



Reducing Latency Between Driving Behaviour and Risk Assessment: A Kafka and Spark Streaming Architecture for Usage-Based Insurance

Mohammed Ataullah Baig Mirza

Big Data Engineer at Allstate Insurance.

Abstract

In usage-based insurance programmes, the value of telematics depends on how quickly recorded driving behaviour is transformed into validated risk assessment. The continuous arrival of telematics events conflicts with batch processing because waiting for the next processing cycle separates the time of the trip from the time when its characteristics become available for analytics. This study proposes treating latency as a property of the entire processing pipeline, including event transmission, ingestion, queue waiting, transformation, data quality validation, and risk score generation. On this basis, the LRV-UBI model is developed, integrating data streaming, operational reliability, and the validity of behavioural features. Kafka and Spark Streaming constitute the technological core of the model; however, the reduction in processing time depends on the coordinated configuration of partitioning, consumer performance, computational parallelism, and state recovery mechanisms. Particular importance is assigned to handling late-arriving, duplicate, and invalid events that may alter previously computed features. Risk assessment is released only after verifying data completeness and consistency; therefore, validation time is included in end-to-end latency. A measurable indicator is defined for each processing stage, making it possible to localize the source of delay and distinguish computational acceleration from the actual availability of results. The LRV-UBI model is intended for designing verifiable streaming architectures for insurance telematics, in which processing speed is evaluated together with recovery resilience and the suitability of data for risk models. The model provides a foundation for future architectural comparisons using standardized indicators without directly transferring incomparable latency and throughput values from other application domains into the insurance context.

Keywords: Insurance Telematics, Driving Behaviour, Insurance Risk Assessment, Stream Processing, Latency Reduction, Data Quality, Processing Reliability.

INTRODUCTION

Usage-based insurance (UBI) assesses insurance risk based on the actual driving behaviour of the policyholder. Mobile applications, onboard devices, and vehicle sensors continuously record speed, acceleration, braking, trip time, and travel parameters [10]. Sudden speed changes, nighttime driving, and harsh braking are associated with an increased number of insurance claims. Behavioral and contextual indicators also improve accident frequency models, while the classification of drivers according to driving style helps distinguish risk groups [8]. Dynamic features reflect changes in behaviour over time and improve accident prediction performance. For commercial transport, trip parameters and alerts generated by in-cabin monitoring systems provide additional information about risk [15]. However, telematics events arrive continuously, whereas a batch processing pipeline processes them only after a predefined data volume

has accumulated or a specified time interval has elapsed. Until the next processing job starts, newly received data remain unavailable for risk scoring. This issue was observed in the de-identified UBI programme of a major U.S. auto insurer. Slow batch-oriented workflows limited the speed at which driving behaviour was transformed into risk assessment analytics [1]. After the transition to a streaming architecture, latency was reduced, although the case materials do not provide verified results in seconds, minutes, or percentage values.

End-to-end latency includes event transmission, ingestion, queue waiting, processing, validation, and risk score generation. A bottleneck at any stage increases the time required before a usable assessment becomes available; therefore, performance depends on the architecture of the entire pipeline rather than on the speed of an individual operation. Consumer lag reflects the accumulation of

Citation: Mohammed Ataullah Baig Mirza, "Reducing Latency Between Driving Behaviour and Risk Assessment: A Kafka and Spark Streaming Architecture for Usage-Based Insurance", Universal Library of Innovative Research and Studies, 2026; 3(4): 08-15. DOI: <https://doi.org/10.70315/uloap.ulirs.2026.0304002>.

unprocessed events and serves as an indicator of increasing latency [2]. Kafka decouples producers from consumers while supporting partitioning, buffering, and message replay, whereas Spark Streaming continuously transforms telematics events into behavioural features. Increasing the number of partitions and processing instances improves throughput until one of the system components reaches saturation [7]. An additional integration layer may increase latency and data loss under high load, whereas coordinated scaling of data and computational resources helps maintain performance. A continuous processing pipeline must also handle late-arriving and duplicate events, recover from short-term failures, and process accumulated workloads. At the same time, missing data, duplicates, incorrect timestamps, and inconsistencies across devices may distort the features used by the risk model [11]. In the examined case, engineers worked together with actuaries and data science specialists to verify data quality and consistency. UBI research primarily evaluates the predictive value of telematics [9], whereas studies on streaming systems focus on latency, throughput, and fault tolerance. The connection between these two research areas, from a vehicle event to the availability of validated risk-scoring data, remains insufficiently explored.

The aim of this study is to develop the LRV-UBI model, integrating Kafka, Spark Streaming, operational reliability mechanisms, observability, and data validation to reduce end-to-end latency between the arrival of a telematics event and the generation of data for insurance risk assessment. To achieve this aim, the following objectives were defined:

- to systematize the application of telematics in assessing driving behaviour and insurance risk;
- to identify the limitations of batch-oriented workflows;
- to determine the functions of Kafka and Spark Streaming within the UBI architecture;
- to analyse the factors affecting latency and throughput;
- to formulate requirements for handling failures, late-arriving events, and duplicate events;
- to define the role of actuarial and data science validation.

The research hypothesis assumes that, under comparable workloads, the transition from batch-oriented workflows to an architecture based on Kafka and Spark Streaming reduces end-to-end latency between the registration of a telematics event and the availability of risk assessment, provided that late-event processing, recovery, duplicate prevention, monitoring, and validation are integrated into the continuous processing pipeline. The available evidence supports this assumption at the level of architectural synthesis, whereas its quantitative verification requires a comparison of batch and streaming implementations using the same telematics data stream.

The scientific novelty of the study lies in representing end-

to-end latency as the sum of the time required for event generation, ingestion, queue waiting, processing, validation, and risk scoring, while incorporating reliability and data validity into the critical path. The proposed LRV-UBI model links each latency component to a specific architectural mechanism and a measurable indicator: ingestion rate, consumer lag, processing latency, the proportion of rejected events, data completeness, and risk-scoring availability time. Since Kafka and Spark Streaming are established technologies, the author's contribution lies in integrating them with recovery mechanisms, observability, behavioural feature verification, and actuarial validation into an operationalizable model for a regulated insurance environment.

MATERIALS AND METHODS

The study was conducted as a structured narrative review followed by the architectural synthesis of the LRV-UBI model. The analysis covered the use of telematics data for driver behaviour classification and insurance risk assessment, as well as the application of Kafka, Spark Streaming, and other stream processing systems to reduce latency, improve scalability, and support recovery from failures. A systematic review protocol and the PRISMA procedure were not applied.

The literature search was performed using Google Scholar, SpringerLink, MDPI, and arXiv by searching titles, abstracts, and keywords. Publications from 2022–2025 were considered. The following search queries were used: (“usage-based insurance” OR UBI) AND (telematics OR “driving behaviour” OR “driving behavior”); (“driver scoring” OR “driving behaviour”) AND (“risk assessment” OR “claim frequency” OR ratemaking); (Kafka AND (“Spark Streaming” OR “stream processing”)) AND (latency OR throughput OR reliability); (“consumer lag” OR “late-arriving events” OR replay OR deduplication) AND (Kafka OR Spark); (“stream processing architecture” OR “edge stream processing”) AND (scalability OR “fault tolerance” OR “event loss”). Additional studies were identified from the reference lists of relevant publications. No direct search was conducted in Scopus, Web of Science, ACM Digital Library, or IEEE Xplore; therefore, the resulting literature corpus cannot be considered exhaustive.

The initial sample comprised 47 bibliographic records. The analysis included full-text journal articles, conference papers, and arXiv publications describing an architecture, method, prototype, experiment, or analytical results related to insurance telematics or stream processing. Duplicate records, publications without accessible full texts, studies published outside the selected period, general studies of Kafka and Spark without relevant architectural findings, and studies on driver behaviour unrelated to risk assessment were excluded. After reviewing titles, abstracts, and full texts, 32 records were excluded, resulting in a final corpus of 15 studies. The number of exclusions at individual screening

stages was not recorded, which limits the reproducibility of the selection process. Independent screening by a second researcher was also not performed.

Information extracted from each publication included the research objective, unit of observation, dataset size, data collection period, processing mode, architecture, quantitative indicators, and reported limitations. Numerical results were retained in their original units and interpreted only within the context of the datasets and system configurations described in the respective studies. No meta-analysis was performed because of differences in workloads, infrastructure, and evaluation metrics. The strength of evidence was assessed according to the presence of a baseline for comparison, the measurability of outcomes, and the transparency of the experimental conditions. Studies satisfying all three criteria were assigned the greatest evidential weight.

The second source of evidence was a de-identified professional case from the UBI programme of a major U.S. auto insurer. Its reconstruction was based on direct professional participation in the project and a verified description of the completed work. Before modernization, a substantial portion of telematics data was processed using slower batch-oriented workflows. During the project, the feasibility of transitioning to a streaming architecture was established. Kafka ingested acceleration, braking, speed, and trip parameter events, while Spark Streaming transformed them into analytics for risk assessment. The suitability of the resulting data was jointly validated by actuarial and data science teams. Internal logs, system configurations, and the results of controlled comparisons are unavailable; therefore, the case supports a reduction in latency relative to the previous batch-based approach without reporting results in seconds, minutes, or percentage values. No independent verification of this outcome was performed.

The analysis distinguished between evidence obtained from the literature, information derived from the professional case, and the proposed components of the LRV-UBI model. Checkpointing, replay, deduplication, idempotent writes, and the isolation of invalid events are components of the author's model. They are not presented as confirmed elements of the client's actual production pipeline.

End-to-end latency was analysed across the following sequential stages: event generation and transmission, ingestion by the streaming platform, queue waiting, transformation into behavioural features, data quality validation, and risk score generation. For each stage, the source of latency, the corresponding control mechanism, and a measurable indicator were identified. The model was evaluated according to the logical completeness of these relationships and the possibility of subsequently measuring end-to-end latency, processing latency, consumer lag, ingestion rate, throughput, event loss, duplicate and late-arriving events, recovery time, data completeness, and

risk-scoring availability time. Experimental measurement of latency reduction was not performed within the scope of this study.

Kafka and Spark Streaming were considered the validated core of the proposed architecture, while Python was regarded as the processing technology used in the implementation. Airflow, AWS, Azure, SQL Server, and Oracle were part of the broader enterprise technology environment; however, their exact relationship to the telematics processing pipeline was not established. For this reason, they were not included in the mandatory critical path of the LRV-UBI model, and the 99.5% availability metric was not attributed to the streaming pipeline examined in this study.

RESULTS

The size of the analysed datasets ranged from 400 drivers [8] to 19,214 drivers [6] and 94,268 vehicles [12]. The actual workload is more accurately characterized by the number of associated records. A total of 1.83 million trips were linked to 1,073 policyholders [3], while monitoring 2,185 trucks produced 105,786 trips and 465,555 safety alerts [15]. The largest dataset contained approximately 426 million records before data cleaning and 395 million records after cleaning [12].

The unit of observation determines the processing structure. A policy links trips to the insurance coverage period, a driver links trips to individual driving behaviour, and a vehicle links them to a sequence of movements. For commercial transport, this structure also includes alerts associated with a specific truck and trip. Feature engineering requires preserving relationships across these data levels; otherwise, indicators from different periods or entities may be incorrectly combined.

In insurance policy datasets, a higher number of insurance claims was associated with large transitions between speed ranges, high maximum speed, nighttime driving, and harsh braking [3]. For weekly accident prediction, speeding, road type, time of day, and features derived from previous observation periods were identified as significant predictors [6]. Driver behaviour classification distinguished accident risk levels more accurately than assessment based on a single composite score [8].

Changes in driving behaviour over time also provide predictive information. The inclusion of temporal features increased the area under the receiver operating characteristic curve from 0.640 to 0.662 and improved balanced accuracy from 0.600 to 0.620 [12]. For commercial transport, the model incorporating in-cabin monitoring alerts outperformed alternative models that excluded these data [15]. Table 1 compares the quantitative characteristics of five telematics studies, including the units and scale of observation, identified risk factors, and the derived requirements for stream processing.

Table 1. Quantitative Characteristics of Telematics-Based Driving Risk Studies (Compiled by the author based on the sources: [3, 6, 8, 12, 15])

Observation unit	Sample size	Observation period and data volume	Risk-related finding	Implication for streaming processing
Policy / policyholder	1.458 / 1.073	Policy start years: 2017–2020; 1.83M trips	Higher claims: speed shifts, maximum speed, night driving, harsh braking	Policy-period alignment; trip aggregation
Driver	19.214	55 weeks; 181.4M km	Speed, road type, time, and lagged factors improved weekly prediction	Weekly keyed windows
Driver	400	Period not reported; 350 train / 50 test	Classification outperformed scoring	Consistent training and live features
Vehicle	94.268 raw / 88.120 clean	2 years; 426M / 395M records	AUC: 0.640 → 0.662; BA: 0.600 → 0.620	Stateful cleaning; temporal features
Truck / trip / warning	2.185 trucks	Period not reported; 105.786 trips; 465.555 warnings	ZIP with warnings outperformed two baselines	Truck-trip-warning joins

The requirements presented in the last column of Table 1 were derived within the framework of the present study based on the structure and volume of the reported datasets. They describe the conditions required for transferring feature engineering methods into a continuous production pipeline rather than the results of architectural evaluations conducted in the original studies.

The comparison of the datasets revealed two sources of complexity. The first is related to data volume. Several thousand insured entities generate millions of trips and hundreds of millions of individual records. The second is determined by data heterogeneity. Indicators are calculated at the levels of the policy, driver, vehicle, trip, and event. A stream processing system must maintain separate state information for these entities, preserve temporal order, and update features as new records arrive.

The reviewed studies confirm the predictive value of telematics but primarily focus on classification quality, prediction accuracy, and the relationship between behavioural features and insurance claims. The time elapsed between event registration and the availability of validated risk assessment is rarely measured in the reviewed literature. Consequently, the demonstrated value of behavioural features has not yet been accompanied by a comparable evaluation of the speed with which they pass through the production pipeline.

The practical component of this study is based on direct professional participation in the modernization of a usage-based insurance programme for a major U.S. auto insurer. The identity of the company, the programme, and its internal systems has been kept confidential. Telematics data were used to assess driver behaviour, but a substantial part of the processing was performed using batch-based workflows.

Data on acceleration, braking, speed, and trip parameters were received continuously, whereas feature generation began only after the accumulation of the next processing batch. This batch boundary increased the interval between the recording of driving behaviour and the availability of data for risk assessment. The verified duration of this interval is not available in the documented professional materials.

During the project, the possibility of transferring processing to a continuous streaming pipeline was identified. Kafka ingested the stream of telematics events, Spark Streaming transformed them into features as they arrived, and Python was used for data processing. Eliminating the need to wait for the next processing batch shortened the interval between event registration and the preparation of analytics for risk assessment.

The development was carried out jointly with actuarial and data science teams. Data completeness, consistency, and the suitability of behavioural features for risk models were verified. The loss of a record reduces the calculated intensity of driving behaviour, duplicate processing inflates it, and incorrect trip assignment distorts the profile of another driver or insurance period. Therefore, latency reduction was evaluated together with the preservation of data quality.

Comparing the professional case with the findings of the literature made it possible to define the reliability requirements for continuous stream processing. A late-arriving event must be incorporated into the appropriate processing period, after which the affected features require revalidation. Temporary source unavailability requires processing to resume after recovery without losing previously accepted records. During periods of peak event arrival, queue growth and consumer lag must be monitored [2]. Following a failure, the system must restore the processing state and prevent duplicate processing of events that have already been handled, consistent with the fault-tolerance requirements of streaming systems [5].

Checkpointing, replay, deduplication, idempotent writes, and the isolation of invalid events are proposed as architectural requirements of the LRV-UBI model. The available evidence does not confirm that each of these mechanisms was implemented in the actual production system.

The transition to Kafka and Spark Streaming reduced the time gap between the arrival of telematics events and the availability of data for risk assessment compared with the previous batch-based approach. This outcome is not expressed in seconds, minutes, or percentage values because

no controlled comparison between the two implementations was performed. Likewise, the professional case does not provide sufficient evidence to claim a quantitative improvement in pricing accuracy or an immediate adjustment of insurance premiums.

DISCUSSION

The workload generated by telematics is determined by the frequency of trips and recorded events rather than by the number of insured entities. In the dataset of Chan et al., each unique policyholder was associated with approximately 1,706 trips [3]. The dataset of Strada et al. contained an average of 4,519 records per vehicle in the original sample [12]. In the commercial transport study, each vehicle corresponded on average to 48.4 trips and 213.1 safety alerts [15]. The average distance travelled in the weekly insurance dataset reached 9,441 km per driver over 55 weeks [6]. These calculated ratios were derived from aggregated values reported in the publications and are not directly comparable; however, they reveal a common architectural relationship. The event stream grows more rapidly than the number of insured entities.

The practical value of timely processing lies in the fact that risk is formed at the level of individual trips and changing driving behaviour. The relationship between harsh acceleration, harsh braking, speed characteristics, and the number of insurance claims has been demonstrated using trip-level data [3]. Weekly context, including time of day and road type, complements behavioural features in accident frequency modelling [6]. Driving behaviour classification distinguishes drivers according to insurance risk levels [8]. Dynamic behavioural features increased accident prediction AUC from 0.640 to 0.662 compared with aggregated indicators [12]. If such features become available only after a prolonged batch-processing cycle, the resulting assessment describes a driver whose behaviour may already have changed.

Reducing technical latency alone does not guarantee higher model accuracy. Rapid processing of incorrect timestamps, missing values, or duplicate events only accelerates the production of distorted results. Therefore, the availability time of risk assessment should be measured after feature validation rather than at the moment when computational processing is completed.

Kafka consumer lag serves as an early indicator of latency accumulation. An increasing gap between the recorded and processed offsets indicates that the event arrival rate has exceeded the current processing capacity of consumers. Monitoring this metric at short intervals makes it possible to detect overload before it affects the timeliness of risk scoring [2]. The interpretation of consumer lag requires contextual information, including the event arrival rate, the number of topic partitions, the number of consumers, and the processing time per event.

Partitioning and parallel execution increase throughput until data exchange or one of the system components reaches saturation. In an experiment using eight processing pipelines and eight Kafka partitions, the achieved speedup reached 7.19×; however, at workloads exceeding 200 MB/s, Kafka became the limiting component [7]. Results obtained from an edge-stream architecture demonstrate a different type of limitation. The intermediate Kafka Connect layer sharply increased data loss as event throughput grew [13]. Eliminating intermediate disk writes reduces the number of operations required before transmitting an event to Kafka, although no direct quantitative comparison of latency was reported in the corresponding study [4]. Scaling the computational cluster supports higher workloads but does not replace state management or data correctness control [14]. Table 2 compares quantitative indicators of latency, throughput, data loss, and monitoring across the selected stream processing systems.

Table 2. Reported Latency, Throughput, Loss, and Monitoring Parameters of Selected Streaming Implementations (Compiled by the author based on the sources: [1, 2, 7, 13, 14])

Application domain	Workload and configuration	Reported latency or processing interval	Throughput, loss, or recovery observation	Implication for the LRV-UBI model
Manufacturing	Kafka; Spark Streaming; Parquet	10 min	Data volume: -95%; final/original ratio: 5%	Processing interval: 10 min; storage ratio: 1:20
IoT monitoring	3 brokers; retention: 5 d; 4 scenarios; 1 month	Kafka metrics: 10 s; service checks: 30 s	Simulated outage: 24 h; outage/retention: 20%	Lag sampling: 10 s; availability sampling: 30 s
Video streaming	8 pipelines; 8 Kafka partitions; load >200 MB/s	<60 s for most 60 s videos	Maximum speedup: 7.19×; scaling efficiency: 89.9%	Pipeline/partition ratio: 1:1; Kafka saturation: >200 MB/s
Edge processing	1,000–14,000 msg/s; Kafka Connect vs direct path	52.2→19.8 ms; Δ: -32.4 ms; -62.1%	Connect loss: 16.3% at 2,000 and 90.6% at 14,000 msg/s; direct loss: ≈1.5% at 14,000 msg/s; ≤1% at 12,000 msg/s	Loss gap at 14,000 msg/s: -89.1 pp*; validated range: 1,000–14,000 msg/s
Event subscriptions	140,000 subscribers; 2→16 nodes; 160 events/s	Runtime: ≈-50% per 2× nodes	Load scaling: 80→640 events/s; performance remained relatively stable	Nodes: 8×; load: 8×; scaling steps: 3

Note: msg/s – messages per second. MB/s – megabytes per second. Pp – percentage points. Values marked with an asterisk were calculated in the present study from the reported source data.

The quantitative findings reveal several important relationships. The use of streaming technology alone does not guarantee low latency. A recording interval of 10 minutes still preserves temporal processing granularity. Frequent state monitoring also does not guarantee recovery, but it reduces the time required to detect consumer lag or system failures. The nearly linear speedup achieved through the coordinated increase in the number of processing pipelines and Kafka partitions ends once Kafka reaches saturation. The most pronounced difference is associated with the intermediate processing component. At a throughput of 14,000 messages per second, the gap between the compared implementations reached 89.1 percentage points in terms of data loss.

The analysed implementations originate from manufacturing, video streaming, edge computing, Internet of Things monitoring, and subscription-based systems. Their absolute performance values cannot be directly transferred to an insurance telematics pipeline because message size, processing logic, hardware configuration, and completion criteria differ substantially. These results demonstrate general relationships between system configuration, workload, latency, and data loss but do not define target performance thresholds for the LRV-UBI model.

A continuous processing pipeline must preserve correctness despite out-of-order events and temporary source unavailability. After recovery, accumulated records create a sudden increase in input workload, while duplicate event delivery may alter features that have already been computed. In the LRV-UBI model, Kafka consumer offsets define the confirmed processing boundary, Spark Streaming checkpoints preserve the computational state, and replay resumes processing from a known position. Consumer lag indicates whether the system is keeping pace with the accumulated event queue.

A repeatedly received event is matched against a persistent event identifier before behavioural features are updated. An invalid record is isolated and excluded from risk assessment until it has been corrected. If a corrected or late-arriving event belongs to an already processed time window, only the affected features are recalculated. Protection of the output stage against repeated execution prevents duplicate publication of the same risk assessment after system recovery.

The operation of these mechanisms requires operational definitions. The proportion of late-arriving events is calculated among accepted records according to the predefined lateness threshold. The proportion of duplicate events is determined using a persistent event identifier, while the proportion of rejected records is calculated from the results of data quality validation. Recovery time covers the interval from failure detection to the resumption of processing and the return of consumer lag to its baseline level. Risk assessment availability time is measured from the original event timestamp to the authorized publication of the assessment result.

These mechanisms belong to the proposed LRV-UBI model. The available evidence from the professional case confirms the use of Kafka, Spark Streaming, and Python, as well as joint data validation with actuarial and data science teams. However, it does not confirm the implementation of checkpoints, deduplication, replay, or output idempotency within the actual production pipeline.

The quantitative evaluation of the client solution is limited by the absence of internal logs and technical reports. The actual frequency of telematics events, latency distributions (p50, p95, and p99), recovery time, and data loss were not documented. The batch-based and streaming implementations were not experimentally compared using the same input stream. Therefore, the professional case confirms a reduction in latency relative to the previous batch-based approach but does not establish its quantitative magnitude.

The quantitative characteristics of the analysed streaming systems were obtained from different application domains under non-comparable workloads. The architectural evaluation of the LRV-UBI model assesses its logical completeness and the measurability of its components; demonstrating an actual reduction in latency requires a separate experimental study. The available evidence does not support a quantitative relationship between stream processing and improved pricing accuracy or the immediate adjustment of insurance premiums.

The LRV-UBI model can serve as a design framework for other continuous scoring systems that process high-frequency events and publish results only after data validation. The specific system configuration depends on the event structure, workload volume, acceptable waiting time, data quality rules, the underlying risk model, and regulatory requirements. These dependencies preclude interpreting the model as a universal architecture.

The LRV-UBI model integrates data flow, operational reliability, observability, and data validity. The primary management unit is a telematics event containing a unique identifier, the source timestamp, and trip-related features. The endpoint of the processing pipeline is the controlled publication of a validated risk assessment.

The data pipeline begins with vehicle or mobile telematics. Initial validation verifies the data format, timestamp, and mandatory fields, after which Kafka ingests the event and assigns it to the appropriate topic partition. Spark Streaming generates behavioural features that are subsequently validated for completeness, admissibility, and consistency. Risk assessment is published only after successful completion of data quality validation. This authorization step separates the computed value from a result considered suitable for insurance use.

Reliability mechanisms are associated with specific processing

stages. Source recovery corresponds to telematics ingestion, offset management belongs to Kafka, and checkpointing is performed within Spark Streaming. Replay and deduplication are executed before the final update of behavioural features. Invalid events are routed to an isolated area, while corrected data are processed through controlled recalculation. Output idempotency is enforced at the publication stage of the processing pipeline.

Observability covers the entire lifecycle of an event. Source-to-ingestion latency characterizes telematics transmission. Ingestion rate reflects Kafka performance. Consumer lag represents the state of the processing queue. Processing latency characterizes Spark Streaming transformations. The proportion of rejected events and data completeness describe

the outcome of data quality validation. Risk assessment availability time ends only after authorized publication; therefore, internal computation alone cannot prematurely improve the final performance indicator.

Actuaries and data science specialists refine the rules for feature generation and the criteria for data quality validation based on model behaviour. This feedback does not directly modify an individual risk assessment. Instead, it is incorporated through controlled updates of the processing rules followed by validation, thereby preserving the reproducibility of the risk-scoring process. Figure 1 presents the author's LRV-UBI model, integrating the data stream, reliability mechanisms, and validation control for insurance risk assessment.

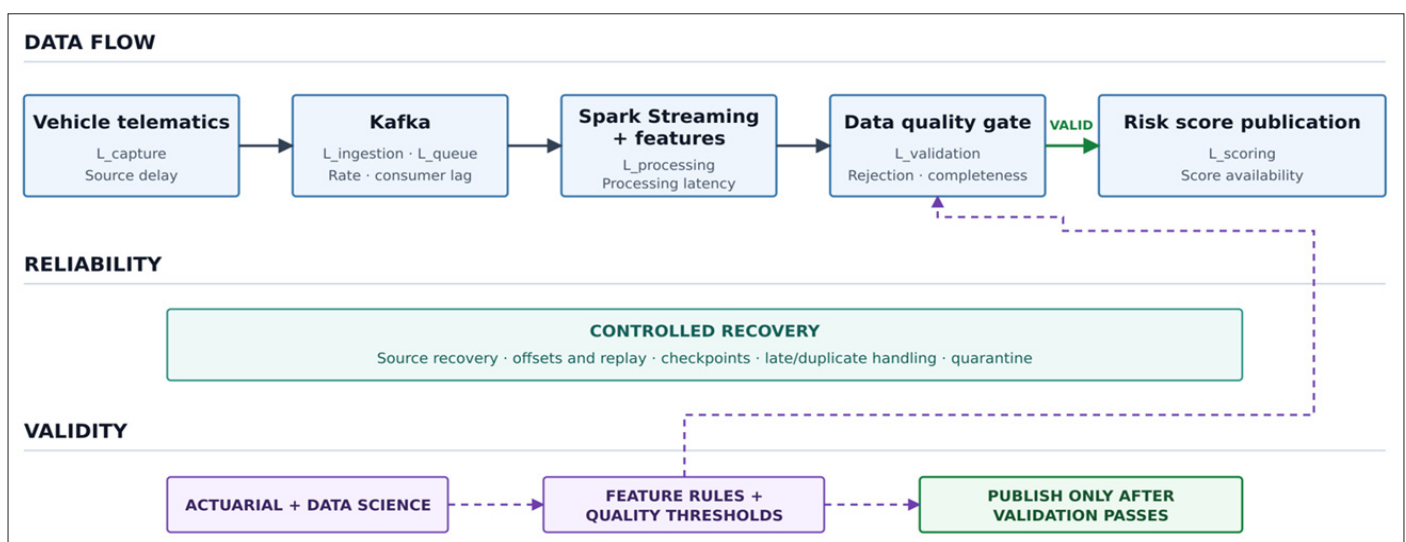


Figure 1. The Proposed LRV-UBI Model Linking Latency Components, Reliability Controls, and Risk Assessment Validity (Developed by the author)

Airflow, AWS, Azure, SQL Server, and Oracle are not included in the model. They belong to the broader enterprise technology environment, and their direct involvement in the telematics processing pipeline considered in this study has not been independently established.

Quantitative validation of the LRV-UBI model should use the same telematics data stream for both batch-based and streaming implementations. The workload should be applied sequentially at 1×, 2×, and 4× relative to the baseline event rate. Temporary source unavailability, late-arriving events, and duplicate events should be modelled separately. For each operating mode, p50, p95, and p99 end-to-end latency, throughput, data loss, duplicate processing, recovery time, the proportion of rejected records, and risk assessment availability time should be measured. Such an experimental design would make it possible to verify latency reduction without confounding the architectural effect with differences in the input data.

CONCLUSION

The speed of insurance risk assessment depends on the design of the entire telematics event processing pipeline.

Batch processing introduces latency even before computation begins because data must wait until the next processing batch is formed. A streaming architecture eliminates this waiting period; however, its advantage is maintained only while the ingestion and processing rates keep pace with the incoming workload. Consumer lag, saturation of individual system components, and unnecessary intermediate operations can reintroduce latency within a streaming architecture. Therefore, increasing computational resources without controlling queue dynamics and pipeline configuration does not ensure stable performance.

An insurance risk assessment should be considered suitable for operational use only after verification of the completeness, consistency, and admissibility of the generated behavioural features. The completion time of computational processing alone does not represent the actual availability of the result. Lost, duplicate, or late-arriving events require state recovery, deduplication, or controlled recalculation. These procedures form part of the critical processing path and must therefore be included when measuring end-to-end latency. This interpretation prevents situations in which processing speed

is improved at the expense of the quality of the input data used by the risk model.

The de-identified professional case confirms that replacing batch processing with Kafka and Spark Streaming can reduce the time gap between the arrival of telematics events and the preparation of analytics for insurance risk assessment. The literature explains this effect through partitioning, parallel execution, consumer lag monitoring, and the elimination of unnecessary intermediate processing stages. Taken together, these findings support the proposed hypothesis at the level of architectural synthesis. However, the magnitude of latency reduction cannot be determined from the available evidence because the batch-based and streaming implementations were not evaluated using the same input stream under comparable workloads.

The LRV-UBI model associates each processing stage with a dedicated latency metric and a corresponding control mechanism while linking the publication of risk assessment to successful completion of data quality validation. Its contribution lies in the operational integration of stream processing, state recovery, and telematics feature validation within a unified insurance processing framework. Quantitative validation of the model should compare batch-based and streaming architectures using identical input data while measuring median and tail latency, throughput, data loss, duplicate processing, recovery time, and the availability of validated risk assessment. Until such an experiment is conducted, the LRV-UBI model should be regarded as a substantiated conceptual design that defines verifiable conditions for latency reduction rather than as an empirically validated production architecture.

REFERENCES

1. Cakir, A., Akın, Ö., Deniz, H. F., et al. (2022). Enabling real time big data solutions for manufacturing at scale. *Journal of Big Data*, 9, 118. <https://doi.org/10.1186/s40537-022-00672-6>
2. Calderon, G., del Campo, G., Saavedra, E., et al. (2024). Monitoring framework for the performance evaluation of an IoT platform with Elasticsearch and Apache Kafka. *Information Systems Frontiers*, 26, 2373–2389. <https://doi.org/10.1007/s10796-023-10409-2>
3. Chan, I. W., Tseung, S. C., Badescu, A. L., & Lin, X. S. (2024). Data mining of telematics data: Unveiling the hidden patterns in driving behaviour (arXiv:2304.10591v2). *arXiv*. <https://doi.org/10.48550/arXiv.2304.10591>
4. Costin, A.-T., Zinca, D., & Dobrota, V. (2023). A real-time streaming system for customized network traffic capture. *Sensors*, 23(14), 6467. <https://doi.org/10.3390/s23146467>
5. Fragkoulis, M., Carbone, P., Kalavri, V., et al. (2024). A survey on the evolution of stream processing systems. *The VLDB Journal*, 33, 507–541. <https://doi.org/10.1007/s00778-023-00819-8>
6. Guillén, M., Pérez-Marín, A. M., & Nielsen, J. P. (2024). Pricing weekly motor insurance drivers' with behavioral and contextual telematics data. *Heliyon*, 10(16), e36501. <https://doi.org/10.1016/j.heliyon.2024.e36501>
7. Kastrinakis, D., & Petrakis, E. G. M. (2023). Video2Flink: Real-time video partitioning in Apache Flink and the cloud. *Machine Vision and Applications*, 34, 42. <https://doi.org/10.1007/s00138-023-01391-5>
8. Liu, Z., Hao, M., & Tian, F. (2022). Ratemaking model of usage based insurance based on driving behaviors classification. *International Journal of Crowd Science*, 6(2), 98–109. <https://doi.org/10.26599/IJCS.2022.9100012>
9. Nai, W., Yang, Z., Wei, Y., Sang, J., Wang, J., Wang, Z., & Mo, P. (2022). A comprehensive review of driving style evaluation approaches and product designs applied to vehicle usage-based insurance. *Sustainability*, 14(13), 7705. <https://doi.org/10.3390/su14137705>
10. Qi, J., Woolway, M., Kyropoulou, M., Kanellopoulos, P., & Steele, D. (2025). Telematics in insurance: Challenges and limitations. *IEEE Access*, 13, 147449–147466. <https://doi.org/10.1109/ACCESS.2025.3600095>
11. Shirole, V., Shahade, A. K., & Deshmukh, P. V. (2025). A comprehensive review on data-driven driver behaviour scoring in vehicles: Technologies, challenges and future directions. *Discover Artificial Intelligence*, 5, 26. <https://doi.org/10.1007/s44163-025-00244-6>
12. Strada, S., Costantini, E., Formentin, S., & Savaresi, S. M. (2024). Machine learning based car accident risk prediction for usage-based insurance. *Intelligent Data Analysis*, 29(1). <https://doi.org/10.3233/IDA-230971>
13. Vitorino, J. P., Simão, J., Datia, N., & Pato, M. (2023). IRONEDGE: Stream processing architecture for edge applications. *Algorithms*, 16(2), 123. <https://doi.org/10.3390/a16020123>
14. Wang, X., Carey, M. J., & Tsotras, V. J. (2022). Subscribing to big data at scale. *Distributed and Parallel Databases*, 40, 475–520. <https://doi.org/10.1007/s10619-022-07406-w>
15. Zhang, X., Wang, X., Bao, Y., & Zhu, X. (2022). Safety assessment of trucks based on GPS and in-vehicle monitoring data. *Accident Analysis & Prevention*, 168, 106619. <https://doi.org/10.1016/j.aap.2022.106619>