SCTE CABLE-TEC EXPO 22
SEPTEMBER 19-22 • PHILADELPHIA, PA

2022 Fall Technical Forum
SCTE • CABLELABS • NCTA

Explainable AI for Data Clean Room Query Validation

A Technical Paper prepared for SCTE by

Srilal Weera PhD
Principal Engineer
Charter Communications
720-699-5079
srilal.weera@charter.com

Table of Contents

List of Figures

List of Tables

1. Introduction

Sophisticated ML models function more or less as black-boxes. A neural network may easily classify a photo of an animal as a cat or dog, but is silent about *why* it made that decision. A recent development is *Explainable AI* (XAI), also called *Interpretable AI*. Dubbed as an enabler for 'third-wave of AI', it helps open up the black-box model [1][2]. XAI has found niche applications in many industries. For example, in credit-risk analysis it is common practice to use machine learning models. If a loan application is denied then XAI can further reveal the reasons why it was deemed risky. Another scenario is in product recommendations. XAI could bring to light the contributing factors as to why a certain product was recommended to a specific customer.

In spite of its prowess, XAI applications in cable industry have been lacking thus far. In this paper, we present a timely application that reflects broad global interest in ways to share customer data in a privacy-compliant way.

An emerging solution is the Data Clean Room (DCR) concept [3]. Its goal is to provide a safe place for partnering companies to bring respective data for analysis in a secure manner. Guidelines are established to restrict any sensitive queries to protect the customer identity. However, sensitive querying may occur unintentionally due to micro-targeting. This is ascribed to how the queries are constructed (e.g. too many conditions in the SQL filter). Since the queries have originated from credible sources, blocking them entirely is not desirable. A pragmatic solution would be to assess and relax the query sensitivity which would lead to efficient database querying. This can be achieved with XAI enabled machine learning, Additionally, the query sensitivity scores can be used to fine-tune the privacy mechanisms. This is illustrated with reference to leading privacy technologies.

[*Please Note*: Charter has a longstanding commitment to protecting the privacy and security of its customers. For example, Charter provides customers with detailed information about its privacy practices, explicitly allows customers to opt out or change sharing preferences at any time, and restricts the collection of information (when enabled by the customer) to what is necessary to provide and optimize service. Learn more at Spectrum.com/Privacy.]

2. Explainable AI – Brief Overview

Explainable artificial intelligence (XAI) attempts to answer the 'why?' question about machine learning models. *Explainability* has gained much attention recently as its potential for trustworthy AI is recognized. A common algorithmic technique is to slightly change (perturb) a single feature at a time and measure the impact on the model prediction.

2.1. How it works

To illustrate how XAI works, consider the widely available Titanic dataset listing the survival rate of passengers. XAI analysis results are shown in Figure 1 (see reference [4]). The passenger survival rate is high on the right half of the diagram (positive values) and vice versa.

The XAI (SHAP) analysis reveals the survivors were either mostly female or paid higher fares or belonged to young age group. The 'embarked destination' seems to have little impact on survival rate.

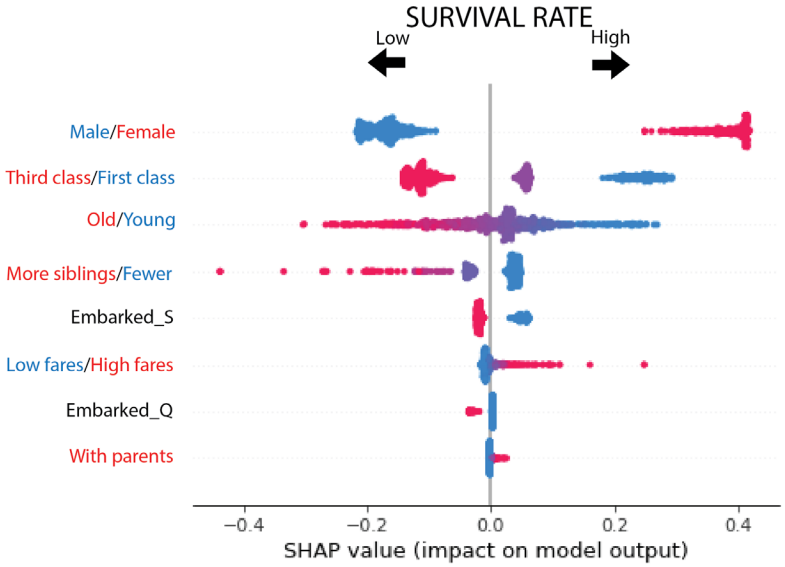

Figure 1 - XAI Example - Titanic survival analysis

2.2. XAI algorithms

The well-known XAI algorithms are:

- Local Interpretable Model-agnostic Explanations (LIME)
- Shapley Additive explanations (SHAP)
- Partial Dependence Plot (PDP)

LIME and SHAP are called surrogate models because they attempt to approximate the predictions of the underlying black-box model. The algorithms tweak the input slightly and test the changes in prediction. If there is appreciable change in the predicted value, then that input variable is considered to have a higher impact on the model prediction. Other techniques include gradient-based saliency maps for image analysis. LIME generates a new dataset by randomly turning the data points (pixels/words) on or off. Hence it is a local approximation to the black-box ML model. The surrogate models are model agnostic since they treat the ML model as a black box.

SHAP has origins in game theory. Corresponding to a game, each feature is considered a 'player' and the prediction is the prize money. In a game, Shapley values determine how to assign payouts to players in proportion to their contributions to the prize money. In machine learning it is the contribution by each feature to the final model prediction. SHAP algorithm determines the average marginal contribution of a feature to the model prediction.

PDP provides a graphical representation of how each feature affects the prediction in a machine learning model. Each feature value is changed in ascending order and the corresponding change in the prediction is plotted. A partial plot depicts the dependency of the target response over the range of input features. One limitation is that PDP assumes no correlation between input variables, an assumption which is not always realistic.

3. Data Clean Room

Recent privacy regulations and app tracking transparency frameworks reflect a growing trend requiring explicit user consent for tracking. Some major companies are also ending the support of third-party cookies and identifiers, making it harder to run effective campaigns or measure attribution. The challenge for businesses is how to share consumer data for analytics without compromising consumer privacy.

An emerging solution is the Data Clean Room (DCR) model. Its goal is to provide a safe place for partnering companies to bring respective data for confidential analytics. Security and privacy-protection measures are applied, such as data anonymization, obfuscation and differential privacy. All data stays within the data clean room and is not shared with outsiders (Figure 2). Guidelines are established to restrict any sensitive queries to protect the identity of customers in the database [6]. One drawback, however, is that when a query is blocked the analyst (querier) has to reconfigure the filters in the query string and resubmit until the blocking is removed. Since the queries have originated from credible sources, blocking them entirely is not desirable.

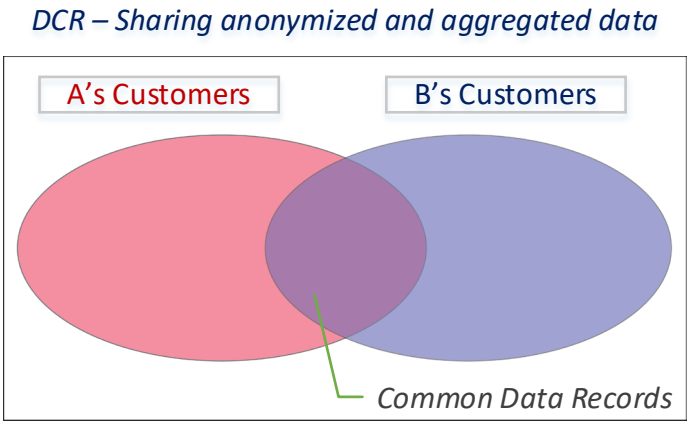

Figure 2 - Data Clean Room concept

Another application scenario is programmatic ad-buying, in which the ad-spaces are bid in real-time on 'ad exchanges', (*cf.* stock exchanges in finance). The highest bidder wins the auction and places the ad alongside the main content. For more effective targeting, a matching audience need to be identified and is done via querying the publisher database. Multiple advertiser proxies are involved in this process and all such entities may query a publisher database to build profiles. Similar to DCR, a common practice is to block any queries that are deemed sensitive.

Secure multi-party computation (MPC) is a cryptographic technique that secures the privacy of data even in collaborative computing. Some DCRs advocate the use of MPC. Confidential Computing is a related paradigm to protect the data 'in use' (compared to data at-rest and data in-transit). Additionally, it may also involve the siloed enclaves with memory partitions and containerized abstractions.

4. Query Validation

Validating a database query prior to execution is done for several reasons:

1. To correct the syntax of a query
2. To prevent hacking attacks targeting security vulnerabilities (e.g. SQL injection)
3. To thwart privacy attacks designed to divulge sensitive data

The third item is the focus of this analysis. While the stored data remain anonymous, sensitive queries could divulge personal data and could pose a privacy risk. As such, a common practice is to block any queries that are perceived as sensitive. This practice however reduces the *utility value* of the database. Also, it disrupts certain business models where a multitude of queries are generated from credible sources.

One such scenario is DCR, where each party query the partner databases to build user profiles. Note that the querier in this scenario is not an adversary but a partnering company. As such, sensitive querying may occur unintentionally due to micro-targeting. This is ascribed to how the queries are constructed, such as having too many conditions in the defined filter. Simply blocking or invalidating such queries is costly and inefficient. It also increases traffic to the database due to repeated attempts of failed queries.

Accordingly, there is a need for an enhanced query validation method to:

1. Notify the querier why a query was deemed sensitive and suggest improvements
2. Determine the conditions to be relaxed for a failed query to be executed successfully
3. Assess the query sensitivity and use it to enhance privacy protection mechanisms

To accomplish this, it is necessary to first identify the factors which contributed to blocking the query from execution. In general, the purpose of a query is to retrieve data records that fulfill a specified criteria. It becomes a *sensitive query* if the query string is constructed in such a way that it could divulge privacy-sensitive data. For example, if a query returns only a handful of matching records, releasing such data may be a privacy risk for the affected individuals or the cohort. Also, in this scenario injecting statistical noise to mask the query response could skew the results. A more appropriate solution would be to devise a way to *relax* the query criteria to enable (unblock) query execution. This applies to other types of querying scenarios such as in graph databases.

In the ensuing sections, a machine learning-based solution is outlined for identifying sensitive queries. Explainable AI techniques are then used to obtain the reasons why the query was deemed sensitive. A distributed workflow is presented for relaxing the query criteria and deriving a quantified query sensitivity, which is then used as input for privacy settings.

4.1. Query Relaxation Example

Relaxing the search criteria constraints will unblock a sensitive query. For example, assume that in the first pass XAI indicated three possible reasons for the blockage.

Blocked by: Ethnic code = Danish, Age group = 50 – 55, Income bracket = 100k – 110k

These conditions are then relaxed iteratively and the query is run recursively. Table 1 shows the contributing factors for blockage at each pass.

Table 1 - Query relaxation Iteration

Query Iteration	ML Outcome	XAI Outcome (Reasons for blocking)
First Pass	query blocked	Ethnicity, Age, Income
Second Pass	query blocked	Ethnicity, Age,
Third Pass	query blocked	Age
Fourth Pass	query validated!	

For example, the age group can be expanded gradually by ±5 years.

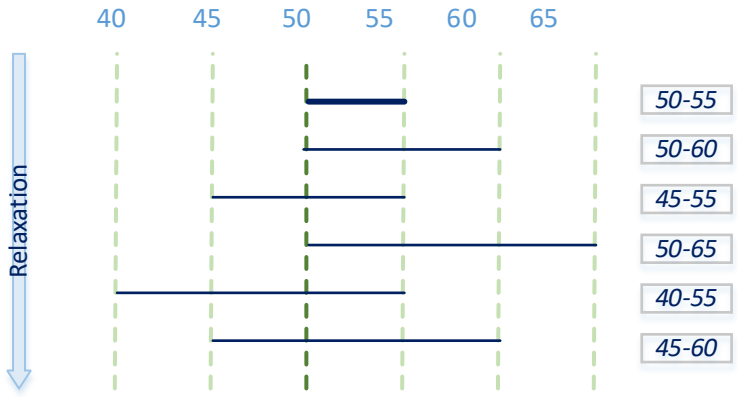

Figure 3 - Query filter relaxation example

Similarly, the ethnic code can be relaxed as in following steps:

1st try – Ethnic Code = Danish
2nd try – Ethnic Code = Danish +Swedish
3rd try – Ethnic Code = all Scandinavian countries
4th try – Ethnic Code = all Nordic countries
5th try – Ethnic Code = Nordic + Baltic countries

The above steps are continued till the blockage is cleared, or until the max depth is reached (as defined in the rules engine and database schema). By relaxing the conditions, a valid query can be achieved. This is vital to the querier as it avoids the trial and error method of submitting multiple queries and getting a "Your query was blocked" error message. Instead, the ML-XAI based solution provides a more productive response: "Your query was deemed sensitive and blocked. The following conditions may need to be relaxed…" Additionally, (if sanctioned by the contract), the query is run with relaxed conditions on the main database. The results (with noise injected), are supplied back to the querier.

5. Machine Learning Module

ML engines are generally trained on the data gathered externally, such as in transfer learning. In the present case, the training data resides in the main database itself. While database-integrated ML engines already exist (e.g. Amazon Aurora and Google BigQuery), the usage described below is different: In addition to query retrieval, its sensitivity is also analyzed. For this task, a collection of queries over a wide range of conditions is amassed and run on the main customer database. The outputs are then classified and graded according to sensitivity. This action is based on Rules Engine settings, additionally supported by human expert analysis. Database and privacy experts review the queries and classify those based on the perceived sensitivity. The outcome is a gradation based on the level of sensitivity. The neural ML engine is trained (weights adjustment), with the classified queries from the query database. Once trained, the ML engine is able to classify whether a fresh query is sensitive or not.

5.1. Machine Learning Model Training Steps

1. Raw queries are collected from multiple sources.
2. Query database is formed with 'unclassified' queries.
3. Queries are run on the main database (which contains user data records).
4. The query outputs are analyzed for sensitivity per rules engine settings.
5. Queries are then classified (graded), according to sensitivity.
6. The sensitive queries are submitted to the XAI module. Reasons for blocking are derived.
7. The 'unclassified' queries are updated with sensitivity data.
8. The 'classified' queries are used to train the ML engine.

The classified* query strings form the inputs for training the ML engine. Additionally, XAI layer provides the explanation why certain queries were considered sensitive. The latter is used for further analysis and refinement of the model. (*the term *classified* here means, 'assigned to different classes'.)

Figure 4 illustrates functioning of the ML Engine for the case of a supervised neural network. The categories (fields) in the database form the input layer. The output layer contains the predicted sensitivity of the query. It could be a binary y/n type or a graded value as shown for more granular assessment.

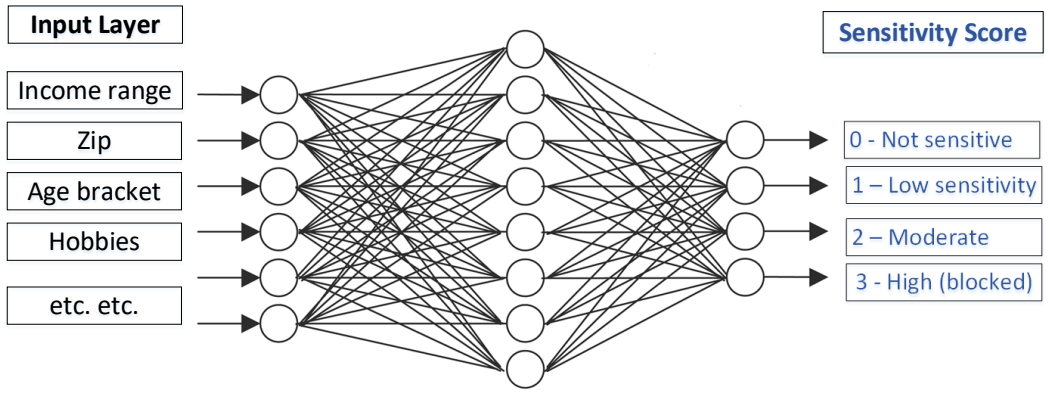

Figure 4 - ML classification of graded query sensitivity

Referring to Table 2, queries form the leftmost column (usually thousands or millions of entries/rows). The next set of columns refer to the types of records in the database, typically ranging from hundreds to several thousand columns. Each row indicates a data record. Column heads are the fields in the database searched by the query. The Sensitivity columns on the right are derived during the training phase as described above. The 'Y's in the column indicate blocked queries that were perceived as privacy compromised. The middle column is a more granular representation of sensitivity. Instead of a binary (Y/N) outcome, the sensitivity is represented as a graded score based on the severity of privacy risk. The last column indicates the assessment from XAI for sensitive/blocked queries.

Table 2 - Sensitivity Analysis - Example

Queries	**Types of Records in the Database**					**Sensitivity score** (binary)	**Sensitivity score** (graded) 1- 5	**Sensitivity analysis (XAI)**
	Income (X1)	Age (X2)	Zip (X3)	Race (X4)	etc. etc.			
Query 1	100-110k	50 - 55	12345	Danish		Y	4	Reason
Query 2								
etc.								
etc.								

5.2. Constrained Optimization

In a typical database, there could be hundreds or even thousands of columns each representing a feature. Without the XAI analysis, it is not easy to pinpoint which feature contributed more for the query sensitivity. Formulating it as a multivariate discrete optimization problem:

If the input variables are denoted by $\mathbf{X_i}$ and the target variable (query sensitivity) as 'y'.

$$\mathbf{X_1} \cap \mathbf{X_2} \cap \mathbf{X_3} \cap \ldots \mathbf{X_n} < y$$

The constraints for each variable: $a_1 < \mathbf{X_1} < b_1$, $a_2 < \mathbf{X_2} < b_2$, $\ldots a_n < \mathbf{X_2} < b_n$

Then to relax the conditions, each X_i is incremented by $\pm\Delta X_i$.

$$(X_1 + \Delta X_1) \cap (X_2 + \Delta X_2) \cap \ldots \cap (X_n + \Delta X_n) \geq y$$

The set of minimum ΔX_i values which meet the constraints would be the optimal result.

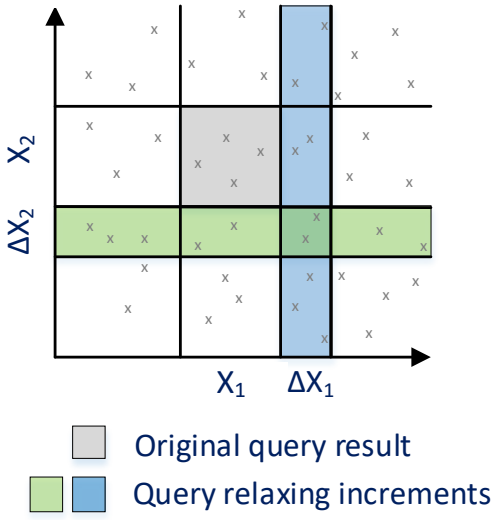

Figure 5 - Incremental query relaxation

For example, consider the simplified case of 2 variables.

Age constraint ($\mathbf{X_1}$) 50 < Age-group < 55

Income constraint ($\mathbf{X_2}$): 100k < Income-bracket < 110k

Assume the sensitivity boundary is 'y=10'. That is, any compound query that returns less than 10 records would be blocked as sensitive.

Age-group ∩ **Income-bracket** < 10

As the field ranges are extended/relaxed, at one point the query will return 10 or more results. i.e. the query is no longer sensitive. It is necessary to find the lowest increments for each variable to meet that criterion. For example, simply expanding the Age-group broadly to 30-70 years might return a large number of records but would be irrelevant. While it may preserve the privacy (query no longer sensitive), the query result will have no utility value to a marketer who is targeting 50-55 age group.

Finding which variable(s) and by how much (ΔX) "minimally" to tweak each is an optimization problem. It gets complicated when dozens or hundreds of variables are involved.

5.3. Model specific considerations

The ML module is based on common classification algorithms such as neural and statistical models. During the training phase, queries with known outcomes (sensitive/not-sensitive) are used to update the weights in the ML engine. (Weights of each link determine the impact of a category). Once trained, the ML engine can classify whether a fresh query is sensitive or not. If the data is volatile, then frequent training would be needed. Statistical ensemble methods such random forests and XGBoost are also options. For more than two outcomes, multinomial logistic regression is another paradigm.

6. Solution Architecture

Process Steps:

1. The Query Controller submits the initial query to the (trained) ML engine.
2. If the query is deemed sensitive, it is run through the XAI module.
3. XAI supplies the reasons for blocking. The querier may be notified.
4. Sensitivity conditions in the string are relaxed per rules engine iteratively.
5. Query is submitted to ML engine recursively until it passes the sensitivity test.
6. The validated query is run on the main database.
7. Query response is assessed for privacy risks and noise injected.

6.1. Workflow

The workflow for the query validation solution are presented below.

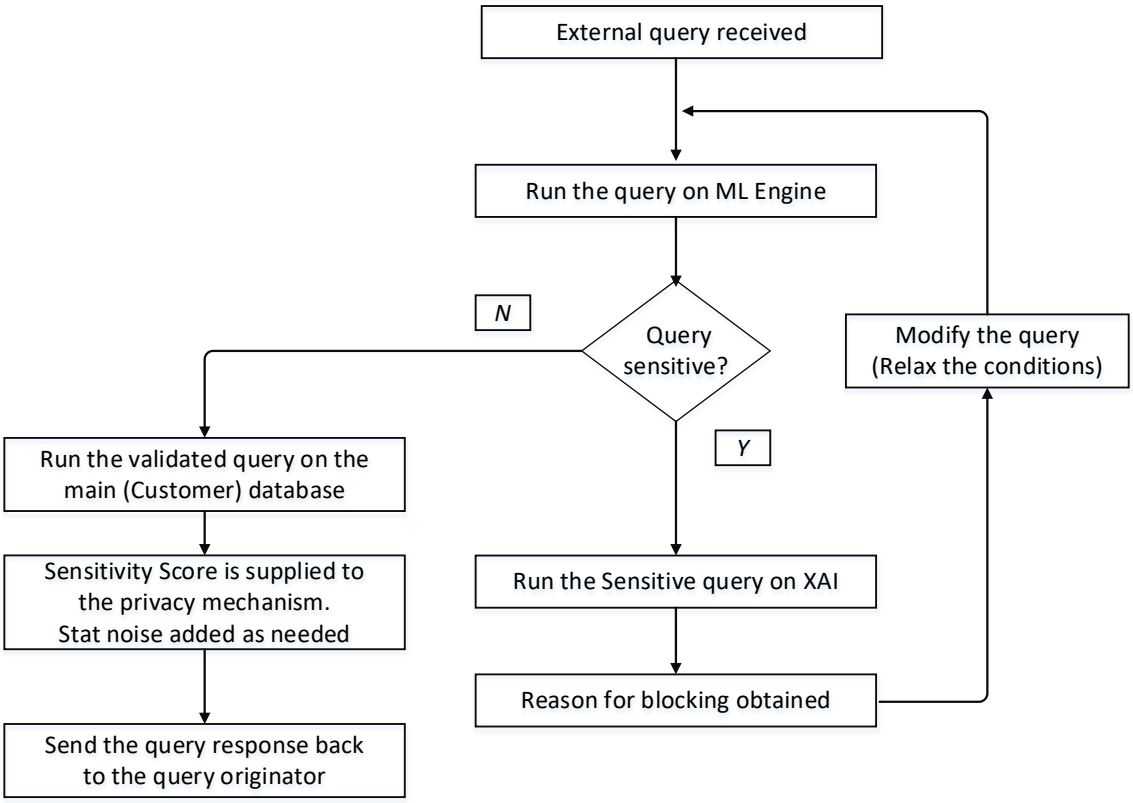

Figure 6 - Enhanced query validation workflow

6.2. Functional components

Figure 7 describes the functional components of the distributed solution. The ML engine, XAI module and the rules engine work in conjunction to define, detect and quantify the query sensitivity. While functionally separate, the ML Engine and XAI are integrated components. XAI can be considered a functionality on top of the machine learning layer and is used recursively with the ML Engine. Once XAI is invoked, it parses the input query and supplies the reasoning for the ML classification.

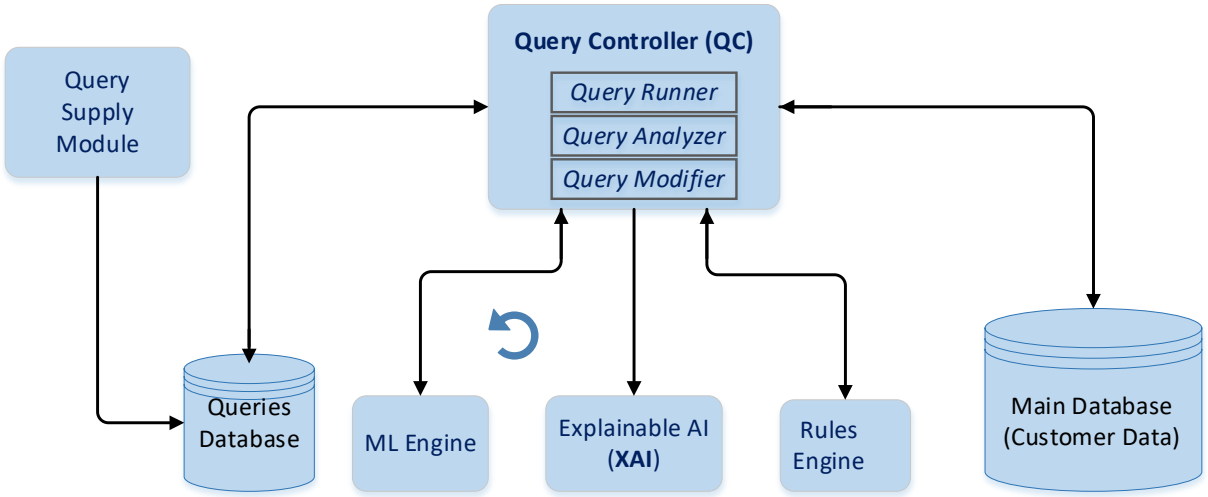

Figure 7 - Solution components

Main Database (Customer Data) – A standard database, such as RDBMS or NoSQL type, either central or distributed. Queries are run on the database and the outputs are classified for sensitivity, then used to train the ML engine.

Query Supply Module – To populate the 'Queries database', queries are supplied in several ways.

- Historical data (a collection of previously run queries)
- Queries obtained from an external entity for training purposes. (Transfer learning is a well-known machine learning training paradigm, in which external data is used first to train an ML engine. Then local data is used to fine-tune the algorithm.)
- Boosting and resampling is a standard ML technique to generate data. Also, the use of a sub-module that auto-creates new queries by modifying the existing queries in the database. Suppose that an existing query has the qualifiers: area code, age group, income bracket, vehicle driven and hobby. Then it is possible to auto-change one qualifier at a time and test how the sensitivity changes. The query is run against the actual data in the main database to classify it as sensitive or not.

Queries Database – The queries database is populated by the Query Supply Module, mentioned above. These queries are used for training the machine learning engine.

Rules Engine – Rules are constructed to define instructions, thresholds etc.

Examples:
- A query is considered sensitive if the query response contains less than five records.
- A query is considered sensitive if a combination of certain categories are present in the query string.
- A query is considered sensitive if similar (coordinated) queries originated from the same source.

Query Controller (QC) – QC module interacts with others systems to automate process flows as well as perform decision making based on the Rules Engine settings. It consists of three subsystems described below.

Query Runner subsystem – Queries (retrieved from the 'Queries database') need to be run on the Main database to train the ML engine. First, each query result is classified as sensitive or not. This can be done based on the settings of the rules engine or with expert input, as it is done offline. During normal operation (after the ML engine is trained), Query Runner's function is to run queries submitted by external entities (queriers).

Query Analyzer subsystem – Analyzing the query for sensitivity based on Rules Engine settings.

Query Modifier subsystem – A software construct with automated processing capability. Assume a query was deemed sensitive and the specific reason for blocking was supplied by XAI module. The Query Modifier will relax the constraint(s) (based on Rules Engine settings) and run the query iteratively until it passes the 'sensitivity test'.

7. Application to Privacy Mechanisms

'Differential Privacy' (DP) is a statistical technique for protecting individual privacy during database querying. When a query is run on a database, DP adds a carefully chosen amount of noise/perturbation is added selectively to the result masking the user identity. For example if the query is to find those who subscribed to a certain TV channel, then some of the user responses (yes/no) are flipped randomly. This gives rise to a new concept called '*plausible deniability'*. So by looking at the data, it is not possible to establish if that data is truly associated with a person or randomly generated. The flip side of adding noise is the need to strike a balance between *utility* and *privacy*. One cannot be enhanced without compromising the other.

Algorithmic basis: Given two neighboring datasets D and D' differing by one data record, the randomized function *K provides* ε-differential privacy when the following probability condition (denoted by 'Pr') is satisfied for all S ⊆ Range(K).

$$Pr[K(D) \in S] \leq exp(\varepsilon) \times Pr[K(D') \in S]$$

ε (epsilon) denotes the privacy loss. For small values, $exp(\varepsilon) \sim 1+\varepsilon$. Note that *differing by one data record* implies that it would not make a difference whether one individual's data are included in the dataset or not. The query result would still be within the statistical error margin. The definition of sensitivity in DP is tied to this formalism.

Small ε (more noise) – better privacy but low utility

Large ε (less noise) – low privacy but high utility

Figure 8 shows the query validator functionality (in schematic form) as input to the DP privacy module. Statistical/random noise is added to the outgoing query response. The role of DP controller is to coordinate the epsilon values across a distributed implementation of differential privacy. Such a solution is warranted in the case of targeted advertising, to account for viewership and billing considerations. See reference [5] for implementation details.

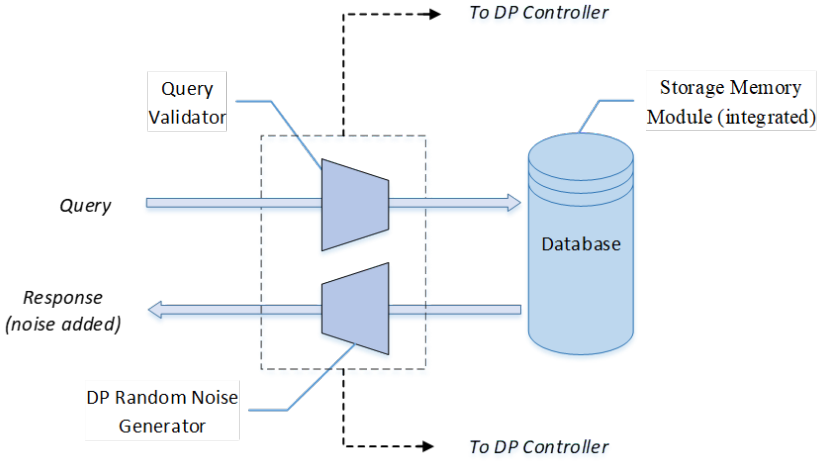

Figure 8 - Query validator and privacy module

7.1. Query sensitivity score and Privacy

The leading privacy technologies contain parameter settings to control the level of protection. Examples are the (ε) epsilon value in differential privacy, 'k' and 'l' values in k-anonymity and l-diversity, respectively. Calibrating these parameters is a trial and error process. In this regard, the graded *query sensitivity score* described above can be used in tuning the privacy parameters as shown below.

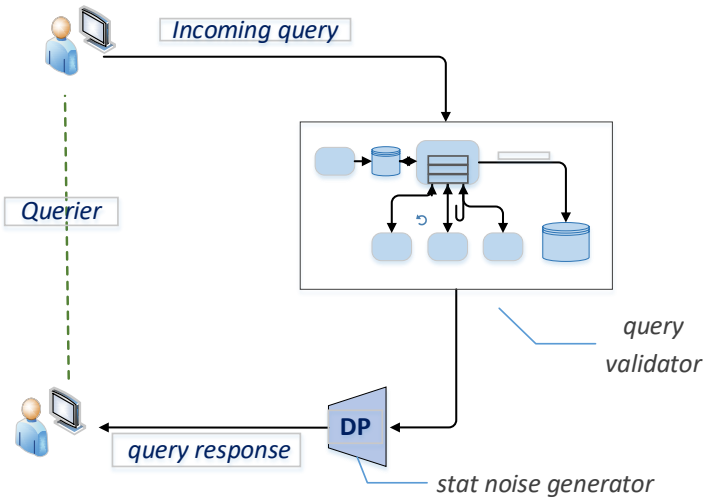

Figure 9 - Query sensitivity input to privacy module

In addition to structured databases, the solution described may also apply to search engines. Instead of a terse message ('no results to display'), the user would appreciate receiving some form of approximate results. This can be achieved with ML based query relaxation.

8. Conclusion

A timely application of Explainable AI to the cable industry was presented, driven by recent privacy regulations. XAI-enabled machine learning leads to more efficient database querying in specific applications. The quantified query sensitivity scores can be used to enhance privacy mechanisms.

9. Abbreviations

DCR	Data Clean Room
DP	Differential Privacy
LIME	Local Interpretable Model-agnostic Explanations
MPC	Secure Multi-Party Computation
PDP	Partial Dependence Plot
SHAP	Shapley Additive explanations
XAI	Explainable AI

10. Bibliography & References

[1] *Explainable Artificial Intelligence* (DARPA article) – https://www.darpa.mil/program/explainable-artificial-intelligence

[2] *Explainability won't save AI;* Brookings Institute, (May 19 2021) – https://www.brookings.edu/techstream/explainability-wont-save-ai/

[3] Examples are Google Ads Data Hub (ADH), Facebook Advanced Analytics (FAA), Amazon Marketing Cloud (AMC), Blockgraph DCR, Snowflake DCR, Habu and Decentriq DCR products.

[4] A sample SHAP analysis for Titanic dataset (reproduced with permission) – https://meichenlu.com/2018-11-10-SHAP-explainable-machine-learning/

[5] *Implementing Differential Privacy for Targeted Advertising;* SCTE Tech. Journal (Mar, 2022) – https://wagtail-prod-storage.s3.amazonaws.com/documents/SCTE_Technical_Journal_V2N1.pdf

[6] Anonymized data is any information from which the person to whom the data relates cannot be identified, whether by the company processing the data or by any other person.

