# **Selecting Sub-tables for Data Exploration**

Kathy Razmadze Tel Aviv University kathyr@mail.tau.ac.il Yael Amsterdamer Bar-Ilan University amstery@cs.biu.ac.il Amit Somech Bar-Ilan University somecha@cs.biu.ac.il

Tova Milo Tel Aviv University milo@post.tau.ac.il

Raw Dataset (6M Rows X 31 Columns):

## ABSTRACT

We present a framework for creating small, informative sub-tables of large data tables to facilitate the first step of data science: data exploration. Given a large data table table T, the goal is to create a sub-table of small, fixed dimensions, by selecting a subset of T's rows and projecting them over a subset of T's columns. The question is: which rows and columns should be selected to yield an informative sub-table?

Susan B. Davidson

University of Pennsylvania susan@cis.upenn.edu

We formalize the notion informativeness based two complementary metrics: *cell coverage*, which measures how well the sub-table captures prominent association rules in T, and *diversity*. Since computing optimal sub-tables using these metrics is shown to be infeasible, we give an efficient algorithm which indirectly accounts for association rules using table embedding. The resulting framework can be used for visualizing the complete sub-table, as well as for displaying the results of queries over the sub-table, enabling the user to quickly understand the results and determine subsequent queries. Experimental results show that we can efficiently compute high-quality sub-tables as measured by our metrics, as well as by feedback from user-studies.

#### **1 INTRODUCTION**

Data exploration is an important first step in data analytics. During this step, the analyst tries to understand an unfamiliar dataset and determine what part of the data is relevant to their task by displaying the table, looking at the table description, or visualizing column values. They may also run simple exploratory queries over the dataset, using selection, projection, sorting and grouping. However, when displaying a large input table or query result only a small subset of the table is typically shown – and without input from the user, the subset is arbitrary. For example, the default display of Pandas<sup>1</sup> tables using the Python display() command includes the first and last several rows and columns. Frequently, this is not very informative as the sub-table may contain a lot of missing values and/or fail to capture the range of possible values in a column; it may also elide columns that are important for further exploration and analysis.

*Example 1.1.* Consider a table *T* taken from the Kaggle *flights* dataset<sup>2</sup> which contains 31 columns and ~6M rows. The analyst is using *T* to predict flight cancellations, and hence is interested in a specific target column, *CANCELLED*. The analyst starts by visually inspecting the data using Pandas display(*T*), which yields the table displayed at the top of Figure 1. This display of *T* is not informative for the analysis task, as it does not include the target column. More

DAY OF WEEK AIR SY DELAY SECURIT DELAY AIRLINI DELAY LATE AIRCRA DELAY WEATHER DELAY 2015 NaN AS NaN NaN NaN 2015 ۸.۵ NaN 2015 US NaN NaN NaN 2015 31 B6 NaN NaN NaN 2015 12 NaN NaN 2015 NaN NaN 12 31 Query Resul Quer SubTab centroid embedding Informative Sub-Table (10 Rows X 10 Columns): ELAPSED TIME DEPAR SCHEDUL SCHEI TIME AIR TIME WHE DEPAF TIME SCHEDULE 263.0 -2.0 247.0 242.0 1505.0 1448.0 140.0 NaN NaN 733 NoN 48.0 -5.0 59.0 31.0 1009 955 165.0 -9.0 166.0 140.0 551.0 93.0 102.0 77.0 1222.0 15.0 1348.0 602 170.0 175.0 135.0 1646.0 1620.0 1920 1901.0 1240.0 1228.0 1235 -7.0 1643 188.0 156.0 1616.0 0.00 79.0 1932.0 1916.0 101.0 2050 94.0 2051.0 1916 -6.0 167.0 124.0 904.0 159.0 1134 1108.0 101.0 -5.0 1932 89.0 58.0

#### Figure 1: System Architecture

importantly, its usefulness for data exploration is limited: e.g., the last five columns contain only NaN values, and other columns include many repetitions of arbitrary values.

Our goal is to support the data exploration task by selecting small, informative sub-tables through which analysts can view data. That is, given a table T with n rows and m columns, our goal is to create a sub-table  $T_{sub}$  with  $k \ll n$  rows and  $l \ll m$  columns which is a subset of k rows projected over a subset of l columns of T. The sub-table should be *informative* in the sense that it captures important data patterns within and across columns in T, where we define patterns using the standard notion of *association rules*. The sub-table should also contain *diverse* cell values out of the ones actually occurring in each selected column. If one or more *target columns* are known in advance to be the focus of the analysis, they must also be included in the l selected columns.

*Example 1.2.* Continuing with the *flights* dataset, an *informative* sub-table is shown at the bottom of Figure 1. The sub-table captures several prominent association rules that hold over the input table and that include the target column *CANCELLED*. We visualize this by highlighting, in each row, the cells that participate in a rule that holds for this row (many more rules hold, to avoid visual clutter

<sup>&</sup>lt;sup>1</sup>Pandas: Python Data Analysis Library. https://pandas.pydata.org/

<sup>&</sup>lt;sup>2</sup>https://www.kaggle.com/usdot/flight-delays?select=flights.csv

we only highlight one rule per row). For example, the first row of the sub-table exemplifies an association rule stating that long flights  $(AIR\_TIME \in [198.0, 422.0] \text{ and } DISTANCE \in [1546.0, 2724.0])$  are likely *not* to be cancelled (highlighted in orange). The second row of the sub-table exemplifies an association rule stating that short afternoon flights (according to the *SCHEDULED\_DEPARTURE* and *SCHEDULED\_ARRIVAL* columns) are likely to be cancelled (*CANCELLED* = 1, highlighted in blue). Beyond these association rules, the sub-table gives useful insights about the column values, by showing diverse rows and diverse values per column.

As alluded to above, we formalize the notion of informativeness based on a combination of two complementary metrics: *cell coverage* and *diversity*. Cell coverage measures how well the sub-table  $T_{sub}$ covers prominent<sup>3</sup> association rules that hold over the input table T. Given a set of prominent association rules, we consider the rules that are captured by the sub-table as well as the marginal contribution of each rule, and combine them in a metric which reflects the number of cells in T that are describable by association rules that are captured in  $T_{sub}$ . If one or more *target columns* are known in advance to be the focus of the analysis, they will be included in the *l* selected columns, and we measure cell coverage only according to association rules that include one or more target columns.

For the second metric, *diversity*, we rely on the average pair-wise similarity between the rows of the sub-table, using a Jaccard-like similarity measure that accounts for categorical as well as continuous columns. These metrics are combined in a *score* to measure the informativeness of a sub-table.

Unfortunately, we show that optimizing this informativeness score directly is infeasible. Moreover, even computing the association rules may not be practical in our setting: although there are several efficient techniques for mining association rules (e.g., [2, 15, 24, 27, 28]), they may still be overly time consuming for large datasets in an interactive setting.

We therefore consider a sub-table computation method which indirectly accounts for association rules using *table embedding* [5–7]. Given a table T we use *binning* [27] to split each column's values into a small set of meaningful groups. We then compute an *embedding* of table cells as vectors. Several different methods have been recently proposed for this task [5–7, 12, 29, 35], among which we chose a fast and effective embedding method based on Word2Vec [21] (and compare with other options in Section 6).

The embedding captures bin co-occurrences, and therefore implicitly corresponds to frequent itemsets and association rules. To select rows and columns for a sub-table we derive from the cell vectors a vector representation for rows and columns, cluster them (separately) and select the centroids as rows and columns that represent diverse patterns in the data.

Empirically, this method achieves near-optimal scores when compared to upper bounds on coverage and diversity.

An important benefit of our solution design is in *responding to queries over* T: during the exploratory data analysis (EDA) session, users typically issue different queries on a given table T (red arrows of Figure 1). Our computation of cell embedding may be viewed as a part of the pre-processing step of a given data table T, along with the

binning of its values (first blue box in Figure 1). This step only has to be executed once upon loading the table. Then, if the analyst issues a selection-projection (SP) query on T and wishes to view its result as a sub-table, we need only to compute the vector representation of rows and columns in Q(T) based on the cells of T that appear in them, and re-execute clustering and centroid selection (*Selecting* step in Figure 1, shown as the second blue box). This significantly speeds up sub-table computation compared with computing everything from scratch (a few seconds instead of up to a minute for large tables).

Contributions. The contributions of this work include:

- (1) A notion of *informativeness* for sub-tables, which includes both cell coverage and diversity. Cell coverage measures how well the sub-table captures prominent association rules in the input table, while diversity ensures that repetition of values within the sub-table are minimized.
- (2) A formalization and complexity analysis of the problem of selecting an optimal sub-table, which shows that computing optimal sub-tables using the informativeness metric is infeasible.
- (3) A *greedy algorithm* for row selection which has approximation guarantees for optimal row selection, and a *semi-greedy algorithm* which further samples column combinations in order to find sub-tables with high cell coverage. Although the algorithm is not practical in an interactive setting, it serves as a baseline against which we compare the quality of computed sub-tables.
- (4) A practical algorithm, SubTab, for computing informative sub-tables which accounts for association rules indirectly using table embedding. The algorithm has two phases: a preprocessing phase which performs binning and embedding, and can be executed as soon as the data is loaded; and a clustering and centroid selection phase that is called for each sub-table display, e.g., on query results.
- (5) An *implementation* for SubTab as a local Python library that hooks into Pandas, and displays tables and query results as informative sub-tables. The implementation includes a UI which optionally highlights association rules (as in Figure 1).
- (6) Experimental results that measure the running time of SubTab and several baseline algorithms, as well as the quality of sub-tables produced. Results show that the quality of subtables computed by SubTab exceed those of other interactive algorithms, and are comparable to algorithms that directly optimize our metric. Experiments with EDA sessions and user-studies show SubTab's effectiveness for enabling analysts to derive useful insights from the data. An analysis of its running time shows that SubTab is suitable for EDA sessions, and that the pre-processing step enables interactive displays of query results during the session.

*Organization.* The rest of the paper is organized as follows. Related work is discussed in Section 2. In Section 3 we define our metrics of cell-coverage and diversity. In Section 4 we show hardness results for the problem of optimal sub-table selection, and the greedy and semi-greedy algorithms. In Section 5 we present our subtable computation method based on table embeddings. In Section 6 we discuss experimental results. We conclude in Section 7.

<sup>&</sup>lt;sup>3</sup>There are standard metrics we can use to measure the prominence of association rules in T, such as Support and Confidence [2].

from SubTab import subTab
sdf = subTab(flights\_df, use\_rules=True)
df1 = flights\_df[(flights\_df['DISTANCE']>2000) & (flights\_df['DEPARTURE\_DELAY']>10)]
sdf.display(df1)

for creating new table vectors, it took 0:00:02.282215
for summary creation, it took 0:00:00.457031
{'cell cov': 0.31. 'jaccard': 0.72}

DESTINATION\_AIRPORT ARRIVAL\_TIME WHEELS\_ON AIRLINE\_DELAY DEPARTURE\_TIME WHEELS\_OFF SCHEDULED\_DEPARTURE LATE\_AIRCRAFT\_DELAY DISTANCE DEPARTURE\_DELAY

57538	МСО	1842.0	1832.0	102.0	1050.0	1107.0	855	0.0	2218	115.0
72874	JFK	742.0	736.0	nan	2333.0	2345.0	2305	nan	2475	28.0
85795	MCO	2255.0	2247.0	2.0	1449.0	1504.0	1400	42.0	2446	49.0
Rule #1	{DESTINATION_AIRPOR	RT = 'SEA'), (AR	RIVAL_TIME =	'high'), (DISTAN(	CE = 'high')}>	{(AIRLINE_DELAY	' = 'low'), (DEPARTURE	_DELAY = 'high')}, supp	ort=0.11, c	onfidence = 0.5, lift = 1
25656	SEA	2300.0	2253.0	20.0	1939.0	2012.0	1807	72.0	2402	92.0
29729	SAN	1941.0	1938.0	nan	1649.0	1702.0	1629	nan	2588	20.0
22207	BOS	1550.0	1545.0	nan	752.0	806.0	730	nan	2611	22.0
3466	SFO	2136.0	2130.0	nan	1820.0	1834.0	1808	nan	2704	12.0
55070	JFK	16.0	3.0	0.0	1603.0	1615.0	1535	17.0	2475	28.0

Figure 2: Example usage of SubTab – presenting an informative, 10X10 sub-table for a large query results-set

#### 2 RELATED WORK

There are three main lines of related work: (1) *row sampling*, (2) *feature selection*, and (3) *data summarization*. Row sampling (resp., feature selection) aims to reduce the number of rows (resp., columns) in the data, while the goal of data summarization is to generate an informative yet compressed summary of the data using a series of aggregations.

*Row Sampling.* The task of sampling or selecting representative *rows* from a large dataset has been studied in previous work for several different use cases. For example, work in *Approximate Query Processing (AQP)* suggests using stratified sampling [1] and dynamic sampling [1, 3] in order to reduce the number of tuples and produce faster yet inexact query results. Row sampling is used for *efficiently generating data visualizations* [25], i.e., reducing the number of underlying data points while minimizing the error in the produced visualization. This is typically done by using a visualization-inspired loss function [4]. Another use case is *query result diversification* [20, 30], in which the goal is to select rows from the query result that are both *relevant* and *diverse*. This is often done with a greedy algorithm which finds local-optimum solutions, given ad-hoc definitions of relevance and diversity [30].

In contrast, our approach SubTab generates a sub-table by directly selecting both rows *and* columns. Also, while most of the works mentioned above are designed for a particular use case or have prior assumptions (e.g., a notion of relevance, or a specific query load [14]), SubTab captures generic patterns in the data, without using prior assumptions, for the purpose of producing informative, small samples of large tables. The latter allows SubTab to support many different data exploration use cases, information needs, and datasets.

*Feature Selection.* The task of reducing the number of columns in a dataset is an important step in many machine learning processes. The goal is typically to reduce the number of input variables considered by the ML model, which reduces the model's complexity as well as the training time [8, 32]. There is a plethora of work on feature selection (see [8] for a survey), which can be roughly categorized as filter methods, that output the Top-k features w.r.t. a given

metric (e.g., Chi-Square, ANOVA and Information-Gain) [32]; as well as embedded [16] and wrapper [18] methods, which directly utilize the ML model to determine feature importance [9].

While this work describes fundamental ML tools, they are illsuited to our problem: first of all, they only select columns, and cannot be easily adapted to also select representative rows. Second, feature selection tools operate w.r.t a predefined, target column and a prediction task, which may not exist in the data exploration phase. As mentioned above, SubTab produces a sub-table by selecting both rows and columns, and does not require a target feature (although it will use target features if they are known).

*Data summarization.* Another line of work attempts to derive compressed forms of the data, or produce a high-level, compact summary of the dataset. These includes dimensionality reduction [11] techniques, data sketches [10] for online streams, and techniques for aggregation-focused AQP [17, 34].

Such summaries do preserve some properties of the original data, however they produce an *altered*, compressed version. While these summaries are very useful for use cases such as AQP and feature engineering [11], they are not suitable for interactive analysis, where the user wants to see the actual data. In contrast, SubTab is focused on interactive data exploration, and thus efficiently generates subtables which contain representative rows and columns taken from the original table.

### **3 METRICS**

In this section, we define our metrics of cell coverage and diversity, and the problem of finding an optimal sub-table based on their combination. We then show that directly optimizing our metrics is infeasible in Section 4, and propose a practical solution in Section 5.1.

#### 3.1 Model

Sub-tables are formally defined as follows: A relational schema  $U = \{u_1, \ldots, u_{|U|}\}$  is a finite set of columns, such that each column  $u_i$  allows values from a subset of the global domain  $\mathcal{D}_i \subseteq \mathcal{D}$  (e.g., for a binary column,  $\mathcal{D}_i = \{0, 1\}$ ). A relational table over U is a finite set  $T \subseteq U \rightarrow \mathcal{D}$  of tuples such that  $t(u_i) \in \mathcal{D}_i$  is the value of

the cell in the row corresponding to a tuple  $t \in T$  and the column  $u_i \in U$ .

Definition 3.1 (Sub-table). Given a table *T* over schema *U*, a sub-table  $T_{sub}$  is a table over some schema  $U_{sub} \subseteq U$  such that each tuple  $t \in T_{sub}$  is the projection of some tuple  $t' \in T$  over the columns of  $U_{sub}$ , i.e., for every  $u \in U_{sub}$ , t(u) = t'(u).

We next define two standard notions that will be useful in the sequel: binning and association rules.

In a schema U, each column  $u_i$  may be *categorical*, namely,  $\mathcal{D}_i$  is discrete, e.g., a column of airline names; or *continuous*, namely,  $\mathcal{D}_i$  is a continuous range, e.g., a column of flight distance. Moreover, in a table T over U a different distribution of values (e.g. uniform or skewed) may occur in each column. *Binning* the column values is technique commonly used to allow a uniform treatment of columns with different ranges and distribution. Formally,

Definition 3.2 (Binning). Given a table T over schema U, a binning function  $\mathcal{B}$  maps each column  $u_i \in U$  to a finite set of bins  $\mathcal{B}_i = \{B_1^i, \ldots, B_{|\mathcal{B}_i|}^i\}$  such that for every  $t \in T$ ,  $t(u_i)$  belongs to exactly one bin  $B_i^i \in \mathcal{B}_i$ .

We discuss in Section 5.1 the method we use for computing such a binning function.

*Example 3.3.* In the flights dataset, the *DISTANCE* column is continuous, and so we split its range into the bins short, medium and long-distance. Depending on the column value distribution, we may obtain  $B_{long}^{DIST} = [1546.0, 2724.0]$ . The continuous *AIRTIME* column may also have bins with the same labels, but different ranges matching is value distribution, e.g.,  $B_{long}^{AT} = [198, 422]$ . The *CAN-CELLED* column is binary, hence we can use its categories as bins. The *AIRLINE* column is also categorical but has many categories. We can create a smaller number of groups by e.g. splitting the airlines according to the continent in which they are headquartered.

Next, we recall the notion of *association rules*, which we use to capture patterns in the data. Association rules will be used to measure and compare the quality of sub-tables. Formally,

Definition 3.4 (Association rules [27]). Given a table *T* over schema *U*, an association rule *R* has the form  $\{(u_1, v_1), \ldots, (u_r, v_{r_R})\} \rightarrow \{(u_{r_R+1}, v_{r_R+1}), \ldots, (u_{r_R+p_R}, v_{r_R+p_R})\}$  where each  $u_i \in U$  is a column and each  $v_i \in \mathcal{D}_i$  is a cell value. Denote by  $U_R = \{u_1, \ldots, u_{r_R+p_R}\} \subseteq U$  the set of columns used in *R*. We say *R* holds for a tuple  $t \in T$  if  $t(u_i) = v_i$  for every  $1 \le i \le r_R + p_R$ . Denote by  $T_R \subseteq T$  the subset of tuples for which *R* holds.

Previous work includes different metrics for the quality of association rules, as well as corresponding algorithms for association rule *mining*, i.e., the discovery of all association rules that meet some quality criteria (e.g., [2, 15, 24, 28]).

In tables with diverse columns, the use of binning may improve the mined association rules [27]: given a table T, we can replace each cell value  $t(u_i)$  with an identifier of its matching bin  $B_j^i$ . The resulting table would have a smaller number of distinct values per column, and each value would occur more frequently; consequently, one may be able to mine association rules that apply to many more tuples. *Example 3.5.* Using the bins from example 3.3, the association rule from Example 1.2 stating that long flights are likely *not* to be cancelled can be written as:

cancence can be written as:  $AIR\_TIME \in B_{long}^{AT}$ ,  $DISTANCE \in B_{long}^{DIST} \rightarrow CANCELLED \in B_0^{CANC}$ . Similarly, the rule for flights likely to be cancelled can be written as  $SCHEDULED\_DEPARTURE \in B_{afternoon}^{SD}$ ,  $SCHEDULED\_ARRIVAL \in B_{afternoon}^{SA} \rightarrow CANCELLED \in B_1^{CANC}$ .

#### 3.2 Informativeness Metrics

We now develop quality metrics for sub-tables. The first type of metric that we develop intiutively measures how well data patterns in the full table are captured by the sub-table.

*Cell coverage*. Given a sub-table  $T_{sub}$  of table T and a set a set  $\mathcal{R}$  of association rules mined from T (e.g., using [27]), to measure the *coverage* of  $T_{sub}$  with respect to T and  $\mathcal{R}$  we consider the following.

- (q1) Which of the rules of  $\mathcal{R}$  are *covered*, i.e., captured by  $T_{sub}$ ?
- (q2) What is the marginal contribution of each covered rule to  $T_{sub}$ 's informativeness?
- (q3) How do marginal contributions aggregate to an overall numerical score for  $T_{sub}$ ?

Since sub-tables include a subset of the table cells, and association rules are also defined at the level of table cells, we propose below formal definitions for q1-q3 that yield a *cell coverage* metric. This metric intuitively reflects the ratio of cells in *T* that are describable by association rules in  $\mathcal{R}$  that are represented in  $T_{sub}$ .

Definition 3.6 (Cell coverage). Let T be a table,  $\mathcal{R}$  a set of association rules mined from T, and  $T_{sub}$  a sub-table of T.

- (d1) A rule  $R \in \mathcal{R}$  is said to be *covered* by  $T_{sub}$  if all the attributes of R are in  $T_{sub}$  ( $U_R \subseteq U_{sub}$ ), and there exists a tuple  $t \in T_{sub}$ for which R holds ( $\{T_{sub}\}_R \neq \emptyset$ ). Let  $\mathcal{R}_{sub}$  be the set of all rules in  $\mathcal{R}$  that are covered by  $T_{sub}$ .
- (d2) The marginal contribution of a rule  $R \in \mathcal{R}_{sub}$  is the subset of table cells it describes:
- $\operatorname{cell}(R,T) \coloneqq \{ \langle t, u \rangle \mid t \in T_R \land u \in U_R \}.$
- (d3) The *cell coverage* of  $T_{sub}$  with respect to  $T, \mathcal{R}$  is denoted by

$$\operatorname{cellCov}_{\mathcal{R}}(T, T_{\operatorname{sub}}) \coloneqq \frac{1}{\operatorname{upcov}} \left| \bigcup_{R \in \mathcal{R}_{\operatorname{sub}}} \operatorname{cell}(R, T) \right|$$
(1)

I.e., it is the (normalized) number of cells in *T* described by any covered rule in  $R \in \mathcal{R}_{sub}$ . The normalization factor upcov :=  $|\bigcup_{R \in \mathcal{R}} cell(R, T)|$  is an upper bound on the number of cells that can be covered, and ensures that cellCov $_{\mathcal{R}}(T, T_{sub}) \in [0, 1]$ .

In developing the cell coverage metric, we considered several alternative approaches. We next briefly describe some of these through an example, to illustrate the benefit of cell coverage.

Alternative coverage metrics. Consider the example table  $\hat{T}$  on the left of Figure 3, illustrating some of the trends in the *flights* dataset mentioned above. The table values represent bin names (e.g. "short", "medium", "long"). Assume a set  $\mathcal{R}$  of all association rules with column *CANCELLED* on the right, and at least two columns on the left, that hold for at least two rows. As an example, the rule *DEP.\_TIME= NaN, YEAR= 2015 →CANCELLED= 1* applies to rows 1-4. For convenience, rules of maximal size are highlighted;

$\hat{T}$	Row	CANCELLED	DEPTIME	YEAR	SCHEDDEP.	DISTANCE	_	$\hat{T}_{\rm sub}^{(1)}$	Row	CANCELLED	DEPTIME	YEAR	DISTANCE
	1	1	NaN	2015	afternoon	short			1	1	NaN	2015	short
	2	1	NaN	2015	afternoon	medium			5	0	morning	2016	medium
	3	1	NaN	2015	morning	medium			7	0	evening	2015	long
	4	1	NaN	2015	morning	short							
	5	0	morning	2016	morning	medium		A(2)					
	6	0	morning	2015	morning	medium		$T_{\rm sub}^{(2)}$	Row	CANCELLED	DEPTIME	YEAR	SCHEDDEP.
	7	0	evening	2015	evening	long			1	1	NaN	2015	afternoon
	8	0	evening	2015	afternoon	long			5	0	morning	2016	morning
							-		7	0	evening	2015	evening

Figure 3: Example Table  $\hat{T}$  with two sub-tables. Association rules (at most one per row) are highlighted.

each highlighted line illustrates a different rule, with colors alternating for clarity.

First, observe that rows with *CANCELLED*=1 are more similar to each other (have more values in common) compared with rows with *CANCELLED*=0 due to the fact that many fields do not apply (i.e. are NaN) when a flight is cancelled.

As a result, there are 13 association rules for the first 4 rows, and only 8 for the last 4. This issue is exacerbated when there are many more columns with fixed values, leading to overlapping rules including subsets of these columns. Hence, using the number of rules covered as a measure of sub-table quality would lead to selecting many representatives from repetitive patterns, whereas we would like to see representatives from different areas of the data. Consequently, we propose to use measures based on *data coverage* rather than rule coverage.

Next, consider the two sub-tables on the right,  $\hat{T}_{sub}^{(1)}$  and  $\hat{T}_{sub}^{(2)}$ , which differ only in the last attribute. These are "good" sub-tables, in the sense that they cover at least one rule for each tuple of  $\hat{T}$ . If we chose to use a row coverage metric, they would therefore have the same score. However,  $\hat{T}_{sub}^{(1)}$  covers larger rules (two of size 4 and one of size 3) compared with  $\hat{T}_{sub}^{(2)}$  (two of size 3 and one of size 4). Accordingly,  $\hat{T}_{sub}^{(1)}$  describes 28 cells of  $\hat{T}$ , whereas  $\hat{T}_{sub}^{(2)}$  describes only 26. By normalizing these numbers, we would get cell coverage of 0.78 and 0.72, respectively, since 36 cells in total can be described by association rules. This motivates us to use a *cell-based metric*.

Finally, we note that, in  $\hat{T}_{sub}^{(1)}$ , if we chose row 3 instead of 1 and row 6 instead of 5 we would have the same cell coverage. However, the sub-table would be more repetitive, containing only 2015 in the year field and two instances of medium distance. This demonstrates that coverage should be accompanied by a diversity metric, which we discuss next.

*Diversity.* To generalize the example shown above, sub-tables with high cell coverage may seem repetitive to humans, since: 1) overlapping covered association rules may lead to repeating values in the sub-table, and 2) values that do not participate in rules are ignored. We therefore combine cell coverage with a diversity metric based on pairwise Jaccard similarity. As with cell coverage, binning the data values is useful in making our diversity metric more meaningful, and we consider two values in the same bin to be similar.

$\hat{T}_{sub}^{(3)}$	Row	CANCELLED	DEPTIME	SCHEDDEP.	DISTANCE
	1	1	NaN	afternoon	short
	5	0	morning	morning	medium
	7	0	evening	evening	long

#### Figure 4: Example of a diverse sub-table

Definition 3.7 (Diversity metric). The similarity of two tuples  $t, t' \in T_{sub}$  is the ratio of cells which fall in the same bin, formally,

$$\operatorname{Jaccard}(t, t', T_{\operatorname{sub}}) := \frac{\left| \{u_i \in U_{\operatorname{sub}} \mid \exists B_j^i \in \mathcal{B}(u_i). t(u_i), t'(u_i) \in B\} \right|}{|U_{\operatorname{sub}}|}$$

We then define the diversity of  $T_{sub}$  as the complement of the average similarity between its tuples, namely,

divers
$$(T_{sub}, \mathcal{B}) \coloneqq 1 - \operatorname{avg}_{t,t' \in T_{sub}} \operatorname{Jaccard}(t, t', T_{sub})$$
 (2)

*Example 3.8.* Consider again Table  $\hat{T}_{sub}^{(1)}$  from Figure 3. Here the shown values are already bin names, so to compute the subtable diversity we need to compute the ratio of identical cells in each pair of rows. In this case, the only overlaps are in *CAN-CELLED* in rows 5 and 7, and the *YEAR* of rows 1 and 7, yielding a diversity divers  $(\hat{T}_{sub}^{(1)}, \mathcal{B}) = 1 - \operatorname{avg}(0.25, 0, 0.25) = 0.83$ . Figure 4 shows an even more diverse sub-table, with divers  $(\hat{T}_{sub}^{(3)}, \mathcal{B}) = 1 - \operatorname{avg}(0, 0, 0.25) = 0.92$ , achieved by excluding the repetitive *YEAR* column. However, this table has a smaller cell coverage, describing only 24 cells, which indicates that there is a trade-off between cell coverage and diversity.

*Optimization problem.* Based on the metrics defined above, we define the optimization problem, OPT-SUB-TABLE. This problem computes a sub-table of a predefined size which balances high cell coverage and diversity. Moreover, we allow the user to specify target columns of interest and focus the sub-table on the target columns by including them in the chosen columns and by considering only association rules that include at least one of the target columns.

More formally, we are given as input a table *T* over schema *U*, dimensions *k*, *l* (the number of rows and columns, respectively), a set of target columns  $U^* \subseteq U$  such that  $|U^*| \leq l$ , a set of association rules  $\mathcal{R}$  mined from *T*, a binning  $\mathcal{B}$  as in Def. 3.2 and a parameter  $\alpha \in [0, 1]$  which is used to balance coverage and diversity (by default,

 $\alpha = 0.5$ ). If there are target attributes  $(U^* \neq \emptyset)$ , we retain only the rules the contain them:  $\mathcal{R}^* := \{R \in \mathcal{R} \mid \{u_{r_1}, \dots, u_{r_R+p_R}\} \cap U^* \neq \emptyset\}$ . Otherwise we retain all rules:  $\mathcal{R}^* = \mathcal{R}$ . Our goal is to find a  $k \times l$  sub-table  $T_{\text{sub}}$  that includes the target attributes  $(U^* \subseteq U_{\text{sub}})$ , and that maximizes the following score among all such tables:

combined 
$$(T_{sub}, T, \mathcal{R}^*, \alpha) =$$
  
 $\alpha \cdot cellCov_{\mathcal{R}^*}(T, T_{sub}) + (1 - \alpha) \cdot divers(T_{sub}, \mathcal{B})$  (3)

*Example 3.9.* Consider again sub-tables  $\hat{T}_{sub}^{(1)}$  and  $\hat{T}_{sub}^{(3)}$  in Figures 3-4. Their combined scores for  $\alpha = 0.5$  are:

 $0.5 \cdot \frac{28}{36} + 0.5 \cdot 0.83 = 0.80$  and

 $0.5 \cdot {}^{24}/_{36} + 0.5 \cdot 0.92 = 0.79$ , respectively. In fact,  $\hat{T}^{(1)}_{sub}$  is the optimal sub-table for this example.

Unfortunately, we will show in the next section that directly optimizing this problem is infeasible, and therefore give in Section 5.1 a practical solution that indirectly accounts for association rules using table embedding.

#### 4 COMPLEXITY

We now analyze the complexity of OPT-SUB-TABLE. For simplicity, we will ignore the use of target columns and of binning, since in the extreme case the set of target columns may be empty and binning may assign each value to a separate bin. We mostly focus here on the sub-problem of MAX-CELL-COVER, namely, finding the sub-table with maximal cell coverage, i.e. solving OPT-SUB-TABLE with  $\alpha = 1$ . We denote by DEC-CELL-COVER the corresponding decision problem: given a table *T* over schema *U*, dimensions *k*, *l*, a set of association rules  $\mathcal{R}$  and a threshold  $\Theta$ , decide whether there exists a sub-table  $T_{sub}$  of size  $k \times l$  whose cell coverage is cellCov $_{\mathcal{R}}(T, T_{sub}) \ge \Theta$ .

#### 4.1 Hardness

Let *n* and *m* be the numbers of tuples and columns, respectively, in the input table *T*. A brute-force algorithm can theoretically traverse all  $O(n^k \cdot m^l)$  sub-tables of size  $k \times l$  and find the one with maximal score. While this algorithm is polynomial in the size of *T*, it is infeasible due to the exponential dependency in *k*, *l*. For reasonable dataset and sub-table sizes, e.g., for n = 10,000 and m, k, l = 5, we need to check  $10^{23}$  sub-tables.

Since k is small, a solution that is exponential in k may still be feasible, but not when the basis is n. Thus, we examine whether our problem is *fixed-parameter tractable*: FPT is class of fixed-parameter tractable problems, defined by having a solution in time  $O(poly(n) \cdot f(k))$  where n is the input size, f is a function and k is a parameter. Unfortunately, we can show that our problem is probably not in FPT by the following proposition.

PROPOSITION 4.1. Given a  $n \times m$  table T over schema U, and sub-table dimensions k, l. Dec-Cell-Cover is W[2]-hard with respect to n = |T| and k as a parameter, assuming m, l = O(n).

PROOF. Dominating Set is the problem of, given an undirected graph of n vertices, select k vertices such that every vertex in the graph is either in the selected set or is connected by an edge to a vertex in the set. This problem is known to be W[2]-complete [13]. The W-hierarchy is a hierarchy of classes defined by properties of the translation of problems into combinatorial circuits. It is known that  $FPT=W[0]\subseteq W[1]\subseteq W[2]\subseteq \ldots$ , and conjectured that this hierarchy is strict, i.e.,  $W[0]\subset W[1]\subset W[2]\subset \ldots$ . We show a reduction from Dominating Set to Dec-Cell-Cover, thus showing the latter is also W[2]-hard and hence not in FPT unless FPT=W[2].

Dominating Set  $\leq$  Dec-Cell-Cover. Given a graph G = (V, E), define a table T with a tuple  $t_v$  for every  $v \in V$ , and an attribute  $u_v$  for every  $v \in V$ , such that  $t_v(a_u) = 1$  iff  $(u, v) \in E$  or u = v, and is NULL otherwise. Let  $\mathcal{R}$  be composed of association rules of the form  $\{u_v \in \{1\}\} \rightarrow \{\}$  is in for every  $u_v$ , and seek a summary of size  $k \times |V|$ , i.e., no column selection is required. In this case, k tuples that cover all the non-NULL cells in I correspond to a dominating set of size k, and vice versa.

The above proof assumes that m, l = O(n), i.e., the number of attributes is large. However, in practical cases it is often assumed that  $m \ll n$ . In this case, our reduction does not apply, and in particular, if m = O(1), a dominating set with bounded degree is not W[2]-hard. However, we can show NP-hardness in k.

PROPOSITION 4.2. Dec-Cell-Cover is NP-hard in k, the number of tuples selected for the summary, even assuming the number of attributes m = O(1).

PROOF. The proof is by a reduction from Vertex Cover, which is known to be hard in k for the selection of k vertices that are adjacent to every edge in a given graph, even if the maximal degree of a vertex is 3. Given a graph G = (V, E) we assign each edge  $e \in E$  a serial number num(e) and define an input table I with 5 attributes and n tuples, such that for each edge  $(u, v) \in E$  there exists tuples  $t_u, t_v \in I$  and an attribute a where  $t_u(a) = t_v(a) = \text{num}((u, v))$ . Since each  $v \in V$  appears in at most 3 edges, the other edges of u and v occupy at most 4 attributes of  $t_u$ and  $t_v$ , and hence we can use the fifth attribute for num((u, v)). Now, similarly to the proof of Prop. 4.1, we will construct the rules such that  $\{a \in \{\text{num}((e)\}\} \rightarrow \{\}$  is a rule in  $\mathcal{R}$ , and so, the maximum coverage summary of size  $k \times 5$  covers all the non-NULL cells iff the vertices corresponding to its tuples form a vertex cover in G.  $\Box$ 

Hardness of diversity optimization. We have so far focused on the hardness of cell coverage maximization (corresponding to solving OPT-SUB-TABLE with  $\alpha = 1$ ). However, we note that diversity maximization (corresponding to OPT-SUB-TABLE with  $\alpha = 0$ ) is also hard: this problem (without column selection) was proven to be NP-hard using an analogous diversity definition in the context of selecting a diverse group of crowd workers [33], and the results hold for our setting as well. Since OPT-SUB-TABLE is strictly harder than both sub-problems, we can conclude that it is NP-hard.

#### 4.2 Approximate Solutions and Limitations

Given the hardness results above, we consider approximate solutions to Max-Cell-Cover, i.e., computing sub-tables with approximatelyoptimal score. In some cases, it may be feasible to enumerate all possible combinations of selecting *l* attributes for the summary, e.g., when m = O(1) or *l* and *m* are very close. In such cases, we have an approximation algorithm, as stated by the following proposition.

PROPOSITION 4.3. Given a table T over U, where |T| = n and |U| = m and a set  $\mathcal{R}$  of association rules, Algorithm 1 computes a

**ColumnSelection**  $(T, k, l, \mathcal{R})$  // Table, dimensions and association rules  $T^*_{\text{sub}} \leftarrow \emptyset, \text{ cov}^* \leftarrow -1;$ 1 for  $U' \subseteq U$  such that |U| = l do 2 // Projection of T on U' $T' \leftarrow \pi_{II'}T;$ 3  $T_{\text{sub}}, \text{cov} \leftarrow \mathbf{GreedyRowSelection}(T', k, \mathcal{R});$ 4 if  $\operatorname{cov} > \operatorname{cov}^*$  then  $\operatorname{cov}^* \leftarrow \operatorname{cov}, T^*_{\operatorname{sub}} \leftarrow T_{\operatorname{sub}};$ 5 return  $T^*_{sub}$ ; 6 **GreedyRowSelection**  $(T', k, \mathcal{R})$  $T_{\text{sub}}^{**} \leftarrow \emptyset, \text{ cov}^{**} \leftarrow -1;$ 7 for i in  $1 \dots k$  do 8  $T_{sub}^{*} \leftarrow T_{sub}^{**}, \text{ cov}^{*} \leftarrow \text{ cov}^{**};$ for  $t \in T' - T_{sub}^{*}$  do  $| \text{ cov} \leftarrow \text{cellCov}_{\mathcal{R}}(T, T_{sub});$ 9 10 11 if  $\operatorname{cov} > \operatorname{cov}^*$  then  $\operatorname{cov}^* \leftarrow \operatorname{cov}, \ T^*_{\operatorname{sub}} \leftarrow T_{\operatorname{sub}};$ 12  $\begin{vmatrix} \text{cov}^{**} \leftarrow \text{cov}^{*}, \ T_{\text{sub}}^{**} \leftarrow T_{\text{sub}}^{*}; \\ \text{return } T_{\text{sub}}^{**}, \text{cov}^{**}; \end{vmatrix}$ 13 14 Algorithm 1: Greedy Sub-Table Selection.

 $t \times l$  sub-table  $T_{sub}$  such that  $cellCov_{\mathcal{R}}(T, T_{sub}) \ge (1 - \frac{1}{e})$  OPT where OPT is the score of the optimal solution to Max-Cell-Cover.

PROOF. Algorithm 1 includes two functions. The ColumnSelection enumerates over the possible column selections and for each computes a sub-table using the function GreedyRowSelection. The latter function iteratively attempts to add each single row to the current sub-table, computes the cell coverage score and records the sub-table with maximal cell coverage. This is repeated k times to select k rows in total. Thus, for each column selection we greedily compute a sub-table, and among all column combinations we take the best one.

To prove the approximation bound, note that for a fixed set of attributes, the cellCov<sub> $\mathcal{R}$ </sub> function is non-negative, monotone and submodular with respect to tuples (adding a tuple only increases the score, and as we add tuples the marginal contribution of tuples can only decrease). Therefore, by the well-known result of [23] a greedy algorithm approximates the optimum by the above-mentioned multiplicative factor. Since we enumerate over all column combinations of size *l*, we achieve this ratio in particular for the same column selection as the optimal sub-table with cell coverage OPT.

By the analysis above, the greedy Algorithm 1 is clearly not feasible in the general case due to the need to enumerate all  $\binom{m}{l}$  options for column selection. Note that we cannot greedily select columns, since the cell coverage metric is *not* sub-modular with respect to columns, due to multi-column association rules. The algorithm also does not take diversity into account.

In Section 6 we show that even a "semi-greedy" variation of Algorithm 1, which traverses the column combinations in a random order, can take more than two days to run on an industrial-grade server and is therefore still impractical. Furthermore, halting the algorithm after some fixed time period before enumerating all  $\binom{m}{l}$  possibilities for column selection is not guaranteed to meet the approximation guarantee.

**Pre-processing**  $(\tilde{T})$  // Raw table  $T \leftarrow$  normalize and bin  $\tilde{T}$ ; 1  $S \leftarrow$  rows and columns of T as text; 2  $\mathcal{M} \leftarrow \text{Word2Vec}(S, \text{windowSize} = \max\{n, m\});$ 3 // Embedding Computation return M: // cell-to-vector model:  $\mathcal{M} : T \times U \to \mathbb{R}^{\gamma}$ 4 **Centroid-based Selection**  $(T, k, l, Q, U^*, \mathcal{M})$ rowVecs, colVecs  $\leftarrow$  empty dictionaries; 5 if  $Q \neq$  NULL then  $T \leftarrow Q(T)$ ; 6  $U \leftarrow \text{columns of } T;$ 7 for  $t \in T$  do 8  $v \leftarrow \operatorname{avg}_{u \in U}(\mathcal{M}(t(u)));$ 9 rowVecs  $\leftarrow$  rowVecs  $\cup$  { $v \mapsto t$ }; 10  $C \leftarrow \text{cluster}(\text{rowVecs}, k);$ 11  $T_{sub} \leftarrow rowVecs. getValues(centroids(C));$ 12 13 for  $u \in U - U^*$  do  $v \leftarrow \operatorname{avg}_{t \in T}(\mathcal{M}(t(u)));$ 14 colVecs  $\leftarrow$  colVecs  $\cup \{v \mapsto u\};$ 15 16  $C \leftarrow \text{cluster}(\text{colVecs}, l - |U^*|);$  $U_{sub} \leftarrow U^* \cup colVecs. getValues(centroids(C));$ 17 18  $T_{\text{sub}} \leftarrow \Pi_{U_{\text{sub}}} T_{\text{sub}};$ return T<sub>sub</sub>, U<sub>sub</sub>; 19

Algorithm 2: SubTab Algorithm for Sub-Table Selection.

#### **5** PRACTICAL SOLUTION

As discussed in Section 3, an ideal sub-table captures a large and diverse set of patterns in the full table. However, optimizing based on a combined score of *cell coverage* and *diversity* is NP-hard, and even approximated solutions are impractical for interactive exploration (Section 4). Therefore, to generate good sub-tables in interactive times (up to several seconds) we take a different approach, based on *tabular embeddings*. The embedding captures bin co-occurrences, and therefore roughly corresponds to frequent itemsets and association rules.

We begin by describing our sub-table selection algorithm, and then discuss its benefits compared to other approaches.

#### 5.1 Sub-table Selection

Our algorithm, shown in Algorithm 2, includes two parts: (1) Preprocessing, in which we compute a vector representation for each cell in the full table T using *table embedding*, and (2) centroid-based sub-table selection, which utilizes the embedded vectors to quickly select a sub-table. Importantly, pre-processing is performed only once, when the table T is loaded, and centroid selection is performed for each exploratory query that the analyst performs over the table Tto produce the corresponding sub-table.

Pre-Processing. Given a raw table T, the first step

is to normalize the values (e.g., remove illegal characters) and bin continuous columns so that values are replaced by their bin name (e.g., binning splits the *Distance* column into short, medium and long distances). Let  $\tilde{T}$  be the normalized, binned table.

We then use a *table embedding process* to generate a real-valued vector for each cell in the table  $\tilde{T}$ . Note that while several recent

works [6, 7, 12, 29, 35] suggest more complex methods for embedding tabular data, e.g., based on graph representations, auto-encoders, etc., we use a simpler yet effective method that quickly computes the vector representations based on [5]. (See Section 6.2 for a comparison with [7]).

Our embedding method transforms the table into a corpus of *sentences* and then uses a fast implementation of word embedding [21]. The corpus consists of *tabular sentences* in which each cell in the table *T* represents a single word. We use two types of sentences: *tuple-sentences*, containing values in each tuple  $t \in T$ , and *column-sentences* that cover the values in  $T(u), \forall u \in U$ . To achieve even faster execution times, we limit the corpus size to 100*K*, where the sentences are chosen uniformly at random.

Using the corpus of tabular sentences we then train a wordembedding [21] model that outputs a numeric vector representation for the table cells. The learned representation of the cells are based on co-occurrences of values in the same row/column, and hence capture recurring patterns which roughly correspond to association rules. The output of this process, as depicted in Line 4, is a mapping between each cell in  $T_{ij}$  to a corresponding, learned *cell-vector*  $\mathcal{M}_{ij}$ .

*Centroid-Based Sub-table Selection.* Once the vector representations  $\mathcal{M}$  are computed for each cell in T, we perform a fast yet effective sub-table selection based on the vectors' centroids. As mentioned above, this is done over the results of each exploratory query performed by the user.

We select a sub-table  $T_{sub}$  of size  $k \times l$  as follows: First, to select the k tuples in  $T_{sub}$  we compute for each tuple  $t \in T$  with columns U, a *tuple-vector*, by taking the component-wise average of its corresponding cell-vectors. Namely, we average the cell-vectors  $\mathcal{M}(t(u_1)), \mathcal{M}(t(u_2)), \ldots, \mathcal{M}(u_{|U|}))$ 

(lines 8-10).

This allows us to obtain a unified representation of each tuple in the input table. We then cluster the tuple-vectors into k clusters and select their centroids as the rows of  $T_{sub}$ . Next, to select the columns of  $U_{sub}$  we perform a similar process, creating *columnvectors*, forming clusters, and finding their centroids.

Since the columns of  $U^*$  must be included in the sub-table, we exclude them from clustering, compute only  $l - |U^*|$  clusters and then add the  $U^*$  columns to the selected centroids.

#### 5.2 Discussion

We conclude this section with three important observations regarding our embedding-based approach for sub-table generation, which explain why our system works well in practice despite the fact that it is not explicitly based on association rules.

First, note that since SubTab does not directly attempt to optimize the cell-coverage and diversity metrics, there are no guarantees it will always obtain high scores.

However, as our experiments will show (Section 6), SubTab computes high quality sub-tables in terms of both cell-coverage and diversity when the underlying rules in T are prominent (rather than arbitrary)

We also show that directly optimizing the metrics, using either a more-feasible *semi-greedy* variation of Algorithm 1 or a *Multi Armed Bandit* sampling algorithm takes over 24 *hours* to achieve the scores SubTab obtains in several seconds. Intuitively, the reason that our embedding-based algorithm achieves good results is twofold. (1) SubTab achieve good cell-coverage scores since the embedded vectors, which are generated based on the frequency of co-occurrences of values in the same rows and columns, capture underlying frequent patterns – as is also done in association rules mining. (2) We achieve a high score for diversity due to the centroid-based columns and tuples selection.

As mentioned earlier, obtaining vector representations for table cells could be done using several different techniques [5–7, 12, 29, 35]. In our experimental evaluation, we compare SubTab to a baseline which uses the embedding method suggested in [7], which is geared towards data integration tasks. We show that not only does SubTab complete the pre-processing phase 26X faster (90 seconds, rather than 40 minutes), but that it also obtains superior scores in terms of cell-coverage and diversity.

Finally, note that it is also possible to apply clustering directly on T, without first generating an embedded representation. This method is also inferior to SubTab as it relies on a "one-hot-encoding" of the data, which does not capture the underlying patterns as well as the embedding-based method (see Section 6).

#### **6** EXPERIMENTS

We performed an extensive experimental evaluation of SubTab in terms of both the quality and usefulness of the resulted sub-tables as well as its running-times. After describing the experimental setup, we report our results. First, to test the sub-table quality (Section 6.2), we conducted two sets of experiments: (1) a twofold user study, where the participants not only rank the usefulness of the sub-tables through a questionnaire, but are also required to list insights and conclusions they identify in each sub-table; and (2) an offline, simulation-based study, in which we retraced real-life analysis sessions, generated a sub-table for each exploratory query, then checked whether the parameters of the next query in the session (e.g., selection term, aggregation column) appear in the sub-table. We also examined whether our combined metric of cell-coverage and diversity is correlated with these quality evaluations.

The final two experiments measure the running-time of SubTab for each dataset (Section 6.3), and the performance using different parameter settings for the packages used in SubTab (Section 6.4). A summary of findings from the experiments are in Section 6.5.

#### 6.1 Experimental Setup

SubTab is implemented in Python 3.8 as a local Python library that hooks into Pandas [31] and therefore can be used, e.g. in common EDA environments such as Jupyter notebooks. The binning method used is based on kernel density estimation and is implemented with *sciPy*<sup>4</sup>. The Word2Vec embedding method is implemented by *gensim*<sup>5</sup>. Centroid selection is performed by creating clusters via KMeans using *sklearn*<sup>6</sup>.The experiments were run on Intel Xeon CPU based server with 24 cores and 96 GB of RAM.

*Metrics implementation.* As the cell coverage metric relies on association rules, we provide details about how they are mined. We compute the association rules using the Apriori Algorithm [2] and

<sup>4</sup>https://scipy.org/

<sup>&</sup>lt;sup>5</sup>https://radimrehurek.com/gensim/

<sup>&</sup>lt;sup>6</sup>https://scikit-learn.org/





Figure 6: Simulated Experiments Results (CY)

implement it using *efficient-apriori*<sup>7</sup>; we set the main parameters, support and confidence, to 0.1 and 0.6, respectively, and the minimum rule size to 3. In Section 6.4, we conduct several experiments that test the effect of each of the parameters, by varying the given parameter while setting all others to their default values. When target columns are selected by the user, the data is split according to the binned values of the target columns. The rules are then mined over each subset separately. An optional extension to our system is coloring the patterns (association rules) in the data that are represented by the sub-table, as illustrated in Figure 3; this was found very helpful by participants in the user-study (Section 6.2). Lastly, to implement our combined cell coverage and diversity metric, we take  $\alpha = 0.5$ by default for the combined score, assigning equal weights to cell coverage and diversity.

Datasets. To demonstrate the performance of SubTab in different domains, we used the following datasets:

- Flights (FL), <sup>8</sup> which has 6M rows and 32 columns
- Cyber-security  $(CY)^9$ , which has 30K rows and 15 columns
- Spotify (SP), <sup>10</sup> which has 42K rows and 15 columns, and
- Credit card frauds (CC), <sup>11</sup> which has 250K rows and 31 columns.

- US Funds (USF), <sup>12</sup> which has 23.5K rows and 298 columns.
  Bank Loans dataset (BL) <sup>13</sup> with 110K rows 19 columns.

Baselines. We have tested two types of baselines. The following baselines support fast response time and are suitable for EDA. They were used in our quality analysis and user study.

- (1) Random (RAN): select uniformly at random k rows and lcolumns. To increase the quality of this baseline, we iteratively repeat the random selection for one minute, and return the sub-table with highest score among all the randomly drawn sub-tables.
- (2) Naive clustering (NC):

We first transform the categorical and textual columns to be continuous values using one-hot encoding<sup>14</sup>. Then we treat each row as a vector of length m (the total number of columns), and cluster the vectors using K-means. The centroids of those clusters are used as the rows in the subtable. We select the columns analogously.

The second set of baselines that we test are too slow to be used in an interactive setting, having running times of over 30 minutes. We nevertheless use them for comparing the quality of our system.

- (4) Multi-Armed Bandit (MAB): we use a version of the Multi-Armed Bandit algorithm [26] that, in each iteration, selects a set of *n* rows and *k* columns and evaluates the sub-table using our metric. The reward (i.e. the cell coverage score) is given to all the columns and rows that participated in the sub-table, and the exploration-exploitation method used is Upper Confidence Bound (UCB) [19].
- (5) Greedy sub-table selection (Greedy): We modify the greedy algorithm outlined in Algorithm 1 by traversing the column combinations in random order (line 2). We can thus halt the algorithm after any number of iterations and use the sub-table with maximal score among the ones found until that point. For our experiments, we use a time limit of 5 hours, which we empirically found to be required for discovering informative sub-tables.
- (6) *EmbDI*:[7]: This algorithm creates a local embedding (inspired by Node2Vec) that is effective for data integration tasks in relational databases. The table is transformed into a graph by representing the columns and rows as nodes, connected by edges which capture structural relationships. The structure created provides a more efficient graph computation than the naive graph transformation. Thus, it captures relationships inherent in the tables, and provides a more efficient computation than the naive graph transformation.

#### 6.2 **Quality Analysis**

To assess the quality of SubTab, we evaluated the sub-tables generated by SubTab, using several experiments: (1) We performed a "live" user-study, where participants used SubTab as well as other baselines in real-life analysis tasks; (2) Offline simulation-based evaluation, where we retraced completed EDA sessions and assessed the usability of a sub-table by determining if it contains elements of

<sup>&</sup>lt;sup>7</sup>https://pypi.org/project/efficient-apriori/

<sup>&</sup>lt;sup>8</sup>https://www.kaggle.com/usdot/flight-delays?select=flights.csv

<sup>9</sup>https://www.honeynet.org/challenges/

<sup>10</sup> https://www.kaggle.com/c/bfh-spotify-challenge/data

<sup>11</sup> https://www.kaggle.com/mlg-ulb/creditcardfraud

<sup>12</sup>https://www.kaggle.com/stefanoleone992/mutual-funds-and-

etfs?select=MutualFunds.csv

<sup>13</sup> https://www.kaggle.com/panamby/bank-loan-status-dataset

<sup>&</sup>lt;sup>14</sup>https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html

the subsequent query; and (3) Evaluation based on our metrics - to further compare SubTab to the baselines, we evaluated the generated sub-tables in terms of our cell coverage and diversity metrics to test whether our metric score is highly correlated with the user-study and simulation-based evaluation. In this part, we also compare our results to slower baselines that could not be tested in the live user study. These experiments are discussed below.

6.2.1 User Study. We conducted a user study to compare the usefulness of SubTab to the baseline approaches. We recruited 15 participants, all with varying degree of expertise in data analysis using Pandas. The participants were first asked to perform an actual data-analysis task in which they used the sub-table to discover insights about a dataset, and then answer a short questionnaire about the quality of the sub-tables. We divided the participants to equal groups, each worked with a different baseline on three different datasets: *SP*, *FL* and *BL*. As this is a "live" experiment, we compared SubTab with the baselines *RAN* and *NC*, which are fast enough for interactive analysis.

For each dataset there was an exploration task involving several queries. Each user was given one baseline, and performed explorations over all three datasets. The user's goal was to write down insights that are relevant to the given task while examining the subtables that were created during the exploration. We then counted the number of correct insights that users had, and averaged the results for each exploration task, per user. To help the users in their exploration task, we also colored the patterns (association rules) that were captured in the sub-table for all the baselines, using the *SP* and *FL* dataset. In the *BL* exploration, we did not color the patterns (in any baseline), displaying only the created sub-table without additional information. By doing this, we tested whether the trends of the exploration task without coloring remains the same as those with coloring.

Insights Discovery Experiment. For each dataset, the participants were presented a notebook containing several exploratory queries, and a *sub-table* of the queries' results (generated by either SubTab, *RAN*, or *NC*). The participants were then instructed to examine each notebook and derive insights about the dataset's particular analysis task. For example, the task in *SP*, containing data about songs and their popularity in the Spotify streaming service, was to discover "*what makes songs popular*". We then manually evaluated the *correctness* of the participants' insights, and removed ones that were statistically incorrect or highly irrelevant to the analysis task.

Table 1 shows the total number of insights, percentage of correct insights, as well as the percentage of users who did not derive insights at all, averaged across all three datasets. First, see that when using SubTab, users derived an average of 4 correct insights per dataset, which is 3X more than *RAN* and 12X more than *NC*. Interestingly, the percentages of correct insights obtained by SubTab is 85% whereas only 30% and 6% of the insights obtained via *RAN* and *NC* (resp.) were correct. Inspecting the incorrect insights, we observed that the users reached false conclusions since many of the sub-tables produced by *RAN* and *NC* were *misleading*. These sub-table contained, for example, non-representative distribution of columns, or presented a random, false correlation between columns. Last, see that using SubTab 100% of users were able to successfully



#### Figure 7: (a) Quality score and (b) Total running time

finish the analysis task, whereas none of the users failed to derive at least one insight.

*Questionnaire Results.* As mentioned above, after the participants completed the insights discovery tasks, they were each given a short questionnaire asked to evaluate the sub-tables on a scale of 1 (strongly disagree) to 5 (strongly agree) according to the following statements:

- Q1: The presented system is better than the standard dataframe sub-table.
- Q2: Would you like to use the sub-table system in future data exploration tasks?
- Q3: The sub-tables' columns were relevant to the queries.
- Q4: The sub tables' rows are representative and capture patterns.

The questionnaire results are summarized in Figure 5. Note that the average users' ranking of SubTab is above 4 for all statements, and are significantly higher than *RAN* and *NC*.

6.2.2 Simulation-Based Study. We conducted an additional offline experiment, in order to further evaluate the quality sub-tables. We used a publicly-available collection of 122 data exploration sessions [22], containing select, project, group-by, and sort operations over the dataset CY. To evaluate the potential usefulness of the sub-tables we replayed each query in a session, and generated a corresponding sub-table using SubTab and the baselines *RAN* and *NC*. We then examined whether the next query in each session contain a fragment (e.g., a group-by attribute, selection term, etc.) that appears in the sub-table of the previous query's results. Intuitively, appearance of next-query fragments in the sub-table, may imply that the sub-table is useful in selecting the next exploration step.

The percentage of captured query fragments are shown in Figure 6, when varying the width (i.e., number of columns) of the subtable from 3 to 7 (out of the 12 columns of the *CY* dataset). See again that SubTab significantly outperforms the baselines, and is able to predict 14% (width=3) to 38% (width=7). Naturally, the results improve as the sub-table covers more columns, however it is still very difficult to cover all query fragments (e.g., practically any value from a column's value domain can be used as a selection term).

*6.2.3 Quality by Our Metrics.* So far we have evaluated the quality of sub-tables based on their performance in external tasks. We now compare to our intrinsic quality metrics: cell coverage, diversity and combined score.

*Comparison to the interactive baselines.* Figure 8 shows the three scores for the three baselines over the FL, SP and CY datasets. For



Figure 8: quality metrics for different baselines and datasets

the three datasets, SubTab achieves a significantly higher cell coverage and combined scores, compared with the baselines. Interestingly, in FL and CY it also achieves a higher diversity score, which means it outperforms the baselines for any choice of  $\alpha$ . In SP, RAN has a slightly better diversity score, but its cell coverage is extremely low. For example, for the *SP* dataset, SubTab achieves a total score of 0.68, were the *RAN* and *NC* achieve 0.47 and 0.51 respectively.

To connect between intrinsic and external metrics, we now compare their ranking of baselines. In our user study (Section 6.2.1), we compute the combined score for each presented sub-table, and average per baseline. The resulting average scores for SubTab, RAN and NC were 0.56, 0.32 and 0.15 respectively, which matches the ranking of these baselines in terms of user ratings (Figure 5). Similarly, for the simulation based study (Section 6.2.2), we have computed the combined score for each computed sub-table, and averaged per baseline and per sub-table size. The resulting ranking between baselines per sub-table size was identical to the ranking according to the percentage of matched steps (Figure 6). This indicates that our metrics correlate with human judgements and with the usefulness of sub-tables in EDA sessions.

Comparison to slower baselines. We have also executed the slower, non-interactive baselines over the FL dataset, and show, in Figure 7, their performance in terms of quality and time and compared with SubTab. SubTab achieves the same combined score as the EmbDI baseline; however, the latter takes 40 minutes to execute, whereas SubTab requires only 1.5 minutes. MAB achieves the worst quality, even though it is executed for a long time. Finally, the Greedy baseline slightly outperforms the other baselines in terms of quality, but is the slowest one – this score was achieved by executing it for 48 hours on a multi-process architecture. Overall, this shows that SubTab computes high-quality tables at interactive speed.

#### 6.3 Effectiveness of Pre-processing

Recall that SubTab has two distinct steps: Pre-processing, which is executed once upon loading a data table, and Selection, which is executed with each SubTab display, both for the table itself and for queries over it (see figure 1). Figure 9 shows the execution times for each step over different datasets. Pre-processing takes the longest time, 90 seconds, for the CC dataset, although it is smaller than FL. The reason is that this data contains only numeric columns that must undergo binning. Still, this is a reasonable time for the set-up phase of an EDA session. Then, the Selection phase takes only a few seconds for all the datasets. We have tested the computation time for



Figure 9: Average running time of SubTab

various sub-table sizes, and the results were similar (the difference is less than 10%). This shows that our reuse of embeddings is indeed effective in achieving fast response time.

#### 6.4 Parameter Tuning

Our metric of cell coverage (formula 1) is measured with respect to an input set of association rules. We have explained above, in Section 5.1, why the embeddings used by our solution implicitly capture the data patterns defined by prominent association rules. We now unwrap the imprecise notion of "prominence" by considering parameters that affect the set of mined association rules, and showing that SubTab performs well for varying values for these parameters. The averaged results for *FL* and *SP* datasets are shown in Figure 10. In each graph we vary one parameter, and use the default value for the others. Note that the sub-tables computed by each of the algorithms *are the same* across all settings, since these algorithms do not use the association rules as input; the variation in parameters only affects the means of evaluating the sub-tables.

The first parameter that we consider is the number of bins per binned column (the default number is 5). A larger number of bins would imply more association rules (since there are more bin combinations) but with lower significance, i.e., they would hold for less tuples. In Figure 10a we can see that the cell coverage achieved by SubTab is much higher than the other baselines, and that this score moderately decreases for all three with the increase in number of bins. Indeed, if association rules hold for less tuples, a sub-table would need to cover more rules in order to describe the same amount of cells.

Next, recall that *support* [2] reflects the ratio of tuples to which an association rule applies. By default, we set the minimum threshold for support to be 0.1, i.e., we are interested only in rules that hold for at least 10% of the tuples. In Figure 10b we vary the support



Figure 10: Parameter tuning experiment

threshold, and observe it only leads to a minor decrease in cell coverage, for the three algorithms. Intuitively, if we increase the threshold, less rules pass it, but these rules are prominent, so the embedding is still likely to capture them.

The *confidence* of an association rule measures the strength of the connection between its parts, i.e., the ratio, among all tuples for which the left-hand-side holds, of tuples where the right-hand-side holds as well. We have also varied the confidence threshold for association rules (by default, it was 0.1) in Figure 10c. The observed trends are similar to the varying of support.

These results show the robustness of our approach across different properties of the association rules against which they are evaluated. In particular, the ranking between algorithms, and the relative gap between their scores is preserved across settings.

#### 6.5 Findings

Results of these experiments highlight the quality of sub-tables computed by SubTab, and show that they exceed those of other interactive algorithms and are comparable even to algorithms that directly optimize our metrics or use time-consuming state-of-the-art embedding methods. They also show that, unlike the baselines, SubTab is suitable for an interactive setting.

From a usability perspective, experiments with pre-recorded EDA sessions show that SubTab outperforms other baselines by more frequently including columns and rows that were later used. The user study confirms that, as compared to the baselines, our sub-tables more frequently help data analysts derive useful insights from the data and increase user satisfaction.

Finally, the results also indicate that our metrics of sub-table quality are sound and robust, and correlate with external means of evaluating sub-table quality.

#### 7 CONCLUSION AND FUTURE WORK

This paper presents SubTab, a framework for creating small, informative sub-tables of large data tables to facilitate the first step in data analytics: data exploration. Given a larger table, SubTab creates a sub-table, with a small subset of rows of the table projected over a small subset of columns, that could be explored manually by the analyst. The rows and columns are chosen as representatives of prominent data patterns within and across columns in the input table. SubTab can also be used for query results, enabling the user to quickly understand them and determine subsequent queries.

There are several directions for future research. Our current work considers only single dataset as input. We could consider handling multiple datasets, as well as optimizing sub-table computation for operations over multiple tables such as joins. Another intriguing future direction is creating sub-tables for other data science tasks, such as visualization and training ML models, both for the supervised and unsupervised setting.

As we extend our approach to different tasks, one could explore different methods for table embedding, that could be performed offline or within a longer period. Finally, there are many other challenging variants of sub-table computation, e.g., computing sub-tables that meet certain fairness requirements with respect to the data they represent.

#### REFERENCES

- [1] Sameer Agarwal, Barzan Mozafari, Aurojit Panda, Henry Milner, Samuel Madden, and Ion Stoica. 2013. BlinkDB: queries with bounded errors and bounded response times on very large data. In Proceedings of the 8th ACM European Conference on Computer Systems. 29–42.
- [2] Rakesh Agrawal, Ramakrishnan Srikant, et al. 1994. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, Vol. 1215. Citeseer, 487–499.
- [3] Brian Babcock, Surajit Chaudhuri, and Gautam Das. 2003. Dynamic sample selection for approximate query processing. In *Proceedings of the 2003 ACM* SIGMOD international conference on Management of data. 539–550.
- [4] L. Battle, M. Stonebraker, and R. Chang. 2013. Dynamic reduction of query result sets for interactive visualizaton. In 2013 IEEE International Conference on Big Data. 1–8. https://doi.org/10.1109/BigData.2013.6691708
- [5] Rajesh Bordawekar and Oded Shmueli. 2019. Exploiting Latent Information in Relational Databases via Word Embedding and Application to Degrees of Disclosure.. In CIDR.
- [6] Riccardo Cappuzzo, Paolo Papotti, and Saravanan Thirumuruganathan. 2020. Creating embeddings of heterogeneous relational datasets for data integration tasks. In SIGMOD.
- [7] Riccardo Cappuzzo, Paolo Papotti, and Saravanan Thirumuruganathan. 2020. Creating Embeddings of Heterogeneous Relational Datasets for Data Integration Tasks. Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (May 2020). https://doi.org/10.1145/3318464.3389742
- [8] Girish Chandrashekar and Ferat Sahin. 2014. A survey on feature selection methods. *Computers & Electrical Engineering* 40, 1 (2014), 16–28.
- [9] Andrzej Cichocki. 2014. Era of big data processing: A new approach via tensor networks and tensor decompositions. arXiv preprint arXiv:1403.2048 (2014).
- [10] Graham Cormode. 2017. Data sketching. Commun. ACM 60, 9 (2017), 48-55.
- [11] John P Cunningham and Zoubin Ghahramani. 2015. Linear dimensionality reduction: Survey, insights, and generalizations. *The Journal of Machine Learning Research* 16, 1 (2015).
- [12] Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. 2020. Turl: Table understanding through representation learning. arXiv preprint arXiv:2006.14806 (2020).
- [13] Rodney G. Downey and Michael R. Fellows. 1999. Parameterized Complexity. Springer. https://doi.org/10.1007/978-1-4612-0515-9
- [14] Alex Galakatos, Andrew Crotty, Emanuel Zgraggen, Carsten Binnig, and Tim Kraska. 2017. Revisiting Reuse for Approximate Query Processing. Proc. VLDB Endow. 10, 10 (June 2017), 1142–1153. https://doi.org/10.14778/3115404. 3115418
- [15] Dimitrios Gunopulos, Roni Khardon, Heikki Mannila, Sanjeev Saluja, Hannu Toivonen, and Ram Sewak Sharm. 2003. Discovering all most specific sentences. ACM TODS 28, 2 (2003).
- [16] Isabelle Guyon and André Elisseeff. 2003. An introduction to variable and feature selection. *Journal of machine learning research* 3, Mar (2003), 1157–1182.
- [17] Joseph M Hellerstein, Ron Avnur, Andy Chou, Christian Hidber, Chris Olston, Vijayshankar Raman, Tali Roth, and Peter J Haas. 1999. Interactive data analysis: The control project. *Computer* 32, 8 (1999), 51–59.
- [18] George H John, Ron Kohavi, and Karl Pfleger. 1994. Irrelevant features and the subset selection problem. In *Machine learning proceedings 1994*. Elsevier,

121-129.

- [19] T.L Lai and Herbert Robbins. 1985. Asymptotically efficient adaptive allocation rules. Advances in Applied Mathematics 6, 1 (1985), 4–22. https://doi.org/10. 1016/0196-8858(85)90002-8
- [20] Ziyang Liu, Peng Sun, and Yi Chen. 2009. Structured search result differentiation. Proceedings of the VLDB Endowment 2, 1 (2009), 313–324.
- [21] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NeurIPS*.
- [22] Tova Milo and Amit Somech. 2018. Next-Step Suggestions for Modern Interactive Data Analysis Platforms. In SIGKDD (London, United Kingdom) (KDD '18). Association for Computing Machinery, New York, NY, USA, 576–585. https: //doi.org/10.1145/3219819.3219848
- [23] George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. 1978. An analysis of approximations for maximizing submodular set functions - I. *Math. Program.* 14, 1 (1978).
- [24] Edward Omiecinski. 2003. Alternative Interest Measures for Mining Associations in Databases. *IEEE TKDE* 15, 1 (2003).
- [25] Y. Park, M. Cafarella, and B. Mozafari. 2016. Visualization-aware sampling for very large databases. In ICDE.
- [26] Aleksandrs Slivkins. 2019. Introduction to Multi-Armed Bandits. CoRR abs/1904.07272 (2019). arXiv:1904.07272 http://arxiv.org/abs/1904.07272
- [27] Ramakrishnan Srikant and Rakesh Agrawal. 1996. Mining Quantitative Association Rules in Large Relational Tables. In SIGMOD.
- [28] R. Srikant and R. Agrawal. 1997. Mining generalized association rules. Future Generation Computer Systems 13, 2 (1997).
- [29] Nan Tang, Ju Fan, Fangyi Li, Jianhong Tu, Xiaoyong Du, Guoliang Li, Sam Madden, and Mourad Ouzzani. 2020. RPT: relational pre-trained transformer is almost all you need towards democratizing data preparation. arXiv preprint arXiv:2012.02469 (2020).
- [30] Marcos R Vieira, Humberto L Razente, Maria CN Barioni, Marios Hadjieleftheriou, Divesh Srivastava, Caetano Traina, and Vassilis J Tsotras. 2011. On query result diversification. In 2011 IEEE 27th International Conference on Data Engineering. IEEE, 1163–1174.
- [31] Wes McKinney. 2010. Data Structures for Statistical Computing in Python. In Proceedings of the 9th Python in Science Conference, Stéfan van der Walt and Jarrod Millman (Eds.). 56 – 61. https://doi.org/10.25080/Majora-92bf1922-00a
- [32] Duch Wlodzisław, Tadeusz Wieczorek, Jacek Biesiada, and Marcin Blachnik. 2004. Comparison of feature ranking methods based on information entropy, Vol. 2. 1415 – 1419 vol.2. https://doi.org/10.1109/IJCNN.2004.1380157
- [33] Ting Wu, Lei Chen, Pan Hui, Chen Jason Zhang, and Weikai Li. 2015. Hear the Whole Story: Towards the Diversity of Opinion in Crowdsourcing Markets. *PVLDB* 8, 5 (2015).
- [34] Kai Zeng, Sameer Agarwal, Ankur Dave, Michael Armbrust, and Ion Stoica. 2015. G-ola: Generalized on-line aggregation for interactive analysis on big data. In Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. 913–918.
- [35] Li Zhang, Shuo Zhang, and Krisztian Balog. 2019. Table2Vec. Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (Jul 2019). https://doi.org/10.1145/3331184.3331333