

Научная статья

УДК 004.724.4, 004.8: 004.056.5

<https://doi.org/10.37493/2307-910X.2026.2.2>

Исследование методов машинного обучения для обнаружения SQL-инъекций

Иван Валерьевич Азаров¹, Наталия Владимировна Кононова², Данила Эдуардович Суренков³,
Глеб Алексеевич Семенов^{4*}, Олег Арменович Санамян⁵

^{1, 3, 4, 5} Северо-Кавказский федеральный университет (д. 1, ул. Пушкина, Ставрополь, 355017, Россия)

² МИРЭА - Российский технологический университет, филиал в г. Ставрополь (д. 8а, пр. Кулакова, Ставрополь, 355035, Россия)

¹ iazarov@ncfu.ru

² knv_fm@mail.ru; <https://orcid.org/0000-0002-8988-0694>

³ dsurienkov@mail.ru

⁴ glebsemenov2003@gmail.com; <https://orcid.org/0009-0004-1850-9021>

⁵ s.oleg.s2003@gmail.com; <https://orcid.org/0009-0002-3667-2901>

* Автор, ответственный за переписку

Аннотация. Введение. В статье рассматривается задача обнаружения SQL-инъекций в HTTP-запросах с использованием методов машинного обучения. Модели обучались на датасете, содержащем более 120 000 HTTP-запросов, разделённых на легитимные и содержащие SQL-инъекции. Каждый запрос формирован признаковым описанием, его размерность уменьшена, для этого использовался метод главных компонент. Число компонент, с которыми проводилось исследование каждой модели и находилось в диапазоне от 2 до 14. **Материалы и методы.** В исследование участвуют различные модели: Probabilistic Neural Network(PNN), Naive Bayes, Multilayer Perceptron с алгоритмом RProp, Fuzzy Rule, k-Nearest Neighbors(KNN), Random Forest, Decision Tree и Gradient Boosted Trees. Оценка качества выполняется по метрикам полноты и точности выявления атак. Оценка переобучения выполняется по метрику ROC-AUC. **Результаты и обсуждение.** Проведено исследование эффективности различных моделей. Среди выбранных моделей по метрикам выделяется модель Probabilistic Neural Network. При 3 главных компонентах значение полноты составляет 0,996, а точность выявления атак равна 0,81. **Заключение.** Полученный результат, демонстрирует что модель обнаруживает почти все SQL-инъекции, но часть обычных запросов определяются как вредоносные.

Ключевые слова: SQL-инъекция, машинное обучение, классификация, PCA, KNIME, обнаружение атак, кибербезопасность

Для цитирования: Азаров И. В., Кононова Н. В., Суренков Д. Э., Семенов Г. А., Санамян О. А. Исследование методов машинного обучения для обнаружения SQL-инъекций // Современная наука и инновации. 2026. № 2. С. 18–32. <https://doi.org/10.37493/2307-910X.2026.2.2>

Конфликт интересов: авторы заявляют об отсутствии конфликта интересов

Статья поступила в редакцию 01.04.2026;
одобрена после рецензирования 21.05.2026;
принята к публикации 21.06.2026.

Investigation of machine learning methods for detecting SQL injections

Ivan V. Azarov¹, Natalia V. Kononova², Danila E. Surenkov³, Gleb A. Semenov^{4*}, Oleg A. Sanamyan⁵

^{1, 3, 4, 5} North-Caucasus Federal University (1, Pushkin St., Stavropol, 355017, Russia)

² MIREA - Russian Technological University, Branch in Stavropol (8a, Kulakova Ave., Stavropol, 355035, Russia)

¹ iazarov@ncfu.ru

² knv_fm@mail.ru, <https://orcid.org/0000-0002-8988-0694>

³ dsurienkov@mail.ru

⁴ glebsemenov2003@gmail.com, <https://orcid.org/0009-0004-1850-9021>

⁵ s.oleg.s2003@gmail.com, <https://orcid.org/0009-0002-3667-2901>

*Corresponding author

Abstract. Introduction. The paper addresses the task of detecting SQL injections in HTTP requests using machine learning methods. The models were trained on a dataset containing over 120,000 HTTP requests, classified into legitimate requests and those containing SQL injections. Each request was represented by a set of features, the dimensionality of which was reduced using the Principal Component Analysis method. The number of components used for the study of each model ranged from 2 to 14. **Materials and methods.** The study involves various models: Probabilistic Neural Network (PNN), Naive Bayes, Multilayer Perceptron with the RProp algorithm, Fuzzy Rule, k-Nearest Neighbors (KNN), Random Forest, Decision Tree, and Gradient Boosted Trees. The quality assessment is performed using recall and precision metrics for attack detection. Overfitting is evaluated using the ROC-AUC metric. **Results and discussion.** The effectiveness of different models was studied. Among the selected models, the Probabilistic Neural Network stands out in terms of performance metrics. With 3 principal components, the recall value is 0.996, and the attack detection precision is 0.81. **Conclusion.** The obtained result demonstrates that the model detects nearly all SQL injections, but classifies some legitimate requests as malicious.

Keywords: SQL injection, machine learning, classification, PCA, KNIME, attack detection, cybersecurity

For citation: Azarov IV, Kononova NV, Surenkov DE, Semenov GA, Sanamyan OA. Investigation of machine learning methods for detecting SQL injections. *Modern Science and Innovations*. 2026;(2):18-32. <https://doi.org/10.37493/2307-910X.2026.2.2>

Conflict of interest: the authors declare no conflicts of interests.

The article was submitted 01.04.2026;

approved after reviewing 21.05.2026;

accepted for publication 21.06.2026.

Introduction. Web applications have long been a common part of information systems. They provide access to databases, personal accounts, internal organizational services, and various management interfaces. For this reason, attacks on web applications remain one of the most common areas of cybersecurity.

SQL injections occupy a special place among such attacks. They involve injecting fragments of SQL code into query parameters or input fields, altering the logic of database access. As a result, an attacker can extract data from tables, bypass authentication, modify database contents, or disrupt application operation. Despite advances in security, SQL injections remain a practical threat, as vulnerabilities of this type continue to arise in real web systems.

SQL injection (SQLi) is one of the most common and dangerous types of cyberattacks. In this attack, an attacker injects malicious SQL code into database queries to gain unauthorized access, modify, or delete data. The primary reason for the success of such attacks is improper processing of user input in web applications.

SQL injection detection traditionally relies on signature rules, query filtering, and analysis of suspicious constructs. However, these approaches don't always yield consistent results, especially when the attack disguises itself as a legitimate query or contains a modified structure. This makes the task of automatically recognizing malicious queries more complex and requires methods capable of detecting not only known patterns but also hidden signs of attack.

Machine learning methods are of particular interest in this regard. Their use allows for the analysis of large numbers of HTTP requests, consideration of a set of features, and the construction of models capable of distinguishing between safe and malicious traffic. However, the quality of detection depends largely on the choice of algorithm, the composition of features, and the method of data preprocessing.

The first mentions of SQL injections date back to the late 1990s, when databases began to be actively used in web development. In 1998, Jeff Forristal described vulnerabilities associated with dynamic SQL queries. Subsequently, in the early 2000s, attacks of this type became widespread. One of the most notorious incidents was the "SQL Slammer" worm attack in 2002, which exploited a vulnerability in Microsoft SQL Server and caused widespread disruptions to network infrastructure.

Over time, the main types of SQL injections have emerged:

- classic SQL injection, based on the introduction of malicious code through input forms (login, password, search fields);
- Blind SQL injection (Blind SQLi), in which an attacker extracts information indirectly by analyzing system behavior or response time;
- A SQL injection vulnerability based on the UNION operator that allows combining legitimate and malicious queries.

The SQL injection mechanism is based on modifying the structure of the original SQL query by injecting additional code via user input. As a result, the application generates a modified query, which is executed by the database management system. A generalized attack flow diagram is shown in Figure 1.

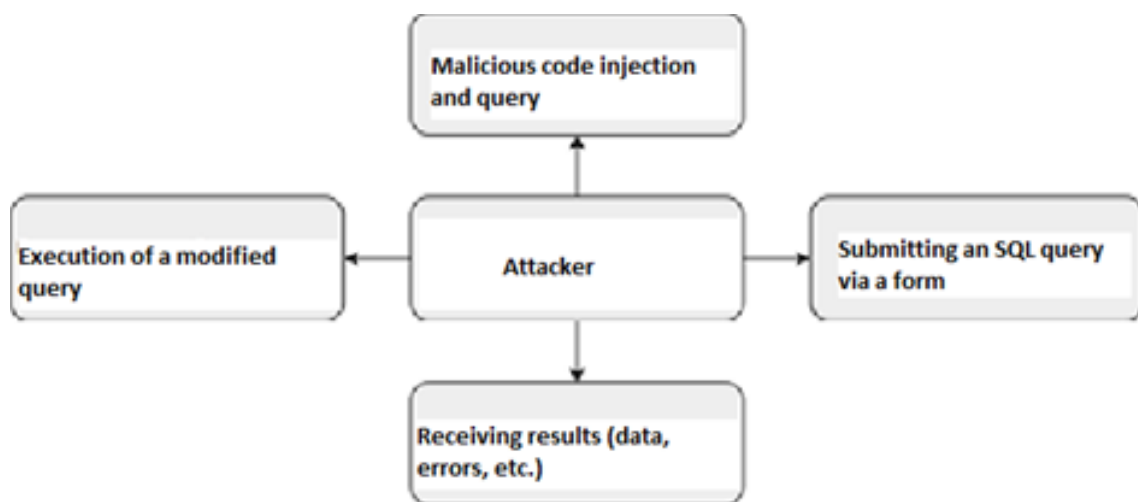


Figure 1. Summary scheme

This process involves the following sequential steps: sending a malicious request through the user interface, injecting SQL code into the request structure, executing the modified request in the database, and analyzing the system response (data or side effects).

Materials and methods. Research on SQL injection detection using machine learning can be roughly divided into several areas. One group of studies uses classical classification algorithms based on pre-extracted SQL query features. These include decision trees, logistic regression, and naive Bayes classifiers.

Other studies employ ensemble methods, particularly Random Forest and Gradient Boosted Trees, which improve classification performance by combining multiple base models. A separate area of research involves neural network approaches, which treat an SQL query as a sequence of characters; such studies utilize CNNs and RNNs. An n-gram approach is also used for query analysis, allowing for the identification of recurring patterns in input data.

This article compares machine learning models used to detect SQL injections in HTTP requests. The impact of feature dimensionality reduction on classification quality is also assessed.

The experiments are conducted in KNIME 5.4.1. This platform is used for data preparation, model training, and metric calculation. Computations are performed on a workstation running Windows 11 Pro with an Intel Xeon E5-2660 v3 processor and 32 GB of RAM.

The study uses a Web Network dataset containing 120,000 HTTP requests. Each request has a class label, so the task is considered a binary classification problem.

The set includes quantitative and categorical features that allow for a formalized description of the query. The main characteristics used in the study are presented in Table 1.

Table 1. Dataset characteristics

Name	Data type	Meaning
method	Categorical	HTTP request method (e.g. GET, POST)
path	Categorical	Request path (URL without domain)
single_q	Quantitative	The number of single quotes (') in the query
double_q	Quantitative	The number of double quotes (") in the query
dashes	Quantitative	Number of hyphens (-) in the query
braces	Quantitative	The number of curly braces ({}) in the query
spaces	Quantitative	Number of spaces in the query
percentages	Quantitative	Number of percent signs (%) in the query
semicolons	Quantitative	The number of semicolons (;) in the query
angle_brackets	Quantitative	The number of angle brackets (<>) in the query
special_chars	Quantitative	Number of special characters in the query
path_length	Quantitative	Request path length (number of characters)
body_length	Quantitative	Request body length (number of characters)
badwords_count	Quantitative	Number of prohibited words in a query
class	Binary	Request class (1 – suspicious, 0 – normal)

Table 1 shows that the dataset used contains both categorical and quantitative features reflecting the structure and substantive characteristics of an HTTP request. The categorical fields method and path define the general context of the resource request, that is, the method used to send the request and its address part. The remaining features are quantitative in nature and allow us to formalize those query elements that are most often informative when detecting SQL injection attacks.

When evaluating the results, we take into account class imbalance: there are more normal queries in the set than malicious ones. Under these conditions, accuracy alone is insufficient, as even a high level does not guarantee reliable attack detection. Therefore, Recall, Precision, and AUC are used in this study.

Recall indicates the proportion of malicious queries identified correctly. This metric is particularly important for this task, as missing SQL injections is a more serious problem than false positives. Precision reflects the proportion of queries classified by the model as malicious that are actually attacks. The AUC metric is used as an additional measure of class separation quality.

The study includes Decision Tree, Random Forest, Gradient Boosted Trees, Tree Ensemble, Naive Bayes, Fuzzy Rule, Probabilistic Neural Network, RProp MLP, and k-Nearest Neighbors. This set of algorithms allows for the comparison of models with different logics on the same set of features and with the same evaluation scheme.

IN research are being considered nine models machine Training methods : Decision Tree, Random Forest, Gradient Boosted Trees, Tree Ensemble, Naive Bayes, Fuzzy Rule, Probabilistic Neural Network (PNN), RProp MLP , and k-Nearest Neighbors (KNN). The sample includes models with different operating principles, allowing them to be compared within the same task.

The Decision Tree is used as the baseline comparison model. Random Forest, Tree Ensemble, and Gradient Boosted Trees represent ensemble methods aimed at achieving more robust

results. Naive Bayes is considered a probabilistic algorithm, and Fuzzy Rule is a fuzzy-rule-based model. The neural network approach is represented by the PNN and RProp MLP models. The k-Nearest Neighbors algorithm is included in the study as a method based on assessing the proximity of objects in a feature space.

This set of models makes it possible to compare different approaches to classification on the same set of features and with the same evaluation scheme.

To reduce the dimensionality of the feature space, principal component analysis (PCA) is used. This transformation reduces the number of features by eliminating correlations between them while preserving the core information. The study evaluates the impact of the number of components on classification quality for various models. Recall and Precision are used as metrics.

To reduce overfitting, parameter selection and feature dimensionality reduction using PCA are used. This reduces the number of redundant features and makes models less sensitive to the characteristics of the training set.

Additionally, the AUC metric is taken into account, which indicates how confidently the model distinguishes between two classes. High AUC values correspond to more reliable separation of normal and malicious queries.

Naive Bayes and Random Forest models performed the most consistently in the study. These models were found to be less prone to overfitting and produced more consistent results.

Results and discussion. To reduce the dimensionality of the feature space, principal component analysis (PCA) is used. This transformation reduces the number of features by eliminating correlations between them while preserving the core information. The study evaluates the impact of the number of components on classification quality for various models. Recall and Precision are used as metrics.

Figure 2 shows the results for the Random Forest model. As the number of components increases, classification performance gradually improves and then plateaus. This indicates that the model performs effectively even with an expanding feature space and is only weakly sensitive to redundant features.

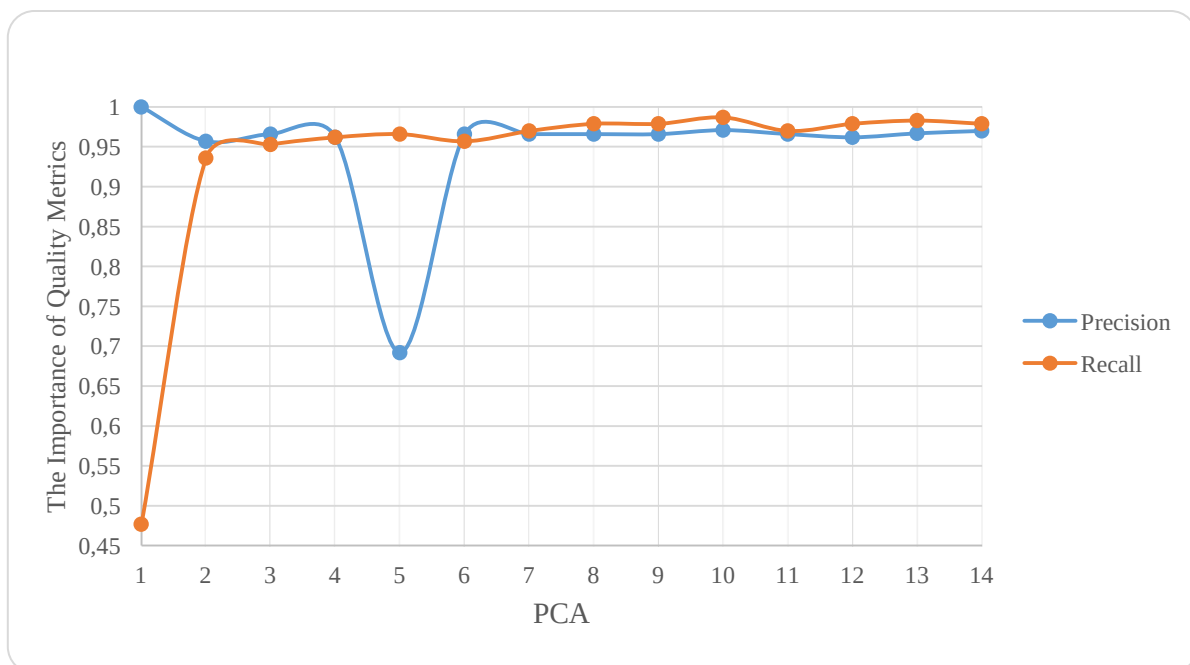


Figure 2. Comparison of quality metrics based on the number of PCA components in the Random Forest model

Figure 3 shows the results for the Decision Tree. Maximum metric values are achieved with a moderate number of components, after which quality begins to decline. This behavior is due to overfitting as feature size increases.

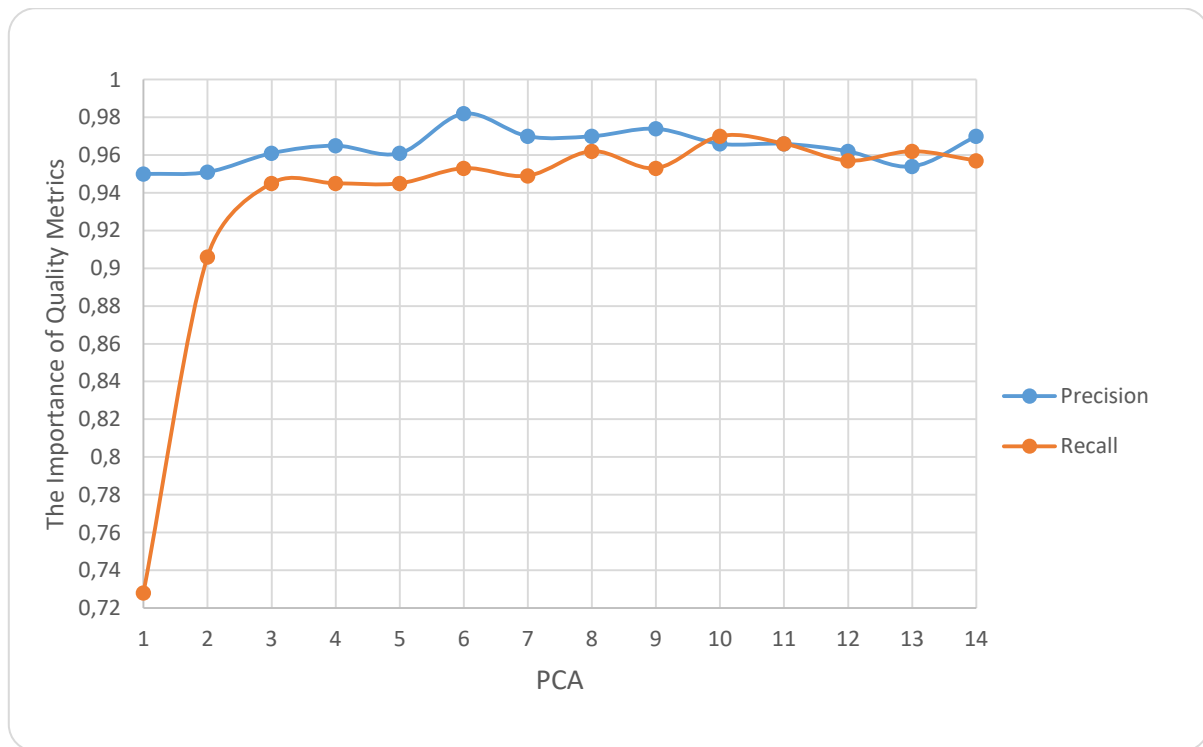


Figure 3. Comparison of quality metrics from the number of PCA components of the Decision Tree model

Figure 4 shows the results for the Tree Ensemble model. The relationship is similar to that of Random Forest: as the number of components increases, the quality improves and then stabilizes. This confirms the robustness of ensemble methods to noise features.

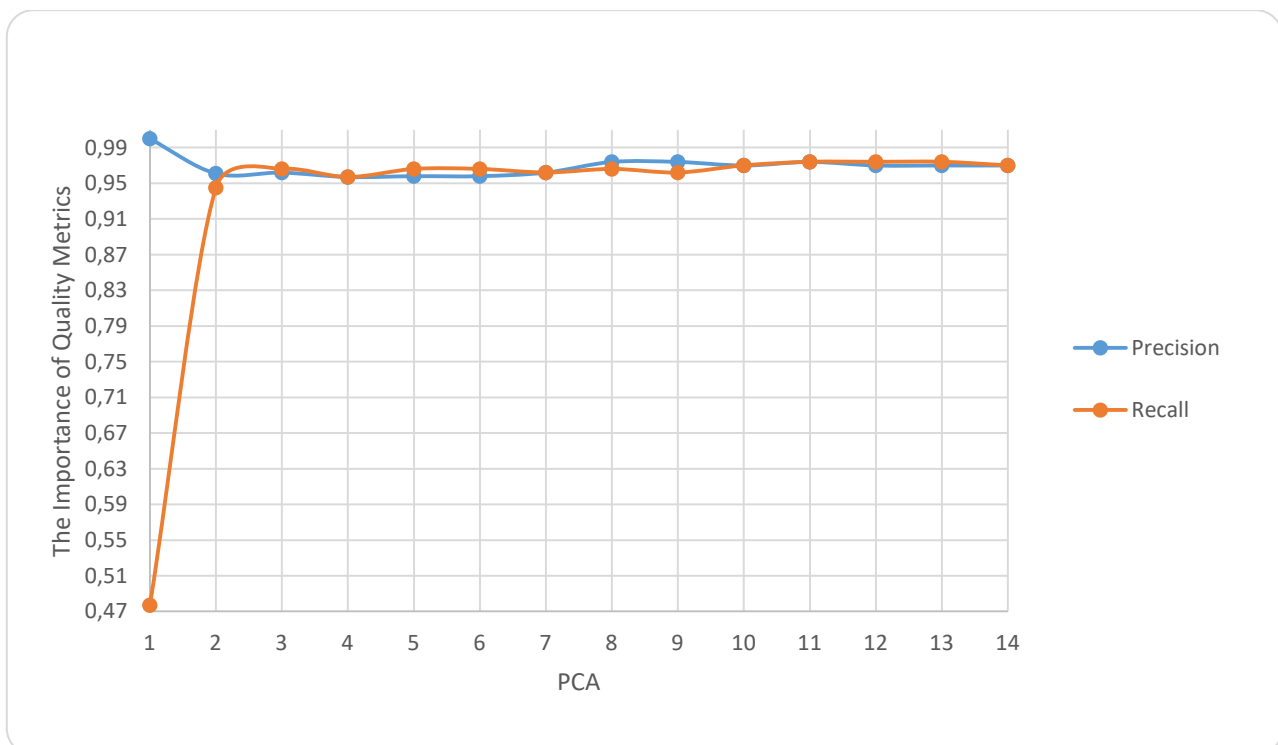


Figure 4. Comparison of quality metrics from the number of PCA components of the Tree Ensemble model

Figure 5 shows the results for Gradient Boosted Trees. Quality increases up to a certain PCA value, after which no significant changes are observed. Thus, there is an optimal number of components for this model.

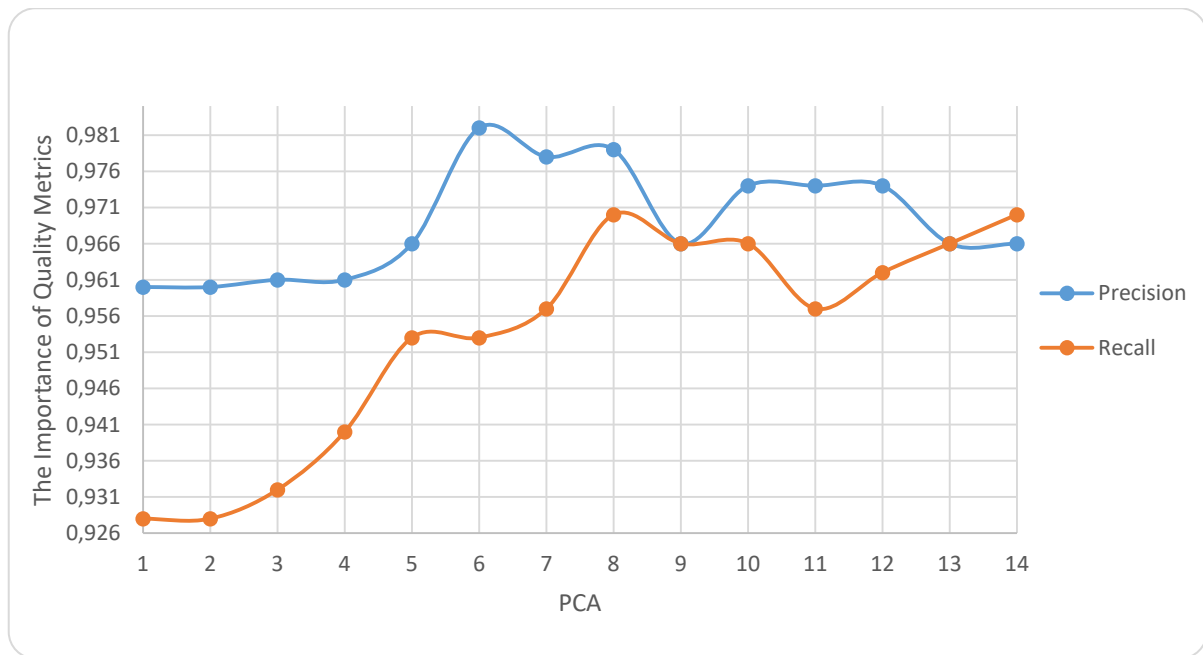


Figure 5. Comparison of quality metrics from the number of PCA components of the Gradient Boosted Trees model

Figure 6 shows the results for the k-nearest neighbors algorithm. The best results are achieved with a moderate number of components. As the dimensionality increases or decreases, the quality decreases, due to the algorithm's sensitivity to the structure of the feature space.

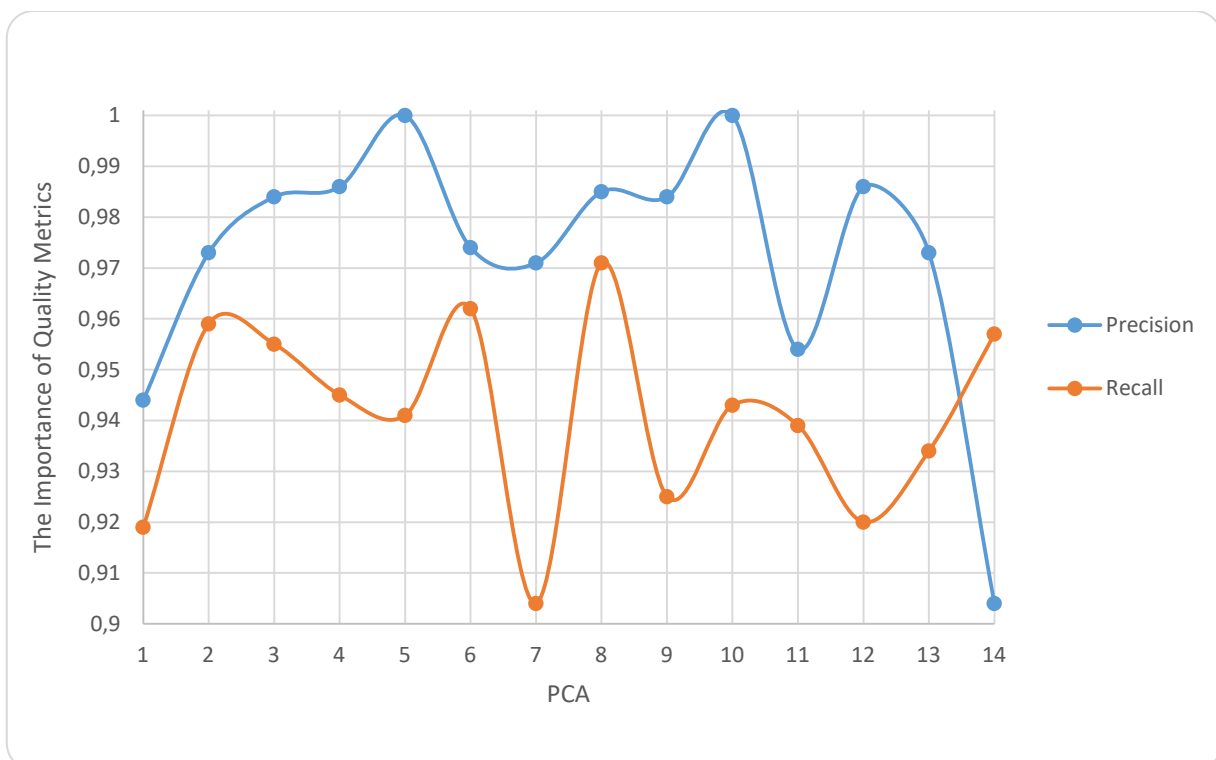


Figure 6. Comparison of performance metrics depending on the number of PCA components for the k- Nearest Neighbors model

Figure 7 shows the results for Naive Bayes. Maximum values are achieved with a small number of components. This is because the model performs worse with correlated features and benefits from their reduction.

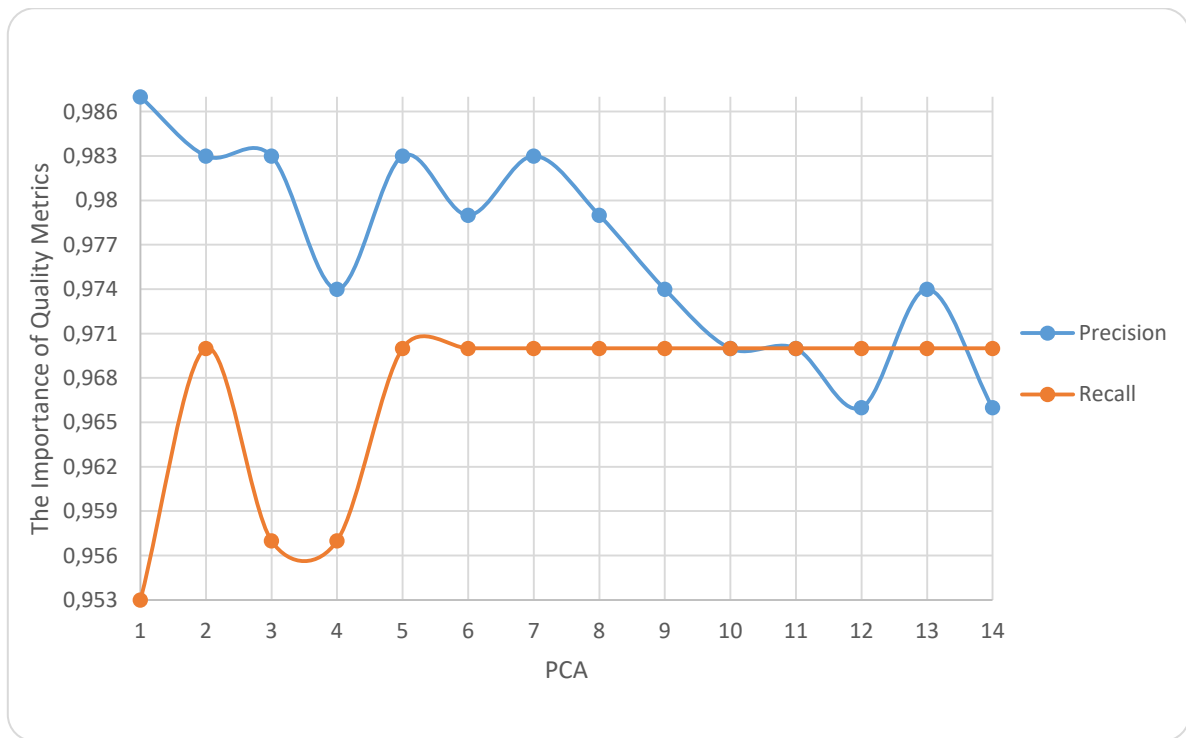


Figure 7. Comparison of quality metrics from the number of PCA components of the Naive Bayes model

Figure 8 shows the results for the Fuzzy Rule model. Quality peaks at a moderate number of components and does not change significantly thereafter.

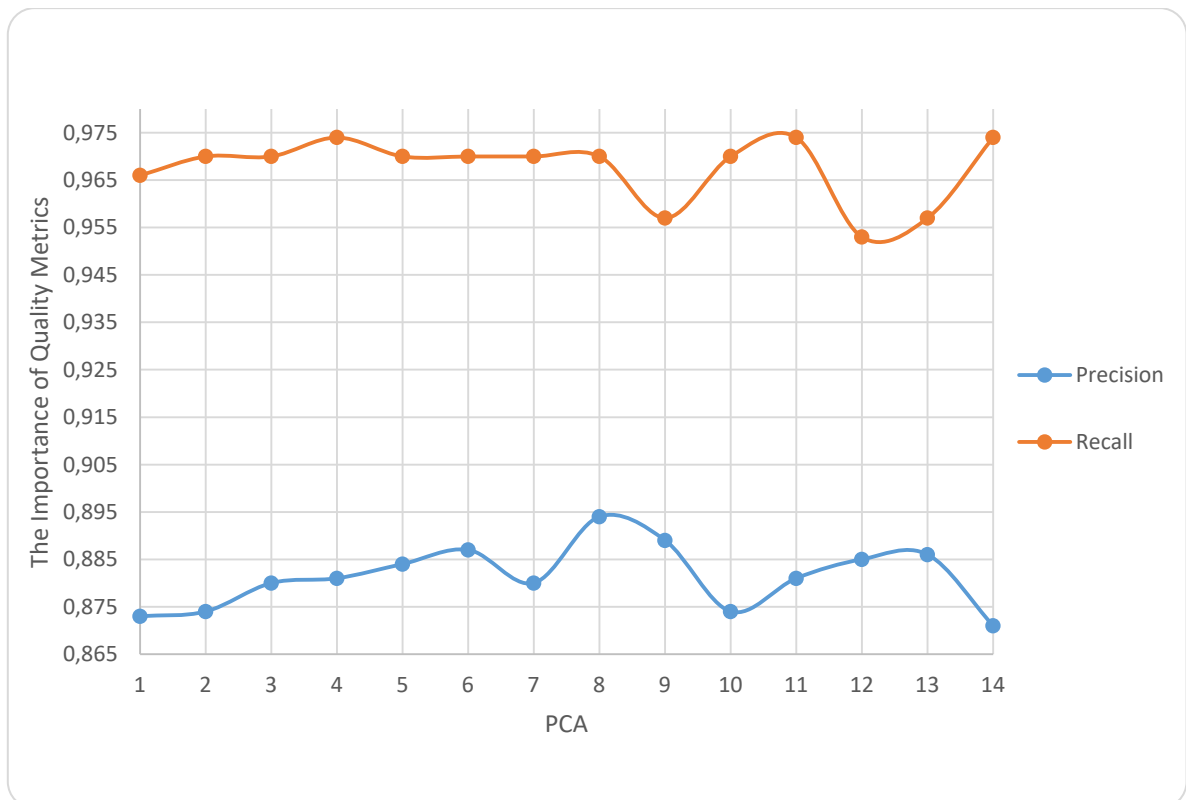


Figure 8. Comparison of quality metrics from the number of PCA components of the Fuzzy Rule model

Figure 9 shows the results for a probabilistic neural network (PNN). With a small number of components, high Recall values are achieved, but Precision remains lower, indicating an increase in false positives.

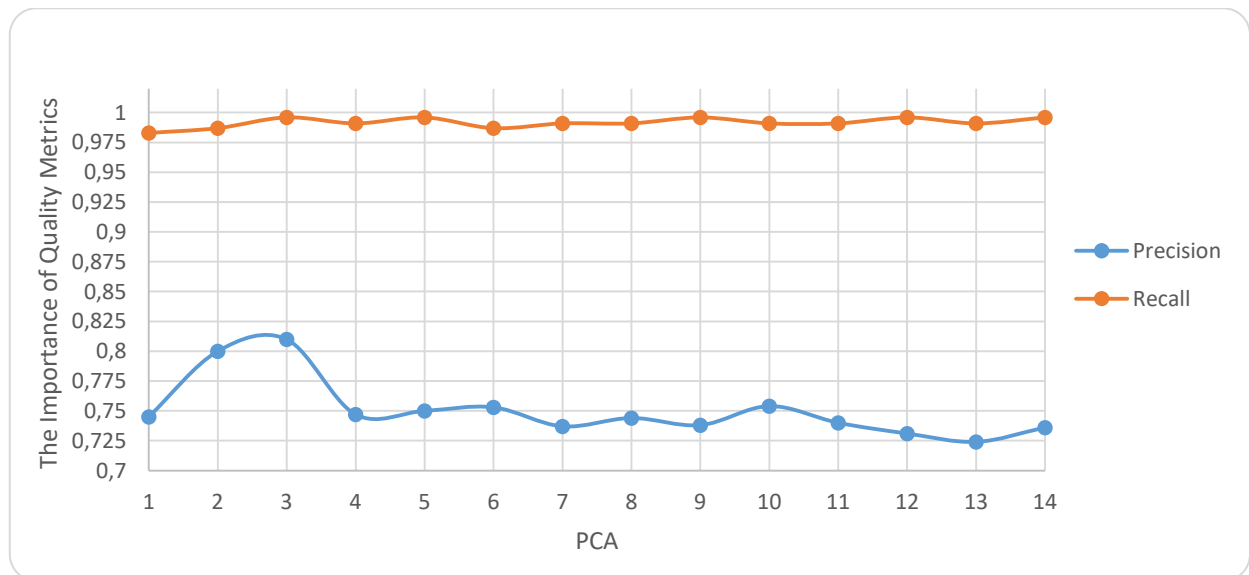


Figure 9. Comparison of quality metrics based on the number of PCA components in the *Probabilistic Neural Network* model

Figure 10 shows the results for the RProp MLP model. Classification performance increases with the number of components up to a certain value, after which the changes become insignificant. This indicates that the neural network model requires a sufficient number of informative features.

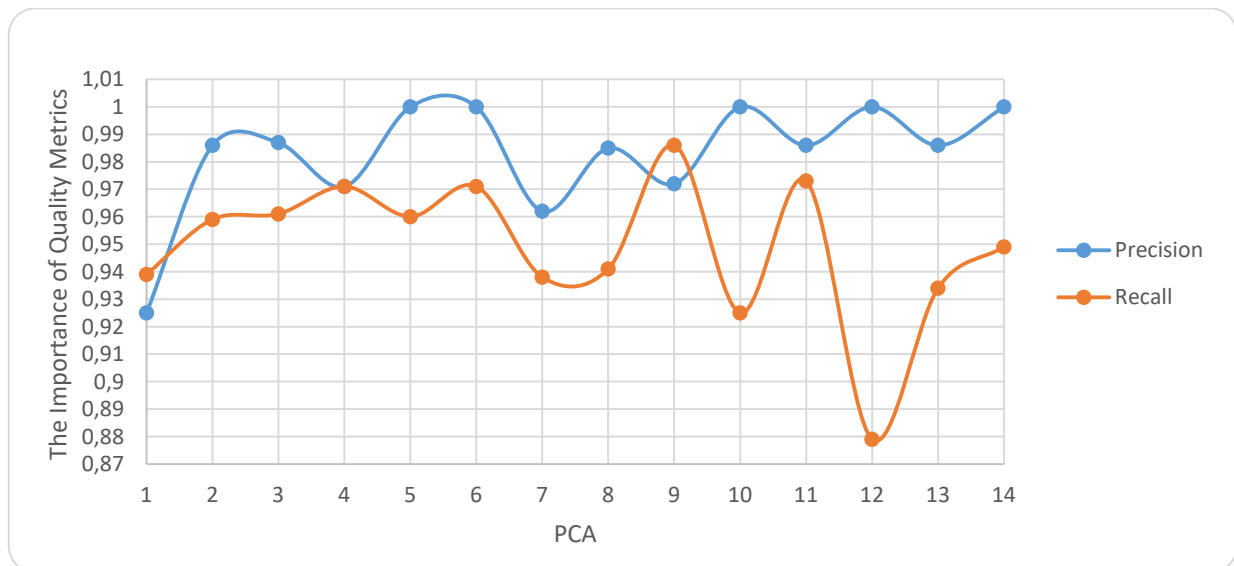


Figure 10. Comparison of quality metrics from the number of PCA components of the RProp MLP model

Thus, the conducted analysis shows that the optimal number of PCA components is a model-dependent parameter.

The summary results are shown in Table 2.

Table 2. Optimal values of the number of PCA components

Model	PCA	Recall	Precision
PNN	3	0.996	0.81
Naive Bayes	2	0.983	0.97
RProp MLP	8	0.985	0.941
Fuzzy Rule	8	0.97	0.894
K-Nearest Neighbors (KNN)	8	0.971	0.985

Model	PCA	Recall	Precision
Random Forest	14	0.979	0.97
Decision Tree	10	0.97	0.966
Decision Tree	11	0.974	0.974
Gradient Boosted Trees	8	0.97	0.979

For the most effective models, the impact of key hyperparameters on classification quality is considered. Evaluation is performed using the Recall and Precision metrics, using previously selected PCA values.

Figure 11 shows the results for the Naive Bayes model with varying default probability. Small default probability values yield the best metric values. As the default probability increases, accuracy decreases, resulting in an increased number of false positives.

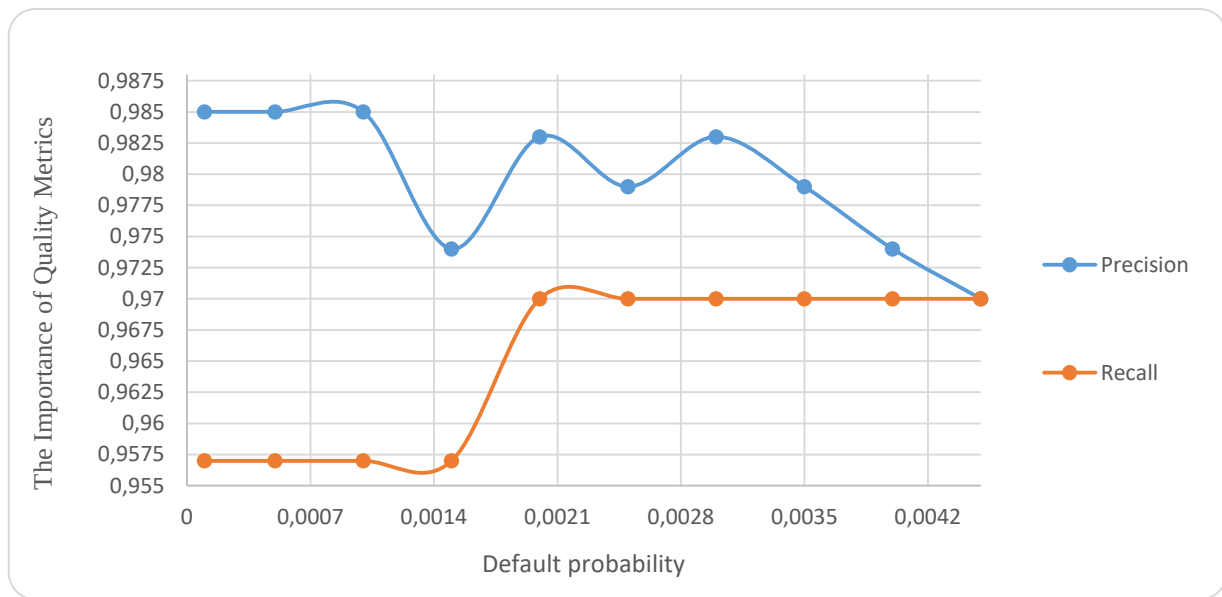


Figure 11. Dependence of quality metrics on the “Default probability” parameter for the Naive Bayes model

Figure 12 shows the results for the Random Forest model with varying tree depth. Classification quality improves with increasing depth, but after a certain value, the changes become insignificant. This indicates that optimal model complexity has been achieved.

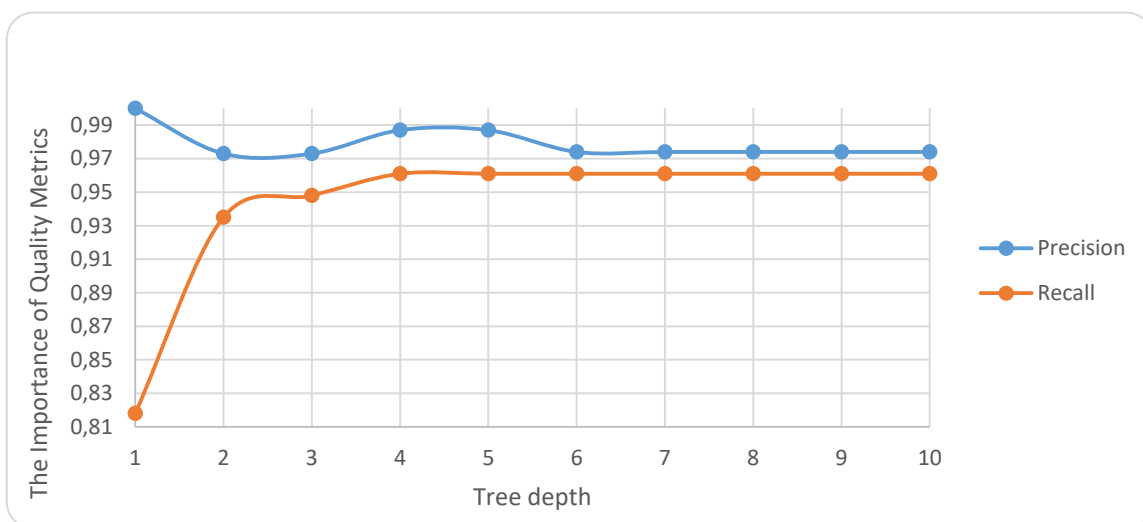


Figure 12. Dependency metrics quality from "Limit number of levels (tree depth)" for Random Forest/

Figure 13 shows the results for the KNN algorithm with varying numbers of neighbors (k). The best metric values are achieved with small values of this parameter. As the number of neighbors increases, classification quality decreases, due to the smoothing of class boundaries and a loss of sensitivity to local data structure.

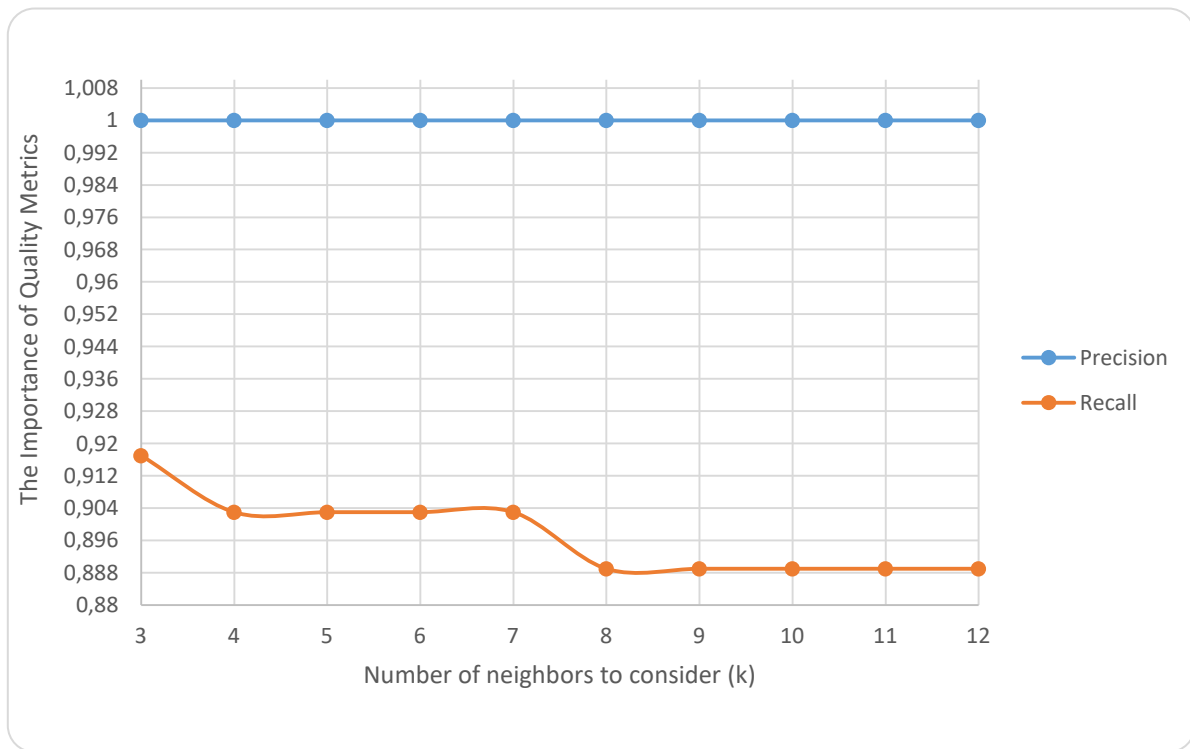


Figure 13. Dependency metrics quality from “Number of neighbors to consider (k)” of the KNN model

Figure 14 shows the results for Gradient Boosted Trees as tree depth varies. As depth increases, classification performance improves up to a certain point, after which changes become insignificant. This indicates that an optimal model complexity has been reached, at which point further complexity does not improve the results.

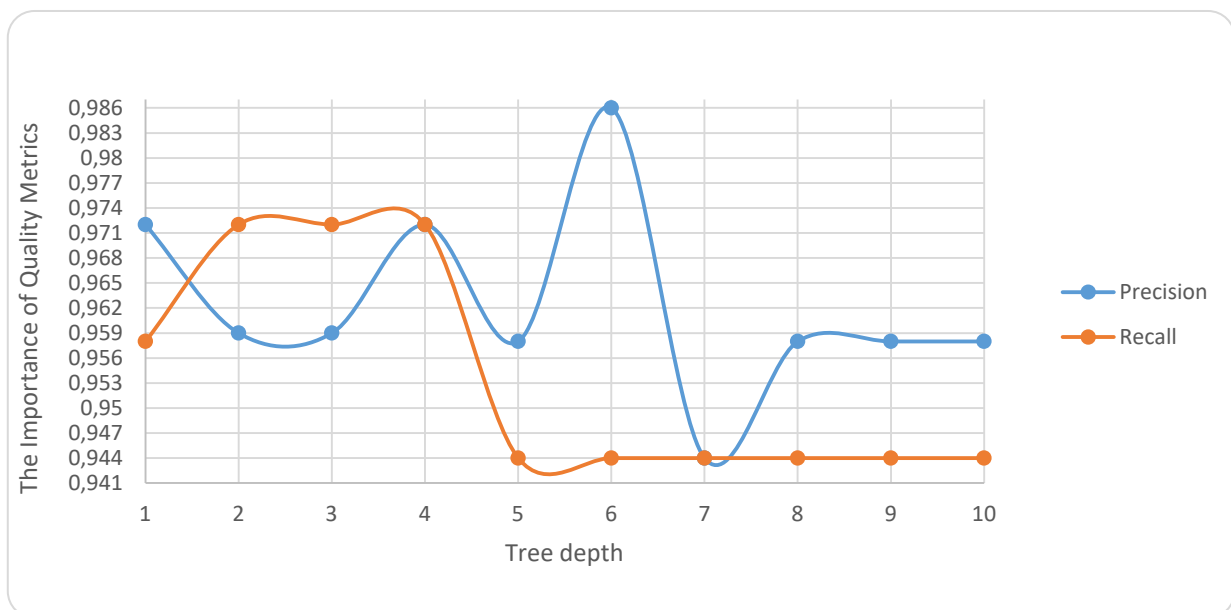


Figure 14. Dependency metrics quality from “Limit number of levels (tree depth)” of the Gradient Boosted Trees model

Figure 15 shows the results for the Tree Ensemble model. The pattern of the dependence is similar to other ensemble methods: there is an optimal depth value that achieves the best performance. Further increases in the parameter do not yield significant improvement.

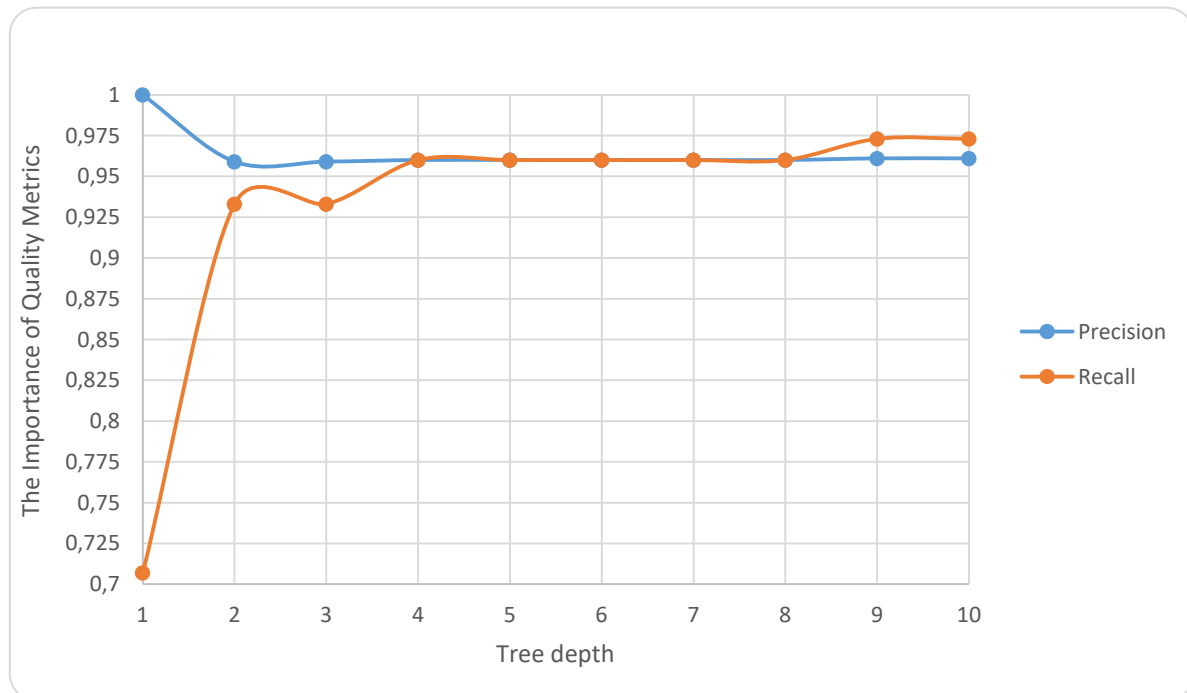


Figure 15. Dependency metrics quality from the “Limit number of levels (tree depth)” model Tree Ensemble

For each model, a range of hyperparameter values is identified that achieves the best Recall and Precision values. Further increases in the parameters do not improve classification quality and, in some cases, even decrease it.

The final metric values for the most effective models are shown in Table 3.

Table 3. Final values of quality metrics for the best models

Model	Recall	Precision	AUC
Naive Bayes	0.97	0.983	0.973
Random Forest	0.987	0.961	0.975
KNN	0.917	1	0.958
Gradient Boosted Trees	0.972	0.959	0.814
Tree Ensemble	0.973	0.961	0.968

According to Table 3, ensemble methods and the Naive Bayes algorithm demonstrate the best metric values. Random Forest and Tree Ensemble models provide the most balanced Recall and Precision metrics. The KNN algorithm is characterized by high accuracy but is inferior in recall, indicating that some malicious queries are missed.

Conclusion. The paper examined machine learning methods for detecting SQL injections in HTTP requests on a dataset of over 120,000 records. The analysis showed that classification quality depended on both the chosen algorithm and the number of principal components used after feature dimensionality reduction.

The most balanced results were obtained for the Random Forest, Tree Ensemble, and Naive Bayes models. For Random Forest, the Recall, Precision, and AUC values were 0.987, 0.961, and 0.975, respectively. For Tree Ensemble, these values were 0.973, 0.961, and 0.968, and for Naive Bayes, they were 0.970, 0.983, and 0.973. These results showed that ensemble methods and the probabilistic approach provided the most stable tradeoff between detection recall and the number of

false positives. Separately, it was found that the k-Nearest Neighbors model achieved a Precision of 1.000, but its Recall value was 0.917, indicating that some malicious queries were missed.

The obtained results confirmed that machine learning methods can be used to detect SQL injections in HTTP requests, but the final classification quality was determined by the choice of model, the number of principal components, and the tuning parameters.

Список источников

1. Ashton K. That "Internet of Things" Thing: In the Real World Things Matter More than Ideas // Rfid Journal. 2009. 22 June. URL: <https://www.rfidjournal.com/expert-views/that-internet-of-things-thing/73881/> (дата обращения: 01.04.2026).
2. Cisco Internet Business Solutions Group. The Internet of Things: How the Next Evolution of the Internet Is Changing Everything. 2011. URL: https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf (дата обращения: 01.04.2026).
3. Воронина В. В., Михеев А. В., Ярушкина Н. Г., Святков К. В. Теория и практика машинного обучения. Ульяновск: УлГТУ, 2017. 290 с.
4. Лапина М. А., Подручный Н. В., Русанов М. А., Бабенко М. Г. Исследование методов машинного обучения для выявления сетевых атак // Труды Института системного программирования РАН. 2025. Т. 37. Вып. 4. Ч. 2. С. 147–174. [https://doi.org/10.15514/ISPRAS-2025-37\(4\)-24](https://doi.org/10.15514/ISPRAS-2025-37(4)-24)
5. Лапина М. А., Мовзалевская В. В., Токмакова М. Е., Бабенко М. Г., Саджид М. Применение технологий машинного обучения для обнаружения веб-атак // Вопросы кибербезопасности. 2024. № 4 (62). С. 92–103. <https://doi.org/10.21681/2311-3456-2024-4-92-103>
6. Лапина М. А., Лукьянов Д. А., Лапин В. Г., Кучеров Н. Н. Исследование методов машинного обучения для обнаружения сайтов-мошенников // Известия ЮФУ. Технические науки. 2025. № 4. С. 250–262. <https://doi.org/10.18522/2311-3103-2025-4-250-262>
7. Лапина М. А., Капшук Н. Р., Русанов М. А., Тимофеева Е. Ф. Обнаружение атак с использованием SQL-инъекций по сетевым журналам с помощью методов машинного обучения // Труды Института системного программирования РАН. 2025. Т. 37. Вып. 5. С. 81–92. [https://doi.org/10.15514/ISPRAS-2025-37\(5\)-6](https://doi.org/10.15514/ISPRAS-2025-37(5)-6)
8. Лапина М. А., Дымуха Р. А., Кучеров Н. Н., Басан Е. С. Исследование методов машинного обучения для обнаружения спуфинг-атак в децентрализованных сетях // Известия ЮФУ. Технические науки. 2025. № 3. С. 16–31. <https://doi.org/10.18522/2311-3103-2025-3-16-31>
9. Лапина М. А., Гриценко Д. Д., Кеньков Е. А., Багаутдинова А. Р., Соломянко А. А. Детектирование атак на отказ в программно-конфигурируемых сетях с использованием методов машинного обучения // Труды Института системного программирования РАН. 2025. Т. 37. Вып. 6. Ч. 4. С. 139–158. [https://doi.org/10.15514/ISPRAS-2025-37\(6\)-55](https://doi.org/10.15514/ISPRAS-2025-37(6)-55)
10. KNIME Analytics Platform [Электронный ресурс]. URL: <https://www.knime.com/knime-analytics-platform> (дата обращения: 01.04.2026).
11. Lashkari A. H. CICFlowmeter-V4.0 (formerly known as ISCXFlowMeter) is a Network Traffic Bi-flow Generator and Analyser for Anomaly Detection [Электронный ресурс]. URL: <https://github.com/ahlashkari/CICFlowMeter> (дата обращения: 01.04.2026).
12. IoT-SDN IDS Dataset [Электронный ресурс]. URL: <https://www.kaggle.com/datasets/hebadhirar/iot-sdn-ids-dataset> (дата обращения: 01.04.2026).
13. Карачанская Е. В., Соседова Н. И. Метод выявления аномалий сетевого трафика, основанный на его самоподобной структуре // Безопасность информационных технологий. 2019. Т. 26. № 1. С. 98–110. <https://doi.org/10.26583/bit.2019.1.10>
14. Jolliffe I. T., Cadima J. Principal component analysis: a review and recent developments // Philosophical Transactions of the Royal Society A. 2016. Vol. 374. No. 2065. Art. 20150202. <https://doi.org/10.1098/rsta.2015.0202>
15. Song Y. Y., Lu Y. Decision tree methods: applications for classification and prediction // Shanghai Archives of Psychiatry. 2015. Vol. 27. No. 2. P. 130–135. <https://doi.org/10.11919/j.issn.1002-0829.215044>
16. Zhang S., Li X., Zong M., Zhu X., Wang R. Efficient kNN classification with different numbers of nearest neighbors // IEEE Transactions on Neural Networks and Learning Systems. 2018. Vol. 29. No. 5. P. 1774–1785. <https://doi.org/10.1109/TNNLS.2017.2673241>

17. Biau G., Scornet E. A random forest guided tour // Test. 2016. Vol. 25. No. 2. P. 197–227. <https://doi.org/10.1007/s11749-016-0481-7>
18. Natekin A., Knoll A. Gradient boosting machines, a tutorial // Frontiers in Neurorobotics. 2013. Vol. 7. Art. 21. <https://doi.org/10.3389/fnbot.2013.00021>
19. Балыбердин А. В., Крылов Г. О. Повышение точности выявления аномалий для систем обнаружения вторжения с помощью ансамблевого обучения // Безопасность информационных технологий. 2025. Т. 32. № 1. С. 153–171. <https://doi.org/10.26583/bit.2025.1.11>
20. Лавров Б. О., Иванов М. А. Интеллектуальное обнаружение аномалий в контейнеризованных приложениях: методы, архитектура и инструменты // Безопасность информационных технологий. 2025. Т. 32. № 4. С. 149–164. <https://doi.org/10.26583/bit.2025.4.11>

References

1. Ashton K. That "Internet of Things" Thing: In the Real World Things Matter More than Ideas. Rfid Journal. 2009. 22 June. Available from: <https://www.rfidjournal.com/expert-views/that-internet-of-things-thing/73881/> [Accessed 1 Apr 2026].
2. Cisco Internet Business Solutions Group. The Internet of Things: How the Next Evolution of the Internet Is Changing Everything. 2011. Available from: https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf [Accessed 1 Apr 2026].
3. Voronina VV, Mikheev AV, Yarushkina NG, Svyatov KV. Theory and practice of machine learning. Ulyanovsk: UISTU, 2017. 290 p. (In Russ.).
4. Lapina MA, Podruchny NV, Rusanov MA, Babenko MG. Research of machine learning methods for detecting network attacks. Proceedings of the Institute for System Programming of the Russian Academy of Sciences. 2025;37(4-2):147-174. (In Russ.). [https://doi.org/10.15514/ISPRAS-2025-37\(4\)-24](https://doi.org/10.15514/ISPRAS-2025-37(4)-24)
5. Lapina MA, Movzalevskaya VV, Tokmakova ME, Babenko MG, Sajid M. Application of machine learning technologies for web attack detection. Cybersecurity Issues. 2024;4(62):92-103. (In Russ.). <https://doi.org/10.21681/2311-3456-2024-4-92-103>
6. Lapina MA, Lukyanov DA, Lapin VG, Kuchеров NN. Research of machine learning methods for detecting fraudulent websites. Izvestiya SFedU. Engineering Sciences. 2025;4:250-262. (In Russ.). <https://doi.org/10.18522/2311-3103-2025-4-250-262>
7. Lapina MA, Kapshuk NR, Rusanov MA, Timofeeva EF. Detection of SQL injection attacks using network logs with machine learning methods. Proceedings of the Institute for System Programming of the Russian Academy of Sciences. 2025;37(5):81-92. (In Russ.). [https://doi.org/10.15514/ISPRAS-2025-37\(5\)-6](https://doi.org/10.15514/ISPRAS-2025-37(5)-6)
8. Lapina MA, Dymukha RA, Kuchеров NN, Basan ES. Research of machine learning methods for detecting spoofing attacks in decentralized networks. Izvestiya SFedU. Engineering Sciences. 2025;3:16-31. (In Russ.). <https://doi.org/10.18522/2311-3103-2025-3-16-31>
9. Lapina MA, Gritsenko DD, Kenkov EA, Bagautdinova AR, Solomyanko AA. Detection of denial of service attacks in software-defined networks using machine learning methods. Proceedings of the Institute for System Programming of the Russian Academy of Sciences. 2025;37(6-4):139-158. (In Russ.). [https://doi.org/10.15514/ISPRAS-2025-37\(6\)-55](https://doi.org/10.15514/ISPRAS-2025-37(6)-55)
10. KNIME Analytics Platform [Internet]. Available from: <https://www.knime.com/knime-analytics-platform> [Accessed 1 Apr 2026].
11. Lashkari AH. CICFlowmeter-V4.0 (formerly known as ISCXFlowMeter) is a Network Traffic Bi-flow Generator and Analyser for Anomaly Detection [Internet]. Available from: <https://github.com/ahlashkari/CICFlowMeter> [Accessed 1 Apr 2026].
12. IoT-SDN IDS Dataset [Internet]. Available from: <https://www.kaggle.com/datasets/hebadhirar/iot-sdn-ids-dataset> [Accessed 1 Apr 2026].
13. Karachanskaya EV, Sosedova NI. A Method for Detecting Network Traffic Anomalies Based on Its Self-Similar Structure. Information Technology Security. 2019;26(1):98-110. (In Russ.). <https://doi.org/10.26583/bit.2019.1.10>
14. Jolliffe IT, Cadima J. Principal component analysis: a review and recent developments. Philosophical Transactions of the Royal Society A. 2016;374(2065):20150202. <https://doi.org/10.1098/rsta.2015.0202>
15. Song YY, Lu Y. Decision tree methods: applications for classification and prediction. Shanghai Archives of Psychiatry. 2015;27(2):130-135. <https://doi.org/10.11919/j.issn.1002-0829.215044>

16. Zhang S, Li X, Zong M, Zhu X, Wang R. Efficient kNN classification with different numbers of nearest neighbors. IEEE Transactions on Neural Networks and Learning Systems. 2018;29(5):1774-1785. <https://doi.org/10.1109/TNNLS.2017.2673241>
17. Biau G, Scornet E. A random forest guided tour. Test. 2016;25(2):197-227. <https://doi.org/10.1007/s11749-016-0481-7>
18. Natekin A, Knoll A. Gradient boosting machines, a tutorial. Frontiers in Neurorobotics. 2013;7:21. <https://doi.org/10.3389/fnbot.2013.00021>
19. Balyberdin AV, Krylov GO. Improving the Accuracy of Anomaly Detection for Intrusion Detection Systems Using Ensemble Learning. Information Technology Security. 2025;32(1):153-171. (In Russ.). <https://doi.org/10.26583/bit.2025.1.11>
20. Lavrov BO, Ivanov MA. Intelligent Anomaly Detection in Containerized Applications: Methods, Architecture and Tools. Information Technology Security. 2025;32(4):149-164. (In Russ.). <https://doi.org/10.26583/bit.2025.4.11>

Информация об авторах

Иван Валерьевич Азаров – доцент, Северо-Кавказский федеральный университет, Ставрополь. Researcher ID: PJA-9864-2026.

Наталья Владимировна Кононова – доцент, филиал РТУ МИРЭА в г. Ставрополь, Ставрополь, Researcher ID: PUE-2199-2026.

Данила Эдуардович Суренков – студент, Северо-Кавказский федеральный университет, Ставрополь.

Глеб Алексеевич Семенов – студент, Северо-Кавказский федеральный университет, Ставрополь, Researcher ID: PII-8176-2026.

Олег Арменович Санамян – студент, Северо-Кавказский федеральный университет, Ставрополь, Researcher ID: PJA-7256-2026.

Вклад авторов: все авторы внесли равный вклад в подготовку публикации.

Information about the authors

Ivan V. Azarov – associate professor, North-Caucasus Federal University, Stavropol, Researcher ID: PJA-9864-2026.

Natalia V. Kononova – associate professor, Branch of RTU MIREA in Stavropol, Stavropol, Researcher ID: PUE-2199-2026.

Danila E. Surenkov – student, North-Caucasus Federal University, Stavropol.

Gleb A. Semenov – student, North-Caucasus Federal University, Stavropol, Researcher ID: PII-8176-2026.

Oleg A. Sanamyan – student, North-Caucasus Federal University, Stavropol, Researcher ID: PJA-7256-2026.

Author contributions: the authors contributed equally to this article.