International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

A Resilient Framework for Detecting and Managing Schema Drift in Evolving Microservice Contracts

Aiswarya Gurram

Submitted: 02/11/2024 **Revised**: 17/12/2024 **Accepted**: 27/12/2024

Abstract: Microservice architectures have allowed organizations to build scalable and independently deployable software systems as a result of their rapid adoption. The ongoing development of services brings about schema drift, whereby modifications in data formatting, API contracts, and service interfaces cause compatibility issues between interdependent elements. The current methods primarily revolve around the reactive validation and failure detection, and have little ability to predict the possibility of contract evolution risks. The study presents a robust model of schema drift detection and control in changing contracts of microservice predictive control based on machine learning. The secondary data composed of 10,000 records of microservice contract evolutions is processed by using data analytics and machine learning with Python. Preprocessing data, feature engineering, and exploratory analysis are conducted to determine key factors of schema drift. The algorithm of Random Forest and Gradient Boosting are used in order to categorize the risks of schema drift as low-risk, medium-risk, and high-risk. The experimental assessment shows that the accuracy and F1-score of the Random Forest are better, with 91.40% and 91.31%, respectively, which is better than that of the Gradient Boosting. The analysis of feature importance demonstrates that the failure count, compatibility status, and version difference are significant factors causing the occurrence of schema drift. The framework offered is a proactive solution to enhance the success of contracts, minimize compatibility breakdowns, and enable ongoing transformation of distributed microservice systems.

Keywords: *Schema Drift, Microservices, Contract Evolution, Machine Learning, API Compatibility, Predictive Analytics, Service Reliability.*

I. INTRODUCTION

Background

Fast pace of cloud applications has led to increased adoption of microservice systems large software systems being broken down into independently deployable and scalable services. Microservices interoperate via established agreements, such as API curriculum, data format and service interfaces, to provide dependable information transfer among disparate units [1]. Constant development of services and frequent deployments, as well as development of services independently of applications, tend to cause schema drift, in which service contracts progressively become inconsistent with applications that rely on them. Schema drift may go unnoticed until the point of creating compatibility errors, misinterpreting the data, service failure, and higher complexity to its maintenance [2]. The customary contract management practices are predominantly based on responding to changes and thus offer limited forecasting potentials of risks in the future. In order to ensure compatibility, reliability and availability in dynamic microservice ecosystems, schema evolution strategies are needed.

Problem Statement

Microservice systems are often prone to schema drift as a result of continual updates, deployment as a service, and evolve dependencies of services. The current contract management methodologies fail to identify the inconsistency of the schema at an early stage and forecast possible failures [3]. The consequence of this limitation is incompatibility of services, risks in operation and data discrepancies

Independent Researcher, USA.

and the high maintenance expenses in changing distributed architecture.

Research Motivation

Microservices are critical to modern industries like finance, healthcare, e-commerce, and cloud platforms where they need to be constantly available and have reliable communication. Uncontrolled schema drift may have an adverse effect on business processes by making APIs break, having transaction errors, and service disruptions [4]. Thus, it is necessary to come up with smarter mechanisms of detecting and controlling schema changes.

Aim and Objectives

This study aim is to create a framework of identifying; forecasting and addressing schema drift in dynamic microservice contracts, to provide improved service compatibility, reliability and ongoing culture change.

Objectives

- ***To study current methods of evolving microservice contracts and determine the issues with contract schema drift management.***
- ***To explore the effects of schema drift on compatibility of the service, reliable communication and performance of the distributed system.***
- ***To come up with a smart structure which combines schema drift detection, automated contract validation and risk prediction by machine learning.***
- ***To test the suggested framework based on the performance, reliability, and predictive analytics performance measures to determine its effectiveness.***

Novel Contribution

➢ The research adds a holistic representation of how to manage schema drift across time in changing microservice architecture.

➢ In contrast to the conventional methods which predominantly detect compatibility violations once the failures have been observed, the suggested framework combines the continuous monitoring of the schema, the process of automatic contract validation, and the computation of risks with the help of machine learning.

➢ The method allows terminating risky schema changes at an early stage, enhances the reliability of services, minimizes operational failures and enables adaptive contract evolution within the complex distributed software context.

II. RELATED WORK

The development of Microservice Architectures and Contract-Based Communication

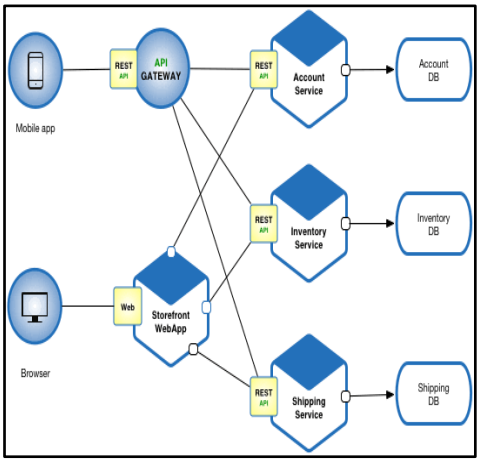


Fig. 1: Microservice Architectures and Contract-Based Communication

The development of microservice architecture has revolutionized the field of software development offering independently deployable, scalable and decentralized services. Effective interaction of distributed services needs to be communicated with clear contracts, such as API interfaces, versioning schemes, and backward compatibility policies. Here, [5] emphasized need of API evolution management by using version control and preservation of compatibility in order to minimize discontinuity as services continuously advance. Similarly, [6] highlighted the use of interaction-oriented software engineering where autonomous components need decentralized systems that are supported by reliable communication abstractions. Moreover, [7] investigated the application of microservices by means of blockchain smart contracts and proved that the protocol of contract-based interaction has potential to enhance the trust and coordination of the services. The study [8] replicated this idea with smart contract microcivilization with emphasis on modular and autonomous service execution. In the same way, [9] introduced a microservice based architecture where blockchain technology is incorporated to develop dependable distributed applications.

The Problems and Effects of Schema Drift on Microservice Reliability

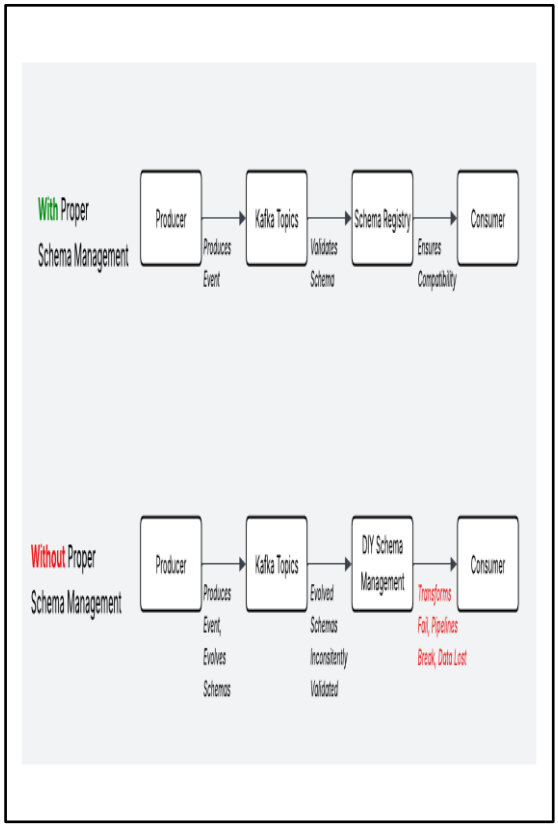


Fig. 2: Schema Drift Challenges and Their Impact

Schema drift is one of the major issues facing microservice environments because of the common data structure changes, the independent deployment of services, and the mounting system dependencies. The significance of automated schema drift detection to DevOps pipelines to detect compatibility problems and avoid deployment failures was emphasized [10]. According to [11], heterogeneous enterprise systems require the consistency of data workflows, and that is why there is a need to have fault-tolerant data workflows. Similarly, [12] talked about self-healing API ecosystems, which are achieved through autonomous integration strategies to minimize operational failures due to changed interfaces. The study [13] determined data quality management as a critical element to credible decision-making in multifaceted data setting. On the other hand, [14] examined the reliability of problems in microservice-based cloud applications and due to the criticality of monitoring service dependencies and propagation of failures, the authors asserted the significance of reliability.

Automated Contract Testing and Schema Evolution Strategies

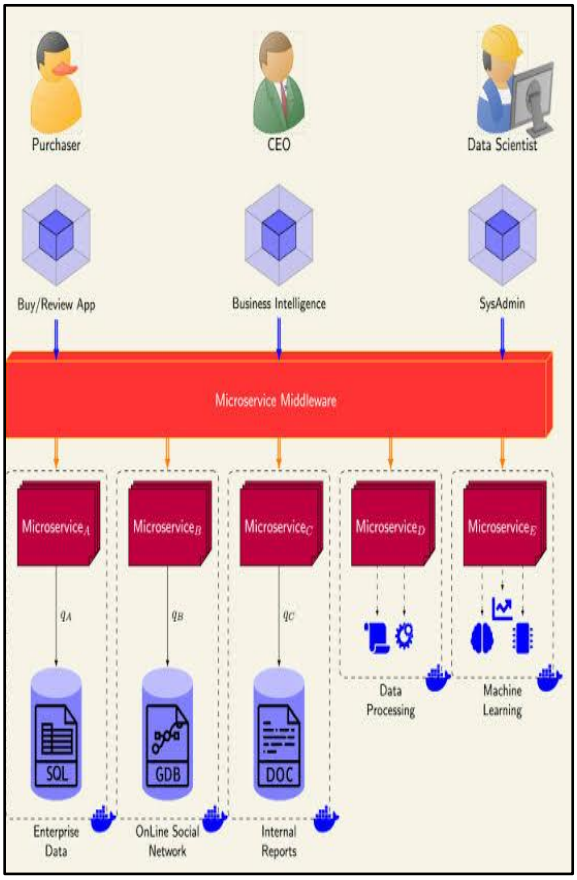


Fig. 3: Contract Testing and Schema Evolution Strategies

Automated contract testing has become a fundamental strategy in ensuring trustworthiness in the face of ongoing microservice evolution through testing interactions between service providers and consumers. Now, [15] introduced the idea of machine learning prediction of the success or failure of any type of data contract, showing how clever methods may be implemented to detect risks of non-compatibility prior to the actual implementation. Similarly, the study [16] emphasized on versioning of API and the backward compatibility tool or platform to facilitate rearrangement of large-scale microservices without causing disruption of services. As per [17], performance of digital contracts which highlighted formal contract mechanisms as the key to reliable automated interactions. According to [18], model-based testing techniques of smart contracts are determined the significance of systematic validation methods of complex distributed use cases. To enhance software reliability, [19] proposed automated integration testing on the basis

of logical contracts which enhanced the inherent behavioral rules.

Detection and Predictive Management of Schema Drift based on Artificial Intelligence

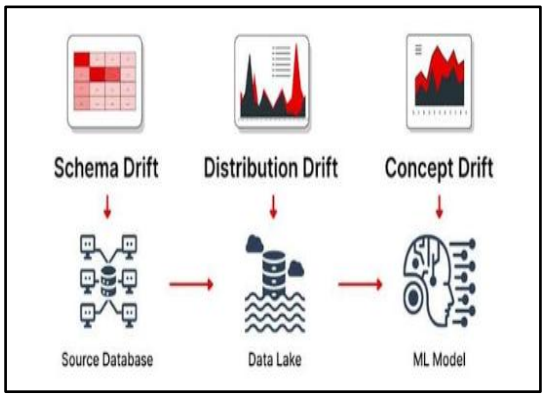


Fig. 4: Artificial Intelligence-Based Detection and Predictive Management of Schema Drift

The use of artificial intelligence and machine learning methods has been attracting growing interest to detect changing dynamics, detect anomalies, and forecast risk in dynamic software. The study [20] recommended a explainable artificial intelligence (AI) based method of model drift detection in unsupervised settings, showing the role of explainable drift detection specifications. Similarly, [21] emphasized the importance of metadata-based data orchestration to operate variations in data environments as well as enhance the reliability of automated data processing. On the other Hand, [22] surveyed concept drift learning techniques and highlighted the limitations of changing intelligent models in response to the time-varying data distributions. Then, [23] proposed a meta-learning-based drift detection methodology with the potential to enhance the early detection of shifting patterns in dynamic patterns. Similarly, [24] presented the research on the methods of prediction and optimization of self-healing cloud applications and demonstrated the possibility of intelligent automation of the system to increase the resilience of the system.

Research Gap

Available literature touches on microservice contract management, schema validation, API versioning, and automated testing but very little has been done on proactive prediction of schema drift as well as smart contract evolution. The existing methods primarily identify failures when they take place instead of predicting compatibility risks. Thus, a unified system that integrates machine learning prediction with drift detection, automated validation is needed.

III. PROPOSED FRAMEWORK

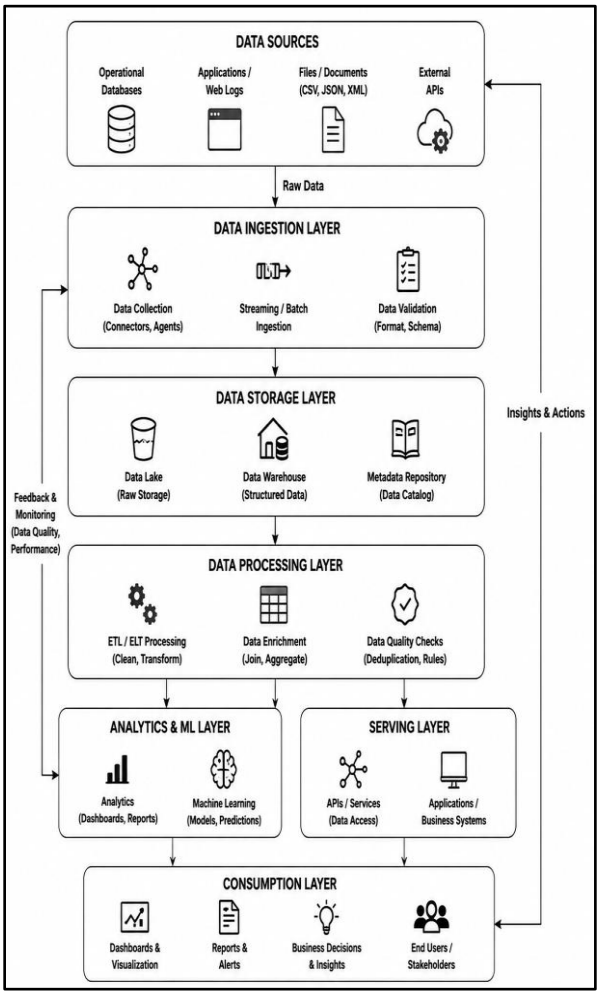


Fig. 5: Proposed Framework

The implemented framework outlines an orderly framework to identify and control schema drift in changing microservice contracts. The framework starts with the microservice ecosystem layer, in which several services use API and data contracts like OpenAPI, JSON Schema, and other interface definitions to communicate with each other. Contract data ingestion layer collects the version histories, dependency relationships, contract information, and runtime monitoring data. The information gathered is fed into the schema drift detection layer where schema comparison, classification of change, detection of drift and analysis of impact are carried out [25]. Schema drifts risk:

$$Risk = f(X, \theta)$$

where:

X = schema change characteristics

θ = model parameters

The intelligence and risk prediction layer uses machine learning-based models, which are the Random Forest and Gradient Boosting to determine the possibility of compatibility risk due to schema changes. Contract management layer entails version control, compatibility checking, automated validation and remediation measures [26]. Lastly, the framework enhances microservice reliability by decreasing failures, serving data consistency, and facilitates ongoing and secure evolution of services via feedback and monitoring systems.

IV. METHODOLOGY

Research Design and Data Collection

The study takes a quantitative experimental approach to formulate and test an intelligent system of identifying and addressing schema drift in the shifting microservice contracts. The paper is based on secondary sources that are summarized and gathered on the web, as in open-source microservice repositories, and API contract datasets, software engineering benchmarks, and service evolution logs. Microservice contract datasets of 10,000 records and 12 attributes defining schema evolution events are used as secondary data in the study. Columns are version of schema, type of change, frequency of API modification, number of dependencies, compatibility of the change, frequency of failures and the degree of drift.

Preprocessing and engineering of data

Processing of the collected secondary data is done using Python based data analytics libraries such as Pandas, NumPy and Scikit-learn. The first step involves preprocessing data to enhance the quality and suitability of the data to be examined by a machine learning model. Inconsistent records, missing values as well as duplicate records are detected and addressed through relevant data cleaning methods [27]. Categorical variables, including types of schema changes, API versions, and categories of compatibility are encoded into numerical values using encoding methods. Standardization of numeric attributes like the number of schema changes, dependency level, service contacts, and failure rates to minimize differences between scales of features is done.

$$Xscaled = \frac{X - \mu}{\sigma}$$

Where μ is the mean and σ is the standard deviation

The feature engineering is carried out to capture meaningful features relating to schema drift behavior. The attributes chosen are schema change frequency, complexity of the contract change, number of services it depends on, the pattern of API usage, version discrepancy and the number of times it has failed historically. The assessment and analysis of correlation and the importance of features are used to determine the most significant inputs to the issue of schema compatibility and reliability of the services. Machine learning models are input with these engineered features and predict the risk of schema drift.

Schema Drift Prediction with Machine Learning

The processed data are then analyzed with a series of supervised machine learning models coded in Python to estimate the likelihood of schema drift in microservice contracts. The framework uses Random Forest and Gradient Boosting models because they have the potential to establish complicated connections between schema alterations, service relationships, and compatibility breakdowns [28]. Random Forest predictive failure is achieved through a combination of a number of decision trees:

$$Y = \frac{1}{N}\sum_{i=1}^{N} \quad Ti(X)$$

where Y is the predicted outcome of a schema drift, Ti(X) is the outcome of prediction of the ith decision tree, and N is the number of trees. Gradient Boosting enhances prediction by improvement, which is based on:

$$Fm(X) = Fm - 1(X) + \gamma hm(X)$$

In which Fm(X) is the revised prediction model, hm(X) is a weak learner and 81 is the learning rate. The dataset is categorized into training and testing part in the ratio of 80:20. The model performance, reliability and generalization performance through hyper parameter optimization using the Grid Search and cross-validation can be used to enhance resource use and prediction accuracy of schema drift.

Framework Evaluation and Measurement of Performance

Performance metrics are used to assess the efficiency of the proposed schema drift management framework to assess multiple performance values, including accuracy, precision, recall, F1-score and ROC-AUC. Accuracy is used to determine the general accuracy of the schema drift prediction whereas precision and recall are used to determine how well the models can recognize dangerous changes in the contract. The F1-score is a balanced estimation of the precision and recall, especially when there are non-uniform distributions of drift and non-drifting instances. To evaluate the ability of the model to differentiate between risk items of various categories, ROC-AUC is taken.

Flow Diagram

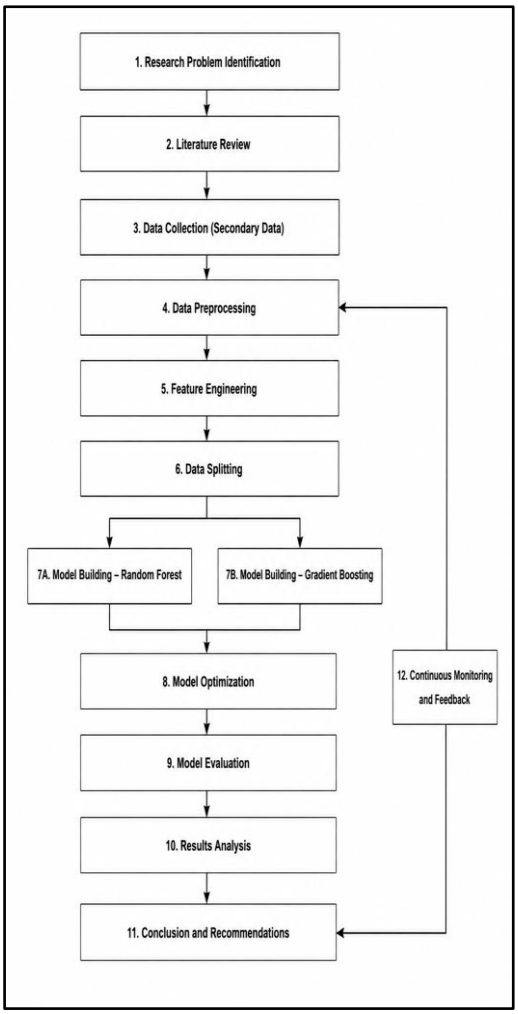


Fig. 6: Flow Diagram of the Research

Pseudo Code

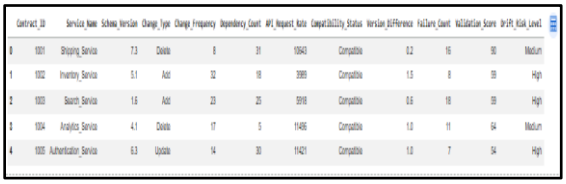
```
1. Load dataset
2. Perform data preprocessing:
    - Handle missing values
    - Encode categorical features
    - Normalize numerical attributes
3. Extract schema drift features:
    - Schema changes
    - API versions
    - Service dependencies
    - Compatibility failures
4. Split dataset into training and testing data (80:20)
5. Train Random Forest model
6. Train Gradient Boosting model
7. Optimize models using Grid Search and
Cross-validation
8. Predict schema drift risk levels
9. Evaluate models using:
    - Accuracy
    - Precision
    - Recall
    - F1-score
    - ROC-AUC
10. Select best-performing model
```

Fig. 7: Pseudo Code

Ethical Consideration

The study deploys publicly available secondary data which do not reveal personal or confidential data. There is proper acknowledgement of data sources and responsible data management practices are useful in the analysis [29]. The analysis is transparent in preprocessing, model development and evaluation. Interpretation is done to machine learning estimations to eliminate bias choices so that a solid schema evolution management could be achieved.

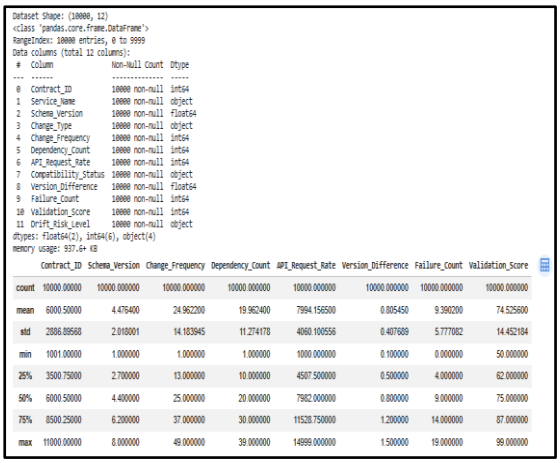
V. RESULTS AND DISCUSSION



	Contract_ID	Service_Name	Schema_Version	Change_Type	Change_Frequency	Dependency_Count	API_Request_Rate	Compatibility_Status	Version_Difference	Failure_Count	Validation_Score	Drift_Risk_Level
0	1001	Shipping_Service	7.3	Delete	8	31	10643	Compatible	0.2	16	90	Medium
1	1002	Inventory_Service	5.1	Add	32	18	3995	Compatible	1.5	8	39	High
2	1003	Search_Service	1.6	Add	23	25	5918	Compatible	0.6	18	39	High
3	1004	Analytics_Service	4.1	Delete	17	5	11406	Compatible	1.0	11	64	Medium
4	1005	Authentication_Service	6.3	Update	14	30	11421	Compatible	1.0	7	54	High

Fig. 8: Distribution of Levels of schema drift risk

The dataset has been loaded into the Python platform using read() function. Here, the details of the dataset have been shown in the above figure.



```
Dataset Shape: (10000, 12)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 12 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Contract_ID          10000 non-null  int64
 1   Service_Name         10000 non-null  object
 2   Schema_Version       10000 non-null  float64
 3   Change_Type          10000 non-null  object
 4   Change_Frequency     10000 non-null  int64
 5   Dependency_Count     10000 non-null  int64
 6   API_Request_Rate     10000 non-null  int64
 7   Compatibility_Status 10000 non-null  object
 8   Version_Difference   10000 non-null  float64
 9   Failure_Count        10000 non-null  int64
 10  Validation_Score     10000 non-null  int64
 11  Drift_Risk_Level     10000 non-null  object
dtypes: float64(2), int64(6), object(4)
memory usage: 937.6+ KB
```

	Contract_ID	Schema_Version	Change_Frequency	Dependency_Count	API_Request_Rate	Version_Difference	Failure_Count	Validation_Score
count	10000.00000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000
mean	6000.50000	4.476400	24.962200	19.962400	7994.156500	0.805450	9.390200	74.525600
std	2886.89568	2.018001	14.103945	11.274178	4060.100556	0.407689	5.777082	14.452184
min	1001.00000	1.000000	1.000000	1.000000	1000.000000	0.100000	0.000000	50.000000
25%	3500.75000	2.700000	13.000000	10.000000	4507.500000	0.500000	4.000000	62.000000
50%	6000.50000	4.400000	25.000000	20.000000	7982.000000	0.800000	9.000000	75.000000
75%	8500.25000	6.200000	37.000000	30.000000	11528.750000	1.200000	14.000000	87.000000
max	11000.00000	8.000000	49.000000	39.000000	14999.000000	1.500000	19.000000	99.000000

Fig. 9: Data Overview and Statistic Characteristics

The data set comprises of 10, 000 records and 12 attributes. Each column has 10000 non-NULLs, and that indicates the presence of complete and perfect data there. Numerical analysis presents the average schema version of 4.47, change average of 24.98, dependency average of 19.96 and a validation average of 74.52, which give appropriate properties to be utilized in machine learning analysis.

	0
Contract_ID	0
Service_Name	0
Schema_Version	0
Change_Type	0
Change_Frequency	0
Dependency_Count	0
API_Request_Rate	0
Compatibility_Status	0
Version_Difference	0
Failure_Count	0
Validation_Score	0
Drift_Risk_Level	0

```
Duplicate Value:  0
```

Fig. 10: Data Quality assessment

The quality assessment of data shows that zero missing values were found in all variables, which guarantees a solid model development and analysis. Also, duplicate detection reports 0 duplicate records which indicates that every microservice contract observation is a distinct schema evolution event.

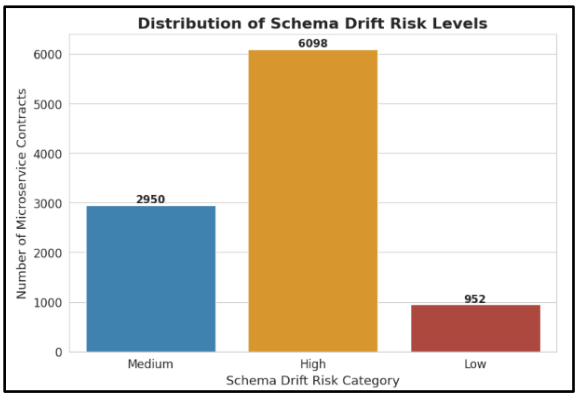


Fig. 11: Distribution of Schema Drift Risk Levels

The risk distribution analysis demonstrates that there is a high variability in the categories of the schema drift. There are 10,000 microservice contracts, of which 6,098 are considered as High Risk, 2,950 as Medium Risk, and 952 as Low Risk. High-risk cases prevail which, combined with the frequent alterations of the contract and dependency, often cause compatibility issues as part of changing microservice environments.

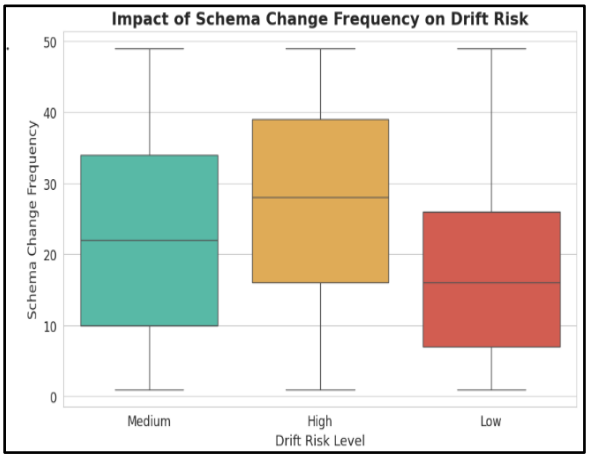


Fig. 12: The Effect of the Change Frequency of the Schema on Drift Risk

Boxplot illustrates that boxplot frequency of schema change can have effects on the classification of the drift risk. The median change frequency of a High-risk contract is highly high of about 28 changes, as opposed to 22 changes of a Medium Risk contract and 16 changes of a Low Risk contract.

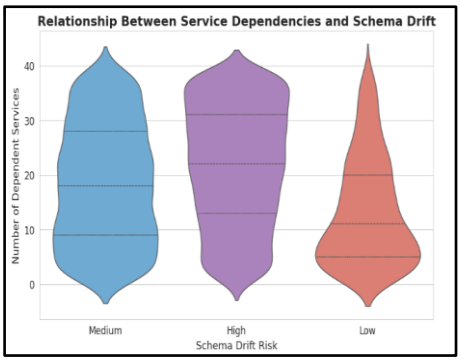


Fig. 13: Service Dependencies and schema drift

The dependency complexity based on the different types of drift is analyzed by the violin plot. The dependency variation in high-risk contracts is higher; there is an average of about 39 linked services whereas in low-risk contracts dependency is usually lower.

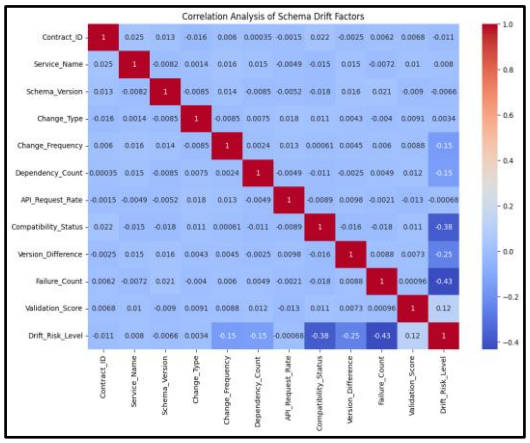


Fig. 14: Correlation Analysis of the factors of Schemas Drift

The correlation heatmap is used to determine the relationships between input variables and schema drift risk. Failure_Count (-0.43) (Compatibility_Status -0.38) Version Difference (-0.25) are the strongest influencing factors. The number of change and dependency show moderate relations (-0.15 each).

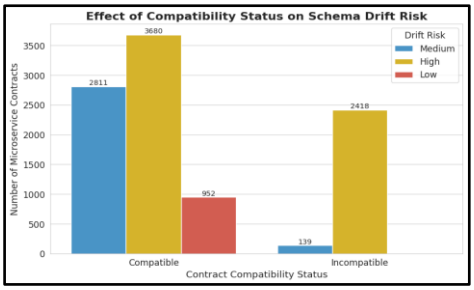


Fig. 15: Effect of Compatibility Status on schema drift risk.

The compatibility analysis measures the implications of contract compatibility on the severity of drift. The services with 3,680 High-risk, 2,811 Medium-risk, and 952 Low-risk cases are compatible and there are 2,418 High-risk and 139 Medium-risk cases incompatible services.

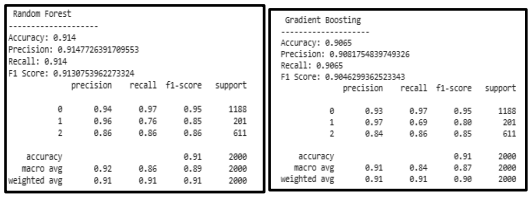


Fig. 16: Classification Performance

The overall accuracy of the Random Forest model is 91.4% and proves to have an effective ability to classify schema drift. The model has a weighted precision of 91.48, a recall of 91.4 and a F1-score of 91.31. Class-level findings show that there is a high level of recognition of Low-risk cases with high recall of 97% whereas the High-risk cases exhibit high recall of 86% cases. Gradient Boosting model is 90.65% accurate, weighted precision is 90.82, recall is 90.65 and F1 score is 90.46. The model achieves recall of 97 and 86 in terms of predicting Low-risk contract and High-risk contract respectively.

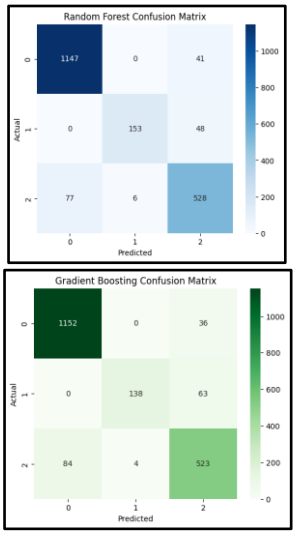


Fig. 17: Comparison of machine learning models on confusion matrices

The confusion matrices are used to test the correctness of prediction of both ensemble models. Random Forest has 1,147 Low-risk, 153 Medium-risk, and 528 High-risk contracts correctly classified whereas Gradient Boosting has 1,152 Low-risk, 138 Medium-risk, and 523 High-risk contracts correctly classified.

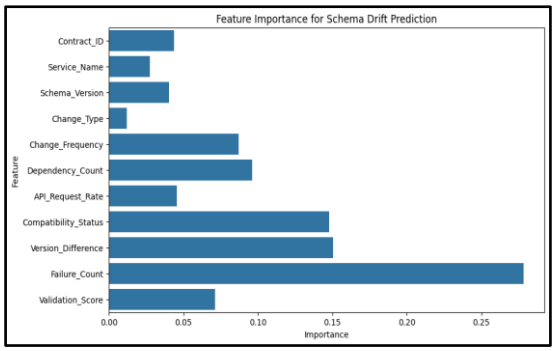


Fig. 18: Importance of Features to predict the existence of schema drift

Failure Count is the most important with about 27 percent contribution followed by Compatibility Status (15 percent), Version difference (15 percent) then dependent count (9 percent). Other predictors of microservice contract drift are seen to be the highest, historical failures, compatibilities, and schema version alternations.

Table 1: Performance Evaluation of Machine Learning Models

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Random Forest	91.40	91.48	91.40	91.31
Gradient Boosting	90.65	90.82	90.65	90.46

Table 2: Schema Drift Risk Distribution

Drift Risk Category	Number of Contracts	Percentage (%)
High Risk	6098	60.98
Medium Risk	2950	29.50
Low Risk	952	9.52
Total	**10000**	**100**

Results Interpretation

The experimental findings indicate the existence of machine learning models that are useful to predict the risks of schema drift in contract currying within microservice contracts. Random Forest has the best results of 91.40 accuracy and 91.31 F1-score, a little better than Gradient Boosting. Android feature analysis determines number of failures, compatibility and version differences as significant indicators of drift. The results confirm the fact that predictive analytics has the ability to anticipate the risk of contract evolution and enhance the reliability of microservices, dealing with compatibility issues, and continually developing services.

VI. RESEARCH LIMITATIONS AND FUTURE DIRECTIONS

Limitation

- The study employs a secondary set of data to simulate the situation of microservice contract evolution, as such, the results might not reflect the complexity, diversity and dynamic behavior of real industrial microservice contexts.
- It only compares the Random Forest with Gradient Boosting model, whereas other more sophisticated algorithms like deep learning, reinforcement learning, and real-time streaming-based detectors are not delved into.

Future direction

This framework can be further refined using future research that will include real-time microservice monitoring data in industrial cloud environments to enhance potential applicability. State-of-the-art artificial intelligence methods, such as deep learning, graph neural networks, and reinforcement learning may be explored in dynamic schema drift forecasting [30]. Also, automated remediation, self-healing contract management, and explainable AI strategies should be included in future studies to enhance transparency, flexibility, and trustworthiness of affected microservice ecosystems constantly changing.

VII. CONCLUSION

This study derives a robust design of schema drifts identification and management in changing microservice contracts through prediction methods of machine learning. The experimental outcomes show that ensemble learning models have a strong ability to detect the risks of schema drift, with the highest accuracy being 91.40% by using Random

Forest. The analysis indicates that the number of failures that occur, compatibility status, and version differences are some of the major factors that affect contract instability. Combining schema-monitoring, feature-analysis and predictive-modelling, the suggested framework improves the proactive evolution of a contract, as well as minimizes the risks of service reliability.

VIII. REFERENCES

[1] Taylor, K. and Reza, A., 2018. Automated Data Quality Monitoring in Streaming Data Transformations. *International Journal of Data Engineering and Intelligent Computing*, *1*(2), pp.01-12. https://worldcometresearchgroup.com/index.php/ijdeic/article/download/557/537

[2] Nkosi, T., 2019. Continuous Data Transformation in Event-Driven Microservices. *International Journal of Artificial Intelligence & Digital Transformation*, *2*(2), pp.01-14. https://worldcometresearchgroup.com/index.php/ijaidt/article/download/202/192

[3] Lan, N.T., 2021. Resilient Data Ingestion Patterns in Cloud-Native Data Mesh Architectures. *International Journal of Artificial Intelligence & Digital Transformation*, *4*(2), pp.01-12. https://worldcometresearchgroup.com/index.php/ijaidt/article/download/224/214

[4] Apolinário, D.R. and de França, B.B., 2021. A method for monitoring the coupling evolution of microservice-based architectures. *Journal of the Brazilian Computer Society*, *27*(1), p.17. https://link.springer.com/content/pdf/10.1186/s13173-021-00120-y.pdf

[5] Singh, D., 2022. Managing API Evolution in Large-scale Microservices: Versioning and Backward Compatibility. *International Journal of Science, Technology and Convergence*, *4*(4). https://ijcdra.us/index.php/IJSTC/article/download/61/8

[6] Chopra, A.K., 2022. Interaction-oriented software engineering: Programming abstractions for autonomy and decentralization. *AI Communications*, *35*(4), pp.381-391. https://www.lancaster.ac.uk/staff/chopraak/pdfs/chopra-vision.pdf

[7] Tonelli, R., Lunesu, M.I., Pinna, A., Taibi, D. and Marchesi, M., 2019, February. Implementing a microservices system with blockchain smart contracts. In *2019 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE)* (pp. 22-31). IEEE. https://iris.unica.it/bitstream/11584/262865/1/Microservices_blockchain-preprint.pdf

[8] Wang, S., Zhang, X., Yu, W., Hu, K. and Zhu, J., 2020, July. Smart contract microservitization. In *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)* (pp. 1569-1574). IEEE. https://iris.unica.it/bitstream/11584/262865/1/Microservices_blockchain-preprint.pdf

[9] Nagothu, D., Xu, R., Nikouei, S.Y. and Chen, Y., 2018, September. A microservice-enabled architecture for smart surveillance using blockchain technology. In *2018 IEEE international smart cities conference (ISC2)* (pp. 1-4). IEEE. https://arxiv.org/pdf/1807.07487

[10] Sirimalla, A., 2022. End-to-end automation for cross-database DevOps deployments: CI/CD pipelines, schema drift detection, and performance regression testing in the cloud. *World Journal of Advanced Research and Reviews*, *14*(3), pp.871-889. https://doi.org/10.30574/wjarr.2022.14.3.0555

[11] Keshireddy, S.R. and Kavuluri, H.V.R., 2019. Design of Fault Tolerant ETL Workflows for Heterogeneous Data Sources in Enterprise Ecosystems. *International Journal of Communication and Computer Technologies*, *7*(1), pp.42-46. https://www.eccsubmit.com/index.php/ijccts/article/download/134/335

[12] Kotla, M.R.T., 2023. Autonomous enterprise integration: The future of self-healing data and API ecosystems. *International Journal of Research and Applied Innovations*, *6*(3), pp.5968-5971. https://www.ijrai.org/index.php/ijrai/article/download/599/550

[13] Bhaskaran, S.V., 2020. Integrating data quality services (dqs) in big data ecosystems: Challenges, best practices, and opportunities for decision-making. *Journal of Applied Big Data Analytics, Decision-Making, and Predictive Modelling Systems*, *4*(11), pp.1-12. https://polarpublications.com/index.php/JABADP/article/download/4/4

[14] Liu, Z., Fan, G., Yu, H. and Chen, L., 2021. An Approach to Modeling and Analyzing Reliability for Microservice-Oriented Cloud Applications.

Wireless Communications and Mobile Computing, *2021*(1), p.5750646. https://onlinelibrary.wiley.com/doi/pdf/10.1155/2021/5750646

[15] Chirumamilla, K.R., 2023. Predicting data contract failures using machine learning. *The Eastasouth Journal of Information System and Computer Science*, *1*(01), pp.144-155. https://esj.eastasouth-institute.com/index.php/esiscs/article/download/843/657

[16] Singh, D., 2022. Managing API Evolution in Large-scale Microservices: Versioning and Backward Compatibility. *International Journal of Science, Technology and Convergence*, *4*(4). https://ijcdra.us/index.php/IJSTC/article/download/61/8

[17] Egelund-Müller, B., Elsman, M., Henglein, F. and Ross, O., 2017. Automated execution of financial contracts on blockchains. *Business & Information Systems Engineering*, *59*(6), pp.457-467. https://papers.ssrn.com/sol3/Delivery.cfm?abstractid=2898670

[18] Sánchez-Gómez, N., Torres-Valderrama, J., García-García, J.A., Gutiérrez, J.J. and Escalona, M.J., 2020. Model-based software design and testing in blockchain smart contracts: A systematic literature review. *IEEE Access*, *8*, pp.164556-164569. https://ieeexplore.ieee.org/iel7/6287639/6514899/09186040.pdf

[19] Xu, D., Xu, W., Tu, M., Shen, N., Chu, W. and Chang, C.H., 2015. Automated integration testing using logical contracts. *IEEE Transactions on Reliability*, *65*(3), pp.1205-1222. https://www.researchgate.net/profile/Manghui-Tu-2/publication/283822941_Automated_Integration_Testing_Using_Logical_Contracts/links/5e4472b0458515072d96d050/Automated-Integration-Testing-Using-Logical-Contracts.pdf

[20] Lee, Y., Lee, Y., Lee, E. and Lee, T., 2023. Explainable artificial intelligence-based model drift detection applicable to unsupervised environments. *Computers, Materials and Continua*, *76*(2), pp.1701-1719. https://www.sciencedirect.com/org/science/article/pii/S1546221823001558

[21] Bukhari, T.T., Oladimeji, O., Etim, E.D. and Ajayi, J.O., 2022. Systematic review of metadata-driven data orchestration in modern analytics engineering. *Gyanshauryam, International Scientific Refereed Research Journal*, *5*(4), pp.536-564. https://gisrrj.com/paper/GISRRJ225429.pdf

[22] Lu, J., Liu, A., Dong, F., Gu, F., Gama, J. and Zhang, G., 2018. Learning under concept drift: A review. *IEEE transactions on knowledge and data engineering*, *31*(12), pp.2346-2363. https://arxiv.org/pdf/2004.05785

[23] Yu, H., Zhang, Q., Liu, T., Lu, J., Wen, Y. and Zhang, G., 2022. Meta-ADD: A meta-learning based pre-trained model for concept drift active detection. *Information Sciences*, *608*, pp.996-1009. https://www.sciencedirect.com/science/article/pii/S0020025522007125

[24] Alonso, J., Orue-Echevarria, L., Osaba, E., López Lobo, J., Martinez, I., Diaz de Arcaya, J. and Etxaniz, I., 2021. Optimization and prediction techniques for self-healing and self-learning applications in a trustworthy cloud continuum. *Information*, *12*(8), p.308. https://www.mdpi.com/2078-2489/12/8/308

[25] Nallaperuma, D., Nawaratne, R., Bandaragoda, T., Adikari, A., Nguyen, S., Kempitiya, T., De Silva, D., Alahakoon, D. and Pothuhera, D., 2019. Online incremental machine learning platform for big data-driven smart traffic management. *IEEE Transactions on Intelligent Transportation Systems*, *20*(12), pp.4679-4690. https://ieeexplore.ieee.org/iel7/6979/4358928/08759919.pdf

[26] Khan, M.N.I., 2022. A Systematic Review of Legal Technology Adoption In Contract Management, Data Governance, And Compliance Monitoring. *American Journal of Interdisciplinary Studies*, *3*(01), pp.01-30. https://ajisresearch.com/index.php/ajis/article/download/27/26

[27] Chu, X., Ilyas, I.F., Krishnan, S. and Wang, J., 2016, June. Data cleaning: Overview and emerging challenges. *In Proceedings of the 2016 international conference on management of data* (pp. 2201-2206).https://dl.acm.org/doi/pdf/10.1145/2882903.2912574

[28] Wang, T.C., Guo, R.S. and Chen, C., 2023. An integrated data-driven procedure for product specification recommendation optimization with LDA-LightGBM and QFD. *Sustainability*, *15*(18),

p.13642. https://www.mdpi.com/2071-1050/15/18/13642

[29] Leonelli, S., 2016. Locating ethics in data science: responsibility and accountability in global and distributed knowledge production systems. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *374*(2083), p.20160122. https://pmc.ncbi.nlm.nih.gov/articles/PMC5124067/pdf/rsta20160122.pdf

[30] Munikoti, S., Agarwal, D., Das, L., Halappanavar, M. and Natarajan, B., 2023. Challenges and opportunities in deep reinforcement learning with graph neural networks: A comprehensive review of algorithms and applications. *IEEE transactions on neural networks and learning systems*, *35*(11), pp.15051-15071. https://arxiv.org/pdf/2206.07922