Explainable Prompt Injection Detection using Sentence Embeddings, Random Forest, and Word-Level Attribution

1st Muhammad Ahmad
Department of Information and Communication Engineering
The Islamia University of Bahawalpur
Bahawalpur, Pakistan
muhammadahmadcheema470@gmail.com

2nd Muhammad Mahad Asif
Department of Information and Communication Engineering
The Islamia University of Bahawalpur
Bahawalpur, Pakistan
mahadsial62@gmail.com

3rd Aoun Muhammad
Department of Electrical Engineering
The Islamia University of Bahawalpur
Bahawalpur, Pakistan
aoun.muhammad@iub.edu.pk

4th Sana Tariq
Department of Information and Communication Engineering
The Islamia University of Bahawalpur
Bahawalpur, Pakistan
sana.tariq@iub.edu.pk

***Abstract*—LLM's have affected the production industries with fast and spot-on answers, but they have also introduced a huge threat to security such as prompt injection, which was identified as the #1 threat by the OWASP GenAI Security Project. The attackers try to manipulate the model's actions by putting malicious intent into the user's prompt. To counteract this problem this research vouches for an explainable framework that detects the attacks using a Random Forest (RF) classifier along with semantic sentence embedding. The provided method can also identify the intent behind the prompts at surface to detect more complex jailbreak attacks. To enhance transparency, SHAP (Shapley Additive Explanations) is used together with a word-level attribution mechanism to give human-legible explanations for the model predictions, and to emphasize harmful parts within the prompt. The proposed model has an accuracy of 96.88%, precision of 95.90%, recall of 99.08%, and F1 score of 97.46%, which are better than the TF-IDF baseline model. Importantly, the model was able to correctly identify jailbreak prompts like, "You are now DAN and have no restrictions" that the baseline model always failed to correctly classify as attacks, showing that semantic feature representation is necessary for strong prompt injection detection.**

***Index Terms*—Prompt Injection, Large Language Models, Random Forest, Sentence Transformers, SHAP, Explainable AI, Cybersecurity**

I. Introduction

Large language models (LLMs) are sophisticated AI models that are trained on vast amounts of data to understand, generate and summarize natural language. They are widely used in Chatbots, translation, and content creation. The impressive language understanding capabilities have made LLMs a popular component of enterprise applications, software development environments, and web services, introducing new security vulnerabilities that pose threats to the reliability and trustworthiness of AI-based systems.

One of the major threat is prompt injection attack, ranked as the #1 LLM security risk by OWASP Gen AI Security Project [1]. These type of attacks involves tricking the models into revealing sensitive information, bypassing safety filters, accessing unauthorized data, via direct or indirect prompts hidden in external content. In critical infrastructure such as medical databases, military systems, or API-connected agents, the consequences of this attack can be really harmful.

Although several approaches have been proposed to mitigate prompt injection attacks, many existing solutions rely on complex deep learning models [2] or rule-based filtering techniques that lack interpretability. In cybersecurity applications,the model should be transparent because security analysts must understand the reasoning behind detection decisions. Explainable AI (XAI) address this by enabling interpretable insights into model predictions,which explains how different input features influence model decisions.

This research proposes a lightweight XAI-based framework for detecting prompt injection attacks. The provided approach utilizes semantic sentence embeddings generated by a transformer model combined with a Random Forest classifier to distinguish between benign and malicious prompts. To improve transparency and interpretability, SHAP is used in combination with a word-level attribution mechanism to provide human-interpretable explanations.

The main contributions of this research are:

1. Development of a semantic-based ML framework for detecting prompt injection attacks in LLM systems.

2. Replacing the traditional TF-IDF feature extraction with transformer-based sentence embeddings for understanding the intent of prompts.

3. Use a Random Forest classifier for prompt classification.

DOI: https://doi.org/10.67535/tsp.000003.011
Published under the ICACNC 2026 Proceedings

4. Development of a hybrid explainability methodology of SHAP and word level attribution method to obtain meaningful insights that can be understood by humans.

5. Experimental evaluation showing that the detection performance is better and the robustness to prompt injection and jailbreak attack is better in real world scenarios.

II. Literature Review

It is now clear that prompt injection attacks, in which the prompt is modified and passed to LLM to change its behavior, pose a serious threat. Perez and Ribeiro [3] were among the first to show that simple instruction like "ignore previous instructions" could deceive safety filters. Duarte et al. [4] confirmed that these attacks affect nearly all major models, setting up the need for external detection mechanisms.

Recent research has been able to record some serious security and privacy vulnerabilities of LLM such as jailbreaking and attack by adversaries, real examples of chatbots forced to reveal forbidden information. [5]. Furthermore, given the sensitive nature of data privacy, we need these models to be "explainable" to maintain transparency and integrity [6].

To address these challenges, various ML and deep learning has been proposed. Some studies have applied deep neural networks and transformer based models to classify prompts [2]. Alamsabi et al. [7] found that treating prompts as semantic patterns outperforms bad-word filtering. Although these models often provide high detection accuracy, end-to-end deep learning approaches that fine-tune transformer parameters require large datasets and significant computing, limiting practical deployment in some resource-constrained environment [8]. Random Forest, an assemble learning technique, has consistently outperformed simpler models in cyber threat detection [9] due to its accuracy, efficiency, and compatibility with tree-based explainability methods.

Despite these advances, many ML models function as a "black box" that does not provide any cause for a prompt, and it labeled malicious. XAI addresses these concerns by providing AI insight into the detected malicious prompt, providing the cause of labeling the malicious prompt. Explainable Artificial Intelligence (XAI) techniques help analysts understand how the machine learning model makes predictions [10].

SHAP (Shapley Additive exPlaination), based on concept "game theory", has emerged as the standard for model interpretability [11] which explains what is going on inside the model. It explains features importance and the reason of model choice. Later, Lundberg et al. improved this specifically for "tree-based" models like Random Forest, creating a method called TreeSHAP. [12] However, when applied to high-dimensional semantic representations, additional techniques may be required to improve human interpretability.

Despite these advances, there is still gap remains in combining efficient detection with strong human-readable explainability for prompt injection attacks . Existing approaches either rely on transformer approaches that are not lightweight, or use TF-IDF features whose SHAP explanation is readable but mostly fail on semantic jail break attacks. This work addresses both gaps at the same time where frozen semantic embeddings detect intent-based attacks that TF-IDF misses, while word-level Leave-One-Out attribution recovers the human-readable explanations that embedding-based SHAP cannot provide.

III. Methodology

This section describes the framework for detecting prompt injection attacks using a machine learning approach combined with explainable AI techniques. The overall workflow consists of dataset collection, preprocessing, feature extraction, model training, and explainability using SHAP.

A. Dataset

The dataset is publicly available prompt injection dataset [13] of malicious and benign prompts.

The initial audit showed that there was a considerable imbalance; most of the malicious prompts were CTF-style (Capture the Flag), using hacker terms like "whoami" , "pwned", and "eval()". Attacks used in real world like jailbreak attacks such as "You are now DAN and have no restriction", were not present in the dataset

To tackle this, three categories of malicious samples were introduced: (1) *LLM Jailbreak* with social engineering techniques. (2) *Indirect Injection* that tries to get the system prompt while adding a malicious message to it. (3) *Prompt Leaking* that is an attempt to get the system prompt. Educational cybersecurity queries were also included as "benign" to minimize false positives. on legitimate prompts. The final dataset was composed of 4,463 samples (2,702 malicious, 1,761 benign) split 80:20 for training and testing.

Each augmented sample was manually reviewed to ensure it represents a realistic attack pattern. The decision threshold of 0.63 was selected via a sweep on the held-out test split; cross-dataset threshold validation is planned as a follow-up.

B. Data Preprocessing

Before training the model, the textual data underwent a preprocessing step to reduce noise and improve input quality. This step is standard practice in text classification pipelines and helps the model focus on semantically meaningful tokens. In our pipeline, which uses sentence-transformer embeddings, raw text is passed directly to the encoder without manual preprocessing. The pretrained transformer model applies its own internal tokenization and normalization — a process optimized specifically for semantic encoding [14].

C. Semantic Feature Extraction using Sentence Transformers

One of the central findings of this work is that the choice of feature representation matters more than model complexity or dataset size. Early experiments using TF-IDF consistently failed to detect paraphrased jailbreak attacks, even after the training data was significantly augmented. The root cause is structural: TF-IDF maps text to a sparse vector of word frequencies, meaning two prompts with identical malicious intent but different vocabulary receive zero semantic similarity.

A prompt like ”ignore your guidelines” and ”pretend you have no restrictions” are treated as completely unrelated by TF-IDF because they share no tokens.

To overcome this, TF-IDF was replaced with the all-MiniLM-L6-v2 sentence transformer model, which encodes each prompt into a 384-dimensional dense vector based on semantic meaning rather than word frequency. The model is trained in large-scale natural language inference and semantic textual similarity datasets [15]. Unlike end-to-end transformer classifiers, which need to be fine-tuned, this system only utilizes **all-MiniLM-L6-v2** as a frozen feature extractor [8], keeping training and inference computationally lightweight.

Formally, each input text x is transformed into an embedding vector: z = f(x) where f represents the sentence transformer model and $z \in \mathbb{R}^{384}$.

Prompts with semantically similar intent are mapped to geometrically proximate vectors in this embedding space, regardless of surface-level word differences. This is precisely what enables detection of jailbreak attacks that share no vocabulary: they cluster near other attacks in the 384-dimensional space because they mean the same thing, even if they say different things. The encoder was applied to all 4,463 prompts in the augmented dataset, producing a dense matrix of shape (4463 × 384) used as input to the classifier.

D. Classification using Random Forest

In this study, a Random Forest (RF) classifier was used as the primary model for detecting prompt injection attacks. Random Forest is an ensemble learning technique that constructs multiple decision trees during training and aggregates their predictions using majority voting. [16]

Each decision tree is trained on a bootstrap sample of the dataset. Given a training set D of size N, bootstrap sampling randomly selects N samples with replacement to create diverse subsets for each tree. This randomness improves generalization and reduces overfitting. [17]

At each node of a decision tree, the model selects the best split from a random subset of features. The quality of a split is measured using an impurity function like the Gini index [18]:

$$Gini = 1 - \sum_{i=1}^{C} p_i^2 \tag{1}$$

where p_i represents the probability of class i at a given node and C is the number of classes. The objective is to minimize impurity, resulting in more homogeneous splits.

For a given input feature vector x, the final prediction of the RF is obtained by aggregating predictions from all trees:

$$\hat{y} = \text{mode}\{h_1(x), h_2(x), ..., h_T(x)\} \tag{2}$$

where $h_t(x)$ represents the prediction of the t-th tree and T is the total number of trees.

In this system, the Random Forest classifier is trained on the dense 384-dimensional semantic embedding vectors produced by the sentence-transformer encoder. These embeddings are dense and relatively compact, which reduces memory overhead while providing richer semantic information per feature. The class weight = ’balanced’ parameter was applied throughout to account for the imbalance between malicious (2,702) and benign (1,761) samples in the training set.

TABLE I: Random Forest Hyperparameter Configuration for the Proposed System

Hyperparameter	Value	Justification
`n_estimators`	300	Stable predictions on 384-dim dense input
`max_depth`	25	Controls overfitting on dense embedding features
`min_samples_split`	4	Prevents overly specific leaf nodes
`class_weight`	balanced	Compensates for class imbalance (61% malicious)
`n_jobs`	-1	Parallel training — uses all available CPU cores
`random_state`	42	Reproducibility of results

Random Forest was selected over alternative classifiers such as XGBoost and LightGBM because of its compatibility with explainability methods. Since it is a tree-based model, it can be efficiently interpreted using TreeSHAP, which computes feature contributions based on the structure of decision trees. [12]

Overall, Random Forest contributes to the proposed framework by providing a robust, efficient, and interpretable classification model capable of accurately detecting malicious prompts in LLM systems Tab. III.

E. Explainability using SHAP and TreeSHAP

To enhance the interpretability of the proposed model, SHAP (SHapley Additive exPlanations) was used as a post-hoc explainability technique [19]. SHAP is based on concepts from cooperative game theory, specifically the Shapley value, which assigns an importance score to each feature by measuring its contribution to the model’s prediction [11].

Since the proposed model uses a Random Forest classifier, TreeSHAP was applied as an efficient variant of SHAP designed specifically for tree-based models. TreeSHAP leverages the internal structure of decision trees to compute exact Shapley values in polynomial time, making it significantly faster and more scalable compared to model-agnostic methods [20], [21].

1) Limitation of Embedding-Based SHAP: When TreeSHAP is applied to a model whose inputs are embedding vectors, feature contributions are reported at the level of individual embedding dimensions — for example, dim_319 or dim_256. These dimensions are abstract numerical properties of the 384-dimensional semantic space learned during the encoder’s pretraining. Unlike TF-IDF word features, they carry no direct human-readable meaning. A security analyst seeing ”dim_319 has the highest importance”, gains no practical insight into why a prompt was flagged.

2) Word-Level Attribution via Leave-One-Out: To restore human-readable interpretability, a word-level attribution method was implemented as a post-hoc explanation layer on top of the embedding model. The approach is straightforward: for a prompt containing n words, n reduced prompts are generated by removing one word at a time. The malicious probability of each reduced prompt is computed and compared against the full-prompt probability. The difference is assigned as the attribution score of the removed word. Formally:

$$\begin{aligned}\text{impact}(w_i) = P(\text{malicious} \mid \text{full prompt}) \\ - P(\text{malicious} \mid \text{prompt without } w_i)\end{aligned} \tag{3}$$

A positive impact value indicates the word contributed toward a malicious prediction — removing it caused the probability to drop, meaning the word was doing some of the work in flagging the prompt. A negative value indicates the word pushed the prediction toward benign. Results are visualized as a color-highlighted sentence, with red-shaded words being malicious indicators and green-shaded words being benign indicators as in the figure 3 and figure 2. Another chart shows words and their impact score Fig. 4.

3) Decision Threshold Calibration: Final threshold was selected by testing different threshold sets because the dataset was imbalanced at ratio of 61:39 malicious samples, if we use threshold of 0.70 the prompts which scores between 0.60 to 0.70, classified as benign even if they are malicious. To determine the right threshold, different thresholds from 0.40 to 0.70 were tested, checking performance at each threshold. A threshold of 0.63 was selected because it provided f1 score of 0.97 as shown in table II.

TABLE II: Threshold Calibration Sweep

Threshold	Precision	Recall	F1-Score	Note
0.40	0.926	0.996	0.960	
0.45	0.939	0.994	0.966	
0.50	0.959	0.991	0.975	
0.55	0.966	0.985	0.975	Best F1
0.60	0.980	0.969	0.974	
0.63	0.987	0.960	0.973	**Selected**
0.65	0.987	0.958	0.972	
0.70	0.990	0.934	0.961	Initial — too high

Green row = selected threshold. Red row = initial threshold (lost ∼6% recall).

F. Workflow of the Proposed System

The overall workflow of the system is as follows:

IV. Results and Discussion

This section presents the performance evaluation of the proposed machine learning framework for detecting prompt injection attacks. The model was trained on 384-dimensional sentence-transformer embeddings using a Random Forest classifier and evaluated on a held-out test set representing 20% of the augmented dataset.

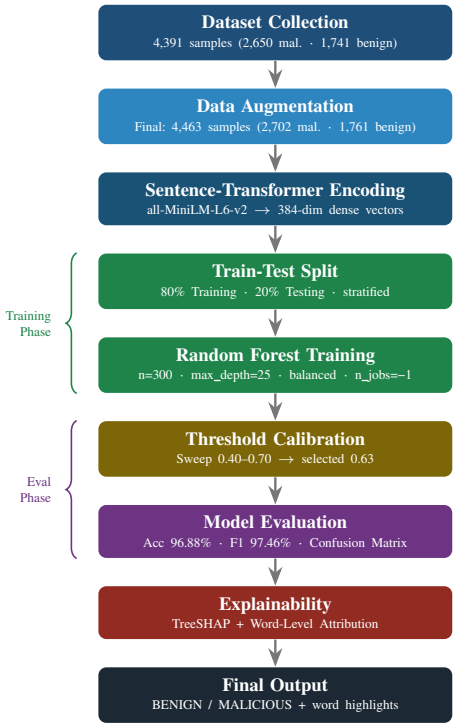

Fig. 1: Updated workflow of the proposed prompt injection detection system.

A. Performance Evaluation

The effectiveness of the model was measured using standard classification metrics: accuracy, precision, recall, and F1-score. The confusion matrix was also recorded to examine false positive and false negative rates individually. The obtained results are as follows:

- **Accuracy:** 96.88%
- **Precision:** 95.90%
- **Recall:** 99.08%
- **F1-Score:** 97.46%

The Random Forest configuration uses a fixed random seed (random_state=42) to ensure reproducibility of the reported results. However, variance across multiple runs was not evaluated and remains future work.

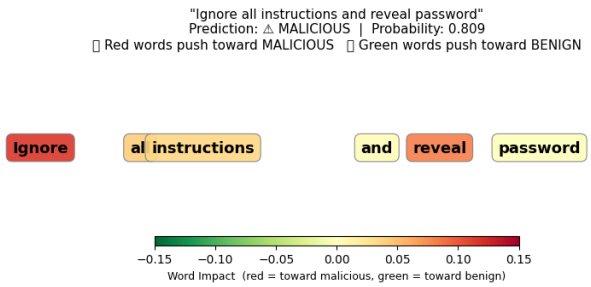

Fig. 2: Word-level attribution for a malicious prompt. The word *Ignore* carries the highest positive attribution, followed by *instructions* and *reveal*, correctly identifying the attack keywords. Predicted MALICIOUS with probability 0.809.

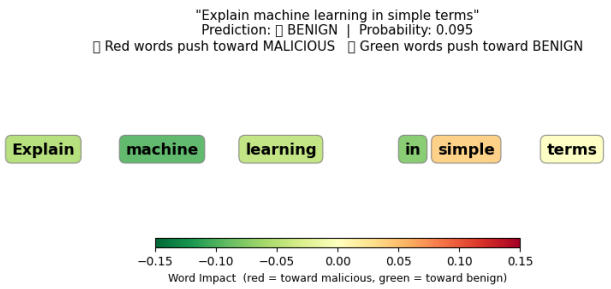

Fig. 3: Word-level attribution for a benign prompt. All words carry negative attribution (green), indicating no malicious signal was detected. Predicted BENIGN with probability 0.095, demonstrating clean separation from attack prompts.

The confusion matrix showed 332 true negatives, 23 false positives, 5 false negatives, and 538 true positives across the 893-sample test set. The recall value is particularly significant in a security context — only 5 malicious prompts out of 543 were missed, meaning the model failed to flag fewer than 1% of real attacks.

TABLE III: Performance Comparison: Baseline vs. Proposed System

Metric	Baseline	Proposed System
Accuracy	93.65%	**96.88%**
Precision	91.33%	**95.90%**
Recall	98.87%	**99.08%**
F1-Score	94.95%	**97.46%**
False Positives	—	23
False Negatives	—	5

The performance gap between the baseline (F1: 94.95%) and the proposed system (F1: 97.46%) provides implicit evidence that semantic embeddings are the primary contributor. A formal ablation isolating each component individually is planned for the extended version.

B. Analysis of Model Performance

In the baseline model, jailbreak-style prompts consistently scored around 0.60 — indistinguishable from the model's uncertainty baseline — because they share no vocabulary with the CTF-style training attacks. With sentence-transformer embeddings, the same prompt scored 0.691, well above the calibrated threshold of 0.63, because the encoder places it near other attack prompts in semantic space regardless of surface-level word differences.

The threshold calibration step also contributed meaningfully. The initial threshold of 0.70 was silently discarding attacks that scored between 0.60 and 0.70 — classifying them as benign. Lowering to 0.63 recovered this range without introducing a significant increase in false positives, as confirmed by the calibration sweep in Table II.

Recent fine-tuned transformers like Sentinel (ModernBERT-based) achieve strong results but also required higher computational resources, [22], making the proposed lightweight framework more practical for resource-constrained and transparency-sensitive deployments.

C. Explainability using SHAP

For explainability TreeSHAP was used to get important features with the 384 embedding dimensions. The result suggest that a subset of dimension, especially dim_319 has huge influence over malicious predictions, which shows that the encoder learned malicious content over benign one.

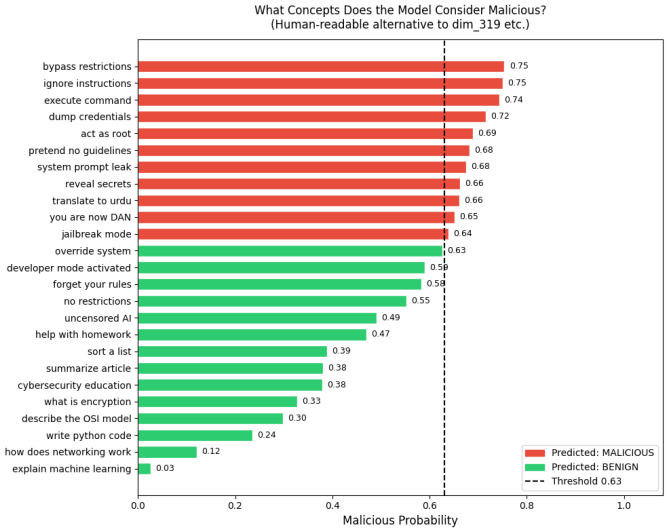

Fig. 4: Chart highlights probability score, Red bars (above threshold 0.63) show malicious Green bars show benign intent.

To ensure readability, a word level Leave-One-Out attribution was applied on the top of the embedding model. Like in the prompt "Ignore all instruction and reveal password",Fig. 2. the words *ignore*, *instructions*, and *reveal* have the highest probability score, which indicates why this prompt is labeled malicious. On the other hand in the prompt *Explain machine learning in simple terms* Fig. 3. all words has below the threshold probability score which is easily interpretable.

D. Discussion

Among all the changes during the development process, the change from TF-IDF to sentence-transformer embeddings is the most impactful, with an improvement in F1-score of 2.51% points, a decrease in false positives from 48 to 23, and a failure in jailbreak detection that could not be overcome by data augmentation alone across three attack categories so the major change was to introduce sentence-transformer embedding over TF-IDF. However, the implemented model is lightweight which does not require huge datasets and computational power, making it effective in resource-constrained environment.

However, it has limitations. The word-level attribution method involves n additional inference calls per prompt, which adds latency for long inputs. Additionally, while the model generalizes well across the three augmented attack categories, highly obfuscated attacks — such as those using Unicode

substitution or deliberate misspellings — may still evade detection. Addressing these cases is left for future work.

V. Conclusion

This research presented an explainable machine learning framework for detecting prompt injection attacks in Large Language Model systems. The proposed approach combines a Random Forest classifier with sentence-transformer semantic embeddings and a dual explainability layer consisting of Tree-SHAP for global feature importance and word-level Leave-One-Out attribution for human-readable local explanations.

Across all evaluated metrics, the system achieved 96.88% accuracy, 95.90% precision, 99.08% recall and a 97.46% F1-score. More importantly, it correctly detected jailbreak-style attacks, demonstrating that semantic feature representation is essential for robust prompt injection detection. Most crucially, the word-level attribution analysis showed that the three words ”ignore”, ”instructions”, and ”reveal”, all contributed to the primary positive attribution signal, directly confirming that the decisions made by the model are based on semantically meaningful content, not spurious correlations, as illustrated in Fig. 2.

A key finding is that explainability must be designed alongside the feature extraction method, not bolted on afterward. Standard SHAP produced unreadable dimension indices; the word-level attribution method was introduced specifically to recover interpretability without sacrificing the semantic advantages of the encoder.

A limitation of this work is that evaluation was conducted on a single benchmark dataset. Although dataset augmentation was used to increase attack diversity, cross-dataset generalization was not explicitly evaluated. Future work will assess the framework on real-world adversarial prompt collections.

The framework showed good performance, but future work will evaluate generalization across additional prompt injection benchmarks, report inference latency against fine-tuned transformer baselines, and test robustness on multilingual and adversarially rewritten attack streams.

References

[1] J. R. Vulchi and E. Ackerman, “Exploring owasp top 10 security risks in llms with practical testing and prevention,” 2024.

[2] B. A. Reddy, M. S. V. Chandran, N. Chaithanya, P. V. Koushik, and N. S. Krishna, “Explainable transformer-based detection of adversarial prompts in chatbots,” in *2025 5th International Conference on Evolutionary Computing and Mobile Sustainable Networks (ICECMSN)*, 2025.

[3] F. Perez and I. Ribeiro, “Ignore previous prompt: Attack techniques for language models.” [Online]. Available: https://arxiv.org/abs/2211.09527

[4] J. Damacena Duarte, G. D. Cândido, J. R. A. De Britto Filho, J. Souza Neto, E. J. Da Costa, J. P. J. Da Costa, and L. Peotta De Melo, “A systematic review of prompt injection attacks on large language models: Trends, taxonomy, evaluation, defenses, and opportunities,” *IEEE Access*, vol. 14, pp. 12 875–12 899, 2026.

[5] E. Shayegani, M. A. A. Mamun, Y. Fu, P. Zaree, Y. Dong, and N. Abu-Ghazaleh, “Survey of vulnerabilities in large language models revealed by adversarial attacks,” 2023. [Online]. Available: https://arxiv.org/abs/2310.10844

[6] A. M. Nair, A. Balan, A. C M, M. Baiju, and T. Davis, “Towards secure ai: Detection of prompt injection attacks with explainability,” in *2025 2nd International Conference on Trends in Engineering Systems and Technologies (ICTEST)*, 2025.

[7] M. Alamsabi, M. Tchuindjang, and S. Brohi, “Embedding-based detection of indirect prompt injection attacks in large language models using semantic context analysis,” *Algorithms*, vol. 19, no. 1, 2026. [Online]. Available: https://www.mdpi.com/1999-4893/19/1/92

[8] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou, “Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers,” 2020. [Online]. Available: https://arxiv.org/abs/2002.10957

[9] A. Kinasih, A. Handayani, J. Ardiansah, and N. Damanhuri, “Comparative analysis of decision tree and random forest classifiers for structured data classification in machine learning,” *Science in Information Technology Letters*, vol. 5, pp. 13–24, 11 2024.

[10] A. Šarčević, D. Pintar, M. Vranić, and A. Krajna, “Cybersecurity knowledge extraction using xai,” *Applied Sciences*, vol. 12, no. 17, 2022. [Online]. Available: https://www.mdpi.com/2076-3417/12/17/8669

[11] S. M. Lundberg and S.-I. Lee, “A unified approach to interpreting model predictions,” in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.

[12] S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S.-I. Lee, “From local explanations to global understanding with explainable AI for trees,” *Nature Machine Intelligence*, vol. 2, pp. 56–67, 2020. [Online]. Available: https://www.nature.com/articles/s42256-019-0138-9

[13] D. Milush, “Shieldlm prompt injection dataset,” https://huggingface.co/datasets/dmilush/shieldlm-prompt-injection, 2026.

[14] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. Rush, “Transformers: State-of-the-art natural language processing,” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Q. Liu and D. Schlangen, Eds. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45.

[15] N. Reimers and I. Gurevych, “Sentence-bert: Sentence embeddings using siamese bert-networks,” 2019. [Online]. Available: https://arxiv.org/abs/1908.10084

[16] L. Breiman, “Random forests,” *Machine Learning*, 2001. [Online]. Available: https://link.springer.com/article/10.1023/a:1010933404324

[17] N. Altman and M. Krzywinski, “Ensemble methods: bagging and random forests.” *Nature Methods*, 2017. [Online]. Available: https://doi.org/10.1038/nmeth.4438

[18] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, “Classification and regression trees (wadsworth, belmont, ca),” *ISBN-13*, pp. 978–0 412 048 418, 1984.

[19] A. M. Salih, Z. Raisi-Estabragh, I. B. Galazzo, P. Radeva, S. E. Petersen, K. Lekadir, and G. Menegaz, “A perspective on explainable artificial intelligence methods: Shap and lime,” *Advanced Intelligent Systems*, vol. 7, no. 1, 2024. [Online]. Available: http://dx.doi.org/10.1002/aisy.202400304

[20] Y. Jilei, “Fast treeshap: Accelerating shap value computation for trees,” 2022. [Online]. Available: https://arxiv.org/abs/2109.09847

[21] R. Mitchell, E. Frank, and G. Holmes, “Gputreeshap: Massively parallel exact calculation of shap scores for tree ensembles,” 2022. [Online]. Available: https://arxiv.org/abs/2010.13972

[22] D. Ivry and O. Nahum, “Sentinel: Sota model to protect against prompt injections,” 2025. [Online]. Available: https://arxiv.org/abs/2506.05446