Literature Review of FEDEX: An Explainability Framework for Data Exploration Steps

CS-520 Data Integration, Warehousing, and Provenance – Fall 2023  
Girish Rajani - *A20503736* Subramanya Ganesh - *A20516250*   
Shriya Prasanna - *A20521733  
Department of Computer Science  
Illinois Institute of Technology*  
Chicago

*Abstract*— Fedex is an explainability framework that automatically generates explanations and visualizations for the output dataframes based on the context of exploratory steps. As Data Scientists, performing Exploratory Data Analysis (EDA) can become a tedious task because analyzing an extraordinary number of rows in a resulting dataframe, and performing further queries make it very difficult to interpret and make decisions. To help in this process, the proposed framework returns the set of most interesting rows from the resulting dataframe in the form of visualizations and explanations. These rows are found by how much a row contributes to the interestingness of different columns based on an interesting score. This process can be computationally expensive, and so, to tackle this problem, the framework pays more attention to sets of data that are connected/similar to each other (semantically related).

Keywords—FEDEX, Exploratory Data Analysis, Data Exploration, Explainability Framework, Visualizations

# **Introduction**

Exploratory Data Analysis (EDA) involves investigating a dataset by performing various queries to answer questions and acquire meaningful insights [2]. However, this can become challenging due to performing repetitive tasks and examining large amounts of data [3] to understand what sets of rows are meaningful. To understand the challenges that we may face during this process, let’s analyze the following scenario.

Figure 1, shown in the paper, illustrates a Spotify dataset containing over 37 features such as artist name, year, decade, popularity score, loudness, and danceability, etc. In order to understand what makes a song popular, a filter operation was performed in the dataset where the popularity score was greater than 65. This operation returns a resulting dataframe with over 7000 rows (a sample can be seen in Figure 1a). The challenge arises when analyzing the results of the dataframe to find out what is interesting about it.

To help answer this question, Clarice performs a filter operation on the original dataset to show songs released after 1990 and performs a group-by operation to show the average loudness and danceability per year. This results in only 30 rows and 3 columns (a sample can be seen in Figure 1b). Although this resulting dataframe may be easier to analyze, it is still difficult to identify patterns and answer questions such as, “Are there any years where songs are louder than other years?”

This is where FEDEX comes in. This explainability framework helps to understand and analyze the results of the dataframe after performing queries. This is done by 1) identifying interesting patterns and 2) determining which sets of rows contribute to the interestingness of the resulting dataframe. Not only does FEDEX find what’s interesting about the result, but it also takes it another step further and visualizes the output. And if that wasn’t enough, what makes this framework unique is its ability to generate explanations for the visualizations so that users can understand exactly what is going on. A perfect example of this can be seen in Figures 1a and 1b, where FEDEX is able to visualize the songs with popularity greater than 65 and identify an important pattern: newer songs are more popular than older songs, and older songs tend to be less loud than newer songs.

# **Solution/Framework**

Now that we have an idea of what FEDEX is able to do and how it can help in EDA, let’s understand what exactly this framework is and how it works.

Fedex is an explainability framework that automatically generates explanations and visualizations for the output dataframes based on the context of exploratory steps.

## Model for Notebook-based EDA

The notebook interface allows the user to import data into a notebook as a dataframe. EDA operations can then be performed on these dataframes.

**Dataframe** – A dataframe(𝑑) is a relational database that consists of several rows (r) and columns (*A* or d[A]) such that such that 𝑟 ∈ 𝑑 and *A* ∈ A (𝑑).

**EDA operations** – Fedex supports four main operations (filter, join, group-by, and union). This can also be extended to consider OLAP operations such as pivot, difference, and roll-up.

An exploratory step can be denoted as 𝑄 = (𝐷𝑖𝑛, 𝑞, 𝑑𝑜𝑢𝑡) whereby Q is the EDA operation, 𝐷𝑖𝑛­­ is the input dataframe and 𝐷out­ is the output dataframe, and for each of these steps, Fedex generates an explanation.

## Interestingness of Exploration Steps

We know that the interestingness score is calculated based on different parameters such as novelty, surprisingness, exceptionality, and diversity. By default, Fedex uses exceptionality and diversity to define the interestingness functions. Fedex also supports custom user-defined implementation of interestingness.

As we know, an operation *q* of the exploratory step Q does not apply equally to all columns. Hence, we calculate the interestingness score of each column denoted as IA (Q).

Different operations supported by Fedex have different implementations of deriving the interestingness score.

**Exceptionality (Filter/Join/Union)** – For the filter operation (*q*f), the filter result is interesting if the output dataframe significantly deviates from the input dataframe. For example, if the distribution of values in a given column is changed significantly after applying a filter operation.

Deviation of a step is defined as 𝑄 = (𝑑𝑖𝑛, 𝑞𝑓, 𝑑𝑜𝑢𝑡) with respect to the column d[A] is computed by values of 𝑑𝑖𝑛[A] and values in 𝑑out[A]. In order to check the deviation, we can use KS test, which determines if two samples come from the same probability distribution.

In order to do this test, first we define a few terms:

1. Column probability distribution given as Pr(d[A]) for each value 𝑣 ∈ 𝑑 [𝐴].
2. This means that Pr(𝑣) is the probability of selecting 𝑣 at random.

## The interestingness score of Filter is defined as:

𝐼𝐴 (𝑑𝑖𝑛, 𝑞𝑓 , 𝑑𝑜𝑢𝑡) := 𝐾𝑆 (𝑃𝑟(𝑑𝑖𝑛 [𝐴]), 𝑃𝑟(𝑑𝑜𝑢𝑡 [𝐴]))

The interestingness score of a column A after performing a filter operation 𝑞𝑓 on the input dataframe 𝑑𝑖𝑛 results in an output dataframe 𝑑out. This is equivalent to applying the KS test on the probability distribution of the input and output dataframe on column A. Similarly, join and union are defined as follows:

## The interestingness score of Join (qj) is defined as:

## 𝐼𝐴 (𝑑’𝑖𝑛, 𝑞j , 𝑑𝑜𝑢𝑡) := 𝐾𝑆 (𝑃𝑟(𝑑’𝑖𝑛 [𝐴]), 𝑃𝑟(𝑑𝑜𝑢𝑡 [𝐴]))

Where 𝑑’𝑖𝑛 ∈ Din such that A ∈ A[din].

## The interestingness score of Union (qu) is defined as:

max𝑑𝑖𝑛 ∈ 𝐷𝑖𝑛 𝐼𝐴 (𝑑𝑖𝑛, 𝑞𝑈 , 𝑑𝑜𝑢𝑡)

This is the maximal difference of the KS function with respect to the output column and each input dataframes.

**Diversity (group-by)** – The group-by step returns a dataframe with highly diverse aggregated values. This illustrates a large difference between groups. Hence, in this step, the interestingness function is equivalent to the coefficients of variation of the columns.

A black text on a white background

Description automatically generated

Where a¯ is the mean value, *n* is the number of groups, the summation defines the aggregated values in column *a*.

In the previous group-by example (Figure 1b), loudness had more diverse values than danceability. Hence, Fedex will focus on loudness in order to generate explanation candidate.

## Contribution of Sets of Rows

In order to further fine-tune the explanation for a given exploratory step, we quantify the contribution of a set of rows in an input dataframe to the interestingness of column A, IA(Q). The contribution of these rows is given by a contribution function, which is defined as

𝐶(𝑅, 𝐴, 𝑄) = 𝐼𝐴 (𝐷𝑖𝑛, 𝑞, 𝑑𝑜𝑢𝑡) − 𝐼𝐴 (𝐷𝑖𝑛 − 𝑅, 𝑞, 𝑑′ 𝑜𝑢𝑡)

𝐼𝐴 (𝐷𝑖𝑛 − 𝑅, 𝑞, 𝑑′ 𝑜𝑢𝑡) means that effectively, we are removing the set of rows R from the input dataframe and running the operation q which gives a new dataframe 𝑑′ 𝑜𝑢𝑡 and a new interestingness score is computed.

If the interestingness of a column A decreases after removing set of rows R and if this decrease is significant w.r.t the previous interestingness score, then this means that the removed rows contribute more and, hence, should not be removed since it is vital for the computation of the interestingness score.

**How we calculate the contribution of rows:**

First, we calculate the interestingness of a given column with all rows R then we remove a set of rows r from R and recalculate. If the decrease is more, then r has high contribution. For example, to calculate the contribution of decade = 2010’s in Figure1a, we eliminate all rows where decade = 2010, then we put a filter of popularity > 65 and finally, recalculate the interestingness. We see that there is a significant change in the interestingness which implies that decade = 2010 rows are significant.

## Explanation Candidates

An explanation candidate (E) is defined for a given exploratory step Q as E := (R, A) where R, set of rows, belongs to Din and A belongs to set of all columns in Dout.

From Figure1a, an example of an explanation candidate can be seen as (𝑅′𝑑𝑒𝑐𝑎𝑑𝑒′=2010𝑠 ′, 𝐴′𝑑𝑒𝑐𝑎𝑑𝑒′). Since the contribution of this explanation candidate is high, this explains most of the deviation after the filter.

## Partitioning the Input Dataframe

As mentioned in the Abstract, Fedex focuses on sets of rows that are semantically related (similar to each other in terms of their context/meaning). Not only does this result in faster computation time, but it allows the generation of more meaningful explanations since it will understand the context of these sets of rows and their relationship with each other.

To explain how Fedex is able to create more meaningful insights when working with a set of semantically related rows rather than individual rows, let’s take a look again at Figure 2b in the paper. We see that if we look at individual rows, it is mentioned that songs in 1991 and 2007 are the highest contributors to the interestingness of the loudness feature. However, if we group semantically related rows and show insights per decade as in Figure 2b, we get the bigger picture and can observe that songs in decade 1990’s are less loud than later decades.

Fedex performs row partition by dividing the input dataframe into n + 1 sets of rows. These sets of rows are disjoint such that they have no element in common.

∀𝑅𝑖 , 𝑅𝑗 ∈ R, 𝑅𝑖 ∩𝑅𝑗 = ∅

There is also an ignore set 𝑅ˆ which contains a special set of rows that cannot be used as an explanation candidate.

To perform this row partition, Fedex uses three methods:

1. **Frequency-based partition** – Given a specified column, this method splits the input dataset into *n* sets of rows using the *n* most interesting values in that row. After which, assigning the remaining rows to the ignore set 𝑅ˆ. An example of this can be seen in Figure 2 where Fedex has used this partition for the year attribute into the *n* most frequent values.
2. **Numeric-based partition** – In this method, *equal-frequency binning* is used to a numeric attribute into *n* equal sets-of-rows. Each set of rows contains an interval range from the numeric attribute. Since there are equal values in each set, the ignore set 𝑅ˆ is empty.
3. **Many-to-one partition** – As the name suggests, this method looks for many-to-one relationships existing between a chosen attribute A and some other attribute B. An attribute B must hold the following conditions:

(1) ∀𝑟𝑖 , 𝑟𝑗 ∈ 𝑑𝑖𝑛, (𝑟𝑖 [𝐴] = 𝑟𝑗 [𝐴]) → (𝑟𝑖 [𝐵] = 𝑟𝑗 [𝐵])

For all 𝑟𝑖 and 𝑟j in the input dataframe, if column A in 𝑟𝑖 and 𝑟j are same then column B in 𝑟𝑖 and 𝑟j are also same.

(2) ∃𝑟𝑖 , 𝑟𝑗 ∈ 𝑑𝑖𝑛, (𝑟𝑖 [𝐵] = 𝑟𝑗 [𝐵]) ∧ (𝑟𝑖 [𝐴] ≠ 𝑟𝑗 [𝐴])

There exists an 𝑟𝑖 and 𝑟j in the input dataframe such that column B in 𝑟𝑖 and 𝑟j are same and column A 𝑟𝑖 and 𝑟j are not same.

After mapping the values from A to B, the Frequency-based partition is used over column B. This method of partitioning is preferable for group-by dataframes. As explained above, Fedex has used frequency-based partition for the year column and then after the group-by operation, the many-to-one partition was used since a decade can have many years and a year belongs to one decade. This in turn gives better insights as explained in figure 2b.

## Quality of Explanation Candidates

A good explanation candidate is of high quality if it has (1) a high interestingness score 𝐼𝐴(𝑄), and (2) the contribution 𝐶¯(𝑅, 𝐴) to the interestingness is high. Let’s explain how Fedex derives the interestingness and contribution of a set of rows.

When finding the significance of contribution, Fedex compares the contribution of a set of rows R to the contribution of other sets of rows in the partition *R*.

A close-up of a word

Description automatically generated

This means that the standardized contribution of R compared to the other sets of rows is given by the contribution of 𝑅 to the interestingness of a column 𝐴 in an exploratory step Q - the mean divided by the standard deviation of all sets of rows in the partition.

**Skyline of contribution & interestingness**

The set of all high-quality explanation candidates EC(*Q*) for an exploratory step Q is given by:

A close up of a logo

Description automatically generated

Using this set, Fedex then returns the candidates that have good 𝐼𝐴(𝑄) and ¯(𝑅, 𝐴). To do this, Fedex uses a skyline-operator calculation. The high-quality explanation candidates EX satisfy the following:

∀(𝑅, 𝐴) ∈ 𝐸𝑋 . ∃ (𝑅 ′ , 𝐴′ ) ∈ 𝐸𝐶(𝑄). (𝐼𝐴′ (𝑄) > 𝐼𝐴 (𝑄)∧ 𝐶¯(𝑅 ′ , 𝐴′ ) > 𝐶¯(𝑅, 𝐴))

This means that for all dominating explanation candidates, there does not exist a set of explanation candidates where the interestingness score and the contribution to the interestingness are higher than the dominating explanation candidates.

**To summarize the algorithm used by Fedex entirely, let’s look at Algorithm 1 in the paper.**

- The user enters an exploratory step, such as a filter query

- for all the columns in the output dataframe, the interestingness score is calculated.

- The algorithm then performs row partition to focus on semantically related sets-of-rows.

- For each partition and column, it iterates through each set of rows in that partition w.r.t to column A to compute the contribution to the interestingness and if this contribution is greater than 0, the 𝐼𝐴(𝑄) and 𝐶¯(𝑅, 𝐴) are stored as a good explanation candidate.

- The skyline-operator calculation will be performed on those explanation candidates to find dominating candidates.

- Visualizations and explanations are created for these dominating candidates w.r.t to the exploratory step Q.

- The run time complexity of the algorithm is Quadratic time complexity.

## Explanations Generation Process

The following is the Explanation generation for a step Q based on Algorithm 1.

**Pre-processing: interestingness & row partitioning**

Based on the operation and given column, we calculate the interestingness score for that column 𝐼𝐴(𝑄) by applying the respective functions as mentioned before.

Multiple partitions are created through the row partitioning technique and unify all sets of rows into SR.

**Forming explanation-candidates and calculating standardized contribution**

The empty dictionary EC stores the quality score for each explanation candidate. Each explanation candidate is obtained by iterating over all partitions w.r.t every output column. For each of these pairs, we iterate over the set of rows in the partition, which is used to compute 𝐶¯(𝑅, 𝐴). If this contribution is positive, then 𝐼𝐴(𝑄) and 𝐶¯(𝑅, 𝐴) are stored in EC.

**Calculating the interestingness/contribution skyline**

The skyline operator takes in EC as the input and returns the most dominant explanation in terms of 𝐼𝐴(𝑄) and 𝐶¯(𝑅, 𝐴). We can further limit the explanation result by using a weighted average between 𝐼𝐴(𝑄) and 𝐶¯(𝑅, 𝐴). This can be done as:

A close up of a logo

Description automatically generated

The skyline then just keeps the top-k explanation candidates w.r.t to the score.

**Generating captioned visualizations for each resulted explanation**

After getting the dominant explanation, Fedex generates visualizations for these based on the type of interestingness.

*Exceptionality-based explanations* – In these cases, join/filter operations are used, so we highlight the deviation of column A. This is done by plotting a side/side plot graph. As shown in Figure 1a, the left hand side shows mean *d*inand right hand side shows mean *dout* of A for all sets of rows in the partition.

To generate captions/text explanations, a natural language template is used, and the attribute name A and label of R are plugged in according to the partition method.

The labels are based on the type of partition used:

1. Numeric-based partition – The end value is set as the label

2. Many-to-one partition – Value in column B is set as the label

3. Frequency-based partition – The value is the label

*Diversity-based explanations* – This highlights the extreme values w.r.t the aggregated values from column A. As shown in Figure 1b, the bar chart visualizes these aggregated values.

The captions for these explanations are generated using a similar approach to exceptionality-based. The only difference is here, this visualization emphasizes more how far the Rj­ is from the mean in terms of standard deviation.

**Sampling optimization (FEDEX-SAMPLING)**

To reduce the generation complexity, we use a sampling approach to optimize Algorithm 1. Instead of considering all rows, the rows of interest are considered, and the 𝐼𝐴(𝑄) is calculated. This uses uniform sampling.

## Customization & Extensions of FEDEX

There were several ways highlighted in the paper in which Fedex can be extended to provide more usability to the user.

**General interestingness functions** –Outside of using established interestingness measures, Fedex is also capable of using any general interestingness function such as compactness, coverage, and surprisingness as input.

**User-specified columns** – If a user is only interested in a specific set of columns, such as danceability and loudness, these can be specified. Rather than computing explanations and visualizations for all columns, Fedex will only return that for the specified columns based on the interestingness score of the sets of rows w.r.t those columns.

**Custom partitioning of rows** – A user can also explicitly specify a partitioning approach outside of the ones explained. However, the proposed method must comply with the row partition approach defined in Section 2.5 such that the input dataframe partitions the dataframe into n + 1 disjointed sets of rows.

# **Discussion/Experiments**

This paper evaluated the performance of Fedex and Fedex-Sampling (an optimized version) frameworks and compared them to existing baselines. The evaluation consists of three real-world datasets being compared with four other baselines that also help in insight discovery and visualizations based on previous work. This will be further explained below. Fedex was also tested by our group on the data curation project dataset to see if any meaningful insights can be generated and how easy it is to implement.

## Setup, Datasets, Queries, and Baselines

**Implementation of FEDEX –** Python 3.8 along with various libraries was used to create the Fedex explainability framework. The Pandas library was used to store and analyze the data, NumPy for performing mathematical operations, and Matplotlib for generating the visualizations.

**Datasets used –** The **Spotify** dataset (174,389 rows and 20 columns) contains features such as genre, popularity, danceability, etc., **Credit Card Customers** dataset (10,127 rows and 21 columns) containing features such as education level, credit card category, etc., and **Products and Sales** dataset (containing a Product table and Sales table with 10 million rows after joining and uniformly sampling the dataset) were used. Since these are real-world datasets, most of the columns are skewed.

**Queries –** For the experiment, five filter/join queries and five group-by queries were performed on each of the datasets by each of the frameworks.

**Baseline and optimized version –** The following frameworks have been used as baseline for the comparison

1. **SeeDB** – This framework generates visualizations automatically by querying over a source code. These visuals showcase interesting observations.
2. **RATH –** Generates top-k visualizations automatically using a single score function.
3. **IO** – **I**nterestingness **O**nly framework measures the difference between the interestingness of an attribute to determine the influence of an attribute.
4. **EXPERT –** This baseline uses interesting facts retrieved manually from expert users who had access to the datasets and queries. Their facts contained detailed explanations of what would be considered interesting within the data.
5. **FEDEX-SAMPLING** – This version of Fedex utilizes row sampling in order to determine the interestingness score. The paper determined that a sample size of 5,000 rows was sufficiently large and still maintained satisfactory accuracy.

## 3.2 User Studies

Two studies were conducted to determine the quality of the explanations created by both FEDEX and FEDEX-SAMPLING when compared to the existing baselines explained in 3.1.

**Comparison to existing baselines**

The study included 25 participants, most of whom were not programming experts. All three datasets tested consisted of filter/join and group-by queries.

Examples filter/join and group-by queries:

SELECT \* FROM spotify WHERE year > 1990;

SELECT \* FROM products INNER JOIN sales ON products.item=sales.item;

SELECT mean(popularity), max(popularity), min(popularity) FROM spotify GROUP BY year;

Each notebook (one for each dataset) contained explanations generated by each of the baselines. Figure 4 in the paper shows that Fedex generates explanations significantly faster than experts who have to manually analyze and write explanations.

To evaluate the quality of the explanations of each framework, each user was shown the input dataset, query, and the query output after, they had to critique each explanation based on coherency, insight level, usefulness. In general, the explanations manually created by experts were found to have the highest rating. However, when comparing the frameworks that generate automatic explanations, FEDEX was the preferred framework, achieving an average score higher than IO, SeeDB, and Rath across all three datasets. An obvious reason for this is because in the explanations, FEDEX generates both visuals and text which increase interpretability compared to the other baselines that may only generate one or the other. It is also important to note that for this study, FEDEX and FEDEX-SAMPLING had identical explanations because they had the same skyline set so only Fedex was shown.

When looking specifically at the products and sales dataset in Figure 4c, Fedex received a very close score to the experts, primarily due to the join operation. Fedex noticed and pointed out a change in the distribution of the data after the join query which the experts missed.

**Comparison to unassisted EDA**

Another user study on how EDA was performed with and without Fedex/Fedex-Sampling. Each dataset had a specified task to be completed on a blank notebook. An expert would confirm whether each user-generated explanation was relevant or not. Figure 5 shows the number of useful insights created by users within 10 minutes with and without Fedex/Fedex-Sampling. It can be seen that using these assisted EDA frameworks plays a very significant role in insight discovery.

**Comparison to augmented baselines**

The SeeDB and Rath baselines only perform visualizations but have been augmented to also include textual explanations so that there can be a more robust comparison with Fedex. Another study was then performed using the same queries from the Credit Card Customers dataset from the initial study.

Since using automatic captioning techniques may result in inconsistent result, an expert manually generated the textual explanations for SeeDB and RathDB. As shown in Figure 6, Fedex is still able to generate more insightful textual explanations compared to the ones generated by the expert.

## 3.3 Simulated Experiments

We have evaluated the quality of explanations generated by Fedex when compared to other baselines in the previous section, and now let’s see how to evaluate the accuracy of these explanations. We will see how the number of sets of rows affects the accuracy. For this experiment, Fedex divides the dataframe into 5 or 10 sets of rows.

**Accuracy of FEDEX-SAMPLING**

Using the queries performed on the Spotify and Products datasets, figure 7a shows the precision score for Fedex-Sampling using different sample sizes using Fedex as the ground truth. A very high precision score was received throughout all sample sizes. This means that even at a small sample size of 5K samples is sufficient to achieve a high accuracy while successfully able to predict the skyline explanations. Figure 7b shows the Kendall-Tau distance and we can see that as the sample size increases, the Kendall-Tau score decreases. When looking at a sample size of 5K, we see that it’s value is 21.6 which is lower than the average score of 33.1. Finally, Figure 7c shows the nDCG score and across all sample size, the score is >90% achieving 99.8% for sample size 5K. These results show that a sample size of 5K is the sweet spot.

The same tests were repeated now using a fixed sample size of 5K on different number of rows from the Products and Sales dataset. This test shows very satisfactory results shown in Figure 8.

**Runtime analysis for varying column number**

The same queries as done in the previous study was performed again on all 3 datasets using all sets of rows but instead, gradually increasing the number of columns to test runtime. Although the columns varied per test, 2 columns were kept constant throughout (the column being queried and the column that returned the highest interestingness score). Figure 9 depicts the runtime of Fedex-Samping, SeeDB and Rath.

Figure 9a shows the runtime on Credit Card Customer dataset where Rath has a high runtime independent of the number of columns, SeeDB gradually takes longer as the number of columns increases and Fedex has the lowest runtime overall regardless of number of columns.

Figure 9b shows the runtime on Spotify dataset. Here, although Fedex does perform well, SeeDB performs better. This is because SeeDB considers both categorical and numeric data and since this dataset consists of mostly numeric columns, it reduced the runtime of SeeDB since it did not have to consider categorical columns.

Lastly, Figure 9c shows the runtime on Products and Sales dataset. Runtime of Rath here was omitted due to its inability to process the large number of data. Fedex significantly took less time than SeeDB making Fedex-Sampling with a sample size of 5K the top contender.

**Runtime analysis for varying row number**

The same test as above was done but this time on varying row number. The results were the same as above. In figure 10, on the Credit Card Customer and Products/Sales datasets, Fedex performed the best regardless of the number of rows while in the Spotify dataset, SeeDB slightly outperformed Fedex due to the same reason mentioned above.

**Accuracy for varying sets-of-rows** **sizes**

The final test as shown in figure 11 showing the contribution score for varying sets of rows. 2 queries were performed (1 from Product and Sales dataset and 1 from Spotify). There was no specific number of sets of rows that performed better. It varied based on the query and values used. However, it was observed that using a small number of sets of rows resulted in clearer visualizations and that is why 5 and 10 sets of rows were used in the earlier user study conducted.

# **Related Work**

Various solutions for assisting in data exploration have been explored. Such solutions include query suggestion frameworks, a non-programming EDA framework, and other guidance tools. However, when looking at related work to Fedex, we are looking at three categories: deriving the interestingness of rows, a framework to write query explanations, and visualizing these explanations for easy interpretation.

## Deriving the interestingness of rows

It can be challenging to decide whether a set of rows in an output is interesting or not. One previously proposed method was to use mining logs from previous EDA to determine interestingness before the next EDA task begins. Another system was to collect real-time data from user by asking them whether they think a set of rows are interesting or not, and the last system was to gain examples of interestingness from the user in advance.

These techniques differ from Fedex in that these previous works recommend interesting data and operations. Fedex takes this a step further and creates explanations based on the most interesting sets of rows. This way, we can understand why something is interesting rather than just telling us that this is interesting.

## Query explanation framework

Previous work in this field explored the area of data provenance, intervention, and influence in causality to understand which set/s of rows affect the result of a query either by their presence or absence. Additionally, some work have been able to explain results generated from aggregation operations and explain outliers. They have also been able to detect the redundant rows created after incorrectly performing an aggregation operation.

However, Fedex is less interested in which rows affect the query results but is more interested in how these rows are impacting the interestingness score of an EDA operation.

## Creating automatic visualizations

As we know, there are lots of systems that are able to take a dataset as an input and generate visualizations. This is done by returning the highest ranked visualizations or top-k visualizations [4]. Using a similar technique, we can take this a step further gaining useful information such as identifying outliers, correlations, trends, etc. Going beyond insight discovery, some previous work have been able to generate random interesting facts.

Although Fedex is also able to generate visualizations and aid in insight discovery, instead of generating random interesting facts, it creates useful insights that is relevant to the exploratory steps performed by the user. According to the paper, this is the only system that is able to use interestingness score and causality assessment such as intervention and influence to not only generate visualizations but also relevant explanations.

## 3.4 Testing Fedex on Divvy Bike Dataset

Since Fedex is an open-source framework, the group decided to implement the Fedex framework on our cleaned Divvy Bike ride dataset from the CS 520 Data Curation Project. This was done to understand how easy it is to implement this framework and what interesting insights it can discover. To install and use this framework we first did *pip install pd-explain* and then imported the pd\_explain library in a Jupyter Notebook. The cleaned Divvy dataset was imported as a dataframe and the following filter query was performed:

SELECT \* FROM divvy\_months\_cleaned where member\_casual = ‘casual’;

This query showed all records where the bike riders were of casual type rather than membership. Fedex was able to find a very interesting insight as shown in Figure 11 below which was trip time above 30 minutes appeared 1.8 times more than before the filter. This meant that specifically for longer bike rides than 30 minutes there were almost double the amount of casual riders when compared to the original dataset consisting of casual and member riders.

A screenshot of a graph

Description automatically generated

Figure 11- Explanation generated by Fedex on Divvy dataset

# **conclusion**

In conclusion, this paper introduces a framework called FEDEX that provides explanations to the user during the EDA phase. This helps the user to find interesting rows in the resulting dataframe and make decisions based on that. These rows are found based on how much they contribute to the interestingness score of a column. These sets of interesting rows are then presented to the user in the form of visualizations and explanations. This is the first framework that is capable of generating explanations and visualizations based on the resulting dataframe. This framework has seen success across various real-world datasets.

The group has also observed that Fedex is a great explainability framework when compared to other baselines because it generates both visualization and text explanations which increases the user understanding. Additionally, Fedex is able to perform very close to explanations manually generated by experts. In some cases, it is able to point out key insights that were overlooked by experts. Additionally, Fedex was also able to generate meaningful insights when implemented on the data curation project cleaned dataset.

Rather than using only measures such as diversity and exceptionality to measure the contribution of rows, in the future, it is planned to introduce a causal responsibility measure. Additional techniques for optimizing the interestingness calculations will also be studied.

##### **References**

1. Amir Gilad, Amit Mualem, Amit Somech, Daniel Deutch and Tova Milo, "fedex: An Explainability Framework for Data Exploration Steps." arXiv preprintarXiv: 2209.06260 (2022).
2. https://www.ibm.com/topics/exploratory-data-analysis
3. https://arxiv.org/pdf/1911.00568.pdf
4. https://dbgroup.cs.tsinghua.edu.cn/ligl/papers/vldbj20-vizsurvey.pdf