot learning scenarios where models apply learned patterns without additional training data. Few-shot learning exploits the models' pre-trained knowledge base, enabling them to identify underlying patterns in sparse data and effectively adapt to novel programming tasks or languages with remarkable efficiency using relatively smaller training samples [18]–[20].

We began our evaluation process with CodeBERT and GraphCodeBERT (described above) [9], [10]. To further re-fine these models, fine-tuning with additional neural network layers, often termed "task-specific heads" or "classification layers," as employed by [21]. These layers are crucial for adapting the model's expansive knowledge to specific tasks, allowing it to detect subtle, task-dependent patterns. With just a handful of examples, models like GraphCodeBERT, LLAMA, LongCoder, CodeLlama, and Mistral can be fine-tuned to significantly enhance their proficiency in making precise code-related predictions or classifications. Through few-shot learning and strategic fine-tuning, these models can become exceptionally adept at interpreting and analyzing code, even in areas with little to no prior exposure.

Preprocessed code fragments underwent tokenization, resulting in sequences of tokens compatible with the respective models. We then processed each tokenized code snippet through all the models. These models transformed the code tokens into high-dimensional vectors, known as embeddings, which effectively encapsulate the syntactic and semantic nuances of the code. To streamline the computational process and manage the high dimensionality of the data, we computed the mean of the tokens for each snippet. This averaging technique yielded a single vector of fixed dimensions per code fragment, which succinctly represented the original richly exhibited data [22]. These condensed vector representations were then used as input in the subsequent stages of sensitivity classification.

In our research, we initially implemented a fully connected neural network (FCN) model. Our preliminary architecture was composed of three layers, containing 64 and 32 hidden units, respectively, with the Rectified Linear Unit (ReLU) serving as the activation function [23]. The neural network predicts the probabilities that the input sample is sensitive or not sensitive. The model was designed and executed using the PyTorch deep learning framework and was optimized using the Adam optimizer [24]. In our model, we used binary cross-entropy as the loss function, which is a standard approach for binary classification tasks [25], [26]. However, we transitioned to a more robust FCN model, consisting of two layers each with 512 nodes. This enhanced network was capable of learning and predicting more intricate patterns in our data, thereby improving the accuracy of sensitivity discernment for variables.

The hyperparameter choices for the models were determined by iterative experimentation and tuning to achieve satisfactory performance. Our initial methodology utilized an 80:20 ratio for training and testing to evaluate the model's performance. To enhance this evaluation, we adopted a k-fold cross-validation method, with k designated as 5 [27]. This method divides the dataset into k equally sized portions, known as 'folds.' In a series of iterations, each fold is used once as the test set while the remaining folds (k-1 in number) serve as the training set. This approach does not just yield an averaged value for metrics such as accuracy, precision, recall, and F1-score, but also captures the variability within the data. It ensures that our performance metrics are not skewed by any specific partition of the dataset. The average values reported from the iterations of the 5-fold cross-validation present a more thorough and reliable measure of the model's performance, having been stringently evaluated across various data subsets.

C. Prompting Large Language Models

In this phase of our methodology, we harnessed the prompt-based classification capabilities of advanced language models, namely ChatGPT 3.5, ChatGPT 4, and Google's Gemini Advanced to assess the privacy sensitivity of the collected code fragments. Zero-shot learning allows these models to perform downstream classifications they were not specifically trained to recognize, by drawing on their extensive pre-training on diverse data. This capability is critical for identifying sensitive information in code (as per GDPR and CCPA guidelines) without the models having previous exposure to such classification tasks [28].

Prior to classification, we assessed the models' knowledge of GDPR and CCPA regulations to ensure their capability to make informed judgments about data sensitivity by asking questions about these regulations. Then, to facilitate classification, we crafted and refined the prompts through an iterative query engineering process. The final prompt directed the models to examine four code fragments, each separated by a specified delimiter symbol in the input, and to determine the presence of sensitive data as per the mentioned guidelines without providing explanations or summaries of the code. We structured the prompt to maximize generalizability and accommodate the language models' interface limitations, asking for a straightforward response indicating which fragments contained sensitive information. Previous work has illustrated that LLMs may validate questions, so our queries were strictly neutral.

The initial query we presented to the models was, "Given below are 4 fragments of code. Each of them is separated by a delimiter. Considering the first code as '1' and the fourth as '4,' identify if the fragments contain sensitive data according to GDPR or CCPA guidelines, indicating your response as 'code fragment number: Sensitive or Not Sensitive'." This query-based approach ensured consistency and uniformity in the sensitivity classification process across the entire dataset. Each code fragment was presented to ChatGPT 3.5, ChatGPT 4, and Gemini models using the same prompt, enabling standardized classification based on their understanding of GDPR and CCPA guidelines. While GPT 3.5 initially showed limitations in performance, its subsequent paid version demonstrated significantly enhanced higher-order reasoning abilities. This iterative refinement of our query approach was informed by comparative studies of large language models in tasks such as sentiment analysis and prompt engineering surveys [29], [30].

In this study, the specific query was carefully formulated to address the challenges posed by the generative nature of language models. Language models have a tendency to provide detailed explanations or summaries of code, which may introduce bias or compromise the objectivity of the analysis. This query maintains a clear restraint over the query parameters by explicitly instructing the language model to refrain from providing explanations or summaries. This ensures that the language model's responses focus solely on assessing the presence of sensitive data in the code fragments. Furthermore, by referencing the GDPR and CCPA guidelines, the query aligns with the legal frameworks for data privacy, enabling a targeted evaluation of code sensitivity. Overall, this query was iteratively optimized to mitigate the risks of information bias, maintain objectivity, and ensure that the analysis is conducted within the scope of the research objectives.

We analyzed the responses generated by ChatGPT 3.5, ChatGPT 4, and Gemini to determine the sensitivity classification of the code fragments. Accuracy, consistency and any challenges associated with the models' inclination to provide detailed explanations rather than straightforward sensitivity labels were considered.

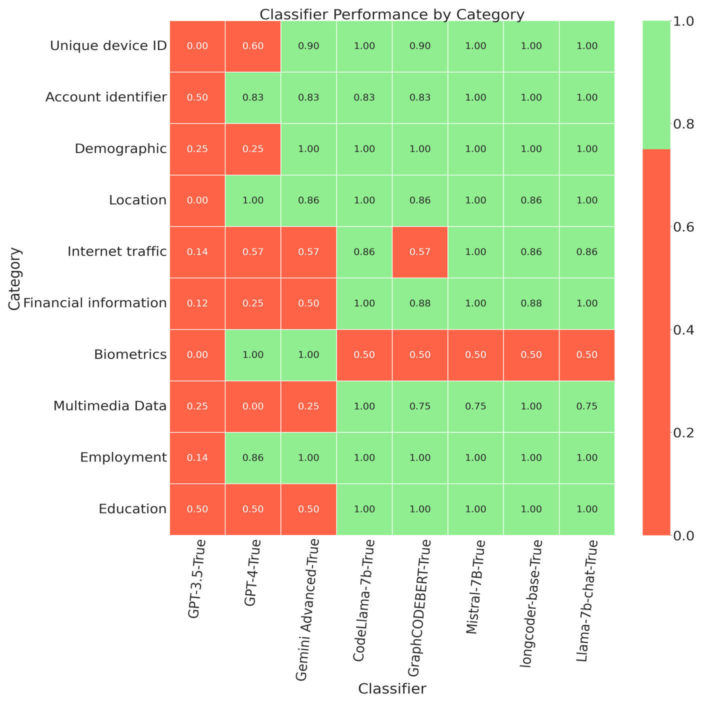

Fig. 2. Heatmap showing how various categories of privacy risks were misclassified. Red squares show poor performance of the language model in the category as the threshold for a good score is 75% accuracy and above

IV. Results

The performance of various models was evaluated using metrics such as Accuracy, F1 score, Precision, and Recall. The results are summarized in Table II. Overall, the results indicate that Mistral 7B and CodeLlama 7B are highly effective at classification of code as containing potentially sensitive data processing. The overall accuracy, precision, and recall of these two LLMs were high, with other models offering varying trade-offs in their levels of performance. Given tolerances for different types of errors, other models may be suitable for different applications.

Mistral 7B outperformed all other models, achieving the highest scores across all metrics with an accuracy of 92.09%. CodeLlama 7B followed closely with an accuracy of 91.26%, and Llama 2 also performed well with an accuracy of 90.39%. Long Coderbase and GraphCodeBERT displayed moderate performance, with accuracies of 88.61% and 80.71%, respectively. The Salesforce models, CodeT5-220m and CodeT5p-770m, showed competitive results, particularly the latter with an accuracy of 81.54%. CodeBERT, while achieving the lowest performance among the evaluated models, still provided useful insights with an accuracy of 72.80%. LLM modes, particularly general chat models, did not perform as well with ChatGPT 3.5, Chat GPT 4.0, and Gemini achieving accuracy of 50%, 73%, and 58% respectively.

TABLE II
Comparison of Model Performance Metrics of finetuned

Model	Accuracy	F1 score	Precision	Recall
Mistral 7B	**92.09%**	**92.04%**	**92.57%**	**92.04%**
CodeLlama 7B	91.26%	91.12%	91.97%	91.12%
Llama 2	90.39%	90.29%	91.62%	90.30%
Long Coderbase	88.61%	88.06%	91.01%	88.48%
GraphCodeBERT	80.71%	80.60%	81.31%	80.51%
CodeT5-220m	78.93%	78.48%	80.47%	78.86%
CodeT5plus-770m	81.54%	81.43%	82.39%	81.66%
CodeBERT	72.80%	72.26%	74.61%	72.87%

V. Discussion

Our results in Table II report a high level of accuracy for fine-tuned models trained on a small corpus. The accuracy of popular chat-based models remained low, regardless of the query. One of our initial limitations was a concern that the corpus of data samples was too small. In contrast, we hypothesized that NLP models that have already been pretrained on extensive datasets could transfer learned knowledge effectively, even when applied to smaller, domain-specific datasets. The accuracy of our results aligns with recent findings in the field, where models trained on less in-distribution data have shown better out-of-distribution performance. These results mitigate concerns about the capabilities of pretrained models [31]. However, the analysis of the results as reflected in the confusion matrices, revealed a concerning trend of high false negatives. Inaccurate classification of sensitive information of Language Models (LLMs) spanned across multiple categories: Unique Device ID, Account/Individual Identifier, Demographics, Internet Traffic, Financial Information, Biometrics, Multimedia Data, Employment Information, Location Information, and Educational Information.

In the case of UniqueID, all the LLMs classified three of sixteen as not sensitive regardless of the fact that such unique identifiers are central to differentiating identifiable from nonidentifiable (and thus less sensitive) data.

We identify the limitations of prominent large language models including ChatGPT and Gemini. These models displayed systematic patterns of failure in classifying code snippets related to both demographic data and geofencing operations. In particular, none of the code snippets which indicated geofencing processes were not categorized as privacy-sensitive by these LLMs. This is not only notable because of extensive research that emphasizes the high privacy risks associated

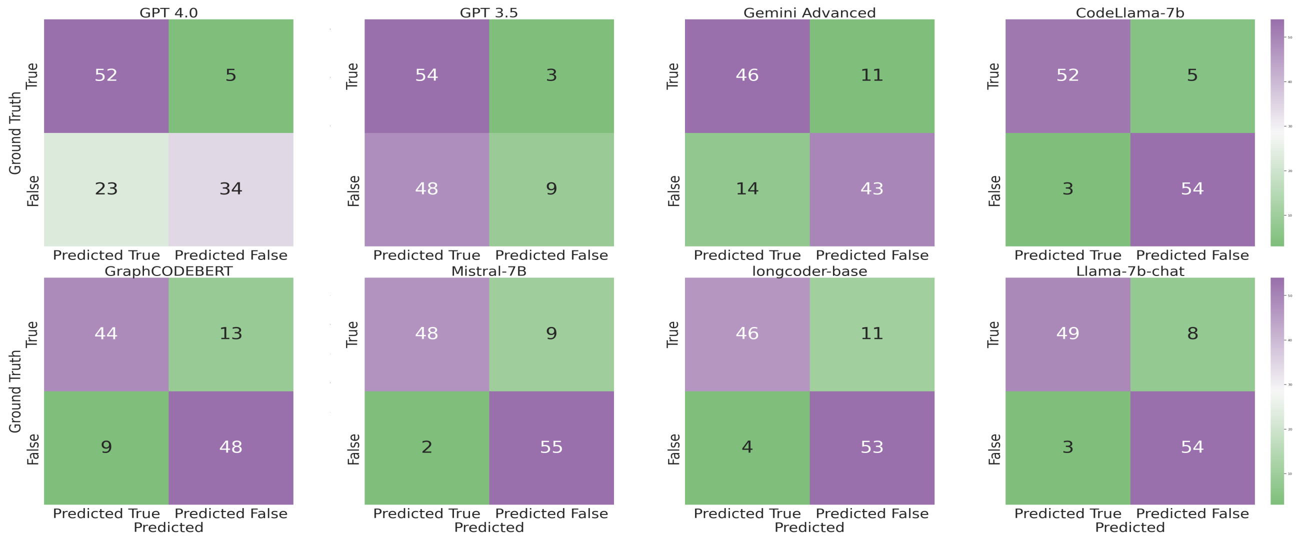

Fig. 3. Confusion Matrices showing the false positive, false negative, true positive, and true negatives of each model. These confusion matrices provide the raw percentages used to calculate precision and accuracy, illustrating the need to explore beyond the report of summary statistics.

with location data (e.g., [32]) with even four spatiotemporal points adequate to uniquely identify an individual within a large dataset, thereby escalating privacy risks [33]. Recall that location data is defined as sensitive.

The processing of demographic data was also frequently misclassified. Three of four instances were subject to incorrect classification. This high rate of failure may be attributed to the small sample size, as discussed in the limitations section of this paper. Our analysis revealed that certain code snippets containing this data type were erroneously classified as non-sensitive by both ChatGPT and Bard. This trend is more notable in light of extensive research documenting the risks of mishandling such sensitive demographic information. Previous studies have indicated that even simple demographic characteristics can uniquely identify individuals within large populations, thus exacerbating privacy concerns [34], [35]. Moreover, careless handling of demographic data not only jeopardizes individual privacy but also has broader societal implications, such as exposing communities to discrimination, stereotyping, and targeted surveillance [36], [37]. Eckersley emphasized that even seemingly innocuous information could uniquely identify users, illustrating the covert ways in which privacy can be compromised [38]. Further adding complexity to this issue are the risks associated with microtargeted ads on platforms such as Facebook, which have been shown to exploit demographic data [39]. Korolova specifically addressed the potential for privacy violations through targeted advertising, shedding light on how such platforms could inadvertently facilitate unauthorized data sharing and re-identification [40]. Given these considerations, our findings underscore the necessity for improved measures to ensure that demographic data are managed as sensitive data, not systematically identified as non-sensitive.

Other categories where all the LLMs misclassified included financial and biometric data. These are contextual in the sense that financial data that was used for payment (other than fraud) are used with consent. Similarly, biometrics may be recognized by LLMs as being components of authentication. Regardless of the presumption of use, financial and biometric data can be sensitive. The systematic misclassification of these categories in comparison with other approaches illustrates the applicability of criticisms of unexamined adoption of LLMs.

These reflect the classic critiques of LLMs listed in [41]. An essential argument of that paper is that the models can be too large to be accurate. Our results can be read as reifying this. The second argument is that social change cannot easily be embedded in a model; consequently, LLMs may be inherently unresponsive to social change. This may apply to the case of privacy, for example, consider network information such as IP addresses. There was a long-standing argument in the first two decades of the century if IP addresses were sensitive, and the differences between jurisdictions remain [42].

The third critique is that LLMs are majoritarian and often inadequately respond to less frequent events (or populations). There is a general problem in privacy (and security) that most code does not have privacy concerns, resulting in biased datasets from the real world. The confusion matrices illustrate this, in the prevalence of false negatives.

Conversely, there were instances where these models displayed an overly cautious approach, erring on the side of overclassification of non-sensitive content as sensitive. Gemini, in particular, demonstrated this flaw when it incorrectly classified an ML classification and cropping algorithm as identifying sensitive financial data in receipts. Upon closer inspection, it was evident that the algorithm in question was solely focused on generic image classification and was not associated with any receipt dataset. Such false positives underline the importance of refining model criteria for sensitivity assessments and ensuring that their determinations are grounded in accurate contextual understanding.

An obvious solution to this would be to simply ask developers to hand label code as sensitive or not sensitive. Yet, previous research on that question illustrated that individual developers disagree about sensitive information, and often do not know that entire categories of data are sensitive. A 2018 survey of 36 developers found that many had never heard of a privacy impact assessment (47.2%), fair information practices (38.9%), nor privacy by design (36.1%) [43]. A later evaluation of 99 developers showed that developers with more experience were not better at identifying code as processing sensitive data [44]. Agreement comparisons between the more expert human coders (0.2033) was slightly higher than the agreement between the models evaluated here (0.2028). (Scores greater than 0.2 indicate consensus.) Human experts categorized sensitive data as not sensitive more often than LLMs and more often than non-experts. Non-expert consensus was measured as 0.2171. The results from LLMs are closer to those of non-experts, perhaps as a result of their training corpus.

VI. Limitations and Future Work

We report that LLMs exhibit systematic failures in identifying sensitive information and thus privacy risks in code. We identified patterns in these errors. However, our data set was too small to determine if these were systematic or random. Some of the errors may indicate an embedded assumption of legitimate use, such as payment information. In other cases, there is no clear reason. Consider that, aside from targeted advertising, the business case for demographic data is limited while the privacy implications are substantial. Some data types, like the date of birth, are widely classified as sensitive personal information and have limited commercial use.

The failures by the models to correctly categorize some code snippets as sensitive underscore a pivotal challenge: despite their ability to process vast quantities of information and generate coherent responses, LLMs are limited in their ability to identify nuanced concerns surrounding data privacy. This gap highlights the urgent need to avoid the unexamined uniform adoption of LLM for the identification of nuanced categorizations. A focus on the recognition and classification of potential privacy threats is needed, especially in light of the substantial risks associated with the mishandling of demographic and geolocation data. Our results support integrating robust location and demographic privacy measures into the ethical guidelines and coding practices that govern LLMs [45].

In future work, we will expand our corpus. We seek a partner to explore how that organizations might employ a Grammarly-like framework to signals parts of code that might violate privacy compliance requirements. We are particularly interesting in working with organizations that share data across operations and research functions.

VII. CONCLUSIONS

There is a need to detect the inclusion of Personally Identifiable Information (PII) in source code. We have reviewed past research focused on identifying privacy risks in code and highlighted the existing gaps and challenges. We report on the accuracy of emerging technologies, such as Generative AI and Large Language Models (LLMs), in offering solutions for automated privacy risk detection in source code. Specifically, we explored the potential for one-shot and few-shot learning for the identification of code that processes sensitive data using a custom-labeled corpus. LLMs varied in their accuracy in the correct classification of sensitive code but were higher than previous research using human evaluators, including previous research on software developers.

Our results underscore the challenges in automated code sensitivity detection and emphasize the need for continuous model refinement. Our results also illustrate that while advancements in language models have provided us with powerful tools for code analysis, understanding their limitations and biases is crucial in deploying them in real-world applications This is arguably especially true for tasks with significant legal and ethical implications such as compliance with privacy laws.

The larger question is if it would be effective for organizations to train LLMs for their own unique combination of code and context. While this justifies the use of a small, carefully labeled dataset, we also acknowledge the size of the dataset is a limitation. Our sample dataset can be augmented with more detailed analysis and is available on the author's Github.[1]

Part of the significance of our work lies in the use of one-shot and few-shot training methodologies to identify privacy sensitivity. A single organization may have different privacy constraints for different units or processes so a single model might not be able to capture the required nuances for all business purposes. Similarly, individuals have differing rights based on their jurisdictions, as well as differing individual personal preferences. Our contributions are to the challenges of selecting products and managing privacy risks based on these constraints. The comparison of the categories and patterns of failures can contribute to the identification of sensitive information in code, and provide insights into the effectiveness of different methods of analysis.

Acknowledgment

This research was supported in part by CTIA and the Comcast Innovation Fund. We would like to acknowledge the contributions of Prof. Tatiana Ringenberg. We acknowledge support from the US Department of Defense [Contract No. W52P1J2093009]. This material is based upon work supported by the U.S. Department of Homeland Security under Grant Award Number 17STQAC00001-07-00." The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of all of these organizations or Indiana University.

References

[1] F. Fischer, K. Böttinger, H. Xiao, C. Stransky, Y. Acar, M. Backes, and S. Fahl, "Stack overflow considered harmful? the impact of copypaste on android application security," in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 121–136.

[1]https://github.com/ambarishgurjar/PrivacySensitivityDataset

[2] European Parliament and Council of the European Union, "Regulation (EU) 2016/679 of the European Parliament and of the Council (General Data Protection Regulation (GDPR))." [Online]. Available: https://data.europa.eu/eli/reg/2016/679/oj

[3] Privado. [Online]. Available: https://github.com/Privado-Inc/privado

[4] J. Desjardins, "The $86 trillion world economy in one chart," 2019.

[5] California Civil Code, "California Consumer Protection Act, (CCPA)," July 2018.

[6] Cybersecurity and Infrastructure Security Agency $SBOM$ Tools Working Group, "Types of Software Bill of Materials (SBOM)," Apr. [Online]. Available: https://www.cisa.gov/resources-tools/resources/types-software-bill-materials-sbom

[7] 93rd United States Congress, "Family Educational Rights and Privacy Act (FERPA) of 1974," 1974, available from: https://www2.ed.gov/policy/gen/guid/fpco/ferpa/index.html.

[8] 104th United States Congress, "Health Insurance Portability and Accountability Act (HIPAA), 1996," accessed on 10-31-2023 Available from: https://www.hhs.gov/hipaa/index.html.

[9] Z. Feng, D. Guo, D. Tang, N. Duan, X. Feng, M. Gong, L. Shou, B. Qin, T. Liu, D. Jiang, and M. Zhou, "CodeBERT: A Pre-Trained Model for Programming and Natural Languages," *Findings of EMNLP 2020*, p. 12, Sep. 2020. [Online]. Available: https://github.com/microsoft/CodeBERT

[10] D. Guo, S. Ren, S. Lu, Z. Feng, D. Tang, S. Liu, L. Zhou, N. Duan, A. Svyatkovskiy, S. Fu *et al.*, "GraphCodeBERT: Pre-training code representations with data flow," *International Conference on Learning Representations*, 2021.

[11] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. d. l. Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier *et al.*, "Mistral 7b," *arXiv preprint arXiv:2310.06825*, 2023.

[12] e. a. p. Hugo Touvron et al, year=2023, "Llama 2: Open foundation and fine-tuned chat models."

[13] B. Roziere, J. Gehring, F. Gloeckle, S. Sootla, I. Gat, X. E. Tan, Y. Adi, J. Liu, T. Remez, J. Rapin *et al.*, "Code llama: Open foundation models for code," *arXiv preprint arXiv:2308.12950*, 2023.

[14] Y. Wang, W. Wang, S. Joty, and S. C. Hoi, "Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation," *arXiv preprint arXiv:2109.00859*, 2021.

[15] Y. Wang, H. Le, A. D. Gotmare, N. D. Bui, J. Li, and S. C. H. Hoi, "Codet5+: Open code large language models for code understanding and generation," *arXiv preprint*, 2023.

[16] M. J. Belotto, "Data analysis methods for qualitative research: Managing the challenges of coding, interrater reliability, and thematic analysis," *The Qualitative Report*, vol. 23, no. 11, pp. 2622–2633, 2018.

[17] C. E. Hmelo-Silver, "Analyzing collaborative knowledge construction: Multiple methods for integrated understanding," *Computers & Education*, vol. 41, no. 4, pp. 397–420, 2003.

[18] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum, "Human-level concept learning through probabilistic program induction," *Science*, vol. 350, no. 6266, pp. 1332–1338, 2015.

[19] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM Comput. Surv.*, vol. 53, no. 3, pp. 1–34, 2020.

[20] 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.

[21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:52967399

[22] J. Wieting, M. Bansal, K. Gimpel, and K. Livescu, "Towards universal paraphrastic sentence embeddings," in *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 2016.

[23] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 807–814, 2010.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[25] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[26] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org

[27] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning*. New York, NY, USA: Springer, 2013.

[28] Y. Xian, C. H. Lampert, B. Schiele, and Z. Akata, "Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 9, pp. 2251–2265, 2019.

[29] J. Lossio-Ventura, R. Weger, A. Lee, E. Guinee, J. Chung, L. Atlas, E. Linos, and F. Pereira, "Sentiment analysis of COVID-19 survey data: A comparison of ChatGPT and fine-tuned OPT against widely used sentiment analysis tools," 2023.

[30] B. Liu, Y. Wang, H. Jin, and P. He, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," *arXiv preprint arXiv:2107.13586*, 2021.

[31] N. F. Liu, A. Kumar, P. Liang, and R. Jia, "Are sample-efficient nlp models more robust?" 2023.

[32] R. Shokri, G. Theodorakopoulos, J.-Y. L. Boudec, and J.-P. Hubaux, "Quantifying location privacy," in *Security and Privacy (SP), 2011 IEEE Symposium on*. IEEE, 2011, pp. 247–262.

[33] Y.-A. de Montjoye, C. A. Hidalgo, M. Verleysen, and V. D. Blondel, "Unique in the crowd: The privacy bounds of human mobility," *Scientific Reports*, vol. 3, p. 1376, 2013.

[34] L. Sweeney, "Simple demographics often identify people uniquely," *Health (San Francisco)*, vol. 671, no. 2000, pp. 1–34, 2000.

[35] P. Golle, "Revisiting the uniqueness of simple demographics in the us population," ser. WPES '06. New York, NY, USA: ACM, 2006, p. 77–80. [Online]. Available: https://doi.org/10.1145/1179601.1179615

[36] S. Barocas and A. D. Selbst, "Big data's disparate impact," *California law review*, pp. 671–732, 2016.

[37] D. Lyon, *Surveillance, Snowden, and Big Data: Capacities, consequences, critique*, 2014, vol. 1, no. 2.

[38] P. Eckersley, "How unique is your web browser?" in *Proceedings of the 10th International Conference on Privacy Enhancing Technologies*, ser. PETS'10. Berlin, Heidelberg: Springer-Verlag, 2010, p. 1–18.

[39] S. C. Matz, M. Kosinski, G. Nave, and D. J. Stillwell, "Psychological targeting as an effective approach to digital mass persuasion," *Proceedings of the National Academy of Sciences*, vol. 114, no. 48, pp. 12 714–12 719, 2017.

[40] A. Korolova, "Privacy violations using microtargeted ads: A case study," in *2010 IEEE International Conference on Data Mining Workshops*. IEEE, 2010, pp. 474–482.

[41] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell, "On the dangers of stochastic parrots: Can language models be too big?" in *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, ser. FAccT '21. New York, NY, USA: ACM, 2021, p. 610–623. [Online]. Available: https://doi.org/10.1145/3442188.3445922

[42] Court of Justice of the European Union, "Judgment of the Court (second chamber) on case c-582/14: Patrick Breyer v. Bundesrepublik Deutschland," European Court Reports, 2016.

[43] A. Senarath and N. A. Arachchilage, "Why developers cannot embed privacy into software systems? an empirical investigation," in *Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018*, 2018, pp. 211–216.

[44] X. Ma, A. Gurjar, A. Chaora, and L. J. Camp, "Programmer's perception of sensitive information in code," in *Symposium on Usable Security and Privacy (USEC) 2024*. Network and Distributed System Security, 2024.

[45] Shokri, Reza and Theodorakopoulos, George and Boudec, Jean-Yves Le and Hubaux, Jean-Pierre, "Privacy games: Optimal user-centric data obfuscation," *Proceedings of Privacy Enhancing Technologies*, vol. 2015, no. 2, pp. 299–315, 2015.